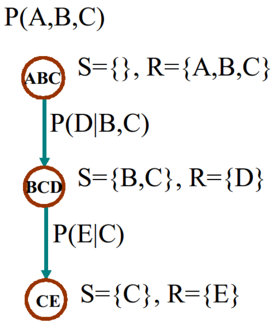
Lecture 10: Probability Propagation in Join Trees

In the last lecture we looked at how the variables of a Bayesian network could be grouped together in cliques of dependency, so that probability propagation could be carried out in a tree of cliques (or join tree), thus getting round the problem of loops in multiply connected networks. We will now look in detail at how this probability propagation is done.

States of a clique

Each node or clique is made up of several variables W, and therefore will have a state for each possible combination of those variables. The variables are divided into those that appear higher in the join tree S and those that do not R. The data associated with the join tree that we used in the last lecture can be summarised as follows:

Node	W	S	R	States
1	ABC		ABC	$(a_1,b_1,c_1)(a_1,b_1,c_2)(a_1,b_2,c_1)(a_1,b_2,c_2)$ $(a_2,b_1,c_1)(a_2,b_1,c_2)(a_2,b_2,c_1)(a_2,b_2,c_2)$
2	BCD	BC	D	$(b_1,c_1,d_1)(b_1,c_2,d_1)(b_2,c_1,d_1)(b_2,c_2,d_1)$ $(b_1,c_1,d_2)(b_1,c_2,d_2)(b_2,c_1,d_2)(b_2,c_2,d_2)$
3	CE	C	E	$(c_1,e_1)(c_1,e_2)(c_2,e_1)(c_2,e_2)$



Calculating the probabilities

The objective of the propagation algorithm is to find the joint probabilities of the join tree nodes for a given instantiation. This is not the same as the potential function of the node. Initially we set the potential function of each node using the conditional probabilities of the original network, allocating them according to the variables whose parents are in the same node. In this case they are:

$$\Psi(W_1) = P(A)P(B|A)P(C|A)$$
$$\Psi(W_2) = P(D|B\&C)$$
$$\Psi(W_3) = P(E|C)$$

and our task is to calculate

$$P(W_1) = P(A\&B\&C)$$
$$P(W_2) = P(B\&C\&D)$$
$$P(W_3) = P(C\&E)$$

If we can manage to get the joint probabilities of the cliques we can then easily get the individual probabilities of the variables in the original causal network using marginalisation. For example:

$$P(B) = \sum_{AC} P(A\&B\&C)$$

where the notation $\sum_{AC}$ means sum the joint distribution over all possible values of A and C. For the individual states of B in the above table we have:

$$P(b_1) = P(a_1\&b_1\&c_1) + P(a_1\&b_1\&c_2) + P(a_2\&b_1\&c_1) + P(a_2\&b_1\&c_2)$$
$$P(b_2) = P(a_1\&b_2\&c_1) + P(a_1\&b_2\&c_2) + P(a_2\&b_2\&c_1) + P(a_2\&b_2\&c_2)$$

Computing probabilities of the cliques

As noted above, each clique has a set of variables, and we have divided them into those that have been seen higher in the tree (S) and those that have not (R). Thus the joint probability of a node can be written:

$$P(W_i) = P(R_i\&S_i) = P(R_i|S_i)P(S_i)$$

We can think of $P(S_i)$ as being information from higher up in the tree, ie it behaves like π evidence.

The algorithm for calculating the probabilities of cliques goes in two stages. First we calculate $P(R_i|S_i)$ moving up the tree, then we calculate $P(S_i)$ moving down the tree. To see how this works, we will now calculate these probabilities for the case where there are no instantiated variables (ie find the initial probabilities for the cliques, equivalent to the initialisation step in the message passing algorithm).

The key idea is that we can use the potential table to calculate a value of $P(R|S)$ using the formula:

$$P(R|S) = \Psi(W_i)/\textstyle\sum_R \Psi(W_i)$$

A general proof of this is given in Neopolitans first book. If we think of the potential table as a joint probability matrix with each row being one joint state of the R variables and each column being one joint state of the S variables, then dividing by $\sum_R \Psi(W_i)$ can be thought of as normalising each column to sum to 1. To clarify this, we will note two special cases that always occur at initialisation. First, if the potential function is the joint probability of the clique variables (as is the case with the root node) then we can write:

$$P(R|S) = P(R\&S)/P(S)$$

and clearly

$$P(S) = \textstyle\sum_R P(R\&S)$$

hence

$$P(R|S) = P(R\&S)/\textstyle\sum_R P(R\&S) = \Psi(W_i)/\textstyle\sum_R \Psi(W_i)$$

Secondly if the potential function is already in the form $P(R|S)$, which it is initially for the other two cliques in our example, we have that

$$\textstyle\sum_R P(R|S) = [1, 1, \cdots 1]$$

since the columns of a conditional probability matrix will sun to 1. Thus the value of P(R—S) remains unchanged.

The term $\sum_R \Psi(W_i)$ is evidence for S (since all the variation with respect to R has been summed out) and it is this that forms the λ message that we propagate. During propagation at a clique we therefore first calculate the λ evidence $\sum_R \Psi(W_i)$, then we divide the cliques potential function by it so that the potential function becomes $P(R|S)$ and then we update the parent clique by multiplying its potential function by the same evidence. Hence, after propagation, the product of the potential functions still represents the joint probability of the original variables:

$$P(V) = \prod_i \Psi(W_i) \tag{1}$$

Let's consider what happens when evidence is multiplied to the potential function of a parent node. If, for example, the evidence was on a single variable $\lambda(C) = [\lambda(c_1)\lambda(c_2)]$ each entry in the parents potential function containing c_1 is multiplied by $\lambda(c_1)$ and each entry containing c_2 is multiplied by $\lambda(c_2)$. Thus, for example $\Psi(b_1, c_2, d_1)$ becomes $\Psi(b_1, c_2, d_1) \times \lambda(c2)$. Looking at Clique 3 we calculate the λ evidence using:

$$\textstyle\sum_R \Psi(W_i) = \textstyle\sum_E P(E|C) = [1, 1]$$

Copndequently the λ message contains no evidence, and neither dividing it into $\Psi(W_3)$ nor multiplying it into $\Psi(W_3)$ have any effect. The potential function of this node and its parent remain unchanged.

We now turn our attention to the question of node 2, and begin by calculating the λ evidence that it sends to node 1.

$$\textstyle\sum_{R_2} \Psi(W_2) = \textstyle\sum_D P(D|B\&C) = [1, 1, 1, 1] = \lambda(B\&C)$$

Again this turns out to be no evidence, so dividing it into $\Psi(W_2)$ and multiplying it into $\Psi(W_1)$ has no effect, and we see that $P(R_2|S_2) = \Psi(W_2)/\sum_{R_2} \Psi(W_2) = P(D|B\&C)/\sum_D P(D|B\&C) = P(D|B\&C)$

The root node has $R = [A, B, C]$ and $S = \Phi$. Thus $P(R|S) = P(R) = P(A\&B\&C) = P(A)P(B|A)P(C|A)$, and it turns out that $P(R_1|S_1) = \Psi(W_1)$.

Overall nothing has changed during the propagation of the λ evidence, and this is not surprising since nothing has been instantiated, and hence there can be no λ evidence.

Finding the $P(S)$ values (π messages)

The $P(S_i)$ values are calculated from the root downwards. Node 1 has $S_1 = \Phi$, so $P(S_1) = 1$, and thus for node 1 we have already found the joint probability distribution of the variables. In this case it has turned out that:

$$P(W_1) = P(R_1|S_1)P(S_1) = P(R_1) = P(A\&B\&C) = P(A)P(B|A)PC|A) = \Psi(W_1)$$

At this stage we can if we wish calculate the probabilities of the individual variables A, B and C:

$$\begin{array}{c} P(A) = \sum_{BC} P(A\&B\&C) = \sum_{BC} \Psi(W_1) \\ P(B) = \sum_{AC} P(A\&B\&C) = \sum_{AC} \Psi(W_1) \\ P(C) = \sum_{AB} P(A\&B\&C) = \sum_{AB} \Psi(W_1) \end{array}$$

At node 2 we have $S_2 = [B\&C]$, and we can calculate $P(S_2)$ by marginalisation from the joint probability of node 1:

$$P(S_2) = P(B\&C) = \sum_A P(A\&B\&C)$$

the individual table entries are calculated using

$$\begin{array}{c} P(b1\&c1) = P(a1\&b1\&c1) + P(a2\&b1\&c1) \\ P(b1\&c2) = P(a1\&b1\&c2) + P(a2\&b1\&c2) \\ \text{etc} \end{array}$$

The value of $P(S_2)$ now allows us to compute the joint probability of the variables of node 2:

$$P(W_2) = P(R_2|S_2)P(S_2) = \Psi(W_2) \sum_A P(A\&B\&C) = P(B\&C\&D)$$

In this case we see that the joint probability table is different from the node potential table. We can now calculate the individual variable probabilities by marginalisation, though it is only necessary to calculate $P(D)$ since we have already calculated $P(B)$ and $P(C)$.

We continue the process as before, and marginalise to find $P(S_3)$:

$$P(S_3) = P(C) = \sum_{BD} P(B\&C\&D)$$

And complete the probability propagation by calculating:

$$P(W_3) = P(R_3|S_3)P(S_3) = \Psi(W3) \sum_{BD} P(B\&C\&D)$$

Finally we calculate $P(E)$ by marginalisation of $P(W_3)$. Just as in the initialisation step of the message passing algorithm, all λ messages turn out to be 1, but the π messages, derived from $P(A)$, propagate downwards.

Instantiation

If we instantiate a node in a causal network the equivalent in a join tree can be thought of as changing the corresponding clique potential function. Let us consider instantiating the variable D to state d_1 in our example. Suppose that the potential table for node 2 was:

	$b_1\&c_1$	$b_1\&c_2$	$b_2\&c_1$	$b_2\&c_2$
d_1	0.8	0.9	0.7	0.05
d_2	0.2	0.1	0.3	0.95

Instantiation can be thought of as setting all the entries for any joint state involving d_2 to zero in the potential function for node 2:

	$b_1\&c_1$	$b_1\&c_2$	$b_2\&c_1$	$b_2\&c_2$
d_1	0.8	0.9	0.7	0.05
d_2	0	0	0	0

After instantiating D we have to re-compute the clique probabilities using the same algorithm as before. We start at the bottom, and again we find that, as before, for node 3:

$$P(R_3|S_3) = \Psi(W_3) = P(E|C)$$

As before the message to be propagated upwards will be [1,1], no evidence. However, in node 2 things have changed because of the instantiation of D: We first calculate the λ message using:

$$\lambda(B\&C) = \sum_D P(D|B\&C)$$

This can be viewed as summing the columns of the potential table:

	$b_1\&c_1$	$b_1\&c_2$	$b_2\&c_1$	$b_2\&c_2$
d_1	0.8	0.9	0.7	0.05
d_2	0	0	0	0
$\sum_D P(D|B\&C)$	0.8	0.9	0.7	0.05

Using the above data for an example, we sum the columns to get:

$$\lambda(B\&C) = [0.8, 0.9, 0.7, 0.05]$$

This is the λ evidence resulting from the instantiation of D which is not seen elsewhere in the tree. Now we compute $P(R_2|S_2) = P(D = d1|B\&C)$, by dividing the potential function of node 2 by the λ message:

$$P(R_2|S_2) = \Psi(W_2)/\sum_{R_2} \Psi(W2)$$

This is the normalisation process which in the tabular format above means dividing each column by its sum. The result is:

	$b_1\&c_1$	$b_1\&c_2$	$b_2\&c_1$	$b_2\&c_2$
d_1	1	1	1	1
d_2	0	0	0	0

Note that the resulting 1s occur because there is just one R variable and it has been instantiated. If there were were more, then there would be more non-zero rows in the potential table, and the columns would express the conditional probability of the uninstantiated R variables given the parent (S) variable states.

At the root node, we now need to combine the evidence to find the posterior probability of the root node. We have that that:

$$\Psi(W_1) = P(A)P(B|A)P(C|A)$$

and we have just calculated $\lambda(B\&C)$, the evidence over the states of $B\&C$ following the instantiation of D which we use to update node 1 by multiplication:

$$\Psi'(W_1) = \Psi(W_1) \times \lambda(B\&C)$$

We now need to normalise all the evidence to give us the posterior probability of clique 1:

$$P'(W_1) = \Psi'(W_1)/\sum_{ABC} \Psi'(W_1)$$

And now the rest of the propagation is exactly as before:

$$P'(S_2) = P'(B\&C) = \sum_A P'(A\&B\&C) = \sum_A P'(W_1)$$

To calculate the joint probability of the variables of node 2 we use:

$$P'(W_2) = P'(R_2|S_2)P'(S_2)$$

and since, as we saw in this case there is no evidence for the parent variables

$$P'(R_2|S_2) = 1 \text{ for all states of the variables in } S_2$$

so

$$P'(W_2) = P'(S_2) = \sum_A P'(W_1)$$

we can now calculate

$$P'(S_3) = P'(C) = \sum_B P'(W_2)$$

(This could also be found using $\sum_{AB} P'(W_1)$)
To calculate the joint probability of the variables of node 2 we use:

$$P'(W_3) = P'(R_3|S_3) \times P'(S_3) = P(E|C) \times \sum_B P'(W_2)$$

Finally, we can compute the posterior probabilities of the original variables from the causal network by marginalisation. This can be done in several different ways, for example:

$$P'(C) = \sum_{AB} P'(W_1) = \sum_B P'(W_2) = \sum_E P'(W_3)$$

Limitations

Although this method gives us a general propagation algorithm, there is a limit to how far we can take it. It works well with sparsely connected graphs. However, the more arrows there are in the original causal networks, the larger the resulting cliques. In practice, large cliques will have large numbers of joint states, and the operations of marginalisation and multiplication begin to take prohibitively long computation times.