Bayesian Networks: Independencies and Inference

Scott Davies and Andrew Moore

What Independencies does a Bayes Net Model?

- In order for a Bayesian network to model a probability distribution, the following must be true by definition:

  Each variable is conditionally independent of all its non-descendants in the graph given the value of all its parents.

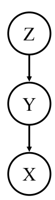
- This implies

$$P(X_1 \ldots X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$$

- But what else does it imply?

What Independencies does a Bayes Net Model?

- Example:



Given *Y*, does learning the value of *Z* tell us nothing new about *X*?

I.e., is $P(X|Y, Z)$ equal to $P(X \mid Y)$?

Yes. Since we know the value of all of *X*'s parents (namely, *Y*), and *Z* is not a descendant of *X*, *X* is conditionally independent of *Z*.

Also, since independence is symmetric, $P(Z|Y, X) = P(Z|Y)$.

Quick proof that independence is symmetric

- Assume: $P(X|Y, Z) = P(X|Y)$
- Then:

$$P(Z \mid X,Y) = \frac{P(X,Y \mid Z)P(Z)}{P(X,Y)} \quad \text{(Bayes's Rule)}$$

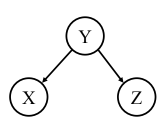
$$= \frac{P(Y \mid Z)P(X \mid Y,Z)P(Z)}{P(X \mid Y)P(Y)} \quad \text{(Chain Rule)}$$

$$= \frac{P(Y \mid Z)P(X \mid Y)P(Z)}{P(X \mid Y)P(Y)} \quad \text{(By Assumption)}$$

$$= \frac{P(Y \mid Z)P(Z)}{P(Y)} = P(Z \mid Y) \quad \text{(Bayes's Rule)}$$

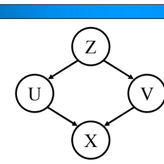
What Independencies does a Bayes Net Model?

- Let $I\langle X,Y,Z \rangle$ represent *X* and *Z* being conditionally independent given *Y*.



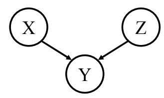
- $I\langle X,Y,Z \rangle$? Yes, just as in previous example: All X's parents given, and Z is not a descendant.

What Independencies does a Bayes Net Model?



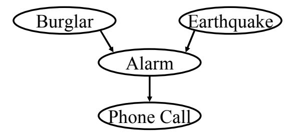
- $I\langle X,\{U\},Z \rangle$? No.
- $I\langle X,\{U,V\},Z \rangle$? Yes.
- Maybe $I\langle X, S, Z \rangle$ iff *S* acts a cutset between *X* and *Z* in an undirected version of the graph…?

Things get a little more confusing



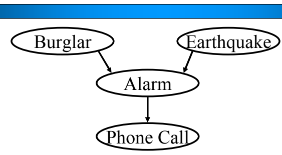
- X has no parents, so we know all its parents' values trivially
- Z is not a descendant of X
- So, *I*<*X*,{},*Z*>, even though there's a undirected path from *X* to Z through an unknown variable *Y*.
- What if we do know the value of *Y*, though? Or one of its descendants?

The "Burglar Alarm" example



- Your house has a twitchy burglar alarm that is also sometimes triggered by earthquakes.
- Earth arguably doesn't care whether your house is currently being burgled
- While you are on vacation, one of your neighbors calls and tells you your home's burglar alarm is ringing. Uh oh!

Things get a lot more confusing



- But now suppose you learn that there was a medium-sized earthquake in your neighborhood. Oh, whew! Probably not a burglar after all.
- Earthquake "explains away" the hypothetical burglar.
- But then it must **not** be the case that

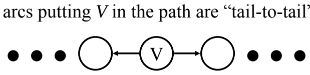
I<Burglar,{Phone Call}, Earthquake>, even though
I<Burglar,{}, Earthquake>!

d-separation to the rescue

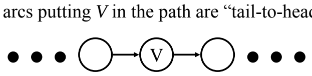
- Fortunately, there is a relatively simple algorithm for determining whether two variables in a Bayesian network are conditionally independent: *d-separation*.
- Definition: *X* and *Z* are *d-separated* by a set of evidence variables *E* iff every undirected path from *X* to *Z* is "blocked", where a path is "blocked" iff one or more of the following conditions is true: ...

A path is "blocked" when...

- There exists a variable *V* on the path such that
  - it **is** in the evidence set *E*
  - the arcs putting *V* in the path are "tail-to-tail"



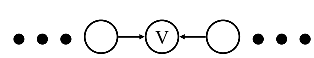
- Or, there exists a variable *V* on the path such that
  - it **is** in the evidence set *E*
  - the arcs putting *V* in the path are "tail-to-head"



- Or, ...

A path is "blocked" when… (the funky case)

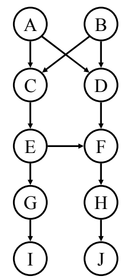
- … Or, there exists a variable *V* on the path such that
  - it **is NOT** in the evidence set *E*
  - **neither are any of its descendants**
  - the arcs putting *V* on the path are "head-to-head"



d-separation to the rescue, cont'd

- Theorem [Verma & Pearl, 1998]:
  - If a set of evidence variables E d-separates X and Z in a Bayesian network's graph, then $I\langle X, E, Z\rangle$.
- *d*-separation can be computed in linear time using a depth-first-search-like algorithm.
- Great! We now have a fast algorithm for automatically inferring whether learning the value of one variable might give us any additional hints about some other variable, given what we already know.
  - "Might": Variables may actually be independent when they're not d-separated, depending on the actual probabilities involved

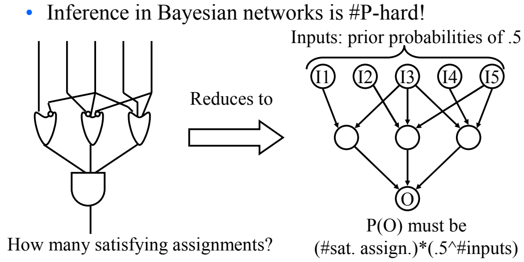
d-separation example



- I<C, {}, D>?
- I<C, {A}, D>?
- I<C, {A, B}, D>?
- I<C, {A, B, J}, D>?
- I<C, {A, B, E, J}, D>?

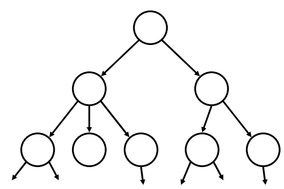
Bayesian Network Inference

- Inference: calculating $P(X|Y)$ for some variables or sets of variables X and Y.
- Inference in Bayesian networks is #P-hard!



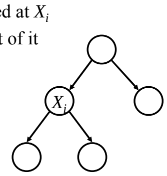
Bayesian Network Inference

- **But**…inference is still tractable in some cases.
- Let's look a special class of networks: *trees / forests* in which each node has at most one parent.



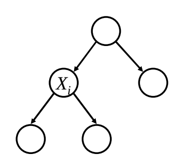
Decomposing the probabilities

- Suppose we want $P(X_i \mid E)$ where E is some set of evidence variables.
- Let's split E into two parts:
  - E_i^- is the part consisting of assignments to variables in the subtree rooted at X_i
  - E_i^+ is the rest of it



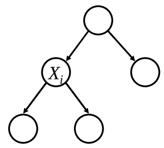
Decomposing the probabilities, cont'd

$$P(X_i \mid E) = P(X_i \mid E_i^-, E_i^+)$$



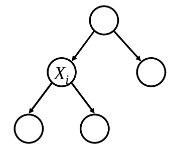
Decomposing the probabilities, cont'd

$$P(X_i \mid E) = P(X_i \mid E_i^-, E_i^+)$$
$$= \frac{P(E_i^- \mid X, E_i^+) P(X \mid E_i^+)}{P(E_i^- \mid E_i^+)}$$



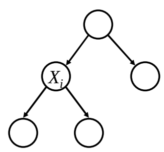
Decomposing the probabilities, cont'd

$$P(X_i \mid E) = P(X_i \mid E_i^-, E_i^+)$$
$$= \frac{P(E_i^- \mid X, E_i^+) P(X \mid E_i^+)}{P(E_i^- \mid E_i^+)}$$
$$= \frac{P(E_i^- \mid X) P(X \mid E_i^+)}{P(E_i^- \mid E_i^+)}$$



Decomposing the probabilities, cont'd

$$P(X_i \mid E) = P(X_i \mid E_i^-, E_i^+)$$
$$= \frac{P(E_i^- \mid X, E_i^+) P(X \mid E_i^+)}{P(E_i^- \mid E_i^+)}$$
$$= \frac{P(E_i^- \mid X) P(X \mid E_i^+)}{P(E_i^- \mid E_i^+)}$$
$$= \alpha\pi(X_i)\lambda(X_i)$$



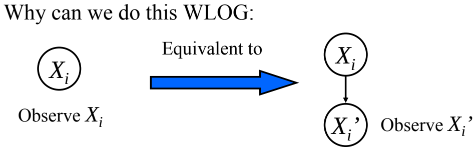
Where:
- α is a constant independent of X_i
- $\pi(X_i) = \mathrm{P}(X_i \mid E_i^+)$
- $\lambda(X_i) = \mathrm{P}(E_i^- \mid X_i)$

Using the decomposition for inference

- We can use this decomposition to do inference as follows. First, compute $\lambda(X_i) = \mathrm{P}(E_i^- \mid X_i)$ for all X_i recursively, using the leaves of the tree as the base case.
- If X_i is a leaf:
  - If X_i is in E: $\lambda(X_i) = 1$ if X_i matches E, 0 otherwise
  - If X_i is not in E: E_i^- is the null set, so $\mathrm{P}(E_i^- \mid X_i) = 1$ (constant)

Quick aside: "Virtual evidence"

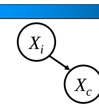
- For theoretical simplicity, but without loss of generality, let's assume that *all* variables in E (the evidence set) are leaves in the tree.
- Why can we do this WLOG:



Where $\mathrm{P}(X_i' \mid X_i) = 1$ if $X_i' = X_i$, 0 otherwise

Calculating $\lambda(X_i)$ for non-leaves

- Suppose X_i has one child, X_c.

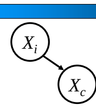


- Then:

$$\lambda(X_i) = P(E_i^- \mid X_i) =$$

Calculating $\lambda(X_i)$ for non-leaves

- Suppose X_i has one child, X_c.

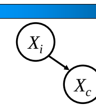


- Then:

$$\lambda(X_i) = P(E_i^- \mid X_i) = \sum_j P(E_i^-, X_C = j \mid X_i)$$

Calculating $\lambda(X_i)$ for non-leaves

- Suppose X_i has one child, X_c.



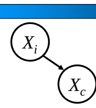
- Then:

$$\lambda(X_i) = P(E_i^- \mid X_i) = \sum_j P(E_i^-, X_C = j \mid X_i)$$

$$= \sum_j P(X_C = j \mid X_i) P(E_i^- \mid X_i, X_C = j)$$

Calculating $\lambda(X_i)$ for non-leaves

- Suppose X_i has one child, X_c.



- Then:

$$\lambda(X_i) = P(E_i^- \mid X_i) = \sum_j P(E_i^-, X_C = j \mid X_i)$$

$$= \sum_j P(X_C = j \mid X_i) P(E_i^- \mid X_i, X_C = j)$$

$$= \sum_j P(X_C = j \mid X_i) P(E_i^- \mid X_C = j)$$

$$= \sum_j P(X_C = j \mid X_i) \lambda(X_C = j)$$

Calculating $\lambda(X_i)$ for non-leaves

- Now, suppose X_i has a set of children, C.
- Since X_i *d-separates* each of its subtrees, the contribution of each subtree to $\lambda(X_i)$ is independent:

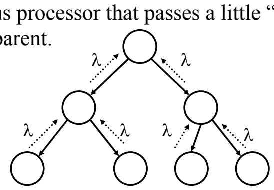
$$\lambda(X_i) = P(E_i^- \mid X_i) = \prod_{X_j \in C} \lambda_j(X_i)$$

$$= \prod_{X_j \in C} \left[\sum_{X_j} P(X_j \mid X_i) \lambda(X_j) \right]$$

where $\lambda_j(X_i)$ is the contribution to $P(E_i^- \mid X_i)$ of the part of the evidence lying in the subtree rooted at one of X_i's children X_j.

We are now λ-happy

- So now we have a way to recursively compute all the $\lambda(X_i)$'s, starting from the root and using the leaves as the base case.
- If we want, we can think of each node in the network as an autonomous processor that passes a little "λ message" to its parent.

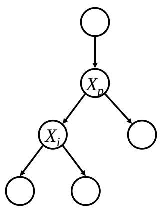


The other half of the problem

- Remember, $P(X_i|E) = \alpha\pi(X_i)\lambda(X_i)$. Now that we have all the $\lambda(X_i)$'s, what about the $\pi(X_i)$'s?

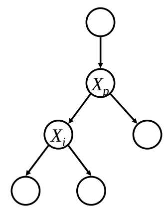
  $\pi(X_i) = P(X_i \mid E_i^+)$.
- What about the root of the tree, X_r? In that case, E_r^+ is the null set, so $\pi(X_r) = P(X_r)$. No sweat. Since we also know $\lambda(X_r)$, we can compute the final $P(X_r)$.
- So for an arbitrary X_i with parent X_p, let's inductively assume we know $\pi(X_p)$ and/or $P(X_p|E)$. How do we get $\pi(X_i)$?

Computing $\pi(X_i)$



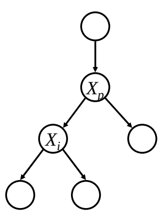
$$\pi(X_i) = P(X_i \mid E_i^+) =$$

Computing $\pi(X_i)$



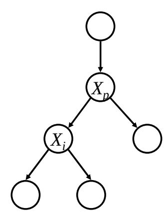
$$\pi(X_i) = P(X_i \mid E_i^+) = \sum_j P(X_i, X_p = j \mid E_i^+)$$

Computing $\pi(X_i)$



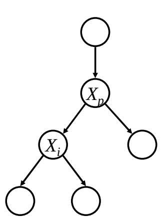
$$\pi(X_i) = P(X_i \mid E_i^+) = \sum_j P(X_i, X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j, E_i^+) P(X_p = j \mid E_i^+)$$

Computing $\pi(X_i)$



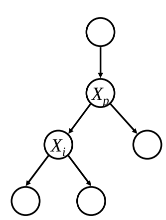
$$\pi(X_i) = P(X_i \mid E_i^+) = \sum_j P(X_i, X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j, E_i^+) P(X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j) P(X_p = j \mid E_i^+)$$

Computing $\pi(X_i)$



$$\pi(X_i) = P(X_i \mid E_i^+) = \sum_j P(X_i, X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j, E_i^+) P(X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j) P(X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j) \frac{P(X_p = j \mid E)}{\lambda_i(X_p = j)}$$

Computing $\pi(X_i)$

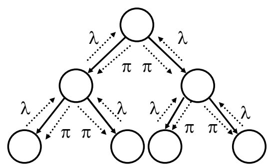


$$\pi(X_i) = P(X_i \mid E_i^+) = \sum_j P(X_i, X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j, E_i^+) P(X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j) P(X_p = j \mid E_i^+)$$
$$= \sum_j P(X_i \mid X_p = j) \frac{P(X_p = j \mid E)}{\lambda_i(X_p = j)}$$
$$= \sum_j P(X_i \mid X_p = j) \pi_i(X_p = j)$$

Where $\pi_i(X_p)$ is defined as $\dfrac{P(X_p \mid E)}{\lambda_i(X_p)}$

We're done. Yay!

- Thus we can compute all the $\pi(X_i)$'s, and, in turn, all the $P(X_i|E)$'s.
- Can think of nodes as autonomous processors passing λ and π messages to their neighbors

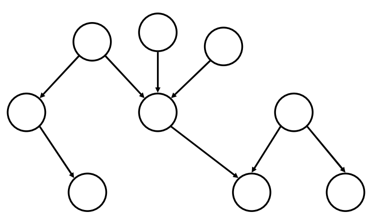


Conjunctive queries

- What if we want, e.g., P(A, B | C) instead of just marginal distributions P(A | C) and P(B | C)?
- Just use chain rule:
  - P(A, B | C) = P(A | C) P(B | A, C)
  - Each of the latter probabilities can be computed using the technique just discussed.

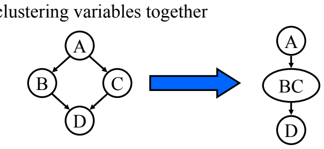
Polytrees

- Technique can be generalized to *polytrees*: undirected versions of the graphs are still trees, but nodes can have more than one parent

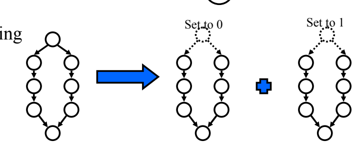


Dealing with cycles

- Can deal with undirected cycles in graph by
  - clustering variables together

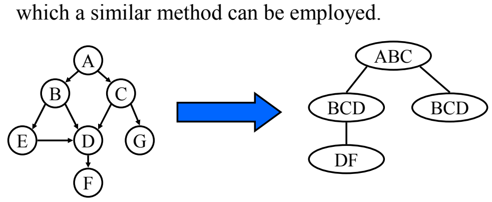


  - Conditioning



Join trees

- Arbitrary Bayesian network can be transformed via some evil graph-theoretic magic into a *join tree* in which a similar method can be employed.



In the worst case the join tree nodes must take on exponentially many combinations of values, but often works well in practice