11 Bayesian Networks for Knowledge Discovery

David Heckerman
Microsoft Research

Abstract

We examine a graphical representation of uncertain knowledge called a Bayesian network. The representation is easy to construct and interpret, yet has formal probabilistic semantics making it suitable for statistical manipulation. We show how we can use the representation to discover new knowledge by combining domain knowledge and statistical data.

11.1 Introduction

Many techniques for knowledge discovery rely solely on data. In contrast, the knowledge encoded in expert systems usually comes solely from an expert. In this chapter, we examine a knowledge representation, called a *Bayesian network,* that lets us have the best of both worlds. Namely, the representation allows us to discover new knowledge by combining expert domain knowledge with statistical data.

A Bayesian network is a graphical representation of uncertain knowledge that most people find easy to construct and interpret. In addition, the representation has formal probabilistic semantics, making it suitable for statistical manipulation (Howard, 1981; Pearl, 1988). Over the last decade, the Bayesian network has become a popular representation for encoding uncertain expert knowledge in expert systems (Heckerman et al., 1995a). More recently, researchers have developed methods for learning Bayesian networks from a combination of expert knowledge and data. The techniques that have been developed are new and still evolving, but they have been shown to be remarkably effective in some domains (Cooper and Herskovits 1992; Aliferis and Cooper 1994; Heckerman et al. 1995b).

Using Bayesian networks, the knowledge discovery process goes as follows. First, we encode the existing knowledge of an expert or set of experts in a Bayesian network, as is done when building a probabilistic expert system. Then, we use a database to update this knowledge, creating one or more new Bayesian networks. The result includes a refinement of the original expert knowledge and sometimes the identification of new

distinctions and relationships. The approach is robust to errors in the knowledge of the expert. Even when expert knowledge is unreliable and incomplete, we can often use it to improve the discovery process.

Knowledge discovery using Bayesian networks is similar to that using neural networks. The process employing Bayesian networks, however, has two important advantages. One, we can easily encode expert knowledge in a Bayesian network and use this knowledge to increase the efficiency and accuracy of knowledge discovery. Two, the nodes and arcs in learned Bayesian networks often correspond to recognizable distinctions and causal relationships. Consequently, we can more easily interpret and understand the knowledge encoded in the representation.

This chapter is a brief tutorial on Bayesian networks and methods for learning them from data. In Section 11.2, we discuss the representation itself. In Sections 11.3 through 11.6, we describe methods for learning the probabilities and structure of a Bayesian network. In Sections 11.7 and 11.8, we discuss methods for identifying new distinctions about the world and integrating these distinctions into a Bayesian network. We restrict our discussion to Bayesian and quasi-Bayesian methods for learning. An interesting and often effective non-Bayesian approach is given by Pearl and Verma (1991) and Spirtes et al. (1993). Also, we limit our discussion to problem domains where variables take on discrete states. More general techniques are given in Buntine (1994) and Heckerman and Geiger (1994). Finally, the interested reader may wish to read the companion article in this volume by Buntine, which provides additional examples of the Bayesian-network representation as well as an introduction to some of the more complex issues associated with learning Bayesian networks.

11.2 Bayesian Networks

Before we discuss methods for learning Bayesian networks, let us examine the Bayesian philosophy and the representation itself. A primary element of the language of probability (Bayesian or otherwise) is the event. By *event*, we mean a state of some part of our world in some time interval in the past, present, or future. A classic example of an event is that a particular flip of a coin will come up heads. A perhaps more interesting event is that gold will close at more than $400 per ounce

on January 1, 2001.

Given an event e, the prevalent conception of its probability is that it is a measure of the frequency with which e occurs, when we repeat many times an experiment with possible outcomes e and $\bar{e}$ (not e). A different notion is that the probability of e represents the *degree of belief* held by a person that the event e will occur in a single experiment. The interpretation of a probability as a frequency in a series of repeat experiments is traditionally referred to as the *objective* or *frequentist* interpretation. In contrast, the interpretation of a probability as a degree of belief is called the *subjective* or *Bayesian* interpretation.

In the Bayesian interpretation, a probability or belief will always depend on the state of knowledge of the person who provides that probability. Thus, we write the probability of e as $p(e|\xi)$, which is read as the probability of e *given* ξ. The symbol ξ represents the state of knowledge of the person who provides the probability. Also, in this interpretation, a person can assess a probability based on information that he *assumes* to be true. We write $p(e_2|e_1, \xi)$ to denote the probability of event e_2 *given* that event e_1 is true and given background knowledge ξ.

A *variable* represents a distinction about the world. It takes on values from a collection of mutually exclusive and collectively exhaustive states, where each state corresponds to some event. A variable may be *discrete*, having a finite or countable number of states, or it may be *continuous*. For example, a two-state or *binary* variable can be used to represent the possible outcomes of a coin flip; whereas a continuous variable can be used to represent the weight of the coin. In this chapter, we use lower-case letters (usually near the end of the alphabet) to represent single variables and upper-case letters to represent sets of variables. We write $x = k$ to denote that variable x is in state k. When we observe the state for every variable in set X, we call this set of observations a state of X, and write $X = k$. Sometimes, we leave the state of a variable or set of variables implicit. The *probability distribution over a set of variables* X, denoted $p(X|\xi)$, is the set of probabilities $p(X = k|\xi)$ for all states of X. Given sets of variables X and Y, we write $p(X|Y, \xi)$ to denote the set probability distributions $p(X|Y = k, \xi)$ for all states of Y.

A Bayesian network is a person's model for some *problem domain* or *universe*, which consists of a set of variables. A Bayesian network for the domain $U = \{x_1, \ldots, x_n\}$ represents the joint probability distribution $p(U|\xi)$. The representation consists of a set of *local* conditional prob-

ability distributions, combined with a set of assertions of conditional independence that allow us to construct the global joint distribution from the local distributions.

To illustrate the representation, let us consider the domain of troubleshooting a car that won't start. The first step in constructing a Bayesian network is to decide what variables and states to model. One possible choice of variables for this domain is *Battery* (b) with states good and bad, *Fuel* (f) with states not empty and empty, *Gauge* (g) with states not empty and empty, *Turn Over* (t) with states yes and no, and *Start* (s) with states yes and no. Of course, we could include many more variables (as we would in a real-world example). Also, we could model the states of one or more of these variables at a finer level of detail. For example, we could let *Gauge* be a continuous variable with states ranging from 0% to 100%.

The second step in constructing a Bayesian network is to construct a directed acyclic graph that encodes assertions of conditional independence. We call this graph the *Bayesian-network structure*. Given a domain $U = \{x_1, \ldots, x_n\}$ and an ordering on the variables $(x_1 \ldots, x_n)$, we can write the joint probability distribution of U using the chain rule of probability as follows:

$$p(x_1, \ldots, x_n|\xi) = \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1}, \xi). \tag{11.2.1}$$

Now, for every x_i, there will be some subset $\Pi_i \subseteq \{x_1, \ldots, x_n\}$ such that x_i and $\{x_1, \ldots, x_n\}$ are conditionally independent given Π_i. That is,

$$p(x_i|x_1, \ldots, x_{i-1}, \xi) = p(x_i|\Pi_i, \xi) \tag{11.2.2}$$

These conditional independencies define the Bayesian-network structure. The nodes in the structure correspond to variables in the domain. The parents of x_i correspond to the set Π_i. In our example, using the ordering (b, f, g, t, s), we have the conditional independencies

$$\begin{aligned} p(f|b, \xi) &= p(f|\xi) \\ p(t|b, f, g, \xi) &= p(t|b, \xi) \\ p(s|b, f, g, t, \xi) &= p(s|f, t, \xi) \end{aligned} \tag{11.2.3}$$

Consequently, we obtain the structure shown in Figure 11.1.

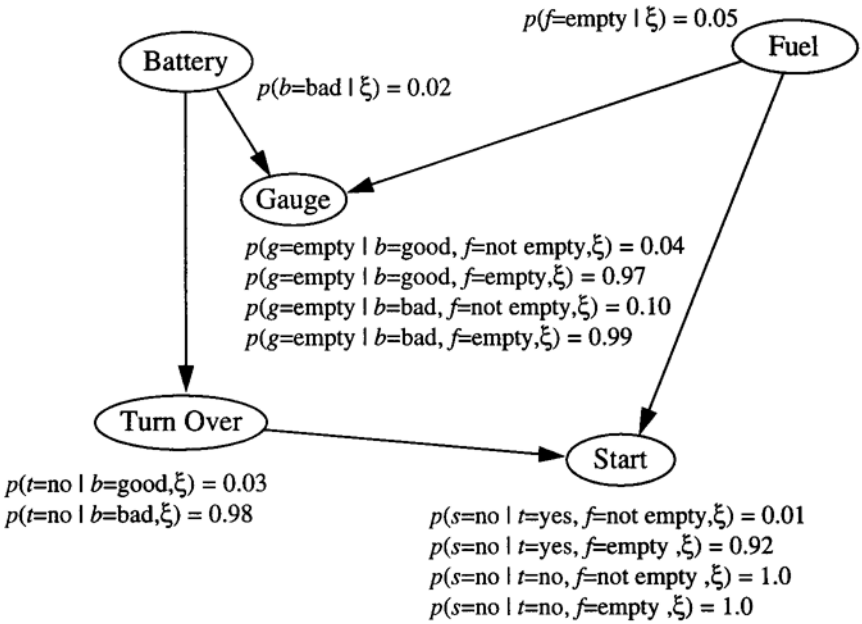

Figure 11.1
A Bayesian-network for troubleshooting a car that won't start. Arcs are drawn from cause to effect. The local probability distribution(s) associated with a node are shown adjacent to the node.

The final step in constructing a Bayesian network is to assess the local distributions $p(x_i|\Pi_i, \xi)$—one distribution for every state of Π_i. These distributions for our automobile example are shown in Figure 11.1. Combining Equations 11.2.1 and 11.2.2, we see that a Bayesian network for U always encodes the joint probability distribution for U.

The problem of computing probabilities of interest from a (possibly implicit) joint probability distribution is called *probabilistic inference.* Given any Bayesian network, we can use the joint probability distribution determined by that network to do probabilistic inference. For example, suppose we want to compute the probability distribution of *Fuel* given that the car doesn't start. From the rules of probability we have

$$p(f|s = no, \xi) = \frac{p(f, s = no|\xi)}{p(s = no|\xi)} = \frac{\sum_{b,g,t} p(b, f, g, t, s = no|\xi)}{\sum_{b,f,g,t} p(b, f, g, t, s = no|\xi)} \quad (11.2.4)$$

In a real-world problem with n variables, this approach is not feasible, because it entails summing over 2^n or more terms. Fortunately, we can

exploit the conditional independencies encoded in a Bayesian network to make this probabilistic inference more efficient; and several algorithms exist for doing so.

A drawback of Bayesian networks as defined is that network structure depends on variable order. If the order is chosen carelessly, the resulting network structure may fail to reveal many conditional independencies in the domain. As an exercise, the reader should construct a Bayesian network for the automobile domain using the ordering (s, t, g, f, b). Fortunately, in practice, we can usually assert causal relationships among variables in a domain, and can use these assertions to construct a Bayesian-network structure without preordering the variables. Namely, to construct a Bayesian network for a given set of variables, we draw arcs from cause variables to their immediate effects. In almost all cases, doing so results in a Bayesian network whose conditional-independence implications are accurate. For example, the network in Figure 11.1 was constructed using the assertions that *Gauge* is the direct causal effect of *Battery* and *Fuel*, *Turn Over* is the direct causal effect of *Battery*, and *Start* is the direct causal effect of *Turn Over* and *Fuel*.

In large part, it is this connection between causal relationships and conditional independence that has made the representation attractive as a modeling tool to expert-system engineers. Nonetheless, there are times when expert knowledge is unreliable, incomplete, or is difficult to obtain. It this case, we can use data to update or learn the probabilities and structure of a Bayesian network. In the remainder of this tutorial, we consider this task.

11.3 Learning Probabilities: The One-Variable Case

Because Bayesian networks have a probabilistic interpretation, we can use traditional techniques from Bayesian statistics to learn these models from data. Several of the techniques that we need can be discussed in the context of learning the probability distribution of a single variable. In this section, we examine this case.

Consider a common thumbtack—one with a round, flat head that can be found in most supermarkets. If we throw the thumbtack up in the air and let in land on a hard, flat surface, it will come to rest either on its

point (*heads*) or on its head (*tails*), as shown in Figure 11.2a.[1] Suppose we give the thumbtack to someone, who then flips the thumbtack many times and measures the fraction of flips that it comes up heads. A frequentist would say this long-run fraction is a probability, and would observe flips of the thumbtack to estimate this probability. In contrast, from the Bayesian perspective, we recognize the possible values of this fraction as a variable—call it θ—whose true value is uncertain. We express our uncertainty about θ with a probability distribution $p(\theta|\xi)$, and update this distribution as we observe flips of the thumbtack.

We note that, although θ is not a degree of belief, collections of long-run fractions like θ satisfy the rules of probability. In this chapter, we shall refer to θ as a *physical probability* (after Good, 1959) to distinguish it from a degree of belief.[2] Figure 11.2 shows one possible probability distribution for θ.

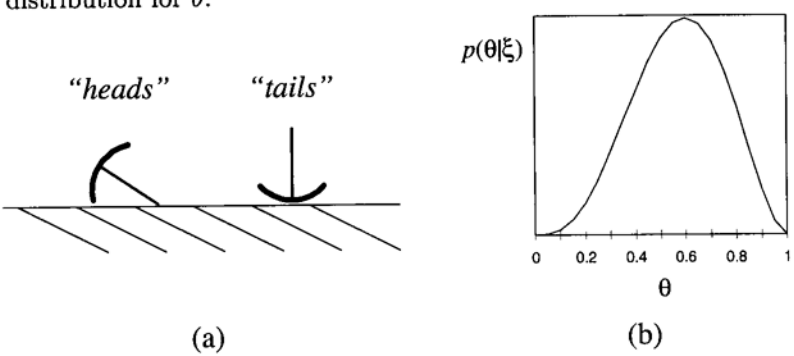

Figure 11.2
(a) The outcomes of a thumbtack flip. (b) A probability distribution for θ, the long-run fraction of heads associated with a thumbtack.

Now suppose we have a database $D = \{x_1, \ldots, x_m\}$ of outcomes of flipping the thumbtack. If we knew the value of θ, then our probability for heads on any flip would be equal to θ, no matter how many outcomes we observe. That is,

$$p(x_l = heads|\theta, x_1, \ldots, x_{l-1}, D, \xi) = \theta \tag{11.3.5}$$

[1]This example is taken from Howard (1970).
[2]The variable θ is also referred to as a frequency, objective probability, and true probability.

where x_l is the outcome of the lth flip of the thumbtack. Similarly, we have

$$p(x_l = tails|\theta, x_1, \ldots, x_{l-1}, D, \xi) = 1 - \theta \tag{11.3.6}$$

In particular, the outcomes are mutually independent given θ.

In reality, we are uncertain about the value of θ. In this case, we can use the expansion rule of probability to determine our probability that the next toss of the thumbtack will come up heads:

$$p(x = heads|\xi) = \int p(x = heads|\theta, \xi)\ p(\theta|\xi)\ d\theta = \int \theta\ p(\theta|\xi)\ d\theta \equiv E(\theta|\xi)$$

where $E(\theta|\xi)$ denotes the expectation of θ with respect to the distribution $p(\theta|\xi)$. That is, our probability for heads on the next toss is just the expectation of θ. Furthermore, suppose we flip the thumbtack once and observe heads. Using Bayes' theorem, the posterior probability distribution for θ becomes

$$p(\theta|x = heads, \xi) = c\ p(x = heads|\theta, \xi)\ p(\theta|\xi) = c\ \theta\ p(\theta|\xi)$$

where c is some normalization constant. That is, we obtain the posterior distribution for θ by multiplying its prior distribution by the function $f(\theta) = \theta$ and renormalizing. This procedure is depicted graphically in Figure 11.3. As expected, the posterior is shifted to the right and is slightly narrower. In general, if we observe h heads and t tails in the database D, then we have

$$p(\theta|t\ heads, h\ tails, \xi) = c\ \theta^h (1 - \theta)^t\ p(\theta|\xi)$$

That is, once we have assessed a prior distribution for θ, we can determine its posterior distribution given any possible database. Note that the order in which we observe the outcomes is irrelevant to the posterior—all that is relevant is the number of heads and the number of tails in the database. We say that h and t are a *sufficient statistic* for the database.

In this simple example, our outcome variable has only two states (*heads* and *tails*). Now, imagine we have a discrete outcome variable x with $r \geq 2$ states. For example, this variable could represent the outcome of a roll of a loaded die ($r = 6$). We denote the physical probabilities of the outcomes $\Theta_x = \{\theta_{x=1}, \ldots, \theta_{x=r}\}$, and assume that each state

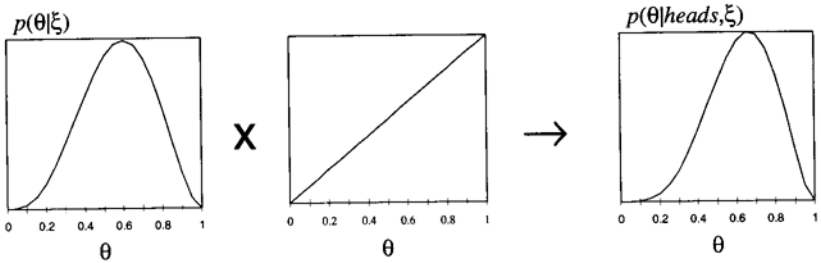

Figure 11.3
A graphical depiction of the use of Bayes' theorem to compute the posterior probability distribution of the physical probability θ.

is possible so that each $\theta_{x=k} > 0$. In addition, we have $\sum_{k=1}^{r} \theta_{x=k} = 1$. Also, if we know these physical probabilities, then the outcome of each "toss" of x will be conditionally independent of the other tosses, and

$$p(x_l = k | x_1, \ldots, x_{l-1}, \Theta_x, \xi) = \theta_{x=k} \tag{11.3.7}$$

Any database of outcomes $\{x_1, \ldots, x_m\}$ that satisfies these conditions is called an $(r-1)$-*dimensional multinomial sample with physical probabilities* Θ_x (Good, 1965). When $r = 2$, as in the thumbtack example, the sequence is said to be a *binomial sample*. The concept of a multinomial sample (and its generalization, the random sample) will be central to the remaining discussions in this chapter.

Analogous to the thumbtack example, we have

$$p(x = k|\xi) = \int \theta_{x=k}\, p(\Theta_x|\xi)\, d\Theta_x \equiv E(\theta_{x=k}|\xi) \tag{11.3.8}$$

where $p(x = k|\xi)$ is our probability that $x = k$ in the next case. Note that, because $\sum_{k=1}^{r} \theta_{x=k} = 1$, the distribution for Θ_x is technically a distribution over the variables $\Theta_x \setminus \{\theta_{x=k}\}$ for some k (the symbol $\setminus$ denotes set difference). Also, given any database D of outcomes, we have

$$p(\Theta_x|D, \xi) = c \cdot \prod_{k=1}^{r} \theta_{x=k}^{N_k}\, p(\Theta_x|\xi) \tag{11.3.9}$$

where N_k is the number of times $x = k$ in D, and c is a normalization constant. Note that the counts $N_1, \ldots, N_r$ are a sufficient statistic for the multinomial sample.

Given a multinomial sample, a user is free to assess any probability distribution for Θ_x. In practice, however, one often uses the Dirichlet distribution because it has several convenient properties. The variables Θ_x are said to have a *Dirichlet distribution with exponents* $N'_1, \ldots, N'_r$ when the probability distribution of Θ_x is given by

$$p(\Theta_x|\xi) = \frac{\Gamma(\sum_{k=1}^r N'_k)}{\prod_{k=1}^r \Gamma(N'_k)} \prod_{k=1}^r \theta_{x=k}^{N'_k - 1}, \quad N'_k > 0 \tag{11.3.10}$$

where $\Gamma(\cdot)$ is the *Gamma* function, which satisfies $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(1) = 1$. When the variables Θ_x have a Dirichlet distribution, we also say that $p(\Theta_x|\xi)$ *is Dirichlet.* The exponents N'_k must be greater than 0 to guarantee that the distribution can be normalized. Note that the exponents N'_k are a function of the user's state of information ξ. When $r = 2$, the Dirichlet distribution is also known as a *beta distribution.* The probability distribution on the left-hand-side of Figure 11.3 is a beta distribution with exponents $N'_{heads} = 3$ and $N'_{tails} = 2$. The probability distribution on the right-hand-side of the figure is a beta distribution with exponents $N'_{heads} = 4$ and $N'_{tails} = 2$.

From Equation 11.3.9, we see that if the prior distribution of Θ_x is Dirichlet, then the posterior distribution of Θ_x given database $D = \{x_1, \ldots, x_m\}$ is also Dirichlet:

$$p(\Theta_x|D, \xi) = c \prod_{k=1}^r \theta_{x=k}^{N'_k + N_k - 1} \tag{11.3.11}$$

We say that the Dirichlet distribution is closed under multinomial sampling, or that the Dirichlet distribution is a *conjugate family of distributions* for multinomial sampling. Also, when Θ_x has a Dirichlet distribution, the expectation of $\theta_{x=k}$ with respect to this distribution—equal to the probability that $x = k$ in the first observation—has a simple expression:

$$E(\theta_{x=k}|\xi) = p(x = k|\xi) = \frac{N'_k}{N'} \tag{11.3.12}$$

where $N' = \sum_{k=1}^r N'_k$. As we shall see, these properties make the Dirichlet a useful prior for learning.

According to Equation 11.3.12, we can assess a Dirichlet distribution for Θ_x by assessing the probability distribution $p(x|\xi)$ for the next observation and N'. The number N' is sometimes called an *equivalent*

sample size for a Dirichlet distribution, because it is equal to the number of observations we would have to make starting from complete ignorance (each N'_k very close to zero) in order to arrive at that distribution. Note that N' is a measure of a user's confidence in the values of Θ_x: the larger the value of N', the more certain the user is about the values.

So far, we have only considered a variable with discrete outcomes. In general, we can imagine a physical probability distribution over a variable (discrete or continuous) from which database cases are drawn at random. This physical probability distribution typically can be characterized by a finite set of *parameters*. If the outcome variable is discrete, then the physical probability distribution has a parameter corresponding to each physical probability in the distribution (and, herein, we sometimes refer to these physical probabilities as parameters). If the outcome variable is continuous, the physical probability distribution may be (e.g.) a normal distribution. In this case, the parameters would be the mean and variance of the distribution. A database of cases drawn from a physical probability distribution is often called a *random sample*.

Given such a physical probability distribution with unknown parameters, we can update our beliefs about these parameters given a random sample from this distribution using techniques similar to those we have discussed. For random samples from many named distributions—including normal, Gamma, and uniform distributions—there exist corresponding conjugate priors that offer convenient properties for learning probabilities similar to those properties of the Dirichlet. These priors are in the *exponential family*. The reader interested in learning about these distributions should read DeGroot (1970, Chapter 9).

11.4 Learning Probabilities: Known Structure

The notion of a random sample generalizes to domains containing more than one variable as well. Given a domain $U = \{x_1, \ldots, x_n\}$, we can imagine a multivariate physical probability distribution for U. If U contains only discrete variables, this distribution is just a finite collection of discrete physical probabilities. If U contains only continuous variables, this distribution could be (e.g.) a multivariate-normal distribution characterized by a mean vector and covariance matrix. Using conjugate priors for the parameters of such distributions, such as those discussed in

DeGroot (1970), we can update our priors about the parameters of these distributions given a database.

Now, however, let us consider the following wrinkle. Suppose we know that this multivariate physical probability distribution can be encoded in some particular Bayesian-network structure B_S. We may have gotten this information—for example—from our causal knowledge about the domain. In this section, we consider the task of learning the parameters of B_S. We discuss only the special case where all the variables in U are discrete and where the random sample (i.e., database) $D = \{C_1, \ldots, C_m\}$ contains no missing data—that is, each case C_l consists of the observation of *all* the variables in U (we say that D is *complete*). In Section 11.7, we consider the more difficult problem where D contains missing data. Buntine (1994) and Heckerman and Geiger (1994) discuss the case where U may contain continuous variables.

When a database D is a random sample from a multivariate physical probability distribution that can be encoded in B_S, we simply say that D is a random sample from B_S. As an example, consider the domain U consisting of two binary variables x and y. Let $\theta_{xy}, \theta_{x\bar{y}}, \theta_{\bar{x}y}$, and $\theta_{\bar{x}\bar{y}}$ denote the parameters (i.e., physical probabilities) for the joint space of U, where $\theta_{x\bar{y}}$ is the physical probability of the event where x is true and y is false, and so on. (Note that, in using the overbar, we are departing from our standard notation.) Then, saying that D is a random sample from the network structure containing no arc between x and y, is the assertion that the parameters of the joint space satisfy the independence constraints $\theta_{xy} = \theta_x\theta_y, \theta_{x\bar{y}} = \theta_x\theta_{\bar{y}}$, and so on, where—for example—$\theta_x = \theta_{xy} + \theta_{x\bar{y}}$ is the physical probability associated with the event where x is true. It is not difficult to show that this assertion is equivalent to the assertion that the database D can be decomposed into two multinomial samples: the observations of x are a multinomial sample with parameter θ_x, and the observations of y are a multinomial sample with parameter θ_y.

As another example, suppose we assert that a database for our two variable domain is a random sample from the network structure $x \rightarrow y$. Here, there are no constraints on the parameters of the joint space. Furthermore, this assertion implies that the database is made up of at most three binomial samples: (1) the observations of x are a binomial sample with parameter θ_x, (2) the observations of y in those cases (if any) where x is true are a binomial sample with parameter $\theta_{y|x}$, and (3) the

observations of y in those cases (if any) where x is false are a binomial sample with parameter $\theta_{y|\bar{x}}$. Consequently, the occurrences of x in D are conditionally independent given θ_x, and y in case C is conditionally independent of the other cases in D given $\theta_{y|x}$, $\theta_{y|\bar{x}}$, and x in case C. We can graphically represent the conditional-independence assertions associated with these random samples using a Bayesian-network structure as shown in Figure 11.4a.

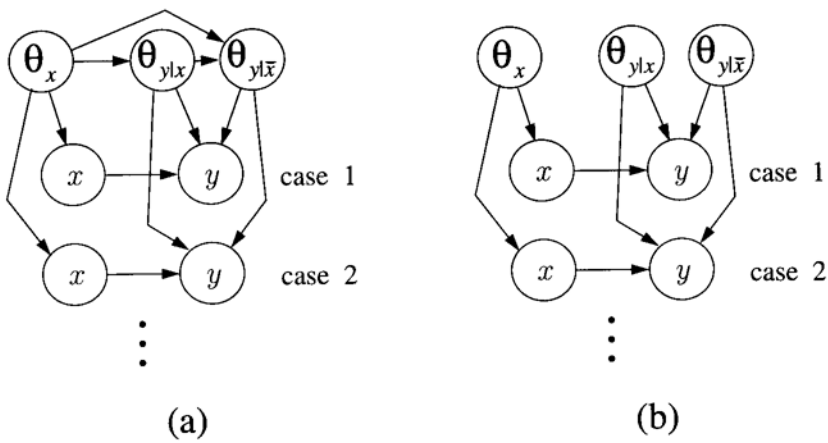

Figure 11.4
(a) A Bayesian-network structure for a two-binary-variable domain $\{x, y\}$ showing conditional independencies associated with the assertion that the database is a random sample from the structure $x \rightarrow y$. (b) Another Bayesian-network structure showing the added assumption of parameter independence.

Given the collection of random samples shown in Figure 11.4a, it is tempting to apply our one-variable techniques to learn each parameter separately. Unfortunately, this approach is not correct when the parameters are dependent as shown in the figure. For example, as we see occurrences of x and update our beliefs about θ_x, our beliefs about $\theta_{y|x}$ and $\theta_{y|\bar{x}}$ will also change. Suppose, however, that all of the parameters are independent, as shown in Figure 11.4b. Then, provided the database is complete, we can update each parameter separately.

In the remainder of this section, we shall assume that all parameters are independent. We call this assumption—introduced by Spiegelhalter and Lauritzen (1990)—*parameter independence*. In Section 11.7, we

discuss methods for handling dependent parameters.

To complete the discussion, we need some notation. Let B_S^h denote the assertion (or hypothesis) that a database D is a random sample from a Bayesian network structure B_S. Given B_S, let r_i be the number of states of variable x_i; and let $q_i = \prod_{x_l \in \Pi_i} r_l$ be the number of states of Π_i. Let θ_{ijk} denote the physical probability of $x_i = k$ given $\Pi_i = j$ for $i = 1, \ldots, n$, $j = 1, \ldots, q_i$, $k = 1, \ldots, r_i$. In addition, let

$$\Theta_{ij} \equiv \cup_{k=1}^{r_i} \{\theta_{ijk}\} \qquad \Theta_{B_S} \equiv \cup_{i=1}^{n} \cup_{j=1}^{q_i} \Theta_{ij}$$

Note that the parameters Θ_{B_S} in conjunction with B_S determine all the physical probabilities of the joint space.

Let us assume that each variable set Θ_{ij} has a Dirichlet distribution:

$$p(\Theta_{ij}|B_S^h, \xi) = c \cdot \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}-1} \tag{11.4.13}$$

where c is a normalization constant. Then, if N_{ijk} is the number of cases in database D in which $x_i = k$ and $\Pi_i = j$, we obtain

$$p(\Theta_{ij}|D, B_S^h, \xi) = c \cdot \prod_{k=1}^{r_i} \theta_{ijk}^{N'_{ijk}+N_{ijk}-1} \tag{11.4.14}$$

where c is some other normalization constant. Furthermore, applying Equation 11.3.12 to each multinomial sample, we can compute the probability that $x_i = k$ and $\Pi_i = j$ in C_{m+1}, the next case to be seen after the database:

$$p(C_{m+1}|D, B_S^h, \xi) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{N'_{ijk} + N_{ijk}}{N'_{ij} + N_{ij}} \tag{11.4.15}$$

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

11.5 Learning Structure

In the previous section, we considered the situation where we are uncertain about the physical probabilities, but certain about the network structure that encodes these probabilities. Now, suppose we are not only uncertain about the probabilities, but also uncertain about the structure that encodes them. As with any set of events, we can express this uncertainty by assigning a prior probability $p(B_S^h|\xi)$ to each possible

hypothesis B_S^h. Furthermore, we can update these probabilities as we see cases. In so doing, we learn about the structure of the domain.

As in the previous section, let B_S^h denote the (now uncertain) hypothesis that the database D is a random sample from the Bayesian network structure B_S. From Bayes' theorem, we have

$$p(B_S^h|D,\xi) = c\, p(B_S^h|\xi)\; p(D|B_S^h,\xi) \tag{11.5.16}$$

where c is a normalization constant. Also, from the product rule, we have

$$p(D|B_S^h,\xi) = \prod_{l=1}^{m} p(C_l|C_1,\ldots,C_{l-1},B_S^h,\xi) \tag{11.5.17}$$

We can evaluate each term on the right-hand-side of this equation using Equation 11.4.15, under the assumption that the database D is complete. For the posterior probability of B_S^h given D, we obtain

$$\begin{aligned} p(B_S^h|D,\xi) = \\ c\cdot p(B_S^h|\xi)\cdot \prod_{i=1}^{n}\prod_{j=1}^{q_i}\left\{\left[\frac{N'_{ij1}}{N'_{ij}}\cdot\frac{N'_{ij1}+1}{N'_{ij}+1}\cdots\frac{N'_{ij1}+N_{ij1}-1}{N'_{ij}+N_{ij1}-1}\right]\cdot\right. \\ \left[\frac{N'_{ij2}}{N'_{ij}+N_{ij1}}\cdot\frac{N'_{ij2}+1}{N'_{ij}+N_{ij1}+1}\cdots\frac{N'_{ij2}+N_{ij2}-1}{N'_{ij}+N_{ij1}+N_{ij2}-1}\right]\cdots \\ \left.\left[\frac{N'_{ijr_i}}{N'_{ij}+\sum_{k=1}^{r_i-1}N_{ijk}}\cdot\frac{N'_{ijr_i}+1}{N'_{ij}+\sum_{k=1}^{r_i-1}N_{ijk}+1}\cdots\frac{N'_{ijr_i}+N_{ijr_i}-1}{N_{ij}+N'_{ij}-1}\right]\right\} = \\ c\cdot p(B_S^h|\xi)\cdot\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})}\cdot\prod_{k=1}^{r_i}\frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})} \end{aligned} \tag{11.5.18}$$

Using these posterior probabilities and Equation 11.4.15, we may compute the probability distribution for the next case to be observed after we have seen a database. From the expansion rule, we obtain

$$p(C_{m+1}|D,\xi) = \sum_{B_S^h} p(C_{m+1}|D,B_S^h,\xi)\; p(B_S^h|D,\xi) \tag{11.5.19}$$

There are two important points to be made about this approach. One, it can happen that two Bayesian-network structures represent exactly

the same sets of probability distributions. We say that the two structures are *equivalent* (Verma and Pearl, 1990). For example, for the three variable domain $\{x, y, z\}$, each of the network structures $x \rightarrow y \rightarrow z$, $x \leftarrow y \rightarrow z$, and $x \leftarrow y \leftarrow z$ represents the distributions where x and z are conditionally independent of y. Consequently, these network structures are equivalent. As another example, a *complete network structure* is one that has no missing edges—that is, it encodes no assertions of conditional independence. A domain containing n variables has $n!$ complete network structures: one network structure for each possible ordering of the variables. All complete network structures for a given domain represent the same joint probability distributions—namely, all possible distributions—and are therefore equivalent.

In general, two network structures are equivalent if and only if they have the same structure ignoring arc directions and the same v-structures (Verma and Pearl, 1990). A *v-structure* is an ordered tuple (x, y, z) such that there is an arc from x to y and from z to y, but no arc between x and y. Using this characterization of network-structure equivalence, Chickering (1995) has created an efficient algorithm for characterizing all Bayesian-network structures that are equivalent to a given network structure.

Given that B_S^h is the assertion that the physical probabilities for the joint space of U can be encoded in the network structure B_S, it follows that the hypotheses associated with two equivalent network structures must be identical. Consequently, two equivalent network structures must have the same (prior and posterior) probability. For example, in the two variable domain $\{x, y\}$, the network structures $x \rightarrow y$ and $y \rightarrow x$ are equivalent, and will have the same probability. In general, this property is called *hypothesis equivalence*. In light of this property, we should associate each hypothesis with an equivalence class of structures rather than a single network structure, and our methods for learning network structure should actually be interpreted as methods for learning equivalence classes of network structures (although, for the sake of brevity, we often blur this distinction).[3]

[3]Hypothesis equivalence holds provided we interpret Bayesian-network structures simply as representations of conditional independence. Nonetheless, stronger definitions of Bayesian networks exist where arcs have a causal interpretation (e.g., Pearl and Verma, 1991). Heckerman et al. (1995b) argue that, although it is unreasonable to assume hypothesis equivalence when working with causal Bayesian networks, it is often reasonable to adopt a weaker assumption of *likelihood equivalence*, which

The second important point about this approach is that, in writing Equation 11.5.19, we have assumed that the hypothesis equivalence classes are mutually exclusive. In reality, these hypotheses are not mutually exclusive. For example, in our two-variable domain, both network structures $x \rightarrow y$ and the empty network structure can encode parameters satisfying the equality $\theta_y = \theta_{y|x}$. Therefore, the hypotheses associated with these non-equivalent network structures overlap. Nonetheless, in this approach, we assume that the priors on parameters for any given network structure have bounded distributions, and hence the overlap of hypotheses will be of measure zero.

In principle, the approach we have discussed in this section is essentially all there is to learning network structure. In practice, when the user believes that only a few alternative network structures are possible, he can directly assess the priors for the possible network structures and their parameters, and subsequently use Equations 11.5.18 and 11.5.19 or their generalizations for continuous variables and missing data. For example, Buntine (1994) has designed a software system whereby a user specifies his priors for a set of possible models using Bayesian networks in a manner similar to that shown in Figure 11.4. The system then compiles this specification into a computer program that learns from a database.

Nonetheless, the number of network structures for a domain containing n variables is more than exponential in n. Consequently, when the user cannot exclude almost all of these network structures, there are several issues that must be considered. In particular, computational constraints can prevent us from summing over all the hypotheses in Equation 11.5.19. Can we approximate $p(C_{m+1}|D,\xi)$ accurately by retaining only a small fraction of these hypotheses in the sum? If so, which hypotheses should we include? In addition, how can we efficiently assign prior probabilities to the many network structures and their parameters? In the subsections that follow, we consider each of these issues.

11.5.1 Scoring Metrics

The most important issue is whether we can approximate $p(C_{m+1}|D,\xi)$ well using just a small number of network-structure hypotheses. This

says that the observations in a database can not help to discriminate two equivalent network structures.

question is difficult to answer in theory. Nonetheless, several researchers have shown experimentally that even a single "good" network structure often provides an excellent approximation (Cooper and Herskovits 1992; Aliferis and Cooper 1994; Heckerman et al., 1995b). We give an example in Section 11.6. These results are somewhat surprising, and are largely responsible for the great deal of recent interest in learning Bayesian networks.

Given this observation, another important consideration is how to identify "good" network structures. The approach that has been adopted by many researchers is to use a *scoring metric* in combination with a search algorithm. The scoring metric takes prior knowledge, a database, and a set of network structures, and computes the goodness of fit of those structures to the prior knowledge and data. The search algorithm identifies network structures to be scored. In this section, we discuss scoring metrics. In Section 11.5.4, we discuss search algorithms.

An obvious scoring metric for a single network-structure (equivalence class) is the relative posterior probability of that structure given the database. For example, we can compute $p(D, B_S^h|\xi) = p(B_S^h|\xi)\, p(D|B_S^h, \xi)$ or compute a *Bayes factor*: $p(B_S^h|D, \xi)/p(B_{S0}^h|D, \xi)$ where B_{S0}^h is some reference network structure such as the empty network structure. When we use Equation 11.5.18 to compute this relative posterior probability, the scoring metric is sometimes called the *Bayesian Dirichlet* (BD) metric. A network structure with the highest posterior probability is often called a *maximum a posteriori* (MAP) structure. To score a set of distinct network structures S we can use $\sum_{B_S \in S} p(D, B_S^h|\xi)$. Note that practitioners typically compute logarithms of the probabilities to avoid numerical underflow.

Madigan and Raferty (1994) suggest an alternative scoring metric that uses relative posterior probability in conjunction with heuristics based on the principle of Occam's Razor.

Other scoring metrics approximate the posterior-probability metric. In Section 11.7, we discuss algorithms that can find a local maximum in the probability $p(D|B_S^h, \Theta_{B_S}, \xi)$ as a function of Θ_{B_S} (the physical probabilities associated with network structure B_S). We cannot use such a local maximum as a score for B_S, because it will always favor the most complex network structures, which place no constraints on the parameters Θ_{B_S}. Nonetheless, we can use a local maximum of $p(D|B_S^h, \Theta_{B_S}, \xi)$ as a score for B_S if we also penalize structures based on their complexity.

Akaike (1974) suggests the scoring metric

$$\log p(D|M, \widehat{\Theta}_{B_S}, \xi) + \mathrm{Dim}(M)$$

where M is a model, $\widehat{\Theta}_{B_S}$ denotes the values of Θ_{B_S} that maximize the probability, and $\mathrm{Dim}(M)$ is the number of logically independent parameters in M. This scoring metric is sometimes called the A information criterion (AIC). For a Bayesian network, the penalty is given by

$$\mathrm{Dim}(B_S) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} q_i(r_i - 1)$$

Schwarz (1978) suggests a similar scoring metric with a penalty term given by $(1/2)\mathrm{Dim}(M)\log(m)$, where m is the number of cases in the database. This metric is sometimes called the Bayesian information criterion (BIC).

Another metric that approximates the posterior-probability metric is minimum description length (MDL) (Rissanen 1987). The MDL of a network structure is the sum of the number of bits required to encode the model (which increases with increasing model complexity) and the number of bits required to encode the database given the model (which decreases with increasing model complexity) relative to a particular coding scheme. We note that, in the limit, as the number of cases in the database approach infinity, the BD metric with uniform priors on structures, BIC, and MDL give the same relative scores (Kass and Raferty, 1993). Unfortunately, in practice, this asymptotic equivalence is rarely achieved.

11.5.2 Priors on Structures

The posterior-probability metrics require that we assign a prior probability to every possible network structure. In this section, we present an efficient method for doing so described by Heckerman et al. (1995b).

The approach requires that the user constructs a *prior network structure* for the domain. The method assumes that this structure is a user's "best guess" of the network structure that encodes the physical probabilities.

Given a prior network structure P, we compute the prior probability of B_S as follows. For every variable x_i in U, let δ_i denote the number of nodes in the symmetric difference of $\Pi_i(B_S)$ and $\Pi_i(P)$: $(\Pi_i(B_S) \cup$

$\Pi_i(P)) \setminus (\Pi_i(B_S) \cap \Pi_i(P))$. Then, B_S and the prior network differ by $\delta = \sum_{i=1}^{n} \delta_i$ arcs. We compute the prior probability by penalizing B_S by a constant factor $0 < \kappa \leq 1$ for each such arc. That is, we set

$$p(B_S^h|\xi) = c\ \kappa^\delta \tag{11.5.20}$$

where c is a normalization constant, which we can ignore. Note that this approach assigns equal priors to equivalent network structures only when the prior network structure is empty (see Heckerman et al. [1995b] for a discussion of this point).

This formula is simple, as it requires only the assessment of a prior network structure and a single constant κ. Nonetheless, if the user is willing, he can provide more detailed knowledge by assessing different penalties for different nodes x_i and for different parent configurations of each node (Buntine, 1991). Another variant of this approach is to allow the user to categorically assert that some arcs in the prior network must be present. We can again use Equation 11.5.20, except that we set to zero the priors of network structures that do not conform to these constraints.

11.5.3 Priors on Network Parameters

The posterior-probability metrics also require that we assign priors to network parameters for all possible network structures. Several authors have discussed similar practical approaches for assigning these priors when many structures are possible (Cooper and Herskovits, 1991, 1992; Buntine, 1991; Spiegelhalter et al., 1993; Heckerman et al., 1995b). In this section, we describe the approach of Heckerman et al.

Their approach is based on a result from Geiger and Heckerman (1995). Namely, if all allowed values of the physical probabilities are possible, then parameter independence and hypothesis equivalence[4] imply that the physical probabilities for complete network structures must have Dirichlet distributions as specified in Equation 11.4.13 with the constraint

$$N'_{ijk} = N'\ p(x_i = k, \Pi_i = j|B_{S_C}^h, \xi) \tag{11.5.21}$$

where N' is the user's equivalent sample size for the domain, $B_{S_C}^h$ is the hypothesis corresponding to any complete network structure, and

[4] Actually, Geiger and Heckerman (1995) proved this result using only likelihood equivalence.

$p(x_i = k, \Pi_i = j | B^h_{S_C}, \xi)$ is the user's probability that $x_i = k$ and $\Pi_i = j$ in the first case to be seen in the database.

Under these conditions, the priors on parameters for all complete network structures may be determined by (1) constructing a *prior network* for the first case to be seen (from which the probabilities in Equation 11.5.21 may be computed) and (2) assessing the equivalent sample size (i.e., confidence) in that prior network. In Section 11.6, we give an example of a prior network.

To determine priors for parameters of incomplete network structures, Heckerman et al. (1995b) use the assumption of *parameter modularity,* which says that given two network structures B_{S1} and B_{S2}, if x_i has the same parents in B_{S1} and B_{S2}, then

$$p(\Theta_{ij} | B^h_{S1}, \xi) = p(\Theta_{ij} | B^h_{S2}, \xi)$$

for $j = 1, \ldots, q_i$. They call this property parameter modularity, because it says that the distributions for parameters Θ_{ij} depend only on the structure of the network that is local to variable x_i—namely, Θ_{ij} only depends on x_i and its parents. For example, consider the network structure $x \rightarrow y$ and the empty structure for our two-variable domain with corresponding hypotheses $B^h_{x \rightarrow y}$ and B^h_{xy}. In both structures, x has the same set of parents (the empty set). Consequently, by parameter modularity, $p(\theta_x | B^h_{x \rightarrow y}, \xi) = p(\theta_x | B^h_{xy}, \xi)$

has par-
network
larity to
the BD
uivalent
metric.

the parameters of each node separately. Furthermore, if node x_i ents Π_i in the given network structure, we identify a complete structure where x_i has these parents, and use parameter modul determine the priors for this node. The result is a special case of metric, called the BDe metric, that assigns equal scores to eq network structures. In Section 11.6, we illustrate the use of this

11.5.4 Search Methods

rk struc-
e use of
Given a
tructure

In this section, we examine search methods for identifying netwo tures with high scores. Essentially all such search methods mak a property of the scoring metrics that we call decomposability. network structure for domain U, we say that a measure on that s

tructure

a property of the scoring metrics that we call decomposability. network structure for domain U, we say that a measure on that s

is *decomposable* if it can be written as a product of measures, each of which is a function only of one node and its parents. For example, from Equation 11.5.18, we see that the probability $p(D|B_S^h, \xi)$ given by the BD metric is decomposable. Consequently, if the prior probabilities of network structures are decomposable (as they are in Equation 11.5.20), then so is the BD metric. Thus, we can write

$$p(D, B_S^h|\xi) = \prod_{i=1}^{n} s(x_i|\Pi_i) \tag{11.5.22}$$

where $s(x_i|\Pi_i)$ is only a function of x_i and its parents. Most Bayesian and non-Bayesian metrics for complete databases are decomposable. Given a decomposable metric, we can compare the score for two network structures that differ by the addition or deletion of arcs pointing to x_i, by computing only the term $s(x_i|\Pi_i)$ for both structures.

First, let us consider the special case of finding the network structure with the highest score among all structures in which every node has at most one parent. For each arc $x_j \rightarrow x_i$ (including cases where x_j is null), we associate a weight $w(x_i, x_j) \equiv \log s(x_i|x_j) - \log s(x_i|\emptyset)$. From Equation 11.5.22, we have

$$\log p(D, B_S^h) = \sum_{i=1}^{n} \log s(x_i|\pi_i) = \sum_{i=1}^{n} w(x_i, \pi_i) + \sum_{i=1}^{n} \log s(x_i|\emptyset) \tag{11.5.23}$$

where π_i is the (possibly) null parent of x_i. The last term in Equation 11.5.23 is the same for all network structures. Thus, among the network structures in which each node has at most one parent, ranking network structures by sum of weights $\sum_{i=1}^{n} w(x_i, \pi_i)$ or by score has the same result. Finding the network structure with the highest weight is a special case of a well-known problem of finding *maximum branchings* (Edmonds, 1967). Algorithms for finding maximum branchings can be used regardless of the metric we use, as long as one can associate a weight with every edge. Therefore, this algorithm is appropriate for any decomposable metric. When using metrics that assign equal scores to equivalent network structures, however, we have

$$s(x_i|x_j)\ s(x_j|\emptyset) = s(x_j|x_i)\ s(x_i|\emptyset)$$

Thus, for any two edges $x_i \rightarrow x_j$ and $x_i \leftarrow x_j$, the weights $w(x_i, x_j)$ and $w(x_j, x_i)$ are equal. Consequently, the directionality of the arcs plays no

role for such metrics, and the problem reduces to finding the undirected forest for which $\sum w(x_i, x_j)$ is a maximum. This search can be done using a maximum spanning tree algorithm.

Now, let us consider the case where we find the best network from the set of all networks in which each node has no more than k parents. Unfortunately, the problem for $k > 1$ is NP-hard (Chickering et al. 1995). Therefore, heuristic search algorithms are used.

Most of the commonly discussed search methods for learning Bayesian networks make successive arc changes to the network, and employ the property of decomposability to evaluate the merit of each change. The possible changes that can be made are easy to identify. For any pair of variables, if there is an arc connecting them, then this arc can either be reversed or removed. If there is no arc connecting them, then an arc can be added in either direction. All changes are subject to the constraint that the resulting network contains no directed cycles. We use E to denote the set of eligible changes to a graph, and $\Delta(e)$ to denote the change in log score of the network resulting from the modification $e \in E$. Given a decomposable metric, if an arc to x_i is added or deleted, only $s(x_i|\Pi_i)$ need be evaluated to determine $\Delta(e)$. If an arc between x_i and x_j is reversed, then only $s(x_i|\Pi_i)$ and $s(x_j|\Pi_j)$ need be evaluated.

One simple heuristic search algorithm is *local search*. First, we choose a graph. Then, we evaluate $\Delta(e)$ for all $e \in E$, and make the change e for which $\Delta(e)$ is a maximum, provided it is positive. We terminate search when there is no e with a positive value for $\Delta(e)$. Using decomposable metrics, we can avoid recomputing all terms $\Delta(e)$ after every change. In particular, if neither x_i, x_j, nor their parents are changed, then $\Delta(e)$ remains unchanged for all changes e involving these nodes as long as the resulting network is acyclic. Candidates for the initial graph include the empty graph, a random graph, a graph determined by one of the polynomial algorithms described previously in this section, and the prior network.

A potential problem with any local-search method is getting stuck at a local maximum. In one method for escaping local maxima, called *iterated hill-climbing*, we apply local search until we hit a local maximum. We then randomly perturb the current network structure, and repeat the process for some manageable number of iterations. Other methods for escaping local maxima include best-first search (Korf, 1993), simulated annealing (Metropolis et al., 1953), and Gibbs' sampling (see

Section 11.7.2).

11.6 A Real-World Example

Figure 11.5 illustrates an application of these techniques to the real-world domain of ICU ventilator management, taken from Heckerman et al. (1995b). Figure 11.5a is a hand-constructed Bayesian network for this domain, called the Alarm network (Beinlich et al., 1989) (the probabilities are not shown). Figure 11.5c is a database of 10,000 cases that is sampled from the Alarm network. Figure 11.5b is a hypothetical prior network for the domain. Heckerman et al. (1995b) constructed this network by adding, deleting, and reversing arcs in the Alarm network and by adding noise to the probabilities of the Alarm network.

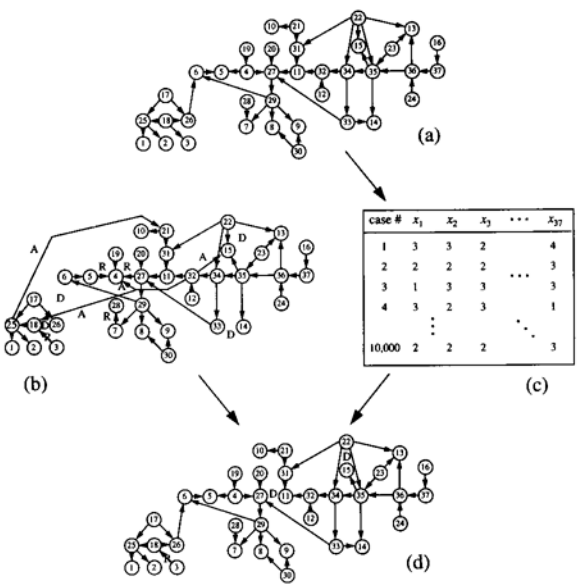

Figure 11.5
(a) The Alarm network structure. (b) A prior network encoding a user's beliefs about the Alarm domain. (c) A 10,000-case database generated from the Alarm network. (d) The network learned from the prior network and a 10,000 case database generated from the Alarm network. Arcs that are added, deleted, or reversed with respect to those in the Alarm network are indicated with A, D, and R, respectively. (Taken from Heckerman et al., 1995b.)

Figure 11.5d shows the most likely network structure found by local search initialized with the prior network structure using the BDe metric, an equivalent sample size $N' = 64$, and priors on network structures determined by Equation 11.5.20 with $\kappa = 1/(N'+1)$. Comparing the three network structures, we see that the learned network structure is much closer to that of the Alarm network than that of the prior network. Furthermore, the joint distribution encoded by the learned network is much closer to that of the Alarm network than that of the prior network. In particular, whereas the cross entropy of the joint distributions of the prior network with respect to that of the Alarm network is 5.6, the cross entropy of the joint distribution of the learned network with respect to that of the Alarm network is 0.03.[5] The learning algorithm has effectively used the database to "correct" the prior knowledge of the user.

11.7 Missing Data

In real databases, observations of one or more variables in one or more cases are typically missing. In this section, we consider extensions to previous methods that can handle missing data. We caution the reader that the methods we discuss assume that whether or not an observation is missing is independent of the actual states of the variables. For example, these methods are not appropriate for a medical database where data about drug response is missing in those patients who became too sick to take the drug.

11.7.1 Fill-In Methods

First, let us consider the simple situation where we observe a single incomplete case C in domain U. Let C' denote the variables not observed in the case. We can compute the posterior distribution of Θ_{ij} as follows:

$$\begin{aligned} p(\Theta_{ij}|C,\xi) &= \sum_U p(U|C,\xi)\; p(\Theta_{ij}|C,C',\xi) \qquad (11.7.24) \\ &= (1 - p(\Pi_i = j|C,\xi))\; \{p(\Theta_{ij}|\xi)\} + \end{aligned}$$

[5] By way of comparison, the cross entropy of an empty network whose probabilities are determined from the marginals of the Alarm network with respect to that of the Alarm network is 13.6.

$$\sum_{k=1}^{r_i} p(x_i = k, \Pi_i = j|C, \xi) \{ p(\Theta_{ij}|x_i = k, \Pi_i = j, \xi)\}$$

where x_i and Π_i refer to these variables in case C. Each term in curly brackets in Equation 11.7.24 is a Dirichlet distribution. Thus, unless both x_i and all the variables in Π_i are observed in case C, the posterior distribution of Θ_{ij} will be a linear combination of Dirichlet distributions. Such distributions are sometimes called *Dirichlet mixtures*; and the probabilities $(1 - p(\Pi_i = j|C, \xi))$ and $p(x_i = k, \Pi_i = j|C, \xi), k = 1, \ldots, r_i$ are called *mixing coefficients*.

If we observe two cases, then the situation becomes more complex, because the computation of the mixing coefficients involves finding the means of Dirichlet mixtures. In general, as shown (e.g.) in Cooper and Herskovits (1992), the computational complexity of the exact computation of $p(D, B_S^h|\xi)$ can be exponential in the number of missing variable entries in the database.

Thus, in practice, we require an approximation. One approach is to approximate each correct posterior distribution Θ_{ij} with a single Dirichlet distribution, and continue to use Equation 11.5.17 along with the formula for the mean of a Dirichlet distribution. Several such approximations have been described in the literature. For example, Titterington (1976) describes a method called *fractional updating*, wherein for each unobserved variable, we pretend that we have observed a fractional number of instances of each state of that variable. In particular, he suggests the approximation:

$$p(\Theta_{ij}|C, \xi) \approx c \prod_{k=1}^{r_i} \theta_{ijk}^{p(x_i=k, \Pi_i=j|C,\xi)} p(\Theta_{ij}|\xi) \tag{11.7.25}$$

One drawback of this method is that it falsely increases the equivalent sample sizes of the Dirichlet distributions associated with each Θ_{ij}, because it replaces each missing datum with a fractional sample. Cowell et al. (1995) suggest an approach that does not have this problem. Namely, they approximate Θ_{ij} by a single Dirichlet whose means and average variance $\sum_{k=1}^{r_i} Var(\theta_{ijk})/r_i$ are the same as those for the correct Dirichlet mixture.

These approximations process the data in the database sequentially, and make use of the assumption of parameter independence and properties of the Dirichlet distribution. Other methods—including Gibbs

sampling, the EM algorithm, and gradient descent—process all the data at once, and can handle continuous domain variables and dependent parameters.

11.7.2 Gibbs Sampling

Gibbs sampling, described—for example—by Geman and Geman (1984), is a special case of Markov chain Monte-Carlo methods for approximate inference (Hastings, 1970). Given variables $X = \{x_1, \ldots, x_n\}$ with some joint distribution $p(X|\xi)$, we can use a Gibbs sampler to approximate the expectation of any function $f(X)$ with respect to $p(X|\xi)$ $(E(f(X)|\xi))$ as follows. First, we choose an initial state of each of the variables in X somehow. Next, we pick some variable x_i, unassign its current state, and compute its probability distribution given the assignments to the other $n-1$ variables. Then, we sample a state for x_i based on this probability distribution, and compute $f(X)$. Finally, we iterate the previous two steps, keeping track of the average value of $f(X)$. In the limit as the number of samples approach infinity, this average is equal to $E(f(X)|\xi)$ provided two conditions are met. First, the Gibbs sampler must be *irreducible*: The probability distribution $p(X)$ must be such that we can eventually sample any possible state of X given any possible initial state of X. For example, if $p(X)$ contains no zero probabilities, then the Gibbs sampler will be irreducible. Second, each x_i must be chosen infinitely often. In practice, an algorithm for deterministically rotating through the variables is typically used. For a good introductory discussion on Gibbs sampling—including methods for initialization and a discussion of convergence—see York (1992).

Now, suppose we have a database $D = \{C_1, \ldots, C_m\}$ with missing data and a new case C_{m+1}, and we want to approximate $p(C_{m+1}|D, B_S^h, \xi)$ for a given network structure B_S. One reasonable variant of Gibbs sampling for performing this estimation goes as follows. First, we initialize the parameters Θ_{B_S} somehow. Second, for each case C_l in D containing missing data and for every variable x_i that is unobserved in C_l, we assign the state of x_i using the assigned values of Θ_{B_S}, creating a complete database D'. This step can be done using any standard Bayesian network inference algorithm. Third, we reassign the parameters Θ_{B_S} according to the posterior distribution $p(\Theta_{B_S}|D', B_S^h, \xi)$. Finally, we iterate the previous two steps, and use the average of $p(C_{m+1}|\Theta_{B_S}, B_S^h, \xi)$ computed after each iteration to approximate $p(C_{m+1}|D, B_S^h, \xi)$. Note

$p(D|B_S^h, \Theta_{B_S}, \xi)$ is in the exponential family, as is the case for discrete variables. When the likelihood does not have this form, we can use general optimization methods to maximize $p(D|B_S^h, \Theta_{B_S}, \xi)$ (Press et al. 1992). These approaches—for example, gradient descent, conjugate gradient, and quasi-Newton methods—require derivatives of the function to be maximized. Buntine (1994) discusses how we can use the structure of B_s to simplify the computation of these derivatives.

11.8 Learning New Variables

In a database with missing data, a particular variable may be observed in some cases, or it may never be observed. In the latter situation, we say that the variable is *hidden.*

Any of the methods described in the previous section can be used to learn Bayesian networks containing hidden variables. The network structure may be fixed and only the physical probabilities uncertain, or both the network structure and parameters may be uncertain. One example of learning the probabilities of a fixed structure with hidden variables is the AutoClass algorithm of Cheeseman and Stutz (in this volume), which performs *unsupervised classification.* The model underlying the algorithm is a Bayesian network with a single hidden variable whose states correspond to unknown classes. The number of states of the hidden variable is uncertain and has a prior distribution. Also, this hidden variable renders sets of observable variables conditionally independent. The algorithm searches over variations of this model (including the number of states of the hidden variable), using a version of the EM algorithm in conjunction with an approximate Bayesian scoring metric to select the model variation with the highest posterior probability.

In addition, we can use methods for learning with missing data to identify (under uncertainty) the existence of new variables. Namely, we hypothesize a mutually exclusive and exhaustive set of Bayesian-network structures, some containing hidden variables and some not. We assign priors to each structure and its parameters, and then update these priors with data using one of the described algorithms for handling missing data.

11.9 Pointers to the Literature

Like all tutorials, this tutorial is incomplete. For those readers interested in learning more about graphical models and methods for learning them, we offer the following additional references. A more detailed guide to the literature can be found in Buntine (1995).

Charniak (1991) provides an easy-to-read introduction to the Bayesian-network representation. Spiegelhalter et al. (1993) and Heckerman et al. (1995) give simple discussions of methods for learning Bayesian networks for domains containing only discrete variables. Buntine (1994) and Heckerman and Geiger (1994) provide more detailed discussions. Experimental comparisons of different learning approaches can be found in Cooper and Herskovits (1992), Aliferis and Cooper (1994), Lauritzen et al. (1994), Cowell et al. (1995), and Heckerman et al. (1995b).

In addition to directed models, researchers have also explored graphs containing undirected edges as a knowledge representation. These representations are discussed (e.g.) in Lauritzen (1982), Verma and Pearl (1990), and Frydenberg (1990). Bayesian methods for learning such models from data are described by Dawid and Lauritzen (1993) and Buntine (1994).

Finally, several software systems for learning graphical models have been implemented. Thomas, Spiegelhalter, and Gilks (1992) have created a system that takes a learning problem specified as a Bayesian network and compiles this problem into a Gibbs-sampler computer program. Badsberg (1992) and Højsgaard et al. (1994) have built systems that can learn directed, undirected, and mixed graphical models using a variety of scoring metrics.

Acknowledgments

I thank David Chickering, Eric Horvitz, Koos Rommelse, and Padhraic Smyth for their comments on earlier versions of this manuscript.

References

Akaike, H. 1974. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control,* 19: 716–723.

Aliferis, C.; and Cooper, G. 1994. An Evaluation of an Algorithm for Inductive Learning of Bayesian Belief Networks Using Simulated Data Sets. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* 8–14. San Francisco: Morgan Kaufmann.

Badsberg, J. 1992. Model Search in Contingency Tables by Coco. In *Computational Statistics,* 251–256, ed. Y. Dodge and J. Wittaker. Heidelberg: Physica Verlag.

Beinlich, I.; Suermondt, H.; Chavez, R.; and Cooper, G. 1989. 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,* 247–256. Berlin: Springer Verlag.

Buntine, W. 1991. Theory Refinement on Bayesian Networks. In *Proceedings of Seventh Conference on Uncertainty in Artificial Intelligence,* 52–60. San Francisco: Morgan Kaufmann.

Buntine, W. 1994. Operations for Learning with Graphical Models. *Journal of Artificial Intelligence Research,* 2: 159–225.

Buntine, W. 1995. A Guide to the Literature on Learning Graphical Models. Technical Report IC-95-05, NASA Ames Research Center.

Charniak, E. 1991. Bayesian Networks Without Tears. *AI Magazine,* 12: 50–63.

Chickering, D. 1995. A Transformational Characterization of Equivalent Bayesian Network Structures. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Montreal, 87–98. San Francisco: Morgan Kaufmann.

Chickering, D.; Geiger, D.; and Heckerman, D. 1995. Learning Bayesian Networks: Search Methods and Experimental Results. In *Proceedings of Fifth Conference on Artificial Intelligence and Statistics,* 112–128. Society for Artificial Intelligence in Statistics.

Cooper, G.; and Herskovits, E. 1992. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning,* 9: 309–347.

Cooper, G.; and Herskovits, E. 1991. A Bayesian Method for the Induction of Probabilistic Networks from Data. Technical Report SMI-91-1, Section on Medical Informatics, Stanford University.

Cowell, R.; Dawid, A.; and Sebastiani, P. 1995. A Comparison of Sequential Learning Methods for Incomplete Data. Technical Report 135, Department of Statistical Science, University College London.

DeGroot, M. 1970. *Optimal Statistical Decisions.* New York: McGraw-Hill.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society,* B 39:1–38.

Edmonds, J. 1967. Optimum Branching. *J. Res. NBS,* 71B: 233–240.

Frydenberg, M. 1990. The Chain Graph Markov Property. *Scandinavian Journal of Statistics,* 17: 333–353.

Geiger, D.; and Heckerman, D. 1995. A Characterization of the Dirichlet Distribution Applicable to Learning Bayesian Networks. Technical Report MSR-TR-94-16, Microsoft, Redmond, Wash.

Geman, S.; and Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 6: 721–742.

Good, I. 1959. Kinds of Probability. *Science,* 129: 443–447.

Good, I. 1965. *The Estimation of Probabilities.* Cambridge, Mass.: The MIT Press.

Hastings, W. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika,* 57: 97–109.

Heckerman, D. and Geiger, D. 1994. Learning Bayesian Networks. Technical Report MSR-TR-95-02, Microsoft, Redmond, Wash.

Heckerman, D.; Mamdani, A.; and Wellman, M. 1995a. Special Issue on Real-World Applications of Bayesian Networks. *Communications of the ACM,* March.

Heckerman, D.; Geiger, D.; and Chickering, D. 1995b. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning.*

Højsgaard, S.; Skjøth, F.; and Thiesson, B. 1994. User's Guide to BIFROST. Technical Report, Department of Mathematics and Computer Science, Aalborg, Denmark.

Howard, R. 1970. Decision Analysis: Perspectives on Inference, Decision, and Experimentation. *Proceedings of the IEEE,* 58: 632–643.

Howard, R.; and Matheson, J. 1981. Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis,* volume II, 721–762. ed. R. Howard and J. Matheson. Menlo Park, Calif.: Strategic Decisions Group.

Kass, R.; and Rafferty, A. 1993. Bayes Factors and Model Uncertainty. Technical Report 571, Department of Statistics, Carnegie Mellon University.

Korf, R. 1993. Linear-Space Best-First Search. *Artificial Intelligence,* 62: 41–78.

Lauritzen, S. 1982. *Lectures on Contingency Tables.* University of Aalborg Press, Aalborg, Denmark.

Lauritzen, S.; Thiesson, B.; and Spiegelhalter, D. 1994. Diagnostic Systems Created by Model Selection Methods: A Case Study. In *AI and Statistics IV,* 143–152, ed. P. Cheeseman and R. Oldford. New York: Springer-Verlag.

Madigan, D.; and Rafferty, A. 1994. Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. *Journal of the American Statistical Association,* 89: 1535–1546.

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; and Teller, E. 1953. *Journal of Chemical Physics,* 21: 1087–1092.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Francisco: Morgan Kaufmann.

Pearl, J.; and Verma, T. 1991. A Theory of Inferred Causation. In *Knowledge Representation and Reasoning: Proceedings of the Second International Conference,* 441–452, ed. J. Allen, R. Fikes, and E. Sandewal. San Francisco: Morgan Kaufmann.

Press, W.; Teukolsky, S.; Vetterling, W.; and Flannery, B. 1992. *Numerical Recipes in C.* New York: Cambridge University Press.

Rissanen, J. 1987. Stochastic Complexity (with Discussion). *Journal of the Royal Statistical Society, Series B,* 49: 223–239, 253–265.

Schwarz, G. 1978. Estimating the Dimension of a Model. *Annals of Statistics,* 6: 461–464.

Spiegelhalter, D.; Dawid, A.; Lauritzen, S.; and Cowell, R. 1993. Bayesian Analysis in Expert Systems. *Statistical Science,* 8: 219–282.

Spiegelhalter, D. and Lauritzen, S. 1990. Sequential Updating of Conditional Probabilities on Directed Graphical Structures. *Networks,* 20: 579–605.

Spirtes, P.; Glymour, C.; and Scheines, R. 1993. *Causation, Prediction, and Search.* New York: Springer-Verlag.

Thomas, A.; Spiegelhalter, D.; and Gilks, W. 1992. Bugs: A Program to Perform Bayesian Inference Using Gibbs Sampling. In *Bayesian Statistics,* 837–842, ed. J. Bernardo, J. Berger, A. Dawid, and A. Smith. New York: Oxford University Press.

Titterington, D. 1976. Updating a Diagnostic System Using Unconfirmed Cases. *Applied Statistics,* 25: 238–247.