CSE 150 Lecture 6 1.25.10

HW #3 available on web site

Gradsource IDS emailed

$\boxed{\text{Review}}$

* d-separation

For sets of nodes X, Y, E:

when is $\left\{ \begin{array}{l} P(Y|E,X) = P(Y|E) \\ P(X|E,Y) = P(X|E) \\ P(X,Y|E) = P(X|E)P(Y|E) \end{array} \right\}$?

True if all paths from X to Y are "blocked."

A path is blocked if it has a node z that satisfies 1, 2, or 3:

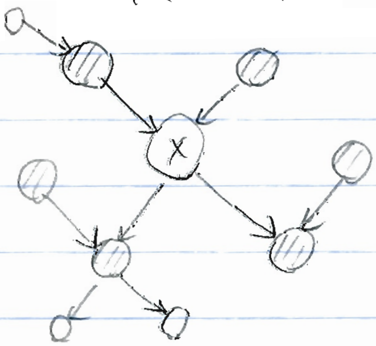
1) $z \in E$ $\quad \rightarrow (z) \rightarrow$ intervening cause

2) $z \in E$ $\quad \leftarrow (z) \rightarrow$ common cause

3) $z \notin E$, $desc(z) \notin E$ $\quad \rightarrow (z) \leftarrow$ no observed common effect

* Markov blanket B_x of node X consists of parents, children, and "spouses" (other parents of children of X)

Thm: $P(X|B_x, Y) = P(X|B_x)$ where $Y \notin \{X, B_x\}$



$\boxed{\text{Inference}}$

* Problem

  E = set of evidence nodes

  Q = set of query nodes

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

* Question: When can we perform inference 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 $P(X|E)$

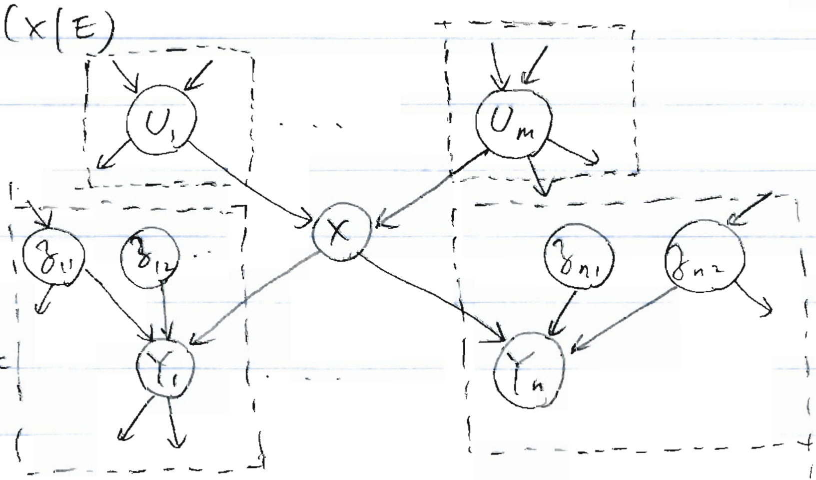
  Node X

  Evidence E

  Parents U_i

  Children Y_j

  parents of children Z_{jk}



Boxes don't overlap: no loops in polytrees.

Types of evidence:

E_X^+ = upstream evidence "above" X, connected to X thru parents.

E_X^- = downstream evidence "below" X, connected to X thru children.

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

assume $X \notin E$; otherwise inference is trivial.

* Inference in polytree

$$P(X|E) = P(X|E_X^-, E_X^+)$$
$$= \frac{P(X, E_X^- | E_X^+)}{P(E_X^- | E_X^+)} \quad \text{Conditionalized product rule.}$$
$$= \frac{P(X, E_X^- | E_X^+)}{\sum_x P(X=x, E_X^- | E_X^+)} \leftarrow \text{Conditional marginalization.}$$

Denominator is same computation as numerator, but summed over different values of X.

* So focus on numerator.

$$P(X, E_X^- | E_X^+) = P(X|E_X^+)\, P(E_X^- | X, E_X^+) \quad \text{conditionalized product rule}$$
$$= \underbrace{P(X|E_X^+)}_{\text{upstream recursion}}\, \underbrace{P(E_X^-|X)}_{\text{downstream recursion.}} \quad \text{d-separation, Case I.}$$

goal:

* "Upstream" recursion

$$P(X|E_X^+) = \sum_{\vec{u}} P(X, \vec{U} = \vec{u} | E_X^+) \quad \text{marginalization over parents.}$$
$$= \sum_{\vec{u}} P(\vec{U}=\vec{u} | E_X^+)\, P(X | \vec{U} = \vec{u}, E_X^+) \quad \text{Conditionalized product rule}$$
$$= \sum_{\vec{u}} P(\vec{U}=\vec{u} | E_X^+)\, P(X | \vec{U} = \vec{u}) \quad \text{d-separation Case I or II}$$
$$= \sum_{\vec{u}} P(X | \vec{U} = \vec{u}) \prod_{i=1}^{m} P(U_i = u_i | E_X^+) \quad \text{d-separation Case III}$$

(X is unobserved common effec

Let $E_{U_i \setminus X}$ denote evidence inside ith parent's box: evidence connected to U_i except via path through X

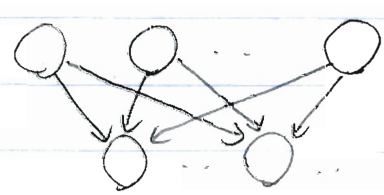
$$P(X|E_X^+) = \sum_{\vec{u}} P(X | \vec{U} = \vec{u}) \prod_{i=1}^{m} P(U_i = u_i | E_X^+)$$
$$= \sum_{\vec{u}} \underbrace{P(X | \vec{U} = \vec{u})}_{\text{CPT at node } X} \prod_{i=1}^{m} \underbrace{P(U_i = u_i | E_{U_i \setminus X})}_{\text{recursive instance of original problem.}} \quad \text{d-separation, Case III}$$

(X is unobserved common effect)

* Downstream recursion
  How to compute $P(E_{\bar{x}}|x)$?
  Possible but slightly more complicated.
* Termination conditions
  - root node (no parents)
  - leaf node (no children)
  - evidence node (trivial)
* Running time:
  linear in # node
  linear in size of CPTs
  (b/c you must sum over parent values $\sum_{\vec{u}} \{ P(x|\vec{U}=\vec{u}) \ldots \}$

$\boxed{\text{* Loopy BNs - how to perform inference?}}$

Ex: medical diagnosis.
2-layer network

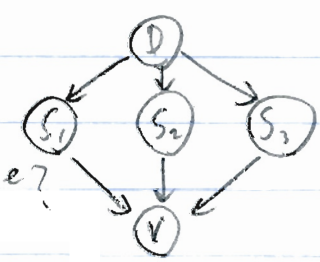


diseases

symptoms

Ex: Simple example

How to do exact inference?



disease

Symptoms (binary)

Visit doctor?

Turn loopy BN into polytree
One approach: node clustering

merge nodes to form polytree
merge S_1, S_2, S_3 into mega-node S
merge CPTs: $P(S_1|D), P(S_2|D), P(S_3|D)$ into $P(S|D)$

S takes on 2^3 values

Apply polytree algorithm: size of mega-node: 2^3
size of mega-CPT: 2^4

* Polytree algorithm. linear in CPT size,
  but CPT size grows exponentially with clustering.
  How to choose optimal clustering?
    Computationally hard problem!