

**What we can calculate**

**Lecture 19**

**Inference in Belief Network**

After we know what the belief network tells us, we want to use it to compute posterior probability.

But doing it in the naive way is very inefficient. We will see how this can be done in the same time complexity as the size of the network.

**Reference:**

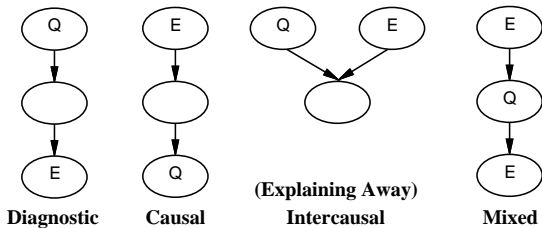
- Textbook Chapter 15, section 15.3–15.4

- The query to answer: given **exact knowledge about some evidence**, what is the **posterior probability that a particular event** happens?
- These variables are called **evidence variables** and **query variable** respectively.
- Usually some variables naturally serves as evidence variables (e.g., JohnCall, MaryCall) while some other variables naturally serve as query variables (e.g., Burglary).
- But the network is **very general**: any variable can serve as evidence variable and query variable.
- Since the network is usually drawn from causes to consequences, we can categorize reasoning into **diagnostic, causal, intercausal** and **mixed**.

AI(0270)

AI(0270)-19.1

**The different types of reasoning**



Intercausal reasoning will make sense only when there are some other evidence down the network. Otherwise the evidence and the query is independent.

Note that these are just terminology used to describe the type of inference in various systems. Using belief network, we **don't need to distinguish the type of reasoning it performs**.

I.e., it treats everything as mixed.

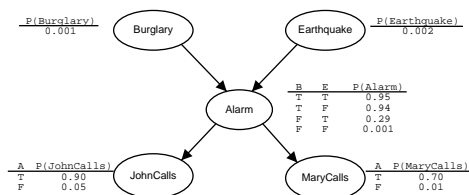
**But we know the joint...**

- First, do we have **enough information** to answer such queries?
- The answer is clearly **yes**. We can compute **any** posterior (conditional) probability given the full joint, ...
- ... and the belief network can be understood as **a way to specify the full joint**. We got all the probabilities that we need.
- Big problem: the number of cells of the full joint which needs to be examine is **exponentially** on the number of variables that are not an evidence variable.  
The difficulty to find a joint entry by multiplying everything becomes just a very minor issue.
- We need a better way, which does **not** require us to find **many probabilities in the full joint**.

AI(0270)-19.2

AI(0270)-19.3

**Example**



Suppose we know JohnCall and MaryCall, and want to find  $P(A | J, M)$ . Before looking onwards, what you expect about this probability? Hint: most people overestimate this probability.

$$P(B, J, M) = P(B, J, M, A, E) + P(B, J, M, A, \neg E) + P(B, J, M, \neg A, E) + P(B, J, M, \neg A, \neg E).$$

And similar for  $P(\neg B, J, M)$ . We list the following table...

**The entries we need...**

Condition	P(B)	P(E)	P(A)	P(J)	P(M)	Product
B, E, A	0.001	0.002	0.95	0.90	0.70	0.000001197
B, E, ¬A	0.001	0.002	0.05	0.05	0.01	0.00000000005
B, ¬E, A	0.001	0.998	0.94	0.90	0.70	0.0005910156
B, ¬E, ¬A	0.001	0.998	0.06	0.05	0.01	0.00000002994
Sum						.00059224259
¬B, E, A	0.999	0.002	0.29	0.90	0.70	0.0003650346
¬B, E, ¬A	0.999	0.002	0.71	0.05	0.01	0.00000070929
¬B, ¬E, A	0.999	0.998	0.001	0.90	0.70	0.00062811126
¬B, ¬E, ¬A	0.999	0.998	0.999	0.05	0.01	0.000498002499
Sum						0.001491857649

Our needed probability:  $\frac{0.000592215644}{0.000592215644 + 0.001491857649}$  which is around 28.4%.

AI(0270)-19.4

AI(0270)-19.5

### Our strategy

- But this quickly gets **too slow** when the number of non-evidence variables grows to more than around 20.
- **How to make it faster?** Two strategies:
  - Ignore the values of the joint, and use only conditional probabilities. Use Bayes rule to separate the evidence variables from below and from above, and turn the whole computation into a recursion. This is the approach taken by the book.
  - Strategically combine some of the rows of the the last table we have seen, to reduce the number of multiplications.
- Both assumes that the network is singly connected, i.e., is a **polytree**. Later we discuss how to deal with multiply connected networks.
- When done correctly, the two approaches should have similar complexity. Following the book, we will just discuss the first approach.

AI(0270)-19.6

### Example

Rather than going directly into the algorithm, let's try to do the evaluation **just using the basic idea to use Bayes rule**.

- Suppose we want to calculate  $P(B | J, M)$  as before.
- With Bayes' rule,  $P(B | J, M) = \alpha P(B)P(J, M | B)$ .  
Why we need to do this? We want our probabilities from top to bottom, since we only have the causal probabilities. So we reverse the original query.
- The first part is easy:  $P(B) = 0.001$ ,  $P(\neg B) = 0.999$ .  
We always calculate both the  $B$  and  $\neg B$  probabilities for normalization.
- But how to deal with the second part?  $J$  and  $M$  are **too far from the cause  $B$** ...
- We would like to make  $A$ , instead of  $B$ , as the condition variable. Then we can read off  $P(J | A)$  and  $P(M | A)$  from the CPTs of  $J$  and  $M$ .

AI(0270)-19.7

### Adding evidence

- But how to make  $A$  instead of  $B$  to be the condition variable?
- We do it be **enumerating all possibilities** for  $B$  to occur. I.e.,  
$$P(J, M | B) = P(J, M | A, B)P(A | B) + P(J, M | \neg A, B)P(\neg A | B)$$
- Because  $A$  d-separate  $J, M$  from  $B$ , we don't need to write  $B$ ...  
$$P(J, M | B) = P(J, M | A)P(A | B) + P(J, M | \neg A)P(\neg A | B)$$
- Now we separate  $J$  and  $M$ :  
$$P(J, M | B) = P(J|A)P(M|A)P(A|B) + P(J|\neg A)P(M|\neg A)P(\neg A|B)$$
- So we have the basic idea: **to push the condition variables down the network** by enumeration, until we hit the evidence variable.
- Then we can directly read off CPTs entries. Here, everything can be read off directly, except  $P(A | B)$ .

AI(0270)-19.8

### The actual calculations

- Now our only problem is  $P(A|B)$ . We use the enumeration trick again.  
$$P(A|B) = P(A|B, E)P(E|B)P(B|B) + P(A|B, \neg E)P(\neg E|B)P(B|B)$$
  
There are two other parts, for the cases when  $\neg B$  holds. But it will be conditioned by  $B$ , so the probabilities are 0.
- Note the important property of the enumeration trick: **we keep the ordering**, i.e., the conditions remains to be causes.
- $P(B | B)$  is of course 1.
- $B$  and  $E$  independent:  $P(A|B) = P(A|B, E)P(E) + P(A|B, \neg E)P(\neg E)$
- So we have...  
$$P(A | B) = 0.95 \times 0.002 + 0.94 \times 0.998 = 0.94002$$
  
$$P(A | \neg B) = 0.29 \times 0.002 + 0.001 \times 0.998 = 0.001578$$
- Finally we get all the probabilities we need...

AI(0270)-19.9

### The posterior probability is...

- We already had this:  
$$P(J, M|B) = P(J|A)P(M|A)P(A|B) + P(J|\neg A)P(M|\neg A)P(\neg A|B)$$
- $P(J, M|B) = 0.90 \times 0.70 \times 0.94002 + 0.05 \times 0.01 \times 0.05998 = 0.59224259$ .
- $P(J, M|\neg B) = 0.90 \times 0.70 \times 0.001578 + 0.05 \times 0.01 \times 0.998422 = 0.001493351$ .
- Now, we can go back to our first formula:  
$$P(B | J, M) = \alpha P(B)P(J, M | B)$$
- $P(B)P(J, M | B) = 0.001 \times 0.59224259 = 0.00059224259$ .
- $P(\neg B)P(J, M | \neg B) = 0.999 \times 0.001493351 = 0.001491857649$ .
- Normalizing:  $P(B | J, M) = \frac{0.00059224259}{0.00059224259 + 0.001491857649} = 28.4\%$

AI(0270)-19.10

### General strategies

So we have seen our **basic strategy**. To summarize:

- If we need to find the probability of a variable given other variables **below** it, then use Bayes rule to reverse the condition and query variables.
- If we need to find the probability of a variable (or a set of variables) given other variables **above** it, then enumerate all variables.
- What if we need to find the probability of a variable give other variables, some **above** and some **below**?
- We will use Bayes' rule to **break it into one part which is from above, and one part which is from below**.
- E.g., if  $A$  is from above,  $B$  is from below, and we query  $Q$ , then we use  
$$P(Q | A, B) = \alpha P(Q | A)P(B | Q, A) = \alpha P(Q | A)P(B | Q)$$
  
The latter comes from that  $Q$  d-separates  $A$  and  $B$ .

AI(0270)-19.11

### General case for polytrees...

Our example is **particularly** easy because...

- There is no evidence above  $B$ .
- $B$  only has one "child" variable.
- $A$  only has one "parent",  $E$ , other than  $B$ .
- $E$  and all children of  $B$  are terminal, no other branches.

The general case is more complicated, and we will want a general strategy for handling them.

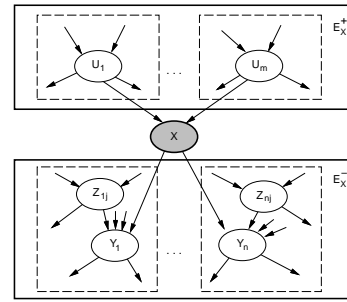
We also want to avoid having to invoke low-level Bayes' rule and enumeration everytime we need it.

And want to make sure the whole procedure is quick enough for all possible query and evidence variables.

AI(0270)-19.12

### Getting the probability

Suppose we want  $P(X | E)$ , where  $E$  is a set of evidence variables.



$U_i$ : variables connected to  $X$  from above

$Y_i$ : variables connected to  $X$  from below

$Z_j$ : variables connected to  $Y$ 's from above

$E_X^+$ : evidence from above.

$E_X^-$ : evidence from below.

$E_{U_m|X}$ : evidence connected to  $U_m$  but not through  $X$ . (i.e., all evidence in top-right subblock)

Note:  $E = E_X^+ \cup E_X^-$

AI(0270)-19.13

### Breaking it up...

Now, let's see how to do  $P(X | E)$ .

- First, some **evidences** might be from the **wrong direction** to  $X$ . We use Bayes' rule to turn it to correct the direction.
- I.e.,  $P(X | E) = P(X | E_X^+, E_X^-) = \alpha P(X | E_X^+) P(E_X^- | X, E_X^+)$   
This is conditioned Bayes' rule, using  $E_X^+$  as background knowledge. Note that every variable is in the direction from cause to consequence now... with respect to  $X$ . But they might be in the wrong direction further down.
- But  $X$  d-separates  $E_X^-$  and  $E_X^+$ , so we have  
 $P(X | E) = P(X | E_X^+, E_X^-) = \alpha P(X | E_X^+) P(E_X^- | X)$
- Now we separately evaluate  $P(X | E_X^+)$  and  $P(E_X^- | X)$

AI(0270)-19.14

### The probability from above

Now let's see how to find  $P(X | E_X^+)$ .

- The enumeration trick is used. So we will introduce **all** causes directly connected to  $X$ , i.e., all the  $U$ 's.
- $P(X | E_X^+) = \sum P(X | E_X^+, U_1 = u_1, U_2 = u_2, \dots) P(U_1 = u_1, U_2 = u_2, \dots | E_X^+)$   
The sum is for all possible values of  $u_i$ 's.
- We abbreviate it by saying the values  $u_i$ 's form a vector  $\mathbf{u}$ , so  
 $P(X | E_X^+) = \sum_{\mathbf{u}} P(X | E_X^+, \mathbf{u}) P(\mathbf{u} | E_X^+)$
- Note that, since we don't have any evidence from below, the variable  $X$  makes all the  $U_i$  independent (technically we say  $E_X^+$  d-separates all the  $U_i$ 's). We get...  
 $P(X | E_X^+) = \sum_{\mathbf{u}} P(X | E_X^+, \mathbf{u}) \prod_i P(U_i = u_i | E_X^+)$

AI(0270)-19.15

### So, we get this...

$$P(X | E) = \alpha P(E_X^- | X) \sum_{\mathbf{u}} P(X | E_X^+, \mathbf{u}) \prod_i P(U_i = u_i | E_X^+)$$

In the 2nd term,  $\mathbf{u}$  d-separate all  $E_X^+$  from  $X$ . In the 3rd term, without knowledge of  $X$  and all its descendents,  $U_i$  is independent of all evidence variables that are not in its own subtree. So...

$$P(X | E) = \alpha P(E_X^- | X) \sum_{\mathbf{u}} P(X | \mathbf{u}) \prod_i P(U_i = u_i | E_{U_i|X})$$

Is it something good? **Yes!** There are 3 terms on the RHS:

- The first term is something we will tackle next.
- The second term is a lookup to the CPT.
- The third term is a recursive call to itself. Note that this **always recurse up the tree, never down**. This means that **every node need only be processed once!**

AI(0270)-19.16

### The remaining probability

Now we need to find  $P(E_X^- | X)$ .

- The variables of  $P(E_X^- | X)$  can be from **different branches**. Since  $X$  d-separate them, they are all independent. So

$$P(E_X^- | X) = \prod_i P(E_{Y_i|X} | X)$$

- Following the example that we have done, we do **enumeration** now. First let's enumerate all possible values of  $Y$ .

$$P(E_X^- | X) = \prod_i \sum_{y_i} P(E_{Y_i|X} | Y_i = y_i, X) P(Y_i = y_i | X)$$

- We want to say  $Y_i$  d-separates  $X$  from  $E_{Y_i|X}$ , but this is **not** the case.
- Why? **Some of the variables in  $E_{Y_i|X}$  might be from above  $Y_i$ !** So only some of the variables are independent. Luckily, it is easy to break them out.

AI(0270)-19.17

### The remaining probability (cont'd)

- We break  $P(E_{Y_i|X}^-)$  into variables from above ( $P(E_{Y_i|X}^+)$ ) and those from below ( $P(E_{Y_i}^-)$ ):

$$P(E_{Y_i|X}^-) = \prod_i \sum_{Y_i} P(E_{Y_i}^-, E_{Y_i|X}^+ | Y_i = y_i, X) P(Y_i = y_i | X)$$

- Given  $Y_i$ , the probability from above  $Y_i$  and the probability from below  $Y_i$  are independent. So are  $X$  and the variables below  $Y_i$ . So

$$P(E_{Y_i|X}^-) = \prod_i \sum_{Y_i} P(E_{Y_i}^- | Y_i = y_i) P(E_{Y_i|X}^+ | Y_i = y_i, X) P(Y_i = y_i | X)$$

- Now the first term is just a **recursion**, which **always recurse down the tree**. Again, this won't require any node to be processed more than once.
- But **how to deal with the second and third term?**  
Our earlier example is simpler in the the second term does not exist.

AI(0270)-19.18

### A bit of manipulations

Our target is now  $P(E_{Y_i|X}^+ | Y_i = y_i, X) P(Y_i = y_i | X)$ .

- Note that the  $E_{Y_i|X}^+$  is in the **wrong direction**. We want to use Bayes rule, but can't because it involves the condition  $X$ .  
Due to explaining away,  $E_{Y_i|X}^+$  are dependent each other and on  $X$ , given  $Y_i$ .

- The solution: **enumerate** the  $Z$  variables as well, so that the  $Z$  variables d-separate all the  $E_{Y_i|X}^+$  variables. We get this:

$$\sum_{\mathbf{z}} P(E_{Y_i|X}^+ | Y_i = y_i, \mathbf{z}, X) P(\mathbf{z} | Y_i = y_i, X) P(Y_i = y_i | X)$$

- The second and third term easily combines to form  $P(\mathbf{z}, Y_i = y_i | X)$ , and redistribute it like  $P(\mathbf{z} | X) P(Y_i = y_i | X, \mathbf{z})$ .
- Without other knowledge below  $Y_i$ ,  $\mathbf{z}$  is independent of  $X$ . Also, with  $\mathbf{z}$ ,  $E_{Y_i|X}^+$  is independent on  $X$  and  $Y_i$ . So we get

$$\sum_{\mathbf{z}} P(E_{Y_i|X}^+ | \mathbf{z}) P(\mathbf{z}) P(Y_i = y_i | X, \mathbf{z})$$

AI(0270)-19.19

### Applying Bayes' rule

Now we can finally apply Bayes' rule on it.

- $P(E_{Y_i|X}^+ | \mathbf{z}) = P(\mathbf{z} | E_{Y_i|X}^+) P(E_{Y_i|X}^+) / P(\mathbf{z})$   
Can't just say  $\alpha$  because we have a sum on  $\mathbf{z}$  outside.

- So the product we seek is...

$$\begin{aligned} & \sum_{\mathbf{z}} P(E_{Y_i|X}^+ | \mathbf{z}) P(\mathbf{z}) P(Y_i = y_i | X, \mathbf{z}) \\ &= \sum_{\mathbf{z}} P(\mathbf{z} | E_{Y_i|X}^+) P(E_{Y_i|X}^+) P(Y_i = y_i | X, \mathbf{z}) \\ &= P(E_{Y_i|X}^+) \sum_{\mathbf{z}} P(\mathbf{z} | E_{Y_i|X}^+) P(Y_i = y_i | X, \mathbf{z}) \end{aligned}$$

- Thus we have

$$\begin{aligned} P(E_{Y_i|X}^- | X) &= \prod_i \sum_{Y_i} P(E_{Y_i}^- | Y_i = y_i) P(E_{Y_i|X}^+) \sum_{\mathbf{z}} P(\mathbf{z} | E_{Y_i|X}^+) P(Y_i = y_i | X, \mathbf{z}) \\ &= \prod_i P(E_{Y_i|X}^+) \sum_{Y_i} P(E_{Y_i}^- | Y_i = y_i) \sum_{\mathbf{z}} P(\mathbf{z} | E_{Y_i|X}^+) P(Y_i = y_i | X, \mathbf{z}) \end{aligned}$$

AI(0270)-19.20

### So we get these recurrences...

- For the third term, since we have no evidence at or below  $Y_i$ , the  $\mathbf{z}$ 's are independent. We will also take the  $P(E_{Y_i|X}^+)$  out as a big normalizing constant. So we have

$$\begin{aligned} P(E_{Y_i|X}^- | X) &= \beta \prod_i \sum_{Y_i} P(E_{Y_i}^- | Y_i = y_i) \sum_{\mathbf{z}} P(Y_i = y_i | X, \mathbf{z}) \prod_j P(Z_{ij} = z_{ij} | E_{Z_{ij}|Y_i}) \end{aligned}$$

- Let's see what the whole expression is...

- First term: simple recursion, always done **down the network**.
- Second term: CPT lookup.
- Third term: recursive instance of  $P(X | E)$ , this time always done **up the network**, but luckily **we don't need to call it for  $X$** .
- So **each node has to be processed exactly once**.

AI(0270)-19.21

### The algorithm

- So the **algorithm** just needs **two recurrence**:

$$P(X | E) = \alpha P(E_X^- | X) \sum_{\mathbf{u}} P(X | \mathbf{u}) \prod_i P(U_i = u_i | E_{U_i|X})$$

$$\begin{aligned} P(E_X^- | X) &= \beta \prod_i \sum_{Y_i} P(E_{Y_i}^- | Y_i = y_i) \sum_{\mathbf{z}} P(Y_i = y_i | X, \mathbf{z}) \prod_j P(Z_{ij} = z_{ij} | E_{Z_{ij}|Y_i}) \end{aligned}$$

- The  $\beta$  don't need to be explicitly calculated: we will merge it into the normalization of  $P(E_X^- | X)$
- The recursion terminates when it **hits an evidence variable**, and when it **reaches the top or the bottom of the network**.
- Each node is processed at most once. If the fan-in is bounded, the algorithm is **linear time**. Otherwise it is linear on the **total size of CPTs** of the network.
- We will see an example run of the algorithm in the next lecture.

AI(0270)-19.22