

CSE 250A. Principles of AI

Probabilistic Reasoning and Decision-Making

Lecture 5 – Inference in polytrees

Lawrence Saul
Department of Computer Science and Engineering
University of California, San Diego

Fall 2021

Outline

- 1 Review
- 2 Polytrees
- 3 Inference



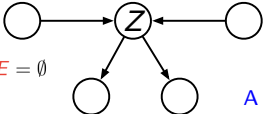
d-separation and conditional independence

- Theorem**

$P(X, Y|E) = P(X|E)P(Y|E)$ if and only if every *path* from a node in X to a node in Y is *blocked* by E .

- Definition**

A path π is **blocked** if there exists a node $Z \in \pi$ for which one of three conditions holds:

- (1) $Z \in E$  **causal chain**
- (2) $Z \in E$  **common cause**
- (3) $Z \notin E$  **unobserved common effect**
 $\text{descendants}(Z) \cap E = \emptyset$

A **descendant** of Z is *any* node (e.g., child, grandchild) that lies on a directed path from Z .

Inference

- **Problem**

Given a set E of evidence nodes, and a set Q of query nodes, how to compute the posterior distribution $P(Q|E)$?

- **More precisely**

How to express $P(Q|E)$ in terms of the CPTs $P(X_i|pa(X_i))$ of the BN, which are assumed to be given?

- **Tools at our disposal**

Bayes rule
marginalization
product rule

} and their conditionalized versions

marginal independence
conditional independence

} that we can test via d-separation

Strategy to compute $P(Q|E)$

Bayes rule

Use to express $P(Q|E)$ in terms of conditional probabilities that respect the order of the DAG.

marginalization

Use to introduce nodes on the left side of the conditioning bar when they need to appear as parents.

product rule

Use to express joint predictions (over multiple variables) in terms of simpler individual predictions.

**marginal and
conditional
independence**

Use to remove non-informative variables from the right side of the conditioning bar.

Questions for today

[Q1] In what type(s) of BNs is exact inference tractable?

[Q2] What makes exact inference tractable in these BNs?

[Q3] How does it scale with the size of the DAG and CPTs?

Questions?

Outline

① Review

② **Polytrees**

③ Inference

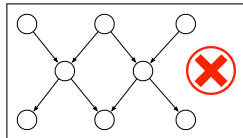
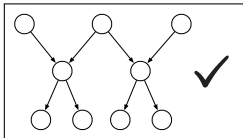
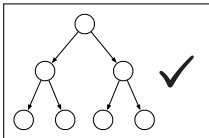
Polytrees

- **Definition**

A **polytree** is a singly connected belief network:
between any two nodes there is at most one path.

Alternatively, a polytree is a belief network without
any loops (i.e., undirected cycles).

- **Examples**



All trees are polytrees.

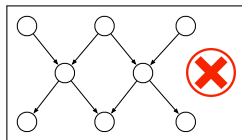
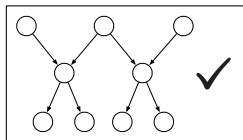
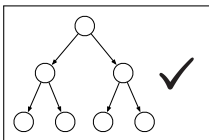
Polytrees

- **Definition**

A **polytree** is a singly connected belief network:
between any two nodes there is at most one path.

Alternatively, a polytree is a belief network without
any loops (i.e., undirected cycles).

- **Examples**



All trees are polytrees.

But not vice versa!

A node in a polytree may
have multiple parents.

Setup

- **Notation**

Let X be a node in a polytree.

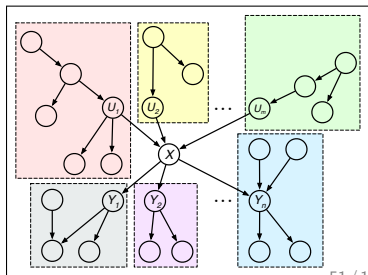
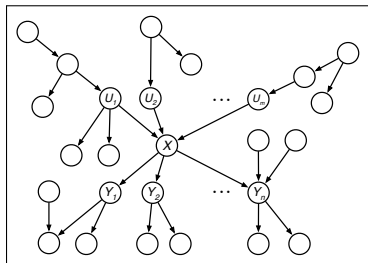
Let U_1, \dots, U_m denote its parents.

Let Y_1, \dots, Y_n denote its children.

- **Subgraphs**

Draw a box around each parent U_i and all nodes connected to it **not** via X .

Draw a box around each child Y_j and all nodes connected to it **not** via X .

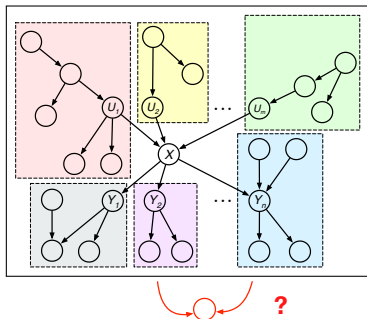


Key property

- **Claim**

In any polytree, these subgraphs have no common nodes.

Put another way, these boxes do not overlap.



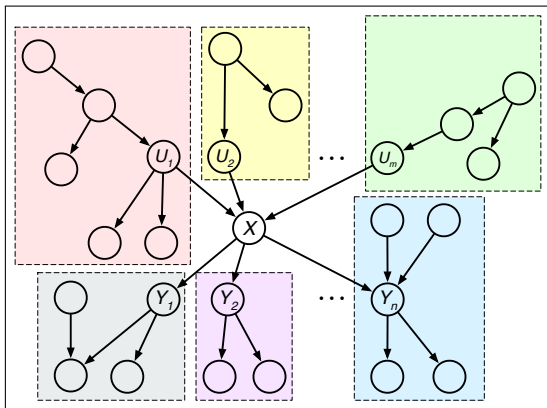
- **Proof**

The parents and children of X are connected via X .

A common node in different boxes would create a **loop**.

But there are no loops in polytrees.

Divide and conquer



Of course each subgraph is also a polytree in its own right. This suggests a **recursive** strategy for efficient inference ...

Questions?

Outline

① Review

② Polytrees

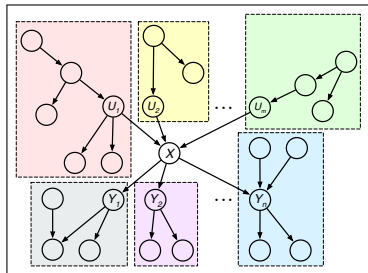
③ **Inference**

Inference in polytrees

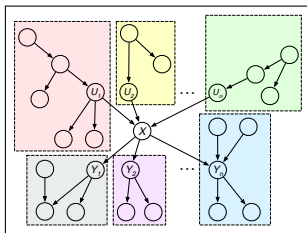
Let X denote a node in a polytree with parents U_1, U_2, \dots, U_m and children Y_1, Y_2, \dots, Y_n .

Let E denote a set of evidence nodes. Assume $X \notin E$.

How to compute $P(X=x|E)$?



Types of evidence



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

- **Evidence from above**

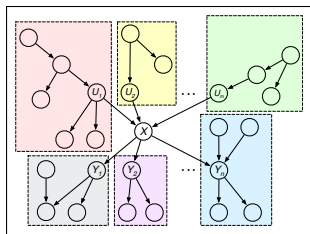
Let E_X^+ denote the evidence nodes connected to X through its parents (in the pink/yellow/green boxes).

- **Evidence from below**

Let E_X^- denote the evidence nodes connected to X through its children (in the gray/purple/blue boxes).

Inference — first steps

$$\begin{aligned}
 P(X=x|E) &= P(X=x|E_X^-, E_X^+) \\
 &= \frac{P(E_X^-|X=x, E_X^+) P(X=x|E_X^+)}{P(E_X^-|E_X^+)} \\
 &= \frac{P(E_X^-|X=x) P(X=x|E_X^+)}{P(E_X^-|E_X^+)}
 \end{aligned}$$



Bayes rule

conditional independence

(Node X blocks all paths from E_X^+ to E_X^- by case 1 of d-separation.)

Plan of attack

$$P(X=x|E) = \frac{P(E_X^-|X=x) P(X=x|E_X^+)}{P(E_X^-|E_X^+)}$$

- Numerator**

The **first term** only involves the subgraphs below X .

The **second term** only involves the subgraphs above X .

- Denominator**

If we know how to compute the numerator (for any x), then the denominator follows easily:

$$\sum_x P(X=x|E) = 1 \quad \boxed{\text{normalized}}$$

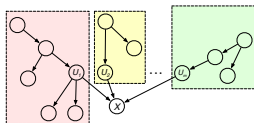
$$\implies \underbrace{P(E_X^-|E_X^+)}_{\text{denominator}} = \sum_x \underbrace{P(E_X^-|X=x) P(X=x|E_X^+)}_{\text{numerator}}$$

Questions?

Recursion through parents of X

Let \vec{U} denote the parents (U_1, U_2, \dots, U_m) of X .

Let \vec{u} denote values (u_1, u_2, \dots, u_m) for \vec{U} .



$$P(X=x|E_X^+)$$

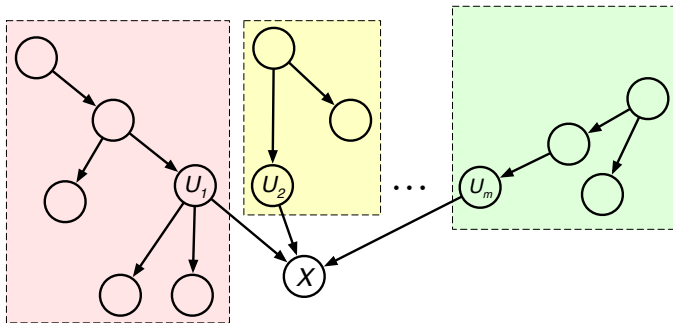
$$= \sum_{\vec{u}} P(X=x, \vec{U}=\vec{u} | E_X^+) \quad \text{marginalization}$$

$$= \sum_{\vec{u}} P(\vec{U}=\vec{u} | E_X^+) P(X=x | \vec{U}=\vec{u}, E_X^+) \quad \text{product rule}$$

$$= \sum_{\vec{u}} P(\vec{U}=\vec{u} | E_X^+) \underbrace{P(X=x | \vec{U}=\vec{u})}_{\text{CPT at node } X} \quad \text{conditional independence}$$

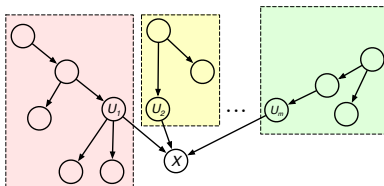
Boxing in the evidence

Let $E_{U_i \setminus X} \subset E_X^+$ denote the evidence in the i^{th} parent's box; it is the evidence connected to U_i , but not through X .



$$E_X^+ = E_{U_1 \setminus X} \cup E_{U_2 \setminus X} \cup \dots \cup E_{U_m \setminus X}$$

Completing the recursion

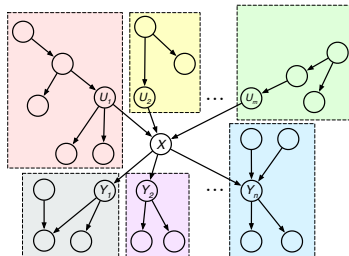


If node X and its descendants are not observed, then X blocks every path between different boxes by case 3 of d-separation.

$$\begin{aligned}
 P(\vec{U} = \vec{u} | E_X^+) &= \prod_{i=1}^m P(U_i = u_i | E_X^+) && \boxed{\text{conditional independence}} \\
 &= \prod_{i=1}^m \underbrace{P(U_i = u_i | E_{U_i \setminus X})}_{\text{recursive instance of original problem!}} && \boxed{\text{conditional independence}}
 \end{aligned}$$

Recap

How to compute $P(X=x|E)$?



- Terms from subgraphs

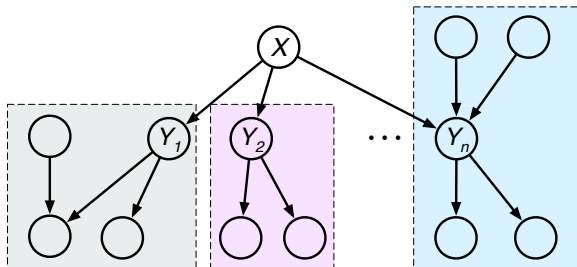
$$P(X=x|E) \propto \underbrace{P(X=x|E_X^+)}_{\text{from above}} \underbrace{P(E_X^-|X=x)}_{\text{from below}}$$

- Recursion through parents

$$P(X=x|E_X^+) = \sum_{\vec{u}} \underbrace{P(X=x|\vec{U}=\vec{u})}_{\text{CPT (given)}} \prod_{i=1}^m \underbrace{P(U_i=u_i|E_{U_i \setminus X})}_{\text{recursive instance}}$$

Questions?

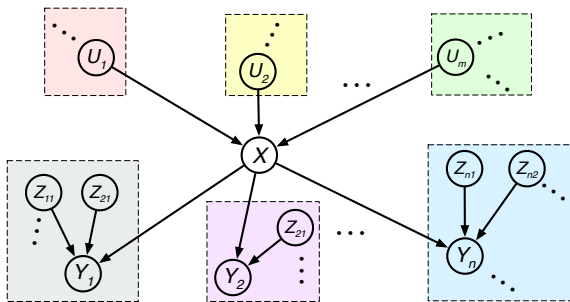
Recursion through children



$$P(E_X^- | X=x) = \prod_{j=1}^n P(E_{Y_j \setminus X} | X=x) \quad \boxed{\text{conditional independence}}$$

When node X is observed, it blocks every path between different boxes by case 2 of d-separation.

More terminology and notation

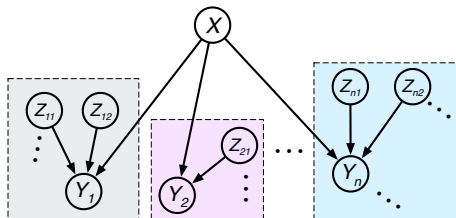


Let Z_{jk} denote the k^{th} parent of Y_j (excluding X itself).
The nodes Z_{jk} are sometimes called the **spouses** of X .

Let \vec{Z}_j denote the other parents of Y_j (excluding X).

Let \vec{z}_j denote a vector of instantiated values for these nodes.

Completing the recursion



$$P(E_X^- | X=x) = \prod_{j=1}^n P(E_{Y_j \setminus X} | X=x)$$

Stated without proof:

$$P(E_{Y_j \setminus X} | X=x) \propto \sum_{y_j} \underbrace{P(E_{Y_j}^- | Y_j=y_j)}_{\text{recursive instance}} \sum_{\vec{z}_j} \underbrace{P(y_j | \vec{z}_j, x)}_{\text{CPT(given)}} \prod_k \underbrace{P(z_{jk} | E_{Z_{jk} \setminus Y_j})}_{\text{recursive instance}}$$

Now we just need to make sure these recursions terminate ...

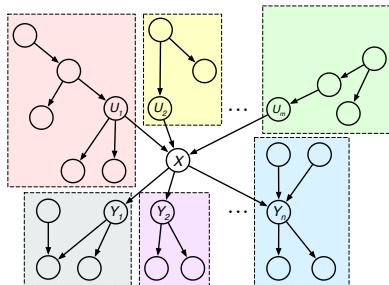
Termination conditions for polytree algorithm

Where does it end?

- 1 **Root nodes** (no parents)
- 2 **Leaf nodes** (no children)
- 3 **Evidence nodes**

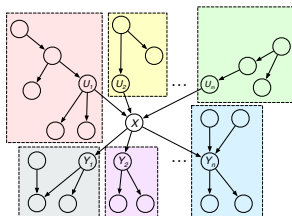
To begin we assumed that $X \notin E$.

Otherwise the inference $P(X=x|E)$ was trivial.



A polytree has no loops, so the process cannot cycle.

Runtime analysis



- **Before terminating:**

The algorithm may visit each node in the BN.

The algorithm may sum over the CPT at each node.

- **Thus the runtime is:**

Linear in the number of nodes of the BN.

Linear in the size of the CPTs.

Key takeaways

- **Relax**

You will never be asked (in this class) to derive or implement the polytree algorithm in its full generality.

- **So why is this algorithm important?**

It showcases the usefulness of d-separation.

It is the basis of many algorithms for inference in BNs.

- **What you need to understand**

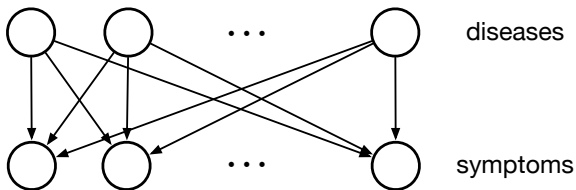
[HW 2] Enough for guided calculations in small BNs.

[HW 7] Enough for full implementations in simpler BNs.

Questions?

Next lecture ...

Many interesting BNs are not polytrees.



What are strategies for inference in these BNs?