ELSEVIER

Pattern Recognition Letters 20 (1999) 1219–1230

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Learning Bayesian networks for clustering by means of constructive induction

J.M. Peña [*], J.A. Lozano, P. Larrañaga

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

Abstract

The purpose of this paper is to present and evaluate a heuristic algorithm for learning Bayesian networks for clustering. Our approach is based upon improving the Naive-Bayes model by means of constructive induction. A key idea in this approach is to treat expected data as real data. This allows us to complete the database and to take advantage of factorable closed forms for the marginal likelihood. In order to get such an advantage, we search for parameter values using the EM algorithm or another alternative approach that we have developed: a hybridization of the Bound and Collapse method and the EM algorithm, which results in a method that exhibits a faster convergence rate and a more effective behaviour than the EM algorithm. Also, we consider the possibility of interleaving runnings of these two methods after each structural change. We evaluate our approach on synthetic and real-world databases. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Clustering; Bayesian networks; Learning from incomplete data; Constructive induction; EM algorithm; Bound and Collapse method; Simulated annealing

1. Introduction

From the point of view adopted in this paper, *data clustering* (Duda and Hart, 1973; Hartigan, 1975; Kaufman and Rousseeuw, 1990) may be defined as the inference of a probability distribution for a database. We assume that, in addition to the observed variables, there is a *hidden* variable. This last unobserved variable would reflect the cluster membership for every case in the database. Thus, the clustering problem is also referred to as an example of learning from incomplete data due to the existence of such a hidden variable. In this paper, we focus on learning *Bayesian networks* for clustering.

In the last few years, several methods for learning Bayesian networks have arisen (Cooper and Herskovits, 1992; Heckerman et al., 1995; Pazzani, 1996b), some of them even for learning from incomplete data (Cheeseman and Stutz, 1995; Friedman, 1998; Meilă and Heckerman, 1998; Thiesson et al., 1998). A key step in the Bayesian approach to learning graphical models in general and Bayesian networks in particular is the computation of the *marginal likelihood* of a database given a model structure. This quantity is the ordinary likelihood of the database averaged over the parameters with respect to their prior distribution. When dealing with incomplete data, the

[*] Corresponding author. Tel.: +34-943-448000 (ext. 5106); fax: +34-943-219-306.
E-mail address: ccbpepaj@si.ehu.es (J.M. Peña)

exact calculation of the marginal likelihood is typically intractable (Cooper and Herskovits, 1992), thus, we have to approximate such a computation (Chickering and Heckerman, 1997). The existing methods are rather inefficient for our purpose of eliciting a Bayesian network from an incomplete database as they do not factor into separate marginal likelihoods for each family (a node and its parents).

To avoid this problem, we develop an algorithm for learning Bayesian networks for clustering based upon an adaptation of the work done in (Thiesson et al., 1998). We search for parameter values for the initial structure by means of the *EM algorithm* (Dempster et al., 1977; McLachlan and Krishnan, 1997). This fact allows us to treat expected data as real data which results in the possibility of completing the database. Therefore, a score criterion that is both in closed and factorable form can be used. In addition, we propose an alternative approach to fulfill the task of the EM algorithm which reveals a faster and more effective method: an alternation of the *Bound and Collapse* method (Ramoni and Sebastiani, 1997, 1998) and the EM algorithm. We also consider the possibility of interleaving one of these two methods (the EM algorithm or the alternative approach) after each structural change to improve the set of parameter values for the new structure.

The remaining part of this paper is structured as follows. In Section 2, we describe Bayesian networks. Section 3 is dedicated to explaining our heuristic algorithm in detail. In Section 4, we present some experimental results. Finally, in Section 5 we draw conclusions.

2. Bayesian networks

We follow the usual convention of denoting variables with 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. We use $p(\boldsymbol{x} \mid \boldsymbol{y})$ to denote the probability that $\boldsymbol{X} = \boldsymbol{x}$ given $\boldsymbol{Y} = \boldsymbol{y}$. We also use $p(\boldsymbol{x} \mid \boldsymbol{y})$ to denote the probability distribution (mass function as we restrict our discussion to the case where all the variables are discrete) for $\boldsymbol{X}$ given $\boldsymbol{Y} = \boldsymbol{y}$. Whether $p(\boldsymbol{x} \mid \boldsymbol{y})$ refers to a probability or a probability distribution should be clear from the context.

Given an n-dimensional variable $\boldsymbol{X} = (X_1, \ldots, X_n)$, a Bayesian network (BN) for $\boldsymbol{X}$ is a graphical factorization of the joint probability distribution of $\boldsymbol{X}$. A BN is defined by a directed acyclic graph $\boldsymbol{b}$ (model structure) determining the conditional independencies among the variables of $\boldsymbol{X}$ and a set of local probability distributions. When there is in $\boldsymbol{b}$ a directed arc from a variable X_j to another variable, X_i, X_j is referred to as a *parent* of X_i. We denote the set of all the parents that the variable X_i has in $\boldsymbol{b}$ as $\boldsymbol{Pa}(\boldsymbol{b})_i$. The model structure yields to a factorization of the joint probability distribution for $\boldsymbol{X}$,

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

where $\boldsymbol{pa}(\boldsymbol{b})_i$ denotes the configuration of the parents of X_i, $\boldsymbol{Pa}(\boldsymbol{b})_i$, consistent with $\boldsymbol{x}$. The local probability distributions of the BN are those in Eq. (1). We assume that the local probability distributions depend on a finite set of parameters $\boldsymbol{\theta_b} \in \boldsymbol{\Theta_b}$. Therefore, Eq. (1) can be rewritten as follows:

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

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

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

In this paper, we limit our discussion to the case in which the BN is defined by multinomial distributions. That is, all the variables are finite discrete variables and the local distributions of each variable in the BN consist of a set of multinomial distributions, one for each configuration of the parents.

3. Learning BNs for clustering through constructive induction

3.1. The BN structure that we aim to learn

Due to the difficulty involved in learning densely connected BNs and the painfully slow probabilistic inference when working with them, there has been a great interest in developing methods for learning the simplest models of BNs, e.g., *Naive-Bayes* (*NB*) models (Duda and Hart, 1973; Peot, 1996) and *Tree Augmented Naive Bayes* models (Friedman et al., 1997; Keogh and Pazzani, 1999). Despite the fact that these models are a weaker representation of some domains than more general BNs, the expressive power of these models is still recognized. Thus, these models are examples of a balance between efficiency and effectiveness, i.e., a balance between the cost of the learning process and the quality of the learnt model.

Keeping this idea in mind, we describe in this paper, a heuristic algorithm for learning BNs with a structure that can be considered as having an intermediate place in-between the NB model and the model with all the predictive variables fully correlated. By doing that we aim to keep the main features of both extremes: simplicity of the NB model and a better performance of the fully correlated model.

This class of models that we aim to learn was proposed by Pazzani (1996a,b) as a Bayesian classifier (supervised learning). In this paper, we extend these works by applying this class of models to perform data clustering. These models are very similar to the NB models, as all the attributes are independent given the class. In our approach, the only difference with the NB models is that the number of nodes in the structures of these models can be smaller than the original number of attributes in the database, as some attributes can be grouped together under the same node as fully correlated attributes (we refer to such nodes as *supernodes*). Therefore, this class of models ensures a better performance than the NB models while it keeps their simplicity. Hence, for the class of models that we learn, it follows that the probability of a case belonging to class c_i, given the values of the attributes as $\boldsymbol{x} = 2(x_1, \dots, x_n)$, is

$$p(c_i \mid \boldsymbol{x}, \boldsymbol{\theta_b}, \boldsymbol{b}^h) \propto p(c_i \mid \boldsymbol{\theta}_c, \boldsymbol{b}^h) \prod_{j=1}^{r} p(\boldsymbol{x}^j \mid c_i, \boldsymbol{\theta}_j, \boldsymbol{b}^h), \tag{4}$$

where $\{\boldsymbol{x}^1, \dots, \boldsymbol{x}^r\}$ is a partition of $\boldsymbol{x}$, r being the number of nodes (including the special nodes referred to as supernodes). Each $\boldsymbol{x}^j$ is, if the node j is a supernode, the set of values in $\boldsymbol{x}$ for the original attributes grouped together under the supernode j, else it is the value in $\boldsymbol{x}$ for the attribute j.

3.2. A heuristic algorithm for learning BNs for clustering

In the remaining part of this paper we adopt the common Bayesian approach to learning BNs. We introduce the log relative posterior probability of model structure as the score criterion to evaluate each model structure with respect to the data. Then, we search for the best model structure from among all possible models, according to this score criterion (*model selection*),

$$\log p(\boldsymbol{b}^h \mid \boldsymbol{d}) \propto \log p(\boldsymbol{d}, \boldsymbol{b}^h) = \log p(\boldsymbol{b}^h) + \log p(\boldsymbol{d} \mid \boldsymbol{b}^h). \tag{5}$$

Assuming uniform model structure priors, this criterion is reduced to the log marginal likelihood, $\log p(\boldsymbol{d} \mid \boldsymbol{b}^h)$ (Eq. (5)).

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 of the parameters given a structure is uniform, the marginal likelihood has a closed form for BNs, which allows us to compute it efficiently. In particular, assuming uniform model structure priors, we have

$$p(\boldsymbol{d} \mid \boldsymbol{b}^h) \propto \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \tag{6}$$

where n is the number of variables, r_i the number of states that the variable X_i can have, q_i the number of states that the parent set of X_i can have, N_{ijk} the number of cases in the database where X_i

has its kth value and the parent set of X_i has its jth value and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ (see (Cooper and Herskovits, 1992) for a derivation).

An important feature of the log marginal likelihood is that it factors into scores for families (under the assumptions described above). When a criterion is factorable, search is more efficient because we need not reevaluate the criterion for the whole structure from anew when only the factors of some families have changed.

When the variable that we want to classify is hidden, the assumption that the data is complete does not hold. When data is incomplete, the exact calculation of the log marginal likelihood is typically intractable (Cooper and Herskovits, 1992), thus, we have to approximate such a computation (Chickering and Heckerman, 1997). However, the existing methods for doing that are rather inefficient for our purpose of eliciting the BN structure from an incomplete database as they do not factor into scores for families.

To avoid this problem, we introduce our heuristic algorithm. A schematic of this algorithm for learning BNs for clustering is shown in Fig. 1. First, the algorithm chooses initial structure and parameter values. Then, it performs a parameter search step to improve the set of parameters for the current structure. These parameter values are used to complete the database. Doing so, we treat expected data as real data. Hence, the log marginal likelihood can be calculated using Eq. (6) in closed form. Furthermore, the factorability of Eq. (6) allows to perform an efficient structure search step. After this step, 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 step are iterated until no change in the structure occurs. We consider the possibility of interleaving the parameter search step after each structural change.

Notice should be taken of the importance of the built-in penalty term for complexity that the log marginal likelihood has. This term makes our algorithm able to avoid very complex models such as the fully correlated model. In (Meilă and Heckerman, 1998), we find a similar use of this built-in penalty term.

3.2.1. Parameter search

To fulfill the parameter search step, we consider two procedures: the EM algorithm and a new method that we present below.

The well-known EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997) is an iterative method to compute maximum a posteriori and maximum likelihood parameters from incomplete data. The EM algorithm finds a local maximum for the parameters. As the convergence rate of the EM algorithm is painfully slow, we present an alternative approach to carry out the task of the EM algorithm. In the remaining part of this paper we refer to this method as BC + EM as it alternates between the Bound and Collapse (BC) method and the EM algorithm.

```
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 for computing the $\log p(\mathbf{d}|\mathbf{b}^h)$
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.
```

Fig. 1. A schematic of the algorithm for learning BNs for clustering.

The BC method (Ramoni and Sebastiani, 1997, 1998) is a deterministic method to estimate conditional probabilities from incomplete databases. 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 that determine a probability interval 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 (see (Ramoni and Sebastiani, 1998) for further details). This method presents all the advantages of a deterministic method and a dramatic gain in efficiency when compared with the EM algorithm.

The BC is described to be used in the presence of missing data, but it is not useful when there is a hidden variable, as in the clustering problem. The reason for this limitation is that the probability intervals returned by the BC method would be too wide and poorly informative, as all the missing entries are concentrated in a single variable. The BC + EM method overcomes this problem by performing a partial completion of the database at each step (see Fig. 2).

For every case x in the database, the BC + EM uses the current parameter values to evaluate the posterior distribution of the class variable given x. Then, it assigns the case x to the class with the highest posterior probability only if this posterior probability is greater than a threshold (which is called *fixing probability threshold*) that the user must determine. The case remains incomplete if there is no class with posterior probability larger 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. Then, the EM algorithm is executed to improve the parameter values that the BC have returned. The process is repeated until convergence.

3.2.2. Structure search

The heuristic that we have presented in Fig. 1 is based upon the work done by Pazzani (1996a,b). Our heuristic algorithm learns BNs for clustering as a result of improving the NB model by searching for dependencies among attributes. In order to find the dependencies, the algorithm performs *constructive induction* (Arciszewski et al., 1995), 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 the NB model are detected and dependencies among attributes are included in the model. Therefore, we reach a better performance while the model that we obtain after the constructive induction process keeps the simplicity of the NB model. We use the term *joining* to refer to the process of creating a new attribute whose values are the Cartesian product of two other attributes.

```
1. for every case x in the database do
   a. calculate the posterior probability distribution $p(c|\mathbf{x}, \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h)$
   b. let $p_{max}$ be the maximum of $p(c|\mathbf{x}, \boldsymbol{\theta}_{\mathbf{b}}, \mathbf{b}^h)$ which is reached for $C = c_{max}$
   c. if $p_{max} > fixing\_probability\_threshold$
      then assign the case x to the class $c_{max}$
2. run the BC method
   a. bound
   b. collapse
3. set the parameter values for the current structure 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.
```

Fig. 2. The BC + EM method.

```
1. consider joining each pair of attributes
2. if there is an improvement in the $\log p(\mathbf{d}|\mathbf{b}^h)$
   then make the joint that improves the $\log p(\mathbf{d}|\mathbf{b}^h)$ the most
   else return the current case representation.
```

Fig. 3. A template for the forward structure search step.

The algorithm for learning BNs for clustering of Fig. 1 starts from one of two possible initial structures: from the NB model or from the model with all the variables fully correlated. When considering the NB model as the initial structure, the heuristic algorithm performs a *forward* structure search step (see Fig. 3). When starting from the fully correlated model, the heuristic algorithm performs a *backward* structure search step (see Fig. 4).

In addition to the hill-climbing search proposed by Pazzani, we also introduce a global search algorithm as *Simulated Annealing* (SA) (Kirkpatrick et al., 1983) in our evaluation due to the obvious limitations of a local search algorithm as hill-climbing.

In our case, a solution of SA is a model structure and the neighbourhood of solutions for the current one consists of all the solutions that can be obtained from the current one either by joining any pair of attributes or by splitting any attribute at any possible point. We use a slow cooling schedule with initial temperature (T_0) equal to 25. For each temperature T_k, we assume that the equilibrium is achieved after visiting 50 solutions, then we apply the temperature reduction function $T_{k+1} = 0.95 \cdot T_k$. The stopping criterion is satisfied when 100 consecutive iterations without changing the current solution are performed. Obviously, the score that we use to guide the search is the log marginal likelihood of the model structure. We do not consider the interleaving of parameter search steps after each structural change in order not to enlarge the long runtime of SA. So, it is senseless to compare the results obtained by SA when the EM algorithm is used and when the BC + EM method is used as they are only used to improve the parameter values of the initial solution and SA does not depend on the initial solution (assuming a proper cooling schedule).

4. Experimental results

Table 1 summarizes the criteria that we use to compare the learnt models. As we have discussed, the log marginal likelihood criterion is used to select a good model structure. We use this score in our comparisons as well. In addition to this, we consider the runtime as valuable information. For the synthetic databases (where the original models are available), we give special importance to the fraction of the nodes in the final BN which were correctly guessed, as it reflects the reliability of our algorithm for learning BNs for clustering.

For both the EM algorithm and the BC + EM method, the convergence criterion is satisfied when either the relative difference between successive values for the log marginal likelihood for the model structure is less than 10^{-6} or 150 iterations are reached.

```
1. consider splitting each attribute at each possible point
2. if there is an improvement in the $\log p(\mathbf{d}|\mathbf{b}^h)$
   then make the split that improves the $\log p(\mathbf{d}|\mathbf{b}^h)$ the most
   else return the current case representation.
```

Fig. 4. A template for the backward structure search step.

Table 1
Performance criteria

Expression	Comment
Nodes	Number of predictive nodes in the original model (including supernodes)
Method	EM algorithm or BC + EM method. The subscript *int* reflects that the method has been interleaved after each structural change and the superscript indicates the fixing probability threshold
$\mathrm{sc}_{\mathrm{initial}}$	Initial score, log marginal likelihood of the initial BN
$\mathrm{sc}_{\mathrm{final}} \pm S_n$	Mean ± standard deviation (over 5 runs) of the final score (log marginal likelihood of the final BN)
Final $\pm S_n$	Mean ± standard deviation (over 5 runs) of the number of nodes in the final BN (including supernodes)
Right $\pm S_n$	Mean ± standard deviation (over 5 runs) of the number of right guessed nodes in the final BN (including supernodes)
$\% \pm S_n$	Mean ± standard deviation (over 5 runs) of the fraction of the nodes in the final BN which were correctly guessed
Time $\pm S_n$	Mean ± standard deviation (over 5 runs) of the runtime (in seconds)

All the experiments are run on a Pentium 233 MHz computer.

Notice should be taken that in all experiments, we assume that the number of classes is known, thus, we do not perform a search to identify the number of classes in the database.

4.1. Synthetic data

In this section, we describe our experimental results on synthetic data. We constructed 4 synthetic databases as follows. There were 20 predictive binary attributes involved and one 4-valued hidden class variable. In order to get the 4 databases, we simulated 4 BNs that had the same structure as the output models of the algorithm that we introduced in Section 3.2 (we refer to them as the original models). We constructed the structure of these 4 BNs as follows. First, we began with the NB model structure where all the 20 predictive variables were independent given the class. Then, we fixed the number of predictive attributes that we wanted to have in each of the 4 output models to be 20, 14, 10 and 5, respectively. Therefore, in the last 3 models, some of the 20 original predictive variables had to be grouped together under supernodes (constructive induction) as the Cartesian product of the original variables. We joined attributes at random until the final number of nodes that we wanted to have was reached. The local probability distributions for the 4 original models were randomly generated. From each of these 4 networks, we sampled 8000 cases by means of the `h_domain_simulate` function provided by the software HUGIN API. [1] Obviously, we discarded all the entries corresponding to the class variable. Finally, every entry corresponding to a supernode was replaced with as many entries as there were original attributes under this supernode. That is, we "decoded" the Cartesian product of original attributes for every entry in the database corresponding to a supernode.

The purpose of evaluating our heuristic algorithms on synthetic databases is to show (i) the reliability of the presented heuristic for learning BNs for clustering, (ii) the improvement in the log marginal likelihood when we learn BNs by means of our algorithm and (iii) the improvement in the performance when using the BC + EM method instead of the EM algorithm.

As Table 2 shows, our algorithm, when performing forward structure search steps, most of the times, is able to reconstruct the original structure with high fidelity and reliability. Also, we can clearly see the improvement of the initial log marginal likelihood. Moreover, Table 2 compares

[1] We used HUGIN API in the implementation of our algorithm. The HUGIN API (Application Program Interface) is a library that allows a program to create and manipulate BNs.

Table 2
Performance of the algorithm for learning BNs for clustering on the 4 synthetic databases (averaged over 5 runs) when forward structure search steps are performed

Nodes	Method	$sc_{initial}$	$sc_{final} \pm S_n$	Right $\pm S_n$	Final $\pm S_n$	% $\pm S_n$	Time $\pm S_n$
20	EM	−45 875	−40 096 ± 325	15.2 ± 5.9	17.0 ± 3.7	0.86 ± 0.03	351 ± 92
	EM_{int}	−44 619	−40 005 ± 190	15.6 ± 4.7	17.2 ± 3.1	0.89 ± 0.13	456 ± 167
	$BC + EM^{0.51}$	−44 812	−39 848 ± 0	20 ± 0.0	20 ± 0.0	1 ± 0.00	228 ± 44
	$BC + EM_{int}^{0.51}$	−45 248	−39 848 ± 0	20 ± 0.0	20 ± 0.0	1 ± 0.00	271 ± 85
	$BC + EM^{0.75}$	−45 725	−40 313 ± 265	10.4 ± 5.0	13.4 ± 3.5	0.74 ± 0.14	219 ± 42
	$BC + EM_{int}^{0.75}$	−44 900	−39 849 ± 0	19.6 ± 0.8	19.8 ± 0.1	0.99 ± 0.00	235 ± 61
14	EM	−46 555	−42 652 ± 61	12.8 ± 1.0	13.4 ± 1.0	0.95 ± 0.04	633 ± 178
	EM_{int}	−46 060	−42 626 ± 88	13 ± 1.3	13.4 ± 0.8	0.97 ± 0.04	1089 ± 217
	$BC + EM^{0.51}$	−47 291	−42 639 ± 55	12.6 ± 1.2	13.2 ± 0.8	0.95 ± 0.04	479 ± 73
	$BC + EM_{int}^{0.51}$	−46 596	−42 619 ± 55	12.4 ± 1.4	13 ± 0.9	0.95 ± 0.04	982 ± 292
	$BC + EM^{0.75}$	−46 838	−42 947 ± 258	8.4 ± 1.2	10.6 ± 0.8	0.79 ± 0.06	234 ± 86
	$BC + EM_{int}^{0.75}$	−46 957	−42 818 ± 67	7 ± 1.1	9.6 ± 0.8	0.73 ± 0.07	891 ± 167
10	EM	−49 706	−47 406 ± 211	5.2 ± 1.6	8.6 ± 0.8	0.59 ± 0.15	954 ± 403
	EM_{int}	−49 561	−47 422 ± 33	8 ± 0.0	9.6 ± 0.8	0.84 ± 0.07	1498 ± 591
	$BC + EM^{0.51}$	−49 596	−47 052 ± 63	3.8 ± 1.2	7.2 ± 1.2	0.53 ± 0.12	501 ± 217
	$BC + EM_{int}^{0.51}$	−49 604	−47 072 ± 13	5.2 ± 2.2	7.4 ± 1.2	0.67 ± 0.22	1276 ± 493
	$BC + EM^{0.75}$	−49 578	−47 424 ± 43	6 ± 0.0	8.0 ± 0.0	0.75 ± 0.00	317 ± 98
	$BC + EM_{int}^{0.75}$	−49 506	−47 058 ± 0	3 ± 0.0	6 ± 0.0	0.5 ± 0.00	868 ± 3
5	EM	−49 955	−47 792 ± 143	3.4 ± 1.0	5.8 ± 0.8	0.6 ± 0.22	904 ± 475
	EM_{int}	−49 995	−47 668 ± 107	4.2 ± 0.4	6.2 ± 1.0	0.7 ± 0.16	1601 ± 466
	$BC + EM^{0.51}$	−50 023	−47 374 ± 101	3.4 ± 0.8	4.2 ± 0.4	0.8 ± 0.10	346 ± 51
	$BC + EM_{int}^{0.51}$	−50 029	−47 364 ± 29	4.2 ± 0.8	5.2 ± 0.8	0.82 ± 0.15	1198 ± 310
	$BC + EM^{0.75}$	−50 090	−47 810 ± 0	2 ± 0.0	6 ± 0.0	0.33 ± 0.00	280 ± 52
	$BC + EM_{int}^{0.75}$	−50 107	−47 319 ± 0	3.0 ± 0.0	4.0 ± 0.0	0.75 ± 0.00	862 ± 1

Table 3
Performance of the algorithm for learning BNs for clustering on the 4 synthetic databases (averaged over 5 runs) when SA is used

Nodes	Method	$sc_{initial}$	$sc_{final} \pm S_n$	Right $\pm S_n$	Final $\pm S_n$	% $\pm S_n$	Time $\pm S_n$
20	EM	−45 066	−40 048 ± 248	16.2 ± 4.7	17.2 ± 3.5	0.92 ± 0.09	15 035 ± 6031
	BC + $EM^{0.51}$	−45 545	−40 128 ± 350	16.4 ± 4.5	19.8 ± 3.5	0.94 ± 0.08	12 247 ± 3797
14	EM	−47 099	−42 755 ± 76	8.2 ± 0.8	11.6 ± 1.2	0.71 ± 0.02	5066 ± 701
	BC + $EM^{0.51}$	−46 777	−42 744 ± 120	8.8 ± 3.0	11 ± 1.7	0.78 ± 0.15	5071 ± 1223
10	EM	−49 789	−47 530 ± 632	5.2 ± 1.9	8.8 ± 1.7	0.57 ± 0.15	7775 ± 4332
	BC + $EM^{0.51}$	−49 661	−47 397 ± 255	5 ± 2.0	7.6 ± 1.0	0.64 ± 0.20	7727 ± 1433
5	EM	−50 205	−47 971 ± 446	1.8 ± 1.3	7.2 ± 1.9	0.28 ± 0.23	6167 ± 724
	BC + $EM^{0.51}$	−49 999	−47 650 ± 158	2.4 ± 1.0	6.2 ± 0.4	0.39 ± 0.18	5116 ± 2328

the performance of the algorithm for learning BNs for clustering when using the EM algorithm as parameter search step and when using the BC + EM method. As we can see in the 4 databases, the use of the BC + EM method outperforms the use of the EM algorithm in terms of final log marginal likelihood, percentage of right guessed nodes and runtime when no interleaving is done.

It seems obvious to assume that better results will be reached by interleaving parameter search steps after each structural change than without. However, most of the times, the EM algorithm needs to be interleaved to approach the performance level that our algorithm exhibits when the BC + EM method without interleaving after each structural change is used. Thus, the saving of runtime (between 50% and 78%) and the gain in effectiveness that we reach when the BC + EM method is used is clearly apparent.

Moreover, in Table 3 we present the results achieved by our heuristic algorithm when SA is used to perform the search. Comparing the results summarized in Table 2 with those presented in Table 3, we conclude that our approach performs surprisingly well. Despite being a global search algorithm, SA reveals a poor level of performance and appears to be unable to find better solutions than those achieved by the algorithm with forward structure search steps.

For the sake of brevity and due to the poor reliability and long runtime of our algorithm when using backward structure search steps, we do not present results. Thus, in the remaining part of this paper only forward structure search steps are considered when using the hill-climbing search.

4.2. Real data

Another source of data for our evaluation consists of two well-known real-world databases from the Machine Learning Repository (Merz et al., 1997): the tic-tac-toe database [2] and the nursery database. [3] The past usage of both databases classifies them as paradigmatic domains for testing constructive induction methods. Obviously, for both databases we delete all the class entries.

Table 4 reveals the effectiveness of our algorithm when forward structure search steps are done, as can be derived from the improvement of the initial log marginal likelihood. These results also show the superiority of using the BC + EM method over using the EM algorithm. For the tic-tac-toe database, better results are achieved with the BC + EM method than with the EM algorithm and in a shorter runtime. For large databases as the nursery database, when the BC + EM method shows all its advantages over the EM algorithm, making our algorithm able to reach better results with up to 11 times less runtime than when using the EM algorithm.

[2] This database contains 958 cases, each of them has 9 3-valued predictive attributes. There are 2 classes.

[3] This database consists of 12 960 cases, each of them has 8 predictive attributes which have between 2 and 5 possible values. There are 5 classes.

Table 4
Performance of the algorithm for learning BNs for clustering on the 2 real-world databases (averaged over 5 runs) when forward structure search steps are performed

Database	Method	sc_{initial}	$sc_{\text{final}} \pm S_n$	Time $\pm S_n$
Tic-tac-toe	EM	−4146	-3943 ± 15	49 ± 7
	EM_{int}	−4150	-3945 ± 20	61 ± 12
	$BC + EM^{0.51}$	−4155	-3939 ± 20	22 ± 11
	$BC + EM^{0.51}_{int}$	−4148	-3944 ± 26	36 ± 10
Nursery	EM	−57 006	$-53\,679 \pm 178$	1238 ± 409
	EM_{int}	−57 133	$-53\,567 \pm 110$	1746 ± 325
	$BC + EM^{0.21}$	−57 042	$-53\,349 \pm 0$	152 ± 13
	$BC + EM^{0.21}_{int}$	−56 913	$-53\,415 \pm 44$	155 ± 17

Table 5
Performance of the algorithm for learning BNs for clustering on the 2 real-world databases (averaged over 5 runs) when SA is used

Database	Method	sc_{initial}	$sc_{\text{final}} \pm S_n$	Time $\pm S_n$
Tic-tac-toe	EM	−4220	-3949 ± 25	806 ± 296
	$BC + EM^{0.51}$	−4144	-3954 ± 25	875 ± 46
Nursery	EM	−57 200	$-54\,193 \pm 625$	2647 ± 803
	$BC + EM^{0.21}$	−57 049	$-53\,730 \pm 597$	5207 ± 3814

The results that we present in Table 5 reinforce the good behaviour of our algorithm. Once again, SA is outperformed by the greedy algorithm with forward structure search steps which, moreover, requires less computational expense.

5. Conclusions

We proposed a new approach to perform data clustering. Our heuristic algorithm relied on treating expected data as real data and on constructive induction to improve the NB structure searching for dependencies among the attributes. In order to do that the EM algorithm played a very important role. Primarily, due to the slow convergence rate of the EM algorithm, we developed the BC + EM method to fulfill the task of the EM algorithm.

Our experimental results suggested that the greedy algorithm for learning BNs for clustering with forward structure search steps performed really well, it even reached a better performance than when SA was used. Furthermore, these experimental results showed the substantial gain in effectiveness and in efficiency that our algorithm reached when the BC + EM method was used instead of the EM algorithm.

Discussion

Hancock: How does your work compare with the structural EM algorithm that Friedman developed a few years ago? (*Note of the editors*: *see* (*Friedman*, *1998*) *in this paper*.)

Peña: In some way this work can be seen as a structural EM algorithm, but there are a few differences. Our algorithm is limited to learn this class of models because it seems to be a balance between efficiency and effectiveness. Also, I think the structural EM takes account of all the possible ways in which the databases can be completed. We just complete them with the most probable values; we do not take account of all possible completions. And we give the choice of interleaving or not interleaving the parameter search step. I think, in the structural EM you do not have such a choice: interleaving is always done. Moreover, we found that if we use the BC + EM method, we can reach a very good performance without interleaving the parameter search step. In almost all the cases for which we evaluated our algorithm, we reached the best performance with the BC + EM algorithm without interleaving, whereas in the EM algorithm interleaving must be performed to reach as good results as our algorithm without interleaving.

Hancock: My second question is about the experimental results. Can you give me some idea of how much structure you can reconstruct and for what fraction of the database you can do this?

Peña: We tested our algorithm on four synthetic databases generated with 20, 14, 10 and 5 nodes. If there are fewer than 20 nodes, this means that there are some supernodes, because all the domains have 20 original attributes. In the case of 20 nodes, almost all the methods are able to recover the original structure. In the most difficult case we reach a performance of 75%.

Hancock: Have you looked at what happens when you have extraneous structure or noise in the data?

Peña: Yes. That is why we tested the algorithm on the real world databases. The only problem is that we have to fix the probability threshold. We are now investigating if we can characterise in some way the behaviour of our algorithm.

Acknowledgements

Jose Manuel Peña wishes to thank Dr. Dag Wedelin for his interest in this work. He made it possible to visit the Department of Computer Science of Chalmers University of Technology at Gothenburg, Sweden. This work was supported by the Spanish Ministerio de Educacion y Cultura under AP97 44673053 grant.

References

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. Heron Island, Queensland, Australia, pp. 397–426.

Cheeseman, P., Stutz, J., 1995. Bayesian classification (AutoClass): Theory and results. In: Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA, pp. 153–180.

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. J. Roy. Statist. Soc. B 39, 1–38.

Duda, R., Hart, P., 1973. Pattern Classification and Scene Analysis. Wiley, New York.

Friedman, N., 1998. The Bayesian structural EM algorithm. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pp. 129–138.

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

Hartigan, J.A., 1975. Clustering Algorithms. Wiley, Canada.

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

Kaufman, L., Rousseeuw, P., 1990. Finding Groups in Data. Wiley, New York.

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, Ft. Lauderdale, FL, pp. 225–230.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

McLachlan, G.J., Krishnan, T., 1997. The EM Algorithm and Extensions. Wiley, New York.

Meilă, M., Heckerman, D., 1998. An experimental comparison of several clustering and initialization methods. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pp. 386–395.

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).

Pazzani, M.J., 1996a. Constructive induction of cartesian product attributes. In: Information, Statistics and Induction in Science. Melbourne, Australia.

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

Peot, M., 1996. Geometric implications of the naive Bayes assumption. In: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco.

Ramoni, M., Sebastiani, P., 1997. Learning Bayesian networks from incomplete databases. In: Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo.

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

Thiesson, B., Meek, C., Chickering, D.M., Heckerman, D., 1998. Learning mixtures of DAG models. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann. San Francisco, CA, pp. 504–513.