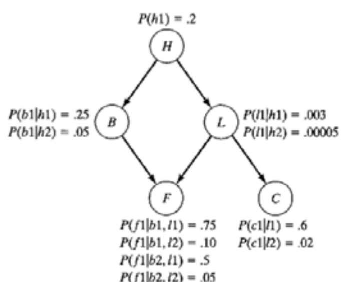
CSCE 970 Lecture 6: Inference on Discrete Variables

Stephen D. Scott

Introduction

- Now that we know what a Bayes net is and what its properties are, we can discuss how they're used
- Recall that a parameterized Bayes net defines a joint probability distribution over its nodes
- We'll take advantage of the factorization properties of the distribution defined by a Bayes net to do inference
  - Given values for a subset of the variables, what is the marginal probability distribution over a subset of the rest of them?

Introduction : Example

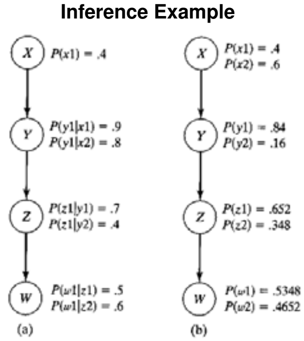


- Above figure is distribution over smoking history, bronchitis, lung cancer, fatigue, and chest X-ray
- If $H = h1$ ("yes" on smoking history) and $C = c1$ (positive chest X-ray), what are probabilities of lung cancer ($P(\ell 1 \mid h1, c1)$) and bronchitis ($P(b1 \mid h1, c1)$)?
  - Each query conditioned on two vars and marginalizes over two

Outline

- Inference examples
- Pearl's message-passing algorithm
  - Binary trees
  - Singly-connected networks
  - Multiply-connected networks
  - Time complexity
- The noisy OR-gate model
- The SPI algorithm

Inference Example

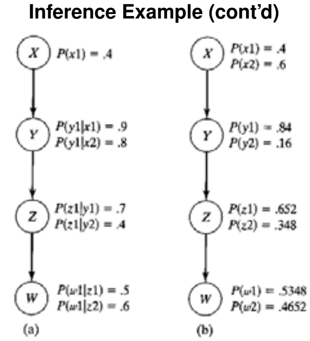


$$P(y1) = P(y1 \mid x1)P(x1) + P(y1 \mid x2)P(x2) = 0.84$$

$$P(z1) = P(z1 \mid y1)P(y1) + P(z1 \mid y2)P(y2) = 0.652$$

$$P(w1) = P(w1 \mid z1)P(z1) + P(w1 \mid z2)P(z2) = 0.5348$$

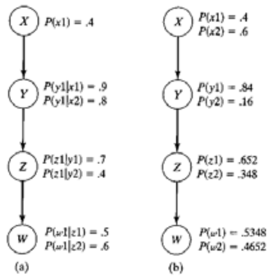
Inference Example (cont'd)



Instantiating X to $x1$:

$$P(y1 \mid x1) = 0.9$$

Inference Example (cont'd)

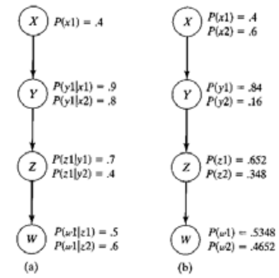


Instantiating X to $x1$:

$$\begin{aligned} P(z1 \mid x1) &= P(z1 \mid y1, x1)P(y1 \mid x1) + P(z1 \mid y2, x1)P(y2 \mid x1) \\ &= P(z1 \mid y1)P(y1 \mid x1) + P(z1 \mid y2)P(y2 \mid x1) \\ &= (0.7)(0.9) + (0.4)(0.1) = 0.67 \end{aligned}$$

(Second equality comes from CI result of Markov property)

Inference Example (cont'd)

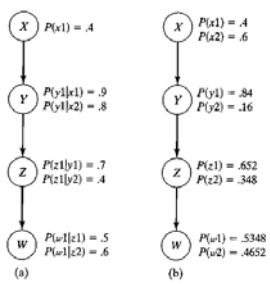


Instantiating X to $x1$:

$$\begin{aligned} P(w1 \mid x1) &= P(w1 \mid z1, x1)P(z1 \mid x1) + P(w1 \mid z2, x1)P(z2 \mid x1) \\ &= P(w1 \mid z1)P(z1 \mid x1) + P(w1 \mid z2)P(z2 \mid x1) \\ &= (0.5)(0.67) + (0.6)(0.33) = 0.533 \end{aligned}$$

Can think of passing messages down the chain

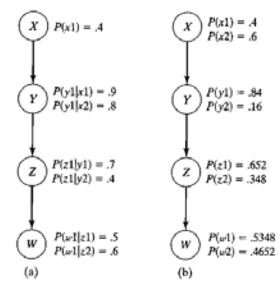
Another Inference Example



Now, instead instantiate W to $w1$:

$$P(z1 \mid w1) = \frac{P(w1 \mid z1)P(z1)}{P(w1)} = \frac{(0.5)(0.652)}{0.5348} = 0.6096$$

Another Inference Example (cont'd)



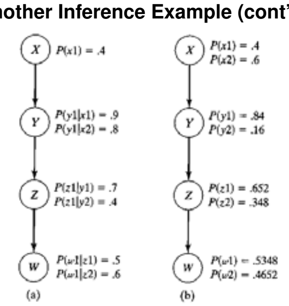
Still instantiating W to $w1$:

$$P(y1 \mid w1) = \frac{P(w1 \mid y1)P(y1)}{P(w1)} = \frac{(0.53)(0.84)}{0.5348} = 0.832$$

where

$$\begin{aligned} P(w1 \mid y1) &= P(w1 \mid z1)P(z1 \mid y1) + P(w1 \mid z2)P(z2 \mid y1) \\ &= (0.5)(0.7) + (0.6)(0.3) = 0.53 \end{aligned}$$

Another Inference Example (cont'd)



Still instantiating W to $w1$:

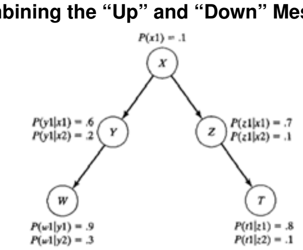
$$P(x1 \mid w1) = \frac{P(w1 \mid x1)P(x1)}{P(w1)}$$

where

$$P(w1 \mid x1) = P(w1 \mid y1)P(y1 \mid x1) + P(w1 \mid y2)P(y2 \mid x1)$$

Can think of passing messages up the chain

Combining the "Up" and "Down" Messages



- Instantiate W to $w1$
- Use upward propagation to get $P(y1 \mid w1)$ and $P(x1 \mid w1)$
- Then use downward propagation to get $P(z1 \mid w1)$ and then $P(t1 \mid w1)$

Pearl's Message Passing Algorithm

- Uses the message-passing principles just described
- Will have two kinds of messages
  - A λ message gets sent from a node to its parent (if it exists)
  - A π message gets sent from a node to its child (if it exists)
- At a node, the λ and π messages arriving from its children and parent are combined into λ and π values
- There is a set of messages and a value at X for each possible value x of X
  - E.g. in previous example, node X will get λ messages $\lambda_Y(x1)$, $\lambda_Y(x2)$, $\lambda_Z(x1)$, and $\lambda_Z(x2)$, and will compute λ values $\lambda(x1)$ and $\lambda(x2)$
  - Also in previous example, node Z will get π messages $\pi_Z(x1)$ and $\pi_Z(x2)$, and will compute π values $\pi(z1)$ and $\pi(z2)$

Pearl's Message Passing Algorithm (cont'd)

- What do the messages and values represent?
- Let $\mathcal{A} \subseteq \mathcal{V}$ be the set of variables instantiated and let a be the values of those variables (the evidence)
- Further, let a_X^+ be the evidence that can be accessed from X through its parent and a_X^- be the evidence that can be accessed from X through its children

Pearl's Message Passing Algorithm (cont'd)

- Then we'll define things such that

$$\lambda(x) = P(\mathrm{a}_X^- \mid x) \quad \text{and} \quad \pi(x) \propto P(x \mid \mathrm{a}_X^+)$$

- And this is all we need, since

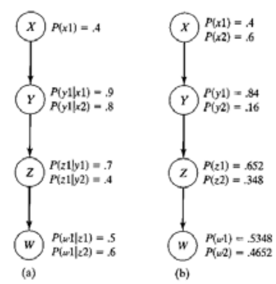
$$\begin{aligned}
P(x \mid \mathrm{a}) &= P(x \mid \mathrm{a}_X^+, \mathrm{a}_X^-) = \frac{P(\mathrm{a}_X^+, \mathrm{a}_X^- \mid x)P(x)}{P(\mathrm{a}_X^+, \mathrm{a}_X^-)} \\
&= \frac{P(\mathrm{a}_X^+ \mid x)P(\mathrm{a}_X^- \mid x)P(x)}{P(\mathrm{a}_X^+, \mathrm{a}_X^-)} = \frac{P(\mathrm{a}_X^+, x)P(\mathrm{a}_X^- \mid x)}{P(\mathrm{a}_X^+, \mathrm{a}_X^-)} \\
&= \frac{P(x \mid \mathrm{a}_X^+)P(\mathrm{a}_X^+)P(\mathrm{a}_X^- \mid x)}{P(\mathrm{a}_X^+, \mathrm{a}_X^-)} \\
&= \pi(x)\,\lambda(x)P(\mathrm{a}_X^+)/P(\mathrm{a}_X^+, \mathrm{a}_X^-)
\end{aligned}$$

(Why does the third equality hold?)

- Can ignore the constant terms until the end, then just renormalize

Pearl's Message Passing Algorithm

λ Messages



When we instantiated W to $w1$, we based calculation of $P(y1 \mid w1)$ on

$$\begin{aligned}
\lambda(y1) &= P(w1 \mid y1) = P(w1 \mid z1)P(z1 \mid y1) + P(w1 \mid z2)P(z2 \mid y1) \\
&= \sum_z P(w1 \mid z)P(z \mid y1) = \sum_z \lambda(z)P(z \mid y1)
\end{aligned}$$

Pearl's Message Passing Algorithm

λ Messages (cont'd)

- That's when Y has only one child
- What happens when a node has multiple children?
- Since we're conditioning on Y, all its children are d-separated:

$$\lambda(y1) = \prod_{U \in \mathrm{CH}(Y)} \left(\sum_u P(u \mid y1)\lambda(u) \right) ,$$

where CH(Y) is the set of children of Y (not necessarily binary)

- Thus the message that child Z sends to parent Y for value $y1$ is

$$\lambda_Z(y1) = \sum_z P(z \mid y1)\lambda(z)$$

and Y's λ value for $y1$ is

$$\lambda(y1) = \prod_{U \in \mathrm{CH}(Y)} \lambda_U(y1)$$

Pearl's Message Passing Algorithm

λ Messages (cont'd)

- Some special cases:
  - If a node X is instatiated to value $\hat{x}$, then $\lambda(\hat{x}) = 1$ and $\lambda(x) = 0$ for $x \neq \hat{x}$
  - If X is uninstantiated and is a leaf, then $\lambda(x) = 1$ for all x

Pearl's Message Passing Algorithm

π Messages

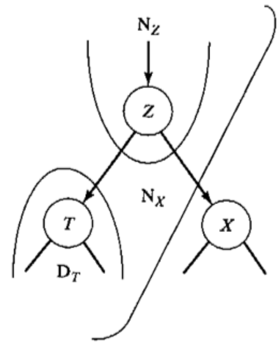
Now need to get

$$\pi(x) \;\propto\; P(x \mid \mathbf{a}_X^+) = \sum_z P(x \mid z) P(z \mid \mathbf{a}_X^+) \;,$$

where Z is X's parent

Pearl's Message Passing Algorithm

π Messages (cont'd)



Partition $\mathbf{a}_X^+$ into $\mathbf{a}_Z^+$ and $\mathbf{a}_T^-$, where T is X's sibling

Pearl's Message Passing Algorithm

π Messages (cont'd)

$$\begin{aligned}
\sum_z P(x \mid z) P(z \mid \mathbf{a}_X^+) &= \sum_z P(x \mid z) P(z \mid \mathbf{a}_Z^+, \mathbf{a}_T^-) \\
&= \sum_z P(x \mid z) \frac{P(\mathbf{a}_Z^+, \mathbf{a}_T^- \mid z) P(z)}{P(\mathbf{a}_Z^+, \mathbf{a}_T^-)} \\
&= \sum_z P(x \mid z) \frac{P(\mathbf{a}_Z^+ \mid z) P(\mathbf{a}_T^- \mid z) P(z)}{P(\mathbf{a}_Z^+, \mathbf{a}_T^-)} \\
&= \sum_z P(x \mid z) \frac{P(z \mid \mathbf{a}_Z^+) P(\mathbf{a}_Z^+) P(\mathbf{a}_T^- \mid z) P(z)}{P(z) P(\mathbf{a}_Z^+, \mathbf{a}_T^-)} \\
&\propto \sum_z P(x \mid z) \pi(z) \lambda_T(z)
\end{aligned}$$

because

$$P(\mathbf{a}_T^- \mid z) \;=\; \sum_t P(t \mid z) P(\mathbf{a}_T^- \mid t) = \sum_t P(t \mid z) \lambda(t) = \lambda_T(z)$$

Pearl's Message Passing Algorithm

π Messages (cont'd)

We've now established

$$P(x \mid \mathbf{a}_X^+) \propto \sum_z P(x \mid z) \pi(z) \lambda_T(z)$$

Thus we can define

$$\pi(x) = \sum_z P(x \mid z) \pi_X(z)$$

where

$$\pi_X(z) = \pi(z) \lambda_T(z)$$

Z is X's parent, T is X's sibling

What if the tree is not binary?

Pearl's Message Passing Algorithm

π Messages (cont'd)

- Some special cases:
  - If a node X is instatiated to value $\hat{x}$, then $\pi(\hat{x}) = 1$ and $\pi(x) = 0$ for $x \neq \hat{x}$
  - If X is uninstantiated and is the root, then $\mathbf{a}_X^+ = \emptyset$ and $\pi(x) = P(x)$ for all x

Pearl's Message Passing Algorithm

- Now we're ready to describe the algorithm
- In presentation of algorithms, will get as input a DAG $G = (\mathcal{V}, \mathcal{E})$ and distribution P (expressed as parameters in nodes)
- Will first initialize message variables for each node in G assuming nothing is instantiated
- Then will, one at a time, instantiate variables for which values are known
  - Add newly-instantiated variable to $\mathcal{A} \subseteq \mathcal{V}$
  - Pass messages as needed to update distribution
- Continue to assume that G is a binary tree

Pearl's Message Passing Algorithm
Initialization

- $\mathcal{A} = \mathbf{a} = \emptyset$
- For each $X \in \mathcal{V}$
  - For each value x of X: $\lambda(x) = 1$
  - For each value z of X's parent Z: $\lambda_X(z) = 1$
- For each value r of the root R: $\pi(r) = P(r \mid \mathbf{a}) = P(r)$
- For each child Y of R
  - R sends a π message to Y

Pearl's Message Passing Algorithm
Updating After Instantiating V to $\hat{v}$

- $\mathcal{A} = \mathcal{A} \cup \{V\}$, $\mathbf{a} = \mathbf{a} \cup \{\hat{v}\}$
- $\lambda(\hat{v}) = 1$, $\pi(\hat{v}) = 1$, $P(\hat{v} \mid \mathbf{a}) = 1$
- For each value $v \neq \hat{v}$: $\lambda(v) = 0$, $\pi(v) = 0$, $P(v \mid \mathbf{a}) = 0$
- If V is not root and V's parent $Z \notin \mathcal{A}$
  - V sends a λ message to Z
- For each child X of V such that $X \notin \mathcal{A}$
  - V sends a π message to X

Pearl's Message Passing Algorithm
Y sends a λ message to X

- For each value x of X:

$$\lambda_Y(x) = \sum_y P(y \mid x)\lambda(y)$$

$$\lambda(x) = \prod_{U \in \text{CH}(X)} \lambda_U(x)$$

$$P(x \mid \mathbf{a}) = \lambda(x)\pi(x)$$

- Normalize $P(x \mid \mathbf{a})$
- If X not root and X's parent $Z \notin \mathcal{A}$
  - X sends a λ message to Z
- For each child W of X such that $W \neq Y$ and $W \notin \mathcal{A}$
  - X sends a π message to W

Pearl's Message Passing Algorithm
Z sends a π message to X

- For each value z of Z:

$$\pi_X(z) = \pi(z) \prod_{Y \in \text{CH}(Z) \setminus \{X\}} \lambda_Y(z)$$

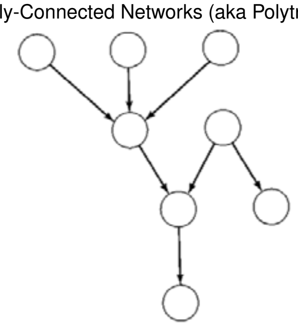
- For each value x of X:

$$\pi(x) = \sum_z P(x \mid z)\pi_X(z)$$

$$P(x \mid \mathbf{a}) = \lambda(x)\pi(x)$$

- Normalize $P(x \mid \mathbf{a})$
- For each child Y of X such that $Y \notin \mathcal{A}$
  - X sends a π message to Y

Pearl's Message Passing Algorithm
Singly-Connected Networks (aka Polytrees)



- Can generalize algorithm to singly-connected networks, where there is at most one path between any pair of nodes (i.e. trees where nodes can have multiple parents)

Pearl's Message Passing Algorithm
Singly-Connected Networks: π Values

- Need $\pi(x) \propto P(x \mid \mathbf{a}_X^+)$, where $\mathbf{a}_X^+$ defined over parents $Z_1, \ldots, Z_j$
- Since X depends on all j of its parents, need to sum over all combinations of values of $Z_1, \ldots, Z_j$:

$$\pi(x) = \sum_{z_1, \ldots, z_j} \left(P(x \mid z_1, \ldots, z_j) \prod_{i=1}^{j} \pi_X(z_i) \right)$$

- Sum over combinations for $P(x \mid z_1, \ldots, z_j)$ since x not independent of its parents
- Multiply over $\pi_X(z_i)$ since parents independent of each other when x uninstantiated
- π messages are the same as for trees

Pearl's Message Passing Algorithm
Singly-Connected Networks: λ Messages

- In computing Y's λ message to one of its parents X, now need to account for its other parents as well

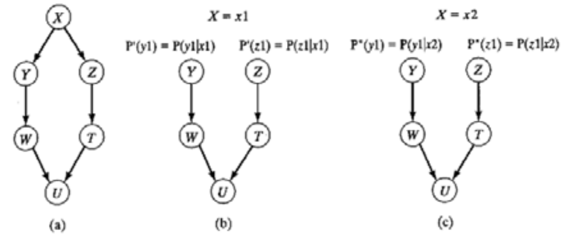
- Let Y be X's child, and $W_1, \ldots, W_k$ be Y's other parents:

$$\lambda_Y(x) = \sum_y \left[\sum_{w_1,\ldots,w_k} \left(P(y \mid x, w_1, \ldots, w_k) \prod_{i=1}^{k} \pi_Y(w_i) \right) \right] \lambda(y)$$

- Sum over combinations for $P(y \mid x, w_1, \ldots, w_j)$ since y not independent of its parents
- Multiply over $\pi_Y(w_i)$ since parents independent of each other when y uninstantiated

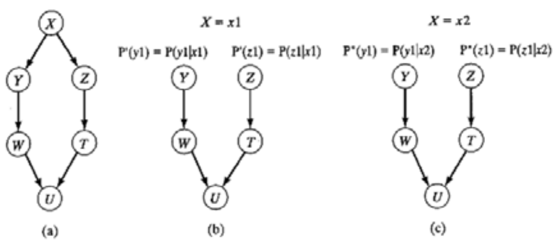
- λ values are the same as for trees

Pearl's Message Passing Algorithm
Multiply-Connected Networks



- When a DAG is multiply-connected, cannot use algorithms already presented since messages may get passed indefinitely
- But can use conditioning on a node to turn a multiply-connected network into multiple singly-connected networks
- E.g. conditioning on X blocks the chain $Y - X - Z$

Pearl's Message Passing Algorithm
Multiply-Connected Networks (cont'd)

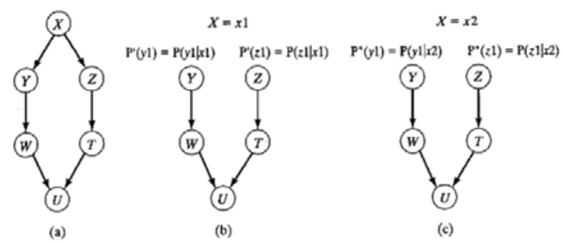


When U instantiated to $u1$,

$$P(w1 \mid u1) = P(w1 \mid x1, u1)P(x1 \mid u1) + P(w1 \mid x2, u1)P(x2 \mid u1)$$

where $P(w1 \mid xi, u1)$, $i \in \{1, 2\}$ come from running the old algorithm on (b) and (c) above, and $P(xi \mid u1) = P(u1 \mid xi)P(xi)/P(u1)$ (first term comes from algorithm, last from normalization)
Averaging results of the two assumptions on X

Pearl's Message Passing Algorithm
Multiply-Connected Networks (cont'd)



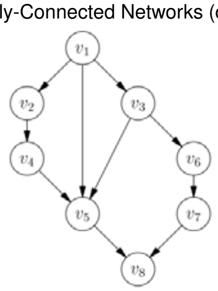
When U instantiated to $u1$ and Y to $y1$,

$$P(w1 \mid u1, y1) = P(w1 \mid x1, u1, y1)P(x1 \mid u1, y1) + P(w1 \mid x2, u1, y1)P(x2 \mid u1, y1)$$

where $P(w1 \mid xi, u1, y1)$ come from running old algorithm, and $P(xi \mid u1, y1) = P(u1, y1 \mid xi)P(xi)/P(u1, y1)$, where

$$P(u1, y1 \mid xi) = P(u1 \mid y1, xi)P(y1 \mid xi)$$

Pearl's Message Passing Algorithm
Multiply-Connected Networks (cont'd)



- A set of nodes $\mathcal{C} \subseteq \mathcal{V}$ is a loop cutset if for each (undirected) loop ℓ in the DAG there is a vertex from $v_i \in \mathcal{C}$ with an outgoing edge in ℓ
  - E.g. $\{v_1, v_7\}$ above, as well as $\{v_1, v_3\}$, etc., but not $\{v_5\}$

- NP-hard to find a minimally-sized $\mathcal{C}$

Pearl's Message Passing Algorithm
Multiply-Connected Networks (cont'd)

- If $\mathcal{C}$ is loop cutset, $\mathcal{E}$ is set of instantiated nodes, then for each node $X \in \mathcal{V} \setminus (\mathcal{E} \cup \mathcal{C})$,

$$P(xi) = \sum_{\mathbf{c}} P(xi \mid \mathbf{e}, \mathbf{c})P(\mathbf{c} \mid \mathbf{e})$$

(c goes over all combinations of values of nodes in $\mathcal{C}$)

- Get $P(xi \mid \mathbf{e}, \mathbf{c})$ from old algorithm

- Also, if $\mathbf{e} = \{e_1, \ldots, e_k\}$,

$$\begin{aligned} P(\mathbf{c} \mid \mathbf{e}) &\propto P(\mathbf{c})P(\mathbf{e} \mid \mathbf{c}) \\ &= P(\mathbf{c})P(e_k \mid \mathbf{c}, e_{k-1}, \ldots, e_1)P(e_{k-1} \mid \mathbf{c}, e_{k-2}, \ldots, e_1) \cdots P(e_1 \mid \mathbf{c}) \end{aligned}$$

  - Each term above comes from old algorithm

Pearl's Message Passing Algorithm

Multiply-Connected Networks (cont'd)

- $P(\mathbf{c})$ easily computed if all nodes in $\mathcal{C}$ are roots (how?)
- If not, then can compute by ordering $\mathcal{C}$'s nodes by predecessor relationship, instantiating them one at a time, and running old algorithm to pass messages [Suermondt & Cooper, 1991]
  - In running algorithm, block messages of all nodes in $\mathcal{C}$, even if not yet instantiated

Pearl's Message Passing Algorithm

Time Complexity

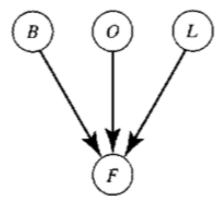
- Trees with n nodes, each with $\leq k$ values and $\leq c$ children:
  - Need k^2 steps to compute node Y's λ messages to its parent X, kc steps to compute node X's λ values, kc steps to compute Z's π messages to all children, and k^2 steps to compute X's π values
  - Repeat for each node $\Rightarrow O(n(k^2 + kc))$ total time
- Singly-connected networks with $\leq p$ parents/node:
  - Only changes were to π values ($k \cdot k^p \cdot p$ steps) and λ messages ($k \cdot k \cdot k^p \cdot p$ steps)
  - Can be big, but still polynomial in size of conditional prob. tables
- Multiply-connected networks with loop cut set $\mathcal{C}$: Run singly-connected algorithm $\Omega(k^{|\mathcal{C}|})$ times

Noisy OR-Gate Model

- An alternative (restricted) representation of probability distributions, reducing the computational and storage complexity
- Assumptions:
  - Each variable takes on two possible values
  - Causal Inhibition: There is a mechanism that inhibits a cause from bringing about its effect, and the cause's presence results in the effect's presence iff the mechanism is off
  - Exception Independence: Each cause's inhibitor is independent of the others
  - Accountability: An effect can occur iff at least one of its causes is present and uninhibited

Noisy OR-Gate Model

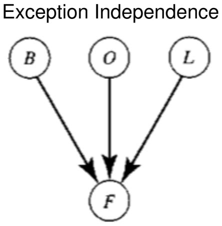
Causal Inhibition



- Bronchitis, Other, Lung Cancer, Fatigue
- Causal inhibition states that bronchitis results in fatigue iff its inhibitor is absent

Noisy OR-Gate Model

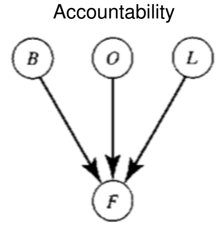
Exception Independence



- Bronchitis, Other, Lung Cancer, Fatigue
- Exception independence states that the mechanism inhibiting bronchitis from causing fatigue is independent of that which inhibits lung cancer from causing fatigue and that which inhibits other causes of fatigue

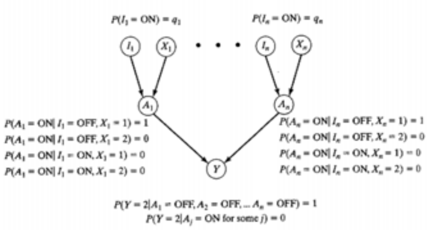
Noisy OR-Gate Model

Accountability



- Bronchitis, Other, Lung Cancer, Fatigue
- Accountability states that fatigue cannot be present unless one of bronchitis, lung cancer, or other is present and uninhibited

Noisy OR-Gate Model
Representing Assumptions as a Bayes Net



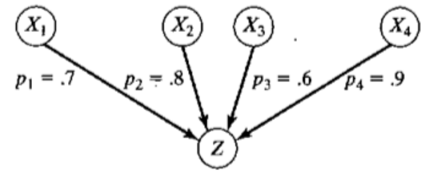
- Causes of Y are $X_1, \ldots, X_n$, cause X_j potentially inhibited by I_j

$\Rightarrow$ A_j is on iff X_j present and uninhibited by I_j

- It's a noisy OR gate since $Y = 1$ (= "ON") iff some $X_j = 1$ and its corresponding inhibitor I_j is OFF
- If $\mathcal{W} = \{X_1, \ldots, X_n\}$ with values $\mathrm{w} = \{x_1, \ldots, x_n\}$, then it's straightforward to see that

$$P(Y = 2 \mid \mathcal{W} = \mathrm{w}) = \prod_{j : x_j = 1} q_j$$

Noisy OR-Gate Model
Representing Assumptions as a Bayes Net (cont'd)



- The formula on the preceding slide allows us to simplify the representation, where $p_j = 1 - q_j$ is X_j's causal strength:

$$p_j = P(Y = 1 \mid X_j = 1, X_i = 2 \, \forall \, i \neq j)$$

- E.g.

$P(Y = 2 \mid X_1 = 1, X_2 = 2, X_3 = 1, X_4 = 1) = (1-p_1)(1-p_3)(1-p_4) = 0.012$

Noisy OR-Gate Model
Advantage of the Model

- This simplified model is more limiting than a general Bayes net, but has advantages
- E.g. to estimate causal strength of lung cancer for fatigue, need look only at fraction of lung cancer patients who are fatigued
  - In contrast, parameterizing more general Bayes net requires large numbers of patients with lung cancer and bronchitis, with lung cancer and no bronchitis, with no lung cancer and bronchitis, etc.
- Inference also simpler

Noisy OR-Gate Model
Inference: λ Messages

- Let node Y have parents $X_1, \ldots, X_n$, and $p_j = 1 - q_j$ be X_j's causal strength for Y
- Let x_j^+ denote that X_j is present, x_j^- denote absence
- Recall old formula for λ messages in singly-connected networks:

$$\lambda_Y(x_j) = \sum_y \left[\sum_{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n} \left(P(y \mid x_1, \ldots, x_n) \prod_{i \neq j} \pi_Y(x_i) \right) \right] \lambda(y)$$

- Can simplify this in noisy OR model:

$$\lambda_Y(x_j^+) = \lambda(y^-) q_j P_j + \lambda(y^+)(1 - q_j P_j)$$

$$\lambda_Y(x_j^-) = \lambda(y^-) P_j + \lambda(y^+)(1 - P_j)$$

$$P_j = \prod_{i \neq j} \left(1 - p_i \pi_Y(x_i^+)\right)$$

Noisy OR-Gate Model
Inference: π Values

- Recall old formula for π values in singly-connected networks:

$$\pi(y) = \sum_{x_1, \ldots, x_n} \left(P(y \mid x_1, \ldots, x_n) \prod_{j=1}^{n} \pi_Y(x_j) \right)$$

- Can simplify this in noisy OR model:

$$\pi(y^+) = 1 - \prod_{j=1}^{n} \left(1 - p_j \pi_Y(x_j^+)\right)$$

$$\pi(y^-) = \prod_{j=1}^{n} \left(1 - p_j \pi_Y(x_j^+)\right)$$