Learning Recursive Bayesian Multinets for Data Clustering by Means of Constructive Induction

JOSE M. PEÑA ccbpepaj@si.ehu.es
JOSE A. LOZANO ccploalj@si.ehu.es
PEDRO LARRAÑAGA ccplamup@si.ehu.es
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, P.O. Box 649, E-20080 Donostia-San Sebastián, Spain*

This paper introduces and evaluates a new class of knowledge model, the recursive Bayesian multinet (RBMN), which encodes the joint probability distribution of a given database. RBMNs extend Bayesian networks (BNs) as well as partitional clustering systems. Briefly, a RBMN is a decision tree with component BNs at the leaves. A RBMN is learnt using a greedy, heuristic approach akin to that used by many supervised decision tree learners, but where BNs are learnt at leaves using constructive induction. A key idea is to treat expected data as real data. This allows us to complete the database and to take advantage of a closed form for the marginal likelihood of the expected complete data that factorizes into separate marginal likelihoods for each family (a node and its parents). Our approach is evaluated on synthetic and real-world databases.

1. Introduction

One of the main problems that arises in a great variety of fields, including pattern recognition, machine learning and statistics, is the so-called *data clustering problem* (Anderberg, 1973; Banfield & Raftery, 1993; Chandon & Pinson 1980; Duda & Hart, 1973; Fisher, 1987; Hartigan, 1975; Kaufman & Rousseeuw, 1990). Data clustering can be viewed as a data-partitioning problem, where we partition data into different clusters based on a quality or similarity criterion (e.g., as in K-Means (MacQueen, 1967)). Alternatively, data clustering is one way of representing the joint probability distribution of a database. We assume that, in addition to the observed or predictive attributes, there is a *hidden* variable. This unobserved variable reflects the cluster membership for every case in the database. Therefore, the data clustering problem is also an example of learning from *incomplete data* due to the existence of such a hidden variable. Incomplete data represents a special case of *missing data*, where all the missing entries are concentrated in a single (hidden) variable. That is, we refer to a given database as incomplete when the classification is not given. Parameter estimation and model comparison in classical and Bayesian statistics provide a solution to the data

*http://www.sc.ehu.es/isg

clustering problem. The most frequently used approaches include mixture density models (e.g., Gaussian mixture models (Banfield & Raftery, 1993)) and Bayesian networks (e.g., AutoClass (Cheeseman et al., 1988)).

We aim to automatically recover the joint probability distribution from a given incomplete database by learning *recursive Bayesian multinets* (*RBMNs*). Roughly, a recursive Bayesian multinet is a *decision tree* (Breiman et al., 1984; Quinlan, 1993) where each *decision path* (i.e., a conjunction of predictive attribute-value pairs) ends in an alternate component *Bayesian network* (*BN*) (Castillo, Gutiérrez, & Hadi, 1997; Jensen, 1996; Lauritzen, 1996; Pearl, 1988).

RBMNs are a natural extension of BNs. While the conditional (in)dependencies encoded by a BN are *context-non-specific conditional (in)dependencies*, RBMNs allow us to work with *context-specific conditional (in)dependencies* (Geiger & Heckerman, 1996; Thiesson et al., 1998), which differ from decision path to decision path.

Our heuristic approach to the learning of RBMNs requires the learning of its component BNs from incomplete data. In the last few years, several methods for learning BNs have arisen (Buntine, 1994; Cooper & Herskovits, 1992; Heckerman, Geiger, & Chickering, 1995; Pazzani, 1996b; Spiegelhalter et al., 1993), some of them that learn from incomplete data (Cheeseman & Stutz, 1995; Friedman, 1998; Meilă & Heckerman, 1998; Peña, Lozano, & Larrañaga, 1999, 2000a; Thiesson et al., 1998). We describe how the Bayesian heuristic algorithm for the learning of BNs for data clustering developed by Peña, Lozano, and Larrañaga (1999) is extended to learn RBMNs.

A key step in the Bayesian approach to learning graphical models in general and BNs in particular is the computation of the *marginal likelihood* of data given the model. This quantity is the ordinary likelihood of data averaged over the parameters with respect to their prior distribution. When dealing with incomplete data, the exact calculation of the marginal likelihood is typically intractable (Cooper & Herskovits, 1992), thus, such a computation has to be approximated (Chickering & Heckerman, 1997). The existing methods are rather inefficient for our purpose of eliciting a RBMN from an incomplete database, since they do not factorize into scores for families (i.e., nodes and their parents). Hence, we would have to recompute the score for the whole structure from anew, although only the factors of some families had changed.

To avoid this problem, we use the algorithm developed in Peña, Lozano, and Larrañaga (1999) based upon the work done in Thiesson et al. (1998). We search for parameter values for the initial structure by means of the *EM algorithm* (Dempster, Laird, & Rubin, 1977; McLachlan & Krishnan, 1997), or by means of the *BC+EM method* (Peña, Lozano, & Larrañaga, 2000a). This allows us to complete the database by using the current model, that is, by treating expected data as real data, which results in the possibility of using a score criterion that is both in closed form and factorable.

The remainder of this paper is organized as follows. In Section 2, we describe BNs, *Bayesian multinets* (*BMNs*) and RBMNs for data clustering. Section 3 is dedicated to the heuristic algorithm for the learning of component BNs from incomplete data. In Section 4, we describe the algorithm for the learning of RBMNs for data clustering. Finally, in Section 5 we present some experimental results. The paper finishes with Section 6 where we draw some conclusions and outline some lines of further research.

2. BNs, BMNs and RBMNs for data clustering

2.1. Notation

We follow the usual convention of denoting variables by upper-case letters and their states by the same letters in lower-case. We use a letter or letters in bold-face upper-case to designate a set of variables and the same bold-face lower-case letter or letters to denote an assignment of state to each variable in a given set. $|\mathbf{X}|$ is used to refer to the number of states of the variable $\mathbf{X}$. We use $p(\mathbf{x} \mid \mathbf{y})$ to denote the probability that $\mathbf{X} = \mathbf{x}$ given $\mathbf{Y} = \mathbf{y}$. We also use $p(\mathbf{x} \mid \mathbf{y})$ to denote the conditional probability distribution (mass function, as we restrict our discussion to the case where all the variables are discrete) for $\mathbf{X}$ given $\mathbf{Y} = \mathbf{y}$. Whether $p(\mathbf{x} \mid \mathbf{y})$ refers to a probability or a conditional probability distribution should be clear from the context.

As we mentioned, when facing a data clustering problem we assume the existence of the n-dimensional random variable $\mathbf{X}$ partitioned as $\mathbf{X} = (\mathbf{Y}, C)$ into an $(n-1)$-dimensional random variable $\mathbf{Y}$ (predictive attributes), and a unidimensional hidden variable C (cluster variable).

2.2. BNs for data clustering

Given an n-dimensional variable $\mathbf{X} = (X_1, \ldots, X_n) = (\mathbf{Y}, C)$, a BN (Castillo, Gutiérrez, & Hadi, 1997; Jensen, 1996; Lauritzen, 1996; Pearl, 1988) for $\mathbf{X}$ is a graphical factorization of the joint probability distribution for $\mathbf{X}$. A BN is defined by a directed acyclic graph $\mathbf{b}$ (model structure) determining the conditional (in)dependencies among the variables of $\mathbf{X}$ and a set of local probability distributions. When $\mathbf{b}$ contains an arc from the variable X_j to the variable X_i, X_j is referred to as a *parent* of X_i. We denote by $\mathbf{Pa}(\mathbf{b})_i$ the set of all the parents that the variable X_i has in $\mathbf{b}$. The structure lends itself to a factorization of the joint probability distribution for $\mathbf{X}$ as follows:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa}(\mathbf{b})_i) \tag{1}$$

where $\mathbf{pa}(\mathbf{b})_i$ denotes the state of the parents of X_i, $\mathbf{Pa}(\mathbf{b})_i$, consistent with $\mathbf{x}$. The local probability distributions of the BN are those in Eq. (1) and we assume that they depend on a finite set of parameters $\boldsymbol{\theta}_\mathbf{b} \in \boldsymbol{\Theta}_\mathbf{b}$. Therefore, Eq. (1) can be rewritten as follows:

$$p(\mathbf{x} \mid \boldsymbol{\theta}_\mathbf{b}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa}(\mathbf{b})_i, \boldsymbol{\theta}_\mathbf{b}). \tag{2}$$

If $\mathbf{b}^h$ denotes the hypothesis that the conditional (in)dependence assertions implied by $\mathbf{b}$ hold in the true joint probability distribution for $\mathbf{X}$, then we obtain from Eq. (2) that:

$$p(\mathbf{x} \mid \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa}(\mathbf{b})_i, \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h). \tag{3}$$

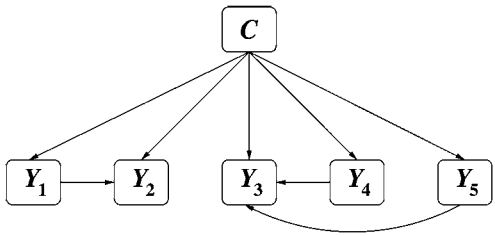


Figure 1. Example of the structure of a BN for data clustering for $\mathbf{X} = (\mathbf{Y}, C) = (Y_1, Y_2, Y_3, Y_4, Y_5, C)$. It follows from the figure that the joint probability distribution factorizes as $p(\mathbf{x}) = p(\mathbf{y}, c) = p(c) \cdot p(y_1 \mid c) \cdot p(y_2 \mid y_1, c) \cdot p(y_3 \mid y_4, y_5, c) \cdot p(y_4 \mid c) \cdot p(y_5 \mid c)$.

According to the partition of $\mathbf{X}$ as $\mathbf{X} = (\mathbf{Y}, C)$, Eq. (3) can be rewritten as follows:

$$\begin{aligned} p(\mathbf{x} \mid \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) &= p(c \mid \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) p(\mathbf{y} \mid c, \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) \\ &= p(c \mid \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) \prod_{i=1}^{n-1} p(y_i \mid \mathbf{prepa}(\mathbf{b})_i, c, \boldsymbol{\theta}_\mathbf{b}, \mathbf{b}^h) \end{aligned} \tag{4}$$

where $\mathbf{prepa}(\mathbf{b})_i$ denotes the state of those parents of Y_i that correspond to predictive attributes, $\mathbf{PrePa}(\mathbf{b})_i$, consistent with $\mathbf{y}$.

Thus, a BN is completely defined by a pair $(\mathbf{b}, \boldsymbol{\theta}_\mathbf{b})$. The first of the two components is the model structure, and the second component is the set of parameters for the local probability distributions corresponding to $\mathbf{b}$. See figure 1 for an example of a BN structure for data clustering with five predictive attributes.

2.3. BMNs for data clustering

The conditional (in)dependencies determined by the structure of a BN are called context-non-specific conditional (in)dependencies (Thiesson et al., 1998), also known as *symmetric conditional (in)dependencies* (Geiger & Heckerman, 1996). That is, if the structure implies that two sets of variables are independent given some configuration (or state) of a third set of variables, then the two first sets are also independent given every other configuration of this third set of variables. A BMN (Geiger & Heckerman, 1996) is a generalization of the BN model that is able to encode context-specific conditional (in)dependencies (Thiesson et al., 1998), also known as *asymmetric conditional (in)dependencies* (Geiger & Heckerman, 1996). Therefore, a BMN structure may imply that two sets of variables are independent given some configuration of a third set, and dependent given another configuration of this third set. Formally, a BMN for $\mathbf{X} = (X_1, \ldots, X_n) = (\mathbf{Y}, C)$ and distinguished variable $G \in \mathbf{Y}$ is a graphical factorization of the joint probability distribution for $\mathbf{X}$. A BMN is defined by a probability distribution for G and a set of component BNs for $\mathbf{X} \backslash \{G\}$, each of which encodes the joint probability distribution for $\mathbf{X} \backslash \{G\}$ given a state of G. Because the structure of each component BN may vary, a BMN can encode context-specific conditional (in)dependence assertions. In this paper, we limit the distinguished variable G to be one of the original predictive attributes. However Thiesson et al. (1998) allows the distinguished

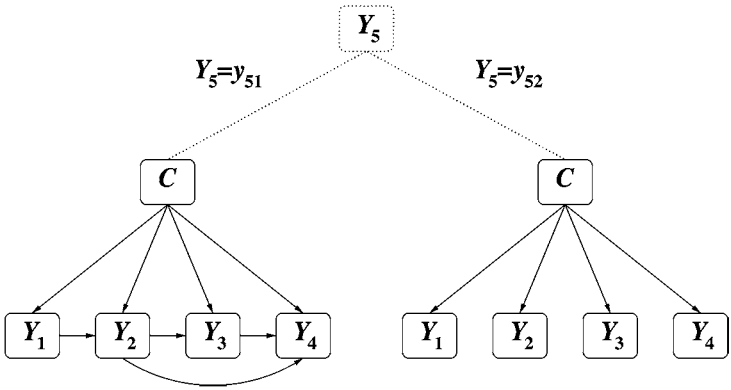


Figure 2. Example of the structure of a BMN for data clustering for $\mathbf{X} = (\mathbf{Y}, C) = (Y_1, Y_2, Y_3, Y_4, Y_5, C)$ and distinguished variable $G = Y_5$. There are two component BNs as the distinguished variable is dichotomic ($|Y_5| = 2$). Dotted lines correspond to the distinguished variable Y_5.

variable to be either one of the predictive attributes or the hidden cluster variable C. When this last happens, each leaf represents a single cluster. These models are called *mixtures of BNs* according to Thiesson et al. (1998). Figure 2 shows the structure of a BMN for data clustering when the distinguished variable G has two values.

Let $\mathbf{s}$ and $\boldsymbol{\theta}_\mathbf{s}$ denote the structure and parameters of a BMN for $\mathbf{X}$ and distinguished variable G. In addition, let us suppose that $\mathbf{b}_g$ and $\boldsymbol{\theta}_g$ denote the structure and parameters of the g-th component BN of the BMN. Also, let $\mathbf{s}^h$ denote the hypothesis that the context-specific conditional (in)dependencies implied by $\mathbf{s}$ hold in the true joint probability distribution for $\mathbf{X}$ and distinguished variable G. Therefore, the joint probability distribution for $\mathbf{X}$ encoded by the BMN is given by:

$$p(\mathbf{x} \mid \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h) = p(g \mid \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h)\ p(\mathbf{x}\backslash\{g\} \mid g, \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h) = \pi_g p\left(\mathbf{x}\backslash\{g\} \,\middle|\, \boldsymbol{\theta}_g, \mathbf{b}_g^h\right) \tag{5}$$

where $\boldsymbol{\theta}_\mathbf{s} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|G|}, \pi_1, \ldots, \pi_{|G|})$ denotes the parameters of the BMN, $\mathbf{b}_g^h$ is a shorthand for the conjunction of $\mathbf{s}^h$ and $G = g$, and $\pi_g = p(g \mid \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h)$. The last term of the previous equation can be further factorized according to the structure of the g-th component BN of the BMN (Eq. (4)).

Thus, a BMN is completely defined by a pair $(\mathbf{s}, \boldsymbol{\theta}_\mathbf{s})$. The first of the two components is the structure of the BMN, and the second component is the set of parameters.

We may see a BMN as a depth-one decision tree (Breiman et al., 1984; Quinlan, 1993), where the distinguished variable is the root and there is a branch for each of its states. At the end of each of these branches is a leaf which is a component BN. Thus, it is helpful to see the dotted lines of figure 2 as conforming a decision tree with component BNs as leaves.

2.4. RBMNs for data clustering

Let us follow with the view of a BMN as a depth-one decision tree where leaves are component BNs. We propose to use deeper decision trees where leaves are still component

BNs. By definition, every component of a BMN is limited to be a BN. A RBMN allows every component to be either a BN (at a leaf) or recursively a RBMN.

RBMNs extend BNs and BMNs, but RBMNs also extend partitional clustering systems (Fisher & Hapanyengwi, 1993). RBMNs can be considered as extensions of BNs because, like BMNs and mixtures of BNs, RBMNs allow us to encode context-specific conditional (in)dependencies. Thus, they constitute a more flexible tool than BNs and provide the user with structured and specialized domain knowledge as alternative component BNs are learnt for every decision path. Moreover, RBMNs generalize the idea behind BMNs by offering the possibility of having decision paths with conjunctions of as many predictive attribute-value pairs as we want. The only constraint is that these decision paths must be represented by a decision tree.

Additionally, RBMNs extend traditional partitional clustering systems. A previous work with the same aim is where Fisher and Hapanyengwi (1993) propose to perform data clustering based upon a decision tree. The measure used to select the divisive attribute at each node during the decision tree construction consists of the computation of the sum of information gains over all attributes, while in the supervised paradigm the measure is limited to the information gain over a single specified class attribute. This is a natural generalization of the works on supervised learning where the performance task comprises the prediction of only one attribute from the knowledge of many, whereas the generic performance task in unsupervised learning is the prediction of many attributes from the knowledge of many. Thus, RBMNs and the work by Fisher and Hapanyengwi aim to learn a decision tree with knowledge at leaves sufficient for making inference along many attributes. This implies that both paradigms are considered extensions of traditional partitional clustering systems as they are concerned with characterizing clusters of observations rather than partitioning them.

We define a RBMN according to the intuitive idea of a decision tree with component BNs as leaves. Let $\mathbf{T}$ be a decision tree, here referred to as *distinguished decision tree*, where (i) every internal node in $\mathbf{T}$ represents a variable of $\mathbf{Y}$, (ii) every internal node has as many children or branches coming out from it as states for the variable represented by the node, (iii) all the leaves are at the same level, and (iv) if $\mathbf{T}(root, l)$ is the set of variables that are in the decision path between the root and the leaf l of the distinguished decision tree, there are then no repeated variables in $\mathbf{T}(root, l)$. Condition (iii) is imposed to simplify the understanding of RBMNs and their learning, but such a constraint can be removed in practice. Let us define $\mathbf{X}\backslash\mathbf{T}(root, l)$ as the set of all the variables in $\mathbf{X}$ except those that are in the decision path between the root and the leaf l of the distinguished decision tree $\mathbf{T}$. Thus, a RBMN for $\mathbf{X} = (X_1, \ldots, X_n) = (\mathbf{Y}, C)$ and distinguished decision tree $\mathbf{T}$ is a graphical factorization of the joint probability distribution for $\mathbf{X}$. A RBMN is defined by a probability distribution for the leaves of $\mathbf{T}$ and a set of component BNs, each of which encodes the joint probability distribution for $\mathbf{X}\backslash\mathbf{T}(root,\ l)$ given the l-th leaf of $\mathbf{T}$. Thus, the component BN at every leaf of the distinguished decision tree does not consider attributes involved in the tests on the decision path leading to the leaf.

Obviously, BMNs are a special case of RBMNs in which $\mathbf{T}$ is a distinguished decision tree with only one internal node, the distinguished variable. Moreover, we could assume that BNs are also a special case of RBMNs in which the distinguished decision tree contains no internal nodes.

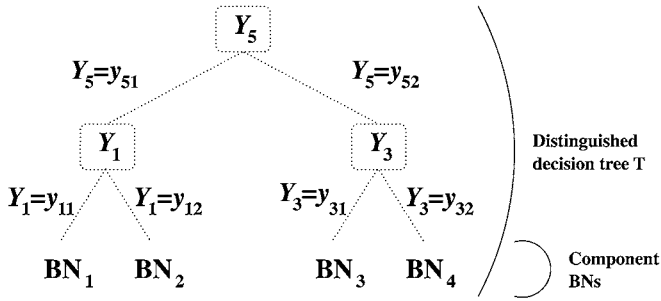


Figure 3. Example of the structure of a 2-levels RBMN for data clustering for $\mathbf{X} = (\mathbf{Y}, C) = (Y_1, Y_2, Y_3, Y_4, Y_5, C)$ and distinguished decision tree $\mathbf{T}$. This RBMN has two component BMNs, each of them with two component BNs (assuming that the variables in the distinguished decision tree are all dichotomic). Dotted lines correspond to the distinguished decision tree $\mathbf{T}$.

Figure 3 helps us to illustrate the structure of a RBMN for data clustering as a decision tree where each internal node is a predictive attribute and branches from that node are states of the variable. Every leaf l is a component BN that does not consider attributes in $\mathbf{T}(root, l)$. So, the induction of the component BNs is simplified. Since every internal node of $\mathbf{T}$ is a predictive attribute, the hidden variable C appears in every component BN. This fact implies that the component BN at each leaf of $\mathbf{T}$ does not represent only one cluster as Fisher and Hapanyengwi (1993) propose but a context-specific data clustering. That is, the data clustering encoded by each component BN is totally unrelated to the data clusterings encoded by the rest. This means that the probabilistic clusters identified by each component BN are not in correspondence with those identified by the rest of component BNs. This is due to the fact that C acts as a context-specific or local hidden cluster variable for every component BN. To be exact, every variable of each component BN is a context-specific variable that does not interact with the variables of any other component BN since the elicitation of every component BN is totally independent of the rest. This is not explicitly reflected in the notation as every branch identifies unambiguously each component BN and its variables. Additionally, this avoids a too complex notation. This reasoning should also be applied to BMNs as they are a special case of RBMNs.

Let $\mathbf{s}$ and $\boldsymbol{\theta}_\mathbf{s}$ denote the structure and parameters of a RBMN for $\mathbf{X}$ and distinguished decision tree $\mathbf{T}$. In addition, let us suppose that $\mathbf{b}_l$ and $\boldsymbol{\theta}_l$ denote the structure and parameters of the l-th component BN of the RBMN. Also, let $\mathbf{s}^h$ denote the hypothesis that the context-specific conditional (in)dependencies implied by $\mathbf{s}$ hold in the true joint probability distribution for $\mathbf{X}$ and distinguished decision tree $\mathbf{T}$. Therefore, the joint probability distribution for $\mathbf{X}$ encoded by the RBMN is given by:

$$\begin{aligned} p(\mathbf{x} \mid \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h) &= p(\mathbf{t}(root, l) \mid \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h)\ p(\mathbf{x} \backslash \mathbf{t}(root, l) \mid \mathbf{t}(root, l), \boldsymbol{\theta}_\mathbf{s}, \mathbf{s}^h) \\ &= \pi_l p\left(\mathbf{x} \backslash \mathbf{t}(root, l) \,\middle|\, \boldsymbol{\theta}_l, \mathbf{b}_l^h\right) \end{aligned} \tag{6}$$

where the leaf l is the only one that makes $\mathbf{x}$ be consistent with $\mathbf{t}(root, l)$, $\boldsymbol{\theta}_\mathbf{s} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L, \pi_1, \ldots, \pi_L)$ denotes the parameters of the RBMN, L is the number of leaves in $\mathbf{T}$, $\mathbf{b}_l^h$ is

a shorthand for the conjunction of $\mathbf{s}^h$ and $\mathbf{T}(root, l) = \mathbf{t}(rool, l)$, and $\pi_l = p(\mathbf{t}(root, l) \mid \boldsymbol{\theta}_{\mathbf{s}}, \mathbf{s}^h)$. The last term of the previous equation can be further factorized according to the structure of the l-th component BN of the RBMN (Eq. (4)).

Thus, a RBMN is completely defined by a pair $(\mathbf{s}, \boldsymbol{\theta}_{\mathbf{s}})$. The first of the two components is the structure of the RBMN, and the second component is the set of parameters.

In this paper, we limit our discussion to the case in which the component BNs are defined by multinomial distributions. That is, all the variables are finite discrete variables and the local distributions at each variable in the component BNs consist of a set of multinomial distributions, one for each configuration of the parents. In addition, we assume that the proportions (probabilities) of data covered by the leaves of $\mathbf{T}$ follow also a multinomial distribution.

As stated, RBMNs extend BNs due to their ability to encode context-specific conditional (in)dependencies which increases the expressive power of RBMNs over BNs. A decision tree effectively identifies subsets of the original database where different component BNs result a better, more flexible way fit to data.

Other works in supervised induction identify instance subspaces through local or component models. Kohavi (1996) links Naive Bayes (NB) classifiers and decision tree learning. On the other hand, the work done by Zheng and Webb (2000) combines the previous work by Kohavi with a lazy learning algorithm to build Bayesian rules where the antecedent is a conjunction of predictive attribute-value pairs, and the consequent is a NB classifier. Thus, both works share the fact that they use conjunctions of predictive attribute-value pairs to define instance subspaces described by NB classifiers. Zheng and Webb (2000) give an extensive experimental comparison between these two and other approaches for supervised learning in some well-known domains.

Langley (1993) proposes to identify instance subspaces where the independence assumptions made by the NB classifier hold. His work is based upon the recursive split of the original database by using decision trees where nodes are NB classifiers and leaves are sets of cases belonging to only one class.

To illustrate how RBMNs structure a clustering for a given database, we use a real-world domain where data clustering was successfully performed by means of probabilistic graphical models (Peña et al., 2001), with the aim of improving knowledge on the geographical distribution of malignant tumors. A geographical clustering of the towns of the Autonomous Community of the Basque Country (north of Spain) was performed. Every town was described by the age-standardized cancer incidence rates of the six most frequent cancer types for patients of each sex between 1986 and 1994. The authors obtained a geographical clustering for male patients and a geographical clustering for female patients as the differences in the geographical patterns of malignant tumors for patients of each sex are well-known by the experts. Each of both clusterings was achieved by means of the learning of a BN. The final clusterings were presented by using coloured maps to partition the towns in such a way that each town was assigned to the most probable cluster according to the learnt BN, i.e., each town was assigned to the cluster with the highest posterior probability.

Due to the different geographical patterns for male and female patients, it seems quite reasonable to assume that a RBMN would be an effective and automatic tool to face the referred real-world problem without relying on human expertise. That is, the learning of a

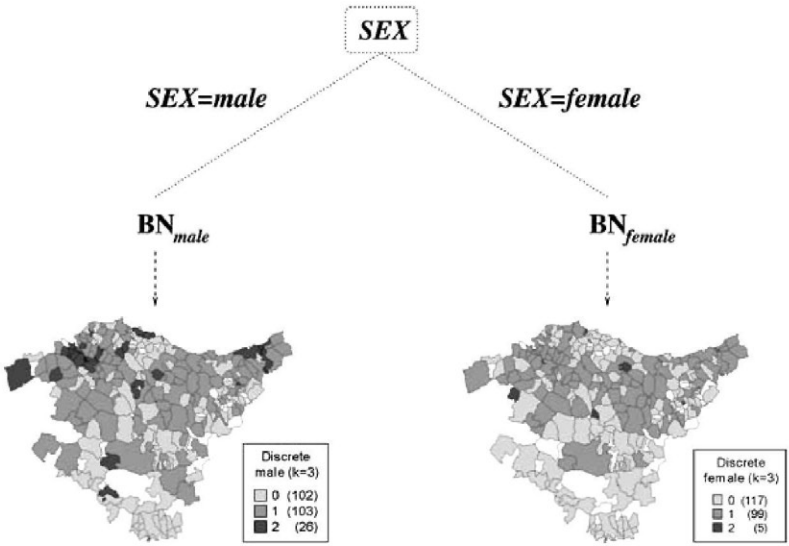


Figure 4. Scheme of the structure of the RBMN that, ideally, would be learnt for the real-world domain described in Peña et al. (2001). Additionally, the clusterings encoded by the component BNs are shown as coloured maps. White towns were excluded from the study.

RBMN would be able to automatically identify that the instance subspace for male patients encodes an underlying model different from the one encoded by the instance subspace for female patients. However, the authors relied on human expertise to divide the original database and treat separately male and female cases.

Figure 4 shows a RBMN that, ideally, would be learnt, and the structured clustering obtained from this model. It is easy to see that the clusterings obtained for male and female patients are different as well as context-specific. Furthermore, Peña et al. (2001) reports that the characterization of each cluster was completely different for male and female patients. These differences in the geographical patterns can not be captured when learning BNs from the original joint database. In this example, figure 4 is also a BMN since the distinguished decision tree contains only one predictive attribute. However, it is easy to see that a RBMN might represent a more complex decision tree that represented a more specialized clustering. For example, we might expect different component BNs for each of the four conjunctions $(SEX = male,\ AGE = child)$, $(SEX = male,\ AGE = adult)$, $(SEX = female,\ AGE = child)$, and $(SEX = female,\ AGE = adult)$. This example could be encoded by a RBMN with a 2-levels distinguished decision tree.

A key idea in our approach to the learning of a RBMN for data clustering is to decompose the problem into learning its component BNs from incomplete data. The component BN corresponding to each leaf l is learnt from an incomplete database that is a subset of the original incomplete database. This subset contains all the cases of the original database that are consistent with $\mathbf{t}(root, l)$. Therefore, there still exists a hidden variable when learning every component BN. That is why the problem of learning a RBMN for data clustering is largely a problem of learning component BNs from incomplete data. Thus, in the following

section, we present a heuristic algorithm for the learning of a BN from an incomplete database.

3. Learning BNs from incomplete data through constructive induction

In this section, we describe a heuristic algorithm to elicit the component BNs from incomplete data. We use this heuristic algorithm as part of the algorithm for the learning of RBMNs for data clustering that we present in the following section.

3.1. Component BN structure

Due to the difficulty involved in learning densely connected BNs and the painfully slow probabilistic inference when working with them, it is desirable to develop methods for learning the simplest BNs that fit the data adequately. Some examples of this trade-off between the cost of the learning process and the quality of the learnt models are NB models (Duda & Hart, 1973; Peot, 1996), *Extended Naive Bayes (ENB) models* (Pazzani, 1996a, 1996b; Peña, Lozano, & Larrañaga, 1999, 2001) and *Tree Augmented Naive Bayes models* (Friedman, Geiger, & Goldszmidt, 1997; Friedman & Goldszmidt, 1996; Friedman, Goldszmidt, & Lee, 1998; Keogh & Pazzani, 1999; Meilă, 1999; Peña, Lozano, & Larrañaga, 2000a). Despite the wide recognition that these models are a weaker representation of some domains than more general BNs, the expressive power of these models is often acceptable. Moreover, these models appeal to human intuition and can be learnt relatively quickly.

For the sake of brevity, the class of compromise BNs that we propose to learn as component BNs will be referred to as ENB (Peña, Lozano, & Larrañaga, 2001). ENB models were introduced by Pazzani (1996a, 1996b) as Bayesian classifiers and later used by Peña, Lozano, and Larrañaga (1999, 2001) for data clustering. ENB models can be considered as having an intermediate place between NB models and models with all the predictive attributes fully correlated (see figure 5). Thus, they keep the main features of both extremes: simplicity from NB models and a better performance from fully correlated models.

ENB models are very similar to NB models since all the attributes are independent given the cluster variable. The only difference with NB models is that the number of nodes in the structure of an ENB model can be shorter than the original number of attributes in the database. The reasons are that (i) a selection of the attributes to be included in the models can

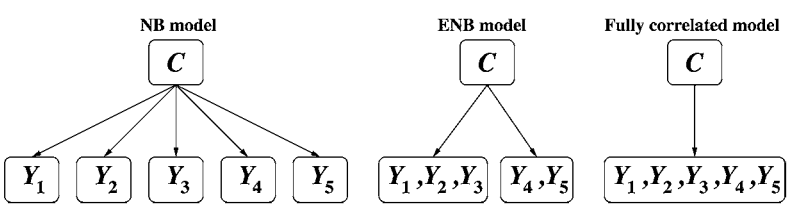


Figure 5. Component BN (ENB model) structure that we propose to learn seen as having a place between NB models and fully correlated models, when applied to the data clustering problem.

be performed, and (ii) some attributes can be grouped together under the same node as fully correlated attributes (we refer to such nodes as *supernodes*[1]). Therefore, the class of ENB models ensures a better performance than NB models while it maintains their simplicity. As we consider all the attributes relevant for the data clustering task, we do not perform attribute selection as proposed by Pazzani.

The structure of an ENB model for data clustering lends itself to a factorization of the joint probability distribution for $\mathbf{X}$ as follows:

$$p(\mathbf{x} \mid \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h) = p(\mathbf{y}, c \mid \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h) = p(c \mid \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h) \prod_{i=1}^{r} p(\mathbf{z}_i \mid c, \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h) \tag{7}$$

where $\{\mathbf{z}_1, \ldots, \mathbf{z}_r\}$ is a partition of $\mathbf{y}$, where r is the number of nodes (including the special nodes referred to as supernodes). Each $\mathbf{z}_i$ is the set of values in $\mathbf{y}$ for the original predictive attributes that are grouped together under a supernode $\mathbf{Z}_i$, or it is the value in $\mathbf{y}$ for a predictive attribute $\mathbf{Z}_i$.

3.2. *Algorithm for learning ENB models from incomplete data*

The log marginal likelihood is often used as the Bayesian criterion to guide the search for the best model structure. An important feature of the log marginal likelihood is that, under some reasonable assumptions, it factorizes into scores for families. When a criterion is factorable, search is more efficient since we need not reevaluate the criterion for the whole structure when only the factors of some families have changed. This is an important feature when working with some heuristic search algorithms, because they iteratively transform the model structure by choosing the transformation that improves the score the most and, usually, this transformation does not affect all the families.

When the variable that we want to classify is hidden the exact calculation of the log marginal likelihood is typically intractable (Cooper & Herskovits, 1992) thus, we have to approximate such a computation (Chickering & Heckerman, 1997). However, the existing methods for doing this are rather inefficient for eliciting the component BNs (ENB models) from incomplete databases as they do not factorize into scores for families.

To avoid this problem, we use the heuristic algorithm presented in Peña, Lozano, and Larrañaga (1999), which is shown in figure 6. First, the algorithm chooses an initial structure and parameter values. Then, it performs a parameter search step to improve the set of parameters for the current model structure. These parameter values are used to complete the database, because the key idea in this approach is to treat expected data as real data (hidden variable completion by means of probabilistic inference with the current model). Hence, the log marginal likelihood of the expected complete data, $\log p(\mathbf{d} \mid \mathbf{b}^h)$, can be calculated by Cooper and Herskovits (1992) in closed form. Furthermore, the factorability of the log marginal likelihood into scores for families allows the performance of an efficient structure search step. After structure search, the algorithm reestimates the parameters for the new structure that it finds to be the maximum likelihood parameters given the complete database. Finally, the probabilistic inference process to complete the database and the structure search are iterated until there is no change in the structure. Figure 6 shows the possibility of

```
1.Choose initial structure and initial set of parameter values
  for the initial structure
2.Parameter search step
3.Probabilistic inference to complete the database
4.Calculate sufficient statistics to compute the log p(d | b^h)
5.Structure search step
6.Reestimate parameter values for the new structure
7.IF no change in the structure has been done
  THEN stop
  ELSE IF interleaving parameter search step
       THEN go to 2
       ELSE go to 3
```

Figure 6. A schematic of the algorithm for the learning of component BNs (ENB models) from incomplete data.

interleaving the parameter search step or not after each structural change, though we will not interleave parameter and structure search in the experiments to follow for reasons of cost.

Another key point is that a penalty term is built into the log marginal likelihood to guard against overly complex models. In Meilă and Heckerman (1998) a similar use of this built-in penalty term can be found.

In the remainder of this section, we describe the parameter search step and the structure search step in more detail.

3.2.1. Parameter search. As seen in figure 6, the heuristic algorithm that we use considers the possibility of interleaving parameter and structure search steps. Concretely, this interleaving process is done, at least, in the first iteration of the algorithm. By doing that, we ensure a good set of initial parameter values. For the remaining iterations we can then decide whether to interleave parameter and structure search steps or not. Although any parameter search procedure can be considered to perform the parameter search step, currently, we propose two alternative techniques: the well-known EM algorithm (Dempster, Laird, & Rubin, 1977; McLachlan & Krishnan, 1997), and the BC+EM method (Peña, Lozano, & Larrañaga, 2000a).

According to Peña, Lozano, and Larrañaga (2000a), the BC+EM method exhibits a faster convergence rate, and more effective and robust behaviour than the EM algorithm. That is why the BC+EM method is used in our experimental evaluation of RBMNs. Basically, the BC+EM method alternates between the *Bound and Collapse (BC) method* (Ramoni & Sebastiani, 1997, 1998) and the EM algorithm.

The BC method is a deterministic method to estimate conditional probabilities from databases with missing entries. It bounds the set of possible estimates consistent with the available information by computing the minimum and the maximum estimate that would be obtained from all possible completions of the database. These bounds are then collapsed into a unique value via a convex combination of the extreme points with weights depending on the assumed pattern of missing data. This method presents all the advantages of a deterministic method and a dramatic gain in efficiency when compared with the EM algorithm (Ramoni & Sebastiani, 1999).

```
1.FOR every case y in the database DO
  a.Calculate the posterior probability distribution p(c | y, θ_b, b^h)
  b.Let p_max be the maximum of p(c | y, θ_b, b^h) which is reached
    for C = c_max
  c.IF p_max > fixing_probability_threshold
    THEN assign the case y to the cluster c_max
2.Run the BC method
  a.Bound
  b.Collapse
3.Set the parameter values for the current BN to be the BC's
  output parameter values
4.Run the EM algorithm until convergence
5.IF BC+EM convergence
  THEN stop
  ELSE go to 1
```

Figure 7. A schematic of the BC+EM method.

The BC method is used in presence of missing data, but it is not useful when there is a hidden variable as in the data clustering problem. The reason for this is that the probability intervals returned by the BC method would be too large and poorly inform the missing entries of the single hidden variable. The BC+EM method overcomes this problem by performing a partial completion of the database at each step. See figure 7 for a schematic of the BC+EM method.

For every case $\mathbf{y}$ in the database, the BC+EM method uses the current parameter values to evaluate the posterior probability distribution for the cluster variable C given $\mathbf{y}$. Then, it assigns the case $\mathbf{y}$ to the cluster with the highest posterior probability only if this posterior probability is greater than a threshold, *fixing_probability_threshold*, that the user must determine. The case remains incomplete if there is no cluster with posterior probability greater than the threshold. As some of the entries of the hidden variable have been completed during this process, we hope to have more informative probability intervals when running the BC method. The EM algorithm is then executed to improve the parameter values that the BC method has returned. The process is repeated until convergence.

3.2.2. Structure search. Chickering (1996) shows that finding the BN structure with the highest log marginal likelihood from the set of all the BN structures in which each node has no more than k parents is NP-hard for $k > 1$. Therefore, it is clear that heuristic methods are needed. Our particular choice is based upon the work done by Pazzani (1996a, 1996b). Pazzani presents algorithms for learning augmented NB classifiers (ENB models) by searching for dependencies among attributes: the *Backward Sequential Elimination and Joining (BSEJ) algorithm* and the *Forward Sequential Selection and Joining (FSSJ) algorithm*. To find attribute dependencies, these algorithms perform *constructive induction* (Arciszewski, Michalski, & Wnek, 1995; Michalski, 1978), which is the process of changing the representation of the cases in the database by creating new attributes (supernodes) from existing attributes. As a result, some violation of conditional independence assumptions made by NB models are detected and dependencies among predictive attributes are

```
1.Consider joining each pair of attributes
2.IF there is an improvement in the log p(d | b^h)
  THEN make the joint that improves the score the most
  ELSE return the current case representation
```

Figure 8. A template for the forward structure search step.

```
1.Consider splitting each attribute at each possible point
2.IF there is an improvement in the log p(d | b^h)
  THEN make the split that improves the score the most
  ELSE return the current case representation
```

Figure 9. A template for the backward structure search step.

included in the model. Ideally, a better performance is reached while the model that we obtain after the constructive induction process maintains the simplicity of NB models. Pazzani uses the term *joining* to refer to the process of creating a new attribute whose values are the Cartesian product of two other attributes. To carry out this change in the representation of the database, Pazzani proposes a hill-climbing search combined with two operators: replacing two existing attributes with a new attribute that is the Cartesian product of the two attributes, and either delete an irrelevant attribute (resulting in the BSEJ) or add a relevant attribute (resulting in the FSSJ).

The algorithm for the learning of component BNs that we use (figure 6) starts from one of two possible initial structures: from a NB model or from a model with all the variables fully correlated. When considering a NB model as the initial structure, the heuristic algorithm performs a forward search step (see figure 8). On the other hand, when starting from a fully correlated model, the heuristic algorithm performs a backward search step (see figure 9).

Notice should be taken that the algorithm of figure 6 has completed the database before the structure search step is performed. Consequently, the log marginal likelihood of the expected complete data has a factorable closed form. The algorithm uses the factorability of the log marginal likelihood to score every possible change in the model structure efficiently.

4. Learning RBMNs for data clustering

In this section, we present our heuristic algorithm for the learning of RBMNs for data clustering. This algorithm performs model selection using the log marginal likelihood of the expected complete data to guide the search. This section starts deriving a factorable closed form for the marginal likelihood of data for RBMNs.

4.1. Marginal likelihood criterion for RBMNs

Under the assumptions that (i) the variables in the database are discrete, (ii) cases occur independently, (iii) the database is complete, and (iv) the prior distribution for the parameters given a structure is uniform, the marginal likelihood of data has a closed form for BNs that

allows us to compute it efficiently. In particular:

$$p(\mathbf{d} \mid \mathbf{b}^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{8}$$

where n is the number of variables, r_i is the number of states of the variable X_i, q_i is the number of states of the parent set of X_i, N_{ijk} is the number of cases in the database where X_i has its k-th value and the parent set of X_i has its j-th value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ (see (Cooper & Herskovits, 1992) for a derivation).

This important result is extended to BMNs in Thiesson et al. (1998) as follows: let $\mathbf{\Theta}_{ig}$ denote the set of parameter variables associated with the local probability distribution of the i-th variable belonging to $\mathbf{X}\backslash\{G\}$ in the g-th component BN. Also, let $\mathbf{\Pi}$ denote the set of parameter variables $(\pi_1, \ldots, \pi_{|G|})$ corresponding to the weights of the mixture of component BNs. If (i) the parameters variables $\mathbf{\Pi}$, $\mathbf{\Theta}_{11}, \ldots, \mathbf{\Theta}_{(n-1)1}, \ldots, \mathbf{\Theta}_{1|G|}, \ldots, \mathbf{\Theta}_{(n-1)|G|}$ are mutually independent given $\mathbf{s}^h$ (parameter independence), (ii) the parameter priors $p(\boldsymbol{\theta}_{ig} \mid \mathbf{s}^h)$ are conjugate for all i and g, and (iii) the data $\mathbf{d}$ is complete, then the marginal likelihood of data has a factorable closed form for BMNs. In particular:

$$\log p(\mathbf{d} \mid \mathbf{s}^h) = \log p(\mathbf{d}^G) + \sum_{g=1}^{|G|} \log p\left(\mathbf{d}^{\mathbf{X},g} \mid \mathbf{b}_g^h\right) \tag{9}$$

where $\mathbf{d}^G$ is the data restricted to the distinguished variable G, and $\mathbf{d}^{\mathbf{X},g}$ is the data restricted to the variables $\mathbf{X}\backslash\{G\}$ and those cases in which $G = g$. The term $p(\mathbf{d}^G)$ is the marginal likelihood of a trivial BN having only a single node G. The terms in the sum are log marginal likelihoods for the component BNs of the BMN.

Furthermore, this observation extends to RBMNs as follows: let $\mathbf{\Theta}_{il}$ denote the set of parameter variables associated with the local probability distribution of the i-th variable belonging to $\mathbf{X}\backslash\mathbf{T}(root, l)$ in the l-th component BN. Also, let L denote the number of leaves in $\mathbf{T}$ and m denote the number of levels (depth) of $\mathbf{T}$. Let $\mathbf{\Pi}$ designate the set of parameter variables $(\pi_1, \ldots, \pi_L)$ corresponding to the weights of the mixture of component BNs. If (i) the parameters variables $\mathbf{\Pi}$, $\mathbf{\Theta}_{11}, \ldots, \mathbf{\Theta}_{(n-m)1}, \ldots, \mathbf{\Theta}_{1L}, \ldots, \mathbf{\Theta}_{(n-m)L}$ are mutually independent given $\mathbf{s}^h$ (parameter independence), (ii) the parameter priors $p(\boldsymbol{\theta}_{il} \mid \mathbf{s}^h)$ are conjugate for all i and l, and (iii) the data $\mathbf{d}$ is complete, then the marginal likelihood of data has a factorable closed form for RBMNs. In particular:

$$\log p(\mathbf{d} \mid \mathbf{s}^h) = \sum_{l=1}^{L} \left[\log p\left(\mathbf{d}^{\mathbf{t}(root,l)}\right) + \log p\left(\mathbf{d}^{\mathbf{X},\mathbf{t}(root,l)} \middle| \mathbf{b}_l^h\right) \right] \tag{10}$$

where $\mathbf{d}^{\mathbf{t}(root,l)}$ is the database restricted to the variables in $\mathbf{T}(root, l)$, and those cases consistent with $\mathbf{t}(root, l)$ $\mathbf{d}^{\mathbf{X},\mathbf{t}(root,l)}$ is the database restricted to the variables in $\mathbf{X}\backslash\mathbf{T}(root, l)$ and those cases consistent with $\mathbf{t}(root, l)$. The sum of the first terms can be easily calculated as the log marginal likelihood of a trivial BN with a single node with as many states as leaves in the distinguished decision tree $\mathbf{T}$. The second terms in the sum are log marginal likelihoods for the component BNs of the RBMN. Thus, under the assumptions referred above, there is a factorable closed form to calculate them (Cooper & Herskovits, 1992).

Therefore, the log marginal likelihood of data has a closed form for RBMNs, and it can be calculated from the log marginal likelihoods of the component BNs. This fact allows us to decompose the problem of learning a RBMN into learning its component BNs.

4.2. *Algorithm for learning RBMNs from incomplete data*

The heuristic algorithm that we present in this section performs data clustering by learning, from incomplete data, RBMNs as they were defined in Section 2.4.

The algorithm starts from an empty distinguished decision tree and, at each iteration, it enlarges the tree in one level until a stopping condition is verified. Stopping might occur at some user-specified depth, or when no further improvement in the log marginal likelihood of the expected complete data for the current model (Eq. (10)) is observed. To enlarge the current tree, every leaf (component BN) should be extended. The extension of each leaf l consists of learning the best BMN for $\mathbf{X}\backslash\mathbf{T}(root, l)$ and distinguished variable Y_i, where $Y_i \in \mathbf{Y}\backslash\mathbf{T}(root, l)$. This BMN replaces the leaf l. For learning each component BN of the BMN, we use the algorithm presented in figure 6. Figure 10 shows an example of a 2-levels RBMN structure that could be the output of the algorithm that we present in figure 11.

In this last figure, we can see that the learning algorithm replaces every leaf l by the best BMN for $\mathbf{X}\backslash\mathbf{T}(root, l)$ and distinguished variable Y_i, where $Y_i \in \mathbf{Y}\backslash\mathbf{T}(root, l)$. This is done as follows: let Y_i be a variable of $\mathbf{Y}\backslash\mathbf{T}(root, l)$, for every state y_{ik} of Y_i, $k = 1, \ldots, |Y_i|$, the algorithm learns a component BN, $\mathbf{b}_k$, for the variables in **ext** from an incomplete database $\mathbf{d}^{\mathbf{ext},k}$ (the cluster variable is still hidden), where $\mathbf{ext} = (\mathbf{X}\backslash\mathbf{T}(root, l))\backslash\{Y_i\}$. This learning is carried out by the heuristic algorithm that we have presented in figure 6. The database $\mathbf{d}^{\mathbf{ext},k}$ is a subset of the original database (instance subspace), in fact, it is the original database $\mathbf{d}$ restricted to the variables in **ext**, and those cases consistent with the decision path $\mathbf{t}(root, l)$ and y_{ik}. After this process, we have a candidate BMN with distinguished variable Y_i as a

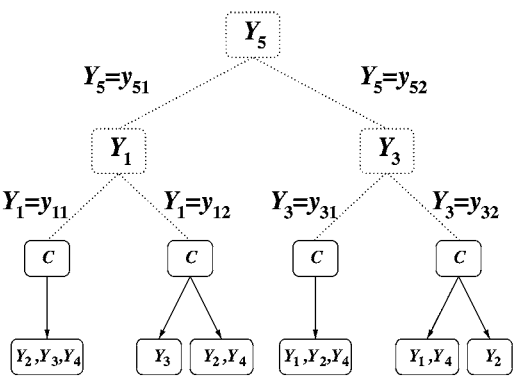


Figure 10. Example of the structure of a 2-levels RBMN for data clustering for $\mathbf{X} = (\mathbf{Y}, C) = (Y_1, Y_2, Y_3, Y_4, Y_5, C)$ and distinguished decision tree **T**. Dotted lines correspond to the distinguished decision tree **T**. The component BN at the leaf l is obtained as a result of improving by constructive induction the NB model for the variables $\mathbf{X}\backslash\mathbf{T}(root, l)$.

```
1.Start from an empty tree l
2.WHILE stopping_condition==FALSE DO
    search_leaf(l)

where search_leaf(l) is

1.IF l is an empty tree or l is a leaf
  THEN extension(l)
  ELSE
      FOR every child ch of l DO
        search_leaf(ch)

and extension(l) is as follows

1.FOR every variable Y_i in X\T(root,l) DO
    a.Set ext = (X\T(root,l))\{Y_i}
    b.FOR every state y_ik of Y_i DO
      i.Let d^{ext,k} be the database restricted to the variables
        in ext, and those cases in the database consistent
        with t(root,l) and y_ik
      ii.Learn a component BN from d^{ext,k} for the variables
         in ext by means of constructive induction
    c.Score the candidate BMN
2.Choose as extension the candidate BMN with the highest score
```

Figure 11. A schematic of the algorithm for the learning of RBMNs for data clustering.

possible extension for the leaf l. Given that Eq. (9) provides us with a closed form for the log marginal likelihood for BMNs, we can use it to score the candidate BMN as follows:

$$\log p\left(\mathbf{d}^{\mathbf{X},\mathbf{t}(root,l)} \mid \mathbf{s}^h\right) = \log p\left(\mathbf{d}^{Y_i,\mathbf{t}(root,l)}\right) + \sum_{k=1}^{|Y_i|} \log p\left(\mathbf{d}^{\mathbf{ext},k} \middle| \mathbf{b}_k^h\right) \tag{11}$$

where $\mathbf{d}^{\mathbf{X},\mathbf{t}(root,l)}$ is as defined before, $\mathbf{d}^{Y_i,\mathbf{t}(root,l)}$ is the database restricted to the predictive attribute Y_i and those cases consistent with $\mathbf{t}(root, l)$, $\mathbf{d}^{\mathbf{ext},k}$ is as defined above, and $\mathbf{b}_k^h$ is the k-th component of the BMN. The first term can be calculated as the log marginal likelihood of a trivial BN having only a single node Y_i, and the terms in the sum are calculated using Eq. (8). Once all the possible candidate BMNs for extending the leaf l have been scored, the algorithm performs the extension with the highest score.

5. Experimental results

This section is devoted to the experimental evaluation of the algorithm for the learning of RBMNs for data clustering using both synthetic and real-world data. All the variables in the domains that we considered were discrete, and all the local probability distributions were multinomial distributions. In all the experiments, we assumed that the real number of clusters was known, thus, we did not perform a search to identify the number of clusters in the databases.

As we have already mentioned, currently, our algorithm for learning RBMNs from incomplete data considers 2 alternative techniques to perform the parameter search for the component BNs: the EM algorithm and the BC+EM method. According to Peña, Lozano, and Larrañaga (2000a), the BC+EM method exhibits a more desirable behaviour than the EM algorithm: faster convergence rate, and more effective and robust behaviour. Thus, the BC+EM method was the one used in our experimental evaluation, although we are aware that alternative techniques exist.

The convergence criterion for the BC+EM method was satisfied when either the relative difference between successive values of the log marginal likelihood for the model structure was less than 10^{-6} or 150 iterations were reached. Following (Peña, Lozano, & Larrañaga, 2000a), we used *fixing_probability_threshold* equal to 0.51.

As shown, the algorithm for the learning of RBMNs runs the algorithm for the learning of component BNs a large number of times. That is why the runtime of the latter algorithm should be kept as short as possible. Thus, throughout the experimental evaluation we did not consider interleaving the parameter search step after each structural change (figure 6), though it is an open question as to whether interleaving parameter and structure search would yield better results. Prior experiments (Peña, Lozano, & Larrañaga, 1999) suggest that interleaved search in our domains, however, do not yield better results. For the same reason, we only considered the forward structure search step (figure 8), thus, the initial structure for each component BN was always a NB model. These decisions were made based upon the results of the work done in Peña, Lozano, and Larrañaga (1999).

5.1. Performance criteria

In this section, we describe the criteria of table 1 that we use to compare the learnt models and to evaluate the learning algorithm. The log marginal likelihood criterion was used to select the best model structure. We use this score to compare the learnt models as well. In addition to this, we consider the runtime as valuable information. We also pay special attention to the performance of the learnt models in predictive tasks (predictive ability). Predictive ability is measured by setting aside a test set. Following learning, the log likelihood of the test set is measured given the learnt model.

All the experiments were run on a Pentium 366 MHz computer. All the results reported for the performance criteria are averages over 5 independent runs.

Table 1. Performance criteria.

Expression	Comment
$\mathbf{sc_{initial}} \pm S_n$	mean ± standard deviation of the log marginal likelihood of the initial model
$\mathbf{sc_{final}} \pm S_n$	mean ± standard deviation of the log marginal likelihood of the learnt model
$\mathbf{10CV} \pm S_n$	mean ± standard deviation of the predictive ability of the learnt model (10-fold cross-validation)
$\mathbf{Time} \pm S_n$	mean ± standard deviation of the runtime of the learning process (in seconds)

In this section, we describe our experimental results on synthetic data. Of course, one of the disadvantages of using synthetic databases is that the comparisons may not be realistic. However, seeing as the original or *gold-standard* models are known, they allow us to show the reliability of the algorithm for the learning of RBMNs from incomplete data and the improvement achieved by RBMNs over the results scored by BNs.

We constructed 4 synthetic databases ($\mathbf{d}_1$, $\mathbf{d}_2$, $\mathbf{d}_3$, and $\mathbf{d}_4$) as follows. In $\mathbf{d}_1$ and $\mathbf{d}_2$, there were 11 predictive attributes involved and 1 4-valued hidden cluster variable. 9 out of the 11 predictive attributes were 3-valued, and the 2 remaining were binary attributes. To obtain $\mathbf{d}_1$ and $\mathbf{d}_2$, we simulated 2 1-level RBMNs. Both models had a distinguished decision tree with only 1 binary predictive attribute. Thus, there were 2 component BNs in each original model. At each of these component BNs several supernodes were randomly created. The parameters for each local probability distribution of the component BNs were randomly generated as far as they defined a local multinomial distribution. Moreover, the weights of the mixture of component BNs were equal to $\frac{1}{2}$, that is, the leaves followed a uniform probability distribution. From each of these 2 RBMNs we sampled 8000 cases resulting in $\mathbf{d}_1$ and $\mathbf{d}_2$, respectively.

On the other hand, in $\mathbf{d}_3$ and $\mathbf{d}_4$, there were 12 predictive attributes involved and 1 4-valued hidden cluster variable. 9 out of the 12 predictive attributes were 3-valued, and the 3 remaining were binary attributes. For getting $\mathbf{d}_3$ and $\mathbf{d}_4$, we simulated 2 2-levels RBMNs. Both models had a distinguished decision tree with 3 binary predictive attributes. Thus, there were 4 component BNs in each original model. At each of these component BNs several supernodes were randomly created. The parameters for each local probability distribution of the component BNs were randomly generated as far as they defined a local multinomial distribution. Moreover, the weights of the mixture of component BNs were equal to $\frac{1}{4}$, that is, the leaves followed a uniform probability distribution. From each of these 2 RBMNs we sampled 16000 cases resulting in $\mathbf{d}_3$ and $\mathbf{d}_4$, respectively. Appendix shows the structures of the 4 original RBMNs sampled.

Obviously, we discarded all the entries corresponding to the cluster variable for the 4 synthetic databases. Finally, every entry corresponding to a supernode was replaced with as many entries as original predictive attributes that were grouped together under this supernode. That is, we "decoded" the Cartesian product of original predictive attributes for every entry in the database corresponding to a supernode.

Table 2 compares the performance of the learnt RBMNs for different values of the column **Depth**, which represents the depth of the distinguished decision trees. Remember that BNs were assumed to be a special case of RBMNs where the depth of the distinguished decision trees was equal to 0. It follows from the table that the algorithm for learning RBMNs from incomplete data is able to discover the complexity of the underlying model: in the databases $\mathbf{d}_1$ and $\mathbf{d}_2$, the models with the highest log marginal likelihood are those with a 1-level distinguished decision tree, whereas, in the databases $\mathbf{d}_3$ and $\mathbf{d}_4$, the learnt RBMNs with the highest log marginal likelihood are those with a 2-levels distinguished decision tree. Thus, the log marginal likelihood of the expected complete data appears to behave effectively when used to guide the search, and when considered as the stopping condition.

Table 2. Performance achieved when learning RBMNs for data clustering from the 4 synthetic databases. All the results are averages over 5 independent runs.

Database	$\mathbf{sc_{initial}} \pm S_n$	**Depth**	$\mathbf{sc_{final}} \pm S_n$	**10CV** $\pm S_n$	**Time** $\pm S_n$
$\mathbf{d}_1$	-40309 ± 188	0	-37806 ± 49	-6101 ± 107	76 ± 18
		1	-37728 ± 50	-5667 ± 128	751 ± 42
		2	-37864 ± 18	-5521 ± 66	1370 ± 50
$\mathbf{d}_2$	-40495 ± 157	0	-37600 ± 102	-5816 ± 127	110 ± 50
		1	-37412 ± 36	-5345 ± 100	956 ± 50
		2	-37554 ± 50	-5403 ± 75	1530 ± 82
$\mathbf{d}_3$	-86179 ± 145	0	-81436 ± 348	-12573 ± 203	209 ± 83
		1	-80881 ± 61	-12941 ± 83	2341 ± 176
		2	-80861 ± 68	-12046 ± 135	3915 ± 136
		3	-81134 ± 84	-11704 ± 60	5159 ± 154
$\mathbf{d}_4$	-86273 ± 285	0	-80013 ± 135	-12940 ± 442	188 ± 23
		1	-79410 ± 112	-12361 ± 44	2535 ± 143
		2	-79275 ± 72	-11231 ± 100	4086 ± 176
		3	-79513 ± 69	-11182 ± 85	5233 ± 137

The detailed analysis of the RBMN learnt in each of the 5 runs for the 4 synthetic databases considered suggests that, in general, the variables used to split the original databases in several instance subspaces (internal nodes of the distinguished decision trees of the RBMNs sampled) are discovered most of the runs. For instance, all the runs on $\mathbf{d}_1$ identify Y_1 as the root of the distinguished decision tree. Then, the learnt RBMNs recover on average 100% of the true instance subspaces. On the other hand, 3 out of the 5 runs on $\mathbf{d}_2$ discover the true attribute that splits the domain in 2 instance subspaces which results in an average of 60% of true instance subspaces discovered. For $\mathbf{d}_3$, 3 out of the 5 runs provide us with a RBMN with Y_{12} as the root of the distinguished decision tree. Moreover, 2 of these 3 runs also identify the rest of true internal nodes of the original 2-levels RBMN. The third of these 3 runs only discovers 1 of the 2 true internal nodes of the second level of the distinguished decision tree. Additionally, the other 2 runs of the 5 on $\mathbf{d}_3$ identify the 3 internal nodes of the distinguished decision tree of the original RBMN (Y_{12}, Y_1 and Y_2) but Y_2 appears as the root and, Y_{12} and Y_1 in the second level of the distinguished decision tree. Then, only 2 of the 4 instance subspaces are effectively discovered in these 2 runs. As a result, the learnt models for $\mathbf{d}_3$ discover on average 60% of the 2 main true instance subspaces and 70% of the 4 more specific true subspaces. For $\mathbf{d}_4$, 3 out of the 5 runs provide us with a RBMN that splits the original data in the 4 true instance subspaces. The remainder 2 runs provide us with RBMNs that have Y_{12} as the root of the distinguished decision trees and Y_2 as 1 of the other 2 internal nodes. However, they fail to identify Y_1 as the second node of the second level of the original distinguished decision tree. Thus, the learnt models for $\mathbf{d}_4$ discover on average 100% of the 2 main true instance subspaces and 80% of the 4 more specific true subspaces.

From the point of view of the predictive task (measured in the **10CV** column), we can report that, in general, the learnt RBMNs outperform BNs. For the databases $\mathbf{d}_1$ and $\mathbf{d}_2$, the biggest difference in the predictive ability is reached between the learnt BNs and the learnt RBMNs with **Depth** equal to 1. Remember that the underlying original models for these databases were 1-level RBMNs. Furthermore, the learnt 1-level RBMNs received the highest $\mathbf{sc}_{\mathbf{final}}$. Exactly the same is observed for the synthetic databases $\mathbf{d}_3$ and $\mathbf{d}_4$, where the biggest increase in the **10CV** is reached between the learnt RBMNs with **Depth** equal to 1 and the learnt RBMNs with 2-levels distinguished decision trees. Again, note that these 2-levels RBMNs were the models scored with the highest $\mathbf{sc}_{\mathbf{final}}$, and that the underlying original models were 2-levels RBMNs.

As the learnt RBMNs have more complex distinguished decision trees, the improvement of their predictive ability decreases. However, as a general rule, the more complex the models are, the higher the predictive ability is. This fact is well-known because 10-fold cross-validation scores the log likelihood of the test database, which does not penalize the complexity of the model, as does the log marginal likelihood. In addition, as the complexity of the distinguished decision tree increases, the instance subspaces where to learn the component BNs reduce and, thus, the uncertainty decreases. In order to avoid very complex models, our results show that the log marginal likelihood is a suitable score to guide the search for the best RBMN.

From the point of view of the efficiency (measured as the runtime of the learning process), our experimental results show that the learning of RBMNs implies a considerable computational expense when compared with the learning of BNs. However, this expense appears justified by the empirical evidence that RBMNs behave more effectively in these synthetic domains, in addition to their outlined advantages (context-specific conditional (in)dependencies, structured clustering, flexibility, etc.).

5.3. Results on real data

Another source of data for our evaluation consisted of 2 real-world databases from the UCI machine learning repository (Merz, Murphy, & Aha, 1997): the tic-tac-toe database and the nursery database. The past usage of the tic-tac-toe database helps to classify it as a paradigmatic domain for testing constructive induction methods. Despite being used for supervised classification due to the presence of the cluster variable, we considered this a good domain to evaluate the performance of our approach once the cluster entries were hidden. Furthermore, the past usage of the nursery database shows its suitability for testing constructive induction methods. In addition to this fact, the presence of 5 clusters and the large number of cases made this database very interesting for our purpose once the cluster entries were hidden.

The tic-tac-toe database contains 958 cases, each of them represents a legal tic-tac-toe endgame board. Each case has 9 3-valued predictive attributes and there are 2 clusters. The nursery database consists of 12960 cases, each of them representing an application for admission in the public school system. Each case has 8 predictive attributes, which have between 2 and 5 possible values. There are 5 clusters. Obviously, for both databases we deleted all the cluster entries.

Table 3. Performance achieved when learning RBMNs for data clustering from the 2 real-world databases. All the results are averages over 5 independent runs.

Database	$\mathbf{sc_{initial}} \pm S_n$	**Depth**	$\mathbf{sc_{final}} \pm S_n$	**10CV** $\pm S_n$	**Time** $\pm S_n$
tic-tac-toe	-4152 ± 15	0	-3932 ± 25	-508 ± 8	4 ± 1
		1	-3922 ± 7	-483 ± 7	28 ± 0
		2	-3966 ± 6	-484 ± 5	50 ± 0
nursery	-57026 ± 120	0	-53910 ± 709	-6453 ± 126	30 ± 6
		1	-53427 ± 45	-6244 ± 4	245 ± 12
		2	-53608 ± 81	-6267 ± 8	440 ± 20

Table 3 reports on the results achieved when learning RBMNs of different depth for the distinguished decision tree from the 2 real-world databases. For both databases, the learnt RBMNs outperform the learnt BNs in terms of both log marginal likelihood for the learnt models and predictive ability. The learnt 1-level RBMNs obtain the highest score for the log marginal likelihood for both domains. Moreover, these learnt RBMNs with 1-level distinguished decision trees appear to be more predictive than more complex models as the learnt RBMNs with 2-levels distinguished decision trees.

6. Conclusions and future research

We have proposed a new approach to perform data clustering based on a new class of knowledge models: recursive Bayesian multinets (RBMNs). These models may be learnt to represent the joint probability distribution from a given, complete or incomplete, database. RBMNs are a generalization of BNs and BMNs, as well as extensions to classical partitional systems. Additionally, we have described a heuristic algorithm for learning RBMNs for data clustering which simplifies the learning to the elicitation of the component BNs from incomplete data. Also, we have presented some of the advantages derived from the use of RBMNs such as codification of context-specific conditional (in)dependencies, structured and specialized domain knowledge, alternate clusterings able to capture different patterns for different instance subspaces, and flexibility.

Our experimental results in both synthetic and real-world domains have shown that the learnt RBMNs overcame the learnt BNs in terms of log marginal likelihood and predictive ability for the learnt model. Moreover, in the synthetic domains, the score to guide the structural search, the log marginal likelihood of the expected complete data, has exhibited a suitable behaviour as the instance subspaces implied by the underlying original models have been effectively discovered.

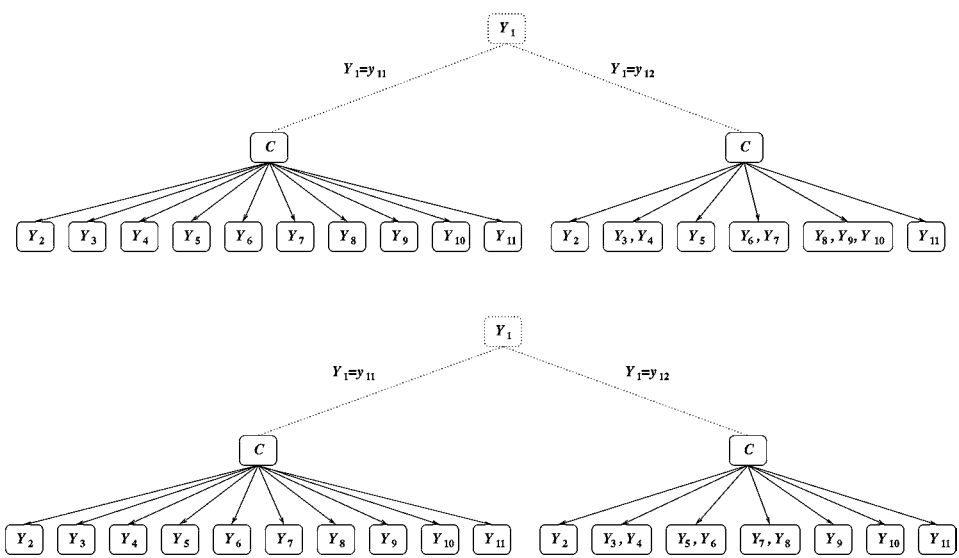
To achieve such a gain there is an obvious increase in the runtime of the learning process for RBMNs when compared with the learning of BNs. Our current research is driven to, by means of a simple data preprocessing, reduce the set of the predictive attributes that are considered to be placed in the distinguished decision tree. This reduction of the search space would imply a huge save in runtime. Since our primary aim was to introduce a new knowledge paradigm to perform data clustering, we did not focus on exploiting all its

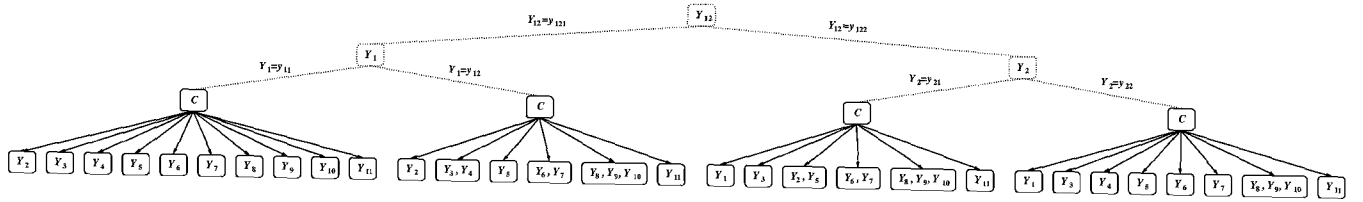
possibilities. For instance, the definition of RBMNs introduced in Section 2.4 limits the modeling power of RBMNs since all the leaves had to be at the same level. This constraint was imposed for the sake of understandability of the new model but it can be removed in practice resulting in the possibility of obtaining more natural data clusterings. A limitation of the presented heuristic algorithm for the learning of RBMNs is its *monothetic* nature, that is, only single attributes are considered at each extension of a distinguished decision tree. We are currently considering the possibility of learning *polythetic* decision paths in order to enrich the modeling power.

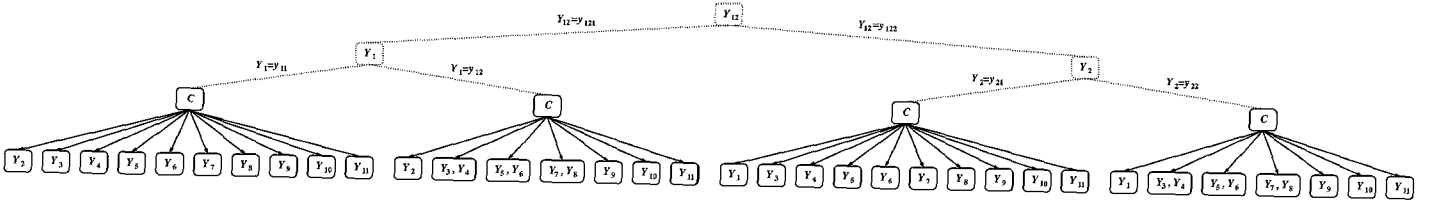
Another line of research that we are investigating is the extension of RBMNs to perform data clustering in continuous domains. In this case, component BNs would have to be able to deal with continuous attributes, thus, they would be *conditional Gaussian networks* (Lauritzen, 1996; Peña, Lozano, & Larrañaga, 2001; Peña et al., 2001). However, this approach would imply to search for the best discretization of the attributes to be considered in the decision paths. Peña et al. (2001) is an example of real-world continuous domain where these mentioned extensions of RBMNs to continuous data could be considered to perform data clustering as different patterns are observed for different instance subspaces of the original data. This extension of RBMNs to continuous domains would decrease the disrupting effects due to the discretization of the original data that would be necessary to apply RMBNs as defined in this paper to the problem domain presented in Peña et al. (2001).

Appendix: Structures of the original RBMNs sampled in order to obtain the synthetic databases

Structures of the original 1-level and 2-levels RBMNs sampled to obtain the synthetic databases. The first two model structures correspond to the RBMNs sampled to generate the synthetic databases $\mathbf{d}_1$ (top) and $\mathbf{d}_2$ (bottom), whereas the last two model structures




Y_{12}
$Y_{12}=y_{121}$
$Y_{12}=y_{122}$
Y_1
Y_2
$Y_1=y_{11}$
$Y_1=y_{12}$
$Y_2=y_{21}$
$Y_2=y_{22}$
C
C
C
C
Y_2 Y_3 Y_4 Y_8 Y_6 Y_7 Y_8 Y_9 Y_{10} Y_{11}
Y_2 Y_3, Y_4 Y_5 Y_6, Y_7 Y_8, Y_9, Y_{10} Y_{11}
Y_1 Y_3 Y_2, Y_5 Y_6, Y_7 Y_8, Y_9, Y_{10} Y_{11}
Y_1 Y_3 Y_4 Y_8 Y_6 Y_7 Y_8, Y_9, Y_{10} Y_{11}


Y_{12}
$Y_{12}=y_{121}$
$Y_{12}=y_{122}$
Y_1
Y_2
$Y_1=y_{11}$
$Y_1=y_{12}$
$Y_2=y_{21}$
$Y_2=y_{22}$
C
C
C
C
Y_2 Y_3 Y_4 Y_8 Y_6 Y_7 Y_8 Y_9 Y_{10} Y_{11}
Y_2 Y_3, Y_4 Y_5, Y_6 Y_7, Y_8 Y_9 Y_{10} Y_{11}
Y_1 Y_3 Y_4 Y_8 Y_6 Y_7 Y_8 Y_9 Y_{10} Y_{11}
Y_1 Y_3, Y_4 Y_5, Y_6 Y_7, Y_8 Y_9 Y_{10} Y_{11}

correspond to the RBMNs sampled to get the synthetic databases $\mathbf{d}_3$ (top) and $\mathbf{d}_4$ (bottom). Dotted lines correspond to the distinguished decision trees. All the predictive attributes were 3-valued except Y_1, Y_2 and Y_{12} which were binary. The cluster variable C was 4-valued.

Acknowledgments

Jose Manuel Peña wishes to thank Dr. Dag Wedelin for his interest in this work. He made the visit at Chalmers University of Technology at Gothenburg (Sweden) possible. Technical support for this work was kindly provided by the Department of Computer Science at Chalmers University of Technology.

Also, the authors would like to thank Prof. Douglas H. Fisher in addition to the two anonymous referees for their useful comments and for addressing interesting readings related to this work.

This work was supported by the spanish Ministerio de Educación y Cultura under AP97 44673053 grant.

Note

1. In the remainder of this paper, we refer to the set of nodes and supernodes of an ENB model simply as nodes.

References

Anderberg, M. R. (1973). *Cluster analysis for applications.* New York, NY: Academic Press.

Arciszewski, T., Michalski, R. S., & Wnek, J. (1995). Constructive induction: The key to design creativity. In *Proceedings of the Third International Round-Table Conference on Computational Models of Creative Design* (pp. 397–425). Heron Island, Queensland, Australia.

Banfield, J., & Raftery, A. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics, 49*, 803–821.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Buntine, W. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research, 2*, 159–225.

Castillo, E., Gutiérrez, J. M., & Hadi, A. S. (1997). *Expert systems and probabilistic network models.* New York, NY: Springer-Verlag.

Chandon, J. L., & Pinson, S. (1980). *Analyse typologique. Théories et applications.* Paris, France: Masson.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AutoClass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). San Francisco, CA: Morgan Kaufmann.

Cheeseman, P., & Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. *Advances in knowledge discovery and data mining* (pp. 153–180). Menlo Park, CA: AAAI Press.

Chickering, D. M. (1996). Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V* (pp. 121–130). New York, NY: Springer-Verlag.

Chickering, D. M., & Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning, 29*, 181–212.

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9*, 309–347.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B 39*, 1–38.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis.* New York, NY: John Wiley & Sons.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Fisher, D., & Hapanyengwi, G. (1993). Database management and analysis tools of machine induction. *Journal of Intelligent Information Systems, 2*, 5–38.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 129–138). San Francisco, CA: Morgan Kaufmann.

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

Friedman, N., & Goldszmidt, M. (1996). Building classifiers using Bayesian networks. In *Proccedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 1277–1284). Menlo Park, CA: AAAI Press.

Friedman, N., Goldszmidt, M., & Lee, T. (1998). Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. In *Proceedings of the Fifteenth National Conference on Machine Learning.*

Geiger, D., & Heckerman, D. (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence, 82*, 45–74.

Hartigan, J. A. (1975). *Clustering algorithms.* New York, NY: John Wiley & Sons.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*, 197–243.

Jensen, F. V. (1996). *An introduction to Bayesian networks.* New York, NY: Springer-Verlag.

Kaufman, L., & Rousseeuw, P. (1990). *Finding groups in data.* New York, NY: John Wiley & Sons.

Keogh, E. J., & Pazzani, M. J. (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics* (pp. 225–230). San Mateo, CA: Morgan Kaufmann.

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 202–207). Menlo Park, CA: AAAI Press.

Langley, P. (1993). Induction of recursive Bayesian classifiers. In *Proceedings of the Eighth European Conference on Machine Learning* (pp. 153–164). Berlin, Germany: Springer-Verlag.

Lauritzen, S. L. (1996). *Graphical models.* Oxford, United Kingdom: Clarendon Press.

MacQueen, J. B. (1967). Some methods for classification analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability I* (pp. 281–297). CA: University of California Press.

McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions.* New York, NY: John Wiley & Sons.

Meilă, M. (1999). Learning with mixtures of trees. Ph.D. Thesis. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Meilă, M., & Heckerman, D. (1998). An experimental comparison of several clustering and initialization methods. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 386–395). San Francisco, CA: Morgan Kaufmann.

Merz, C., Murphy, P., & Aha, D. (1997). UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Michalski, R. S. (1978). Pattern recognition as knowledge-guided computer induction. Technical Report No. 927, Department of Computer Science, University of Illinois, Urbana, IL.

Pazzani, M. J. (1996a). Constructive induction of Cartesian product attributes. In *Information, Statistics and Induction in Science* (pp. 66–77). Melbourne, Australia: World Scientific.

Pazzani, M. J. (1996b). Searching for dependencies in Bayesian classifiers. In *Learning from Data: Artificial Intelligence and Statistics V* (pp. 239–248). New York, NY: Springer-Verlag.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems.* San Mateo, CA: Morgan Kaufmann.

Peña, J. M., Lozano, J. A., & Larrañaga, P. (1999). Learning Bayesian networks for clustering by means of constructive induction. *Pattern Recognition Letters, 20*, 1219–1230.

Peña, J. M., Lozano, J. A., & Larrañaga, P. (2000a). An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters, 21*, 779–786.

Peña, J. M., Lozano, J. A., & Larrañaga, P. (2001). Performance evaluation of compromise conditional Gaussian networks for data clustering. *International Journal of Approximate Reasoning, 28*, 23–50.

Peña, J. M., Izarzugaza, I., Lozano, J. A., Aldasoro, E., & Larrañaga, P. (2001). Geographical clustering of cancer incidence by means of Bayesian networks and conditional Gaussian networks. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics* (pp. 266–271). San Francisco, CA: Morgan Kaufmann.

Peot, M. (1996). Geometric implications of the naive Bayes assumption. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence* (pp. 414–419). San Francisco, CA: Morgan Kaufmann.

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

Ramoni, M., & Sebastiani, P. (1997). Learning Bayesian networks from incomplete databases. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (pp. 401–408). San Mateo, CA: Morgan Kaufmann.

Ramoni, M., & Sebastiani, P. (1998). Parameter estimation in Bayesian networks from incomplete databases. *Intelligent Data Analysis, 2*.

Ramoni, M., & Sebastiani, P. (1999). Learning conditional probabilities from incomplete data: An experimental comparison. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics* (pp. 260–265). San Mateo, CA: Morgan Kaufmann.

Spiegelhalter, D., Dawid, A., Lauritzen, S. L., & Cowell, R. (1993). Bayesian analysis in expert systems. *Statistical Science, 8*, 219–282.

Thiesson, B., Meek, C., Chickering, D. M., & Heckerman, D. (1998). Learning mixtures of DAG models. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 504–513). San Francisco, CA: Morgan Kaufmann.

Zheng, Z. & Webb, G. I. (2000). Lazy learning of Bayesian rules. *Machine Learning, 41*, 53–84.