UNIT II PROBABILISTIC REASONING

Acting under uncertainty – Bayesian inference – naïve bayes models. Probabilistic reasoning – Bayesian networks – exact inference in BN – approximate inference in BN – causal networks.

1. Acting under uncertainty

Uncertainty:

- ❖ Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.
- ❖ So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertainreasoning or probabilistic reasoning.

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

- ❖ Information occurred from unreliable sources.
- ❖ Experimental Errors
- ❖ Equipment fault
- ❖ Temperature variation
- ❖ Climate change.

2. Bayesian inference

Bayes' theorem in Artificial intelligence

Bayes' theorem:

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability ofan event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an applicationof Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B). Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancermore accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A

with known event B: As from product rule we can write:

P(A ∧ B)= P(A|B) P(B) or

Similarly, the probability of event B with known event A:

P(A ∧ B)= P(B|A) P(A)

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systemsfor **probabilistic inference**.

It shows the simple relationship between joint and conditional probabilities. Here,

P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when wehave occurred an evidence B.

P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence. P(A) is called the **prior probability**, probability of hypothesis before considering the evidence

P(B) is called **marginal probability**, pure probability of an evidence.

Applying Bayes' rule:

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(*B*), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})\, P(\text{cause})}{P(\text{effect})}$$

Example-1:

Question: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.

Solution:

$$P(\text{king}|\text{face}) = \frac{P(\text{Face}|\text{king}) * P(\text{King})}{P(\text{Face})} \quad ....(i)$$

P(king): probability that the card is King=

4/52= 1/13 P(face): probability that a card is

a face card= 3/13

P(Face|King): probability of face card when we assume it

is a king = 1 Putting all values in equation (i) we will get:

$$P(\text{king}|\text{face}) = \frac{1 * (\frac{1}{13})}{(\frac{3}{13})} = \mathbf{1/3,}$$ **it is a probability that a face card is a king card.**

Application of Bayes' theorem in Artificial intelligence:

Following are some applications of Bayes' theorem:

- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem.

3. Probabilistic reasoning

Probabilistic reasoning:

- Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.
- We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.
- In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- **Bayes' rule**
- **Bayesian Statistics**

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$0 \leq P(A) \leq 1$, where P(A) is the probability of an event A. P(A) = 0, indicates total uncertainty in an event A.

P(A) =1, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- o P(¬A) = probability of a not happening event.
- o P(¬A) + P(A) = 1.

4. Bayesian networks or Belief networks

Bayesian Belief Network in artificial intelligence

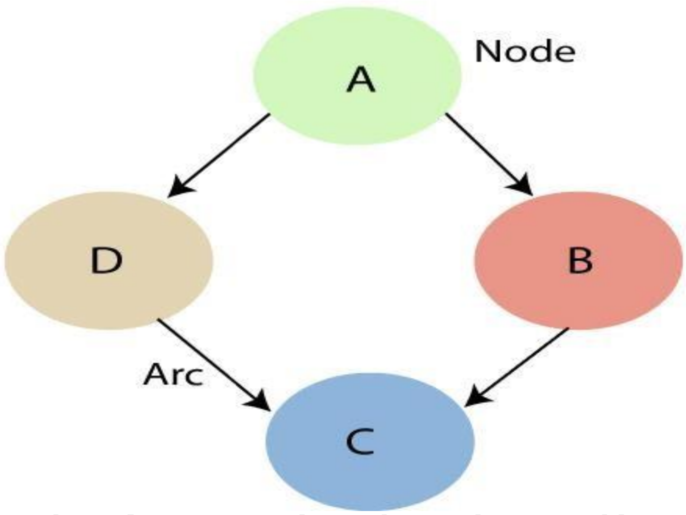
- ❖ Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem whichhas uncertainty. We can define a Bayesian network as:
- ❖ "A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditionaldependencies using a directed acyclic graph."
- ❖ It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.
- ❖ Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.
- ❖ Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we needa Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge isknown as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph. These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
  - **In the above diagram, A, B, C, and D are random variables represented by the nodes of thenetwork graph.**
  - **If we are considering node B, which is connected with node A by a directed arrow, then nodeA is called the parent of Node B.**
  - **Node C is independent of node A.**

Note: The Bayesian network graph does not contain any cyclic graph. Hence, it is known as a directed acyclic graph or DAG

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution **$P(X_i \mid Parent(X_i))$**, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint probability distribution:

If we have variables x1, x2, x3, , xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as

Joint probability distribution.

$P[x_1, x_2, x_3, , x_n]$, it can be written as the following way in terms of the joint probability distribution.

$= P[x_1 \mid x_2, x_3, \ldots, x_n] P[x_2, x_3, , x_n]$

$= P[x_1 \mid x_2, x_3, \ldots, x_n] P[x_2 \mid x_3, \ldots, x_n]. \quad P[x_{n-1} \mid x_n] P[x_n].$

In general for each variable Xi, we can write the equation as:

$P(X_i \mid X_{i-1}, \ldots, X_1) = P(X_i \mid Parents(X_i))$

Explanation of Bayesian network:

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem:

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as **conditional probabilities table or CPT.**
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

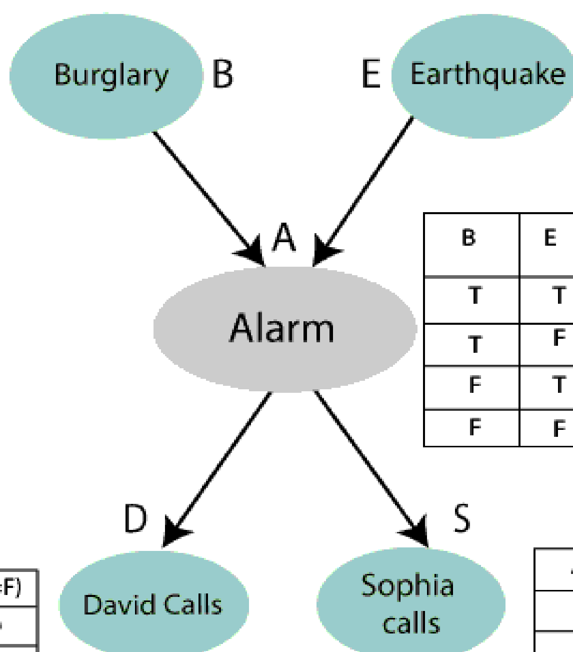
=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]

T	0.002
F	0.998



T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Let's take the observed probability for the Burglary and earthquake component: $P(B= True) = 0.002$, which is the probability of burglary.

$P(B= False)= 0.998$, which is the probability of no burglary.

$P(E= True)= 0.001$, which is the probability of a minor earthquake

$P(E= False)= 0.999$, Which is the probability that an earthquake not occurred.

5. Inference in Bayesian Networks

1. Exact inference
2. Approximate inference

1. **Exact inference**:

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

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

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)?

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

So, in general, we'd like to be able to compute a whole probability distribution over some variable orvariables 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 thequery

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

$$\Pr(d) = \sum_{ABC} \Pr(a,b,c,d)$$
$$= \sum_{a \in \mathrm{dom}(A)} \sum_{b \in \mathrm{dom}(B)} \sum_{c \in \mathrm{dom}(C)} \Pr(A = a \wedge B = b \wedge C = c)$$

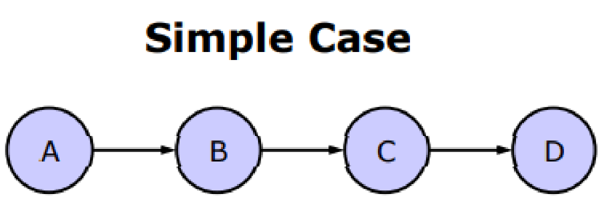
So, in a domain with four variables, A, B, C, and D, the probability that variable D has value d is the sum over all possible combinations of values of the other three variables of the joint probability of all four values. This is exactly the same as the procedure we went through in the last lecture,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

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.

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

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.

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.



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.

Variable Elimination Algorithm

Given a Bayesian network, and an elimination order for the non-query variables , compute For i = m downto 1

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

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.

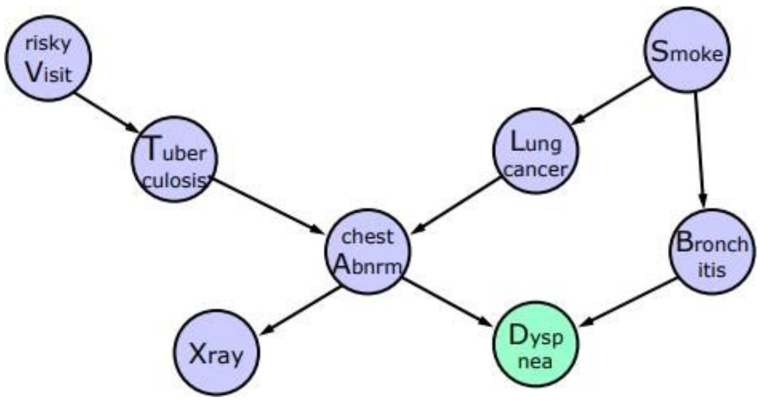
We can express the probability of the query variables as a sum over each value of each of the non- query variables of a product over each node in the network, of the probability that that variable has the given value given the values of its parents.

So, we'll eliminate the variables from the inside out. Starting with variable Xm and finishing with variable X1 To eliminate variable Xi, we start by gathering up all of the factors that mention Xi, and removingthem from our set of factors. Let's say there are k such factors.

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

We then sum the table over the Xi dimension, resulting in a k-dimensional table. This table is our new factor, and we put a term for it back into our set of factors. Once we've eliminated all the summations, we have the desired value.

One more example



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

$$\Pr(d) = \sum_{A,B,L,T,S,X} \begin{matrix} \Pr(d \mid a,b)\Pr(a \mid t,l)\Pr(b \mid s)\Pr(l \mid s)\Pr(s) \\ \Pr(x \mid a)\underbrace{\sum_{V}\Pr(t \mid v)\Pr(v)}_{f_1(t)} \end{matrix}$$

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

$$\Pr(d) = \sum_{A,B,L,T,S} \begin{matrix} \Pr(d \mid a,b)\Pr(a \mid t,l)\Pr(b \mid s)\Pr(l \mid s)\Pr(s) f_1(t) \\ \underbrace{\sum_{X}\Pr(x \mid a)}_{1} \end{matrix}$$

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

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

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

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

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

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

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).

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

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 f1 of T.

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

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.

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.

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.

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

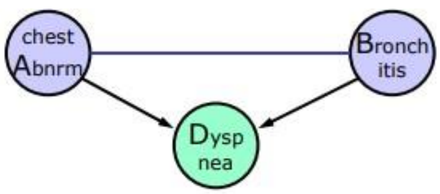
Now we can substitute that factor back into our expression. We can also eliminate node S. But in eliminating S, we've added a direct dependency between L and B (they used to be dependent via S, but now the dependency is encoded explicitly in f2(b). We'll show that in the graph by drawing a line between the two nodes. It's not exactly a standard directed conditional dependence, but it's still useful to show that they're coupled.

Now we eliminate T. It involves two terms, which themselves involve variables A and L. So we make anew factor f3 of A and L.

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

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

Next we eliminate L. It involves these two factors, which depend on variables A and B. So we make a new factor, f4 of A and B, and substitute it in. We remove node L, but couple A and B.



$$\Pr(d) = \sum_{A,B} \Pr(d \mid a,b) f_4(a,b)$$

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.

$$\Pr(d) = \sum_{A} \underbrace{\sum_{B} \Pr(d \mid a,b) f_4(a,b)}_{f_5(a)}$$

It involves both of our remaining terms, and it seems to depend on variables A and D. However, in thiscase, 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).

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

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

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

- 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

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 time = O(n^2 b^k).

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.

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.

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.

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 thelargest 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 additionalconnections into the network.

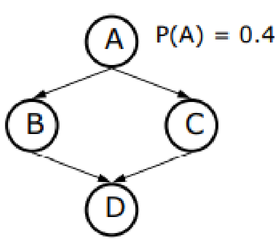
So, inference in polytrees is efficient, and even in many large non-polytree networks, it's possible to keepthe factors small, and therefore to do inference relatively efficiently.

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

2. <u>Approximate inference:</u>

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

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 thatit'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.

So, the other thing that we can do is the stochastic simulation or sampling. In sampling, what we do is welook 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.

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.

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

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

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 coinwith that probability, in order to get a value for D.

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 variousquestions.

Estimate:

P*(D|A) = #D,A / #A

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 wastrue, and that would give you an unbiased estimate of the probability of D given A. The more samples, themore confidence you’d have that the estimated probability is close to the true one.

Estimation

- Some probabilities are easier than others to estimate
- In generating the table, the rare events will not be well represented
- P(Disease| spots-on-your-tongue, sore toe)
- If spots-on-your-tongue and sore toe are not root nodes, you would generate a huge table butthe 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 yourquestion

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

Exactly because of the process we're using to generate the samples, the majority of them will be the typical cases. Oh, it's someone with a cold, someone with a cold, someone with a cold, someone with a cold, someone with a cold, someone with malaria, someone with a cold, someone with a cold. So the rareresults are not going to come up very often. And so doing this sampling naively can make it really hard toestimate the probability of a rare event. If it's something that happens one in ten thousand times, well, you know for sure you're going to need, some number of tens of thousands of samples to get even a reasonable estimate of that probability.

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 youwant to know what's the probability that they have some disease.

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

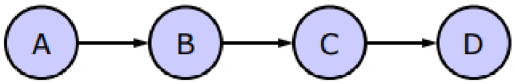
But if the symptoms aren't root nodes then if you do naïve sampling, you would generate a giant table ofsamples, and you'd have to go and look and say, gosh, how many cases do I have where somebody has spots on their tongue and a sore toe; and the answer would be, well, maybe zero or not very many.

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'san 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?

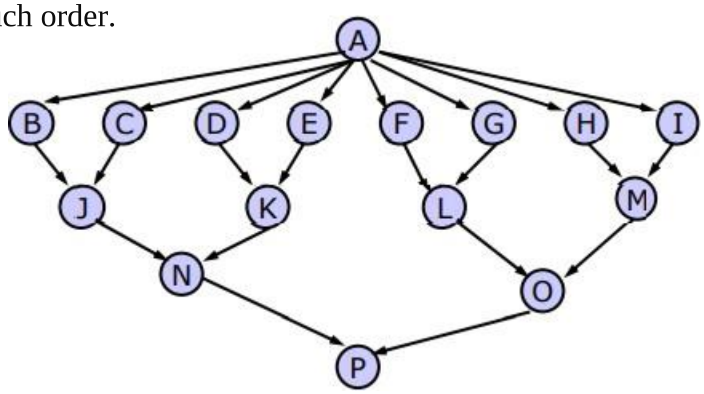
- What if you wanted to compute the whole marginal distribution P(D)?



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.

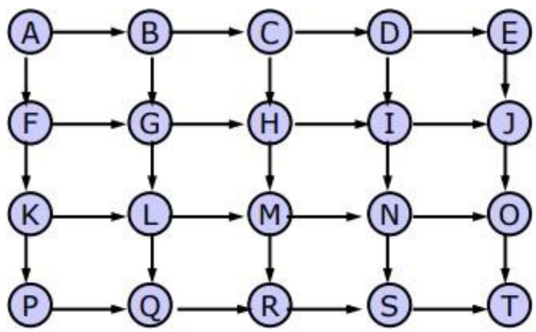


Here's a pretty complicated graph. But notice that no node has more than 2 parents, so none of the CPTs are huge. The question is, is this graph hard for variable elimination? More concretely, can you find an elimination order that results only in fairly small factors? Is there an elimination order that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

The Last Recitation Problem

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

The illustration above shows a causal network corresponding to the rules $\{B\,B \to A,\ A\,A\,B \to B\,A\,A\,B\}$ (applied in a left-to-right scan) and initial condition $A\ B\ A\ A\ B$



6. <u>Casual Networks:</u>

A causal network is an acyclic digraph arising from an evolution of a substitution system, and representing its history.