Machine Learning, 9, 309–347 (1992)

A Bayesian Method for the Induction of Probabilistic Networks from Data

GREGORY F. COOPER GFC@MED.PITT.EDU
Section of Medical Informatics, Department of Medicine, University of Pittsburgh, B50A Lothrop Hall, Pittsburgh, PA 15261

EDWARD HERSKOVITS EHH@SUMEX-AIM.STANFORD.EDU
Noetic Systems, Incorporated, 2504 Maryland Avenue, Baltimore, MD 21218

Editor: Tom Dietterich

Abstract. This paper presents a Bayesian method for constructing probabilistic networks from databases. In particular, we focus on constructing Bayesian belief networks. Potential applications include computer-assisted hypothesis testing, automated scientific discovery, and automated construction of probabilistic expert systems. We extend the basic method to handle missing data and hidden (latent) variables. We show how to perform probabilistic inference by averaging over the inferences of multiple belief networks. Results are presented of a preliminary evaluation of an algorithm for constructing a belief network from a database of cases. Finally, we relate the methods in this paper to previous work, and we discuss open problems.

Keywords. probabilistic networks, Bayesian belief networks, machine learning, induction

1. Introduction

In this paper, we present a Bayesian method for constructing a probabilistic network from a database of records, which we call *cases*. Once constructed, such a network can provide insight into probabilistic dependencies that exist among the variables in the database. One application is the automated discovery of dependency relationships. The computer program searches for a probabilistic-network structure that has a high posterior probability given the database, and outputs the structure and its probability. A related task is computer-assisted hypothesis testing: The user enters a hypothetical structure of the dependency relationships among a set of variables, and the program calculates the probability of the structure given a database of cases on the variables.

We can also construct a network and use it for computer-based diagnosis. For example, suppose we have a database in which a case contains data about the behavior of some system (i.e., findings). Suppose further that a case contains data about whether this particular behavior follows from proper system operation, or alternatively, is caused by one of several possible faults. Assume that the database contains many such cases from previous episodes of proper and faulty behavior. The method that we present in this paper can be used to construct from the database a probabilistic network that captures the probabilistic dependencies among findings and faults. Such a network then can be applied to classify future cases of system behavior by assigning a posterior probability to each of the possible faults and to the event "proper system operation." In this paper, we also shall discuss diagnostic inference that is based on combining the inferences of multiple alternative networks.

Table 1. A database example. The term *case* in the first column denotes a single training instance (record) in the database—as for example, a patient case. For brevity, in the text we sometimes use 0 to denote *absent* and 1 to denote *present*.

	Variable values for each case		
Case	x_1	x_2	x_3
1	*present*	*absent*	*absent*
2	*present*	*present*	*present*
3	*absent*	*absent*	*present*
4	*present*	*present*	*present*
5	*absent*	*absent*	*absent*
6	*absent*	*present*	*present*
7	*present*	*present*	*present*
8	*absent*	*absent*	*absent*
9	*present*	*present*	*present*
10	*absent*	*absent*	*absent*

Let us consider a simple example of the tasks just described. Suppose the fictitious database of cases in table 1 is the training set. Suppose further that x_1 represents a fault in the system, and that x_2 and x_3 represent two findings. Given the database, what are the qualitative dependency relationships among the variables? For example, do x_1 and x_3 influence each other directly, or do they do so only through x_2? What is the probability that x_3 will be present if x_1 is present? Clearly, there are no categorically correct answers to each of these questions. The answers depend on a number of factors, such as the model that we use to represent the data, and our prior knowledge about the data in the database and the relationships among the variables.

In this paper, we do not attempt to consider all such factors in their full generality. Rather, we specialize the general task by presenting one particular framework for constructing probabilistic networks from databases (as, for example, the database in table 1) such that these networks can be used for probabilistic inference (as, for example, the calculation of $P(x_3 = present | x_1 = present)$). In particular, we focus on using a Bayesian belief network as a model of probabilistic dependency. Our primary goal is to construct such a network (or networks), given a database and a set of explicit assumptions about our prior probabilistic knowledge of the domain.

A Bayesian belief-network *structure* B_S is a directed acyclic graph in which nodes represent domain variables and arcs between nodes represent probabilistic dependencies (Cooper, 1989; Horvitz, Breese, & Henrion, 1988; Lauritzen & Spiegelhalter, 1988; Neapolitan, 1990; Pearl, 1986; Pearl, 1988; Shachter, 1988). A variable in a Bayesian belief-network structure may be continuous (Shachter & Kenley, 1989) or discrete. In this paper, we shall focus our discussion on discrete variables. Figure 1 shows an example of a belief-network structure containing three variables. In this figure, we have drawn an arc from x_1 to x_2 to indicate that these two variables are probabilistically dependent. Similarly, the arc from x_2 to x_3 indicates a probabilistic dependency between these two variables. The absence of an arc from x_1 to x_3 implies that there is no direct probabilistic dependency between x_1 and x_3. In particular, the probability of each value of x_3 is conditionally independent of

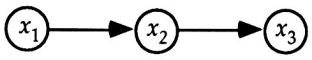

Figure 1. An example of a belief-network structure, which we shall denote as B_{S1}.

the value of x_1 *given* that the value of x_2 is known. The representation of conditional dependencies and independencies is the essential function of belief networks. For a detailed discussion of the semantics of Bayesian belief networks, see (Pearl, 1988).

A Bayesian belief-network structure, B_S, is augmented by conditional probabilities, B_P, to form a Bayesian belief network B. Thus, $B = (B_S, B_P)$. For brevity, we call B a belief network. For each node[1] in a belief-network structure, there is a conditional-probability function that relates this node to its immediate predecessors (parents). We shall use π_i to denote the parent nodes of variable x_i. If a node has no parents, then a prior-probability function, $P(x_i)$, is specified. A set of probabilities is shown in table 2 for the belief-network structure in figure 1. We used the probabilities in table 2 to generate the cases in table 1 by applying Monte Carlo simulation.

We shall use the term *conditional probability* to refer to a probability statement, such as $P(x_2 = present | x_1 = present)$. We use the term *conditional-probability assignment* to denote a numerical assignment to a conditional probability, as, for example, the assignment $P(x_2 = present | x_1 = present) = 0.8$. The network structure B_{S1} in figure 1 and the probabilities B_{P1} in table 2 together define a belief network which we denote as B_1.

Belief networks are capable of representing the probabilities over any discrete sample space: The probability of any sample point in that space can be computed from the probabilities in the belief network. The key feature of belief networks is their explicit representation of the conditional independence and dependence among events. In particular, investigators have shown (Kiiveri, Speed, & Carlin, 1984; Pearl, 1988; Shachter, 1986) that the joint probability of any particular instantiation[2] of all n variables in a belief network can be calculated as follows:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \pi_i), \tag{1}$$

where X_i represents the instantiation of variable x_i and π_i represents the instantiation of the parents of x_i.

Therefore, the joint probability of any instantiation of all the variables in a belief network can be computed as the product of only n probabilities. In principle, we can recover the

Table 2. The probability assignments associated with the belief-network structure B_{S1} in figure 1. We shall denote these probability assignments as B_{P1}.

$P(x_1 = present) = 0.6$	$P(x_1 = absent) = 0.4$
$P(x_2 = present \mid x_1 = present) = 0.8$	$P(x_2 = absent \mid x_1 = present) = 0.2$
$P(x_2 = present \mid x_1 = absent) = 0.3$	$P(x_2 = absent \mid x_1 = absent) = 0.7$
$P(x_3 = present \mid x_2 = present) = 0.9$	$P(x_3 = absent \mid x_2 = present) = 0.1$
$P(x_3 = present \mid x_2 = absent) = 0.15$	$P(x_3 = absent \mid x_2 = absent) = 0.85$

complete joint-probability space from the belief-network representation by calculating the joint probabilities that result from every possible instantiation of the n variables in the network. Thus, we can determine any probability of the form $P(W \mid V)$, where W and V are sets of variables with known values (instantiated variables). For example, for our sample three-node belief network B_1, $P(x_3 = present \mid x_1 = present) = 0.75$.

In the last few years, researchers have made significant progress in formalizing the theory of belief networks (Neapolitan, 1990; Pearl, 1988), and in developing more efficient algorithms for probabilistic inference on belief networks (Henrion, 1990); for some complex networks, however, additional efficiency is still needed. The feasibility of using belief networks in constructing diagnostic systems has been demonstrated in several domains (Agogino & Rege, 1987; Andreassen, Woldbye, Falck, & Andersen, 1987; Beinlich, Suermondt, Chavez, & Cooper, 1989; Chavez & Cooper, 1990; Cooper, 1984; Heckerman, Horvitz, & Nathwani, 1989; Henrion & Cooley, 1987; Holtzman, 1989; Suermondt & Amylon, 1989).

Although researchers have made substantial advances in developing the theory and application of belief networks, the actual construction of these networks often remains a difficult, time-consuming task. The task is time-consuming because typically it must be performed manually by an expert or with the help of an expert. Important progress has been made in developing graphics-based methods that improve the efficiency of knowledge acquisition from experts for construction of belief networks (Heckerman, 1990). These methods are likely to remain important in domains of small to moderate size in which there are readily available experts. Some domains, however, are large. In others, there are few, if any, readily available experts. Methods are needed for augmenting the manual expert-based methods of knowledge acquisition for belief-network construction. In this paper, we present one such method.

The remainder of this paper is organized as follows. In section 2, we present a method for determining the relative probabilities of different belief-network structures, given a database of cases and a set of explicit assumptions. This method is the primary result of the paper. As an example, consider the database in table 1, which we call D. Let B_{S1} denote the belief-network structure in figure 1, and let B_{S2} denote the structure in figure 2. The basic method presented in section 2 allows us to determine the probability of B_{S1} relative to B_{S2}. We show that $P(B_{S1} \mid D)$ is 10 times greater than $P(B_{S2} \mid D)$, under the assumption that B_{S1} and B_{S2} have equal prior probabilities. In section 3, we discuss methods for searching for the most probable belief-network structures, and we introduce techniques for handling missing data and hidden variables. Section 4 describes techniques for employing

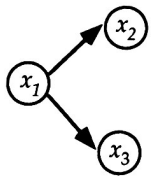

Figure 2. A belief-network structure that is an alternative to the structure in figure 1 for characterizing the probabilistic dependencies among the three variables shown. We shall use B_{S2} to denote this structure.

the methods in section 2 to perform probabilistic inference. In section 5, we report the results of an experiment that evaluates how accurately a 37-node belief network can be reconstructed from a database that was generated from this belief network. Section 6 contains a discussion of previous work. Section 7 concludes the paper with a summary and discussion of open problems.

2. The basic model

Let us now consider the problem of finding the most probable belief-network structure, given a database. Once such a structure is found, we can derive numerical probabilities from the database (we discuss this task in section 4). We can use the resulting belief network for performing probabilistic inference, such as calculating the value of $P(x_3 = \textit{present} | x_1 = \textit{present})$. In addition, the structure may lend insight into the dependency relationships among the variables in the database; for example, it may indicate possible causal relationships.

Let D be a database of cases, Z be the set of variables represented by D, and B_{s_i} and B_{s_j} be two belief-network structures containing exactly those variables that are in Z. In this section, we develop a method for computing $P(B_{s_i} | D)/P(B_{s_j} | D)$. By computing such ratios for pairs of belief-network structures, we can rank order a set of structures by their posterior probabilities. To calculate the ratio of posterior probabilities, we shall calculate $P(B_{S_i}, D)$ and $P(B_{S_j}, D)$ and use the following equivalence:

$$\frac{P(B_{S_i} | D)}{P(B_{S_j} | D)} = \frac{\dfrac{P(B_{S_i}, D)}{P(D)}}{\dfrac{P(B_{S_j}, D)}{P(D)}} = \frac{P(B_{S_i}, D)}{P(B_{S_j}, D)}. \tag{2}$$

Let B_S represent an arbitrary belief-network structure containing just the variables in Z. In section 2.1, we present a method for calculating $P(B_S, D)$. In doing so, we shall introduce several explicit assumptions that render this calculation computationally tractable. A proof of the method for calculating $P(B_S, D)$ is presented in theorem 1 in the appendix.

2.1. A formula for computing $P(B_S, D)$

In this section, we present an efficient formula for computing $P(B_S, D)$. We do so by first introducing four assumptions.

Assumption 1. The database variables, which we denote as Z, are discrete.

As this assumption states, we shall not consider continuous variables in this paper. One way to handle continuous variables is to discretize them; however, we shall not discuss here the issues involved in such a transformation.

A belief network, which consists of a graphical structure plus a set of conditional probabilities, is sufficient to capture any probability distribution over the variables in Z (Pearl, 1988). A belief-network structure alone, containing just the variables in Z, can capture many—but not all—of the independence relationships that might exist in an arbitrary probability distribution over Z (For a detailed discussion, see (Pearl, 1988)).

In this section, we assume that B_S contains just the variables in Z. In section 3.2, we allow B_S to contain variables in addition to those in Z.

The application of assumption 1 yields

$$P(B_S, D) = \int_{B_P} P(D \mid B_S, B_P) f(B_P \mid B_S) P(B_S) dB_P, \tag{3}$$

where B_P is a vector whose values denote the conditional-probability assignments associated with belief-network structure B_S, and f is the conditional-probability density function over B_P given B_S. Note that our assumption of discrete variables leads us to use the probability mass function $P(D \mid B_S, B_P)$ in equation 3, rather than the density function $f(D \mid B_S, B_P)$. The integral in equation (3) is over all possible value assignments to B_P. Thus, we are integrating over all possible belief networks that can have structure B_S. The integral represents a multiple integral and the variables of integration are the conditional probabilities associated with structure B_S.

Example: Consider an example in which B_S is the structure B_{S1} shown in figure 1 and D is the database given by table 1. Let B_P denote an assignment of numerical probability values to a belief network that has structure B_{S1}. Thus, the numerical assignments shown in table 2 constitute one particular value of B_P—call it B_P'. Integrating over all possible B_P corresponds to changing the numbers shown in table 2 in all possible ways that are consistent with the axioms of probability theory. The term $f(B_P' \mid B_{S1})$ denotes the likelihood of the particular numerical probability assignments shown in table 2 for the belief-network structure B_{S1}. The term $P(D \mid B_S, B_P')$ denotes the probability of seeing the data in table 1, given a belief network with structure B_{S1} and with probabilities given by table 2. The term $P(B_{S1})$ is our probability—prior to observing the data in database D—that the data-generating process is a belief network with structure B_{S1}. □

The term $P(B_S)$ in equation (3) can be viewed as one form of *preference bias* (Buntine, 1990a; Mitchell, 1980) for network structure B_S. Utgoff defines a preference bias as "the set of all factors that collectively influence hypothesis selection" (Utgoff, 1986). A computer-based system may use any prior knowledge and methods at its disposal to determine $P(B_S)$. This capability provides considerable flexibility in integrating diverse belief construction methods in artificial intelligence (AI) with the learning method discussed in this paper.

Assumption 2. Cases occur independently, given a belief-network model.

A simple version of assumption 2 occurs in the following, well-known example: If a coin is believed with certainty to be fair (i.e., to have a 0.5 chance of landing heads), then the fact that the first flip landed heads (case 1) does not influence our belief that the second flip (case 2) will land heads.

It follows from the conditional independence of cases expressed in assumption 2 that

$$P(B_S, D) = \int_{B_P} \left[\prod_{h=1}^{m} P(C_h \mid B_S, B_P) \right] f(B_P \mid B_S) P(B_S) dB_P, \tag{4}$$

where m is the number of cases in D and C_h is the hth case in D.

Assumption 3. There are no cases that have variables with missing values.

Assumption 3 generally is not valid for real-world databases, where often there are some missing values. This assumption, however, facilitates the derivation of our basic method for computing $P(B_S, D)$. In section 3.2.1 we discuss methods for relaxing assumption 3 to allow missing data.

Assumption 4. The density function $f(B_P|B_S)$ in equations (3) and (4) is uniform.

This assumption states that, before we observe database D, we are indifferent regarding the numerical probabilities to place on belief-network structure B_S. Thus, for example, it follows for structure B_{S1} in figure 1 that we believe that $P(x_2 = present|x_1 = present)$ is just as likely to have the value 0.3 as to have the value 0.6 (or to have any other real-number value in the interval [0, 1]). In corollary 1 in the appendix, we relax assumption 4 to permit the user to employ Dirichlet distributions to specify prior probabilities on the components of $f(B_P|B_S)$.

We now introduce additional notation that will facilitate our application of the preceding assumptions. We shall represent the parents of X_i as a list (vector) of variables, which we denote as π_i. We shall use w_{ij} to designate the jth unique instantiation of the values of the variables in π_i, relative to the ordering of the cases in D. We say that w_{ij} is a *value* or an *instantiation* of π_i. For example, consider node x_2 in B_{S1} and table 1. Node x_1 is the parent of x_2 in B_{S1}, and therefore $\pi_2 = (x_1)$. In this example, $w_{21} = present$, because in table 1 the first value of x_1 is the value *present*. Furthermore, $w_{22} = absent$, because the second unique value of x_1 in table 1 (relative to the ordering of the cases in that table) is the value *absent*.

Given assumptions 1 through 4, we prove the following result in the appendix.

Theorem 1. Let Z be a set of n discrete variables, where a variable x_i in Z has r_i possible value assignments: $(v_{i1}, \ldots, v_{ir_i})$. Let D be a database of m cases, where each case contains a value assignment for each variable in Z. Let B_S denote a belief-network structure containing just the variables in Z. Each variable x_i in B_S has a set of parents, which we represent with a list of variables π_i. Let w_{ij} denote the jth unique instantiation of π_i relative to D. Suppose there are q_i such unique instantiations of π_i. Define N_{ijk} to be the number of cases in D in which variable x_i has the value v_{ik} and π_i is instantiated as w_{ij}. Let

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

Given assumptions 1 through 4 of this section, it follows that

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{5}$$

□

Example: Applying equation (5) to compute $P(B_{S1}, D)$, given belief-network structure B_{S1} in figure 1 and database D in table 1, yields

$$P(B_{S1}, D) = P(B_{S1}) \frac{(2-1)!\,5!\,5!}{(10+2-1)!} \frac{(2-1)!\,1!\,4!}{(5+2-1)!} \frac{(2-1)!\,4!\,1!}{(5+2-1)!} \frac{(2-1)!\,0!\,5!}{(5+2-1)!} \frac{(2-1)!\,4!\,1!}{(5+2-1)!}$$

$$= P(B_{S1})\ 2.23 \times 10^{-9}.$$

By applying equation (5) for B_{S2} in figure 2, we obtain $P(B_{S2}, D) = P(B_{S2})\ 2.23 \times 10^{-10}$. If we assume that $P(B_{S1}) = P(B_{S2})$, then by equation (2), $P(B_{S1} | D)/P(B_{S2} | D) = 10$. Given the assumptions in this section, the data imply that B_{S1} is 10 times more likely than B_{S2}. This result is not surprising, because we used B_1 to generate D by the application of Monte Carlo sampling. □

2.2. *Time complexity of computing $P(B_S, D)$*

In this section, we derive a worst case time complexity of computing equation (5). In the process, we describe an efficient method for computing that equation. Let r be the maximum number of possible values for any variable, given by $r = \max_{1 \le i \le n}[r_i]$. Define t_{B_S} to be the time required to compute the prior probability of structure B_S. For now, assume that we have determined the values of N_{ijk}, and have stored them in an array. For a given variable x_i the number of unique instantiations of the parents of x_i, given by q_i, is at most m, because there are only m cases in the database. For a given i and j, by definition $N_{ij} = \Sigma_{1 \le k \le r_i} N_{ijk}$, and therefore we can compute N_{ij} in $O(r)$ time. Since there are at most $m\,n$ terms of the form N_{ij}, we can compute all of these terms in $O(m\,n\,r)$ time. Using this result and substituting m for q_i and r for r_i in equation (5), we find that the complexity of computing equation (5) is $O(m\,n\,r + t_{B_S})$, given that the values of N_{ijk} are known.

Now consider the complexity of computing the values of N_{ijk} for a node x_i. For a given x_i, we construct an *index tree* T_i, which we define as follows. Assume that π_i is a list of the parents of x_i. Each branch out of a node at level d in T_i represents a value of the $(d + 1)$th parent of x_i. A path from the root to a leaf corresponds to some instantiation for the parents of x_i. Thus, the depth of the tree is equal to the number of parents of x_i. A given leaf in T_i contains counts for the values of x_i (i.e., for the values $v_{i1}, \ldots, v_{ir_i}$) that are conditioned on the instantiation of the parents of x_i as specified by the path from the root to the leaf. If this path corresponds to the jth unique instantiation of π_i (i.e., $\pi_i = w_{ij}$), then we denote the leaf as l_j. Thus, l_j in T_i corresponds to the list of values of N_{ijk} for $k = 1$ to r_i. We can link the leaves in the tree using a list L_i. Figure 3 shows an

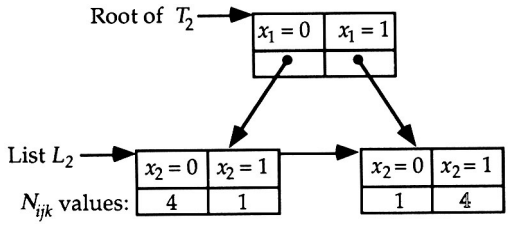

Figure 3. An index tree for node x_2 in structure B_{S1} using the data in table 1. In B_{S1}, x_2 has only one parent—namely, x_1; thus, its index tree has a depth of 1. A 4 is used to highlight an entry that is discussed in the text.

index tree for the node x_2 in B_{S1} using the database in table 1. For example, in table 1, there are four cases in which x_2 is assigned the value *present* (i.e., $x_2 = 1$) and its parent x_1 is assigned the value *present* (i.e., $x_1 = 1$); this situation corresponds to the second column in the second cell of list L_2 in figure 3, which is shown as 4.

Since x_i has at most $n - 1$ parents, the depth of T_i is $O(n)$. Because a variable has at most r values, the size of each node in T_i is $O(r)$. To enter a case into T_i, we must branch on or construct a path that has a total of $O(n)$ nodes, each of size $O(r)$. Thus, a case can be entered in $O(n\ r)$ time. If the database contained every possible case, then T_i would have $O(r^n)$ leaves. However, there are only m cases in the database, so even in the worst case only $O(m)$ leaves will be created. Hence, the time required to construct T_i for node x_i is $O(m\ n\ r)$. Because there are n nodes, the complexity of constructing index trees for all n nodes is $O(m\ n^2\ r)$. The overall complexity of both constructing index trees and using them to compute equation (5) is therefore $O(m\ n^2\ r) + O(m\ n\ r + t_{B_S}) = O(m\ n^2\ r + t_{B_S})$.[3] If the maximum number of parents of any node is u, then the overall complexity is just $O(m\ u\ n\ r + t_{B_S})$, by a straightforward restriction of the previous analysis.[4] If $O(t_{B_S}) = O(u\ n\ r)$, and u and r can be bounded from above by constants, then the overall complexity becomes simply $O(m\ n)$.

2.3. *Computing* $P(B_s|D)$

If we maximize $P(B_S, D)$ over all B_S for the database in table 1, we find that $x_3 \rightarrow x_2 \rightarrow x_1$ is the most likely structure; we shall use B_{S3} to designate this structure. Applying equation (5), we find that $P(B_{S3}, D) = P(B_{S3})\ 2.29 \times 10^{-9}$. If we assume that the database was generated by some belief network containing just the variables in Z, then we can compute $P(D)$ by summing $P(B_S, D)$ over all possible B_S containing just the variables in Z. In the remainder of section 2.3, we shall make this assumption. For the example, there are 25 possible belief-network structures. For simplicity, let us assume that each of these structures is equally likely, a priori. By summing $P(B_S, D)$ over all 25 belief-network structures, we obtain $P(D) = 8.21 \times 10^{-10}$. Therefore, $P(B_{S3}|D) = P(B_{S3}, D)/P(D) = (1/25) \times 2.29 \times 10^{-9}/8.21 \times 10^{-10} = 0.112$. Similarly, we find that $P(B_{S1}|D) = 0.109$, and $P(B_{S2}|D) = 0.011$.

Now we consider the general case. Let Q be the set of all those belief-network structures that contain just the variables in set Z. Then, we have

$$P(B_{S_i} \mid D) = \frac{P(B_{S_i}, D)}{\sum_{B_S \in Q} P(B_S, D)}. \tag{6}$$

As we discuss in section 3.1, the size of Q grows rapidly as a function of the size of Z. Consider, however, the situation in which $\Sigma_{B_S \in Y} P(B_S, D) \approx P(D)$, for some set $Y \subseteq Q$, where $|Y|$ is small. If Y can be located efficiently, then $P(B_{S_i} \mid D)$ can be approximated closely and computed efficiently. An open problem is to develop heuristic methods that attempt to find such a set Y. One approach to computing equation (6) is to use sampling methods to generate a tractable number of belief-network structures and to use these structures to derive an estimate of $P(B_{S_i} \mid D)$.

Let G be a belief-network structure, such that the variables in G are a subset of the variables in Z. Let R be the set of those belief-network structures in Q that contain G as a subgraph. We can calculate the posterior probability of G as follows:

$$P(G \mid D) = \frac{\sum_{B_S \in R} P(B_S, D)}{\sum_{B_S \in Q} P(B_S, D)}. \tag{7}$$

For example, suppose $Z = \{x_1, x_2, x_3\}$, and G is the graph $x_1 \rightarrow x_2$. Then, Q is equal to the 25 possible belief-network structures that contain just the variables in Z, and R is equal to the 8 possible belief-network structures in Q that contain the subgraph $x_1 \rightarrow x_2$. Applying equation (7), we obtain $P(x_1 \rightarrow x_2 \mid D)$, which is the posterior probability that there is an arc from node x_1 to node x_2 in the underlying belief-network process that generated data D (given that the assumptions in section 2.1 hold and that we restrict our model of data generation to belief networks). Probabilities (such as the probability $P(x_1 \rightarrow x_2 \mid D)$) could be used to annotate arcs (such as the arc $x_1 \rightarrow x_2$) to convey to the user the likelihoods of the existences of possible arcs among the variables in Z. Such annotations may be particularly useful for those arcs that have relatively high probabilities. It may be possible to develop efficient heuristic and estimation methods for the computation of equation (7), which are similar to the methods that we mentioned for the computation of equation (6).

When arcs are given a causal interpretation, and specific assumptions are met, we can use previously developed methods to infer causality from data (Pearl & Verma, 1991; Spirtes, Glymour, & Scheines, 1990b). These methods do not, however, annotate each arc with its probability of being true. Thus, the resulting categorical statements of causality that are output by these methods may be invalid, particularly when the database of cases is small. In this context, arc probabilities that are derived from equation (7)—such as $P(x_1 \rightarrow x_2 \mid D)$—can be viewed as providing information about the likelihood of a causal relationship being true, rather than a categorical statement about that relationship's truth.

We also can calculate the posterior probability of an undirected graph. Let G' be an undirected graph, such that the variables in G' are a subset of the variables in Z. Let $R' = \{B_S | B_S$ is in Q, and **if** for distinct nodes x and y in G' there is an edge between x and y in G', **then** it is the case that $x \rightarrow y$ is in B_S or $y \rightarrow x$ is in B_S, **else** it is the case that x and y are not adjacent in $B_S\}$. By replacing R with R' and G with G' in equation (7), we obtain a formula for $P(G' | D)$. Thus, for example, if we use "—" to denote an undirected edge, then $P(x_1 - x_2 | D)$ is the posterior probability that the underlying belief-network process that generated data D contains either an arc from x_1 to x_2 or an arc from x_2 to x_1.

3. Application and extension of the basic model

In this section, we apply the results of section 2 to develop methods that locate the most probable belief-network structures. We also discuss techniques for handling databases that contain missing values and belief-network structures that contain hidden variables.

3.1. Finding the most probable belief-network structures

Consider the problem of determining a belief-network structure B_S that maximizes $P(B_S | D)$. In general, there may be more than one such structure. To simplify our exposition in this section, we shall assume that there is only one maximizing structure; finding the entire set of maximally probable structures is a straightforward generalization. For a given database D, $P(B_S, D) \propto P(B_S | D)$, and therefore finding the B_S that maximizes $P(B_S | D)$ is equivalent to finding the B_S that maximizes $P(B_S, D)$. We can maximize $P(B_S, D)$ by applying equation (5) exhaustively for every possible B_S.

As a function of the number of nodes, the number of possible structures grows exponentially. Thus, an exhaustive enumeration of all network structures is not feasible in most domains. In particular, Robinson (1977) derives the following efficiently computable recursive function for determining the number of possible belief-network structures that contain n nodes:

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n - i).$$

For $n = 2$, the number of possible structures is 3; for $n = 3$, it is 25; for $n = 5$, it is 29,000; and for $n = 10$, it is approximately 4.2×10^{18}. Clearly, we need a method for locating the B_S that maximizes $P(B_S | D)$ that is more efficient than exhaustive enumeration. In section 3.1.1, we introduce additional assumptions and conditions that reduce the time complexity for determining the most probable B_S. The complexity of this task, however, remains exponential. Thus, in section 3.1.2, we modify an algorithm from section 3.1.1 to construct a heuristic method that has polynomial time complexity.

3.1.1. Exact methods

Let us assume, for now, that we can specify an ordering on all n variables, such that, if x_i precedes x_j in the ordering, then we do not allow structures in which there is an arc from x_j to x_i. Given such an ordering as a constraint, there remain $2^{\binom{n}{2}} = 2^{n(n-1)/2}$ possible belief-network structures. For large n, it is not feasible to apply equation 5 for each of $2^{n(n-1)/2}$ possible structures. Therefore, in addition to a node ordering, let us assume equal priors on B_S. That is, initially, before we observe the data D, we believe that all structures are equally likely. In that case, we obtain

$$P(B_S, D) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! , \tag{8}$$

where c is the constant prior probability, $P(B_S)$, for each B_S. To maximize equation (8), we need only to find the parent set of each variable that maximizes the second inner product. Thus, we have that

$$\max_{B_S}[P(B_S, D)] = c \prod_{i=1}^{n} \max_{\pi_i} \left[\prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right] , \tag{9}$$

where the maximization on the right of equation (9) takes place over every instantiation of the parents π_i of x_i that is consistent with the ordering on the nodes.

A node x_i can have at most $n - 1$ nodes as parents. Thus, over all possible B_S consistent with the ordering, x_i can have no more than 2^{n-1} unique sets of parents. Therefore, the maximization on the right of equation (9) occurs over at most 2^{n-1} parent sets. It follows from the results in section 2.2 that the products within the maximization operator in equation (9) can be computed in $O(m\ n\ r)$ time. Therefore, the time complexity of computing equation (9) is $O(m\ n^2\ r\ 2^n)$. If we assume that a node can have at most u parents, then the complexity is only $O(m\ u\ n\ r\ T(n, u))$, where

$$T(n, u) = \sum_{0 \le k \le u} \binom{n}{k} .$$

Let us now consider a generalization of equation (9). Let π_i^S be the parents of x_i in B_S, denoted as $\pi_i^S \rightarrow x_i$. Assume that $P(B_S)$ can be calculated as $P(B_S) = \Pi_{1 \le i \le n} P(\pi_i^S \rightarrow x_i)$. Thus, for all distinct pairs of variables x_i and x_j, our belief about x_i having some set of parents is independent of our belief about x_j having some set of parents. Using this assumption of independence of priors, we can express equation (5) as

$$P(B_S, D) = \prod_{i=1}^{n} P(\pi_i^S \rightarrow x_i) \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! . \tag{10}$$

The probability $P(\pi_i^S \rightarrow x_i)$ could be assessed directly or be derived with additional methods. For example, one method would be to assume that the presence of an arc in $\pi_i^S \rightarrow x_i$ is independent of the presence of the other arcs there; if the probability of each arc in $\pi_i^S \rightarrow x_i$ is specified, we then can compute $P(\pi_i^S \rightarrow x_i)$. Suppose, as before, that we have an ordering on the nodes. Then, from equation (10), we see that

$$\max_{B_S}[P(B_S, D)] = \prod_{i=1}^{n} \max_{\pi_i} \left[P(\pi_i \rightarrow x_i) \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right], \quad (11)$$

where the maximization on the right of equation (11) is taken over all possible sets π_i consistent with the node ordering. The complexity of computing equation (11) is the same as that of computing equation (9), except for an additional term that represents an upper bound on the complexity of computing $P(\pi_i \rightarrow x_i)$. From equation (11), we see that the determination of the most likely belief-network structure is computationally feasible if we assume (1) that there is an ordering on the nodes, (2) that there exists a sufficiently tight limit on the number of parents of any node, and (3) that $P(\pi_i \rightarrow x_i)$ and $P(\pi_j \rightarrow x_j)$ are marginally independent when $i \neq j$, and we can compute such prior probabilities efficiently. Unfortunately, the second assumption in the previous sentence may be particularly difficult to justify in practice. For this reason, we have developed a polynomial-time heuristic algorithm that requires no restriction on the number of parents of a node, although it does permit such a restriction.

3.1.2. A heuristic method

We propose here one heuristic-search method, among many possibilities, for maximizing $P(B_S, D)$. We shall use equation (9) as our starting point, with the attendant assumptions that we have an ordering on the domain variables and that, a priori, all structures are considered equally likely. We shall modify the maximization operation on the right of equation (9) to use a greedy-search method. In particular, we use an algorithm that begins by making the assumption that a node has no parents, and then adds incrementally that parent whose addition most increases the probability of the resulting structure. When the addition of no single parent can increase the probability, we stop adding parents to the node. Researchers have made extensive use of similar greedy-search methods in classification systems—for example, to construct classification trees (Quinlan, 1986) and to perform variable selection (James, 1985).

We shall use the following function:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!, \quad (12)$$

where the N_{ijk} are computed relative to π_i being the parents of x_i and relative to a database D, which we leave implicit. From section 2.2, it follows that $g(i, \pi_i)$ can be computed

in $O(m\ u\ r)$ time, where u is the maximum number of parents that any node is permitted to have, as designated by the user. We also shall use a function $\text{Pred}(x_i)$ that returns the set of nodes that precede x_i in the node ordering. The following pseudocode expresses the heuristic search algorithm, which we call K2.[5]

```
1. procedure K2;
2. {Input: A set of n nodes, an ordering on the nodes, an upper bound u on the
3.         number of parents a node may have, and a database D containing m cases.}
4. {Output: For each node, a printout of the parents of the node.}
5. for i := 1 to n do
6.     π_i := ∅;
7.     P_old := g(i, π_i); {This function is computed using equation (12).}
8.     OKToProceed := true
9.     while OKToProceed and |π_i| < u do
10.        let z be the node in Pred(x_i) − π_i that maximizes g(i, π_i ∪ {z});
11.        P_new := g(i, π_i ∪ {z});
12.        if P_new > P_old then
13.            P_old := P_new;
14.            π_i := π_i ∪ {z}
15.        else OKToProceed := false;
16.    end {while};
17.    write('Node:', x_i, 'Parents of this node:', π_i)
18. end {for};
19. end {K2};
```

We now analyze the time complexity of K2. We shall assume that the factorials that are required to compute equation (12) have been precomputed and have been stored in an array. Equation (12) contains no factorial greater than $(m + r - 1)!$, because N_{ij} can have a value no greater than m. We can compute and store the factorials of the integers from 1 to $(m + r - 1)$ in $O(m + r - 1)$ time. A given execution of line 10 of the K2 procedure requires that g be called at most $n - 1$ times, because x_i has at most $n - 1$ predecessors in the ordering. Since each call to g requires $O(m\ u\ r)$ time, line 10 requires $O(m\ u\ n\ r)$ time. The other statements in the **while** statement require $O(1)$ time. Each time the **while** statement is entered, it loops $O(u)$ times. The **for** statement loops n times. Combining these results, the overall complexity of K2 is $O(m + r - 1) + O(m\ u\ n\ r)\ O(u)\ n = O(m\ u^2\ n^2\ r)$. In the worst case, $u = n$, and the complexity of K2 is $O(m\ n^4\ r)$.

We can improve the run-time speed of K2 by replacing $g(i\ ,\ \pi_i)$ and $g(i,\ \pi_i \cup \{z\})$ by $\log(g(i\ ,\ \pi_i))$ and $\log(g(i,\ \pi_i \cup \{z\}))$, respectively. Run-time savings result because the logarithmic version of equation (12) requires only addition and subtraction, rather than multiplication and division. If the logarithmic version of equation (12) is used in K2, then the logarithms of factorials should be precomputed and should be stored in an array.

We emphasize that K2 is just one of many possible methods for searching the space of belief networks to maximize the probability metric given by equation (5). Accordingly, theorem 1 and equation (5) represent more fundamental results than does the K2 algorithm. Nonetheless, K2 has proved valuable as an initial search method for obtaining preliminary

test results, which we shall describe in section 5. An open research problem is to explore other search methods. For example, consider an algorithm that differs from K2 only in that it begins with a fully connected belief-network structure (relative to a given node order) and performs a greedy search by removing arcs; call this algorithm K2R (K2 Reverse). We might apply K2 to obtain a belief-network structure, then apply K2R to obtain another structure, and finally report whichever structure is more probable according to equation (5). Another method of search is to generate multiple random node orders, to apply K2 using each node order, and to report which among the belief-network structures output by K2 is most probable. Other search techniques that may prove useful include methods that use beam search, branch-and-bound techniques, and simulated annealing.

3.2. Missing data and hidden variables

In this section, we introduce normative methods for handling missing data and hidden variables in the induction of belief networks from databases. These two methods are fundamentally the same. As we present them, neither method is efficient enough to be practical in most real-world applications. We introduce them here for two reasons. First, they demonstrate that the Bayesian approach developed in this paper admits conceptually simple and theoretically sound methods for handling the difficult problems of missing data and hidden variables. Second, these methods establish a theoretical basis from which it may be possible to develop more efficient approaches to these two problems. Without such a theoretical basis, it may be difficult to develop sound methods for addressing the problems pragmatically.

3.2.1. Missing data

In this section, we consider cases in database D that may contain missing values for some variables. Let C_h denote the set of variable assignments for those variables in the hth case that have known values and let C_h' denote the set of variables in the case that have missing values. The probability of the hth case can be computed as

$$P(C_h | B_S, B_P) = \sum_{C_h'} P(C_h, C_h' | B_S, B_P), \tag{13}$$

where $\Sigma_{C_h'}$ means that all the variables in C_h' are running through all their possible values.

By substituting equation (13) into equation (4), we obtain

$$P(B_S, D) = \int_{B_P} \left[\prod_{h=1}^{m} \left[\sum_{C_h'} P(C_h, C_h' | B_S, B_P) \right] \right] f(B_P | B_S) P(B_S) dB_P. \tag{14}$$

To facilitate the next step of the derivation, we now introduce additional notation to describe the value assignments of variables. Let x_i be an arbitrary variable in C_h' or C_h. We shall write a value assignment of x_i as $x_i = d_{ih}$, where d_{ih} is the value of x_i in case h. For a

variable x_i in C_h', d_{ih} is not known, because x_i is a variable with a missing value. The sum in equation (13) means that for each variable x_i in C_h' we have d_{ih} assume each value that is possible for x_i. The overall effect is the same as stated previously for equation (13).

As an example, consider a database containing three binary variables that each have *present* or *absent* as a possible value. Suppose in case 7 that variable x_1 has the value *present* and the values of variables x_2 and x_3 are not known. In this example, $C_7 = \{x_1 = present\}$, and $C_7' = \{x_2 = d_{27}, x_3 = d_{37}\}$. For case 7, equation (13) states that the sum is taken over the following four joint substitutions of values for d_{27} and d_{37}: $\{d_{27} \leftarrow absent, d_{37} \leftarrow absent\}$, $\{d_{27} \leftarrow absent, d_{37} \leftarrow present\}$, $\{d_{27} \leftarrow present, d_{37} \leftarrow absent\}$, and $\{d_{27} \leftarrow present, d_{37} \leftarrow present\}$. For each such joint substitution, we evaluate the probability within the sum of equation (13).

The reason we introduced the d_{ih} notation is that it allows us to assign case-specific values to variables with missing values. We need this ability in order to move the summation in equation (14) to the outside of the integral. In particular, we now can rearrange equation (14) as follows:

$$P(B_S, D) = \sum_{C_1'} \cdots \sum_{C_m'} \int_{B_P} \left[\prod_{h=1}^{m} P(C_h, C_h' | B_S, B_P) \right] f(B_P | B_S) P(B_S) dB_P. \quad (15)$$

Equation 15 is a sum of the type of integrals represented by equation (4), which we solved using equation (5). Thus, equation (15) can be solved by multiple applications of equation (5).

The complexity of computing equation (15) is exponential in the number of missing values in the database. As stated previously, this level of complexity is not computationally tractable for most real-world applications. Equation 15 does, however, provide us with a theoretical starting point for seeking efficient approximation and special-case algorithms, and we are pursuing the development of such algorithms. Meanwhile, we are using a more efficient approach for handling missing data. In particular, if a variable in a case has a missing value, then we give it the value U (for unknown). Thus, for example, a binary variable could be instantiated to one of three values: *absent*, *present*, or U. Other approaches are possible, including those that compute estimates of the missing values and use these estimates to fill in the values.

Example: Suppose that our database D is limited to the first two cases in table 1, and that the value of x_2 in the first case is missing. Let us calculate $P(B_{S1}, D)$. Applying equation (14), we have

$$P(B_{S1}, D) = \int_{B_P} [P(x_1 = 1, x_2 = 0, x_3 = 0 | B_{S1}, B_P) + P(x_1 = 1, x_2 = 1,$$

$$x_3 = 0 | B_{S1}, B_P)] \times P(x_1 = 1, x_2 = 1, x_3 = 1 | B_{S1}, B_P) f(B_P | B_{S1}) P(B_{S1}) dB_P$$

which, by equation (15), is equal to

$$\int_{B_P} P(x_1 = 1, x_2 = 0, x_3 = 0 | B_{S1}, B_P) P(x_1 = 1, x_2 = 1, x_3 = 1 | B_{S1}, B_P) f(B_P | B_{S1}) P(B_{S1}) dB_P$$

$$+ \int_{B_P} P(x_1 = 1, x_2 = 1, x_3 = 0 | B_{S1}, B_P) P(x_1 = 1, x_2 = 1, x_3 = 1 | B_{S1}, B_P) f(B_P | B_{S1}) P(B_{S1}) dB_P.$$

Each of these last two integrals can be solved by the application of equation (5).

3.2.2. *Hidden variables*

A hidden (latent) variable represents a postulated entity about which we have no data. For example, we may wish to postulate the existence of a hidden variable if we are looking for a hidden causal factor that influences the production of the data that we *do* observe. We can handle a hidden variable (or variables) by applying equation (15), where the hidden variable is assigned a missing value for *each* case in the database. In a belief-network structure, the hidden variable is represented as a single node, just as is any other variable.

Example: Assume the availability of the database shown in table 3, which we shall denote as D.

Suppose that we wish to know $P(B_{S2}, D)$, where B_{S2} is the network structure shown in figure 2. Note that, relative to D, x_1 is a hidden variable, because D contains no data about x_1. Let us assume for this example that x_1 is a binary variable. Applying equation (15), we obtain the following result:

$$P(B_{S2}, D) =$$

$$\int_{B_P} P(x_1 = 0, x_2 = 0, x_3 = 0 | B_{S2}, B_P)\, P(x_1 = 0, x_2 = 1, x_3 = 1 | B_{S2}, B_P) f(B_P | B_{S2}) P(B_{S2}) dB_P$$

$$+ \int_{B_P} P(x_1 = 0, x_2 = 0, x_3 = 0 | B_{S2}, B_P)\, P(x_1 = 1, x_2 = 1, x_3 = 1 | B_{S2}, B_P) f(B_P | B_{S2}) P(B_{S2}) dB_P$$

$$+ \int_{B_P} P(x_1 = 1, x_2 = 0, x_3 = 0 | B_{S2}, B_P)\, P(x_1 = 0, x_2 = 1, x_3 = 1 | B_{S2}, B_P) f(B_P | B_{S2}) P(B_{S2}) dB_P$$

$$+ \int_{B_P} P(x_1 = 1, x_2 = 0, x_3 = 0 | B_{S2}, B_P)\, P(x_1 = 1, x_2 = 1, x_3 = 1 | B_{S2}, B_P) f(B_P | B_{S2}) P(B_{S2}) dB_P.$$

Each of these four integrals can be solved by application of equation (5). □

Table 3. The database for the hidden variable example.

Case	x_2	x_3
1	*absent*	*absent*
2	*present*	*present*

One difficulty in considering the possibility of hidden variables is that there is an unlimited number of them and thus an unlimited number of belief-network structures that can contain them. There are many possible approaches to this problem; we shall outline here the approaches that we believe are particularly promising. One way to avoid the problem is simply to limit the number of hidden variables in the belief networks that we postulate. Another approach is to specify explicitly nonzero priors for only a limited number of belief-network structures that contain hidden variables. In addition, we may be able to use statistical indicators that suggest probable hidden variables, as discussed in (Pearl & Verma, 1991; Spirtes & Glymour, 1990; Spirtes et al., 1990b; Verma & Pearl, 1990); we then could limit ourselves to postulating hidden variables only where these indicators suggest that hidden variables may exist.

A related problem is to determine the number of values to define for a hidden variable. One approach is to try different numbers of values. That is, we make the number of values of each hidden variable be a parameter in the search space of belief-network structures. We note that some types of unsupervised learning have close parallels to discovering the number of values to assign to hidden variables. For example, researchers have successfully applied unsupervised Bayesian learning methods to determine the most probable number of values of a single, hidden classification variable (Cheeseman, Self, Kelly, Taylor, Freeman, & Stutz, 1988). We believe that similar methods may prove useful in addressing the problem of learning the number of values of hidden variables in belief networks.

4. Expectations of probabilities

The previous sections concentrated on belief-network structures. In this section, we focus on deriving numerical probabilities when given a database and a belief-network structure (or structures). In particular, we shall focus on determining the expectation of probabilities.

4.1. Expectations of network conditional probabilities

Let θ_{ijk} denote the conditional probability $P(x_i = v_{ik} | \pi_i = w_{ij})$—that is, the probability that x_i has value v_{ik}, for some k from 1 to r_i, given that the parents of x_i, represented by π_i, are instantiated as w_{ij}. Call θ_{ijk} a *network conditional probability.* Let ξ denote the four assumptions in section 2.1. Consider the value of $E[\theta_{ijk} | D, B_S, \xi]$, which is the expected

value of θ_{ijk} given database D, the belief-network structure B_S, and the assumptions ξ. In theorem 2 in the appendix, we derive the following result:

$$E[\theta_{ijk} | D, B_S, \xi] = \frac{N_{ijk} + 1}{N_{ij} + r_i}. \tag{16}$$

In corollary 2 in the appendix, we derive a more general version of $E[\theta_{ijk} | D, B_S, \xi]$ by relaxing assumption 4 in section 2.1 to allow the user to express prior probabilities on the values of network conditional probabilities. $E[\theta_{ijk} | D, B_S, \xi]$ is sometimes called the *Bayes' estimator* of θ_{ijk}. The value of $E[\theta_{ijk} | D, B_S, \xi]$ in equation (16) is equal to the expectation of θ_{ijk} as calculated using a uniform probability distribution and using the data in D (deGroot, 1970). We note that Spiegelhalter and Lauritzen (1990) also have used such expectations in their work on updating belief-network conditional probabilities.

By applying an analogous analysis for variance, we can show that (Wilks, 1962)

$$\text{Var}[\theta_{ijk} | D, B_S, \xi] = \frac{(N_{ijk} + 1)(N_{ij} + r_i - N_{ijk} - 1)}{(N_{ij} + r_i)^2 (N_{ij} + r_i + 1)}. \tag{17}$$

Example: Consider the probability $P(x_2 = present | x_1 = present)$ for belief-network structure B_{S1}. Let θ_{212} represent $P(x_2 = present | x_1 = present)$. We now wish to determine $E[\theta_{212} | D, B_S, \xi]$ and $\text{Var}[\theta_{212} | D, B_S, \xi]$, where D is the database in table 1. Since x_2 is a binary variable, $r_2 = 2$. There are five cases in D in which $x_1 = present$ and therefore, $N_{21} = 5$. Of these five, there are four cases in which $x_1 = present$ and $x_2 = present$, and, thus, $N_{212} = 4$. Substituting these values into equations (16) and (17), we obtain $E[\theta_{212} | D, B_S, \xi] = 0.71$ and $\text{Var}[\theta_{212} | D, B_S, \xi] = 0.03$. □

4.2. *Expectations of general conditional probabilities given a network structure*

A common application of a belief network is to determine $E[P(W_1 | W_2)]$, where W_1 and W_2 are sets of instantiated variables. For example, W_1 might be a disease state and W_2 a set of symptoms. Consider a decision that depends on just the likelihood of W_1, given that W_2 is known. Researchers have shown that $E[P(W_1 | W_2)]$ provides sufficient information to determine the optimal decision to make within a decision-theoretic framework, as long as the decision must be made without the benefit of additional information (Howard, 1988). Thus, in many situations, knowledge of $E[P(W_1 | W_2)]$ is sufficient for decision making.

Since, in this paper we are constructing belief networks based on a database D, we wish to know $E[P(W_1 | W_2) | D, B_S, \xi]$. In (Cooper & Herskovits, 1991), we derive the following equation:

$$E[P(W_1 | W_2) | D, B_S, \xi] = P(W_1 | W_2), \tag{18}$$

where $P(W_1 | W_2)$ is computed with a belief network that uses the probabilities given by equation (16).

4.3. Expectations of general conditional probabilities over all network structures

On the right side of equation (18), D, B_S and ξ are implicit conditioning information. To be more explicit, we can rewrite that equation as

$$E[P(W_1 \mid W_2) \mid D, B_S, \xi] = P(W_1 \mid W_2, D, B_S, \xi) \tag{19}$$

where $P(W_1 \mid W_2, D, B_S, \xi)$ may be calculated as $P(W_1 \mid W_2)$ using a belief network with a structure B_S and with conditional probabilities that are derived using equation (16). For optimal decision making, however, we actually wish to know $E[P(W_1 \mid W_2) \mid D, \xi]$, rather than $E[P(W_1 \mid W_2) \mid D, B_S, \xi]$ for some particular B_S about which we are uncertain. We can derive $E[P(W_1 \mid W_2) \mid D, \xi]$ as

$$E[P(W_1 \mid W_2) \mid D, \xi] = \sum_{B_S} E[P(W_1 \mid W_2) \mid D, B_S, \xi] \, P(B_S \mid W_2, D \, \xi),$$

which, by equation (19), becomes

$$E[P(W_1 \mid W_2) \mid D, \xi] = \sum_{B_S} P(W_1 \mid W_2, D, B_S, \xi) \, P(B_S \mid W_2, D \, \xi). \tag{20}$$

The probability $P(B_S \mid W_2, D, \xi)$ is interesting because it contains W_2 as conditioning information. We can view W_2 as additional data that augment D. If D is large, we may choose to approximate $P(B_S \mid W_2, D, \xi)$ as $P(B_S \mid D, \xi)$. Alternatively, we may choose to assume that W_2 provides no additional information about B_S, and therefore that $P(B_S \mid W_2, D, \xi) = P(B_S \mid D, \xi)$. Otherwise, we must treat W_2 as an additional case in the database. Typically, W_2 will represent an incomplete case in which some model variables have unknown values. In this situation, the techniques we discuss in section 3.2.1 for handling missing data can be used to compute $P(B_S \mid W_2, D, \xi)$.

Although it is not computationally feasible to calculate equation (20) for models with more than a few variables, this equation provides a theoretical framework for seeking rapid and accurate special-case, approximate and heuristic solutions. For example, techniques—such as those discussed in the final paragraph of section 3.1—might be used in searching for belief-network structures that yield relatively high values for $P(B_S \mid W_2, D, \xi)$. If we normalize over this set of structures, we can apply equation (20) to estimate heuristically the value of $E[P(W_1 \mid W_2) \mid D, \xi]$. Another possible approach toward estimating $E[P(W_1 \mid W_2) \mid D, \xi]$ is to apply sampling techniques that use stochastic simulation.

Example: Suppose we wish to know $P(x_2 = \textit{present} \mid x_1 = \textit{present})$ given database D, which is shown in table 4.

Let us compute $P(x_2 = \textit{present} \mid x_1 = \textit{present})$ by using equation (20) and the assumption that $P(B_S \mid x_1 = \textit{present}, D, \xi) = P(B_S \mid D, \xi)$. For simplicity, we abbreviate $P(x_2 = \textit{present} \mid x_1 = \textit{present})$ as $P(x_2 \mid x_1)$, leaving the values of x_1 and x_2 implicit. We shall enclose network structures in braces for clarity; so, for example, $\{x_1 \rightarrow x_2\}$ means that

Table 4. The database used in the example of the application of equation (20).

Case	x_1	x_2
1	*present*	*present*
2	*present*	*present*
3	*present*	*present*
4	*absent*	*present*
5	*absent*	*absent*

x_1 is the parent of x_2. Given a model with two variables, there are only three possible belief-network structures—namely, $\{x_1 \rightarrow x_2\}$, $\{x_2 \rightarrow x_1\}$, and $\{x_1 \quad x_2\}$. Thus, by equation (20)

$$\begin{aligned} E[P(x_2|x_1)|D, \xi] &= P(x_2|x_1, D, \{x_1 \rightarrow x_2\}, \xi)P(\{x_1 \rightarrow x_2\}|D, \xi) \\ &\quad + P(x_2|x_1, D, \{x_2 \rightarrow x_1\}, \xi)P(\{x_2 \rightarrow x_1\}|D, \xi) \\ &\quad + P(x_2|x_1, D, \{x_1 \quad x_2\}, \xi)P(\{x_1 \quad x_2\}|D, \xi) \\ &= 0.80 \times 0.33 + 0.83 \times 0.40 + 0.71 \times 0.27 = 0.79, \end{aligned}$$

where (1) the probabilities 0.80, 0.83, and 0.71 were computed with the three respective belief networks that each contain network conditional probabilities derived using equation (16), and (2) the probabilities 0.33, 0.40, and 0.27 were computed using the methods discussed in section 2.3.

5. Preliminary results

In this section, we describe an experiment in which we generated a database from a belief network by simulation, and then attempted to reconstruct the belief network from the database. In particular, we applied the K2 algorithm discussed in section 3.1.2 to a database of 10,000 cases generated from the ALARM belief network, which has the structure shown in figure 4. Beinlich constructed the ALARM network as an initial research prototype to model potential anesthesia problems in the operating room (Beinlich et al., 1989). To keep figure 4 uncluttered, we have replaced the node names in ALARM with the numbers shown in the figure. For example, node 20 represents that *the patient is receiving insufficient anesthesia or analgesia*, node 27 represents *an increased release of adrenaline by the patient*, node 29 represents *an increased patient heart rate*, and node 8 represents that *the EKG is measuring an increased patient heart rate*. When ALARM is given input findings—such as heart rate measurements—it outputs a probability distribution over a set of possible problems—such as insufficient anesthesia. ALARM represents 8 diagnostic problems, 16 findings, and 13 intermediate variables that connect diagnostic problems to findings. ALARM contains a total of 46 arcs and 37 nodes, and each node has from two to four possible values. Knowledge for constructing ALARM came from Beinlich's reading of the literature and

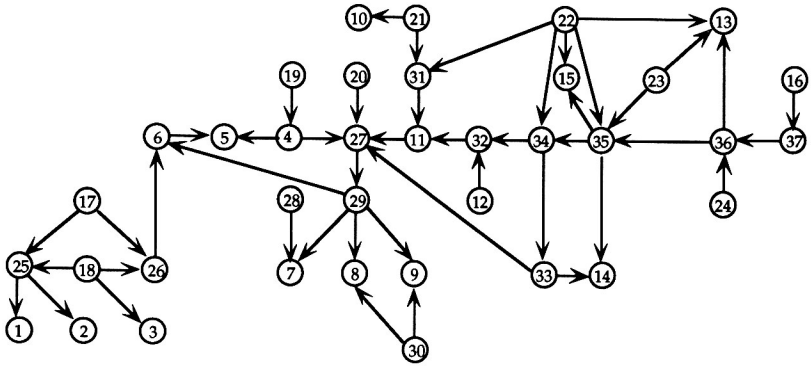

Figure 4. The ALARM belief-network structure, containing 37 nodes and 46 arcs.

from his own experience as an anesthesiologist. It took Beinlich approximately 10 hours to construct the ALARM belief-network structure, and about 20 hours to fill in all the corresponding probability tables.

We generated cases from ALARM by using a Monte Carlo technique developed by Henrion for belief networks (Henrion, 1988). Each case corresponds to a value assignment for each of the 37 variables. The Monte Carlo technique is an unbiased generator of cases, in the sense that the probability that a particular case is generated is equal to the probability of the case existing according to the belief network. We generated 10,000 such cases to create a database that we used as input to the K2 algorithm. We also supplied K2 with an ordering on the 37 nodes that is consistent with the partial order of the nodes as specified by ALARM. Thus, for example, node 21 necessarily appears in the ordering before node 10, but it is not necessary for node 21 to appear immediately before node 10 in the ordering. Observing this ordering constraint, we manually generated a node order using the ALARM structure.[6] In particular, we added a node to the node-order list only when all of that node's parents were already in the list. During the process of constructing this node order, we did not consider the meanings of the nodes.

From the 10,000 cases, the K2 algorithm constructed a network identical to the ALARM network, except that the arc from node 12 to node 32 was missing and an arc from node 15 to node 34 was added. A subsequent analysis revealed that the arc from node 12 to node 32 is not strongly supported by the 10,000 cases. The extra arc from node 15 to node 34 was added due to the greedy nature of the K2 search algorithm. The total search time for the reconstruction was approximately 16 minutes and 38 seconds on a Macintosh II running LightSpeed Pascal, Version 2.0. We analyzed the performance of K2 when given the first 100, 200, 500, 1000, 2000 and 3000 cases from the same 10,000-case database. The results of applying K2 to these databases are summarized in table 5. Using only 3000 cases, K2 produced the same belief network that it created using the full 10,000 cases.

Although preliminary, these results are encouraging because they demonstrate that K2 can reconstruct a moderately complex belief network rapidly from a set of cases using readily available computer hardware. (For the results of K2 applied to databases from other domains, see (Herskovits, 1991).) We plan to investigate the extent to which the performance

Table 5. The results of applying K2 with subsets of the 10,000 ALARM cases.

100	5	33	19
200	4	19	29
500	2	7	55
1,000	1	5	108
2,000	1	3	204
3,000	1	1	297
10,000	1	1	998

of K2 is sensitive to the ordering of the nodes in ALARM and in other domains. In addition, we plan to explore methods that do not require an ordering.

6. Related work

In sections 2 through 5, we described a Bayesian approach to learning the qualitative and quantitative dependency relationships among a set of discrete variables. For notational simplicity, we shall call the approach BLN (*B*ayesian *l*earning of belief *n*etworks). Many diverse methods for automated learning from data have been developed in fields such as statistics (Glymour, Scheines, Spirtes, & Kelley, 1987; James, 1985; Johnson & Wichern, 1982) and AI (Blum, 1982; Carbonell, 1990; Hinton, 1990; Michalski, Carbonell, & Mitchell, 1983; Michalski, Carbonell, & Mitchell, 1986). Since it is impractical to survey all these methods, we shall restrict our review to representative methods that we believe are closest to BLN. We group methods into several classes to organize our discussion, but acknowledge that this classification is not absolute and that some methods may cross boundaries.

6.1. Methods based on probabilistic-graph models

In this section, we discuss three classes of techniques for constructing probabilistic-graph models from databases.

6.1.1. Belief-network methods

Chow and Liu (1968) developed a method that constructs a tree-structured Markov graph, which we shall call simply a *tree*, from a database of discrete variables. If the data are being generated by an underlying distribution P that can be represented as a tree, then the Chow-Liu algorithm constructs a tree with a probability distribution that converges to P as the size of the database increases. If the data are not generated by a tree, then the algorithm constructs the tree that most closely approximates the underlying distribution P (in the sense of cross-entropy).

A polytree (singly connected network) is a belief network that contains at most one undirected path (i.e., a path that ignores the direction of arcs) between any two nodes in the network. Rebane and Pearl (1987) used the Chow-Liu algorithm as the basis for an algorithm

that recovers polytrees from a probability distribution. In cases where the orientation of an arc cannot be determined from the distribution, an undirected edge is used. In determining the orientation of arcs, the Rebane-Pearl algorithm assumes the availability of a conditional-independence (CI) test—a test that determines categorically whether the following conditional independence relation is true or false: Variables in a set X are independent of variables in a set Y, given that the variables in a set Z are instantiated. In degenerate cases, the algorithm may not return the structure of the underlying belief network. In addition, for a probability distribution P that cannot be represented by a polytree, the algorithm is not guaranteed to construct the polytree that most closely approximates P (in the sense of cross-entropy). An algorithm by Geiger, Paz, and Pearl (1990) generalizes the Rebane-Pearl algorithm to recover polytrees by using less restrictive assumptions about the distribution P.

Several algorithms have been developed that use a CI test to recover a *multiply connected belief network*, which is a belief network containing at least one pair of nodes that have at least two undirected paths between them. All such algorithms we describe here run in time that is exponential in the number of nodes in the worst case. Wermuth and Lauritzen (1983) describe a method that takes as input an ordering on all model nodes and then applies a CI test to a distribution to construct a belief network that is a minimal I-map.[7] Srinivas, Russell, and Agogino (1990) allow the user to specify a weaker set of constraints on the ordering of nodes, and then use a heuristic algorithm to search for a belief network I-map (possibly nonminimal).

Spirtes, Glymour, and Scheines (1990b) developed an algorithm that does not require a node ordering in order to recover multiply connected belief networks. Verma and Pearl (1990) subsequently presented a related algorithm, which we now shall describe. The algorithm first constructs an undirected adjacency graph among the nodes. Then, it orients edges in the graph, when this step is possible given the probability distribution. The method assumes that there is some belief-network structure that can represent all the dependencies and independencies among the variables in the underlying probability distribution that generated the data. There are, however, probability distributions for which this assumption is not valid. Verma and Pearl also introduce a method for detecting the presence of hidden variables, given a distribution over a set of measured variables. They further suggest an information-theoretic measure as the basis for a CI test. The CI test, however, requires determining a number of independence relations that is on the order of $n - 2$. Such tests may be unreliable, unless the volume of data is enormous.

Spirtes, Glymour, and Scheines (1991) have developed an algorithm, called PC, that, for graphs with a sparse number of edges, permits reliable testing of independence using a relatively small number of data. PC does not require a node ordering. For dense graphs with limited data, however, the test may be unreliable. For discrete data, the PC algorithm uses a CI test that is based on the chi-square distribution with a fixed alpha level. Spirtes and colleagues applied PC with the 10,000 ALARM cases discussed in section 5. PC reconstructed ALARM, except that three arcs were missing and two extra arcs were added; the algorithm required about 6 minutes of computer time on a DecStation 3100 to perform this task (Spirtes, Glymour, & Scheines, 1990a).

6.1.2. Markov graph methods

Fung and Crawford (1990) have developed an algorithm called Constructor that constructs an undirected graph by performing a search to find the Markov boundary of each node. The algorithm uses a chi-squared statistic as a CI test. In general, the smaller the Markov boundary of the nodes, the more reliable the CI test statistic. For nodes with large Markov boundaries, the test can be unreliable, unless there is a large number of data. A probability distribution for the resulting undirected graph is estimated from the database. The method of Lauritzen and Spiegelhalter (1988) then is applied to perform probabilistic inference using the undirected graph. An interesting characteristic of Constructor is that it pretunes the CI test statistic. In particular, instead of assuming a fixed alpha level for the test statistic, the algorithm searches for a level that maximizes classification accuracy on a test subset of cases in the database. Constructor has been applied successfully to build a belief network that performs information retrieval (Fung, Crawford, Appelbaum, & Tong, 1990).

6.1.3. Entropy-based methods

In the field of system science, the reconstruction problem focuses on constructing from a database an undirected adjacency graph that captures node dependencies (Pittarelli, 1990). Intuitively, the idea is to find the smallest graph that permits the accurate representation of a given probability distribution. The adequacy of a graph often is determined using entropy as a measure of information content. Since the number of possible graphs typically is enormous, heuristics are necessary to render search tractable. For example, one reconstruction algorithm searches for an adjacency graph by starting with a fully connected graph. The search is terminated when there is no edge that can be removed from the current graph G_1 to form a graph G_2, such that the information loss in going from G_1 and G_2 is below a set threshold. In this case, G_1 is output as the dependency graph.

The Kutató algorithm, which is described in (Herskovits, 1991; Herskovits & Cooper, 1990), shares some similarities with the system-science reconstruction algorithms. In particular, Kutató uses an entropy measure and greedy search to construct a model. One key difference, however, is that Kutató constructs a belief network rather than an undirected graph. The algorithm starts with no arcs and adds arcs until a halting condition is reached. Using the 10,000 cases generated from the ALARM belief network discussed in section 5, Kutató reconstructed ALARM, except that two arcs were missing and two extra arcs were added. The reconstruction required approximately 22.5 hours of computer time on a Macintosh II computer. For a detailed analysis of the relationship between entropy-based algorithms such as Kutató, and Bayesian algorithms such as K2, see (Herskovits, 1991).

An algorithm developed by Cheeseman (1983) and extended by Gevarter (1986) implicitly searches for a model of undirected edges in the form of variable constraints. The algorithm adds constraints incrementally to a growing model. If the maximum-entropy distribution of models containing constraints of order $n + 1$ is not significantly different from that of models containing constraints of order n, then the search is halted. Otherwise, constraints of order $n + 1$ are added until no significant difference exists; then, constraints of order $n + 2$ are considered, and so on.

6.2. Classification trees

Another class of algorithms constructs classification trees[8] from databases (Breiman, Friedman, Olshen, & Stone, 1984; Buntine, 1990b; Hunt, Marin, & Stone, 1966; Quinlan, 1986). In its most basic form, a classification tree is a rooted binary tree, where each pair of branches out of a node corresponds to two disjoint values (or value ranges) of a domain variable (attribute). A leaf node corresponds to a classification category or to a probability distribution over the possible categories. We can apply a classification tree by using known attribute values to traverse a path down the tree to a leaf node. In constructing a classification tree, the typical goal is to build the single tree that maximizes expected classification accuracy on new cases. Several criteria, including information-theoretic measures, have been explored for determining how to construct a tree. Typically, a one-step lookahead is used in constructing branch points. In an attempt to avoid overfitting, trees often are pruned by collapsing subtrees into leaves. CART is a well-known method for constructing a classification tree from data (Breiman et al., 1984). CART has been studied in a variety of domains such as signal analysis, medical diagnosis, and mass spectra classification; it has performed well relative to several pattern-recognition methods, including nearest-neighbor algorithms (Breiman et al., 1984).

Buntine (1990b) independently has developed methods for learning and using classification trees that are similar to the methods we discuss for belief networks in this paper. In particular, he has developed Bayesian methods for (1) calculating the probability of a classification-tree structure given a database of cases, and (2) computing the expected value of the probability of a classification instance by using many tree structures (called the option-trees method). Buntine empirically evaluated the classification accuracy of several algorithms on 12 databases from varied domains, including the LED database of Breiman et al. (1984) and the iris database of Fisher. He concluded that "option trees was the only approach that was usually significantly superior to others in accuracy on most data sets" (Buntine, 1990b, page 110).

Kwok and Carter (1990) evaluated a simple version of the option-trees method on two databases. In particular, they averaged the classification results of multiple classification trees on a set of problems. The averaging method usually yielded more accurate classification than did any single tree, including the tree generated by Quinlan's ID3 algorithm (Quinlan, 1986). Averaging over as few as three trees yielded significantly improved classification accuracy. In addition, averaging over trees with different structures produced classification more accurate than that produced by averaging over trees with similar structures.

In the remainder of section 6.2, we present a brief comparison of classification trees and belief networks. For a more detailed discussion, see (Crawford and Fung, 1991). Classification trees can readily handle both discrete and continuous variables. A classification tree is restricted, however, to representing the distribution on one variable of interest—the classification variable. With this constraint, however, classification trees often can represent compactly the attributes that influence the distribution of the classification variable. It is simple and efficient to apply a classification tree to perform classification. For belief networks, there exist approximation and special-case methods for handling continuous variables (Shachter, 1990). Currently, however, the most common way of handling these variables is to discretize them. Belief networks can capture the probabilistic relationships among

multiple variables, without the need to designate a classification variable. These networks provide a natural representation for capturing causal relationships among a set of variables (see (Crawford & Fung, 1991) for a case study). In addition, inference algorithms exist for computing the probability of any subset of variables conditioned on the values of any other subset. In the worst case, however, these inference algorithms have a computational time complexity that is exponential in the size of the belief network. Nonetheless, for networks that are not densely connected, there exist efficient exact inference algorithms (Henrion, 1990). In representing the relationship between a node and its parents, there are certain types of value-specific conditional independencies that cannot be captured easily in a belief network. In some instances, classification trees can represent these independencies efficiently and naturally. Researchers recently have begun to explore extensions to belief networks that capture this type of independence (Fung & Shachter, 1991; Geiger and Heckerman, 1991).

6.3. Methods that handle hidden variables

In the general case, discovering belief networks with hidden variables remains an unsolved problem. Nonetheless, researchers have made progress in developing methods for detecting the presence of hidden variables in some situations (Spirtes & Glymour, 1990; Spirtes et al., 1990b; Verma & Pearl, 1990). Pearl developed a method for constructing from data a tree-structured belief network with hidden variables (Pearl, 1986). Other researchers have developed algorithms that are less sensitive to noise than is Pearl's method, but that still are restricted to tree-structured networks (Golmard & Mallet, 1989; Liu, Wilkins, Yin, & Bian, 1990). The Tetrad program is a semiautomated method for discovering causal relationships among continuous variables (Glymour et al., 1987; Glymour & Spirtes, 1988). Tetrad considers only normal linear models. By making the assumption that linearity holds, the program is able to use an elegant method based on tetrads and partial correlations to introduce likely latent (hidden) variables into causal models; these methods have been evaluated and compared to statistical techniques such as LISREL and EQS (Spirtes, Scheines, & Glymour, 1990c). Researchers have made little progress, however, in developing general nonparametric methods for discovering hidden variables in multiply connected belief networks.

7. Summary and open problems

The BLN approach presented in this paper can represent arbitrary belief-network structures and arbitrary probability distributions on discrete variables. Thus, in terms of its representation, BLN is nearest to the most general probabilistic network approaches discussed in section 6.1.

The BLN learning methodology, however, is closest to the Bayesian classification-tree method discussed in section 6.2. Like that method, BLN calculates the probability of a structure of variable relationships given a database. The probability of multiple structures can be computed and displayed to the user. Like the option-trees method, BLN also can use multiple structures in performing inference, as discussed in section 4.3. The BLN

approach, however, uses a directed acyclic graph on nodes that represent variables rather than a tree on nodes that represent variable values or value ranges. When the number of domain variables is large, the combinatorics of enumerating all possible belief network structures becomes prohibitive. Developing methods for efficiently locating highly probable structures remains an open area of research.

Except for Bayesian classification trees, the methods discussed in section 6 are non-Bayesian. These methods emphasize finding the single most likely structure, which they then may use for inference. They do not, however, quantify the likelihood of that structure. If a single structure is used for inference, implicitly the probability of that structure is assumed to be 1. Section 6.2 discussed results suggesting that using multiple structures may improve the accuracy of classification inference. Also, the non-Bayesian methods rely on having threshold values for determining when conditional independence holds. BLN does not require the use of such thresholds.

BLN is data-driven by the cases in the database and model-driven by prior probabilities. BLN is able to represent the prior probabilities of belief-network structures. In section 2.1 we suggested the possibility that these probabilities may provide one way to bridge BLN to other AI methods. Prior-probability distributions also can be placed on the conditional probabilities of a particular belief network, as we show in corollaries 1 and 2 in the appendix. If the prior-probability distributions on structures and on conditional probabilities are not available to the computer, then uniform priors may be assumed. Additional methods are needed, however, that facilitate the representation and specification of prior probabilities, particularly priors on belief-network structures.

As we discussed in section 6.3, there has been some progress in developing methods for detecting hidden variables, and in the case of some parametric distributions, for searching for a likely model containing hidden variables. BLN can compute the probability of an arbitrary belief-network structure that contains hidden variables and missing data without assuming a parametric distribution. More specifically, no additional assumptions or heuristics are needed for handling hidden variables and missing data in BLN, beyond the assumptions made in section 2.1 for handling known variables and complete data. Additional research is needed, however, for developing ways to search efficiently the vast space of possible hidden-variable networks to locate the most likely networks.

Although BLN shows promise as a method for learning and inference, there remain numerous open problems, several of which we summarize here. For databases that are generated from a belief network, it is important to prove that, as the number of cases in the database increases, BLN converges to the underlying generating network or to a network that is statistically indistinguishable from the generating network. This result has been proved in the special case that we assume a node order (Herskovits, 1991). Proofs of convergence in the presence of hidden variables also are needed. Related problems are to determine the expected number of cases required to recover a generating network and to determine the variance of $P(B_S|D)$. The theoretical and empirical sensitivities of BLN to different types of noisy data need to be investigated as well. Another area of research is Bayesian learning of undirected networks, or, more generally, of mixed directed and undirected networks. Also, recall that the K2 method presented in section 3.1.2 requires an ordering on the nodes. We would like to avoid such a requirement. One approach is to search for likely undirected graphs and to use these as starting points in searching for directed graphs.

Extending BLN to handle continuous variables is another open problem. One approach to this problem is to use Bayesian methods to discretize continuous variables. Finally, regarding evaluation, the results in section 5 are promising, but are limited in scope. Significantly more empirical work is needed to investigate the practicality of the BLN method when applied to databases from different domains.

Acknowledgments

We thank Lyn Dupré, Clark Glymour, the anonymous reviewers, and the Editor for helpful comments on earlier drafts. We also thank Ingo Beinlich for allowing us to use the ALARM belief network. The research reported in this paper was performed in part while the authors were in the Section on Medical Informatics at Stanford University. Support was provided by the National Science Foundation under grants IRI-8703710 and IRI-9111590, by the U.S. Army Research Office under grant P-25514-EL, and by the National Library of Medicine under grant LM-04136. Computing resources were provided in part by the SUMEX-AIM resource under grant LM-05208 from the National Library of Medicine.

Notes

1. Since there is a one-to-one correspondence between a node in B_S and a variable in B_P, we shall use the terms *node* and *variable* interchangeably.
2. An instantiated variable is a variable with an assigned value.
3. If hashing is used to store information equivalent to that in an index tree, then it may be possible to obtain a bound tighter than $O(m\ n^2\ r + t_{B_S})$ for the *average* performance. In the *worst case*, however, due to the collisions of hash keys, an approach that uses hashing may be less efficient than the method described in this section.
4. Binary trees can be used to represent the values of nodes in the index trees we have described. We note, but shall not prove here, that the overall complexity is reduced to $O(m\ n^2 \lg r + t_{B_S})$ if we use such binary trees in computing the values of N_{ijk} and N_{ij}.
5. The algorithm is named K2 because it evolved from a system named Kutató (Herskovits & Cooper, 1990) that applies the same greedy-search heuristics. As we discuss in section 6.1.3, Kutató uses entropy to score network structures.
6. The particular ordering that we used is as follows: 12 16 17 18 19 20 21 22 23 24 25 26 28 30 31 37 1 2 3 4 10 36 13 35 15 34 32 33 11 14 27 29 6 7 8 9 5.
7. A belief network B is an *I-map* of a probability distribution P if every CI relation specified by the structure of B corresponds to a CI relation in P. Further, B is a *minimal I-map* of P if it is an I-map of P and the removal of any arc from B yields a belief network that is not an I-map of P.
8. Classification trees also are known as decision trees, which are different from the decision trees used in decision analysis. To avoid any ambiguity, we shall use the term classification tree.

Appendix

This appendix includes two theorems and two corollaries that are referenced in the paper. The proofs of the theorems are derived in detail. Although this level of detail lengthens the proofs, it avoids our relying on previous results that may not be familiar to some readers. Thus, the proofs are largely self-contained.

Theorem 1. Let Z be a set of n discrete variables, where a variable x_i in Z has r_i possible value assignments: $(v_{i1}, \ldots, v_{ir_i})$. Let D be a database of m cases, where each case contains a value assignment for each variable in Z. Let B_S denote a belief-network structure containing just the variables in Z. Each variable x_i in B_S has a set of parents, which we represent with a list of variables π_i. Let w_{ij} denote the jth unique instantiation of π_i relative to D. Suppose there are q_i such unique instantiations of π_i. Define N_{ijk} to be the number of cases in D in which variable x_i has the value v_{ik} and π_i is instantiated as w_{ij}. Let

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}.$$

Suppose the following assumptions hold:

1. The variables in Z are discrete
2. Cases occur independently, given a belief-network model
3. There are no cases that have variables with missing values
4. Before observing D, we are indifferent regarding which numerical probabilities to assign to the belief network with structure B_S.

From these four assumptions, it follows that

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!.$$

Proof. By applying assumptions 1 through 4, we derive a multiple integral over a product of multinomial variables, which we then solve.

The application of assumption 1 yields

$$P(B_S, D) = \int_{B_P} P(D|B_S, B_P) f(B_P|B_S) P(B_S) dB_P, \tag{A1}$$

where B_P is a vector whose values denote the conditional-probability assignments associated with belief-network structure B_S, and f is the conditional-probability-density function over B_P given B_S. The integral is over all possible value assignments to B_P.

Since $P(B_s)$ is a constant within equation (A1), we can move it outside the integral:

$$P(B_S, D) = P(B_S) \int_{B_P} P(D|B_S, B_P) f(B_P|B_S) dB_P. \tag{A2}$$

It follows from the conditional independence of cases expressed in assumption 2 that equation (A2) can be rewritten as

$$P(B_S, D) = P(B_S) \int_{B_P} \left[\prod_{h=1}^{m} P(C_h | B_S, B_P) \right] f(B_P | B_S) dB_P, \quad \text{(A3)}$$

where m is the number of cases in D, and C_h is the hth case in D.

We now introduce additional notation to facilitate the application of assumption 3. Let d_{ih} denote the value assignment of variable i in case h. For example, for the database in table 1, $d_{21} = 0$, since $x_2 = 0$ in case 1. In B_S, for every variable x_i, there is a set of parents π_i (possibly the empty set). For each case in D, the variables in the list π_i are each assigned a particular value. Let w_i denote a list of the unique instantiations for the parents of x_i as seen in D. An element in w_i designates a list of values that are assigned to the respective variables in the list π_i. If x_i has no parents, then we define w_i to be the list (Ø), where Ø represents the empty set of parents. Although the ordering of the elements in w_i is arbitrary, we shall use a list (vector), rather than a set, so that we can refer to members of w_i using an index. For example, consider variable x_2 in B_{S1}, which has the parent list $\pi_2 = (x_1)$. In this example, $w_2 = ((1), (0))$, because there are cases in D where x_1 has the value 1 and cases where it has the value 0. Define w_{ij} to be the jth element of w_i. Thus, for example, w_{21} is equal to (1). Let $\sigma(i, h)$ be an index function, such that the instantiation of π_i in case h is the $\sigma(i, h)$th element of w_i. Thus, for example, $\sigma(2, 3) = 2$, because in case 3 the parent of variable x_2—namely, x_1—is instantiated to the value 0, which is represented by the second element of w_2. Therefore, $w_{2,\sigma(2,3)}$ is equal to (0). Since, according to assumption 3, cases are complete, we can use equation (1) in section 1 to represent the probability of each case; thus, we can expand equation (A3) to become

$$P(B_S, D) = P(B_S) \int_{B_P} \left[\prod_{h=1}^{m} \prod_{i=1}^{n} P(x_i = d_{ih} | \pi_i = w_{i\sigma(i,h)}, B_P) \right] f(B_P | B_S) dB_P. \quad \text{(A4)}$$

The innermost product of equation (A4) computes the probability of a case in terms of the conditional probabilities of the variables in the case, as defined by belief network (B_S, B_P).

By grouping terms, we can rewrite equation (A4) as

$$P(B_S, D) = P(B_S) \int_{B_P} \left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} P(x_i = v_{ik} | \pi_i = w_{ij}, B_P)^{N_{ijk}} \right] f(B_P | B_S) dB_P. \quad \text{(A5)}$$

Let θ_{ijk} denote the conditional probability $P(x_i = v_{ik} | \pi_i = w_{ij}, B_P)$. We shall call an assignment of numerical probabilities to θ_{ijk}, for $k = 1$ to r_i, a *probability distribution*, which we represent as the list $(\theta_{ij1}, \ldots, \theta_{ijr_i})$. Note that, since the values of v_{ik}, for $k = 1$ to r_i, are mutually exclusive and exhaustive, it follows that $\Sigma_{1 \le k \le r_i} \theta_{ijk} = 1$. In addition, for a given x_i and w_{ij}, let $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ denote the probability density function over $(\theta_{ij1}, \ldots, \theta_{ijr_i})$. We call $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ a *second-order probability distribution* because it is a probability distribution over a probability distribution.

Two assumptions follow from assumption 4:

4a. The distribution $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ is independent of the distribution $f(\theta_{i'j'1}, \ldots, \theta_{i'j'r_{i'}})$, for $1 \leq i, i' \leq n$, $1 \leq j \leq q_i$, $1 \leq j' \leq q_{i'}$, and $ij \neq i'j'$;
4b. Distribution $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ is uniform, for $1 \leq i \leq n$, $1 \leq j \leq q_i$.

Assumption 4a can be expressed equivalently as

$$f(B_P|B_S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} f(\theta_{ij1}, \ldots, \theta_{ijr_i}). \tag{A6}$$

Equation (A6) states that our belief about the values of a second-order probability distribution $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ is not influenced by our belief about the values of other second-order probability distributions. That is, the distributions are taken to be independent.

Assumption 4b states that, initially, before we observe database D, we are indifferent regarding giving one assignment of values to the conditional probabilities $\theta_{ij1}, \ldots, \theta_{ijr_i}$, versus some other assignment.

By substituting θ_{ijk} for $P(x_i = v_{ik}|\pi_i = w_{ij}, B_P)$ in equation (A5), and substituting equation (A6) into equation (A5), we obtain

$$P(B_S, D) =$$

$$P(B_S) \int_{\theta_{ijk}} \cdots \int \left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \right] \left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} f(\theta_{ij1}, \ldots, \theta_{ijr_i}) \right]$$

$$d\theta_{111}, \ldots, d\theta_{ijk}, \ldots, d\theta_{nq_n r_n} \tag{A7}$$

where the integral is taken over all θ_{ijk} for $i = 1$ to n, $j = 1$ to q_i, and $k = 1$ to r_i, such that $0 \leq \theta_{ijk} \leq 1$, and for every i and j the following condition holds: $\Sigma_k \theta_{ijk} = 1$. These constraints on the variables of integration apply to all the integrals that follow, but for brevity we will not repeat them.

By using the independence of the terms in equation (A7), we can convert the integral of products in that equation to a product of integrals:

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \int_{\theta_{ijk}} \cdots \int \left[\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \right] f(\theta_{ij1}, \ldots, \theta_{ijr_i}) d\theta_{ij1}, \ldots, d\theta_{ijr_i}. \tag{A8}$$

By Assumption 4b, it follows that $f(\theta_{ij1}, \ldots, \theta_{ijr_i}) = C_{ij}$ for some constant C_{ij}. Since $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ is a probability-density function, it necessarily follows that, for a given i and j,

$$\int_{\theta_{ijk}} \cdots \int C_{ij}\, d\theta_{ij1}, \ldots, d\theta_{ijr_i} = 1. \tag{A9}$$

We show later in this proof that solving equation (A9) for C_{ij} yields $C_{ij} = (r_i - 1)!$, and, therefore, that $f(\theta_{ij1}, \ldots, \theta_{ijr_i}) = (r_i - 1)!$. Substituting this result into equation (A8), we obtain

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \int_{\theta_{ijk}} \cdots \int \left[\prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \right] (r_i - 1)!\, d\theta_{ij1}, \ldots, d\theta_{ijr_i}. \quad \text{(A10)}$$

Since $(r_i - 1)!$ is a constant within the integral in equation (A10), we can move it outside the integral to obtain

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} (r_i - 1)! \int_{\theta_{ijk}} \cdots \int \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}\, d\theta_{ij1}, \ldots, d\theta_{ijr_i}. \quad \text{(A11)}$$

The multiple integral in equation (A11) is Dirichlet's integral, and has the following solution (Wilks, 1962):

$$\int_{\theta_{ijk}} \cdots \int \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}\, d\theta_{ij1}, \ldots, d\theta_{ijr_i} = \frac{\prod_{k=1}^{r_i} N_{ijk}!}{(N_{ij} + r_i - 1)!}. \quad \text{(A12)}$$

Note that, by applying equation (A12) with $N_{ijk} = 0$, and therefore $N_{ij} = 0$, we can solve equation (A9), as previously stated, to obtain $C_{ij} = (r_i - 1)!$.

Substituting equation (A12) into equation (A11), we complete the proof:

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \quad \text{(A13)}$$

□

Note that the symbol D in theorem 1 represents the cases *in the particular order that they were observed*. Let D' represent the cases without regard to order. By assumption 2, the cases are independent of one another, given some belief network (B_S, B_P). Thus, $P(D'|B_S, B_P) = k\, P(D|B_S, B_P)$, where k is the number of unique ways of ordering the cases in D, known as the *multiplicity*. Since k is a constant relative to D, by equation (2) in section 1 the ordering of $P(B_{S_i}, D)$ and $P(B_{S_j}, D)$ is the same as the ordering of $P(B_{S_i}, D')$ and $P(B_{S_j}, D')$. Furthermore, by Bayes' rule, it is straightforward to show that, if $P(D'|B_S, B_P) = k\, P(D|B_S, B_P)$, then $P(B_{S_i}|D) = P(B_{S_i}|D')$. Thus, in this paper, we consider only the use of D.

Assumption 4 in theorem 1 implies that second-order probabilities are uniformly distributed (Assumption 4b), from which we derived that $f(\theta_{ij1}, \ldots, \theta_{ijr_i}) = (r_i - 1)!$. This probability density function is, however, just a special case of the Dirichlet distribution (deGroot, 1970). We can generalize assumption 4b by representing each $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ with a Dirichlet distribution:

$$f(\theta_{ij1}, \ldots, \theta_{ijr_i}) = \frac{(N'_{ij} + r_i - 1)!}{\prod_{k=1}^{r_i} N'_{ijk}!} \theta_{ij1}^{N'_{ij1}} \ldots \theta_{ijr_i}^{N'_{ijr_i}}, \tag{A14}$$

where

$$N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}.$$

The values we assign to N'_{ijk} determine our prior-probability distribution over the values of $\theta_{ij1}, \ldots, \theta_{ijr_i}$. All else being the same, the higher we make a particular N'_{ijk}, the higher we expect (a priori) the probability θ_{ijk} to be. As we discussed in section 2.1, we can view the term $P(B_S)$ as one form of preference bias for belief-network *structure* B_S. Likewise, we can view the terms N'_{ijk} in equation (A14) as establishing our preference bias for the *numerical* probabilities to place on a given belief-network structure B_S. We summarize the result of this generalization of assumption 4 with the following corollary.

Corollary 1. If assumptions 1, 2, 3, and 4a of theorem 1 hold and second-order probabilities are represented using Dirichlet distributions as given by equation (A14), then

$$P(B_S, D) = P(B_S) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(N'_{ij} + r_i - 1)!}{(N_{ij} + N'_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \frac{(N_{ijk} + N'_{ijk})!}{N'_{ijk}!}. \tag{A15}$$

Proof. Equation (A15) results when we substitute equation (A14) into equation (A8) and apply the steps in the proof of theorem 1 that follow equation (A8). □

Note that when $N'_{ijk} = 0$, for all possible i, j, and k, the Dirichlet distribution, given by equation (A14), reduces to the uniform distribution, and equation (A15) reduces to equation (A13), as we would expect.

Theorem 2. Given the four assumptions of theorem 1, it follows that

$$E[\theta_{ijk} | D, B_S, \xi] = \frac{N_{ijk} + 1}{N_{ij} + r_i}.$$

Proof. This proof will be specific to determining conditional probabilities in belief networks; however, we note that it parallels related results regarding the expected value of probabilities given a Dirichlet distribution (Wilks, 1962). To simplify our notation, we shall use $E[\theta_{ijk} | D]$ to designate $E[\theta_{ijk} | D, B_S, \xi]$ in this proof. Also, for brevity, in this proof, we shall leave implicit the following constraints on the variables of integration: all integrals are taken over all θ_{ijk} for $i = 1$ to n, $j = 1$ to q_i, and $k = 1$ to r_i, such that $0 \leq \theta_{ijk} \leq 1$, and for every i and j the condition $\Sigma_k \theta_{ijk} = 1$ holds.

By the definition of expectation,

$$E[\theta_{ijk}|D] = \int_{\theta_{ij1}} \cdots \int_{\theta_{ijr_i}} \theta_{ijk} f(\theta_{ij1}, \ldots, \theta_{ijr_i}|D) d\theta_{ij1}, \ldots, d\theta_{ijr_i}. \quad \text{(A16)}$$

The function $f(\theta_{ij1}, \ldots, \theta_{ijr_i}|D)$ in equation (A16) is known as the posterior density function, and it can be expressed as

$$f(\theta_{ij1}, \ldots, \theta_{ijr_i}|D) = \frac{P(D(i,j)|\theta_{ij1}, \ldots, \theta_{ijr_i}) f(\theta_{ij1}, \ldots, \theta_{ijr_i})}{P(D(i,j))}, \quad \text{(A17)}$$

where $D(i, j)$ denotes the distribution of x_i in D for those cases in which the parents of x_i have the values designated by w_{ij}. Solving for $P(D(i, j))$ in equation (A17), we obtain

$$\begin{aligned} P(D(i,j)) &= \int_{\theta_{ij1}} \cdots \int_{\theta_{ijr_i}} P(D(i,j)|\theta_{ij1}, \ldots, \theta_{ijr_i}) f(\theta_{ij1}, \ldots, \theta_{ijr_i}) d\theta_{ij1}, \ldots, d\theta_{ijr_i} \\ &= \int_{\theta_{ij1}} \cdots \int_{\theta_{ijr_i}} \left[\prod_{\kappa=1}^{r_i} \theta_{ij\kappa}^{N_{ij\kappa}} \right] f(\theta_{ij1}, \ldots, \theta_{ijr_i}) d\theta_{ij1}, \ldots, d\theta_{ijr_i}, \end{aligned} \quad \text{(A18)}$$

which, when the assumptions and methods in the proof of theorem 1 are applied, yields

$$P(D(i,j)) = \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{\kappa=1}^{r_i} N_{ij\kappa}!, \quad \text{(A19)}$$

where we use κ as an index variable in the product, since in this theorem k is fixed. Similarly, note that the numerator of equation (A17) can be written as

$$P(D(i,j)|\theta_{ij1}, \ldots, \theta_{ijr_i}) f(\theta_{ij1}, \ldots, \theta_{ijr_i}) = \left[\prod_{\kappa=1}^{r_i} \theta_{ij\kappa}^{N_{ij\kappa}} \right] f(\theta_{ij1}, \ldots, \theta_{ijr_i}). \quad \text{(A20)}$$

Substituting equations (A19) and (A20) into equation (A17), and substituting the resulting version of equation (A17) into equation (A16), we obtain

$$E[\theta_{ijk}|D] = \frac{(N_{ij} + r_i - 1)!}{(r_i - 1)! \prod_{\kappa=1}^{r_i} N_{ij\kappa}!} \int_{\theta_{ij1}} \cdots \int_{\theta_{ijr_i}} \theta_{ijk} \left[\prod_{\kappa=1}^{r_i} \theta_{ij\kappa}^{N_{ij\kappa}} \right]$$

$$f(\theta_{ij1}, \ldots, \theta_{ijr_i}) d\theta_{ij1}, \ldots, d\theta_{ijr_i}$$

$$= \frac{(N_{ij} + r_i - 1)!}{(r_i - 1)! \prod_{\kappa=1}^{r_i} N_{ij\kappa}!} \int_{\theta_{ij1}} \cdots \int_{\theta_{ijr_i}} \theta_{ij1}^{N_{ij1}} \cdots \theta_{ijk}^{N_{ijk}+1} \cdots \theta_{ijr_i}^{N_{ijr_i}}$$

$$f(\theta_{ij1}, \ldots, \theta_{ijr_i}) d\theta_{ij1}, \ldots, d\theta_{ijr_i}. \quad \text{(A21)}$$

The multiple integral in equation (A21) can be solved by the methods in the proof of theorem 1 to complete the current proof:

$$E[\theta_{ijk} | D, B_S, \xi] = \frac{(N_{ij} + r_i - 1)!}{(r_i - 1)! \prod_{\kappa=1}^{r_i} N_{ij\kappa}!} \frac{(r_i - 1)! N_{ij1}! \ldots (N_{ijk} + 1)! \ldots N_{ijr_i}!}{(N_{ij} + r_i)!}$$

$$= \frac{N_{ijk} + 1}{N_{ij} + r_i},$$

where, in the left-hand side of this equation, we have expanded our previous shorthand for the expectation. □

Just as corollary 1 generalizes theorem 1, in the following corollary we generalize theorem 2 by permitting second-order probability distributions to be expressed as Dirichlet distributions.

Corollary 2. If assumptions 1, 2, 3, and 4a of theorem 1 hold and second-order probabilities are represented using Dirichlet distributions as given by equation (A14), then

$$E[\theta_{ijk} | D, B_S, \xi] = \frac{N_{ijk} + N'_{ijk} + 1}{N_{ij} + N'_{ij} + r_i}. \quad \text{(A22)}$$

Proof. Equation (A22) results when we substitute equation (A14) into equations (A17) and (A18), and apply the steps in the proof of theorem 2 that follow equation (A17).

References

Agogino, A.M., & Rege, A. (1987). IDES: Influence diagram based expert system. *Mathematical Modelling*, 8, 227-233.

Andreassen, S., Woldbye, M., Falck, B., & Andersen, S.K. (1987). MUNIN—A causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 366-372). Milan, Italy: Morgan Kaufmann.

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., & Cooper, G.F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence in Medicine* (pp. 247-256). London, England.

Blum, R.L. (1982). Discovery, confirmation, and incorporation of causal relationships from a large time-oriented clinical database: The RX project. *Computers and Biomedical Research*, 15, 164-187.

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

Buntine, W.L. (1990a). Myths and legends in learning classification rules. *Proceedings of AAAI* (pp. 736-742). Boston, MA: MIT Press.

Buntine, W.L. (1990b). *A theory of learning classification rules*. Doctoral dissertation, School of Computing Science, University of Technology, Sydney, Australia.

Carbonell, J.G. (Ed.) (1990). Special volume on machine learning. *Artificial Intelligence*, 40, 1-385.

Chavez, R.M. & Cooper, G.F. (1990). KNET: Integrating hypermedia and normative Bayesian modeling. In R.D. Shachter, T.S. Levitt, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence* 4. Amsterdam: North-Holland.

Cheeseman, P. (1983). A method of computing generalized Bayesian probability values for expert systems. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 198-202). Karlsruhe, West Germany: Morgan Kaufmann.

Cheeseman, P., Self, M., Kelly, J., Taylor, W., Freeman, D., & Stutz, J. (1988). Bayesian classification. *Proceedings of AAAI* (pp. 607-611). St. Paul, MN: Morgan Kaufmann.

Chow, C.K. & Liu, C.N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14, 462-467.

Cooper, G.F. (1984). *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge*. Doctoral dissertation, Medical Information Sciences, Stanford University, Stanford, CA.

Cooper G.F. (1989). Current research directions in the development of expert systems based on belief networks. *Applied Stochastic Models and Data Analysis*, 5, 39-52.

Cooper, G.F. & Herskovits, E.H. (1991). *A Bayesian method for the induction of probabilistic networks from data* (Report SMI-91-1). Pittsburgh PA: University of Pittsburgh, Section of Medical Informatics. (Also available as Report KSL-91-02, from the Section on Medical Informatics, Stanford University, Stanford, CA.)

Crawford, S.L. & Fung, R.M. (1991). An analysis of two probabilistic model induction techniques. *Proceedings of the Third International Workshop on AI and Statistics* (in press).

deGroot, M.H. (1970). *Optimal statistical decisions*. New York: McGraw-Hill.

Fung, R. & Shachter, R.D. (1991). *Contingent influence diagrams* (Research report 90-10). Mountain View, CA: Advanced Decision Systems.

Fung, R.M. & Crawford, S.L. (1990a). Constructor: A system for the induction of probabilistic models. *Proceedings of AAAI* (pp. 762-769). Boston, MA: MIT Press.

Fung, R.M., Crawford, S.L., Appelbaum, L.A., & Tong, R.M. (1990b). An architecture for probabilistic concept-based information retrieval. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 392-404). Cambridge, MA.

Geiger, D. & Heckerman, D.E. (1991). Advances in probabilistic reasoning. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 118-126). Los Angeles, CA: Morgan Kaufmann.

Geiger, D., Paz, A., & Pearl, J. (1990). Learning causal trees from dependence information. *Proceedings of AAAI* (pp. 770-776). Boston, MA: MIT Press.

Gevarter, W.B. (1986). *Automatic probabilistic knowledge acquisition from data* NASA Technical Memorandum 88224). Mt. View, CA: NASA Ames Research Center.

Glymour, C., Scheines, R., Spirtes, P., & Kelley, K. (1987). *Discovering causal structure*. New York: Academic Press.

Glymour, C. & Spirtes, P. (1988). Latent variables, causal models and overidentifying constraints. *Journal of Econometrics*, 39, 175-198.

Golmard, J.L., & Mallet, A. (1989). Learning probabilities in causal trees from incomplete databases. *Proceedings of the IJCAI Workshop on Knowledge Discovery in Databases* (pp. 117-126). Detroit, MI.

Heckerman, D.E. (1990). Probabilistic similarity networks. *Networks*, 20, 607-636.

Heckerman, D.E., Horvitz, E.J., & Nathwani, B.N. (1989). Update on the Pathfinder project. *Proceedings of the Symposium on Computer Applications in Medical Care* (pp. 203-207). Washington, DC: IEEE Computer Society Press.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by logic sampling. In J.F. Lemmer & L.N. Kanal (Eds.), *Uncertainty in artificial intelligence 2*. Amsterdam: North-Holland.

Henrion, M. (1990). An introduction to algorithms for inference in belief nets. In M. Henrion, R.D. Shachter, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence 5*. Amsterdam: North-Holland.

Henrion, M. & Cooley, D.R. (1987). An experimental comparison of knowledge engineering for expert systems and for decision analysis. *Proceedings of AAAI* (pp. 471-476). Seattle, WA: Morgan Kaufmann.

Herskovits, E.H. (1991). *Computer-based probabilistic network construction*. Doctoral dissertation, Medical Information Sciences, Stanford University, Stanford, CA.

Herskovits, E.H. & Cooper, G.F. (1990). Kutató: An entropy-driven system for the construction of probabilistic expert systems from databases. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 54–62). Cambridge, MA.

Hinton, G.E. (1990). Connectionist learning procedures. *Artificial Intelligence*, 40, 185–234.

Holtzman, S. (1989). *Intelligent decision systems*. Reading, MA: Addison-Wesley.

Horvitz, E.J., Breese, J.S. & Henrion, M. (1988). Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, 2, 247–302.

Howard, R.A. (1988). Uncertainty about probability: A decision analysis perspective. *Risk Analysis*, 8, 91–98.

Hunt, E.B., Marin, J., & Stone, P.T. (1966). *Experiments in induction*. New York: Academic Press.

James, M. (1985). *Classification algorithms*. New York: John Wiley & Sons.

Johnson, R.A. & Wichern, D.W. (1982). *Applied multivariate statistical analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Kiiveri, H., Speed, T.P., & Carlin, J.B. (1984). Recursive causal models. *Journal of the Australian Mathematical Society*, 36, 30–52.

Kwok, S.W. & Carter, C. (1990). Multiple decision trees. In R.D. Shachter, T.S. Levitt, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence 4*. Amsterdam: North-Holland.

Lauritzen, S.L. & Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society (Series B)*, 50, 157–224.

Liu, L., Wilkins, D.C., Ying, X., & Bian, Z. (1990). Minimum error tree decomposition. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 180–185). Cambridge, MA.

Michalski, R.S., Carbonell, J.G., & Mitchell, T.M. (Eds.) (1983). *Machine learning: An artificial intelligence approach* (Vol. 1). Palo Alto, CA: Tioga Press.

Michalski, R.S., Carbonell, J.G., & Mitchell, T.M. (Eds.) (1986). *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Mitchell, T.M. (1980). *The need for biases in learning generalizations* (Report CBM-TR-5-110). New Brunswick, NJ: Rutgers University, Department of Computer Science.

Neapolitan, R. (1990). *Probabilistic reasoning in expert systems*. New York: John Wiley & Sons.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29, 241–288.

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

Pearl. J. & Verma, T.S. (1991). A theory of inferred causality. *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning* (pp. 441–452). Boston, MA: Morgan Kaufmann.

Pittarelli, M. (1990). Reconstructability analysis: An overview. *Revue Internationale de Systemique*, 4, 5–32.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.

Rebane, G. & Pearl, J. (1987). The recovery of causal poly-trees from statistical data. *Proceedings of the Workshop on Uncertainty in Artificial Intelligence* (pp. 222–228). Seattle, Washington.

Robinson, R.W. (1977). Counting unlabeled acyclic digraphs. In C.H.C. Little (Ed.), *Lecture notes in mathematics, 622: Combinatorial mathematics V*. New York: Springer-Verlag. (Note: This paper also discusses counting of labeled acyclic graphs.)

Shachter, R.D. (1986). Intelligent probabilistic inference. In L.N. Kanal & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence 1*. Amsterdam: North-Holland.

Shachter, R.D. (1988). Probabilistic inference and influence diagrams. *Operations Research* 36, 589–604.

Shachter, R.D. (1990). A linear approximation method for probabilistic inference. In R.D. Shachter, T.S. Levitt, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence 4*. Amsterdam: North-Holland.

Shachter, R.D. & Kenley, C.R. (1989). Gaussian influence diagrams. *Management Science*, 35, 527–550.

Spiegelhalter, D.J. & Lauritzen, S.L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20, 579–606.

Spirtes, P. & Glymour, C. (1990). *Causal structure among measured variables preserved with unmeasured variables* (Report CMU-LCL-90-5). Pittsburgh, PA: Carnegie Mellon University, Department of Philosophy.

Spirtes, P., Glymour, C., & Scheines, R. (1990a). *Causal hypotheses, statistical inference, and automated model specification* (Research report). Pittsburgh, PA: Carnegie Mellon University, Department of Philosophy.

Spirtes, P., Glymour, C., & Scheines, R. (1990b). Causality from probability. In G. McKee (Ed.), *Evolving knowledge in natural and artificial intelligence*. London: Pitman.

Spirtes, P., Glymour, C., & Scheines, R. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9, 62–72.

Spirtes, P., Scheines, R., & Glymour, C. (1990c). Simulation studies of the reliability of computer-aided model specification using the Tetrad II, EQS, and LISREL programs. *Sociological Methods and Research*, 19, 3–66.

Srinivas, S., Russell, S., & Agogino, A. (190). Automated construction of sparse Bayesian networks for unstructured probabilistic models and domain information. In M. Henrion, R.D. Shachter, L.N. Kanal, & J.F. Lemmer (Eds.), *Uncertainty in artificial intelligence 5*. Amsterdam: North-Holland.

Suermondt, H.J. & Amylon, M.D. (1989). Probabilistic prediction of the outcome of bone-marrow transplantation. *Proceedings of the Symposium on Computer Applications in Medical Care* (pp. 208–212). Washington, DC: IEEE Computer Society Press.

Utgoff, P.E. (1986). *Machine learning of inductive bias*. Boston, MA: Kluwer Academic.

Verma, T.S. & Pearl, J. (1990). Equivalence and synthesis of causal models. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (pp. 220–227). Cambridge, MA.

Wermuth, N. & Lauritzen, S. (1983). Graphical and recursive models for contingency tables. *Biometrika*, 72, 537–552.

Wilks, S.S. (1962). *Mathematical statistics*. New York: John Wiley & Sons.