

This Is a Publication of
The American Association for Artificial Intelligence

This electronic document has been retrieved from the
American Association for Artificial Intelligence
445 Burgess Drive
Menlo Park, California 94025
(415) 328-3123
(415) 321-4457
info@aaai.org
http://www.aaai.org

*(For membership information,
consult our web page)*

The material herein is copyrighted material. It may not be
reproduced in any form by any electronic or
mechanical means (including photocopying, recording,
or information storage and retrieval) without permission
in writing from AAAI.

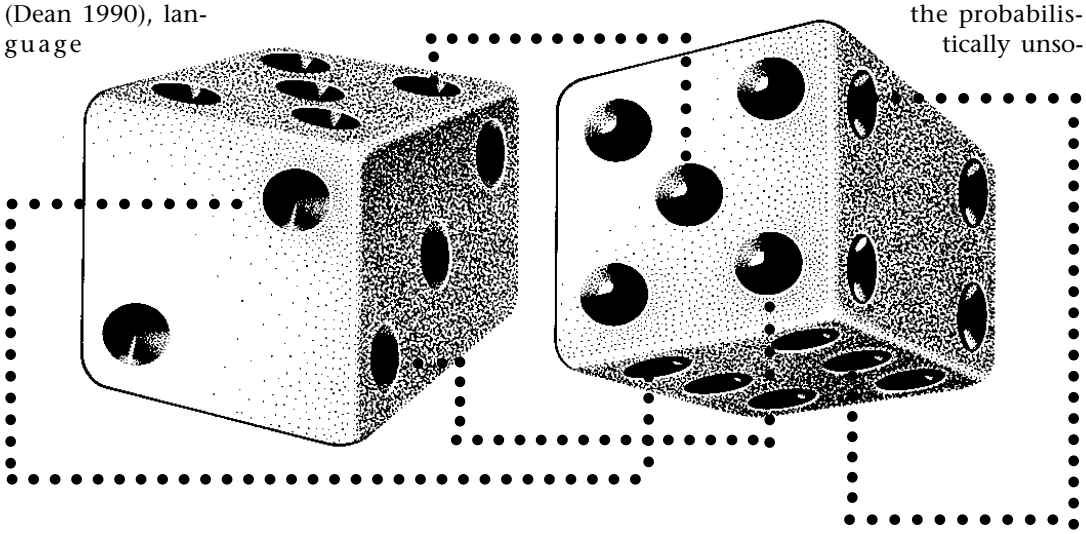
…making Bayesian networks more accessible to the probabilistically unsophisticated.

Bayesian Networks without Tears

Eugene Charniak

I give an introduction to Bayesian networks for AI researchers with a limited grounding in probability theory. Over the last few years, this method of reasoning using probabilities has become popular within the AI probability and uncertainty community. Indeed, it is probably fair to say that Bayesian networks are to a large segment of the AI-uncertainty community what resolution theorem proving is to the AI-logic community. Nevertheless, despite what seems to be their obvious importance, the ideas and techniques have not spread much beyond the research community responsible for them. This is probably because the ideas and techniques are not that easy to understand. I hope to rectify this situation by making Bayesian networks more accessible to the probabilistically unsophisticated.

Over the last few years, a method of reasoning using probabilities, variously called belief networks, Bayesian networks, knowledge maps, probabilistic causal networks, and so on, has become popular within the AI probability and uncertainty community. This method is best summarized in Judea Pearl's (1988) book, but the ideas are a product of many hands. I adopted Pearl's name, Bayesian networks, on the grounds that the name is completely neutral about the status of the networks (do they really represent beliefs, causality, or what?). Bayesian networks have been applied to problems in medical diagnosis (Heckerman 1990; Spiegelhalter, Franklin, and Bull 1989), map learning (Dean 1990), language understanding (Charniak and Goldman 1989a, 1989b; Goldman 1990), vision (Levitt, Mullin, and Binford 1989), heuristic search (Hansson and Mayer 1989), and so on. It is probably fair to say that Bayesian networks are to a large segment of the AI-uncertainty community what resolution theorem proving is to the AI-logic community. Nevertheless, despite what seems to be their obvious importance, the ideas and techniques have not spread much beyond the research community responsible for them. This is probably because the ideas and techniques are not that easy to understand. I hope to rectify this situation by making Bayesian networks more accessible to the probabilistically unso-



0738-4602/91/$4.00 ©1991 AAAI

phisticated. That is, this article tries to make the basic ideas and intuitions accessible to someone with a limited grounding in probability theory (equivalent to what is presented in Charniak and McDermott [1985]).

An Example Bayesian Network

The best way to understand Bayesian networks is to imagine trying to model a situation in which causality plays a role but where our understanding of what is actually going on is incomplete, so we need to describe things probabilistically. Suppose when I go home at night, I want to know if my family is home before I try the doors. (Perhaps the most convenient door to enter is double locked when nobody is home.) Now, often when my wife leaves the house, she turns on an outdoor light. However, she sometimes turns on this light if she is expecting a guest. Also, we have a dog. When nobody is home, the dog is put in the back yard. The same is true if the dog has bowel troubles. Finally, if the dog is in the backyard, I will probably hear her barking (or what I think is her barking), but sometimes I can be confused by other dogs barking. This example, partially inspired by Pearl's (1988) earthquake example, is illustrated in figure 1. There we find a graph not unlike many we see in AI. We might want to use such diagrams to predict what will happen (if my family goes out, the dog goes out) or to infer causes from observed effects (if the light is on and the dog is out, then my family is probably out).

The important thing to note about this example is that the causal connections are not absolute. Often, my family will have left without putting out the dog or turning on a light. Sometimes we can use these diagrams anyway, but in such cases, it is hard to know what to infer when not all the evidence points the same way. Should I assume the family is out if the light is on, but I do not hear the dog? What if I hear the dog, but the light is out? Naturally, if we knew the relevant probabilities, such as $P(\text{family-out} \mid \text{light-on}, \neg\ \text{hear-bark})$, then we would be all set. However, typically, such numbers are not available for all possible combinations of circumstances. Bayesian networks allow us to calculate them from a small set of probabilities, relating only neighboring nodes.

Bayesian networks are *directed acyclic graphs* (DAGs) (like figure 1), where the nodes are random variables, and certain independence assumptions hold, the nature of which I discuss later. (I assume without loss of generality that DAG is connected.) Often, as in figure 1, the random variables can be thought of as states of affairs, and the variables have two possible values, true and false. However, this need not be the case. We could, say, have a node denoting the intensity of an earthquake with values no-quake, trembler, rattler, major, and catastrophe. Indeed, the variable values do not even need to be discrete. For example, the value of the variable earthquake might be a Richter scale number. (However, the algorithms I discuss only work for discrete values, so I stick to this case.)

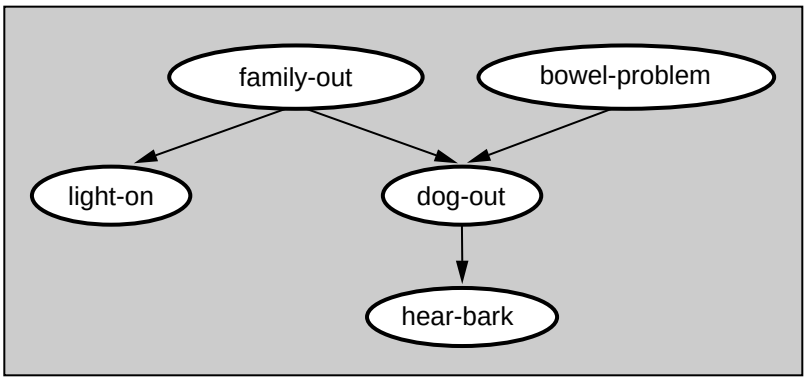


Figure 1. A Causal Graph.

The nodes denote states of affairs, and the arcs can be interpreted as causal connections.

In what follows, I use a sans serif font for the names of random variables, as in earthquake. I use the name of the variable in italics to denote the proposition that the variable takes on some particular value (but where we are not concerned with which one), for example, *earthquake*. For the special case of Boolean variables (with values true and false), I use the variable name in a sans serif font to denote the proposition that the variable has the value true (for example, family-out). I also show the arrows pointing downward so that "above" and "below" can be understood to indicate arrow direction.

The arcs in a Bayesian network specify the independence assumptions that must hold between the random variables. These independence assumptions determine what probability information is required to specify the probability distribution among the random variables in the network. The reader should note that in informally talking about DAG, I said that the arcs denote causality, whereas in the Bayesian network, I am saying that they specify things about the probabilities. The next section resolves this conflict.

To specify the probability distribution of a Bayesian network, one must give the prior probabilities of all root nodes (nodes with no predecessors) and the conditional probabilities

…the complete specification of a probability distribution requires absurdly many numbers.

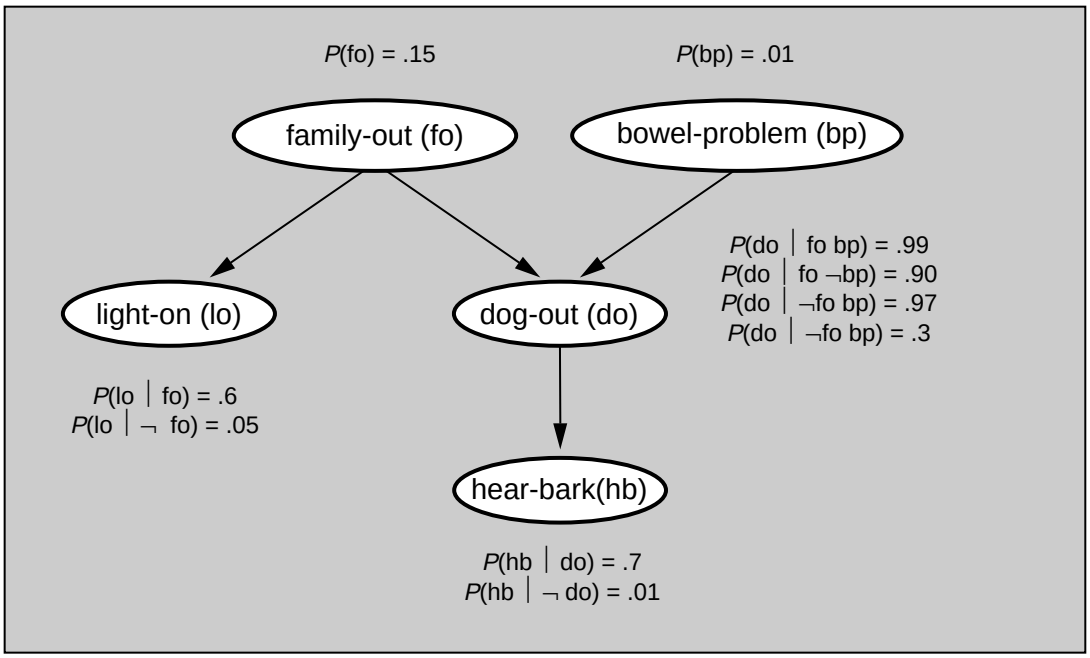


Figure 2. A Bayesian Network for the family-out *Problem.*

I added the prior probabilities for root nodes and the posterior probabilities for nonroots given all possible values of their parents.

of all nonroot nodes given all possible combinations of their direct predecessors. Thus, figure 2 shows a fully specified Bayesian network corresponding to figure 1. For example, it states that if family members leave our house, they will turn on the outside light 60 percent of the time, but the light will be turned on even when they do not leave 5 percent of the time (say, because someone is expected).

Bayesian networks allow one to calculate the conditional probabilities of the nodes in the network given that the values of some of the nodes have been observed. To take the earlier example, if I observe that the light is on (light-on = true) but do not hear my dog (hear-bark = false), I can calculate the conditional probability of family-out given these pieces of evidence. (For this case, it is .5.) I talk of this calculation as *evaluating* the Bayesian network (given the evidence). In more realistic cases, the networks would consist of hundreds or thousands of nodes, and they might be evaluated many times as new information comes in. As evidence comes in, it is tempting to think of the probabilities of the nodes changing, but, of course, what is changing is the conditional probability of the nodes given the changing evidence. Sometimes people talk about the belief of the node changing. This way of talking is probably harmless provided that one keeps in mind that here, *belief* is simply the conditional probability given the evidence.

In the remainder of this article, I first describe the independence assumptions implicit in Bayesian networks and show how they relate to the causal interpretation of arcs (Independence Assumptions). I then show that given these independence assumptions, the numbers I specified are, in fact, all that are needed (Consistent Probabilities). Evaluating Networks describes how Bayesian networks are evaluated, and the next section describes some of their applications.

Independence Assumptions

One objection to the use of probability theory is that the complete specification of a probability distribution requires absurdly many numbers. For example, if there are n binary random variables, the complete distribution is specified by 2^n-1 joint probabilities. (If you do not know where this 2^n-1 comes from, wait until the next section, where I define joint probabilities.) Thus, the complete distribution for figure 2 would require 31 values, yet we only specified 10. This savings might not seem great, but if we doubled the

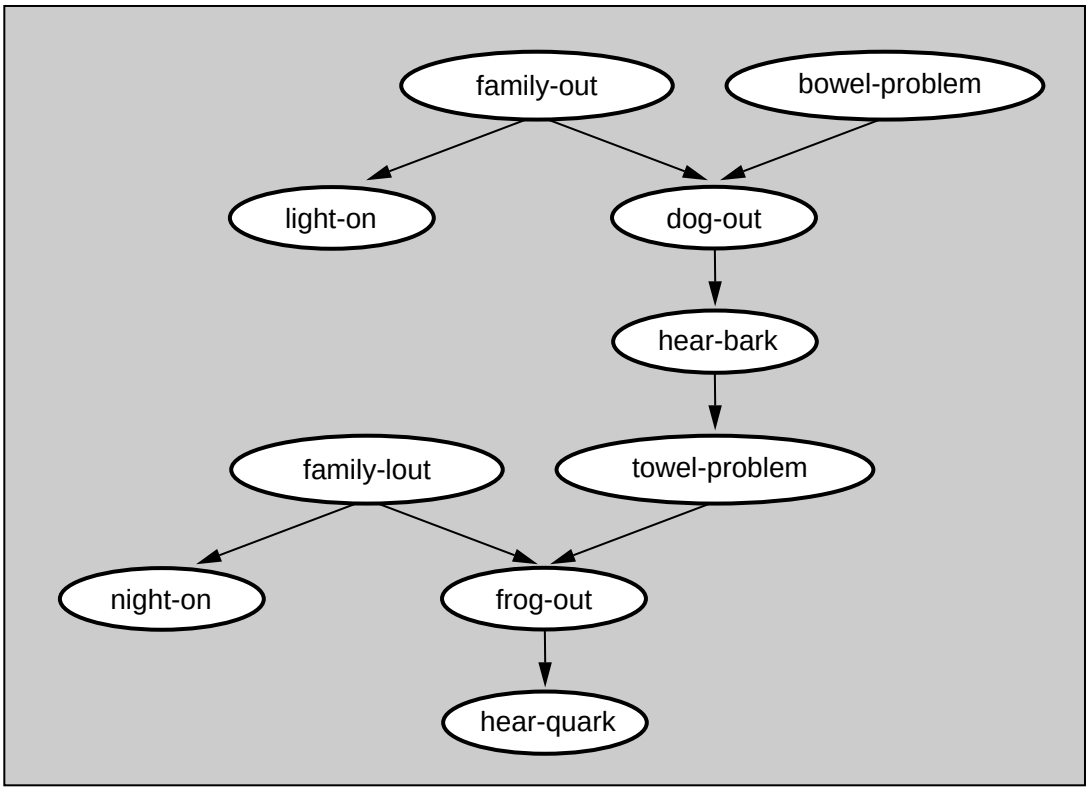


Figure 3. A Network with 10 Nodes.

This illustration is two copies of the graph from figure 1 attached to each other. Nonsense names were given to the nodes in the second copy.

size of the network by grafting on a copy, as shown in figure 3, 2^n-1 would be 1023, but we would only need to give 21. Where does this savings come from?

The answer is that Bayesian networks have built-in independence assumptions. To take a simple case, consider the random variables family-out and hear-bark. Are these variables independent? Intuitively not, because if my family leaves home, then the dog is more likely to be out, and thus, I am more likely to hear it. However, what if I happen to know that the dog is definitely in (or out of) the house? Is hear-bark independent of family-out then? That is, is *P*(hear-bark | family-out, dog-out) = *P*(hear-bark | dog-out)? The answer now is yes. After all, my hearing her bark was dependent on her being in or out. Once I know whether she is in or out, then where the family is is of no concern.

We are beginning to tie the interpretation of the arcs as direct causality to their probabilistic interpretation. The causal interpretation of the arcs says that the family being out has a direct causal connection to the dog being out, which, in turn, is directly connected to my hearing her. In the probabilistic interpretation, we adopt the independence assumptions that the causal interpretation suggests. Note that if I had wanted to say that the location of the family was directly relevant to my hearing the dog, then I would have to put another arc directly between the two. Direct relevance would occur, say, if the dog is more likely to bark when the family is away than when it is at home. This is not the case for my dog.

In the rest of this section, I define the independence assumptions in Bayesian networks and then show how they correspond to what one would expect given the interpretation of the arcs as causal. In the next section, I formally show that once one makes these independence assumptions, the probabilities needed are reduced to the ones that I specified (for roots, the priors; for nonroots, the conditionals given immediate predecessors).

First, I give the rule specifying dependence and independence in Bayesian networks:

In a Bayesian network, a variable a is

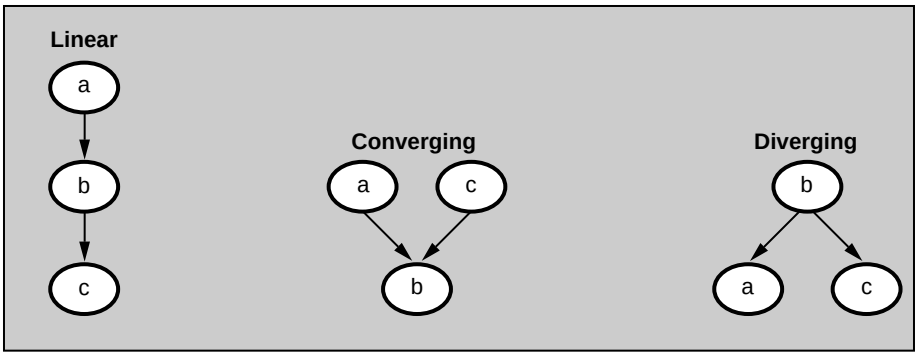


Figure 4. The Three Connection Types.
In each case, node b *is between* a *and* c *in the undirected path between the two.*

dependent on a variable b given evidence E = {e_1 … e_n} if there is a *d-connecting* path from a to b given E. (I call E the evidence nodes. E can be empty. It can not include a or b.) If a is not dependent on b given E, a is independent of b given E.

Note that for any random variable {f} it is possible for two variables to be independent of each other given E but dependent given E ∪ {f} and vise versa (they may be dependent given E but independent given E ∪ {f}. In particular, if we say that two variables a and b are independent of each other, we simply mean that $P(a \mid b) = P(a)$. It might still be the case that they are not independent given, say, e (that is, $P(a \mid b,e) \neq P(a \mid e)$).

To connect this definition to the claim that family-out is independent of hear-bark given dog-out, we see when I explain d-connecting that there is no d-connecting path from family-out to hear-bark given dog-out because dog-out, in effect, blocks the path between the two.

To understand d-connecting paths, we need to keep in mind the three kinds of connections between a random variable b and its two immediate neighbors in the path, a and c. The three possibilities are shown in figure 4 and correspond to the possible combinations of arrow directions from b to a and c. In the first case, one node is above b and the other below; in the second case, both are above; and in the third, both are below. (Remember, we assume that arrows in the diagram go from high to low, so going in the direction of the arrow is going down.) We can say that a node b in a path *P* is linear, converging or diverging in *P* depending on which situation it finds itself according to figure 4.

Now I give the definition of a *d-connecting path*:

A path from q to r is d-connecting with respect to the evidence nodes E if every interior node n in the path has the property that either

1. it is linear or diverging and not a member of E or

2. it is converging, and either n or one of its descendants is in E.

In the literature, the term d-separation is more common. Two nodes are *d-separated* if there is no d-connecting path between them. I find the explanation in terms of d-connecting slightly easier to understand. I go through this definition slowly in a moment, but roughly speaking, two nodes are d-connected if either there is a causal path between them (part 1 of the definition), or there is evidence that renders the two nodes correlated with each other (part 2).

To understand this definition, let's start by pretending the part (2) is not there. Then we would be saying that a d-connecting path must not be blocked by evidence, and there can be no converging interior nodes. We already saw why we want the evidence blocking restriction. This restriction is what says that once we know about a middle node, we do not need to know about anything further away.

What about the restriction on converging nodes? Again, consider figure 2. In this diagram, I am saying that both bowel-problem and family-out can cause dog-out. However, does the probability of bowel-problem depend on that of family-out? No, not really. (We could imagine a case where they were dependent, but this case would be another ball game and another Bayesian network.) Note that the only path between the two is by way of a converging node for this path, namely, dog-out. To put it another way, if two things can cause the same state of affairs and have no other connection, then the two things are independent. Thus, any time we have two potential causes for a state of affairs, we have a converging node. Because one major use of Bayesian networks is deciding which potential cause is the most likely, converging nodes are common.

Now let us consider part 2 in the definition of d-connecting path. Suppose we know that the dog is out (that is, dog-out is a member of E). Now, are family-away and bowel-problem independent? No, even though they were independent of each other when there was no evidence, as I just showed. For example, knowing that the family is at home should raise (slightly) the probability that the dog has bowel problems. Because we eliminated

the most likely explanation for the dog being out, less likely explanations become more likely. This situation is covered by part 2. Here, the d-connecting path is from family-away to bowel-problem . It goes through a converging node (dog-out), but dog-out is itself a conditioning node. We would have a similar situation if we did not know that the dog was out but merely heard the barking. In this case, we would not be sure the dog was out, but we do have relevant evidence (which raises the probability), so hear-bark, in effect, connects the two nodes above the converging node. Intuitively, part 2 means that a path can only go through a converging node if we are conditioning on an (indirect) effect of the converging node.

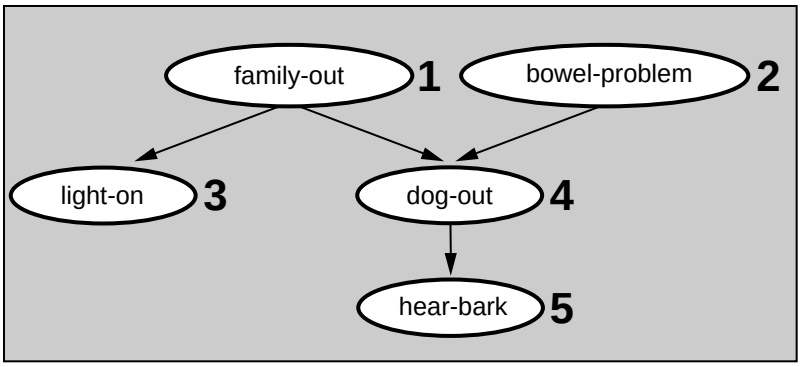


Figure 5. A Topological Ordering.
In this case, I made it a simple top-down numbering.

Consistent Probabilities

One problem that can plague a naive probabilistic scheme is inconsistent probabilities. For example, consider a system in which we have $P(a \mid b) = .7$, $P(b \mid a) = .3$, and $P(b) = .5$. Just eyeballing these equations, nothing looks amiss, but a quick application of Bayes's law shows that these probabilities are not consistent because they require $P(a) > 1$. By Bayes's law,

$P(a)\ P(b \mid a) / P(b) = P(a \mid b)$;

so,

$P(a) = P(b)\ P(b \mid a) / P(b \mid a) = .5 * .7 / .3 = .35 / .3)$.

Needless to say, in a system with a lot of such numbers, making sure they are consistent can be a problem, and one system (PROSPECTOR) had to implement special-purpose techniques to handle such inconsistencies (Duda, Hart, and Nilsson 1976). Therefore, it is a nice property of the Bayesian networks that if you specify the required numbers (the probability of every node given all possible combinations of its parents), then (1) the numbers will be consistent and (2) the network will uniquely define a distribution. Furthermore, it is not too hard to see that this claim is true. To see it, we must first introduce the notion of joint distribution.

A *joint distribution* of a set of random variables $v_1 \ldots v_n$ is defined as $P(v_1 \ldots v_n)$ for all values of $v_1 \ldots v_n$. That is, for the set of Boolean variables (a,b), we need the probabilities $P(a\ b)$, $P(\neg a\ b)$, $P(a \neg b)$, and $P(\neg a \neg b)$. A joint distribution for a set of random variables gives all the information there is about the distribution. For example, suppose we had the just-mentioned joint distribution for (a,b), and we wanted to compute, say, $P(a \mid b)$:

$P(a \mid b) = P(a\ b) / P(b) = P(a\ b) / (P(a\ b) + P(\neg a\ b)$.

Note that for n Boolean variables, the joint distribution contains 2^n values. However, the sum of all the joint probabilities must be 1 because the probability of all possible outcomes must be 1. Thus, to specify the joint distribution, one needs to specify 2_n -1 numbers, thus the 2^n -1 in the last section.

I now show that the joint distribution for a Bayesian network is uniquely defined by the product of the individual distributions for each random variable. That is, for the network in figure 2 and for any combination of values *fo*, *bp*, *lo*, *hb* (for example, *t*, *f*, *f*, *t*, *t*), the joint probability is

$P(fo\ bp\ lo\ do\ hb) = P(fo)P(bp)P(lo \mid fo)P(do \mid fo\ bp)P(hb \mid do)$.

Consider a network N consisting of variables $v_1 \ldots v_n$. Now, an easily proven law of probability is that

$P(v_1 \ldots v_n) = P(v_1)P(v_2 \mid v_1) \ldots P(v_n \mid v_1 \ldots v_{n-1})$.

This equation is true for any set of random variables. We use the equation to factor our joint distribution into the component parts specified on the right-hand side of the equation. Exactly how a particular joint distribution is factored according to this equation depends on how we order the random variables, that is, which variable we make v_1, v_2, and so on. For the proof, I use what is called a *topological sort* on the random variables. This sort is an ordering of the variables such that every variable comes before all its descendants in the graph. Let us assume that $v_1 \ldots v_n$ is such an ordering. In figure 5, I show one such ordering for figure 1.

Let us consider one of the terms in this product, $P(v_j \mid v_{j-1})$. An illustration of what nodes $v_1 \ldots v_j$ might look like is given in figure 6. In this graph, I show the nodes immediately above v_j and otherwise ignore everything except v_c, which we are concen-

…the most important constraint…is …that…this computation is NP-hard…

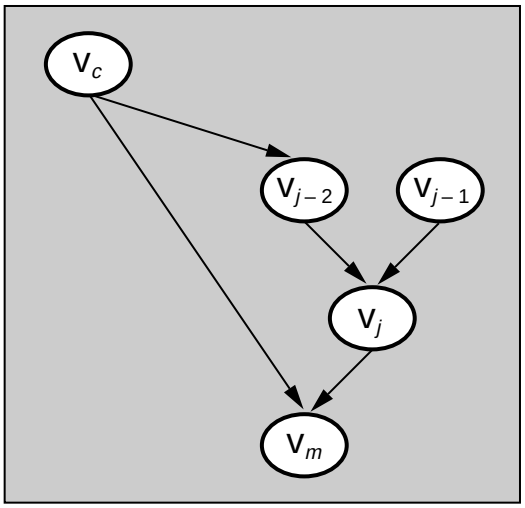


Figure 6. Node v_j in a Network.

I show that when conditioning v_j only on its successors, its value is dependent only on its immediate successors, v_{j-1} and v_{j-2}.

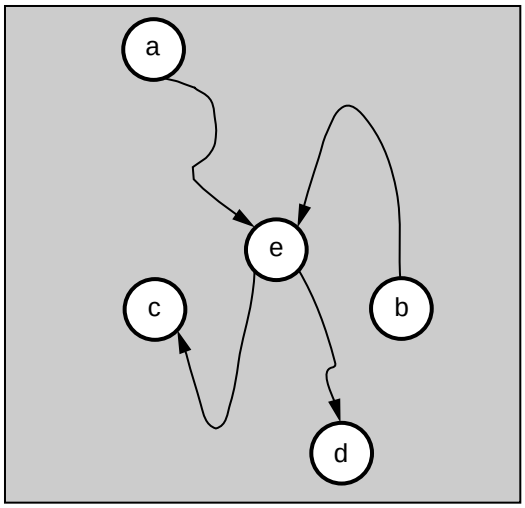


Figure 7. Nodes in a Singly Connected Network.

Because of the singly connected property, any two nodes connected to node e *have only one path between them—the path that goes through* e.

trating on and which connects with v_j in two different ways that we call the left and right paths, respectively. We can see from figure 6 that none of the *conditioning nodes* (the nodes being conditioned on in the conditional probability) in $P(v_j \mid v_1 \ldots v_j - 1)$ is below v_j (in particular, v_m is not a conditioning node). This condition holds because of the way in which we did the numbering.

Next, we want to show that all and only the parents of v_j need be in the conditioning portion of this term in the factorization. To see that this is true, suppose v_c is not immediately above v_j but comes before v_j in the numbering. Then any path between v_c and v_j must either be blocked by the nodes just above v_j (as is the right path from v_c in figure 6) or go through a node lower than v_j (as is the left path in figure 6). In this latter case, the path is not d-connecting because it goes through a converging node v_m where neither it nor any of its descendants is part of the conditioning nodes (because of the way we numbered). Thus, no path from v_c to v_j can be d-connecting, and we can eliminate v_c from the conditioning section because by the independence assumptions in Bayesian networks, v_j is independent of v_c given the other conditioning nodes. In this fashion, we can remove all the nodes from the conditioning case for $P(v_j \mid v_1 \ldots v_{j-1})$ except those immediately above v_j. In figure 6, this reduction would leave us with $P(v_j \mid v_{j-1}\, v_{j-2})$. We can do this for all the nodes in the product. Thus, for figure 2, we get

$P(fo\ bp\ lo\ do\ hb) = P(fo)P(bp)P(lo \mid fo)P(do \mid fo\ bp)P(hb \mid do)$.

We have shown that the numbers specified by the Bayesian network formalism in fact define a single joint distribution, thus uniqueness. Furthermore, if the numbers for each local distribution are consistent, then the global distribution is consistent. (Local consistency is just a matter of having the right numbers sum to 1.)

Evaluating Networks

As I already noted, the basic computation on belief networks is the computation of every node's belief (its conditional probability) given the evidence that has been observed so far. Probably the most important constraint on the use of Bayesian networks is the fact that in general, this computation is NP-hard (Cooper 1987). Furthermore, the exponential time limitation can and does show up on realistic networks that people actually want solved. Depending on the particulars of the network, the algorithm used, and the care taken in the implementation, networks as small as tens of nodes can take too long, or networks in the thousands of nodes can be done in acceptable time.

The first issue is whether one wants an

Bayesian networks have been extended to handle decision theory.

exact solution (which is NP-hard) or if one can make do with an approximate answer (that is, the answer one gets is not exact but with high probability is within some small distance of the correct answer). I start with algorithms for finding exact solutions.

Exact Solutions

Although evaluating Bayesian networks is, in general, NP-hard, there is a restricted class of networks that can efficiently be solved in time linear in the number of nodes. The class is that of singly connected networks. A *singly connected network* (also called a *polytree*) is one in which the underlying undirected graph has no more than one path between any two nodes. (The underlying undirected graph is the graph one gets if one simply ignores the directions on the edges.) Thus, for example, the Bayesian network in figure 5 is singly connected, but the network in figure 6 is not. Note that the direction of the arrows does not matter. The left path from v_c to v_j requires one to go against the direction of the arrow from v_m to v_j. Nevertheless, it counts as a path from v_m to v_j.

The algorithm for solving singly connected Bayesian networks is complicated, so I do not give it here. However, it is not hard to see why the singly connected case is so much easier. Suppose we have the case sketchily illustrated in figure 7 in which we want to know the probability of e given particular values for a, b, c, and d. We specify that a and b are above e in the sense that the last step in going from them to e takes us along an arrow pointing down into e. Similarly, we assume c and d are below e in the same sense. Nothing in what we say depends on exactly how a and b are above e or how d and c are below. A little examination of what follows shows that we could have any two sets of evidence (possibly empty) being above and below e rather than the sets {a b} and {c d}. We have just been particular to save a bit on notation.

What does matter is that there is only one way to get from any of these nodes to e and that the only way to get from any of the nodes a, b, c, d to any of the others (for example, from b to d) is through e. This claim follows from the fact that the network is singly connected. Given the singly connected condition, we show that it is possible to break up the problem of determining $P(e \mid a\ b\ c\ d)$ into two simpler problems involving the network from e up and the network from e down. First, from Bayes's rule,

$$P(e \mid a\ b\ c\ d) = P(e)\ P(a\ b\ c\ d \mid e) / P(a\ b\ c\ d) \ .$$

Taking the second term in the numerator, we can break it up using conditioning:

$$P(e \mid a\ b\ c\ d) = P(e)\ P(a\ b \mid e)\ P(c\ d \mid a\ b\ e) / P(a\ b\ c\ d) \ .$$

Next, note that $P(c\ d \mid a\ b\ e) = P(c\ d \mid e\ )$ because e separates a and b from c and d (by the singly connected condition). Substituting this term for the last term in the numerator and conditioning the denominator on a, b, we get

$$P(e \mid a\ b\ c\ d) = P(e)\ P(a\ b \mid e)\ P(c\ d \mid e) / P(a\ b)\ P(c\ d \mid a\ b) \ .$$

Next, we rearrange the terms to get

$$P(e \mid a\ b\ c\ d) = (P(e)\ P(a\ b \mid e) / P(a\ b))\ (P(c\ d \mid e)\ (P(c\ d \mid a\ b)) \ .$$

Apply Bayes's rule in reverse to the first collection of terms, and we get

$$P(e \mid a\ b\ c\ d) = (P(e \mid a\ b\ )\ P(c\ d \mid e))\ (1 / P(c\ d \mid a\ b)) \ .$$

We have now done what we set out to do. The first term only involves the material from e up and the second from e down. The last term involves both, but it need not be calculated. Rather, we solve this equation for all values of e (just true and false if e is Boolean). The last term remains the same, so we can calculate it by making sure that the probabilities for all the values of E sum to 1. Naturally, to make this sketch into a real algorithm for finding conditional probabilities for polytree Bayesian networks, we need to show how to calculate $P(e \mid a\ b)$ and $P(c\ d \mid e)$, but the ease with which we divided the problem into two distinct parts should serve to indicate that these calculations can efficiently be done. For a complete description of the algorithm, see Pearl (1988) or Neapolitan (1990).

Now, at several points in the previous discussion, we made use of the fact that the network was singly connected, so the same argument does not work for the general case.

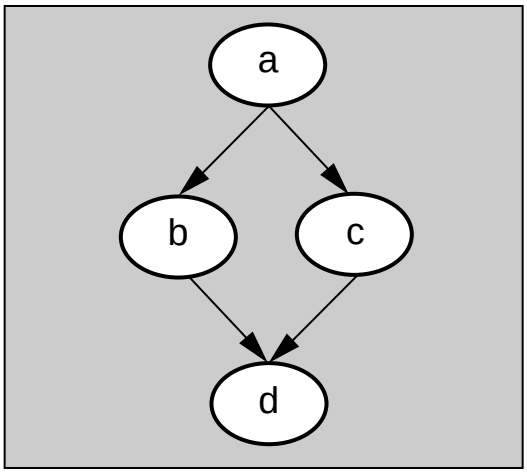


Figure 8. A Multiply Connected Network.
There are two paths between node a *and node* d.

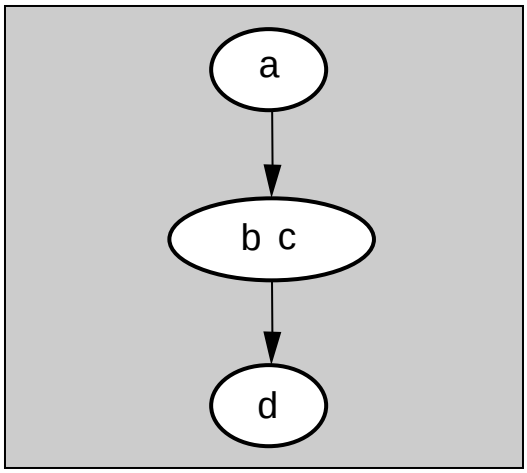


Figure 9. A Clustered, Multiply Connected Network.
By clustering nodes b *and* c*, we turned the graph of figure 8 into a singly connected network.*

However, exactly what is it that makes multiply connected networks hard? At first glance, it might seem that any belief network ought to be easy to evaluate. We get some evidence. Assume it is the value of a particular node. (If it is the values of several nodes, we just take one at a time, reevaluating the network as we consider each extra fact in turn.) It seems that we located at every node all the information we need to decide on its probability. That is, once we know the probability of its neighbors, we can determine its probability. (In fact, all we really need is the probability of its parents.)

These claims are correct but misleading. In singly connected networks, a change in one neighbor of e cannot change another neighbor of e except by going through e itself. This is because of the single-connection condition. Once we allow multiple connections between nodes, calculations are not as easy. Consider figure 8. Suppose we learn that node d has the value true, and we want to know the conditional probabilities at node c. In this network, the change at d will affect c in more than one way. Not only does c have to account for the direct change in d but also the change in a that will be caused by d through b. Unlike before, these changes do not separate cleanly.

To evaluate multiply connected networks exactly, one has to turn the network into an equivalent singly connected one. There are a few ways to perform this task. The most common ways are variations on a technique called clustering. In *clustering*, one combines nodes until the resulting graph is singly connected. Thus, to turn figure 8 into a singly connected network, one can combine nodes b and c. The resulting graph is shown in figure 9. Note now that the node {b c} has as its values the cross-product of the values of b and c singly. There are well-understood techniques for producing the necessary local probabilities for the clustered network. Then one evaluates the network using the singly connected algorithm. The values for the variables from the original network can then be read off those of the clustered network. (For example, the values of b and c can easily be calculated from the values for {b c}.) At the moment, a variant of this technique proposed by Lauritzen and Spiegelhalter (1988) and improved by Jensen (1989) is the fastest exact algorithm for most applications. The problem, of course, is that the nodes one creates might have large numbers of values. A node that was the combination of 10 Boolean-valued nodes would have 1024 values. For dense networks, this explosion of values and worse can happen. Thus, one often considers settling for approximations of the exact value. We turn to this area next.

Approximate Solutions

There are a lot of ways to find approximations of the conditional probabilities in a Bayesian network. Which way is the best depends on the exact nature of the network.

However, many of the algorithms have a lot in common. Essentially, they randomly posit values for some of the nodes and then use them to pick values for the other nodes. One then keeps statistics on the values that the nodes take, and these statistics give the answer. To take a particularly clear case, the technique called *logic sampling* (Henrion 1988) guesses the values of the root nodes in accordance with their prior probabilities. Thus, if v is a root node, and $P(v) = .2$, one randomly chooses a value for this node but in such a way that it is true about 20 percent of the time. One then works one's way down the network, guessing the value of the next lower node on the basis of the values of the higher nodes. Thus, if, say, the nodes a and b, which are above c, have been assigned true and false, respectively, and $P(c \mid \neg b) = .8$, then we pick a random number between 0 and 1, and if it is less than .8, we assign c to true, otherwise, false. We do this procedure all the way down and track how often each of our nodes is assigned to each of its values. Note that, as described, this procedure does not take evidence nodes into account. This problem can be fixed, and there are variations that improve it for such cases (Shacter and Peot 1989; Shwe and Cooper 1990). There are also different approximation techniques (see Horvitz, Suermondt, and Cooper [1989]). At the moment, however, there does not seem to be a single technique, either approximate or exact, that works well for all kinds of networks. (It is interesting that for the exact algorithms, the feature of the network that determines performance is the topology, but for the approximation algorithms, it is the quantities.) Given the NP-hard result, it is unlikely that we will ever get an exact algorithm that works well for all kinds of Bayesian networks. It might be possible to find an approximation scheme that works well for everything, but it might be that in the end, we will simply have a library of algorithms, and researchers will have to choose the one that best suits their problem.

Finally, I should mention that for those who have Bayesian networks to evaluate but do not care to implement the algorithms themselves, at least two software packages are around that implement some of the algorithms I mentioned: IDEAL (Srinivas and Breese 1989, 1990) and HUGIN (Andersen 1989).

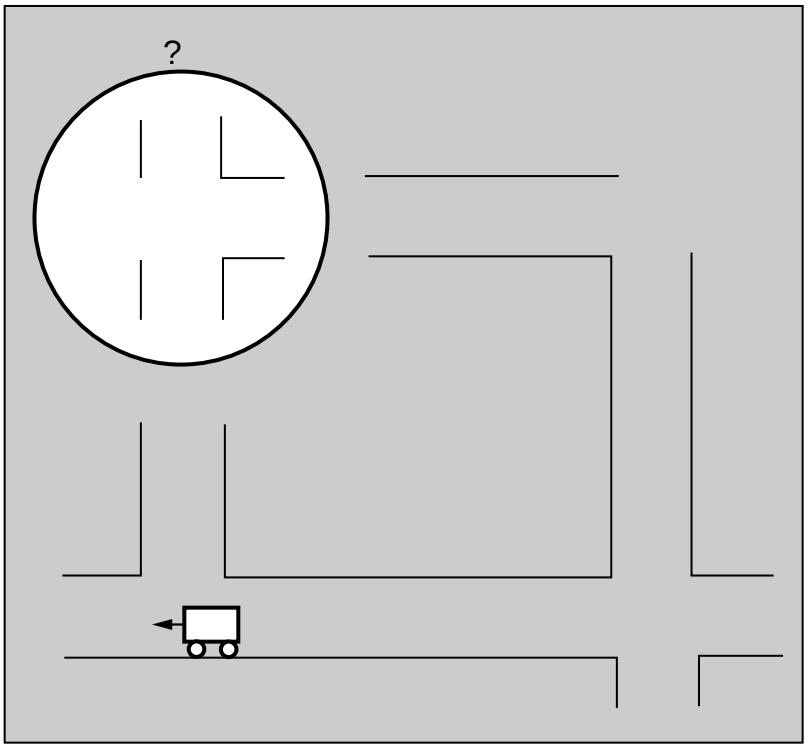


Figure 10. Map Learning.

Finding the north-south corridor makes it more likely that there is an intersection north of the robot's current location.

Applications

As I stated in the introduction, Bayesian networks are now being used in a variety of applications. As one would expect, the most common is diagnosis problems, particularly, medical diagnosis. A current example of the use of Bayesian networks in this area is PATHFINDER (Heckerman 1990), a program to diagnose diseases of the lymph node. A patient suspected of having a lymph node disease has a lymph node removed and examined by a pathologist. The pathologist examines it under a microscope, and the information gained thereby, possibly together with other tests on the node, leads to a diagnosis. PATHFINDER allows a physician to enter the information and get the conditional probabilities of the diseases given the evidence so far.

PATHFINDER also uses decision theory. Decision theory is a close cousin of *probability theory* in which one also specifies the desirability of various outcomes (their utility) and the costs of various actions that might be performed to affect the outcomes. The idea is to find the action (or plan) that maximizes the expected utility minus costs. Bayesian networks have been extended to handle decision theory. A Bayesian network that incorporates *decision nodes* (nodes indicating actions that can be performed) and *value nodes* (nodes indicating the values of various outcomes) is

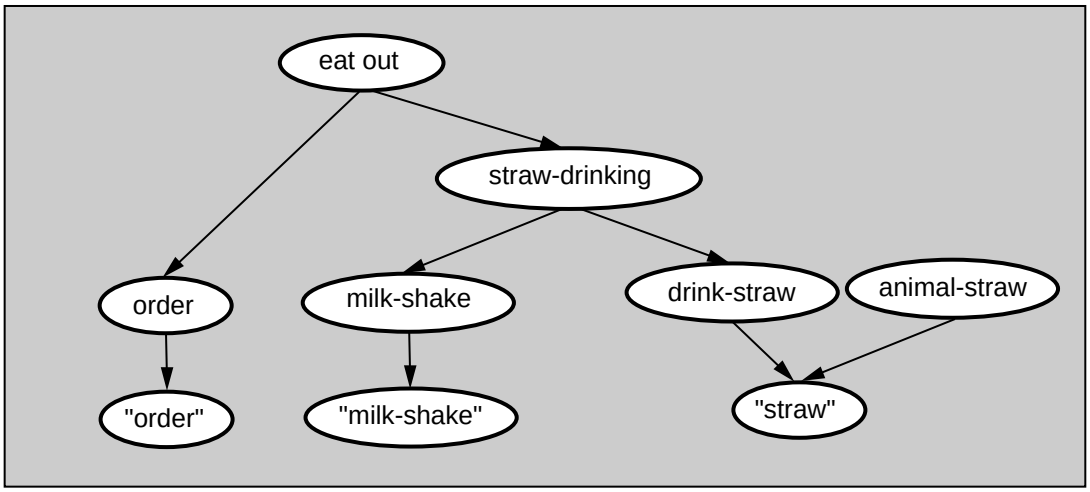


Figure 11. Bayesian Network for a Simple Story.
Connecting "straw" to the earlier context makes the drink-straw *reading more likely.*

called an *influence diagram*, a concept invented by Howard (Howard and Matheson 1981). In PATHFINDER, decision theory is used to choose the next test to be performed when the current tests are not sufficient to make a diagnosis. PATHFINDER has the ability to make treatment decisions as well but is not used for this purpose because the decisions seem to be sensitive to details of the utilities. (For example, how much treatment pain would you tolerate to decrease the risk of death by a certain percentage?)

PATHFINDER's model of lymph node diseases includes 60 diseases and over 130 features that can be observed to make the diagnosis. Many of the features have more than 2 possible outcomes (that is, they are not binary valued). (Nonbinary values are common for laboratory tests with real-number results. One could conceivably have the possible values of the random variable be the real numbers, but typically to keep the number of values finite, one breaks the values into significant regions. I gave an example of this early on with earthquake, where we divided the Richter scale for earthquake intensities into 5 regions.) Various versions of the program have been implemented (the current one is PATHFINDER-4), and the use of Bayesian networks and decision theory has proven better than (1) MYCIN-style certainty factors (Shortliffe 1976), (2) Dempster-Shafer theory of belief (Shafer 1976), and (3) simpler Bayesian models (ones with less realistic independence assumptions). Indeed, the program has achieved expert-level performance and has been implemented commercially.

Bayesian networks are being used in less obvious applications as well. At Brown University, there are two such applications: map learning (the work of Ken Basye and Tom Dean) and story understanding (Robert Goldman and myself). To see how Bayesian networks can be used for map learning, imagine that a robot has gone down a particular corridor for the first time, heading, say, west. At some point, its sensors pick up some features that most likely indicate a corridor heading off to the north (figure 10). Because of its current task, the robot keeps heading west. Nevertheless, because of this information, the robot should increase the probability that a known east-west corridor, slightly to the north of the current one, will also intersect with this north-south corridor. In this domain, rather than having diseases that cause certain abnormalities, which, in turn, are reflected as test results, particular corridor layouts cause certain kinds of junctions between corridors, which, in turn, cause certain sensor readings. Just as in diagnosis, the problem is to reason backward from the tests to the diseases; in map learning, the problem is to reason backward from the sensor readings to the corridor layout (that is, the map). Here, too, the intent is to combine this diagnostic problem with decision theory, so the robot could weigh the alternative of deviating from its planned course to explore portions of the building for which it has no map.

My own work on story understanding (Charniak and Goldman 1989a, 1991; Goldman 1990) depends on a similar analogy.

Imagine, to keep things simple, that the story we are reading was created when the writer observed some sequence of events and wrote them down so that the reader would know what happened. For example, suppose Sally is engaged in shopping at the supermarket. Our observer sees Sally get on a bus, get off at the supermarket, and buy some bread. He/she writes this story down as a string of English words. Now the "disease" is the high-level hypothesis about Sally's task (shopping). The intermediate levels would include things such as what the writer actually saw (which was things such as traveling to the supermarket—note that "shopping" is not immediately observable but, rather, has to be put together from simpler observations). The bottom layer in the network, the "evidence," is now the English words that the author put down on paper.

In this framework, problems such as, say, word-sense ambiguity, become intermediate random variables in the network. For example, figure 11 shows a simplified version of the network after the story "Sue ordered a milkshake. She picked up the straw." At the top, we see a hypothesis that Sue is eating out. Below this hypothesis is one that she will drink a milkshake (in a particular way called, there, straw-drinking). Because this action requires a drinking straw, we get a connection to this word sense. At the bottom of the network, we see the word straw, which could have been used if the author intended us to understand the word as describing either a drinking straw or some animal straw (that is, the kind animals sleep on). As one would expect for this network, the probability of drinking-straw will be much higher given the evidence from the words because the evidence suggests a drinking event, which, in turn, makes a drinking straw more probable. Note that the program has a knowledge base that tells it how, in general, eating out relates to drinking (and, thus, to straw drinking), how straw drinking relates to straws, and so on. This knowledge base is then used to construct, on the fly, a Bayesian network (like the one in figure 11) that represents a particular story.

But Where Do the Numbers Come From?

One of the points I made in this article is the beneficial reduction in the number of parameters required by Bayesian networks. Indeed, if anything, I overstated how many numbers are typically required in a Bayesian network. For example, a common situation is to have several causes for the same result. This situation occurs when a symptom is caused by several diseases, or a person's action could be the result of several plans. This situation is shown in figure 12. Assuming all Boolean nodes, the node fever would require 8 conditional probabilities. However, doctors would be unlikely to know such numbers. Rather, they might know that the probability of a fever is .8 given a cold; .98 given pneumonia; and, say, .4 given chicken pox. They would probably also say that the probability of fever given 2 of them is slightly higher than either alone. Pearl suggested that in such cases, we should specify the probabilities given individual causes but use stereotypical combination rules for combining them when more than 1 case is present. The current case would be handled by Pearl's noisy-Or random variable. Thus, rather than specifying 8 numbers, we only need to specify 3. We require still fewer numbers.

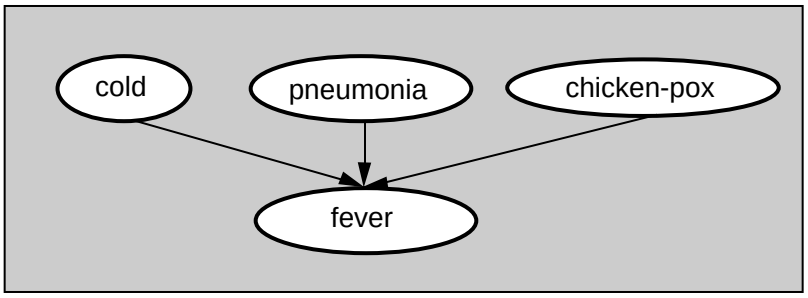


Figure 12. Three Causes for a Fever.

Viewing the fever node as a noisy-Or node makes it easier to construct the posterior distribution for it.

However, fewer numbers is not no numbers at all, and the skeptic might still wonder how the numbers that are still required are, in fact, obtained. In all the examples described previously, they are made up. Naturally, nobody actually makes this statement. What one really says is that they are elicited from an expert who subjectively assesses them. This statement sounds a lot better, but there is really nothing wrong with making up numbers. For one thing, experts are fairly good at it. In one study (Spiegelhalter, Franklin, and Bull 1989), doctors' assessments of the numbers required for a Bayesian network were compared to the numbers that were subsequently collected and found to be pretty close (except the doctors were typically too quick in saying that things had zero probability). I also suspect that some of the prejudice against making up numbers (but not, for

…the major drawback to their use is the time of evaluation…

example, against making up rules) is that one fears that any set of examples can be explained away by merely producing the appropriate numbers. However, with the reduced number set required by Bayesian networks, this fear is no longer justified; any reasonably extensive group of test examples overconstrains the numbers required.

In a few cases, of course, it might actually be possible to collect data and produce the required numbers in this way. When this is possible, we have the ideal case. Indeed, there is another way of using probabilities, where one constrains one's theories to fit one's data-collection abilities. Mostly, however, Bayesian network practitioners subjectively access the probabilities they need.

Conclusions

Bayesian networks offer the AI researcher a convenient way to attack a multitude of problems in which one wants to come to conclusions that are not warranted logically but, rather, probabilistically. Furthermore, they allow you to attack these problems without the traditional hurdles of specifying a set of numbers that grows exponentially with the complexity of the model. Probably the major drawback to their use is the time of evaluation (exponential time for the general case). However, because a large number of people are now using Bayesian networks, there is a great deal of research on efficient exact solution methods as well as a variety of approximation schemes. It is my belief that Bayesian networks or their descendants are the wave of the future.

Acknowledgments

Thanks to Robert Goldman, Solomon Shimony, Charles Moylan, Dzung Hoang, Dilip Barman, and Cindy Grimm for comments on an earlier draft of this article and to Geoffrey Hinton for a better title, which, unfortunately, I could not use. This work was supported by the National Science Foundation under grant IRI-8911122 and the Office of Naval Research under contract N00014-88-K-0589.

References

Andersen, S. 1989. HUGIN—A Shell for Building Bayesian Belief Universes for Expert Systems. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1080–1085. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Charniak, E., and Goldman, R. 1991. A Probabilistic Model of Plan Recognition. In Proceedings of the Ninth National Conference on Artificial Intelligence, 160–165. Menlo Park, Calif.: American Association for Artificial Intelligence.

Charniak, E., and Goldman, R. 1989a. A Semantics for Probabilistic Quantifier-Free First-Order Languages with Particular Application to Story Understanding. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1074–1079. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Charniak, E., and Goldman, R. 1989b. Plan Recognition in Stories and in Life. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 54–60. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Charniak, E., and McDermott, D. 1985. *Introduction to Artificial Intelligence*. Reading, Mass.: Addison-Wesley.

Cooper, G. F. 1987. Probabilistic Inference Using Belief Networks is NP-Hard, Technical Report, KSL-87-27, Medical Computer Science Group, Stanford Univ.

Dean, T. 1990. Coping with Uncertainty in a Control System for Navigation and Exploration. In Proceedings of the Ninth National Conference on Artificial Intelligence, 1010–1015. Menlo Park, Calif.: American Association for Artificial Intelligence.

Duda, R.; Hart, P.; and Nilsson, N. 1976. Subjective Bayesian Methods for Rule-Based Inference Systems. In Proceedings of the American Federation of Information Processing Societies National Computer Conference, 1075–1082. Washington, D.C.: American Federation of Information Processing Societies.

Goldman, R. 1990. A Probabilistic Approach to Language Understanding, Technical Report, CS-90-34, Dept. of Computer Science, Brown Univ.

Hansson, O., and Mayer, A. 1989. Heuristic Search as Evidential Reasoning. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 152–161. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Heckerman, D. 1990. Probabilistic Similarity Networks, Technical Report, STAN-CS-1316, Depts. of Computer Science and Medicine, Stanford Univ.

Henrion, M. 1988. Propagating Uncertainty in Bayesian Networks by Logic Sampling. In *Uncertainty in Artificial Intelligence 2*, eds. J. Lemmer and L. Kanal, 149–163. Amsterdam: North Holland.

Horvitz, E.; Suermondt, H.; and Cooper, G. 1989. Bounded Conditioning: Flexible Inference for Decisions under Scarce Resources. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 182–193. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Howard, R. A., and Matheson, J. E. 1981. Influence Diagrams. In *Applications of Decision Analysis,* volume 2, eds. R. A. Howard and J. E. Matheson, 721–762. Menlo Park, Calif.: Strategic Decisions Group.

Jensen, F. 1989. Bayesian Updating in Recursive Graphical Models by Local Computations, Technical Report, R-89-15, Dept. of Mathematics and Computer Science, University of Aalborg.

Lauritzen, S., and Spiegelhalter, D. 1988. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society* 50:157–224.

Levitt, T.; Mullin, J.; and Binford, T. 1989. Model-Based Influence Diagrams for Machine Vision. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 233–244. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Neapolitan, E. 1990. *Probabilistic Reasoning in Expert Systems*. New York: Wiley.

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

Shacter, R., and Peot, M. 1989. Simulation Approaches to General Probabilistic Inference on Belief Networks. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 311–318. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Shafer, G. 1976. *A Mathematical Theory of Evidence*. Princeton, N.J.: Princeton University Press.

Shortliffe, E. 1976. *Computer-Based Medical Consultations: MYCIN*. New York: American Elsevier.

Shwe, M., and Cooper, G. 1990. An Empirical Analysis of Likelihood-Weighting Simulation on a Large, Multiply Connected Belief Network. In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, 498–508. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Spiegelhalter, D.; Franklin, R.; and Bull, K. 1989. Assessment Criticism and Improvement of Imprecise Subjective Probabilities for a Medical Expert System. In Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, 335–342. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Srinivas, S., and Breese, J. 1990. IDEAL: A Software Package for Analysis of Influence Diagrams. In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, 212–219. Mountain View, Calif.: Association for Uncertainty in Artificial Intelligence.

Srinivas, S., and Breese, J. 1989. IDEAL: Influence Diagram Evaluation and Analysis in Lisp, Technical Report, Rockwell International Science Center, Palo Alto, California.



Eugene Charniak is a professor of computer science and cognitive science at Brown University and the chairman of the Department of Computer Science. He received his B.A. degree in physics from the University of Chicago and his Ph.D. in computer science from the Massachusetts Institute of Technology. He has published three books: *Computational Semantics*, with Yorick Wilks (North Holland, 1976); *Artificial Intelligence Programming* (now in a second edition), with Chris Riesbeck, Drew McDermott, and James Meehan (Lawrence Erlbaum, 1980, 1987); and *Introduction to Artificial Intelligence,* with Drew McDermott (Addison-Wesley, 1985). He is a fellow of the American Association of Artificial Intelligence and was previously a councilor of the organization. He is on the editorial boards of the journals *Cognitive Science* (of which he is was a founding editor) and *Computational Linguistics*. His research has been in the area of language understanding (particularly in the resolution of ambiguity in noun-phrase reference, syntactic analysis, case determination, and word-sense selection); plan recognition; and, more generally, abduction. In the last few years, his work has concentrated on the use of probability theory to elucidate these problems, particularly on the use of Bayesian nets (or belief nets) therein.