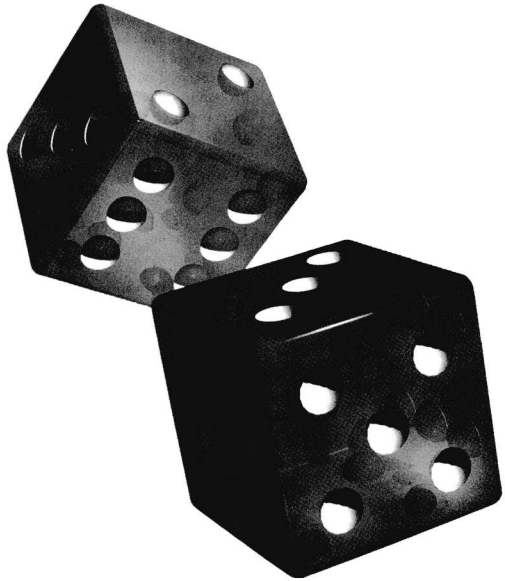
David Heckerman and Michael P. Wellman



Bayesian *Networks*

This brief tutorial on Bayesian networks serves to introduce readers to some of the concepts, terminology, and notation employed by articles in this special section. In a Bayesian network, a variable takes on values from a collection of mutually exclusive and collective exhaustive states. A variable may be discrete, having a finite or countable number of states, or it may be continuous. Often the choice of states itself presents an interesting modeling question. For example, in a system for troubleshooting a problem with printing, we may choose to model the variable "print output" with two states—"present" and "absent"—or we may want to model the variable with finer distinctions such as "absent," "blurred ," "cut off," and "ok."

In describing a Bayesian network, we use lower-case letters 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 an instance of X. The joint space of a set of variables U is the set of all instances of U. The joint probability distribution over U is the probability distri-

bution over the joint space of U. We use $p(X|Y)$ to denote the set of joint probability distributions over X, one for each conditional on every instance in the joint space of Y.

A problem domain is just a set of variables. A Bayesian network for the domain $\{x_1, \ldots, x_n\}$ represents a joint probability distribution over those variables. The representation consists of a set of local conditional probability distributions, combined with a set of assertions of conditional independence that allow us to construct the global joint distribution from the local distributions. The decomposition is based on the chain rule of probability, which dictates that

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

For each variable x_i, let $\Pi_i \subseteq \{x_1, \ldots, x_{i-1}\}$ be a set of variables that renders x_i and $\{x_1, \ldots, x_{i-1}\}$ conditionally independent. That is,

$$p(x_i|x_1, \ldots, x_{i-1}) = p(x_i|\Pi_i) \quad (2)$$

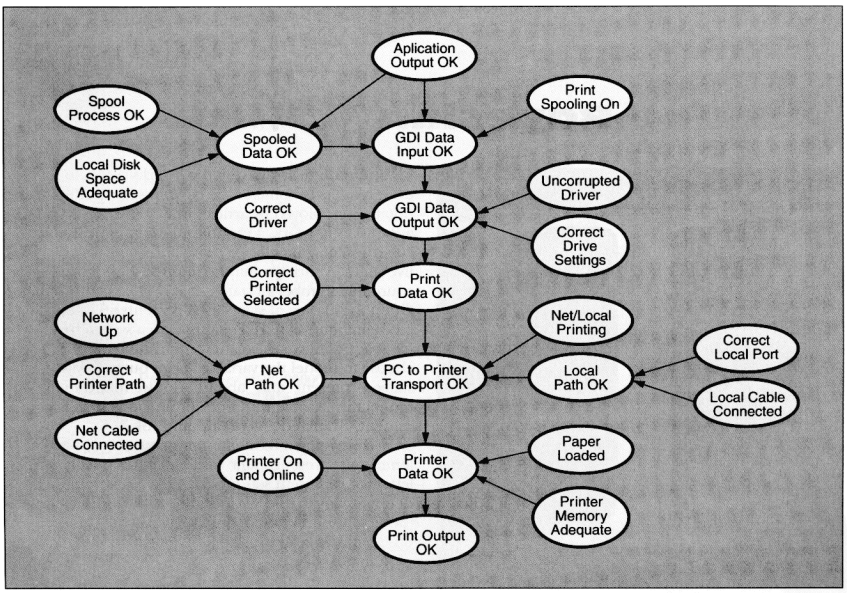
The idea is that the distribution of x_i can often be described conditional on a parent set Π_i that is substantially smaller than the set $\{x_1, \ldots, x_{i-1}\}$. Given these sets, a Bayesian network can be described as a directed acyclic graph such that each variable $x_1, \ldots, x_n$ corresponds to a node in that graph and the parents of the node corresponding to x_i are the nodes corresponding to the variables in Π_i. Note that since the parents in the graph coincide with the conditioning sets Π_i, the Bayesian network structure directly encodes the assertions of conditional independence in (2).

Associated with each node x_i are the conditional probability distributions $p(x_i|\Pi_i)$—one distribution for each instance of Π_i. Combining (1) and (2), we see that any Bayesian network for $\{x_1, \ldots, x_n\}$ uniquely determines a joint probability distribution for those variables. That is,

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i|\Pi_i) \quad (3)$$

Although the formal definition of a Bayesian network is based on conditional independence, in practice a Bayesian network typically is constructed using notions of cause and effect. Loosely speaking, 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. Figure 1 illustrates the structure of a Bayesian network for troubleshooting printing problems using the Windows™ operating system, which was constructed

Figure 1. A Bayesian network structure for troubleshooting a printing problem. Arcs are drawn from cause to effect.



using cause-and-effect considerations. For example, "Net Path OK" is caused by "Network Up," "Correct Printer Path," and "Net Cable Connected."

When a node has many parents, specifying even its local distribution can be quite onerous in the general case. According to the definition of Bayesian networks, we must assess the probability distribution of the node conditional on every instance of its parents. Thus, for example, if a node has n binary-valued parents, we must specify 2^n probability distributions for the node. In such cases, we can often reduce the burden of this assessment by introducing more structure into the model.

For example, suppose we have n binary causes $c_1, \ldots, c_n$ bearing on a single binary effect e, as shown in Figure 2(a). In many cases, we can model the n-way interaction by associating with each cause an inhibitory mechanism that prevents the cause from producing the effect. The effect will be absent only if all the inhibitory mechanisms associated with present causes are active. For example, consider the node "Spooled Data OK" and its parents in our print troubleshooter model. Although the spool process may be bad for a given font due to a programming bug, this cause of bad spooled output will be inhibited if the document being printed does not use that font. Also, local disk space may be inadequate, but this cause of bad spooled output will be inhibited if the print job is small. Figure 2(b) represents this scenario, under the assumption that the inhibitory mechanisms $m_1, \ldots, m_n$ are independent. This model, called the noisy-OR model [11], reduces the assessment burden from exponential to linear in n. Other models that are useful in practice include "noisy" versions of AND, MAX, MIN, and ADDER [4].

Because a Bayesian network for any domain determines a joint probability distribution for that domain, we can—in principle—use a Bayesian network to compute any probability of interest. For example, suppose we have the simple Bayesian network with structure $w \rightarrow x \rightarrow y \rightarrow z$, and we want to know $p(w|z)$. From the rules of probability we have

$$p(w|z) = \frac{p(w, z)}{p(z)} = \frac{\Sigma_{x,y}\, p(w, x, y, z)}{\Sigma_{w,x,y}\, p(w, x, y, z)} \qquad (4)$$

where $p(w, x, y, z)$ is the joint distribution determined from the Bayesian network. In practice, this approach is not feasible, because it entails summing over an exponential number of terms. Fortunately, we can exploit the conditional independencies encoded in a Bayesian network to make this computation more efficient. In this case, given the network structure, (4) becomes

$$p(w|z) = \frac{\Sigma_{xy}\, p(w, x, y, z)}{\Sigma_{w,x,y}\, p(w, x, y, z)} = \frac{p(w)\, \Sigma_x\, p(x|w)\, \Sigma_y\, p(y|x) p(z|y)}{\Sigma_w\, p(w)\, \Sigma_x\, p(w) p(x|w)\, \Sigma_y\, p(y|x) p(z|y)} \qquad (5)$$

That is, using conditional independence, we can often reduce the dimensionality of the problem by rewriting the sums over multiple variables as the product of sums over a single variable (or at least smaller numbers of variables).

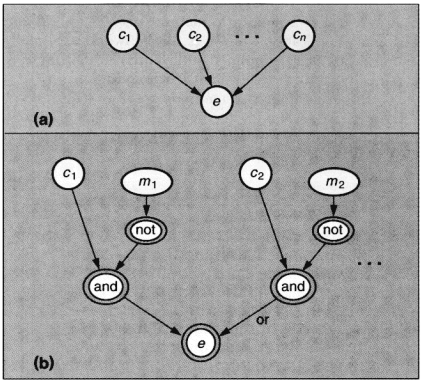


Figure 2. (a) A Bayesian network structure for multiple causes and a single effect. (b) A noisy-OR model for the multiple-cause interaction in (a). Here, independent mechanisms may inhibit the expression of each cause. Nodes with double borders, called deterministic nodes, are deterministic functions of their parents. Assuming all variables are binary, the model in (a) requires $O(2^n)$ probabilities, whereas the model in (b) requires $O(n)$ probabilities.

The general problem of computing probabilities of interest from a (possibly implicit) joint probability distribution is called probabilistic inference. All exact algorithms for probabilistic inference in Bayesian networks exploit conditional independence roughly as we have described, although with different twists. For example, Howard and Matheson [6], Olmsted [9], and Shachter [13] have developed an algorithm that reverses arcs in the network structure until the answer to the given probabilistic query can be read directly from the graph. In this algorithm, each arc reversal corresponds to an application of Bayes' theorem. Pearl [10] has developed a message-passing scheme that updates the probability distributions for each node in a Bayesian network in response to observations of one or more variables. Lauritzen and Spiegelhalter [8] have created an algorithm that first builds an undirected graph from the Bayesian network. The algorithm then exploits several mathematical properties of undirected graphs to perform probabilistic inference. Most recently, D'Ambrosio [3] has developed an inference algorithm that simplifies sums and products symbolically, as in the transformation from (4) to (5).

Although we can exploit assertions of conditional independence in a Bayesian network for probabilistic inference, exact inference in an arbitrary Bayesian net-

work is NP-hard [1]. Even approximate inference (for example, using Monte Carlo methods) is NP-hard [2]. For many applications, however, the networks are small enough (or can be simplified sufficiently) so that these complexity results are not fatal. For applications in which the usual inference methods are impractical, researchers are developing techniques custom-tailored to particular network topologies [5, 15], or particular inference queries [7, 12, 14]. ■

References

1. Cooper, G. Computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artif. Intell. 42* (1990), 393–405.
2. Dagum, P., and Luby, M. Approximately probabilistic reasoning in Bayesian belief networks is NP-hard. *Artif. Intell.* (1993), pp. 141–153.
3. D'Ambrosio, B. Local expression languages for probabilistic dependence. In *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence* (Los Angeles, Calif.). Morgan Kaufmann, San Mateo, Calif., 1991, pp. 95–102.
4. Heckerman, D. Causal independence for knowledge acquisition and inference. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence* (Washington, D.C.), Morgan Kaufmann, San Mateo, Calif., 1993, pp. 122–127.
5. Heckerman, D.E. A tractable algorithm for diagnosing multiple diseases. In *Proceedings of the 5th Workshop on Uncertainty in Artificial Intelligence* (Windsor, Ontario). Association for Uncertainty in Artificial Intelligence, Mountain View, Calif., 1989, pp. 174–181. Also in Henrion, M., Schachter, R., Kanal, L., and Lemmer, J., eds. *Uncertainty in Artificial Intelligence 5*. North-Holland, New York, 1990, pp. 163–171.
6. Howard, R.A., and Matheson, J.E. Influence diagrams. In R.A. Howard and J.E. Matheson, eds., *Readings on the Principles and Applications of Decision Analysis*. Vol. 2. Strategic Decisions Group, Menlo Park, Calif., 1981, pp. 721–762.
7. Jensen, F., and Anderson, S.K. Approximations in Bayesian belief universes for knowledge based systems. Tech. Rep., Institute of Electronic Systems, Aalborg Univ., Aalborg, Denmark, 1990.
8. Lauritzen, S.L., and Spiegelhalter, D.J. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Stat. Soc. B 50* (1988), 157–224.
9. Olmsted, S. On representing and solving decision problems. Ph.D. dissertation, Dept. of Engineering-Economic Systems, Stanford Univ., 1983.
10. Pearl, J. Fusion, propagation, and structuring in belief networks. *Artif. Intell. 29* (1986), 241–288.
11. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, Calif., 1988.
12. Ramamurthi, K., and Agogino, A.M. Real time expert system for fault tolerant supervisory control. In V.A. Tipnis and E.M. Patton, eds., *Computers in Engineering*. American Society of Mechanical Engineers, Corte Madera, Calif., 1988, pp. 333–339.
13. Shachter, R.D. Probabilistic inference and influence diagrams. *Oper. Res. 36* (1988), 589–604.
14. Shachter, R.D., Andersen, S.K., and Poh, K.L. Directed reduction algorithms and decomposable graphs. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence* (Boston, Mass.). Association for Uncertainty in Artificial Intelligence, Mountain View, Calif., 1990, pp. 237–244.
15. Suermondt, H.J., and Cooper, G.F. A combination of exact algorithms for inference on Bayesian belief networks. *Int. J. Approximate Reasoning 5* (1991), 521–542.