KMi

Knowledge Media Institute

Robust Learning with Missing Data

Marco Ramoni Paola Sebastiani

KMI-TR-28

22 July 1996

The Open University

Robust Learning with Missing Data

Marco Ramoni
Knowledge Media Institute
The Open University

Paola Sebastiani
Department of Actuarial Science and Statistics
City University

Abstract

Bayesian methods are becoming increasingly popular in the development of intelligent machines. *Bayesian Belief Networks* (BBNs) are nowaday a prominent reasoning method and, during the past few years, several efforts have been addressed to develop methods able to learn BBNs directly from databases. However, all these methods assume that the database is complete or, at least, that unreported data are missing at random. Unfortunately, real-world databases are rarely complete and the "Missing at Random" assumption is often unrealistic. This paper shows that this assumption can dramatically affect the reliability of the learned BBN and introduces a *robust* method to learn conditional probabilities in a BBN, which does not rely on this assumption. In order to drop this assumption, we have to change the overall learning strategy used by traditional Bayesian methods: our method bounds the set of all posterior probabilities consistent with the database and proceed by refining this set as more information becomes available. An experimental comparison — using both an artificial example and a real medical application — of our method with a powerful stochastic simulator will show a dramatic gain in robustness and the computational advantages of our deterministic method.

Keywords: Bayesian Belief Networks; Bayesian Learning; Robustness; Dirichlet Distribution; Missing Data; Gibbs Sampling.

Reference: Knowledge Media Institute Technical Report KMI-TR-28.

Address: Knowledge Media Institute. The Open University. Walton Hall, Milton Keynes, United Kingdom MK7 6AA. PHONE: +44 (1908) 655721, FAX: +44 (1908) 653169, EMAIL: `M.Ramoni@open.ac.uk`.

1 Introduction

Probabilistic methods play a central role in the development of Artificial Intelligence (AI) applications because they can deal with for the degrees of uncertainty so often embedded into real-world problems within a sound mathematical framework. Unfortunately, the number of constraints needed to define a probabilistic model, the so called *joint probability distribution*, grows exponentially with the number of variables in the domain, thus making infeasible the straight use of these methods.

However, the assumption that some of the variables are stochastically independent, given a subset of the remaining variables in the domain, dramatically reduces the number of constraints needed to specify the probabilistic model. In order to exploit this property, researchers developed a new formalism to capture, in graphical terms, the assumptions of independence in a domain, thus reducing the number of probabilities to be assessed. This formalism is known as *Bayesian Belief Networks* (BBNs) [Pearl, 1988]. BBNs provide a compact representation for encoding probabilistic information and they may be easily extended into a powerful decision-theoretic formalism called Influence Diagrams. More technically, a BBN is a direct acyclic graph in which nodes represent stochastic variables and links represent conditional dependencies among variables. A variable bears a set of possible values and may be regarded as a set of mutually exclusive and exhaustive states, each representing the assignment of a particular value to the variable. A conditional dependency links a *child* variable to a set of *parent* variables and it is defined by the set of conditional probabilities of each state of the child variable given each combination of states of the parent variables in the dependency.

In the original development of BBNs, domain experts were supposed to be their main source of information: their independence assumptions, when coupled with their subjective assessment of the conditional dependencies among the variables, produces a sound and compact probabilistic representation of their domain knowledge. However, BBNs have a strong statistical root and, during the past few years, this root prompted for the development of methods able to learn BBNs directly from databases of cases rather than from the insight of human domain experts [Cooper and Herskovitz, 1992, Buntine, 1994, Heckerman and Chickering, 1995]. This choice can be extremely rewarding when the domain of applications generates large amounts of statistical information and aspects of the domain knowledge are still unknown or controversial, or too complex to be encoded as subjective probabilities of few domain experts.

Given the nature of BBNs, there are two main tasks involved in the learning process of a BBN from a database: the induction of the graphical model of conditional independence best fitting the database at hand, and the extraction of the conditional probabilities defining the dependencies in a given graphical model from the database. Under certain constraints, this second task can be accomplished exactly and efficiently when the database is complete [Cooper and Herskovitz, 1992, Heckerman and Chickering, 1995]. We can call this assumption the "Database Completeness" assumption. Unfortunately, databases are rarely complete: unreported, lost, and corrupted data are a distinguished feature of real-world databases. In order to move on applications, methods to learn BBNs have to face the challenge of learning from databases with unreported data. During the past few years, several methods have been proposed to learn conditional probabilities in BBNs from incomplete databases, either deterministic, such as sequential updating [Spiegelhalter and Lauritzen, 1990], and [Cowel *et al.*, 1996], or the EM-algorithm [Dempster *et al.*, 1977], or stochastic such as the Gibbs Sampling [Neal, 1993]. All these methods share the common assumption that unreported data are missing at random. Unfortunately this assumption is as unrealistic as the "Database Completeness" assumption because in the real world there is often a reason so that data are missing.

This paper introduces a new method to learn conditional probabilities in BBNs from incomplete databases which does not rely on the "Missing at Random" assumption. The major feature of our method is the ability to learn BBNs which are *robust* with respect to the pattern of missing data and to return sets of estimates,instead of point estimates, whose width is a monotonic increasing function of the information available in the database. The reminder of this paper is structured as follows. In the next Section we will start by establishing some notation and reviewing the background and the motivation of this research. In Section 3 we will outline the theoretical framework of our robust approach to learn probabilities in BBNs and in Section 4 we will describe in details the algorithms we used to implement such an approach. In Section 5 we will compare the behavior of the system implemented using our method with an implementation of the Gibbs Sampling and finally we will draw our conclusions.

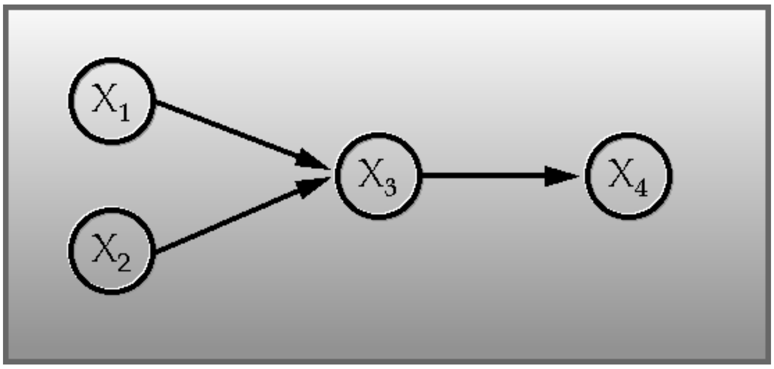

Figure 1: The graphical structure of a BBN.

2 Background

A BBN is defined by a set of *variables* $\mathcal{X} = \{X_1, \ldots, X_n\}$ and a network structure S defining a graph of conditional dependencies among the elements of $\mathcal{X}$. We will limit attention to variables which take a finite number of values and therefore, for each variable $X_i \in \mathcal{X}$, we can define a set of mutually exclusive and exhaustive *states* $x_i = \{x_{i1}, \ldots, x_{im}\}$ representing the assignments to X_i of all its m possible values.

Figure 1 shows a BBN we will use to illustrate our notation. In this case we have $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$ and we assume that each variable may take two values, $true$ (1) and $false$ (0).

The network structure S defines a set of conditional dependencies among the variables in $\mathcal{X}$. In the case reported in Figure 1, there are two dependencies: one linking the variable X_3 to its parents X_1 and X_2, which are marginally independent, and the other one linking the variable X_4 to its parent X_3. Since there are no direct links between X_1, X_2 and X_4, and X_3 separates X_4 from the nodes X_1, X_2, we have that X_4 is conditionally independent of X_1, X_2, given X_3. The conditional dependencies are defined by a set of twelve conditional probabilities: eight are $p(X_3 = x_{3j} | X_1 = x_{1h}, X_2 = x_{2k})$, $x_{1h}, x_{2k}, x_{3j} = 0, 1$, $j, h, k = 1, 2$, and four for $p(X_4 = x_{4l} | X_3 = x_{3j})$, $x_{4l} = 0, 1$, $j, l = 1, 2$. These conditional probabilities and the marginal probabilities of the parent nodes X_1 and X_2, that is $p(X_1 = x_{1h})$ and $p(X_2 = x_{2k})$, are then used to write down the joint probability of each network configuration, for instance $\{X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 1\}$, as

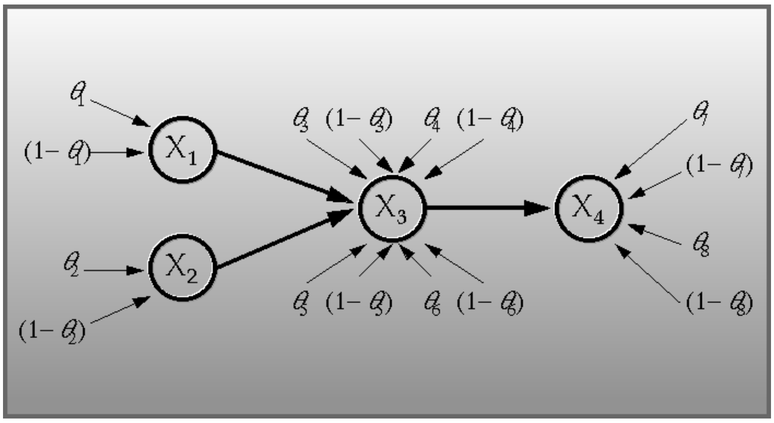

Figure 2: The graphical structure of a BBN with the associated parameters.

$$p(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 1) =$$
$$p(X_1 = 1)p(X_2 = 0)p(X_3 = 1|X_1 = 1, X_2 = 0)p(X_4 = 1|X_3 = 1).$$

Using a generic structure S, the joint probability of a particular set of values of the variables in $\mathcal{X}$, say $x = \{x_{1j}, \dots, x_{nj}\}$, can be decomposed as

$$p(\mathcal{X} = x) = \prod_{i=1}^{n} p(X_i = x_{ij}|pa(X_i) = x_{pa(x_i)})$$

where $pa(X_i)$ are the parent nodes of X_i, and $x_{pa(X_i)}$ denotes the combination of states of $pa(X_i)$ in x. If X_i is a root node, then $p(X_i = x_{ij}|pa(X_i) = x_{pa(x_i)}) = p(X_i = x_{ij})$. In the following we will denote the events $X_i = x_{ij}$ as x_{ij}, and $pa(X_i) = x_{pa(x_i)}$ as $pa(x_i)$.

We are given a database of cases $\mathcal{D} = \{C_1, \dots, C_l\}$, each case $C_j = \{x_{1j}, \dots, x_{nj}\}$ being a set of entries. We will assume that the cases are mutually independent. Given a network structure S, the task here is to learn the conditional probabilities defining the dependencies in the BBN from $\mathcal{D}$. We will consider the conditional probabilities defining the BBN as parameters $\theta = \{\theta_1, \dots, \theta_k\}$, so that the joint probability of each case in the database, say $\mathcal{X} = C_j$, becomes

$$p(\mathcal{X} = C_j|\theta) = \prod_{i=1}^{n} p(x_{ij}|pa(x_i), \theta_i),$$

X_1	X_2	X_3: 1	X_3: 0
0	0	θ_3	1- θ_3
0	1	θ_4	1- θ_4
1	0	θ_5	1- θ_5
1	1	θ_6	1- θ_6

X_3	X_4: 1	X_4: 0
0	θ_7	1- θ_7
1	θ_8	1- θ_8

Table 1: Parameters defining the conditional probabilities of the BBN displayed in Figure 1.

and θ_i parameterizes the probability of x_{ij} given the parent configuration $pa(x_i)$. For instance, in the network in Figure 1 we need only eight parameters to describe all the dependencies, two of them are needed to define the probability distributions of X_1 and X_2, say $\theta_1 = p(X_1 = 1)$ and $\theta_2 = p(X_2 = 1)$ and the other six to define the probability distributions of X_3 given $pa(x_3)$, and X_4 given $pa(x_4)$, (see Table 1.) A graphical representation is given in Figure 2.

Our task is to assess the set of parameters induced by the database $\mathcal{D}$ over the network structure S. The classical statistical parameter estimation provides the basis to learn these parameters. When the database is complete, the common approach is Maximum Likelihood which returns the parameter values that make the database most likely. Given the values of $\mathcal{X}$ in the database, the joint probability of $\mathcal{D}$ is

$$p(\mathcal{D}|\theta) = \prod_{j=1}^{l} p(\mathcal{X} = C_j|\theta).$$

This is a function of the parameters θ only, usually called the likelihood function: $l(\theta)$. The Maximum Likelihood estimate of θ is the value which maximizes $l(\theta)$. For discrete variables the Maximum Likelihood estimates of the conditional probabilities are the observed relative frequencies of the relevant cases in the database. Let $n(x_{ij}|pa(x_i))$ be the observed frequency of cases in the database with x_{ij}, given the parent configuration $pa(x_i)$, and let $n(pa(x_i))$ be the observed frequency of cases with $pa(x_i)$. The Maximum Likelihood estimate of the conditional probability of $x_{ij}|pa(x_i)$ is simply $n(x_{ij}|pa(x_i))/n(pa(x_i))$.

Consider for instance the database given in Table 2. The Maximum

Case	X_1	X_2	X_3	X_4
C_1	1	0	1	1
C_2	0	0	0	1
C_3	1	1	0	1
C_4	0	1	0	1
C_5	0	0	1	1
C_6	0	0	0	0

Table 2: An artificial database for the BBN displayed in Figure 1.

Likelihood estimate of θ_3 is $1/3$.

The Bayesian approach extend the classical parameter estimation technique in two ways: (i) the set of parameters θ are regarded as random variables, and (ii) the likelihood is augmented with a prior, say $\pi(\theta)$, representing the observer's belief about the parameters *before* observing any data. Given the information in the database, the prior density is updated in the posterior density using Bayes' theorem, and hence

$$\pi(\theta|\mathcal{D}) = \frac{\pi(\theta)p(\mathcal{D}|\theta)}{p(\mathcal{D})} \quad \text{where} \quad p(\mathcal{D}) = \int_{\mathcal{R}^k} \pi(\theta)p(\mathcal{D}|\theta)d\theta.$$

The Bayesian estimate of θ is then the expectation of θ under the posterior distribution.

The common assumptions in the Bayesian approach to learn a BBN with discrete variables are that the parameters (i) are independent from each other, and (ii) have a Dirichlet distribution, which simplifies to a Beta distribution for binary variables. Assumption (i) allows to factorise the joint prior density of θ as

$$\pi(\theta) = \prod_{i=1}^{k} \pi(\theta_i),$$

thus allowing "local computations", while (ii) facilitates computations of the posterior density by taking advantages of conjugate analysis. Full details are given for instance by [Spiegelhalter and Lauritzen, 1990]. Here we just outline the standard conjugate analysis with a Dirichlet prior.Consider for instance the variable X_i and let $\theta_i = \{\theta_{i1}, \ldots, \theta_{im}\}$ be the parameters associated to the conditional probabilities $p(x_{ij}|pa(x_i))$, $j = 1, \ldots, m$, so that

$\sum_j \theta_{ij} = 1$. A Dirichlet prior for θ_i, denoted as $D(\alpha_{i1}, \ldots, \alpha_{im})$, $\alpha_{ij} > 0$, is a continuous multivariate distribution with density function proportional to

$$\prod_j \theta_{ij}^{\alpha_{ij}-1}.$$

The hyper-parameters α_{ij}s have the following interpretation: $\alpha_{i+} = \sum_j \alpha_{ij}$ can be regarded as an imaginary sample size needed to formulate this prior information about θ_i, and the mean of θ_{ij} is α_{ij}/α_{i+}, $j = 1, \ldots, m$. Note that this prior mean is also the probability of x_{ij} given the parent configuration $pa(x_i)$. Hence, a uniform prior with $\alpha_{ij} = 1$, for all j, would assign uniform probabilities to each x_{ij}, given the parent configuration $pa(x_i)$.

With complete data, the posterior distribution of the parameters can be computed exactly using standard conjugate analysis, yielding a posterior density $D(\alpha_{i1} + n(x_{i1}|pa(x_i)), \ldots, \alpha_{im} + n(x_{im}|pa(x_i)))$, and the posterior means represent their estimates. Thus the Bayes estimate of the conditional probability of $x_{ij}|pa(x_i)$ is $\{\alpha_{ij} + n(x_{ij}|pa(x_i))\}/\{\alpha_{i+} + n(pa(x_i))\}$.

Consider again the example in Table 2. If the prior distribution of θ_3 is $D(1,1)$ so that *a priori* the conditional probability of $X_3 = 1|X_1 = 0, X_2 = 0$ is 0.5, the posterior distribution would be updated into $D(2,3)$, yielding a Bayesian estimate 2/5.

Unfortunately, the situation is quite different when some of the entries in the database are missing. When a datum is missing, we have to face a set of possible *complete* databases, one for each possible value of the variable for which the datum is missing. Exact analysis would require to compute the joint posterior distribution of the parameters given each possible completion of the database, and then to mix these over all possible completions. This is apparently infeasible. A deterministic method, proposed by [Spiegelhalter and Lauritzen, 1990] and improved by [Cowel *et al.*, 1996], provides a way to approximate the exact posterior distribution by processing data sequentially. However, [Spiegelhalter and Cowel, 1992] show that this method is not robust enough to cope with systematically missing data: in this case the estimates rely heavily on the prior distribution.

When deterministic methods fail, current practice has to resort to an estimate of the posterior means using stochastic methods. Here we will describe only the most popular stochastic method for Bayesian inference: the Gibbs Sampling. The basic idea of the Gibbs Sampling algorithm is (i) for each parameter θ_i sample a value from the conditional distribution of θ_i given all the other parameters and the data in the database, (ii) repeat this for all the parameters, and (iii) iterates these steps several times. It can be proved that, under broad conditions, this algorithm provides a sample from the joint posterior distribution of θ given the information in the database. This sample can then be used to compute empirical estimates of the posterior means or any other function of the parameters. In practical applications, the algorithm iterates a number of times and then, when stability seems to be reached, a final sample from the joint posterior distribution of the parameters is taken [Buntine, 1996].

When some of the entries in the database are missing, the Gibbs Sampling treats the missing data as unknown parameters, so that, for each missing entry, a values is sampled from the conditional distribution of the corresponding variables, given all the parameters and the available data. The algorithm is then iterated to reach stability, and then a sample from the joint posterior distribution is taken which can be used to provide empirical estimates of the posterior means [Thomas *et al.*, 1992]. This approach relies on the assumption that the unreported data are missing at random. When the "Missing at random" assumption is violated, as when data are systematically missing, such method suffers of a dramatic decrease in accuracy. The completion of the database using the available information in the database itself leads the learning system to ascribe the missing data to known values in the database and, in the case of systematically missing data, to twist the estimates of the probabilities in the database. It is apparent that this behavior can prevent the applicability of the learning method to generate BBNs because, for the general case, it can produce unreliable estimates.

3 Theory

The solution we propose is a robust method to learn parameters in a BBN. Our method computes the set of possible posterior distributions consistent with the available information in the database and proceeds by refining this set as more information becomes available. The set contains all possible estimates of the parameters in the BBN and can be efficiently computed by calculating the *extreme* posterior distributions consistent with the database $\mathcal{D}$. In this Section, we will first define some basic concepts about convex sets of probabilities and probability intervals, and then we will describe the theoretical basis of our learning method.

3.1 Probability Intervals

Traditionally, a probability function $p(x)$ assigns a single real number to the probability of x. This requirement, called *Credal Uniqueness Assumption*, is one of the most controversial points of Bayesian theory. The difficulty of assessing precise, real-valued quantitative probability measures is a long standing challenge for probability and decision theory and it motivated the development of alternative formalisms able to encode probability distributions as *intervals* rather than real numbers. The original interest for the notion of belief functions proposed by Dempster [Dempster, 1967] and further developed by Shafer [Shafer, 1976] was mainly due to its ability to represent credal states as intervals rather than point-valued probability measures [Grosof, 1986].

Several efforts have been addressed to interpret probability intervals within a coherent Bayesian framework, thus preserving the probabilistic soundness and the normative character of traditional probability and decision theory [Levi, 1980, Kyburg, 1983, Stiling and Morrel, 1991]. This approach regards probability intervals as a convex sets of standard probability distributions and is therefore referred to as *Convex Bayesianism*. Convex Bayesianism subsumes standard Bayesian probability and decision theory as a special case in which all credal states are evaluated by convex sets of probability functions containing just one element.

Convex Bayesianism relaxes the Credal Uniqueness Assumption by replacing the real-valued function $p(x)$ with a convex set of these functions. A *convex set of probability functions* $P(x)$ is the set

$$P(x) = \{p_\alpha(x) | p_\alpha(x) = \alpha p^\bullet(x) + (1-\alpha)p_\bullet(x)\}$$

where $p_\bullet(x)$ and $p^\bullet(x)$ are two probability functions and $0 \leq \alpha \leq 1$ is a real number. We will use $p_\bullet$ and $p^\bullet$ to denote the minimum and the maximum probability of the set P, respectively.

The intuition behind Convex Bayesianism is that even if an agent cannot choose a real-valued probability $p(x)$, his information can be enough to constrain $p(x)$ within a set $P(x)$ of possible values. An appealing feature of this approach is that each probability function $p_\alpha(x) \in P(x)$ fulfills the requirements of a standard probability function, thus preserving the probabilistic soundness and the normative character of traditional Bayesian theory.

The most conservative interpretation of Convex Bayesianism is the so called *Sensitivity Analysis* or *Robust Bayesian* approach [Berger, 1984], within which the set $P(x)$ is regarded as a set of precise probability functions and inference is carried out over the possible combinations of these probability functions. Good [Good, 1962] proposes to model this process as a "black box" which translates these probability functions into a set of constraints of an ideal, point-valued probability function. A review of these methods may be found in [Walley, 1991]. This theoretical framework is especially appealing when we are faced with the problem of assessing probabilities from an incomplete database. Indeed, in this case we can safely assume the existence of such an ideal, point valued probability as the probability value we would assess if the database was complete.

The suitability of Convex Bayesianism to represent incomplete probabilistic information motivated the development of computational methods to reason on the basis of probability intervals [White, 1986, Snow, 1991]. A natural evolution of this approach leaded to the combination of the computational advantages provided by conditional independence assumptions with the expressive power of probability intervals [van der Gaag, 1991]. Therefore, some efforts have been addressed to extend BBNs from real-valued probabilities to interval probabilities [Breeze and Fertig, 1990, Ramoni, 1995], thus combining the advantages of conditional independence assumptions with the explicit representation of ignorance. These efforts provide different methods to use the BBNs we can learn with our method.

3.2 Learning

We will describe the method we propose by using the artificial database in Table 2 for the BBN given in Figure 1. Table 3 reports the database given in Table 2 where some of the entries, denoted by ?, are missing. We will proceed by analyzing the database sequentially, just to show how we can derive our results. We stress here that our method does not require a sequential updating.

We start by assuming total ignorance, and hence the parameters θ_i, $i = 1, \ldots, 8$ are all given a prior distribution $D(1,1)$. We also assume that the θ_is are independent. Consider the first entry in the database, this is a complete case so that we update the distributions of θ_1, θ_2, θ_3 and θ_8 in $\theta_1|C_1 \sim D(2,1)$, $\theta_2|C_1 \sim D(1,2)$, $\theta_5|C_1 \sim D(2,1)$ and $\theta_8|C_1 \sim D(2,1)$. The distributions of θ_3, θ_4, θ_6 and θ_7 are not changed since the entries in C_1 do not have any information about them. After processing the first case we thus have that the Bayes estimates of the parameters are:

Case	X_1	X_2	X_3	X_4
C_1	1	0	1	1
C_2	0	0	?	1
C_3	1	1	?	1
C_4	0	1	0	?
C_5	0	0	1	?
C_6	0	0	0	0

Table 3: The incomplete database displayed in Table 2.

$$\begin{array}{ccccccccc} & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 \\ C_1 & 2/3 & 1/3 & 1/2 & 1/2 & 2/3 & 1/2 & 1/2 & 2/3 \end{array}.$$

Consider now case C_2. The observation on X_3 is missing, and we know that it can be either 1 or 0. Thus the possible completions of case C_2 are

$$\begin{array}{ccccc} C_{2_{c1}} & 0 & 0 & 0 & 1 \\ C_{2_{c2}} & 0 & 0 & 1 & 1 \end{array}.$$

The two possible completions would yield $\theta_1|C_1, C_2 \sim D(2,2)$, $\theta_2|C_1, C_2 \sim D(1,3)$, while θ_4, θ_5 and θ_6 would not be updated, whichever is the completion, since only the parent configuration $0,0$ is observed. Consider now the updating of the distributions of θ_3, θ_7 and θ_8. We have that $\theta_3|C_1, C_{2_{c1}} \sim D(1,2)$ and this completion would lead to $\theta_7|C_1, C_{2_{c1}} \sim D(2,1)$, and $\theta_8|C_1, C_{2_{c1}} \sim D(2,1)$. Instead $C_{2_{c2}}$ would lead to $\theta_3|C_1, C_{2_{c2}} \sim D(2,1)$, $\theta_7|C_1, C_{2_{c2}} \sim D(1,1)$ and $\theta_8|C_1, C_{2_{c2}} \sim D(3,1)$. Hence the possible Bayes estimates are:

$$\begin{array}{ccccccccc} & \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 \\ C_1, C_{2_{c1}} & 1/2 & 1/4 & 1/3 & 1/2 & 2/3 & 1/2 & 2/3 & 2/3 \\ C_1, C_{2_{c2}} & 1/2 & 1/4 & 2/3 & 1/2 & 2/3 & 1/2 & 1/2 & 3/4 \end{array}.$$

Instead of summarizing this information *somehow*, we can represent it via intervals, whose extreme points are the minimum and the maximum Bayes estimates that we would have from the possible completions of the database.

Thus, for instance, from the two possible posterior distributions of $\theta_3|C_1, C_2$ we learn $p_\bullet(X_3 = 1|X_1 = 0, X_2 = 0) = 1/3$ and $p^\bullet(X_3 = 1|X_1 = 0, X_2 = 0) = 2/3$ and similarly from the posterior distributions of θ_7 and θ_8 we learn $p_\bullet(X_4 = 1|X_3 = 0) = 1/2$ and $p^\bullet(X_4 = 1|X_3 = 0) = 2/3$, $p_\bullet(X_4 = 1|X_3 = 1) = 2/3$ and $p^\bullet(X_4 = 1|X_3 = 1) = 3/4$. Clearly, we also have $p_\bullet(X_3 = 0|X_1 = 0, X_2 = 0) = 1/3$ and $p^\bullet(X_3 = 0|X_1 = 0, X_2 = 0) = 2/3$; $p_\bullet(X_4 = 0|X_3 = 0) = 1/3$ and $p^\bullet(X_4 = 0|X_3 = 0) = 1/2$; $p_\bullet(X_4 = 0|X_3 = 1) = 1/4$ and $p^\bullet(X_4 = 0|X_3 = 1) = 1/3$ so that $p_\bullet(X_4 = 0|X_3 = 1) + p^\bullet(X_4 = 1|X_3 = 1) = 1$.

Consider now the third case in the database. Again the observation on X_3 is missing, and as before we can consider the two possible completions of this case, and proceed by updating the relevant distributions. Now however we need to "update intervals", and this can be done by updating the distributions corresponding to the extreme points of each interval. This would yield four distributions, and hence 4 Bayes estimates of the relevant conditional probabilities from which we can extract the extreme ones. For instance, the completion $C_{3_{c1}} = 1, 1, 0, 1$ would lead to the updating, among others, of the distribution of θ_7, so that the two distributions obtained after processing the first two cases in the database would become $\theta_7|C_1, C_{2_{c1}}, C_{3_{c1}} \sim D(3, 1)$, and $\theta_7|C_1, C_{2_{c2}}, C_{3_{c1}} \sim D(2, 1)$, and θ_8 is not updated. If we consider the completion $C_{3_{c2}} = 1, 1, 1, 1$, then θ_7 is not updated, while $\theta_8|C_1, C_{2_{c1}}, C_{3_{c2}} \sim D(3, 1)$ and $\theta_8|C_1, C_{2_{c2}}, C_{3_{c2}} \sim D(4, 1)$. If we sort the corresponding estimates in increasing order and find the minimum and the maximum we obtain $p_\bullet(X_4 = 1|X_3 = 0) = 1/2$ and $p^\bullet(X_4 = 1|X_3 = 0) = 3/4$, $p_\bullet(X_4 = 1|X_3 = 1) = 2/3$ and $p^\bullet(X_4 = 1|X_3 = 1) = 4/5$.

Consider $p_\bullet(X_4 = 1|X_3 = 0) = 1/2$ and $p^\bullet(X_4 = 1|X_3 = 0) = 3/4$. The minimum probability is achieved by assuming that all the completions in the database assign 1 to X_3; the maximum is achieved by assuming that all the completions in the database assign 0 to X_3. If we now process the fourth case, for instance the distribution of θ_7 would be updated by considering the completions $0, 1, 0, 1$ and $0, 1, 0, 0$ leading to extreme probabilities $p_\bullet(X_4 = 1|X_3 = 0) = 1/3$ and $p^\bullet(X_4 = 1|X_3 = 0) = 4/5$. Thus the minimum is obtained by assigning 0 to X_4 and the maximum by assigning 1 to X_4.

This can be generalized as follows. Let X_i be a binary variable in the BBN and denote by $n^\bullet(1|pa(x_i))$ the frequency of cases with $X_i = 1$, given the parent configuration $pa(x_i)$, which have been obtained by completing incomplete cases. Similarly, let $n^\bullet(0|pa(x_i))$ denote the frequency of cases with $X_i = 0$, given the parent configuration $pa(x_i)$, which have been obtained by completion of incomplete cases. Suppose further that we start from total ignorance, thus the parameter θ_i which is associated to $p(1|pa(x_i))$ is assigned a $D(1,1)$ prior, before processing the information in the database. Then

$$p(X_i = 1|pa(x_i)) \geq \frac{1 + n(1|pa(x_i))}{2 + n(1|pa(x_i)) + n(0|pa(x_i)) + n^{\bullet}(0|pa(x_i))} \tag{1}$$

and

$$p(X_i = 1|pa(x_i)) \leq \frac{1 + n(1|pa(x_i)) + n^{\bullet}(1|pa(x_i))}{2 + n(1|pa(x_i)) + n(0|pa(x_i)) + n^{\bullet}(1|pa(x_i))}. \tag{2}$$

The minimum and maximum probability of the complementary event are such that $p_{\bullet}(1|pa(x_i)) = 1 - p^{\bullet}(0|pa(x_i))$, and viceversa.

This result can be easily generalized to discrete variables with k states leading to

$$p(x_{ij}|pa(x_i)) \geq \frac{1 + n(x_{ij}|pa(x_i))}{k + \sum_j n(x_{ij}|pa(x_i)) + \max_{h \neq j} n^{\bullet}(x_{ih}|pa(x_i))} \tag{3}$$

and

$$p(x_{ij}|pa(x_i)) \leq \frac{1 + n(x_{ij}|pa(x_i)) + n^{\bullet}(x_{ij}|pa(x_i))}{k + \sum_j n(x_{ij}|pa(x_i)) + n^{\bullet}(x_{ij}|pa(x_i))} \tag{4}$$

Note that in this case the sum of the maximum probability of $x_{ij}|pa(x_i))$ and the minimum probabilities of $x_{ih}|pa(x_i)$, $h \neq j$, is one.

It is worth noting that the bounds depend only on the frequencies of complete entries in the database, and the "artificial" frequencies of the completed entries, so that they can be computed in batch mode.

4 Method

This Section is devoted to the description of the algorithms implementing the method outlined in Section 3. We will first outline the overall procedure to extract the conditional probabilities from a database given a network structure S. Then, we will describe the procedure to store the observations in the database, and we will analyze the computational complexity of the algorithms. Finally, we will provide details about the current implementation.

4.1 Overview

Section 3 described the process of learning conditional probabilities as a sequential updating of a prior probability distribution. The nature of our method allows us to implement the method as a batch procedure which first parses the database and stores the observations about the variables and then, as a final step, computes the conditional probabilities needed to specify a BBN from these observations.

The procedure takes as input a database $\mathcal{D}$, as defined in Section 2, and a network structure S. The network structure S is identified by a set of conditional dependencies $\{d_{X_1}, \ldots, d_{X_n}\}$ associated to each variable in $\mathcal{X}$. A dependency d_{X_i} is an ordered tuple $(X_i, pa(X_i))$ where X_i is a child variable in $\mathcal{X}$ with parent nodes $pa(X_i)$.

The learning procedure takes each case in the database as a *statistical unit* and parses it using the dependencies defining the network structure S. Therefore, for each entry in the case, the procedure recalls the dependency within which the entry variable appears as a child and identifies the states of its parent variables in the case. In this way, for each case in the database, the procedure detects the configuration of states for each dependency in S. Recall that the probabilities of the states of the child variables given the states of the parent variables are the parameters θ we want to learn from the database.

For each configuration $(x_{ij}|pa(x_i))$, the procedure maintains two counters, say $n(x_{ij}|pa(x_i))$ and $n^{\bullet}(x_{ij}|pa(x_i))$. When the detected configuration does not contain any missing datum, the first counter $n(x_{ij}|pa(x_i))$ is increased by one. When the datum for one or more variables in the combination is missing, the procedure increases by one the second counter $n^{\bullet}(x_{ij}|pa(x_i))$, for each configuration of states of the variable of each missing entry. In other words, the procedure uses the counter $n^{\bullet}(x_{ij}|pa(x_i))$ to ascribe a sort of *virtual observation* to each possible state of a variable whose value is missing in the case. Once the database has been parsed, we just need to collect the counters of each configuration and compute the two extreme lower bound using formula 3, and the upper bound using formula 4.

Let $\mathcal{D}$ be a database and $\mathcal{X}$ the set of variables in a BBN. We will denote as STATES(X_i) the set of states for the variable $X_i \in \mathcal{X}$ and with $pa(X_i)$ the set of its immediate predecessors in the BBN. The learning procedure is defined as follows.

```
procedure LEARN(𝒟, 𝒳)
  i←1;
  while i ≤ |𝒟| do
    j←1; R←𝒟(i);
    while j ≤ |R| do
      k←1; c←(𝒳(j), R(j)); P←pa(𝒳(j)); A←∅;
      while k ≤ |P| do
        l←1;
        while l ≤ |𝒳| do
          if P(k) = 𝒳(l) then
            A←A ∪ (𝒳(l), R(l));
          l←l + 1;
        end
        k←k + 1;
      end
      STORE(c, A,false)
      j←j + 1;
    end
    i←i + 1;
  end
  COLLECT(𝒳)
end
```

The procedure STORE stores the counters for each configuration of parent states and it will be described in the next subsection. The procedure COLLECT simply collects all the counters for each state in each variable in $\mathcal{X}$.

4.2 Storing

It is apparent that the procedure STORE plays a crucial role for the efficiency of the procedure. In order to develop an efficient algorithm, we used *discrimination trees* to store the parameter counters, following a slightly modified version of the approach proposed by [Ramoni *et al.*, 1995].

Along this approach, each state of each variable of the network is assigned to a discrimination tree. Each level of the discrimination tree is defined by the possible states of a parent variable. Each path in the discrimination tree represents a possible configuration of parent variables for that state. In this way, each path is associated to a single parameter in the network. Each leaf of the discrimination tree holds the pair of counters $n(x_{ij}|pa(x_i))$ and

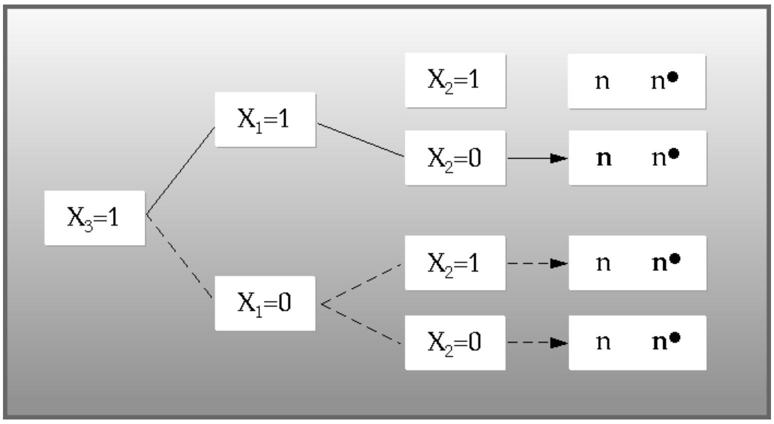

Figure 3: The discrimination tree associated to the state $X_3 = 1$.

$n^\bullet(x_{ij}|pa(x_i))$. For each entry, if there is no missing datum, we just need to follow a path in the discrimination tree to identify the counters to update. In order to save memory storage, the discrimination trees are incrementally built: each branch in the tree is created the first time the procedure needs to walk through it.

The procedure STORE uses ordered tuples to identify the variable states of interest. A state is identified by an ordered tuple (X, x), where X is the variable and x the reported value. If the value is not reported, x will be ?. The procedure STORE takes as input three arguments: an ordered tuple c representing the current state of the child variable, the set A of ordered tuples for each state of the parent variables of c in the current case, and a flag dictating whether the update is induced by a missing datum or not.

procedure STORE(c, A, v)
  if $c(2) =?$ **then**
    $i \leftarrow 1$; $S \leftarrow$STATES$(c(1))$;
    while $i \leq |S|$ **do**
      STORE$(S(i), A, true)$
      $i \leftarrow i + 1$;
    end
  else
    $p \leftarrow A(1)$; $R \leftarrow \{A - p\}$;
    if $p(2) =?$ **then**
      $S \leftarrow$STATES$(p(1))$; $j \leftarrow 1$;

```
        while j ≤ |S| do
            STORE(c, {(p(1), p(j)) ∪ R}, true)
            j←j + 1;
        end
    else
        if R = ∅ then
            if v = true then
                COUNTER•(p)←COUNTER•(p) + 1;
            else
                COUNTER(p)←COUNTER(p) + 1;
            end
            return
        else
            STORE(c, R, v)
        end
    end
  end
end
```

The functions COUNTER and COUNTER$^\bullet$ identify the counters for n and for $n^\bullet$, respectively, associated to each leaf node in the tree.

In order to illustrate how these procedures work, let's turn back to the example described in Section 2. The procedure LEARN starts by parsing the first line of the database reported in Table 3. It uses the network structure S depicted in Figure 1 to partition the case into four relevant elements:

$$
\begin{array}{lll}
X_1 = 1 & & \\
X_2 = 0 & & \\
X_1 = 1 & X_2 = 0 & X_3 = 1 \\
X_3 = 1 & X_4 = 1 &
\end{array}
$$

corresponding to a parameters $\theta_1, \theta_2, \theta_5$ and θ_8 in the network structure. Now, suppose the third entry has to be stored in the discrimination tree associated to the state $X_3 = 1$. Figure 3 displays this discrimination tree associated to the state $X_3 = 1$. The procedure walks along the solid line in Figure 3 and updates the counter $n(X_3 = 1|X_1 = 1, X_2 = 0)$ because the entry does not include any missing datum.

Suppose now the entry

$$X_1 = 0 \quad X_2 = ? \quad X_3 = 1$$

has to be stored. Figure 3 identifies with a dashed line the paths followed by the procedure in this case: when the procedure hits a state of the variable whose datum is not reported, it walks through all the possible branches following it and updates the $n^{\bullet}$ counters at the end of each path.

4.3 Computational Complexity

The learning procedure takes advantage of the modular nature of BBNs: it partions the search space using the dependencies in the network. This partitioning is the main task performed by the procedure LEARN. LEARN starts by scanning the database $\mathcal{D}$ and, for each elements of a row, it scans the row again to identify the patterns of the dependencies in the BBN. Suppose the database $\mathcal{D}$ contains n columns and m rows, the superior bound of the execution time of this part of the algorithm is $O(gmn^2)$, where g is the maximum number of parents for a variable in the BBN. Note that the number of columns is equal to the number of variables in the BBN, that is $|\mathcal{X}| = n$.

The procedure COLLECT scans the generated discrimination trees and is applied once during the procedure. The number of discrimination trees generated during the learning process is $\sum_{i=1}^{n} |\text{STATES}(X_i)|$. A discrimination tree has a number of leaves $\prod_{j=1}^{g} |\text{STATES}(p_j)|$, where $P = \{p_1, \ldots, p_g\}$ are the parents of the variable of the state associated to the discrimination tree. Note that this is the number of conditional distributions we need to learn to define a conditional dependency.

The main job of the algorithm is left to the procedure STORE Using discrimination trees, the time required by the procedure STORE to ascribe one entry to the appropriate counter is linear in the number n of parent variables in the dependency, when the reported entry about the parameter does not contain any missing datum. When data are missing, the procedure is, in the worse case, exponential in the number of parent variables with unreported data in the row. It is worth noting that the main source of complexity in the algorithm does not depend on the dimension of the database but on the topology of the BBN.

Our previous example makes clear that the order of variables in the tree plays a crucial role in the performance of the algorithm: if the positions of the states of variable X_1 and X_2 were exchanged, the procedure had to walk through the whole tree rather than just half of it. Efficiency may be gained by a careful sorting of the variable states in the tree using precompilation techniques to estimate in advance those variables whose values are missing more often in the database. This task can be accomplished using common techniques in the machine learning community to estimate the information structure of classification trees [Quinlan, 1984].

4.4 Implementation

This method has been implemented in Common Lisp on a Machintosh Performa 6300 under Machintosh Common Lisp. A porting to CLISP running on a Sun Sparc Station is under development. The system has been implemented as a module of a general environment for probabilistic inference called *Epistemic Refinment Architecture* (ERA) [Ramoni, 1995]. ERA has been originally developed on a Sun Sparc 10 using the Lucid Common Lisp development environment. We were very careful to use only standard Common Lisp resources in order to develop code conforming to the new established ANSI standard. Therefore, the code should be easily portable on any Common Lisp development environment.

The implementation of the learning system deeply exploits the modularity allowed by the Common Lisp Object System (CLOS) protocol included in the ANSI standard: for instance, the learning algorithm uses the CLOS classes implemented in the architecture to represent the elements of the network structure. This strategy allows a straightforward integration of the learning module within the reasoning modules of ERA. In this way, the results of the learning process are immediately available to the reasoning modules of ERA to draw inferences and make decisions.

5 Experimental Evaluation

Gibbs Sampling is currently the most popular stochastic method for Bayesian inference in complex problems, such as learning when some of the data are missing, although its limitations are well-known: the convergence rate is slow and resource consuming. However, given its popularity, we have compared the accuracy of our method with one of its implementations. In this Section, we will report the results of two sets of experimental comparisons, one using a real-world problem and one using an artificial example. The aim of these experiments is to compare the accuracy of the parameter estimates provided by the Gibbs Sampling and our method as the available information in the database decreases.

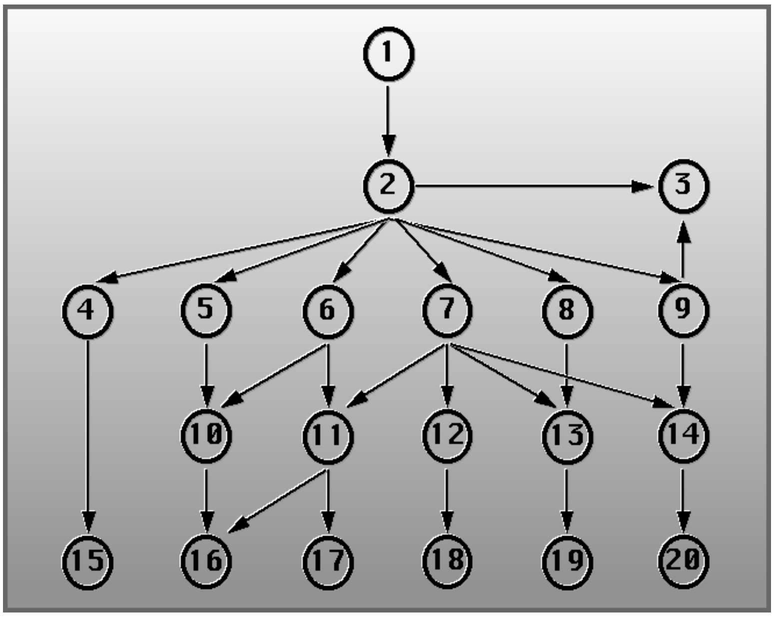

Figure 4: The network structure of the BBN used for the first set of experiments.

5.1 Materials

In order to experimentally compare our method to the Gibbs Sampling, we choose the program BUGS [Thomas *et al.*, 1992] which is commonly regarded as a reliable implementation of such a technique [Buntine, 1996]. In the following experiments, we used the implementation of BUGS version 0.5 running on a Sun Sparc 5 under SunOS 5.5 and the ERA implementation of our method running on a Machintosh PowerBook 5300 under Machintosh Common Lisp version 3.9. All the experiments reported in this Section share these materials. Different materials used for each set of experiments will be illustrated during the description of each experiment.

5.2 Experiment 1: The CHILD Network

In the first set of experiments, we used a well-known medical problem. This problem has been already used in the BBNs literature [Spiegelhalter *et al.*, 1993] and concerns the early diagnosis of congenital hearth disease in newborn babies.

#	**Name**	**States**
1	Birth Asphyxia	yes no
2	Disease	PFC TGA Fallot PAIVS TAPVD Lung
3	Age	0-3-days 4-10-days 11-30-days
4	LVH	yes no
5	Duct flow	Lt-to-Rt None Rt-to-Lt
6	Cardiac mixing	None Mild Complete Transp
7	Lung parenchema	Normal Congested Abnormal
8	Lung flow	Normal Low High
9	Sick	yes no
10	Hypoxia Distribution	Equal Unequal
11	Hypoxia in O2	Mild Moderate Severe
12	CO2	Normal Low High
13	Chest X-ray	Normal Oligaemic Plethoric Grd-Glass Asy/Patchy
14	Grunting	yes no
15	LVH Report	yes no
16	RUQ O2	<5 5-12 12+
17	Lower Body O2	<5 5-12 12+
18	CO2 Report	<7.5 >=7.5
19	X-ray Report	Normal Oligaemic Plethoric Grd-Glass Asy/Patchy
20	Grunting Report	yes no

Table 4: Definition of the variables in the CHILD BBN: the first column reports the numeric index used in Figure 4, the second the variable name, and the third its possible states.

5.2.1 Materials

Figure 4 displays the network structure of our medical problem. Table 4 associates each number in the BBN to the name of a variable and reports its possible values. The clinical problem underlying the BBN depicted in Figure 4 is described in [Frankin *et al.*, 1989]. The task of the BBN is to diagnose the occurrence of congenital heart disease in pediatric patients using clinical data reported over the phone by referring pediatricians. Authors report that the referral process generates a large amount of clinical information but this information is often incomplete and includes missing and unreported data.

These features, together with the reasonable but realistic size of the BBN, makes of this problem an ideal testbed for our experiments: the BBN in Figure 4 is defined by 344 conditional probabilities (although the minimal set

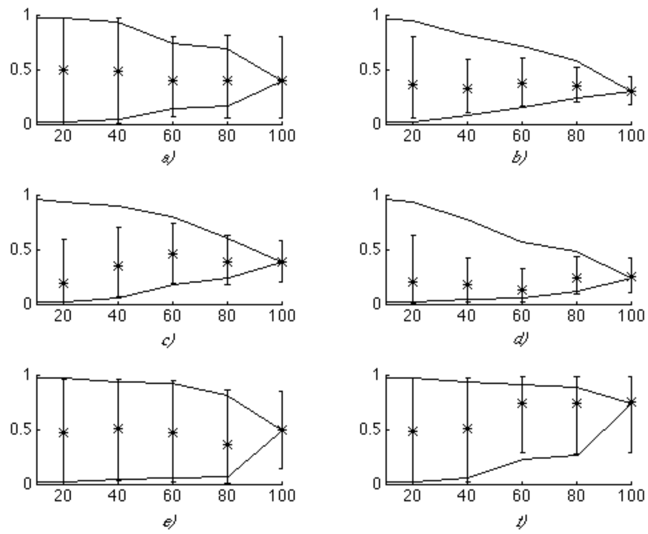

Figure 5: Plots of estimates against amount of information for the parameters *a)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{PCF})$, *b)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{TGA})$, *c)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{fallot})$, *d)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{PAIVS})$, *e)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{TAPVD})$, and *f)* $p(\mathtt{n9} = \mathtt{yes}|\mathtt{n2} = \mathtt{lung})$.

is slightly more than 240 since some of the variables in the BBN are binary) and the number of cases reported in the original database was more than 100. Since the aim of our experiment is to test the accuracy of the competing methods, we cannot use the original database because we need a reliable measure of the probabilities we want the systems to assess. Therefore, still using the original network, we generated a complete random sample of 100 cases from a known distribution and we used this sample as the database to learn from. Using this database and the BBN in Figure 4, we run two different tests.

5.2.2 Test 1: Missing at Random

The goal of the first test is to characterize the behavior of our method with respect to the Gibbs Sampling when the “Missing at Random” assumption holds.

Method. We start with a complete database, where all the parameters are independent and uniformly distributed, and run both our learning algorithm and the Gibbs Sampling on it. Then, we proceed by randomly deleting the 20% of entries in the database, and by running the two methods on the incomplete database, until the database is empty. For each incomplete database we run 10,000 iterations of the Gibbs Sampling, which appeared to be enough to reach stability, and the estimates returned are based on a final sample of 5,000 values.

Results. Figure 5 shows the estimates of the conditional probabilities defining the dependency linking the variable `n2` (*disease*) to the variable `n9` (*sick*) inferred by the two learning methods, for 5 different proportions of completeness. We report this dependency as an example of the overall behavior of the systems during the test. Stars report the point estimates given by the Gibbs Sampling, while errorbars indicates 95% confidence interval about the estimates. Solid lines indicates the lower and upper bounds of the probability intervals inferred by our method. We choose to represent the outcomes of the Gibbs Sampling and our system in two different ways because there is a basic semantic clash between the Gibbs Sampler confidence intervals and the intervals returned by our system. The estimates given by the Gibbs Sampling, when the database is incomplete, are based on the most likely reconstruction of the missing entries, and this relies on the prior belief about the parameters and the complete data. The 95% confidence intervals represent the posterior uncertainty about the parameters, which depend on the inferred database and the prior uncertainty. Thus the larger the intervals, the less reliable are the estimates. Intervals returned by our method represent the set of posterior estimates of the parameters that we would obtain in considering *all* possible completions of the database. In particular, both the estimates based on the original database and the estimates returned by the Gibbs Sampling will be one of them. The width of our intervals is then a measure of the uncertainty in considering all possible completions of the database.

When the information in the database is complete, the intervals of our systems degenerate to a single point, and this point coincides with the exact estimates. The semantic difference between the intervals returned by the two systems accounts for some of the slightly tighter intervals returned by the Gibbs Sampling, together with the fact that, by assuming that the data are missing at random, the Gibbs Sampling can exploit a piece of information

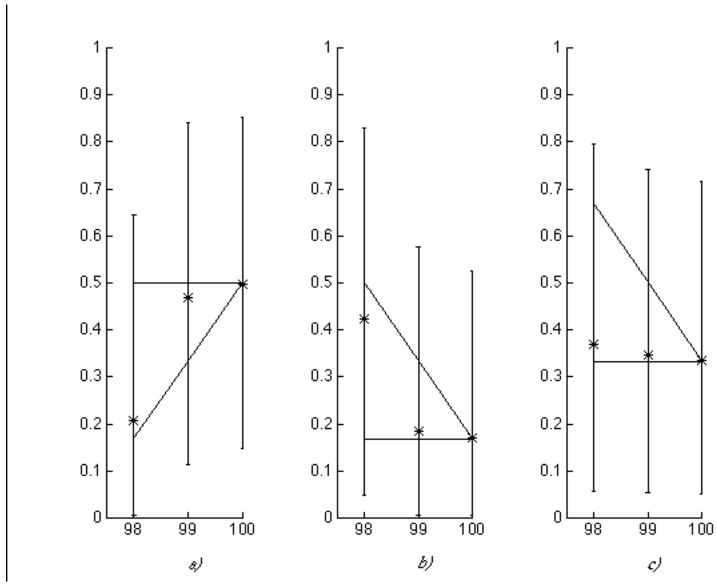

Figure 6: Plots of estimates against amount of information for the parameters *a)* $p(\texttt{n7} = \texttt{normal}|\texttt{n2} = \texttt{PCF})$, *b)* $p(\texttt{n7} = \texttt{oedema}|\texttt{n2} = \texttt{PCF})$, and *c)* $p(\texttt{n7} = \texttt{abnormal}|\texttt{n2} = \texttt{PCF})$.

not available to our system. Nonetheless, the width of the intervals returned by the two systems is overall comparable in the case reported in Figure 5, as well as for all the remaining parameters in the BBN.

The main difference was in the execution time: in the worse case, Gibbs Sampling took over 37 minutes to run to completion on a Sun Sparc 5, while our system ran to completion in less than 0.20 seconds on a Machintosh PowerBook 5300.

5.2.3 Test 2: Systematically Missing

What does it happen when the data are not missing at random and therefore the Gibbs Sampling is using a wrong guess? The aim of the second test is to compare the behavior of the two systems when the data are not missing at random, but the value of a variable is systematically removed from the database.

Method. The procedure used for this test is a slight modified version of the test for the randomly missing data. In this case, we iteratively deleted

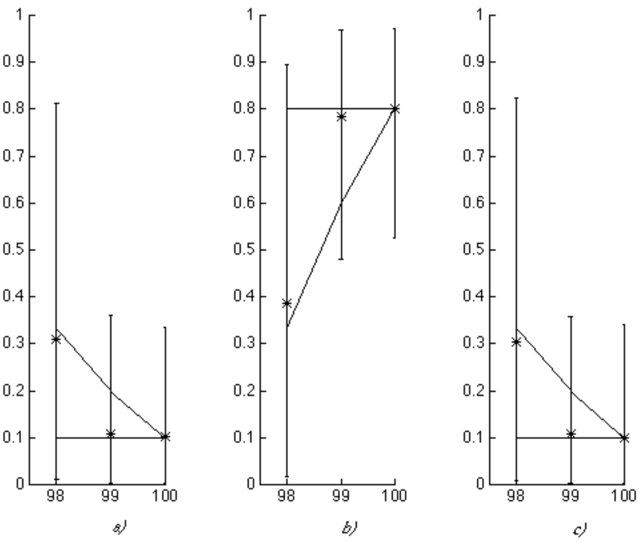

Figure 7: Plots of estimates against amount of information for the parameters *a)* $p(\texttt{n11} = \texttt{none}|\texttt{n7} = \texttt{normal}, \texttt{n6} = \texttt{mild})$, *b)* $p(\texttt{n11} = \texttt{moderate}|\texttt{n7} = \texttt{normal}, \texttt{n6} = \texttt{mild})$, and *c)* $p(\texttt{n11} = \texttt{severe}|\texttt{n7} = \texttt{normal}, \texttt{n6} = \texttt{mild})$.

1% of the database by systematically removing the entries reporting the value `normal` for the variable `n7` (*lung parenchema*) and we ran the two learning algorithms. Each run of the Gibbs Sampling is based on 10,000 iterations, and a final sample of 5,000 cases. This procedure was iterated until no value `normal` was reported for the variable `n7` in the database.

Results. The local independencies induced by the network structure are such that the modification of values of the variable `n7` in the database only affect the estimates of the conditional probabilities for its immediate predecessors and successors. Therefore, we will focus on the estimates of the parameters affected by the changes. Figure 6 shows the estimates provided by the two systems for the conditional probabilities of the states of variable `n7` given the state `n2 = PCF` of its immediate predecessor `n2` (*disease*) in the BBN. Note that the information percentage reported on the $x-$axis starts at 98%, meaning that the number of entries for the state `n7 = normal` account for the 2% the complete database. At 98%, the database contains no entry for `n7 = normal`. Results report that the point-valued estimates of the Gibbs Sampling always fall within the intervals calculated by our system. However, plots *a* and *b* display the behavior of the Gibbs Sampling estimates, jumping from the lower bound to the upper bound of the set calculated by our method as the missing data are replaces with entries `n7 = normal`. The jump testifies a 30% error in the point estimate provided by the Gibbs Sampling when all the entries `n7 = normal` are missing. The behavior of both systems does not change when the data are missing on the parent variables rather than on the child variable. Figure 7 plots the estimates of some parameters of the dependency linking the parent variables `n7` (*lung-parench*) and `n6` (*Cardiac Mixing*) to the child variable `n11` (*Hypoxia in O2*). In this case, the initial error of the Gibbs Sampling goes up to the 45%, as shown in the plot *b*. This error is even more remarkable once we realize that the 98% of the overall information is still available.

The difference in execution time was comparable to the one of the previous test: in the worse case, Gibbs Sampling took over 16 minutes to run to completion on a Sun Sparc 5, while our system ran to completion in less than 0.20 seconds on a Machintosh PowerBook 5300.

5.3 Experiment 2: An Artificial Network

Results from experiments on the CHILD network show a bias of the point estimates given by the Gibbs Sampling, although the associated confidence intervals are always large enough to include the estimates calculated from the complete database. Since the width of the confidence intervals is a function of the sample size, we are left with the doubt that a large database could give tighter intervals around biased estimates. These results prompt for a more accurate investigation of this behavior as the size of the database increases. Thus, in this Section, we will use an artificial — and therefore more controllable — example in order to amplify this bias. The rationale behind this second experiment is twofold: (i) to display the effect on the learning process of the bias induced by the "Missing at Random" assumption when in fact data are systematically missing and the size of the database is large, and (ii) to show the effect of this bias on the predictive performance on the BBN.

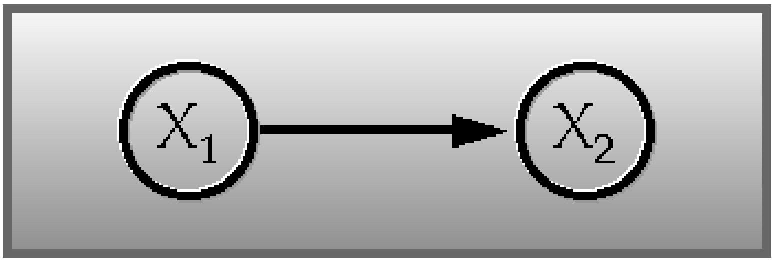

Figure 8: The simple network used for the second set of experiments.

5.3.1 Materials

Figure 8 shows the simple BBN we used for our second experiment: two binary variables linked by a dependency. We generated a database of 1000 random cases from the following probability distribution:

$$\begin{aligned} p(X_1 = 1) &= 0.5 \\ p(X_2 = 1|X_1 = 1) &= 0.9 \\ p(X_2 = 1|X_1 = 0) &= 0.1 \end{aligned}$$

The parameters in the BBN were all assumed to be independent and uniformly distributed.

5.3.2 Learning

Using the BBN displayed in Figure 8, we tried to make clearer the bias effect detected in the previous set of experiments.

Method. We followed a procedure analogue to that used in the second test of the previous experiment: we iteratively deleted the 10% of entries with $X_1 = 0$, until no value with $X_1 = 0$ was left in the database. We then run our method and the Gibbs Sampling on each incomplete database. Each run of the Gibbs Sampling is based on 1,000 iterations to reach stability, and a final sample of 2,000 values.

Results. Figure 9 shows the parameter estimates given by the two systems. The bias of the Gibbs Sampling is absolutely clear: when 75% of the information is available in the database but all the entries of $X_1 = 0$ are missing, the estimate given by the Gibbs Sampling for $p(X_1 = 0)$ lies on the lower extreme of the interval estimated by our method. However,

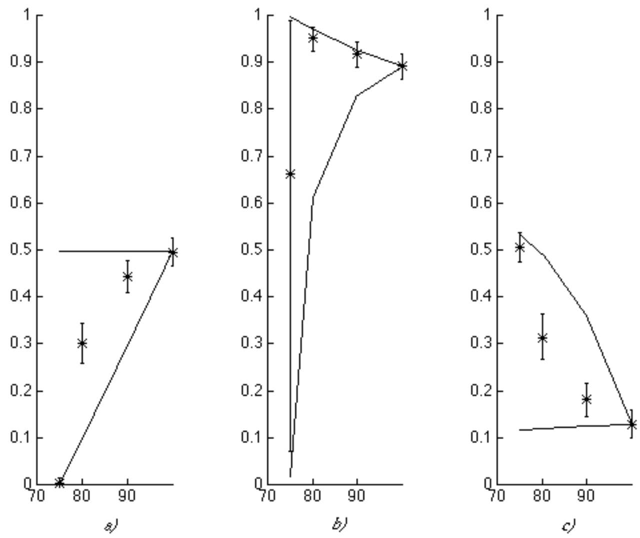

Figure 9: Plots of estimates agaist amount of information for the parameters *a)* $p(X_1 = 0)$, *b)* $p(X_2 = 0|X_1 = 0)$, and *c)* $p(X_2 = 0|X_1 = 1)$.

the sample size tight up the confidence interval around the estimate 0.0034 so the "true" estimate 0.4955, which would be computed in the complete database, is definitely excluded, with an error overpassing the 40%.

The estimate of $p(X_2 = 0|X_1 = 0)$ moves from 0.8919 in the complete database to 0.662 when no entries with $X_1 = 0$ are left in the database. However, the wide interval associated to the estimate testifies the low confidence of the Gibbs Sampler in it.

More dramatic is the effect of the *Missing at Random* assumption on the estimates of the conditional probability $p(X_2 = 0|X_1 = 1)$, which moves from 0.1286 in the complete database, to 0.5059 , with a very narrow confidence interval: 0.4752,0.5367. This narrow interval overestimates the reliability of the inferred value and excludes the *true* value inferred from the complete database.

Execution time was 10 minutes for the Gibbs Sampling and less than 0.20 seconds for our system.

5.4 Prediction

The goal of learning BBNs is to use them to perform different reasoning tasks, such as prediction. The aim of this second test is to evaluate the reliability of the predictions given by the learned BBN.

Materials. We used the BBN learned by the two systems in the previous test to predict the value of X_2 given an observation about X_1.

Results. The effect of the strong bias in the estimates returned by the Gibbs Sampling is remarkable in the predictive performance of the network. Suppose that $X_1 = 1$ is observed and we want to predict the value of X_2. Since in this case $p(X_2 = 0)$ reduces to $p(X_2 = 0|X_1 = 1)$, Figure 9 c) plots the marginal probability of $p(X_2 = 0)$ as well. Suppose that we use the estimates learned by the Gibbs Sampling with 75% of the complete data, when all the entries $X_1 = 0$ are missing. The prediction of the Gibbs Sampling is $p(X_2 = 0) = p(X_2 = 0|X_1 = 1) = 0.5069$, against the value 0.1286 that we would have inferred from the complete database. Instead, our method returns the probability interval $[0.12, 0.54]$, thus including the "true" value.

5.5 Discussion

The results of these experiments match our expectations. The accuracy of the two systems is overall comparable when data are missing at random. However, the estimates given by the Gibbs Sampling are prone to bias when data are systematically missing. The reason of this behavior is easy to identify. Consider for instance the artificial example in Figure 8. In the reconstruction of the original database, the Gibbs Sampling exploits the available information to ascribe the missing entries mainly to $X_1 = 1$, and hence the high confidence on the estimate of the conditional distribution of $X_2|X_1 = 1$.

These results show also that our method is robust because instead of "betting" on the most likely complete database that we could infer from the available information from the incomplete database at hand, our method returns results which make the problem solver aware of his own ignorance. This feature is even more important when we consider the remarkable effect of the strong bias in the estimates returned by the Gibbs Sampling on the predictive performance of the BBN.

A last word about the execution time. The computational cost of Gibbs Sampling is a well known issue and the results of our experiments show the computational advantages of our deterministic method with respect to the stochastic simulator.

6 Conclusions

Incompleteness is an common feature of real-world databases and the ability of learning from databases with incomplete data is a basic challenge researchers have to face in order to move their methods to applications. A key issue for a method able to learn from incomplete database is the reliability of the knowledge bases it will generate. The results of our investigation shows that the common Missing at Random assumption exploited by current learning methods can dramatically affect the accuracy of their results.

This paper introduced a *robust* method to learn conditional probabilities in a BBN which does not rely on this assumption. In order to drop this assumption, we had to change the overall learning strategy with respect to traditional Bayesian methods: rather than guessing the value of missing data on the basis of the available information, our method bounds the set of all posterior probabilities consistent with the database and proceed by refining this set as more information becomes available.

The main feature of this method is its robustness with respect to the distribution of missing data: it does not relies on the assumption that data are missing at random because it does not try to infer them from the available information. The basic intuition behind our method is that we are better off if, rather than trying to complete the database by guessing the value of missing data, we regard the available information as a set of constraints on the possible distributions in the database and we reason on the basis of the set of probability distributions consistent with the database at hand.

An experimental comparison between our method and a powerful stochastic method shows a remarkable difference in accuracy between the two methods and the computational advantages of our deterministic method with respect to the stochastic one.

Acknowledgments

Authors thank Greg Cooper, Pat Langley, Zdenek Zdrahal for their useful suggestions during the development of this research. Equipment has been provided by generous donations from Apple Computers and Sun Microsystems.

References

[Berger, 1984] J.O. Berger. The robust bayesian viewpoint. In J.B. Kadane, editor, *Robustness of Bayesian Analyses*. Elsevier, 1984.

[Breeze and Fertig, 1990] J. Breeze and K.W. Fertig. Decision making with interval influence diagrams. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 122–129, 1990.

[Buntine, 1994] W. L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[Buntine, 1996] W. Buntine. A guide to the literature on learning probabilistic networs from data. *ieeetdke*, 1996. To appear.

[Cooper and Herskovitz, 1992] G.F. Cooper and E. Herskovitz. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[Cowel *et al.*, 1996] R G Cowel, A.P. Dawid, and P. Sebastiani. A comparison of sequential learning methods for incomplete data. *Bayesian Statistics 5*, pages 581–588, 1996.

[Dempster *et al.*, 1977] A. Dempster, D. Laird, and Rubin D. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Statist. Soc. B*, 39:1–38, 1977.

[Dempster, 1967] A. P. Dempster. Upper and lower probabilities induced by multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[Frankin *et al.*, 1989] R. C. G. Frankin, D J Spiegelhalter, F Macrtney, and K Bull. Combining clinical judgments and clinical data in expert systems. *Internation Journal of Clinical Monitoring and Computing*, 6:157–166, 1989.

[Good, 1962] I.J. Good. Subjective probability as a measure of a non-measurable set. In E. Nagel, P. Suppes, and A. Tarsky, editors, *Logic,* *Methodology and Philosophy of Science*, pages 319–329. Stanford University Press, Stanford, CA, 1962.

[Grosof, 1986] B. N. Grosof. An inequality paradigm for probabilistic knowledge. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 259–275. North-Holland, Amsterdam, 1986.

[Heckerman and Chickering, 1995] D. Heckerman, D. amd Geiger and D.M. Chickering. Learning bayesian networks: The combinations of knowledge and statistical data. *Machine Learning*, 1995.

[Kyburg, 1983] H.E. Kyburg. Rational belief. *Behavioral and Brain Sciences*, 6:231–273, 1983.

[Levi, 1980] I. Levi. *The Enterprise of Knowledge. An Essay on Knowledge, Credal Probability and Chance.* MIT Press, Cambridge, MA, 1980.

[Neal, 1993] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, Department of Computer Science, University of Toronto, 1993.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of plausible inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[Quinlan, 1984] J. R. Quinlan. Learning efficient classification procedures and their application to chess and games. In R.S. Michalsky, J.G Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–481. Springer Verlag, Berlin, 1984.

[Ramoni *et al.*, 1995] M. Ramoni, A. Riva, M. Stefanelli, and V. Patel. An ignorant belief network to forecast glucose concentration from clinical databases. *Artificial Intelligence in Medicine Journal*, 7:541–559, 1995.

[Ramoni, 1995] M. Ramoni. Ignorant influence diagrams. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1808–1814, S. Mateo, CA, 1995. Morgan Kaufman.

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[Snow, 1991] P. Snow. Improved posterior probability estimates from prior and linear constraint system. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:464–469, 1991.

[Spiegelhalter and Cowel, 1992] D J Spiegelhalter and R G Cowel. Learning in probabilistic expert systems. *Bayesian Statistics 4*, pages 447–466, 1992.

[Spiegelhalter and Lauritzen, 1990] D.J. Spiegelhalter and S.L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:157–224, 1990.

[Spiegelhalter *et al.*, 1993] D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, and R.G. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8:219–247, 1993.

[Stiling and Morrel, 1991] W. Stiling and D. Morrel. Covex bayesian decision theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:173–183, 1991.

[Thomas *et al.*, 1992] A Thomas, D J Spiegelhalter, and W R Gilks. Bugs: A program to perform bayesian inference using gibbs sampling. *Bayesian Statistics 4*, pages 837–42, 1992.

[van der Gaag, 1991] L. van der Gaag. Computing probability intervals under independency constraints. In *Uncertainty in Artificial Intelligence*, pages 491–497, 1991.

[Walley, 1991] P. Walley. *Statistical Reasoning with Imprecise Probabilities.* Chapman and Hall, London, 1991.

[White, 1986] C.C. White. A posteriori representations based on linear inequality descriptions of a priori conditional probabilities. *IEEE Transactions on Systems, Man, and Cybernetics*, 16:570–573, 1986.