

6.825 Techniques in Artificial Intelligence

Inference in Bayesian Networks

Lecture 16 • 1

Now that we know what the semantics of Bayes nets are; what it means when we have one, we need to understand how to use it. Typically, we'll be in a situation in which we have some evidence, that is, some of the variables are instantiated, and we want to infer something about the probability distribution of some other variables.

6.825 Techniques in Artificial Intelligence

Inference in Bayesian Networks

- Exact inference

Lecture 16 • 2

In exact inference, we analytically compute the conditional probability distribution over the variables of interest.

6.825 Techniques in Artificial Intelligence

Inference in Bayesian Networks

- Exact inference
- Approximate inference

Lecture 16 • 3

But sometimes, that's too hard to do, in which case we can use approximation techniques based on statistical sampling.

Query Types

Given a Bayesian network, what questions might we want to ask?

Given a Bayesian network, what kinds of questions might we want to ask?

Query Types

Given a Bayesian network, what questions might we want to ask?

- Conditional probability query: $P(x \mid e)$

The most usual is a conditional probability query. Given instantiations for some of the variables (we'll use e here to stand for the values of all the instantiated variables; it doesn't have to be just one), what is the probability that node X has a particular value x ?

Query Types

Given a Bayesian network, what questions might we want to ask?

- Conditional probability query: $P(x | e)$
- Maximum a posteriori probability:
What value of x maximizes $P(x|e)$?

Lecture 16 • 6

Another interesting question you might ask is, what is the most likely explanation for some evidence? We can think of that as the value of node X (or of some group of nodes) that maximizes the probability that you would have seen the evidence you did. This is called the maximum a posteriori probability or MAP query.

Query Types

Given a Bayesian network, what questions might we want to ask?

- Conditional probability query: $P(x | e)$
- Maximum a posteriori probability:
What value of x maximizes $P(x|e)$?

General question: What's the whole probability distribution over variable X given evidence e , $P(X | e)$?

Lecture 16 • 7

In our discrete probability situation, the only way to answer a MAP query is to compute the probability of x given e for all possible values of x and see which one is greatest.

Query Types

Given a Bayesian network, what questions might we want to ask?

- Conditional probability query: $P(x | e)$
- Maximum a posteriori probability:
What value of x maximizes $P(x|e)$?

General question: What's the whole probability distribution over variable X given evidence e , $P(X | e)$?

Lecture 16 • 8

So, in general, we'd like to be able to compute a whole probability distribution over some variable or variables X , given instantiations of a set of variables e .

Using the joint distribution

To answer any query involving a conjunction of variables, sum over the variables not involved in the query.

Given the joint distribution over the variables, we can easily answer any question about the value of a single variable by summing (or marginalizing) over the other variables.

Using the joint distribution

To answer any query involving a conjunction of variables, sum over the variables not involved in the query.

$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a,b,c,d) \\ &= \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} \sum_{c \in \text{dom}(C)} \Pr(A = a \wedge B = b \wedge C = c)\end{aligned}$$

So, in a domain with four variables, A, B, C, and D, the probability that variable D has value d is the sum over all possible combinations of values of the other three variables of the joint probability of all four values. This is exactly the same as the procedure we went through in the last lecture, where to compute the probability of cavity, we added up the probability of cavity and toothache and the probability of cavity and not toothache.

Using the joint distribution

To answer any query involving a conjunction of variables, sum over the variables not involved in the query.

$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a,b,c,d) \\ &= \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} \sum_{c \in \text{dom}(C)} \Pr(A = a \wedge B = b \wedge C = c)\end{aligned}$$

In general, we'll use the first notation, with a single summation indexed by a list of variable names, and a joint probability expression that mentions values of those variables. But here we can see the completely written-out definition, just so we all know what the shorthand is supposed to mean.

Using the joint distribution

To answer any query involving a conjunction of variables, sum over the variables not involved in the query.

$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a,b,c,d) \\ &= \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} \sum_{c \in \text{dom}(C)} \Pr(A = a \wedge B = b \wedge C = c)\end{aligned}$$

$$\Pr(d | b) = \frac{\Pr(b,d)}{\Pr(b)} = \frac{\sum_{AC} \Pr(a,b,c,d)}{\sum_{ACD} \Pr(a,b,c,d)}$$

To compute a conditional probability, we reduce it to a ratio of conjunctive queries using the definition of conditional probability, and then answer each of those queries by marginalizing out the variables not mentioned.

Using the joint distribution

To answer any query involving a conjunction of variables, sum over the variables not involved in the query.

$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a,b,c,d) \\ &= \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} \sum_{c \in \text{dom}(C)} \Pr(A = a \wedge B = b \wedge C = c)\end{aligned}$$

$$\Pr(d | b) = \frac{\Pr(b,d)}{\Pr(b)} = \frac{\sum_{AC} \Pr(a,b,c,d)}{\sum_{ACD} \Pr(a,b,c,d)}$$

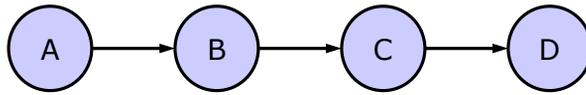
In the numerator, here, you can see that we're only summing over variables A and C, because b and d are instantiated in the query.

Simple Case

Lecture 16 • 14

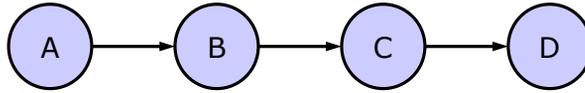
We're going to learn a general purpose algorithm for answering these joint queries fairly efficiently. We'll start by looking at a very simple case to build up our intuitions, then we'll write down the algorithm, then we'll apply it to a more complex case.

Simple Case



Okay. Here's our very simple case. It's a bayes net with four nodes, arranged in a chain.

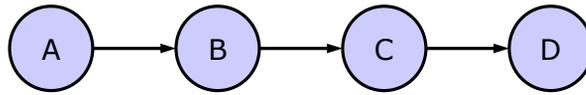
Simple Case



$$\Pr(d) = \sum_{ABC} \Pr(a, b, c, d)$$

So, we know from before that the probability that variable D has some value little d is the sum over A, B, and C of the joint distribution, with d fixed.

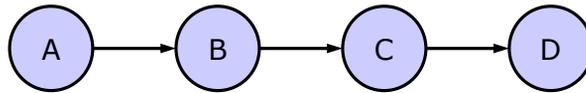
Simple Case



$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a, b, c, d) \\ &= \sum_{ABC} \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a)\end{aligned}$$

Now, using the chain rule of Bayesian networks, we can write down the joint probability as a product over the nodes of the probability of each node's value given the values of its parents. So, in this case, we get $P(d|c)$ times $P(c|b)$ times $P(b|a)$ times $P(a)$.

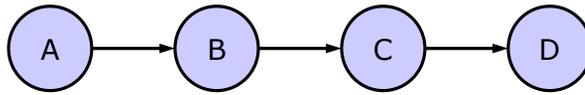
Simple Case



$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a, b, c, d) \\ &= \sum_{ABC} \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a)\end{aligned}$$

This expression gives us a method for answering the query, given the conditional probabilities that are stored in the net. And this method can be applied directly to any other bayes net. But there's a problem with it: it requires enumerating all possible combinations of assignments to A, B, and C, and then, for each one, multiplying the factors for each node. That's an enormous amount of work and we'd like to avoid it if at all possible.

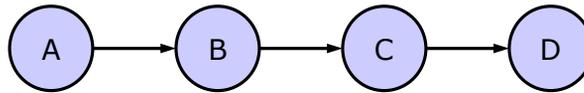
Simple Case



$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a, b, c, d) \\ &= \sum_{ABC} \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a) \\ &= \sum_C \sum_B \sum_A \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a)\end{aligned}$$

So, we'll try rewriting the expression into something that might be more efficient to evaluate. First, we can make our summation into three separate summations, one over each variable.

Simple Case

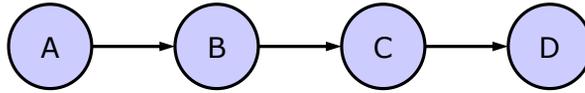


$$\begin{aligned}\Pr(d) &= \sum_{ABC} \Pr(a,b,c,d) \\ &= \sum_{ABC} \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a) \\ &= \sum_C \sum_B \sum_A \Pr(d | c) \Pr(c | b) \Pr(b | a) \Pr(a) \\ &= \sum_C \Pr(d | c) \sum_B \Pr(c | b) \sum_A \Pr(b | a) \Pr(a)\end{aligned}$$

Lecture 16 • 20

Then, by distributivity of addition over multiplication, we can push the summations in, so that the sum over A includes all the terms that mention A, but no others, and so on. It's pretty clear that this expression is the same as the previous one in value, but it can be evaluated more efficiently. We're still, eventually, enumerating all assignments to the three variables, but we're doing somewhat fewer multiplications than before. So this is still not completely satisfactory.

Simple Case

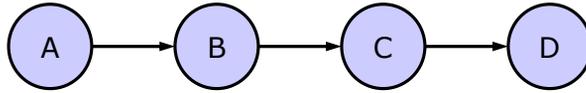


$$\Pr(d) = \sum_C \Pr(d | c) \sum_B \Pr(c | b) \underbrace{\sum_A \Pr(b | a) \Pr(a)}$$

$$\begin{bmatrix} \Pr(b_1 | a_1) \Pr(a_1) & \Pr(b_1 | a_2) \Pr(a_2) \\ \Pr(b_2 | a_1) \Pr(a_1) & \Pr(b_2 | a_2) \Pr(a_2) \end{bmatrix}$$

If you look, for a minute, at the terms inside the summation over A, you'll see that we're doing these multiplications over for each value of C, which isn't necessary, because they're independent of C. Our idea, here, is to do the multiplications once and store them for later use. So, first, for each value of A and B, we can compute the product, generating a two dimensional matrix.

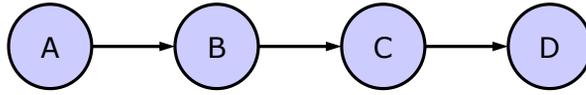
Simple Case



$$\Pr(d) = \sum_C \Pr(d | c) \sum_B \Pr(c | b) \underbrace{\sum_A \Pr(b | a) \Pr(a)}_{\left[\begin{array}{l} \sum \Pr(b_1 | a) \Pr(a) \\ \sum_A \Pr(b_2 | a) \Pr(a) \end{array} \right]}$$

Then, we can sum over the rows of the matrix, yielding one value of the sum for each possible value of b .

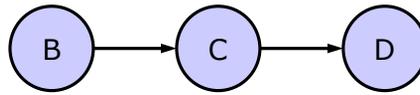
Simple Case



$$\Pr(d) = \sum_C \Pr(d | c) \sum_B \Pr(c | b) \underbrace{\sum_A \Pr(b | a) \Pr(a)}_{f_1(b)}$$

We'll call this set of values, which depends on b , f_1 of b .

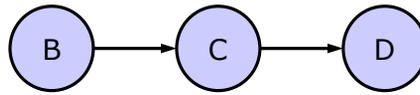
Simple Case



$$\Pr(d) = \sum_C \Pr(d | c) \underbrace{\sum_B \Pr(c | b) f_1(b)}_{f_2(c)}$$

Now, we can substitute f_1 of b in for the sum over A in our previous expression. And, effectively, we can remove node A from our diagram. Now, we express the contribution of b , which takes the contribution of a into account, as f_1 of b .

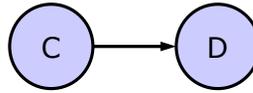
Simple Case



$$\Pr(d) = \sum_C \Pr(d | c) \underbrace{\sum_B \Pr(c | b) f_1(b)}_{f_2(c)}$$

We can continue the process in basically the same way. We can look at the summation over b and see that the only other variable it involves is c . We can summarize those products as a set of factors, one for each value of c . We'll call those factors f_2 of c .

Simple Case



$$\Pr(d) = \sum_c \Pr(d | c) f_2(c)$$

We substitute f_2 of c into the formula, remove node b from the diagram, and now we're down to a simple expression in which d is known and we have to sum over values of c .

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables

Lecture 16 • 27

That was a simple special case. Now we can look at the algorithm in the general case. Let's assume that we're given a Bayesian network and an ordering on the variables that aren't fixed in the query. We'll come back later to the question of the influence of the order, and how we might find a good one.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

We can express the probability of the query variables as a sum over each value of each of the non-query variables of a product over each node in the network, of the probability that that variable has the given value given the values of its parents.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

For $i = m$ downto 1

So, we'll eliminate the variables from the inside out. Starting with variable X_m and finishing with variable X_1 .

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

For $i = m$ downto 1

- remove all the factors that mention X_i

To eliminate variable X_i , we start by gathering up all of the factors that mention X_i , and removing them from our set of factors. Let's say there are k such factors.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j | Pa(x_j))$$

For $i = m$ downto 1

- remove all the factors that mention X_i
- multiply those factors, getting a value for each combination of mentioned variables

Now, we make a $k+1$ dimensional table, indexed by X_i as well as each of the other variables that is mentioned in our set of factors.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

For $i = m$ downto 1

- remove all the factors that mention X_i
- multiply those factors, getting a value for each combination of mentioned variables
- sum over X_i

We then sum the table over the X_i dimension, resulting in a k -dimensional table.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

For $i = m$ downto 1

- remove all the factors that mention X_i
- multiply those factors, getting a value for each combination of mentioned variables
- sum over X_i
- put this new factor into the factor set

This table is our new factor, and we put a term for it back into our set of factors.

Variable Elimination Algorithm

Given a Bayesian network, and an *elimination order* for the non-query variables, compute

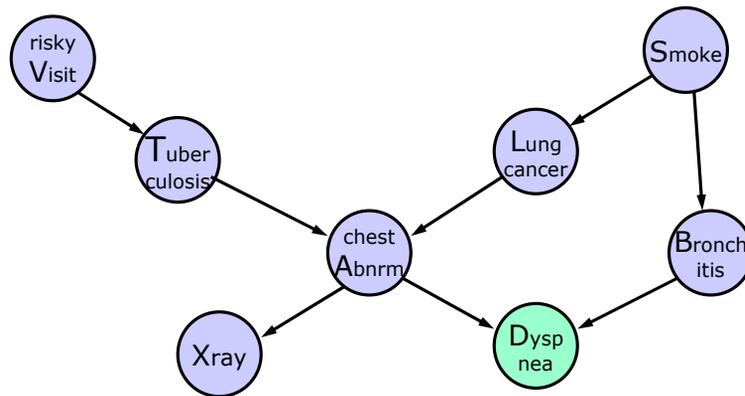
$$\sum_{X_1} \sum_{X_2} \dots \sum_{X_m} \prod_j \Pr(x_j \mid Pa(x_j))$$

For $i = m$ downto 1

- remove all the factors that mention X_i
- multiply those factors, getting a value for each combination of mentioned variables
- sum over X_i
- put this new factor into the factor set

Once we've eliminated all the summations, we have the desired value.

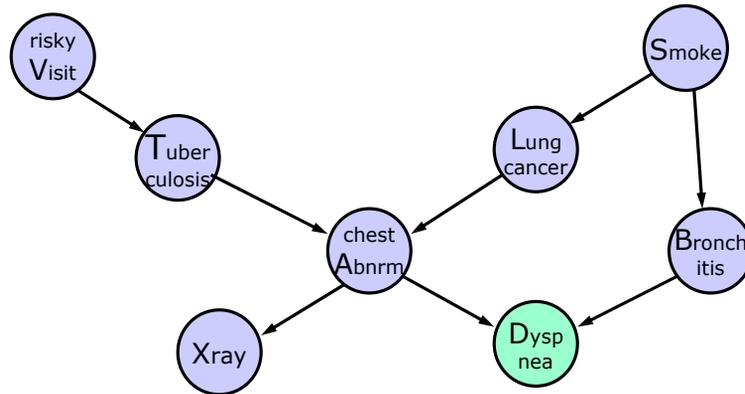
One more example



Lecture 16 • 35

Here's a more complicated example, to illustrate the variable elimination algorithm in a more general case. We have this big network that encodes a domain for diagnosing lung disease. (Dyspnea, as I understand it, is shortness of breath).

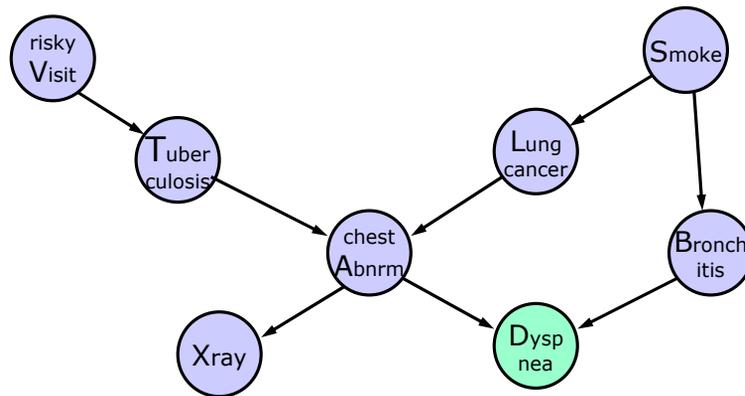
One more example



$$\Pr(d) = \sum_{A,B,L,T,S,X,V} \Pr(d | a,b) \Pr(a | t,l) \Pr(b | s) \Pr(l | s) \Pr(s) \Pr(x | a) \Pr(t | v) \Pr(v)$$

We'll do variable elimination on this graph using elimination order A, B, L, T, S, X, V.

One more example

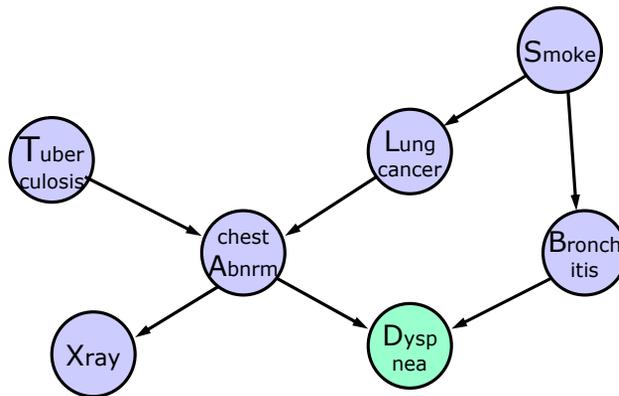


$$\Pr(d) = \sum_{A,B,L,T,S,X} \Pr(d | a,b) \Pr(a | t,l) \Pr(b | s) \Pr(l | s) \Pr(s) \Pr(x | a) \underbrace{\sum_V \Pr(t | v) \Pr(v)}_{f_1(t)}$$

Lecture 16 • 37

So, we start by eliminating V . We gather the two terms that mention V and see that they also involve variable T . So, we compute the product for each value of T , and summarize those in the factor f_1 of T .

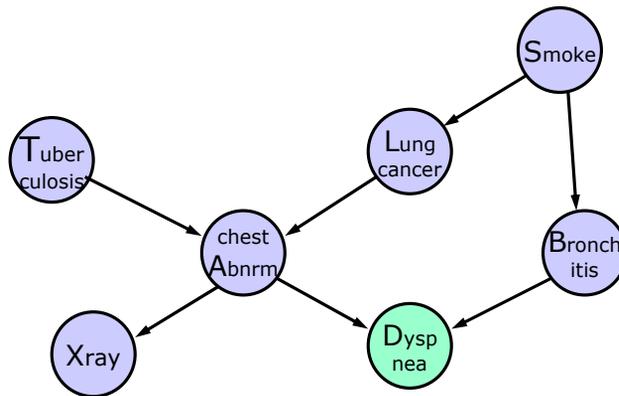
One more example



$$\Pr(d) = \sum_{A,B,L,T,S,X} \Pr(d | a,b) \Pr(a | t,l) \Pr(b | s) \Pr(l | s) \Pr(s) \Pr(x | a) f_1(t)$$

Now we can substitute that factor in for the summation, and remove the node from the network.

One more example

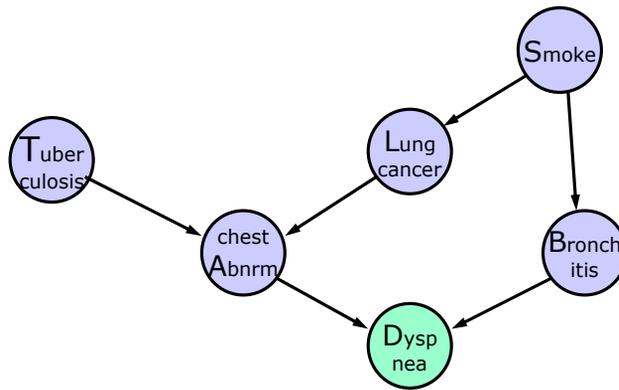


$$\Pr(d) = \sum_{A,B,L,T,S} \underbrace{\sum_X \Pr(x|a)}_1 \Pr(d|a,b) \Pr(a|t,l) \Pr(b|s) \Pr(l|s) \Pr(s) f_i(t)$$

Lecture 16 • 39

The next variable to be eliminated is X . There is actually only one term involving X , and it also involves variable A . So, for each value of A , we compute the sum over X of $P(x|a)$. But wait! We know what this value is! If we fix a and sum over x , these probabilities have to add up to 1.

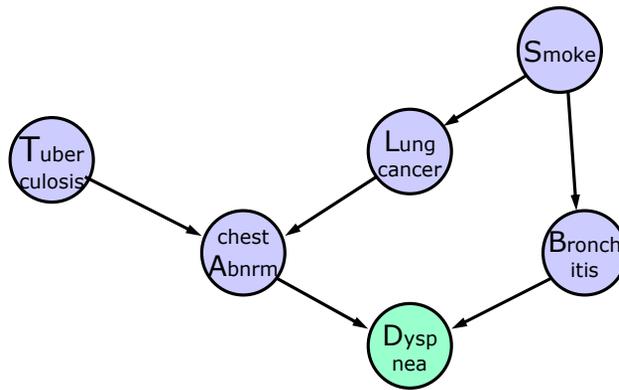
One more example



$$\Pr(d) = \sum_{A,B,L,T,S} \Pr(d | a,b) \Pr(a | t,l) \Pr(b | s) \Pr(l | s) \Pr(s) f_1(t)$$

So, rather than adding another factor to our expression, we can just remove the whole sum. In general, the only nodes that will have an influence on the probability of D are its ancestors.

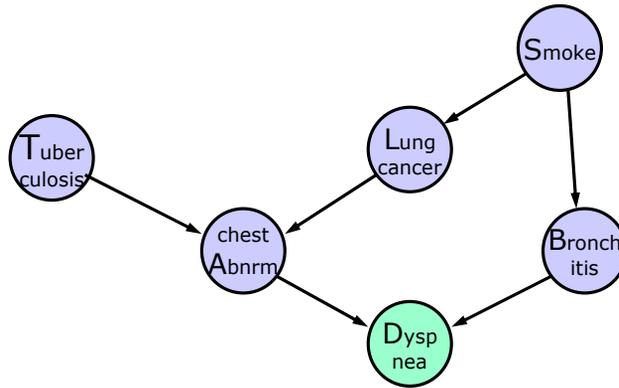
One more example



$$\Pr(d) = \sum_{A,B,L,T} \Pr(d | a,b) \Pr(a | t,l) f_i(t) \underbrace{\sum_s \Pr(b | s) \Pr(l | s) \Pr(s)}_S$$

Now, it's time to eliminate S . We find that there are three terms involving S , and we gather them into the sum. These three terms involve two other variables, B and L . So we have to make a factor that specifies, for each value of B and L , the value of the sum of products.

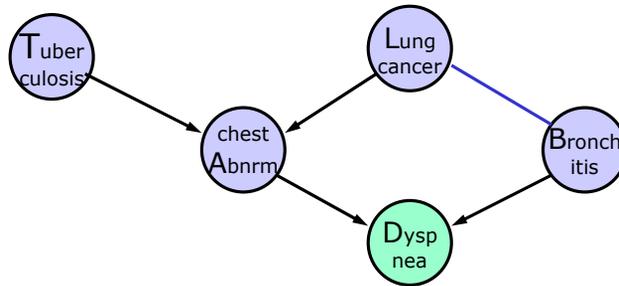
One more example



$$\Pr(d) = \sum_{A,B,L,T} \Pr(d | a,b) \Pr(a | t,l) f_1(t) \underbrace{\sum_s \Pr(b | s) \Pr(l | s) \Pr(s)}_{f_2(b,l)}$$

We'll call that factor f_2 of b and l .

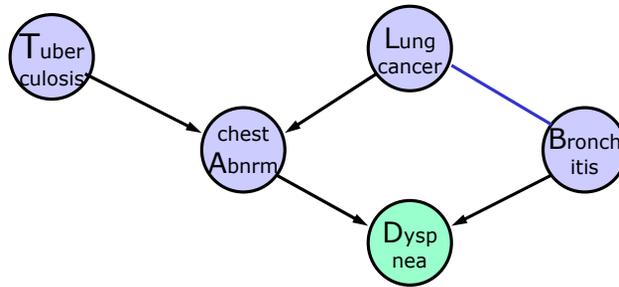
One more example



$$\Pr(d) = \sum_{A,B,L,T} \Pr(d | a,b) \Pr(a | t,l) f_1(t) f_2(b,l)$$

Now we can substitute that factor back into our expression. We can also eliminate node S. But in eliminating S, we've added a direct dependency between L and B (they used to be dependent via S, but now the dependency is encoded explicitly in $f_2(b)$). We'll show that in the graph by drawing a line between the two nodes. It's not exactly a standard directed conditional dependence, but it's still useful to show that they're coupled.

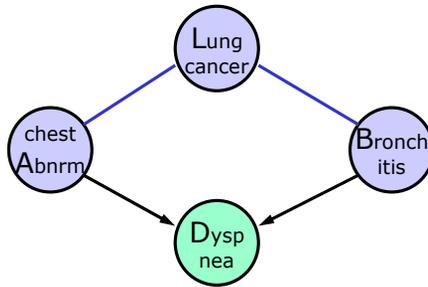
One more example



$$\Pr(d) = \sum_{A,B,L} \Pr(d | a,b) f_2(b,l) \underbrace{\sum_T \Pr(a | t,l) f_1(t)}_{f_3(a,l)}$$

Now we eliminate T. It involves two terms, which themselves involve variables A and L. So we make a new factor f_3 of A and L.

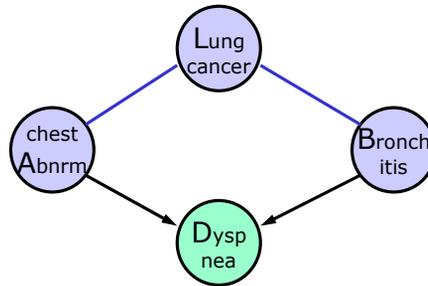
One more example



$$\Pr(d) = \sum_{A,B,L} \Pr(d | a,b) f_2(b,l) f_3(a,l)$$

We can substitute in that factor and eliminate T. We're getting close!

One more example

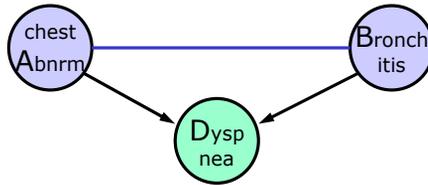


$$\Pr(d) = \sum_{A,B} \Pr(d | a,b) \underbrace{\sum_L f_2(b,l) f_3(a,l)}_{f_4(a,b)}$$

Lecture 16 • 46

Next we eliminate L. It involves these two factors, which depend on variables A and B. So we make a new factor, f_4 of A and B, and substitute it in. We remove node L, but couple A and B.

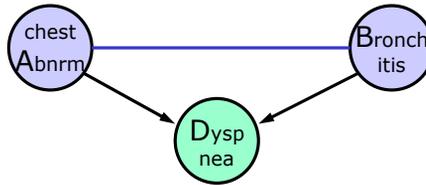
One more example



$$\Pr(d) = \sum_{A,B} \Pr(d | a,b) f_A(a,b)$$

At this point, we could just do the summations over A and B and be done. But to finish out the algorithm the way a computer would, it's time to eliminate variable B.

One more example



$$\Pr(d) = \sum_A \sum_B \underbrace{\Pr(d | a, b) f_4(a, b)}_{f_5(a)}$$

Lecture 16 • 48

It involves both of our remaining terms, and it seems to depend on variables A and D. However, in this case, we're interested in the probability of a particular value, little d of D, and so the variable d is instantiated. Thus, we can treat it as a constant in this expression, and we only need to generate a factor over a, which we'll call f5 of a. And we can now, in some sense, remove D from our network as well (because we've already factored it into our answer).

One more example



$$\Pr(d) = \sum_A f_5(a)$$

Finally, to get the probability that variable D has value little d, we simply sum factor f5 over all values of a. Yay! We did it.

Properties of Variable Elimination

Lecture 16 • 50

Let's see how the variable elimination algorithm performs, both in theory and in practice.

Properties of Variable Elimination

- Time is exponential in size of largest factor

First of all, it's pretty easy to see that it runs in time exponential in the number of variables involved in the largest factor. Creating a factor with k variables involves making a $k+1$ dimensional table. If you have b values per variable, that's a table of size b^{k+1} . To make each entry, you have to multiply at most n numbers, where n is the number of nodes. We have to do this for each variable to be eliminated (which is usually close to n). So we have something like $\text{time} = O(n^2 b^k)$.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors

How big the factors are depends on the elimination order. You'll see in one of the recitation exercises just how dramatic the difference in factor sizes can be. A bad elimination order can generate huge factors.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors
- NP Hard to find the best elimination order
- Even the best elimination order may generate large factors

So, we'd like to use the elimination order that generates the smallest factors.
Unfortunately, it turns out to be NP hard to find the best elimination order.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors
- NP Hard to find the best elimination order
- Even the best elimination order may generate large factors
- There are reasonable heuristics for picking an elimination order (such as choosing the variable that results in the smallest next factor)

Lecture 16 • 54

At least, there are some fairly reasonable heuristics for choosing an elimination order. It's usually done dynamically. So, rather than fixing the elimination order in advance, as we suggested in the algorithm description, you can pick the next variable to be eliminated depending on the situation. In particular, one reasonable heuristic is to pick the variable to eliminate next that will result in the smallest factor. This greedy approach won't always be optimal, but it's not usually too bad.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors
- NP Hard to find the best elimination order
- Even the best elimination order may generate large factors
- There are reasonable heuristics for picking an elimination order (such as choosing the variable that results in the smallest next factor)

- Inference in polytrees (nets with no cycles) is linear in size of the network (the largest CPT)

There is one case where Bayes net inference in general, and the variable elimination algorithm in particular is fairly efficient, and that's when the network is a polytree. A polytree is a network with no cycles. That is, a network in which, for any two nodes, there is only one path between them. In a polytree, inference is linear in the size of the network, where the size of the network is defined to be the size of the largest conditional probability table (or exponential in the maximum number of parents of any node). In a polytree, the optimal elimination order is to start at the root nodes, and work downwards, always eliminating a variable that no longer has any parents. In doing so, we never introduce additional connections into the network.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors
- NP Hard to find the best elimination order
- Even the best elimination order may generate large factors
- There are reasonable heuristics for picking an elimination order (such as choosing the variable that results in the smallest next factor)

- Inference in polytrees (nets with no cycles) is linear in size of the network (the largest CPT)
- Many problems with very large nets have only small factors, and thus efficient inference

Lecture 16 • 56

So, inference in polytrees is efficient, and even in many large non-polytree networks, it's possible to keep the factors small, and therefore to do inference relatively efficiently.

Properties of Variable Elimination

- Time is exponential in size of largest factor
- Bad elimination order can generate huge factors
- NP Hard to find the best elimination order
- Even the best elimination order may generate large factors
- There are reasonable heuristics for picking an elimination order (such as choosing the variable that results in the smallest next factor)

- Inference in polytrees (nets with no cycles) is linear in size of the network (the largest CPT)
- Many problems with very large nets have only small factors, and thus efficient inference

Lecture 16 • 57

When the network is such that the factors are, of necessity, large, we'll have to turn to a different class of methods.

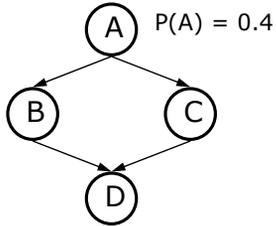
Sampling

Lecture 16 • 58

Another strategy, which is a theme that comes up also more and more in AI actually, is to say, well, we didn't really want the right answer anyway. Let's try to do an approximation. And you can also show that it's computationally hard to get an approximation that's within epsilon of the answer that you want, but again that doesn't keep us from trying.

Sampling

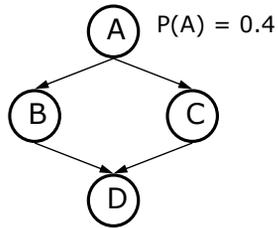
To get approximate answer we can do **stochastic simulation** (**sampling**).



So, the other thing that we can do is the stochastic simulation or sampling. In sampling, what we do is we look at the root nodes of our graph, and attached to this root node is some probability that A is going to be true, right? Maybe it's .4. So we flip a coin that comes up heads with probability .4 and see if we get true or false.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



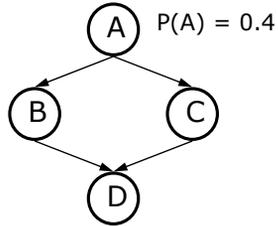
A	B	C	D
T	F	T	T
...			

- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A

We flip our coin, let's say, and we get true for A -- this time. And now, given the assignment of true to A, we look in the conditional probability table for B given A = true, and that gives us a probability for B.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



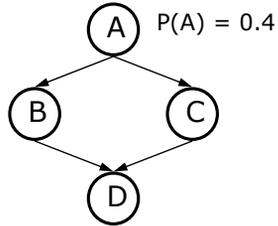
A	B	C	D
T	F	T	T
...			

- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A
- Given $A=T$, lookup $P(B|A=T)$ and flip a coin with that prob., assume we get F

Now, we flip a coin with that probability. Say we get False. We enter that into the table.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



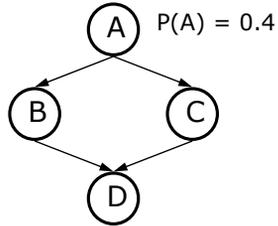
A	B	C	D
T	F	T	T
...			

- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A
- Given $A=T$, lookup $P(B|A=T)$ and flip a coin with that prob., assume we get F
- Similarly for C and D

We do the same thing for C, and let's say we get True.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



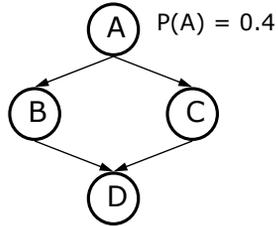
A	B	C	D
T	F	T	T
...			

- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A
- Given $A=T$, lookup $P(B|A=T)$ and flip a coin with that prob., assume we get F
- Similarly for C and D

Now, we look in the CPT for D given B and C, for the case where B is false and C is true, and we flip a coin with that probability, in order to get a value for D.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



A	B	C	D
T	F	T	T
...			

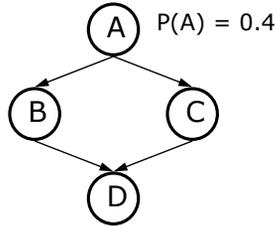
- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A
- Given $A=T$, lookup $P(B|A=T)$ and flip a coin with that prob., assume we get F
- Similarly for C and D
- We get one sample from joint distribution of these four vars

Lecture 16 • 64

So, there's one sample from the joint distribution of these four variables. And you can just keep doing this, all day and all night, and generate a big pile of samples, using that algorithm. And now you can ask various questions.

Sampling

To get approximate answer we can do **stochastic simulation** (**sampling**).



A	B	C	D
T	F	T	T
...			

Estimate:

$$P^*(D|A) = \#D,A / \#A$$

- Flip a coin where $P(T)=0.4$, assume we get T, use that value for A
- Given $A=T$, lookup $P(B|A=T)$ and flip a coin with that prob., assume we get F
- Similarly for C and D
- We get one sample from joint distribution of these four vars

Lecture 16 • 65

Let's say you want to know the probability of D given A. How would you answer -
- given all the examples -- what would you do to compute the probability of D given A? You would just count. You'd count the number of cases in which A and D were true, and you'd divide that by the number of cases in which A was true, and that would give you an unbiased estimate of the probability of D given A. The more samples, the more confidence you'd have that the estimated probability is close to the true one.

Estimation

- Some probabilities are easier than others to estimate

It's going to turn out that some probabilities are easier than other ones to estimate.

Estimation

- Some probabilities are easier than others to estimate
- In generating the table, the rare events will not be well represented
- $P(\text{Disease} | \text{spots-on-your-tongue, sore toe})$

Imagine that you want to estimate the probability of some disease given -- oh, I don't know -- spots on your tongue and a sore toe. Somebody walks in and they have a really peculiar set of symptoms, and you want to know what's the probability that they have some disease.

Estimation

- Some probabilities are easier than others to estimate
- In generating the table, the rare events will not be well represented
- $P(\text{Disease} \mid \text{spots-on-your-tongue, sore toe})$
- If spots-on-your-tongue and sore toe are not root nodes, you would generate a huge table but the cases of interest would be very sparse in the table

Lecture 16 • 69

Well, if the symptoms are root nodes, it's easy. If the symptoms were root nodes, you could just assign the root nodes to have their observed values and then simulate the rest of the network as before.

Estimation

- Some probabilities are easier than others to estimate
- In generating the table, the rare events will not be well represented
- $P(\text{Disease} \mid \text{spots-on-your-tongue, sore toe})$
- If spots-on-your-tongue and sore toe are not root nodes, you would generate a huge table but the cases of interest would be very sparse in the table

Lecture 16 • 70

But if the symptoms aren't root nodes then if you do naïve sampling, you would generate a giant table of samples, and you'd have to go and look and say, gosh, how many cases do I have where somebody has spots on their tongue and a sore toe; and the answer would be, well, maybe zero or not very many.

Estimation

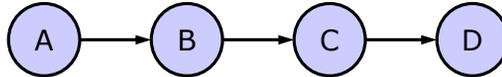
- Some probabilities are easier than others to estimate
- In generating the table, the rare events will not be well represented
- $P(\text{Disease} \mid \text{spots-on-your-tongue, sore toe})$
- If spots-on-your-tongue and sore toe are not root nodes, you would generate a huge table but the cases of interest would be very sparse in the table
- **Importance sampling** lets you focus on the set of cases that are important to answering your question

Lecture 16 • 71

There's a technique called importance sampling, which allows you to draw examples from a distribution that's going to be more helpful and then reweight them so that you can still get an unbiased estimate of the desired conditional probability. It's a bit beyond the scope of this class to get into the details, but it's an important and effective idea.

Recitation Problem

- Do the variable elimination algorithm on the net below using the elimination order A,B,C (that is, eliminate node C first). In computing $P(D=d)$, what factors do you get?
- What if you wanted to compute the whole marginal distribution $P(D)$?

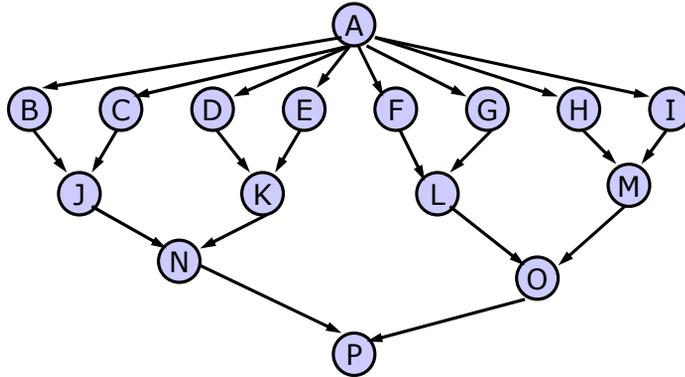


Lecture 16 • 72

Here's the network we started with. We used elimination order C, B, A (we eliminated A first). Now we're going to explore what happens when we eliminate the variables in the opposite order. First, work on the case we did, where we're trying to calculate the probability that node D takes on a particular value, little d. Remember that little d is a constant in this case. Now, do the case where we're trying to find the whole distribution over D, so we don't know a particular value for little d.

Another Recitation Problem

Find an elimination order that keeps the factors small for the net below, or show that there is no such order.



Lecture 16 • 73

Here's a pretty complicated graph. But notice that no node has more than 2 parents, so none of the CPTs are huge. The question is, is this graph hard for variable elimination? More concretely, can you find an elimination order that results only in fairly small factors? Is there an elimination order that generates a huge factor?

The Last Recitation Problem (in this lecture)

Bayesian networks (or related models) are often used in computer vision, but they almost always require sampling. What happens when you try to do variable elimination on a model like the grid below?

