A Bayesian Method for Learning Belief Networks that Contain Hidden Variables

Gregory F. Cooper
Section of Medical Informatics
University of Pittsburgh
B50A Lothrop Hall
Pittsburgh, PA 15261
gfc@med.pitt.edu

1. Introduction

This paper presents a Bayesian method for computing the probability of a Bayesian belief-network structure from a database. In particular, the paper focuses on computing the probability of a belief-network structure that contains a hidden (latent) variable. A hidden 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.

A belief-network structure can provide insight into probabilistic dependencies that exist among the variables in a database. One application is the automated discovery of dependency relationships. The computer program searches for a belief-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 will emphasize this task in the current paper. We also note that given a belief-network structure and a database, we can construct a belief network and use it for computer-based diagnosis and prediction.

In [8] a Bayesian method is presented for computing the probability of a belief-network structure that contains hidden variables. The major problem with this approach is that its computational time complexity is exponential in the number of database cases (i.e., records or instances). In this paper, we present a method that has a time complexity that is polynomial in the number of cases, yet it always computes the same result as the exponential method. By being more efficient, this polynomial-time method permits us to solve modeling problems that previously were not computationally feasible. In some instances, which we explain, the polynomial degree of the new method can be high. As the polynomial degree increases, the number of cases considered in the database must decrease in order to maintain computational tractability. We will initially focus our discussion on handling a single hidden variable, and then extend the method to handle multiple hidden variables.

2. Background on Belief Networks

A belief-network *structure* B_S is a directed acyclic graph in which nodes represent domain variables and arcs between nodes represent probabilistic dependencies [7, 18]. The representation of conditional dependence and independence among variables is the essential function of belief networks. For a detailed discussion of the formal mathematical properties of belief networks, see [18].

A belief-network structure, B_S, is augmented by conditional probabilities, B_P, to form a belief network B. Thus, $B = (B_S, B_P)$. 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.

---

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.

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. As mentioned, the key feature of belief networks is their explicit representation of the conditional independence and dependence among events. In particular, investigators have shown [16, 18, 19] that the joint probability of any particular instantiation[1] of all n variables in a belief network can be calculated as follows:

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

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 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 have sufficient information to determine any probability of the form $P(W \mid V)$, where W and V are sets of variables with known values (instantiated variables).

In the last few years, researchers have made significant progress in formalizing the theory of belief networks [17, 18], and in developing more efficient algorithms for probabilistic inference on belief networks [12]. The feasibility of using belief networks in constructing diagnostic systems has been demonstrated in several domains [1, 2, 3, 4, 6, 9, 13, 15, 24].

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. In domains that are large or in which there are few, if any, readily available experts, automated 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 and show how it can handle hidden variables.

Researchers have developed non-Bayesian methods for discovering the presence of hidden variables in belief-network structures [22, 23, 26]. In general, the validity of the output of these algorithms depends on whether the conditional independence and dependence relationships among variables can be determined categorically from the available sample of cases. These methods do not provide the probability of a belief-network structure that contains one or more hidden variables. Bayesian methods can provide such probabilities. In this paper we discuss an exact Bayesian method for determining the probability of a belief-network structure when there are hidden variables or missing data. We note that in related work researchers have developed simulation [27] and approximation [20, 21] techniques to perform Bayesian parameter estimation in belief-networks when there is missing data.

3. An Example

In this section we introduce a simple example that we will use throughout the paper to illustrate basic concepts. Assume we have the database shown in Table 1, which we designate as database D. To be more concrete, suppose x_1 represents a disease etiology, and x_2 and x_3 represent two patient symptoms. Assume from independent knowledge that we are quite certain neither symptom causally leads to the other symptom. More particularly, suppose we believe that the causal relationships among the three variables correspond to one of the following two possibilities: (1) the symptom variables are conditionally independent given the state of the disease-etiology variable (see Figure 1), or (2) the symptom variables are caused by an intermediate pathophysiological process between the disease-etiology variable and the two symptoms (see Figure 2). Figures 1 and 2 show the two possible belief-network structures under consideration, which we denote as BS_1 and BS_2. In BS_2 the variable h represents

[1] An instantiated variable is a variable with an assigned value.

an unknown pathophysiological process. To keep the example simple, we will assume that h is a binary variable and that we have the following prior probabilities for BS_1 and BS_2: $P(BS_1) = P(BS_2) = 0.5$.

Although the current example is hypothetical, Henrion has documented a case in which an expert (who was building a belief network to diagnosis particular apple-tree diseases) first constructed the subnetwork structure shown in Figure 1 [11]. The expert later realized that the belief-network structure in Figure 2 more accurately reflects the relationships among the variables. The methods we propose in this paper are intended to automatically suggest refinements like the one in this example, as well as other refinements.

Table 1. A database example, which we denote as D. The term *case* in the first column denotes a single training instance (record) in the database—as for example, a patient case.

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

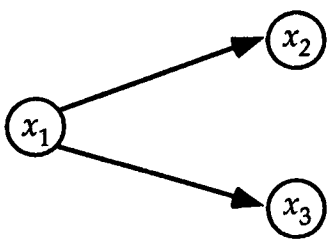

Figure 1. Belief-network structure BS_1, which represents that x_2 and x_3 are conditionally independent given x_1.

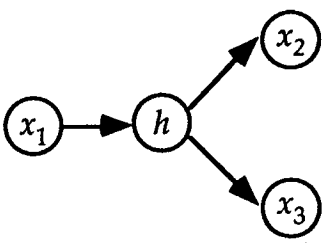

Figure 2. Belief-network structure BS_2, which represents that x_2 and x_3 are conditionally independent given some hidden variable h. In general, for a belief-network with this structure, x_2 and x_3 will not be conditionally independent given x_1.

Our goal is to determine $P(BS_1 \mid D)$ and $P(BS_2 \mid D)$. Using the assumptions and techniques to be presented in Section 4, we can show that $P(BS_1 \mid D) = 0.25$ and $P(BS_2 \mid D) = 0.75$. Deriving these results with the methods in Section 4, however, requires that we perform 1024 operations on the data in Table 1. In Section 5, we introduce a more efficient method that derives the same results in only 240 operations. In Sections 4 and 5 we provide a general analysis of the computational time complexity of both methods.

4. Previous Results

Previously, a method was described for Bayesian belief-network-structure learning from a database that is generated by a process that can be accurately modeled by a belief network with no missing values or hidden variables [8]. In Section 4.1, we present those results. In Section 4.2, we show previous extensions of this method that learn belief-network structures containing hidden variables.

4.1 Results When There Are No Missing Data or Hidden Variables

Suppose for the moment that we have a method for calculating $P(B_{S_i}, D)$ for some belief-network structure B_{S_i} and database D. Let Q be the set of all those belief-network structures that have a non-zero prior probability. We can derive the posterior probability of B_{S_i} given D as $P(B_{S_i} \mid D) = P(B_{S_i}, D)/\Sigma_{B_S \in Q} P(B_{S_i}, D)$. Sometimes all we want to know is the ratio of the posterior probabilities of two belief-network structures. To calculate such a ratio for belief-network structures B_{S_i} and B_{S_j}, we can use the equivalence that $P(B_{S_i} \mid D)/P(B_{S_j} \mid D) = P(B_{S_i}, D)/P(B_{S_j}, D)$. An algorithm called K2 is reported in [8] that makes such comparisons of belief-network structures, as it searches heuristically for the most likely structure.

Our focus in this paper will be on computing the term $P(B_{S_i}, D)$, which we can derive using the equivalence that $P(B_{S_i}, D) = P(D \mid B_{S_i})\, P(B_{S_i})$. The term $P(B_{S_i})$ represents our prior probability that a process with belief-network structure B_{S_i} generated data D. We will assume that $P(B_{S_i})$ is specified by the user and available. The likelihood term $P(D \mid B_{S_i})$ remains to be determined. The following theorem, which is proved in [8], provides a method for computing $P(D \mid B_S)$.

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 fours assumptions, it follows that

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

□

The complete proof of Theorem 1 is given in the Appendix in [8]. We only discuss here the parts of the proof we will need later in this paper. From Assumption 1 we have that

$$P(D \mid B_S) = \int_{B_P} P(D \mid B_S, B_P)\, f(B_P \mid B_S)\, dB_P, \quad (2)$$

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. It follows from the conditional independence of cases expressed in Assumption 2 that Equation 2 can be rewritten as

$$P(D \mid B_S) = \int_{B_P} \left[\prod_{t=1}^{m} P(C_t \mid B_S, B_P)\right] f(B_P \mid B_S)\, dB_P, \quad (3)$$

where m is the number of cases in D, and C_t is the tth case in D.

As shown in [8], we can apply Assumptions 3 and 4 to derive Equation 1 from Equation 3. Since Equation 3 is all we need for the current paper, we will not show these additional steps here.

As a point of illustration, we describe now one application of Theorem 1. The K2 algorithm mentioned previously uses Equation 1 and assumes uniform priors over belief-network structures. We briefly summarize here the results of applying K2 to data generated by the ALARM belief network. The ALARM network is a laboratory research prototype that models potential anesthesia problems in the operating room [3]. ALARM, which contains 37 nodes and 46 arcs, was used to stochastically generate a database of 3000 cases. When given these cases, and a node order, K2 reconstructed the ALARM belief-network structure exactly, except there was one missing arc and one extra arc. The missing arc was not strongly supported by the 3000 cases, and the extra arc was due to the greedy nature of the K2 search algorithm. Additional details about this experiment are discussed in [8], and further evaluation of K2 on other databases is presented in [14].

In this paper, we will adopt Assumptions 1, 2, and 4 listed in Theorem 1, although Assumption 4 can be relaxed, as shown in [8]. We note that Assumption 4 implies a uniform prior over each of the conditional probability distributions (the parameters of the model) associated with a given B_S (the model structure).

In this paper, we will relax Assumption 3 in one important way: All variables in the belief network are assumed to be known, except one variable for which we know none of its values. This variable is a hidden variable.

4.2 Previous Hidden-Variable Results

In this section, we consider belief-network structures that contain a single hidden variable, which we denote as h. In [8] we describe a more general (yet still inefficient) method that handles both missing data and multiple hidden variables.

Let h denote a hidden variable in B_S. We use h_t as a variable that represents h in case t — we call h_t a *case-specific hidden variable*. Let C_t denote the set of variable assignments for those variables in the tth case that have known values (i.e., all the variables except h). The probability of the tth case can be computed as

$$P(C_t \mid B_S, B_P) = \sum_{h_t} P(C_t, h_t \mid B_S, B_P), \quad (4)$$

where the sum Σ_{h_t} indicates that h_t is run through all the possible values of h.

By substituting Equation 4 into Equation 3, we obtain

$$P(D \mid B_S) = \int_{B_P} \left[\prod_{t=1}^{m} \left[\sum_{h_t} P(C_t, h_t \mid B_S, B_P)\right]\right] f(B_P \mid B_S)\, dB_P. \quad (5)$$

We now can rearrange Equation 5 as follows by converting an integral of a product of a sum into sums of an integral of a product:

$$P(D \mid B_S) = \sum_{h_1} \cdots \sum_{h_m} \int_{B_P} [\prod_{t=1}^{m} P(C_t, h_t \mid B_S, B_P)] \; f(B_P \mid B_S)\, dB_P . \quad (6)$$

The sums in Equation 6 are taken over every possible value of the hidden variable for every possible case in the database. Equation 6 is a sum over the type integral that occurs in Equation 3, and as Theorem 1 shows, we can solve such integrals with Equation 1. Thus, Equation 6 can be solved by multiple applications of Equation 1. The complexity of computing Equation 6 is exponential in the number of cases m in the database. In particular, if the hidden variable can take on one of r possible values, then to solve Equation 6 directly requires r^m applications of Equation 1. This high level of complexity is not computationally tractable for most applications. In Section 5, we describe a more efficient method for computing Equation 6.

Example

Let us augment Table 1 to represent a hidden variable. In particular, this new table is the same as Table 1, except that a column is added for the hidden variable h. For each of the ten cases in the table, the variable h is represented by a *case-specific hidden variable* h_t, for t = 1 to 10. Each case-specific hidden variable can be instantiated to either the value *absent* or *present*. We apply Equation 6 to this table in order to compute $P(D \mid B_{S_2})$, where B_{S_2} is the belief-network structure in Figure 2. Since there are ten cases in D, the total number of instantiations of the ten binary variables $h_1, h_2, \ldots, h_{10}$ is 2^{10} = 1024. Thus, by using Equation 6 to solve for $P(D \mid B_{S_2})$, Equation 1 must be solved 1024 times. If we use Equation 6, then as the number of cases with a hidden variable increases, the number of times we must solve Equation 1 grows exponentially. To address this problem of exponential growth in complexity, we introduce a more efficient solution method in the next section.

5. A Method for Handling Hidden Variables More Efficiently

In this section we present a method that computes Equation 6, but does not use exhaustive enumeration over the case-specific hidden variables. Instead, we aggregate cases in the database into groups, and we use these groups to calculate a result equivalent to one produced by Equation 6. Before introducing the general method, we first illustrate the basic ideas underlying it by showing how to apply the method to the example previously introduced.

5.1 Application to the Example

In this section we partially formalize the new hidden-variable method by using the previous example to illustrate basic concepts. In Section 5.2 we complete the formalization to the general case.

According to Assumption 2 in Theorem 1, cases are assumed to occur independently given a belief-network model, which implies that the order of the cases in the database does not matter (i.e., the cases are exchangeable). Therefore, in Table 2 we have aggregated identical cases into groups. There are six groups: four of the groups contain only one case, one group contains two cases, and another group contains four cases.

The basic idea underlying a more efficient method for handling hidden variable h is this: we compute the integrals in Equation 6 using groups of cases rather than individual cases. Recall that the integral in Equation 6 is solved by Equation 1. According to Equation 1, it does not matter *which* case-specific hidden variable in a given group has a particular value. Rather, all that matters is the total number of values of *present* and of *absent* for the case-specific hidden variables in a group. Thus, for example, the solution of the integral in Equation 6 will be the same, regardless of whether we

instantiate the case-specific hidden variables in group 2 as (h_6 := *present*, h_9 := *absent*) or as (h_6 := *absent*, h_9 := *present*).

Suppose a given group G_i contains c_i case-specific hidden variables, and it has d_i of those variables instantiated to the value *present* and c_i - d_i of the variables instantiated to *absent*. In this situation, we will say that G_i is *set* to the value d_i. Note that d_i can range from 0 to c_i, and thus, G_i has c_i + 1 possible settings. Rather than compute the integral in Equation 6 over all possible instantiations of the case-specific hidden variables in D, we compute only over the possible settings for each group. Note that there are $\binom{c_i}{d_i}$ ways to instantiate the case-specific hidden variables in G_i so that d_i of them have the value *present* and c_i - d_i of them have the value *absent*. Suppose there are u total groups. According to Equation 6, for a given setting of each group, the integral in that equation is solved $\Pi_{i=1,u}\binom{c_i}{d_i}$ times. Thus, for a given setting of the groups, we must multiply our solution of the integral by the factor $\Pi_{i=1,u}\binom{c_i}{d_i}$.

Table 2. In this table cases are aggregated into groups that are demarcated by bold horizontal lines. Note in the second column that cases are classified according to the group they are in.

		Variable values for each case			
Case	Group	x_1	h	x_2	x_3
2	1	*present*	h_2	*present*	*present*
4	1	*present*	h_4	*present*	*present*
7	1	*present*	h_7	*present*	*present*
8	1	*present*	h_8	*present*	*present*
6	2	*absent*	h_6	*present*	*present*
9	2	*absent*	h_9	*present*	*present*
3	3	*absent*	h_3	*absent*	*absent*
10	4	*present*	h_{10}	*absent*	*absent*
1	5	*present*	h_1	*absent*	*present*
5	6	*absent*	h_5	*present*	*absent*

Consider, as an example, computing the integral in Equation 6 for the situation in which two of the case-specific hidden variables in group 1 have the value *present* (and thus, two must have the value *absent*), one case-specific hidden variable in group 2 has the value *present*, and in groups 3 through 6, each case-specific hidden variable has the value *present*. These settings of the groups correspond to $\binom{4}{2}\binom{2}{1}\binom{1}{1}\binom{1}{1}\binom{1}{1}\binom{1}{1}$ = 12 solutions to the integral in Equation 6, if the exhaustive technique represented by that equation is used. However, each of these 12 solutions will be the same. The solutions are the same because for a given setting of the groups, Equation 1 is insensitive to the particular instantiations of the case-specific hidden variables that are consistent with those given group settings. Thus, to be more efficient (for the given group settings in this example), we solve the integral in Equation 6 only once (by using Equation 1), and we multiply that solution to 12. We perform this procedure for each possible setting of the groups. We sum the results of each such procedure application to obtain a solution equivalent to that obtained by using Equation 6. The total number of joint settings of all the groups is equal the product of the possible settings of each group. For the example, this product is $5 \times 3 \times 2 \times 2 \times 2 \times 2 = 240$. Thus, we must solve Equation 1 only 240 times, rather than the 1024 times required by Equation 6.

5.2 The General Method

In this section we generalize the approach discussed in Section 5.1. We extend hidden variables to be r-valued discrete variables, for any $r \geq 2$. We also provide a general formula for computing $P(B_S \mid D)$ that is more efficient than Equation 6. In the next paragraph we establish some terms and definitions that will be used in a theorem that follows.

We will designate the possible values of h as $(v_{h1}, \ldots, v_{hr_h})$, where $r_h \geq 2$. Let $MB(h)$ be the Markov blanket [18] of variable h.[3] Consider a set of cases in D of cardinality c_i that each have the same values (instantiations) for the variables in $MB(h)$; let G_i represent the set of case-specific hidden variables in these cases. A *setting* of G_i is an instantiation of the variables in G_i that is consistent with the distribution $(y_{i1}, \ldots, y_{ir_h})$, where y_{ij} designates the number of variables in G_i that are instantiated to the value v_{hj}. If an instantiation I_a of the variables in G_i has the same distribution $(y_{i1}, \ldots, y_{ir_h})$ as an instantiation I_b, then I_a and I_b correspond to the same setting of G_i. Let Σ_{G_i} denote the sum over all the possible settings of G_i. For each setting of G_i in this summation, the case-specific hidden variables in G_i are instantiated to some set of values that correspond to this setting of G_i. Let $f(G_1, \ldots, G_u)$ denote the total number of possible instantiations of the m case-specific hidden variables in $G_1 \cup \ldots \cup G_u$ that are consistent with the joint setting of the groups $G_1, \ldots, G_u$. Since each G_i contains a unique set of case-specific variables, it follows that $f(G_1, \ldots, G_u) = f(G_1) \times \ldots \times f(G_u)$. A term $f(G_i)$ denotes the total number of possible settings of the c_i variables in G_i, and as shown in [25, page 187, Theorem 1], this term can be computed as

$$f(G_i) = \frac{c_i!}{y_{i1}!\, y_{i2}! \ldots y_{ir_h}!}.$$

Theorem 2 Let D be a database of m cases, where each case contains a value assignment for each variable in Z, except for a hidden variable h, which has no value assignments. Let B_S denote a belief-network structure containing just the variables in Z. Suppose that among the m cases in D there are u unique instantiations of the variables in $MB(h)$. Given these conditions and the Assumptions 1, 2 and 4 in Theorem 1, it follows that

$$P(D \mid B_S) = \sum_{G_1} \ldots \sum_{G_u} f(G_1, \ldots, G_u) \int_{B_P} \left[\prod_{t=1}^{m} P(C_t, h_t \mid B_S, B_P) \right] f(B_P \mid B_S)\, dB_P . \qquad (7)$$

Proof (Sketch) Recall that Equation 1 solves the integral in Equation 6. Note that the result determined by Equation 1 is only sensitive to the values of r_i, N_{ijk} and N_{ij}. The values of N_{ijk} and N_{ij} represent numerical summaries of the database (i.e., counts), which are insensitive to the particular order of cases that produce those counts. Thus, in Equation 6, if one instantiation of $h_1, \ldots, h_m$ leads to the same values of N_{ijk} and N_{ij} as another instantiation, then the result from Equation 1 will be the same for both instantiations. A given setting of $G_1, \ldots, G_u$ in Equation 7 corresponds to those instantiations of $h_1, \ldots, h_m$ that produce the same values of N_{ijk} and N_{ij} in Equation 1. In particular, for a given setting of $G_1, \ldots, G_u$ there are $f(G_1, \ldots, G_u)$ instantiations of $h_1, \ldots, h_m$ in Equation 6 that produce the same result when the integral in that equation is solved using Equation 1. Thus, to compute $P(D \mid B_S)$ using Equation 6, we can solve the integral in that equation using Equation 1 for each setting of $G_1, \ldots, G_u$ and then multiply by $f(G_1, \ldots, G_u)$ the solution of the integral for each such setting. Equation 7 computes $P(D \mid B_S)$ in the manner just outlined.

□

---

[3] A Markov blanket of a node x_i is given by the set of nodes consisting of the parents of x_i, the children of x_i, and all the parents of the children of x_i except x_i itself. Knowing the values of the nodes in the Markov blanket of x_i makes the probability distribution over x_i independent of the values of all nodes outside the Markov blanket of x_i.

5.3 Computational Time Complexity

In this section, we derive the time complexity for computing Equation 7. We will assume that each variable in the database can be assigned any one of at most r possible discrete values. As shown in [8], the complexity of computing the integral in Equation 7 is $O(m\, n^2\, r)$. Now consider the number of times we must solve the integral in Equation 7. We will use the following lemma.

Lemma 1 Let $x_1, \ldots, x_u$ denote real variables, and let m and k designate non-negative real constants. Subject to the following constraints: (1) $x_1 + \ldots + x_u = m$, and (2) $0 \le x_1 \le m, \ldots, 0 \le x_u \le m$, the following equivalence is valid:

$$\max_{x_1, \ldots, x_u} [\prod_{i=1}^{u} (x_i + k)] = \prod_{i=1}^{u} (\frac{m}{u} + k).$$

If the variables $x_1, \ldots, x_u$ are further constrained to be integer valued, then the maximization on the left is less than or equal to the product on the right.

□

Theorem 3 The time required to solve Equation 7 is $O((m\, n^2\, r)(m/u + r - 1)^{u\,(r-1)})$. (8)

Proof The number of settings of G_i is equivalent to the number of ways of distributing c_i identical objects into r_h different boxes, which as shown in [25, page 194] is equal to $(c_i + r_h - 1)!/c_i!\,(r_h - 1)!$. Thus, the integral in Equation 7 is solved $\Pi_{i=1,u}\, (c_i + r_h - 1)!/c_i!\,(r_h - 1)!$ times. We can express $(c_i + r_h - 1)!/c_i!\,(r_h - 1)!$ equivalently as $(c_i + r_h - 1)\,(c_i + r_h - 2) \ldots (c_i + 1)/(r_h - 1)!$, which is no greater than $(c_i + r_h - 1)^{r_h - 1}$. Therefore, the integral in Equation 7 is solved no more than $\Pi_{i=1,u}\, (c_i + r_h - 1)^{r_h - 1} = (\Pi_{i=1,u}\, (c_i + r_h - 1))^{r_h - 1}$ times. By Lemma 1, this last product can be no greater than $\Pi_{i=1,u}\, (m/u + r_h - 1)$, and thus the integral in Equation 7 is solved no more than the following number of times: $(\Pi_{i=1,u}\, (m/u + r_h - 1))^{r_h - 1} = \Pi_{i=1,u}\, (m/u + r_h - 1)^{r_h - 1} = (m/u + r_h - 1)^{u\,(r_h - 1)}$. Thus, the number of times the integral in Equation 7 is solved is $O((m/u + r_h - 1)^{u\,(r_h - 1)})$.

Since the integral in Equation 7 is solved $O(m/u + r_h - 1)^{u\,(r_h - 1)})$ times and each solution requires, $O(m\, n^2\, r)$ time, the total time required to solve Equation 7 is given by Equation 8.

Equation 8 does not take into account the time required to create the groups $G_1, \ldots, G_u$. We can generate these groups by inserting the m cases into a rooted tree with a branching factor of at most r, where each path in the tree represents a unique instantiation of the variables in $MB(h)$. Such a tree can be constructed in $O(m\, r\, |MB(h)|) = O(m\, n\, r)$ time, and thus, Equation 8 still represents the time complexity required to solve Equation 7.

□

Recall that u denotes only the number of unique instantiations *actually realized* in database D of the variables in the Markov blanket of hidden variable h. The number of such unique instantiations significantly influences the efficiency with which we can compute Equation 7. For any finite belief-network, the number of such unique instantiations reaches a maximum, regardless of how many cases there are in the database. Recall that r denotes the maximum number of possible values for any variable in the database. If u and r are bounded from above, then the time to solve Equation 7 is bounded from above by a function that is polynomial in the number of variables n and the number of cases m. If u or r is large, however, the polynomial will be of high degree.

6. Results on an Example Using Simulated Data

This section provides an example application of the methods discussed in this paper. We emphasize that this section contains only a single example, rather than a thorough empirical evaluation of the methods in the paper.

We generated a database of 45 cases from a belief network by using a stochastic sampling method [10]. Let us denote this belief network as B^* and the database as D^*. The belief-network structure of B^* is shown in Figure 2. The probabilities of B^* are as follows:

$$
\begin{aligned}
P(x_1 = present) &= 0.6 \\
P(h = present \mid x_1 = present) &= 0.9 \\
P(h = present \mid x_1 = absent) &= 0.3 \\
P(x_2 = present \mid h = present) &= 0.9 \\
P(x_2 = present \mid h = absent) &= 0.05 \\
P(x_3 = present \mid h = present) &= 0.8 \\
P(x_3 = present \mid h = absent) &= 0.05
\end{aligned}
$$

Using the 45 cases in D^*, we computed the probability of the structures in Figures 1 and 2 by applying both the exhaustive method in Section 4.2 (call it EXH) corresponding to Equation 6, and the Markov-blanket method in Section 5.2 (call is MB) corresponding to Equation 7. Table 3 shows the timing results of applying EXH and MB to compute the posterior probability of the structure in Figure 2 for the first 5, 10, 15, ... , and 45 cases in D^* when using a Macintosh IIci (with a Daystar cache card) and Think Pascal 4.0. The EXH method could feasibly handle a database with up to about 15 cases on this machine, whereas the MB method could feasibly handle about 40 cases.

Table 3. The time required by the EXH and MB methods to compute the posterior probability of belief-network structure B_{S_2} shown in Figure 2, using the first 5, 10, 15, ... , or 45 cases in database D^*. The symbol s denotes seconds, m denotes minutes, h denotes hours, and y denotes years. Only the first 5, 10, and 15 cases were computed using EXH; the remaining computation times for EXH were made by using a conservative extrapolation that bounds the results from below.

cases:	5	10	15	20	25	30	35	40	45
EXH:	0.1 s	2.4 s	101 s	>53 m	>28 h	>38 d	>3 y	>107 y	>3424 y
MB:	0.1 s	0.2 s	1 s	9 s	21 s	77 s	139 s	232 s	867 s

These results, while limited in scope, do illustrate two general points. First, the MB method has a run time that in general grows much more slowly than does the run time of EXH. Second, the run time of MB can still grow rapidly, even though it is a much slower growth than that of EXH. In this example, the run time of MB grows polynomially in the number of cases, where the degree of the polynomial is 8, because 8 is the number of unique instantiations in D^* of the 3 binary variables in the Markov blanket of h (namely, x_1, x_2, and x_3).

Let us assume that belief-network structure B_{S_1} in Figure 1 and structure B_{S_2} in Figure 2 each have a prior probability of 0.5. Given these priors, the probability $P(B_{S_2} \mid D)$ is plotted in Figure 3 as a function of the number of cases in D, when D is taken to be the first 5, 10, 15, ... , and 45 cases in D^*. The first 5 cases did not provide enough data to indicate that the most likely data-generating belief network had structure B_{S_2}. Beginning with the first 10 cases, however, the posterior probability of B_{S_2} remains greater than that of B_{S_1}. The dip from between case size 10 and 25 in Figure 3 is likely the

result of variation from statistical sampling, which typically is more prominent when the number of samples is not large.

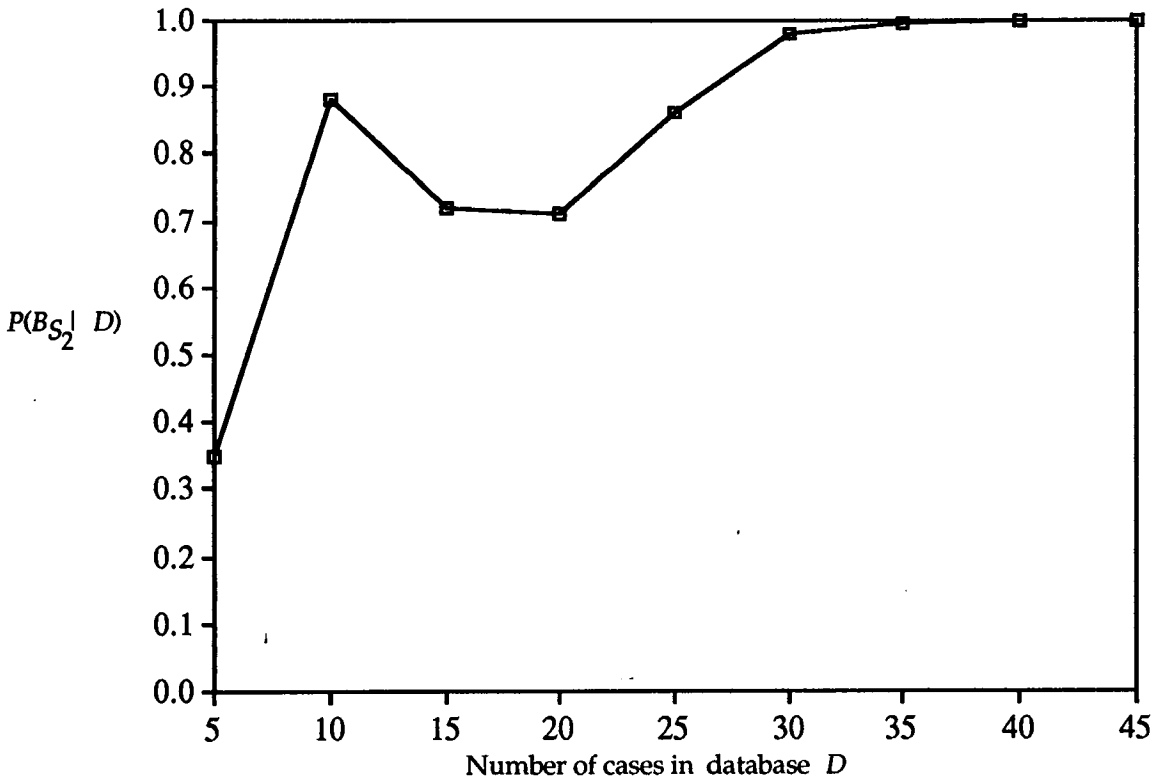

Figure 3. A plot of the posterior probability of belief-network structure B_{S_2} as D is set to be the first 5, 10, 15, ... , and 45 cases in database D^*. Lines are drawn between data points to highlight changes between those points.

7. Extensions and Open Problems

In this section we briefly discuss several possible extensions to the MB method. We can improve the efficiency of computing Equation 7 by moving some terms to the outside of the sums. This becomes more apparent if Equation 7 is rewritten with the integral replaced by its solution (as given by Equation 1):

$$P(D \mid B_S) = \sum_{G_1} \cdots \sum_{G_u} f(G_1, \ldots, G_u) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \,. \tag{9}$$

If a node x_i is not equal to h and is not a child of h, then result of the products is not influenced by the settings made in the sums, and thus we can move these products in front of the sums, and calculate them only once. In this situation, Equation 9 becomes

$$P(D \mid B_S) = \left[\prod_{i \in Z - C^*(h)} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right] \left[\sum_{G_1} \cdots \sum_{G_u} f(G_1, \ldots, G_u) \prod_{i \in C^*(h)} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \right] \tag{10}$$

where $C^*(h)$ is a set containing just h and the children of h.

If there is a hidden variable in the set $Z - C^*(h)$ that is not in the Markov blanket of h, then we can apply Equation 7 to compute the products that appear in the first set of square brackets in Equation 10. This process of successively moving products out of sums can be repeated for any number of hidden variables, as long as none of the hidden variables have intersecting Markov boundaries. If the Markov boundaries of two or more hidden variables do have a non-empty intersection, then we can in effect treat these variables like a single hidden variable (of higher dimension), and apply Equation 7 as before. This technique therefore provides a general approach for handling multiple hidden variables. The technique can, however, become computationally demanding, and thus, it is an open problem to discover more efficient methods for handling multiple variables that are hidden or that have some missing data.

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. This is a difficult, open problem for which we outline here several approaches. One way to address the problem is simply to limit the number of hidden variables in the belief networks that we postulate, and then use a greedy search method similar to K2 [8] to try to locate a highly probable belief-network structure that contains hidden variables. Another approach is to specify explicitly nonzero priors for only a limited number of belief-network structures that contain hidden variables. This may be possible if we have knowledge that tightly constrains a priori the possible set of structures. One other possible approach is to use non-Bayesian methods to suggest likely locations for hidden variables, as discussed in [22, 23, 26]. We then could begin our search using Bayesian methods by initially postulating hidden variables where the non-Bayesian methods suggest they might exist.

Another 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 [5]. We believe that similar methods may prove useful in addressing the problem of learning the number of values of hidden variables in belief networks.

Missing values in the database can be viewed as partially hidden variables. We can use a slight variation of Equation 7 to handle missing values for a given variable h that is partially hidden. Since the number of missing values of a variable is typically much less than the total number of cases, handling missing values will usually be less computationally demanding than handling hidden variables, for the methods discussed in this paper. By Assumption 4 in Theorem 1, for a given belief-network structure B_S, if a variable x has a missing value in a case, then x is believed to be just as likely to have one value as any other. If this assumption is not valid, then Assumption 4 should be modified to allow the expression of more general distributions of missing values, as for example, by using Dirichlet distributions [8].

Acknowledgements

Support was provided by the National Science Foundation under grant IRI-9111590.

References

1. Agogino, A.M. and Rege, A., IDES: Influence diagram based expert system, *Mathematical Modelling* 8 (1987) 227–233.
2. Andreassen, S., Woldbye, M., Falck, B. and Andersen, S.K., MUNIN — A causal probabilistic network for interpretation of electromyographic findings, In: *Proceedings of the International Joint Conference on Artificial Intelligence*, Milan, Italy (1987) 366–372.
3. Beinlich, I.A., Suermondt, H.J., Chavez, R.M. and Cooper, G.F., The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, In: *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, London, England (1989) 247–256.

4. Chavez, R.M. and Cooper, G.F., KNET: Integrating hypermedia and normative Bayesian modeling. In: Shachter R.D., Levitt T.S., Kanal L.N. and Lemmer J.F. (Ed.), *Uncertainty in Artificial Intelligence 4* (North-Holland, Amsterdam, 1990) 339–349.
5. Cheeseman, P., Self, M., Kelly, J., Taylor, W., Freeman, D. and Stutz, J., Bayesian classification, In: *Proceedings of AAAI*, St. Paul, MN (1988) 607–611.
6. Cooper, G.F., *NESTOR: A Computer-Based Medical Diagnostic Aid that Integrates Causal and Probabilistic Knowledge*, Doctoral dissertation, Medical Information Sciences, Stanford University (1984).
7. Cooper, G.F., Current research directions in the development of expert systems based on belief networks, *Applied Stochastic Models and Data Analysis* **5** (1989) 39–52.
8. Cooper, G.F. and Herskovits, E., A Bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9** (1992) 309-347.
9. Heckerman, D.E., Horvitz, E.J. and Nathwani, B.N., Toward normative expert systems: Part I. The Pathfinder project., *Methods of Information in Medicine* **31** (1992) 90-105.
10. Henrion, M., Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer J.F. and Kanal L.N. (Eds.), *Uncertainty in Artificial Intelligence 2* (North-Holland, Amsterdam, 1988) 149–163.
11. Henrion, M., Some practical issues in constructing belief networks. In: Kanal L.N., Levitt T.S. and Lemmer J.F. (Eds.), *Uncertainty in Artificial Intelligence 3* (North-Holland, Amsterdam, 1989) 161–173.
12. Henrion, M., An introduction to algorithms for inference in belief nets. In: Henrion M., Shachter R.D., Kanal L.N. and Lemmer J.F. (Eds.), *Uncertainty in Artificial Intelligence 5* (North-Holland, Amsterdam, 1990) 129–138.
13. Henrion, M. and Cooley, D.R., An experimental comparison of knowledge engineering for expert systems and for decision analysis, In: *Proceedings of AAAI*, Seattle, WA (1987) 471–476.
14. Herskovits, E.H., *Computer-Based Probabilistic-Network Construction*, Doctoral dissertation, Medical Information Sciences, Stanford University (1991).
15. Holtzman, S., *Intelligent Decision Systems* (Addison-Wesley, Reading, MA, 1989).
16. Kiiveri, H., Speed, T.P. and Carlin, J.B., Recursive causal models, *Journal of the Australian Mathematical Society* **36** (1984) 30–52.
17. Neapolitan, R., *Probabilistic Reasoning in Expert Systems* (John Wiley & Sons, New York, 1990).
18. Pearl, J., *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, San Mateo, CA, 1988).
19. Shachter, R.D., Intelligent probabilistic inference. In: Kanal L.N. and Lemmer J.F. (Eds.), *Uncertainty in Artificial Intelligence* (North-Holland, Amsterdam, 1986) 371–382.
20. Spiegelhalter, D.J. and Cowell, R.G., Learning in probabilistic expert systems. In: Bernardo J.M., Berger J.O., Dawid A.P. and Smith A.F.M. (Eds.), *Bayesian Statistics 4* (Oxford University Press, Oxford, 1992) 1–17.
21. Spiegelhalter, D.J. and Lauritzen, S.L., Sequential updating of conditional probabilities on directed graphical structures, *Networks* **20** (1990) 579–606.
22. Spirtes, P. and Glymour, C., Causal structure among measured variables preserved with unmeasured variables, Report CMU-LCL-90-5, Department of Philosophy, Carnegie–Mellon University, 1990.
23. Spirtes, P., Glymour, C. and Scheines, R., *Causality, Prediction, and Search* (Springer-Verlag, New York, 1993).
24. Suermondt, H.J. and Amylon, M.D., Probabilistic prediction of the outcome of bone-marrow transplantation, In: *Proceedings of the Symposium on Computer Applications in Medical Care*, Washington, DC (1989) 208–212.
25. Tucker, A., *Applied Combinatorics* (John Wiley & Sons, New York, 1984).
26. Verma, T.S. and Pearl, J., Equivalence and synthesis of causal models, In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Cambridge, Massachusetts (1990) 220–227.
27. York, J. and Madigan, D., Markov chain Monte Carlo methods for hierarchical Bayesian expert systems, In: *Proceedings of the Conference on Artificial Intelligence and Statistics*, Orlando, Florida (1993) 433–439.