7-5

Review:

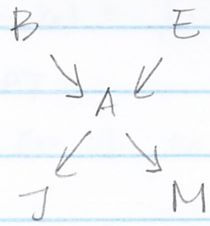
B = burglar

E = earthquake

A = alarm

J = John calls

M = Mary calls



* Belief network (BN)

  = directed acyclic graph (DAG)

  + conditional probability tables (CPTs)

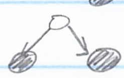
* conditional independence

$$P(X_1, X_2, \cdots, X_n) = P(X_1)\, P(X_2 | X_1) \cdots P(X_n | X_1, \cdots X_{n-1}) \quad \text{product rule}$$

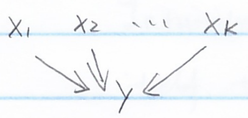
$$= \prod_{i=1}^{n} P(X_i | X_1, \cdots, X_{i-1})$$

$$= \prod_{i=1}^{n} P(X_i | pa(X_i)) \quad \text{where } pa(X_i) = \text{parents of } X_i \text{ is subset of } \{X_1, X_2, \cdots, X_{i-1}\}$$

* Types of reasoning

  1. competing explanations of observed event 
  2. multiple events with common explanation 
  3. intervening events 

* Reprensenting CPTs



Assume $X_i \in \{0, 1\}$

$Y \in \{0, 1\}$

How to represent CPT $P(Y=1 | X_1, X_2, \cdots X_k)$?

Option #1: look up table $O(2^k)$ can store arbitrary CPT

	X_1	X_2	$\cdots$	X_K	$P(Y=1 \mid X_1, X_2, \cdots X_K)$
2^k rows	0	0	$\cdots$	0	0.6
	1	0		0	0.3
	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	1	1	$\cdots$	1	0.8

But what if k is very large?

Option #2. "deterministic" node

"AND" $P(Y=1 \mid X_1, X_2, \cdots X_k) = \prod_{i=1}^{k} x_i$

"OR" $P(Y=0 \mid X_1, X_2, \cdots X_k) = \prod_{i=1}^{k} (1-x_i)$

Option #3. noisy-OR CPT

Use k numbers $p_i \in [0,1]$ to parameterize $O(2^k)$ elements of CPT:

$$P(Y=0 \mid X_1, X_2, \cdots, X_k) = \prod_{i=1}^{k} (1-p_i)^{x_i} \quad \text{where } x_i \in \{0,1\}$$

$$P(Y=1 \mid X_1, X_2, \cdots, X_k) = 1 - \prod_{i=1}^{k} (1-p_i)^{x_i}$$

Why call "noisy-OR"?

- Look at

$$P(Y=1 \mid \overbrace{X_1=0, X_2=0, \cdots, X_k=0}^{\text{all parents are off}}) = 1 - \prod_{i=1}^{k} (1-p_i)^0$$
$$= 1 - \prod_{i=1}^{k} (1)$$
$$= 0$$

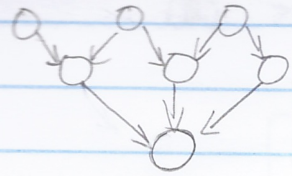
If $x_i = 0$ for all i, then $Y=0$. recovers behavior of "OR".

- Look at

$$P(Y=1 \mid X_1=0, X_2=0, \cdots X_{i-1}=0, \underbrace{X_i=1}_{\text{on}}, X_{i+1}=0, \cdots, X_k=0)$$
$$= 1 - \prod_{j=1}^{k} (1-p_j)^{x_j}$$
$$= 1 - (1-p_i)^1 \prod_{j \neq i} (1-p_j)^0$$
$$= 1 - (1-p_i)$$
$$= p_i$$

Intuitively, p_i represents the probability that $x_i = 1$ by itself triggers $Y=1$.

Conditional independence



A node is conditionally independent of its non-parent ancestors given its parents.

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

* More generally:

Let X, Y, and E refer to disjoint sets of nodes.

When is X conditionally independent of Y given evidence E?

When is $P(X \mid E, Y) = P(X \mid E)$?

$P(Y \mid E, X) = P(Y \mid E)$?

$P(X, Y \mid E) = P(X \mid E)\, P(Y \mid E)$?

We've done already the special case:

$X = \{X_i\}$

$E = \{pa(X_i)\}$

$Y = \underbrace{\{X_1, X_2, \cdots, X_{i-1}\}}_{\text{ancestors}} - \underbrace{\{pa(X_i)\}}_{\text{parents}}$

$$P(X \mid E) = P(X \mid E, Y)$$

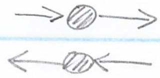
* d-separation

"direction-dependent" separation

Relates conditional independence to graph-theoretic properties.

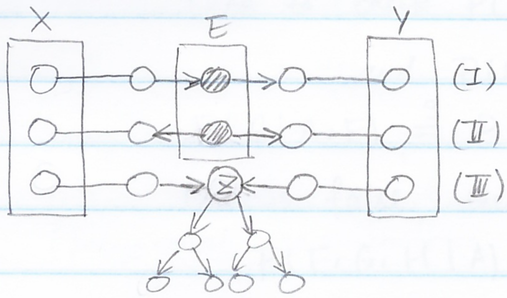
* Thm: $P(X, Y \mid E) = P(X \mid E)\, P(Y \mid E)$ if and only if every undirected path from a node in X to a node in Y is "d-separated" by E
(blocked)

* Def. a path π is d-separated if there exists a node $Z \in \pi$ for which one of 3 conditions hold:

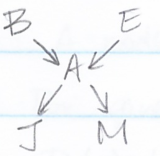
(I) $Z \in E$ with  is an "intervening" event in a causal chain.

(II) $Z \in E$ with $\leftarrow \bullet \rightarrow$ is a "common explanation"

(III) $Z \notin E$ descendants$(Z) \notin E$ with $\rightarrow Z \leftarrow$.
no observed "common effects"



* Proof that d-separation $\Longleftrightarrow$ conditional independence is beyond course
* Efficient algorithms exist for testing d-separation.
* Alarm BN example



True or False

(1) $P(B \mid A, M) \stackrel{?}{=} P(B \mid A)$

(condition I)

True – alarm is the intervening event

(2) $P(J, M \mid A) \stackrel{?}{=} P(J \mid A)\, P(M \mid A)$

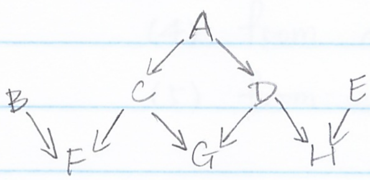
(condition II)

True – alarm is common explanation

(3) $P(B) \stackrel{?}{=} P(B \mid E)$

True – independent when there is no observed common effect. (condition III)

* Loopy BN example

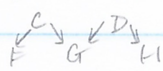


True or False

$P(D \mid H) \stackrel{?}{=} P(D \mid E, H)$ False by (III)

$P(F, H \mid A) \stackrel{?}{=} P(F \mid A)\, P(H \mid A)$ True

check path  true by (II)

check path  true by (III)

$$P(F, G, H \mid A) \overset{?}{=} P(F \mid A)\, P(G \mid A)\, P(H \mid A)$$

product rule $P(F \mid A)\, P(G \mid A, F)\, P(H \mid A, F, G) \overset{?}{=} P(F \mid A)\, P(G \mid A)\, P(H \mid A)$

Need to check $P(G \mid A, F) \overset{?}{=} P(G \mid A)$

and $P(H \mid A, F, G) \overset{?}{=} P(H \mid A)$

$P(G \mid A, F) \overset{?}{=} P(G \mid A)$ $F \leftarrow C \rightarrow G$ False

One is false, then

$P(F, G, H \mid A) \overset{?}{=} P(F \mid A)\, P(G \mid A)\, P(H \mid A)$ False.

* Def: Markov blanket B_X of individual node X consists of parents of X, children of X, and spouses of X

↳ parents of children of X, not including X itself.

* Thm: A node X is conditionally independent of nodes outside B_X given nodes inside B_X

$$P(X \mid B_X, Y) = P(X \mid B_X) \text{ when } Y \notin \{B_X, X\}$$

Proof: For any node $Y \notin \{B_X, X\}$, the undirected path from Y to X must pass through B_X. There are five cases to consider:

(1) from parent of parent of X — case I
(2) from child of parent of X — case II
(3) from parent of spouse of X — case I
(4) from child of child of X — case I
(5) from child of spouse of X — case II

$\Big\}$ All paths are d-seperated

$\boxed{\text{Inference}}$

* Problem

  E = set of evidence nodes

  Q = set of query nodes

  How to compute posterior probabilities $P(Q|E)$?

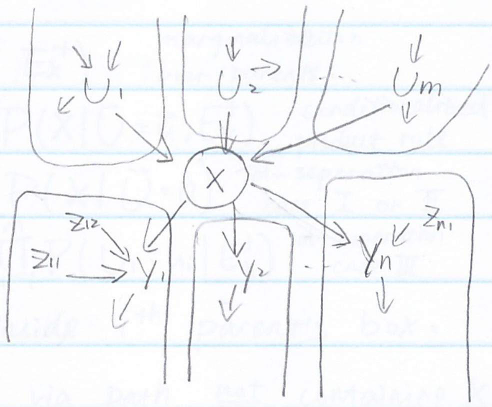
* Question: When can we perform this efficiently? (polynomial time in size of DAG and CPTs)?

Answer: Polytrees.

* Def: polytree = singly connected network; at most one undirected path between any two nodes, no loops.

Goal: Compute $P(X|E)$

Claim: Boxes don't overlap! No loops!



Polytree: Node X

Parents $U_1, U_2, \dots U_m$

Children $Y_1, Y_2, \dots Y_n$

Spouse: Z_{jk} (k^{th} parent of j^{th} child)

Type of evidence:

E_X^+ = evidence connected to X thru its parents.

E_X^- = evidence connected to X thru its children

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

Assume $X \notin E$, otherwise inference is trivial.

* Inference in polytree

$$P(X|E) = P(X|E_x^-, E_x^+)$$
$$= \frac{P(x, E_x^- | E_x^+)}{P(E_x^- | E_x^+)}$$ conditionalized product rule
$$= \frac{P(x, E_x^- | E_x^+)}{\sum_x P(X=x, E_x^- | E_x^+)}$$ conditionalized marginalization

Denominator is just numerator summed over X.

Focus on numerator.

* Numerator:

$$P(x, E_x^- | E_x^+) = P(X|E_x^+)\, P(E_x^- | X, E_x^+)$$ conditionalized product rule
$$= \underbrace{P(X|E_x^+)}_{\text{goal: "upstream recursion"}}\, \underbrace{P(E_x^- | X)}_{\text{"downstream recursion"}}$$ conditional independent (case I)

* "Upstream" recursion

$$P(X|E_x^+) = \sum_{\vec{u}} P(X, \vec{U}=\vec{u} | E_x^+)$$ marginalization over parents
$$= \sum_{\vec{u}} P(\vec{U}=\vec{u}|E_x^+)\, P(X|\vec{U}=\vec{u}, E_x^+)$$ conditionalized product rule
$$= \sum_{\vec{u}} P(\vec{U}=\vec{u}|E_x^+)\, P(X|\vec{U}=\vec{u})$$ d-seperation case I or II
$$= \sum_{\vec{u}} P(X|\vec{U}=\vec{u}) \prod_{i=1}^{m} P(U_i=u_i|E_x^+)$$ d-seperation case III

Let $E_{U_i \backslash X}$ denote evidence inside i^{th} parent's box:

evidence connected to U_i via path <u>not</u> containing X.

$$P(X|E_x^+) = \sum_{\vec{u}} P(x|\vec{U}=\vec{u}) \prod_{i=1}^{m} P(U_i=u_i|E_x^+)$$
$$= \sum_{\vec{u}} \underbrace{P(x|\vec{U}=\vec{u})}_{\text{CPT at node } X} \prod_{i=1}^{m} \underbrace{P(U_i=u_i|E_{U_i \backslash X})}_{\text{recursive instance of original problem}}$$ d-seperation case III

* "Downstream" recursion

How to compute $P(E_x^-|x)$?

Possible but somewhat more complicated.

* Termination conditions:

- root node (no parents)
- leaf node (no child)
- evidence node (trivial)

} algorithm terminates b/c polytree has no loops.

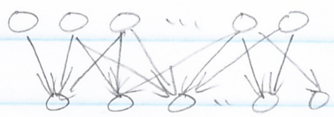
* Running Time

- linear in # nodes of network
- linear in size of CPTs (b/c we must sum over the parent values $\sum_{\vec{u}} P(x \mid \vec{U} = \vec{u}) \cdots$)

Loopy BNs – how to perform inference

Ex: Medical Diagnosis

2-layer network

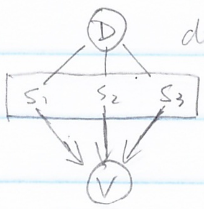


diseases

symptoms

Ex: Simpler Example

How to do inference?



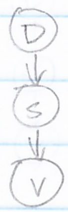
diseases?

symptoms (binary)

visit doctors?

Turn loopy BN into polytree by clustering nodes.

Merge nodes to form polytree.



- Merge S_1, S_2, S_3 into some mega-node S
- S takes on 2^3 possible values
- Merge CPTs $P(S_1 \mid D)$, $P(S_2 \mid D)$, $P(S_3 \mid D)$ into $P(S \mid D)$
- Apply polytree algorithm.

S_1	S_2	S_3	S
0	1	0	1
1	0	0	2
0	1	0	3
0	0	1	4
⋮	⋮	⋮	⋮

$P(S=2 \mid D)$

$= P(S_1=1, S_2=0, S_3=0 \mid D)$

$P(V \mid S=2) = P(V \mid S_1=1, S_2=0, S_3=0)$

* Polytree algorithm
  linear in size of CPTs

But CPT size grows exponentially with # nodes that are clustered.

How to $\underbrace{\text{choose optimal clustering}}_{\text{Computationally hard results}}$?