1 Variable Elimination in Bayesian Networks

Review: Inference

Given a joint probability distribution over variables a set of variables $X = X_1, X_2, \ldots, X_n$, we can make inferences of the form $(Y|Z)$, where $Y \subset X$ is the set of *query* variables, $Z \subset X$ are the *evidence* variables. The other variables H (those not mentioned in the query) are called *hidden* variables. To be clear, $X = Y \cup Z \cup H$.

The most naive and expensive way to do inference is to use the full joint probability distribution and sum out the hidden variables. By the product rule, $P(Y|Z)P(Z) = P(Y, Z)$. (Note that this is true for any distribution. This does not have anything to do with independence.) So to answer the query $P(Y|Z)$, we can compute $\frac{P(Y,Z)}{P(Z)}$. Note that $P(Y, Z) = \sum_H P(Y, Z, H)$. From the view of the full joint probability table, we are summing the probabilities for all of the entries in the table that match the values of the query variables and evidence variables (this includes entries for all of the combinations of values for the hidden variables) and dividing by the sum of all the entries that match the values of the evidence variables. (Note that the values in the first sum are a subset of the values in the latter sum.) This method is straightforward conceptually but has many problems. First, you have to compute and store the values in the full joint distribution. The number of entries is $\prod_i Options(X_i)$, where $Options(X_i)$ is the number of different values that variable X_i may have. So for n Boolean variables, the full joint has 2^n entries. For large networks, it is unreasonable to store/compute this many values. Furthermore, if a query requires summing out most of the variables, the time complexity is $O(2^n)$ as well.

Normalizing

Suppose we have variables A and B. We want to compute $P(A|b)$. We use b to denote $B = true$. The result of our query $P(A, b)$ is a two-element table that specifies $P(a, b)$ and $P(\neg a, b)$.Since $P(A|b)P(b) = P(A, b)$, we can compute $P(A|b) = \frac{P(A,b)}{P(b)}$. But we also know that $P(b) = P(a, b) + P(\neg a, b)$. Since we need both $P(a, b)$ and $P(\neg, b)$ to answer the query, this may actually be the best way to compute $P(b)$. The way this is expressed in the book is that $P(A|b) = \alpha P(A, b)$, where α is a normalizing constant that makes the values in the table $P(A|b)$ sum to 1. (In this case, α is $P(b)$, but the idea is that it doesn't matter what it is; we just know that we need to normalize to make sure that the values in $P(A|b)$ sum to 1.

Enumeration

Using the burglar alarm network, suppose we wish to compute the probability that there is an earthquake, given that both John and Mary call:

$$P(E|j, m) = \alpha P(E, j, m)$$

We need to sum over all the values of the hidden vars:

$$\alpha P(E, j, m) = \sum_a \sum_b P(E, j, m, b, a)$$

This may be a slight abuse of notation. The lower case j, m mean that $JohnCalls = true$ and $MaryCalls = true$. The lowercase b, a mean that we are summing over all values of the variables B, A. So now we have a sum over entries from the full joint. We can retrieve these using our Bayesian network structure:

$$\sum_b \sum_a P(E, j, m, b, a) = \sum_b \sum_a P(b)P(E)P(a|b, E)P(j|a)P(m|a)$$

In general, sums of this form could take $O(n2^n)$ time to compute. There may be n values to multiply together to compute each product term, and up to $O(2^n)$ total terms to sum up. We can save some computations by pushing the $\sum$'s inward as much as possible:

$$\sum_b \sum_a P(b)P(E)P(a|b, E)P(j|a)P(m|a) = P(E) \sum_b P(b) \sum_a P(a|b, E)P(j|a)P(m|a)$$

We need to do this sum both for $E = true$ and $E = false$. For the first case:

$$P(e) \left(\begin{array}{c} P(b) \left(P(a|b, e)P(j|a)P(m|a) + P(\neg a|b, e)P(j|\neg a)P(m|\neg a) \right) + \\ P(\neg b) \left(P(a|\neg b, e)P(j|a)P(m|a) + P(\neg a|\neg b, e)P(j\neg a)P(m|\neg a) \right) \end{array} \right)$$

Even here, we can see some wasted computations: the product $P(j|a)P(m|a)$ is computed twice, and so is $P(j|\neg a)P(m|\neg a)$. In this example, we do two extra multiplications, which is no big deal. But in a large network, this kind of waste can be debilitating.

Variable Elimination

(We follow Section 14.4 in Russell-Norvig with some additional details.) Here we introduce a variable elimination algorithm that will help us avoid the duplicate computations; it is a form of dynamic programming. We introduce tables called *factors* to help us do the bookkeeping. Initially, we will have one factor for each CPT term in our expression:

$$\alpha \underbrace{P(E)}_{E} \sum_b \underbrace{P(b)}_{B} \sum_a \underbrace{P(a|b, E)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}$$

E	$f_E(E)$
T	.002
F	.998

B	$f_B(B)$
T	.001
F	.999

A	B	E	$f_A(A, B, E)$
T	T	T	.95
T	T	F	.94
T	F	T	.29
T	F	F	.001
F	T	T	.05
F	T	F	.06
F	F	T	.71
F	F	F	.999

A	$f_J(A)$
T	.9
F	.05

A	$f_M(A)$
T	.7
F	.01

The other operation we need for factors is summing out. If, for example, we sum out A from $f_{AJM}(A, B, E)$, the resulting factor will be named $f_{\bar{A}JM}(B, E)$ (with the bar on the A in the subscript denoting that A was summed out). There is one entry in $f_{\bar{A}JM}(B, E)$ for each combination of values of B, E in $f_{AJM}(B, E)$. The value of the entry is the sum of all entries in $f_{AJM}(B, E)$ that have the same values of B, E (but differing values of A):

B	E	$f_{\bar{A}JM}(B, E)$
T	T	.95 * .63 + .05 * .0005 = .5985
T	F	.94 * .63 + .06 * .0005 = .5922
F	T	.29 * .63 + .71 * .0005 = .1830
F	F	.001 * .63 + .999 * .0005 = .001129

We are now ready to compute the answer to our query:

$\alpha P(E) \sum_b P(b) \sum_a P(a|b, e)P(j|a)P(m|a)$

$$
\begin{aligned}
&= \alpha f_E(E) \textstyle\sum_B f_B(B) \sum_a f_A(A, B, E) f_J(A) f_m(A) \\
&= \alpha f_E(E) \textstyle\sum_B f_B(B) \sum_a f_A(A, B, E) f_{JM}(A) \\
&= \alpha f_E(E) \textstyle\sum_B f_B(B) \sum_a f_{AJM}(A, B, E) \\
&= \alpha f_E(E) \textstyle\sum_B f_B(B) f_{\bar{A}JM}(B, E) \\
&= \alpha f_E(E) \textstyle\sum_B f_{B\bar{A}JM}(B, E) \\
&= \alpha f_E(E) f_{\bar{B}\bar{A}JM}(E) \\
&= \alpha f_{E\bar{B}\bar{A}JM}(E)
\end{aligned}
$$

We have already computed $f_{\bar{A}JM}$, so we can proceed from step 5, where we compute $f_{B\bar{A}JM}(B, E)$:

B	E	$f_{B\bar{A}JM}(B, E)$
T	T	.5985 * .001
T	F	.5922 * .001
F	T	.1830 * .999
F	F	.001129 * .999

=

B	$f_B(B)$
T	.001
F	.999

B	E	$f_{\bar{A}JM}(B, E)$
T	T	.5985
T	F	.5922
F	T	.1830
F	F	.001129

Next, we sum out B to produce $f_{\bar{B}\bar{A}JM}(E)$.

E	$f_{\bar{B}\bar{A}JM}(E)$
T	.0005985 + .1828= .1834
F	.0005922 * .001128= .001720

Now we compute $f_{E\bar{B}\bar{A}JM}(E)$:

E	$f_{\bar{B}\bar{A}JM}(E)$
T	.1834 * .002 = .0003699
F	.001720 * .998 = .001717

=

E	$f_E(E)$
T	.002
F	.998

E	$f_{\bar{B}\bar{A}JM}(E)$
T	.1834
F	.001720

The initial tables store the values from the CPTs that we will need over the course of the query evaluation. The factor for E, denoted $f_E(E)$, is a two-element vector because we will be computing both $P(e, j, m)$ and $P(\neg e, j, m)$ (we will normalize at the end to compute $P(e|j, m)$ and $P(\neg e|j, m)$). The subscript E names the factor, and the (E) means that the value of E varies in the factor. The factor for B is also a two-element vector $f_B(B)$, because we will need to sum out the variable B. The factor $f_A(A, B, E)$ requires $2 \times 2 \times 2$ elements because we will need all combinations of values for A, B, and E in the expression. And even though J and M both have parents in the network, the factors $f_J(A)$ and $f_M(A)$ have only two elements because throughout the expression, we are only interested in cases where $JohnCalls = true$ and $MaryCalls = true$.

There are two operations that we need to be able to perform on factors: multiplication and summing out. The multiplication operation is similar to a natural join on database tables, as opposed to a matrix multiplication or an element-by-element multiplication. The set of variables in the product of two factors is the union of the sets of variables in the operands. In the product factor, there will be one entry for every combination of values for variables in the resulting set. The value of each entry is the product of the entries in the original tables that match the values for all relevant variables. For example, consider the product $f_{JM}(A) = f_J(A)f_M(A)$:

A	$f_{JM}(A)$
T	.9 * .7
F	.05 * .01

=

A	$f_J(A)$
T	.9
F	.05

A	$f_M(A)$
T	.7
F	.01

The union of variables in the operands is A. The entry for T in the resulting table is the product of the entries where $A = True$ in the operands. Now consider $f_{AJM}(A, B, E) = f_A(A, B, E)f_{JM}(A)$.

A	B	E	$f_{AJM}(A, B, E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

=

A	B	E	$f_{AJM}(A, B, E)$
T	T	T	.95
T	T	F	.94
T	F	T	.29
T	F	F	.001
F	T	T	.05
F	T	F	.06
F	F	T	.71
F	F	F	.999

A	$f_{JM}(A)$
T	.63
F	.0005

The entry for $A = T, B = T, E = T$ in $f_{AJM}(A, B, E)$ is the product of corresponding entries from $f_A(A, B, E)$ where $A = T, B = T, E = T$ and from $f_JM(A)$ where $A = T$ (since B, E do not appear in $f_JM(A)$. In general, we have

$$f(X_1, \ldots, X_j, Y_1, \ldots, Y_k, Z_1 \ldots, Z_l) = f_1(X_1, \ldots, X_j, Y_1, \ldots, Y_k)f_2(Y_1, \ldots, Y_k, Z_1, \ldots, Z_l)$$

The factors f_1 and f_2 share common variables Y. If f_1 has a total of $j + k$ variables and f_2 has $k + l$ variables, then the product will have $j + k + l$ variables. Each entry in the result will have all the variables $X \cup Y \cup Z$ and the entry for a particular valuation of X, Y, Z will be the product of the entry in f_1 that has the same values for X, Y and the entry in f_2 that has the same values for Y, Z.

So we have $\alpha f_{E\bar{B}\bar{A}JM}(E) = \alpha\langle .0003699, .001717\rangle = \langle .1772, .8228\rangle$, or in other words, $P(e|j,m) = .1772$.

Another Example

Using the Bayesian Network from Homework 2 Problem 3, we will compute $P(E|d)$:

$$\begin{aligned}
P(E|d) &= \alpha P(E,d) \\
&= \alpha \textstyle\sum_{a,b,c} P(a,b,c,E,d) \\
&= \alpha \textstyle\sum_a P(a) \sum_c P(c|a) \sum_b P(b)P(d|b)P(E|b,c) \\
&= \alpha \textstyle\sum_a f_A(A) \sum_c f_C(A,C) \sum_b f_B(B) f_D(B) f_E(B,C,E) \\
&= \alpha \textstyle\sum_a f_A(A) \sum_c f_C(A,C) \sum_b f_{BDE}(B,C,E) \\
&= \alpha \textstyle\sum_a f_A(A) \sum_c f_C(A,C) f_{\bar{B}DE}(C,E) \\
&= \alpha \textstyle\sum_a f_A(A) \sum_c f_{C\bar{B}DE}(A,C,E) \\
&= \alpha \textstyle\sum_a f_A(A) f_{\bar{C}\bar{B}DE}(A,E) \\
&= \alpha \textstyle\sum_a f_{A\bar{C}\bar{B}DE}(A,E) \\
&= \alpha f_{\bar{A}\bar{C}\bar{B}DE}(E)
\end{aligned}$$

The only variable in the expression that is fixed is D. So the initial factors are:

A	$f_A(A)$
T	.3
F	.7

A	C	$f_C(A,C)$
T	T	.2
T	F	.8
F	T	.4
F	F	.6

B	$f_B(B)$
T	.8
F	.2

B	$f_D(B)$
T	.1
F	.6

B	C	E	$f_E(B,C,E)$
T	T	T	.5
T	T	F	.5
T	F	T	.9
T	F	F	.1
F	T	T	.4
F	T	F	.6
F	F	T	.7
F	F	F	.3

We can compute $f_{BDE}(B,C,E)$ from the rightmost three factors in one step:

B	C	E	$f_{BDE}(B,C,E)$
T	T	T	.8 * .1 * .5 = .04
T	T	F	.8 * .1 * .5 = .04
T	F	T	.8 * .1 * .9 = .072
T	F	F	.8 * .1 * .1 = .008
F	T	T	.2 * .6 * .4 = .048
F	T	F	.2 * .6 * .6 = .072
F	F	T	.2 * .6 * .7 = .084
F	F	F	.2 * .6 * .3 = .036

=

B	$f_B(B)$
T	.8
F	.2

B	$f_D(B)$
T	.1
F	.6

B	C	E	$f_E(B,C,E)$
T	T	T	.5
T	T	F	.5
T	F	T	.9
T	F	F	.1
F	T	T	.4
F	T	F	.6
F	F	T	.7
F	F	F	.3

Now we sum out B:

C	E	$f_{\bar{B}DE}(C,E)$
T	T	.04 + .048 = .088
T	F	.04 + .072 = .112
F	T	.072 + .084 = .156
F	F	.008 + .036 = .044

Next, we compute $f_{C\bar{B}DE}(A,C,E)$:

A	C	E	$f_{C\bar{B}DE}(A,C,E)$
T	T	T	.2 * .088 = .0176
T	T	F	.2 * .112 = .0224
T	F	T	.8 * .156 = .1248
T	F	F	.8 * .044 = .0352
F	T	T	.4 * .088 = .0352
F	T	F	.4 * .112 = .0448
F	F	T	.6 * .156 = .0936
F	F	F	.6 * .044 = .0264

=

A	C	$f_C(A,C)$
T	T	.2
T	F	.8
F	T	.4
F	F	.6

C	E	$f_{\bar{B}DE}(C,E)$
T	T	.088
T	F	.112
F	T	.156
F	F	.044

Now we sum out C:

A	E	$f_{\bar{C}\bar{B}DE}(A,E)$
T	T	.0176 + .1248 = .1424
T	F	.0224 + .0352 = .0576
F	T	.0352 + .0936 = .1288
F	F	.0448 + .0264 = .0712

Next, we compute $f_{A\bar{C}\bar{B}DE}(A,E)$:

A	E	$f_{A\bar{C}\bar{B}DE}(A,E)$
T	T	.3 * .1424 = .04272
T	F	.3 * .0576 = .01728
F	T	.7 * .1288 = .09016
F	F	.7 * .0712 = .04984

=

A	$f_A(A)$
T	.3
F	.7

A	E	$f_{\bar{C}\bar{B}DE}(A,E)$
T	T	.1424
T	F	.0576
F	T	.1288
F	F	.0712

Then we sum out A

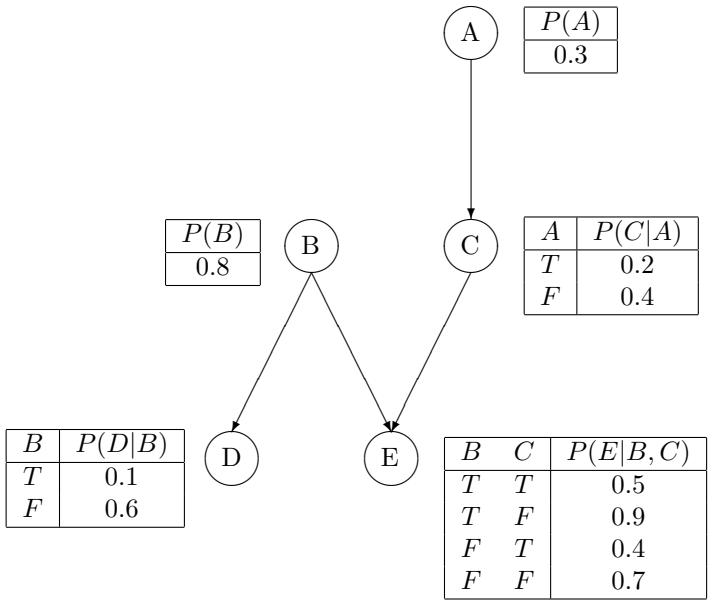
E	$f_{\bar{A}\bar{C}\bar{B}DE}(A,E)$
T	.04272 + .09016 = .13288
F	.01728 + .04984 = .06712

Finally, $\alpha f_{\bar{A}\bar{C}\bar{B}DE}(E) = \alpha\langle .13288, .06712\rangle = \langle .6644, .3356\rangle$. So $P(\neg e|d) = .3356$.

In these small-network examples, it may seem like the variable elimination algorithm actually requires *more* work than enumeration. But in some large networks, the computational savings can be large. The complexity of variable elimination depends on the size of the largest factor that must

be created during the computation. This in turn depends on the order in which the variables are eliminated. In polytree networks with a consistent ordering of the variables (i.e., a parent comes before any of its children), the time and space complexity is linear in the size of the network.

3. Given the following Bayesian network of Boolean variables:



$P(A)$
0.3

$P(B)$
0.8

A	$P(C\|A)$
T	0.2
F	0.4

B	$P(D\|B)$
T	0.1
F	0.6

B	C	$P(E\|B,C)$
T	T	0.5
T	F	0.9
F	T	0.4
F	F	0.7