Exact inference (Ch. 14)
A Bayesian network (Bayes net) is:
(1) a directed graph
(2) acyclic

Additionally, Bayesian networks are assumed to be defined by conditional probability tables
(3) \( P(x \mid \text{Parents}(x)) \)

We have actually used one of these before...
I have been lax on capitalization (e.g. $P(a)$ vs. $P(A)$), but not today

Capitalization = set of outcomes
Lower-case = a single outcome
(by letter, so “a” is an outcome of “A”)

So $P(A) = <P(a), P(\neg a)>$
P($A$, $B$)=<$P(a,b)$, $P(a, \neg b)$, $P(\neg a,b)$, $P(\neg a,\neg b)$>
Bayesian Network

Bayesian network above represented by:

\[
P(a, \neg b, c, \neg d) = P(\neg d|a, \neg b, c)P(c|\neg b, a)P(\neg b|a)P(a)
= P(\neg d|\neg b, c)P(c|\neg b)P(\neg b|a)P(a)
\]

Last time we discussed how to go left to right, when making the network

Today we look at right to left (inference)
Our primary tool beyond this breakdown of $P(a,b,c,d)$ is the sum rule:

$$P(b, c, d) = \sum_a P(a, b, c, d) = P(a, b, c, d) + P(\neg a, b, c, d)$$

We will also use the normalization trick for conditional probability (and not divide)

$$P(a, b) = \alpha P(a | b)$$

$$P(a | b) = \alpha P(a, b)$$

... or ...

$$\alpha (P(a | b) + P(\neg a | b)) = 1$$

$$P(a, b) + P(\neg a, b) = 1/\alpha$$

need to sum all non-given info
Using just these facts, we can brute-force:

\[
P(D|a) = \sum_c \sum_b P(b, c, D|a)
\]

\[
= \alpha \sum_c \sum_b P(a, b, c, D)
\]

\[
= \alpha \sum_c \sum_b P(D|b, c) P(c|b) P(b|a) P(a)
\]

\[
= \alpha P(a) \sum_b P(b|a) \sum_c P(c|b) P(D|b, c)
\]

Upper-case is both pos and neg (thus \(P(D|a)\) is array... here do formula twice) ... to find alpha

more efficient than previous
Exact Inference: Enumeration

\[
\sum_b \sum_c P(D|b,c)P(c|b)P(b|a)P(a)
\]

\[
P(a) \sum_b P(b|a) \sum_c P(D|b,c)P(c|b)
\]

nested double for-loop

non-summed = multiplied
Exact Inference: Enumeration

\[
\sum_b \sum_c P(D|b, c)P(c|b)P(b|a)P(a)
\]

\[
P(a) \sum_b P(b|a) \sum_c P(D|b, c)P(c|b)
\]

Used in computation more than once (inefficient)
We got lucky last time that we could eliminate all redundant calculations... not always so:

\[ P(D|b) = \alpha \sum_a P(b|a) P(a) \sum_c P(D|b,c) P(c|b) \]

We can always eliminate all redundancy, but need another approach:

Dynamic programming
Two common ways to compute the Fibonacci numbers are (which is better?):

(1) Recursive (like prior slides: enumeration)
   ```python
def fib(n):
    return fib(n-1) + fib(n-2)
```

(2) Array based (like upcoming slides)
   ```python
   a, b = 0, 1
   while b < 50:
       a, b = b, a + b
   ```

Dynamic Programming TL;DR
Dynamic programming exploits the structure between parts of the problem.

Rather than going top-down and having redundant computations along the way...

... dynamic programming goes bottom up and stores temporary results along the way.
Exact Inference: Var. Elim.

Variable elimination is the dynamic programming version for Bayesian networks.

This requires two new ideas:
(1) factors (denoted by “f”)
(2) “x” operator (called “pointwise product”)

Factors are the “stored info” that will represent the current product of probabilities.
Exact Inference: Var. Elim.

Factors are basically partial truth-tables (or matrices) depending on “input” variables.

The input variables: $f(A,B)$ are what effects the factors (much like probability $P(A,B)$).

When combining two factors with the “$\times$” operator, the input variables are union-ed:

$$f_{new}(A, B, C) = f_1(A, B) \times f_2(A, C)$$

Summing removes variables (like probabilities).
How the “x” operation works is: multiply “matching” T/F values

\[ f_{\text{new}}(A, B, C) = f_1(A, B) \times f_2(A, C) \]

For example (rand. numbers):

\[ f_{\text{new}}(a, \neg b, c) = f_1(a, \neg b) \cdot f_2(a, c) = 0.34 \cdot 0.41 = 0.1394 \]

<table>
<thead>
<tr>
<th>( f_1(A, B) )</th>
<th>( f_2(A, C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b )</td>
</tr>
<tr>
<td>( a )</td>
<td>( \neg b )</td>
</tr>
<tr>
<td>( \neg a )</td>
<td>( b )</td>
</tr>
<tr>
<td>( \neg a )</td>
<td>( \neg b )</td>
</tr>
<tr>
<td>( a )</td>
<td>( c )</td>
</tr>
<tr>
<td>( \neg a )</td>
<td>( \neg c )</td>
</tr>
<tr>
<td>( \neg a )</td>
<td>( c )</td>
</tr>
<tr>
<td>( \neg a )</td>
<td>( \neg c )</td>
</tr>
</tbody>
</table>

or w/e type of values
Exact Inference: Var. Elim.

Summation over the factors will work basically the same as probabilities:

$$f_{\text{new}}(A, B, C) = f_1(A, B) \times f_2(A, C)$$

You sum parts and remove it...

$$\sum_a f_1(A, B) = f_1(a, B) + f_1(\neg a, B) = f_{\text{new}}(B)$$

<table>
<thead>
<tr>
<th>$f_1(A, B)$</th>
<th>$f_{\text{new}}(B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b 0.12</td>
</tr>
<tr>
<td>a</td>
<td>¬b 0.34</td>
</tr>
<tr>
<td>¬a</td>
<td>b 0.56</td>
</tr>
<tr>
<td>¬a</td>
<td>¬b 0.78</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b 0.12 + 0.56</td>
</tr>
<tr>
<td></td>
<td>¬b 0.34 + 0.78</td>
</tr>
</tbody>
</table>
Exact Inference: Var. Elim.

Now we just represent the probabilities by factors and do “x” not normal multiplication

\[ P(D|b) = \alpha \sum_a P(b|a) P(a) \sum_c P(D|b, c) P(c|b) \]

\[ = \alpha \sum_a f_1(A) \times f_2(A) \times \sum_c f_3(C, D) \times f_4(C) \]

b is never negative, so not a variable

\[ f_{3,4}(C, D) = f_3(C, D) \times f_4(C) \]

... then repeat “x” and sum (sum is normal sum over all T/F values (in this case))
Exact Inference: Var. Elim.

\[ P(D|b) = \alpha \sum_a f_1(A) \times f_2(A) \times \sum_c \left[ f_{3,4}(C, D) \right] \]

\[ f_{3,4,c}(D) = f_{3,4}(c, D) + f_{3,4}(\neg c, D) \]

\[ P(D|b) = \alpha \sum_a f_1(A) \times \left[ f_2(A) \times f_{3,4,c}(D) \right] \]

\[ f_{2,3,4,c}(A, D) = f_2(A) \times f_{3,4,c}(D) \]

\[ P(D|b) = \alpha \sum_a f_{1,2,3,4,c}(A, D) \]

\[ P(D|b) = \alpha f_{1,2,3,4,a,c}(D) = \langle P(d|b), P(\neg d|b) \rangle \]

could also just call this \( f_5 \) or something
Using variable elimination, find:

\[ P(C \mid d, \neg a) \]
\[ P(C|\neg a, d) = \alpha \frac{P(\neg a)}{\sum_b \frac{P(C|b)}{f_1()} \frac{P(d|b, C)}{f_2(B,C)} \frac{P(b|\neg a)}{f_3(B,C)} f_4(B)} \]

\[ f_{3,4}(b, c) = P(d|b, c)P(b|\neg a) = 0.25 \cdot 0.3 = 0.075 \]
\[ f_{3,4}(b, \neg c) = P(d|b, \neg c)P(b|\neg a) = 1 \cdot 0.3 = 0.3 \]
\[ f_{3,4}(\neg b, c) = P(d|\neg b, c)P(\neg b|\neg a) = 0.15 \cdot (1 - 0.3) = 0.105 \]
\[ f_{3,4}(\neg b, \neg c) = P(d|\neg b, \neg c)P(\neg b|\neg a) = 0.05 \cdot (1 - 0.3) = 0.035 \]

\[ P(C|\neg a, d) = \alpha f_1() \times \sum_b f_2(B, C) \times f_{3,4}(B, C) \]

\[ f_{2,3,4}(b, c) = P(c|b) \times f(b, c) = 0.4 \cdot 0.075 = 0.03 \]
\[ f_{2,3,4}(b, \neg c) = P(\neg c|b) \times f(b, \neg c) = (1 - 0.4) \cdot 0.3 = 0.18 \]
\[ f_{2,3,4}(\neg b, c) = P(c|\neg b) \times f(\neg b, c) = 0.5 \cdot 0.105 = 0.0525 \]
\[ f_{2,3,4}(\neg b, \neg c) = P(\neg c|\neg b) \times f(\neg b, \neg c) = (1 - 0.5) \cdot 0.035 = 0.0175 \]

\[ P(C|\neg a, d) = \alpha f_1() \times \sum_b f_{2,3,4}(B, C) \]
\[ P(C|\neg a, d) = \alpha f_1() \times \sum_b f_{2,3,4}(B, C) \]

\[ f_{2,3,4,b}(c) = f_{2,3,4}(b, c) + f_{2,3,4}(\neg b, c) = 0.03 + 0.0525 = 0.0825 \]
\[ f_{2,3,4,b}(\neg c) = f_{2,3,4}(b, \neg c) + f_{2,3,4}(\neg b, \neg c) = 0.18 + 0.0175 = 0.1975 \]

\[ P(C|\neg a, d) = \alpha f_1() \times f_{2,3,4,b}(C) \]

\[ f_{1,2,3,4,b}(c) = f_1() \cdot f_{2,3,4,b}(c) = P(\neg a) \cdot f_{2,3,4,b}(c) = 0.9 \cdot 0.0825 = 0.07425 \]
\[ f_{1,2,3,4,b}(\neg c) = f_1() \cdot f_{2,3,4,b}(\neg c) = P(\neg a) \cdot f_{2,3,4,b}(\neg c) = 0.9 \cdot 0.1975 = 0.17775 \]

\[ P(C|\neg a, d) = \alpha f_{1,2,3,4,b}(C) \]

\[ P(c|\neg a, d) = 0.29464285714 \]
\[ P(\neg c|\neg a, d) = 0.70535714286 \]
Exact Inference: Side Note

If you try to find $P(a|b)$ using either of these approaches:

$$P(a|b) = \alpha \sum_c \sum_d P(d|b, c) P(c|b) P(\neg b|a) P(a)$$

$$= \alpha P(b|a) P(a) \sum_c P(c|b) \sum_d P(d|\neg b, c)$$

$$= \alpha P(b|a) P(a) \sum_c P(c|b) \cdot 1$$

$$= \alpha P(b|a) P(a)$$

True for every non-ancestor of “b” or “a”
The order that you sum/combine factors can have a significant effect on runtime.

However, there is no fast (i.e. worthwhile) way to compute the best ordering.

Instead, people quite often just use a greedy choice: combine/eliminate factors/variables to minimize resultant factor size.
Efficiency

A polytree is a graph where there is at most one undirected path between nodes/variables.

![Diagram of a polytree](image1)

NOT polytree
(two routes to d: b->c->d & b->d)

![Diagram of a non-polytree](image2)

Yes, polytree
(multiple roots are allowed)
Efficiency

Using the non-variable elimination way can result in exponential runtime

Using variable elimination:
On polytrees: Linear runtime
On non-polytrees: Exponential runtime :(

The details are a bit more nuanced, but basically exact inference is infeasible on non-polytrees (approximate methods for these)
Efficiency

You can do some preprocessing on graphs to cluster various parts:

The “b+c” node is much more complex (4 T/F value pairs, rather than a simple two T/F vals.)

Clustering can help when:

1. Can be efficient to change into polytree
2. Finding multiple probabilities
Efficiency

Not all nodes might be probabilistic

For example, if A is true then B is always true and if A false then B false (100% of the time)

Cases where nodes follow some formula (B=A), more efficient to not make a table

Two common formula are: noisy-OR and noisy-max (makes assumptions about parents)
We have primarily stuck to true/false values for variables for simplicity sake.

Variables could be any random variable (probability-value pair).

This includes continuous variables like normal/Gaussian distribution.
Non-discrete

Sometimes you can discretize continuous variables (much like pixels or grids on map)

Otherwise you can use them directly and integrate instead of summing (yuck)

Things can get a bit complicated if the Bayesian network has both continuous and discrete variables
Non-discrete

Discrete parent of continuous:
- Simply do by cases

\[ P(\text{cont}|\text{disc}) = \begin{cases} 
\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-4)^2}{2}}, & \text{if disc} \\
\frac{1}{\sqrt{8\pi}} e^{-\frac{(x-4)^2}{8}}, & \text{if } \neg \text{disc} 
\end{cases} \]

Continuous to discrete:
- Have to correlate ranges with probabilities

\[ P(\text{disc}|\text{cont} = x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \]

disc is true given cont has value \( x \) is: percent under the normal\((0,1)\) curve \( \leq x \)

![Shaded area represents probability](image)