Inference

- Three sets of variables:
  - Query variables: $E_1$
  - Evidence variables: $E_2$
  - Hidden variables: The rest, $E_3$

$$P(W \mid \text{Cloudy = True})$$

- $E_1 = \{W\}$
- $E_2 = \{\text{Cloudy = True}\}$
- $E_3 = \{\text{Sprinkler, Rain}\}$

$$P(E_1 \mid E_2) = P(E_1 \land E_2) / P(E_2)$$

Problem: Need to sum over the possible assignments of the hidden variables, which is exponential in the number of variables. Is exact inference possible?
A Simple Case

• Suppose that we want to compute $P(D = d)$ from this network.

\[
\sum_{c} \sum_{b} \sum_{a} P(A, B, C, D) = P(D = d)
\]

A Simple Case

• Compute $P(D = d)$ by summing the joint probability over all possible values of the remaining variables $A$, $B$, and $C$:

\[
P(D = d) = \sum_{a, b, c} P(A = a, B = b, C = c, D = d)
\]
A Simple Case

- Decompose the joint by using the fact that it is the product of terms of the form:
  \[ P(X \mid \text{Parents}(X)) \]

\[
P(D = d) = \sum_{a,b,c} P(D = d \mid C = c) P(C = c \mid B = b) P(B = b \mid A = a) P(A = a)
\]

A Simple Case

- We can avoid computing the sum for all possible triplets \((A,B,C)\) by distributing the sums inside the product

\[
P(D = d) = \sum_c P(D = d \mid C = c) \sum_b P(C = c \mid B = b) \sum_a P(B = b \mid A = a) P(A = a)
\]
A Simple Case

This term depends only on $B$ and can be written as a 2-valued function $f_A(b)$

$$P(D = d) = \sum_c P(D = d \mid C = c) \sum_b P(C = c \mid B = b) \sum_a P(B = b \mid A = a) P(A = a)$$

A Simple Case

This term depends only on $C$ and can be written as a 2-valued function $f_B(c)$

$$P(D = d) = \sum_c P(D = d \mid C = c) \sum_b P(C = c \mid B = b) f_A(b)$$

..... We have converted the computation of a sum over $2^3$ values to 3 sums over 2 values
A More Complicated Case

\[ P(D = d) = \sum_{a,b,c} P(D = d \mid C = c) P(C = c \mid B = b, A = a) P(B = b) P(A = a) \]

\[ = \sum_{c} P(D = d \mid C = c) \sum_{b} P(B = b) \sum_{a} P(C = c \mid B = b, A = a) P(A = a) \]

\[ = \sum_{c} P(D = d \mid C = c) \sum_{b} P(B = b) \sum_{a} f_1(a, b, c) \]

\[ = \sum_{c} P(D = d \mid C = c) \sum_{b} f_2(b, c) \]

Note: This is a more complicated term \( \bar{\text{requires a larger table to represent}} \)

In general: Overall complexity is exponential in the size of the largest term

Consequence: Complexity depends on the order in which the variables are eliminated \( \rightarrow \) Finding the optimal order can be exponential?
General Case: Variable Elimination

• Write the desired probability as a sum over all the unassigned variables
  \[ P(D = d) = \sum_{a,b,c} P(A = a, B = b, C = c, D = d) \]

• Distribute the sums inside the expression
  – Pick a variable
  – Group together all the terms that contain this variable
  \[ P(D = d) = \sum_c P(D = d | C = c) \sum_b P(C = c | B = b) \sum_a P(B = b | A = a) P(A = a) \]
  – Represent as a single function of the variables appearing in the group
  \[ P(D = d) = \sum_c P(D = d | C = c) \sum_b P(C = c | B = b) f_A(b) \]
  – Repeat until no more variables are left

Computation exponential in the size of the largest group \( \Rightarrow \) The order in which the variables are selected is important.
Important Special Case

• Still a bad worst-case scenario, but inference can be done efficiently in one case:
  – Polytrees: Undirected version of the graph is a tree = there is a single undirected path between two nodes

• In this case: Inference (P(X|E)) is linear in the size of the network $\Rightarrow O(2^k n)$ ($k$ = number of parents)

Note: All the examples so far have used binary variables for convenience. If the variables have domains of size $d$, all the results remain valid, replacing 2 by $d$:

- Size of the CPTs: $d^k$
- Complexity for polytrees: $d^k n$

• In both cases: Inference (P(X|E)) is linear in the size of the network $\Rightarrow O(2^k n)$
Joining Nodes

- The graph can be converted to a polytree by merging the variables (Join Tree), but the size of the CPTs has increased. Sprinkler+Rain has 4 possible values, which requires 8 numbers to represent the CPT $P(SR|C)$ instead of 4 to represent $P(S|C)$ and $P(R|C)$.
- It is always possible to build a join tree but, in general, may require exponentially many combinations of values. The hardness of the problem does not disappear.

Inference Summary

- Variable elimination polynomial on polytrees.
- NP-hard on arbitrary networks.
- Algorithms exist for converting networks to polytrees by grouping variables, but still NP-hard.
- Note: If the probabilities were all 0 and 1, this is similar to inference in logic seen earlier.
- Sampling algorithms used to get around the complexity problem.
Example
• Diagnostic system for lymph-node diseases.
• 60 diseases and 100 symptoms and test-results.
• 14,000 probabilities
• Expert consulted to make net.
• 8 hours to determine variables.
• 35 hours for net topology.
• 40 hours for probability table values.
• Apparently, the experts found it quite easy to invent the causal links and probabilities.
• Pathfinder is now outperforming the world experts in diagnosis. Extended to several dozen other medical domains.
Example

- Horvitz’ Lumiere system (precursor to Office Assistant)

For details:
• Intensive care monitoring
• 37 nodes
• 509 probabilities (instead of $2^{37}$)

Other Examples

• Plant monitoring
• Stock prices prediction
• Credit card fraud
• Spam filtering
• Design of science experiments
• Insurance risks
• ......
Bayes Nets

- Methodology for building Bayes nets.
- Requires exponential storage in the maximum number of parents of any node.
- We can compute the value of any assignment to the variables (entry in the joint distribution) in time linear in the number of variables.
- We can compute the answer to any question (any conditional probability)
- But inference is NP-hard in general
- Sampling (stochastic simulation) for approximate inference
- D-separation criterion to be used for extracting additional independence relations (and simplifying inference)
- Variable elimination and factorization for exact inference
- Exact inference in linear time for polytrees

Bayes Nets

- Material covered in Russell & Norvig, Chapter 14
- Not covered in lectures: Networks with continuous variables
- Not covered in chapter: d-separation