

CS 188: Artificial Intelligence

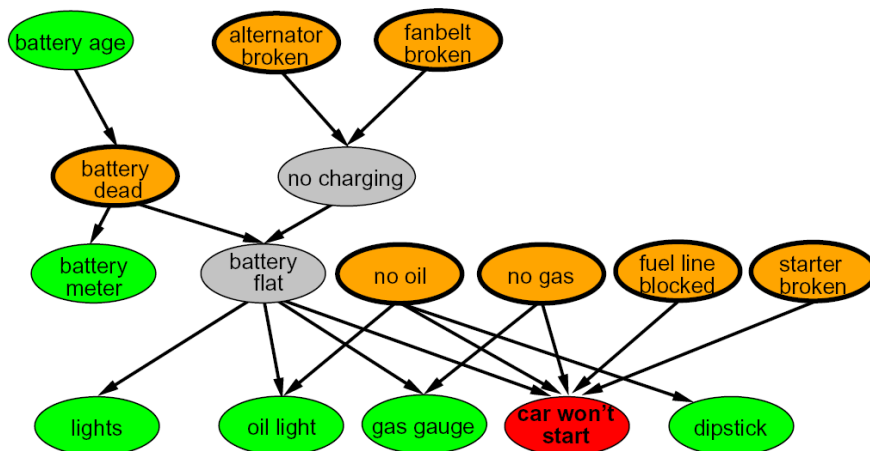
Fall 2007

Lecture 17: Bayes Nets III

10/25/2007

Dan Klein – UC Berkeley

Representing Knowledge

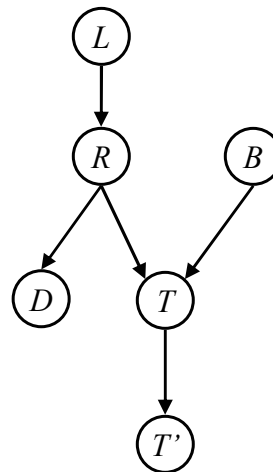


Properties of BNs

- **Bayes' nets:**
 - Specify complex joint distributions using simple local conditional distributions
 - Conditional independence makes this possible
 - The graph structure of a BN guarantees certain conditional independences
- **Questions:**
 - What independences does a BN have?
 - How to compute quantities we want from quantities we have

Independence?

- **Recipe: shade evidence nodes**
- **Attempt 1: if two nodes are connected by an undirected path not blocked by a shaded node, they are conditionally independent**
- **Almost works, but not quite**
 - Where does it break?
 - Answer: the v-structure at T doesn't count as a link in a path unless shaded



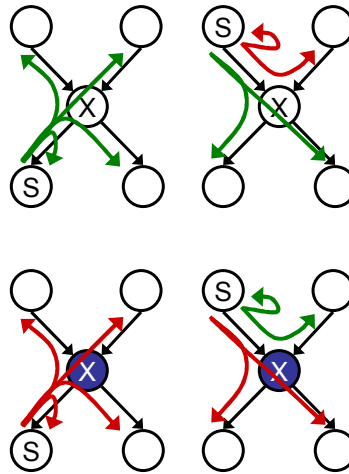
Reachability (the Bayes' Ball)

- **Correct algorithm:**
 - Shade in evidence
 - Start at source node
 - Try to reach target by search

- States: pair of (node X, previous state S)

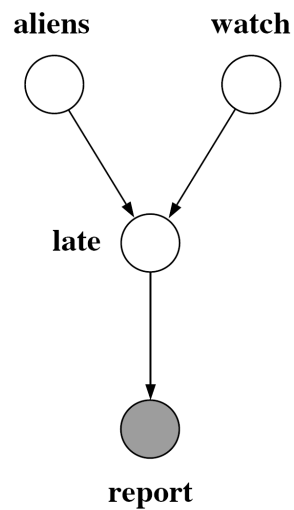
- **Successor function:**
 - X unobserved:
 - To any child
 - To any parent if coming from a child
 - X observed:
 - From parent to parent

- If you can't reach a node, it's conditionally independent of the start node given evidence



Example

$A \perp\!\!\!\perp W$ **Yes**
 $A \perp\!\!\!\perp W | R$



Example

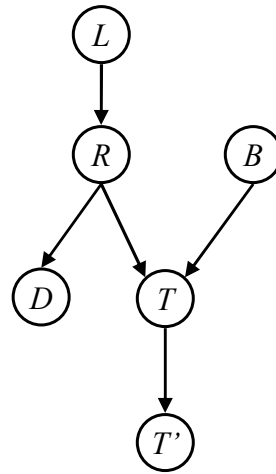
$L \perp\!\!\!\perp T' | T$ **Yes**

$L \perp\!\!\!\perp B$ **Yes**

$L \perp\!\!\!\perp B | T$

$L \perp\!\!\!\perp B | T'$

$L \perp\!\!\!\perp B | T, R$ **Yes**



Example

- Variables:

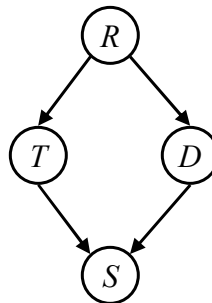
- R: Raining
- T: Traffic
- D: Roof drips
- S: I'm sad

- Questions:

$T \perp\!\!\!\perp D$

$T \perp\!\!\!\perp D | R$ **Yes**

$T \perp\!\!\!\perp D | R, S$

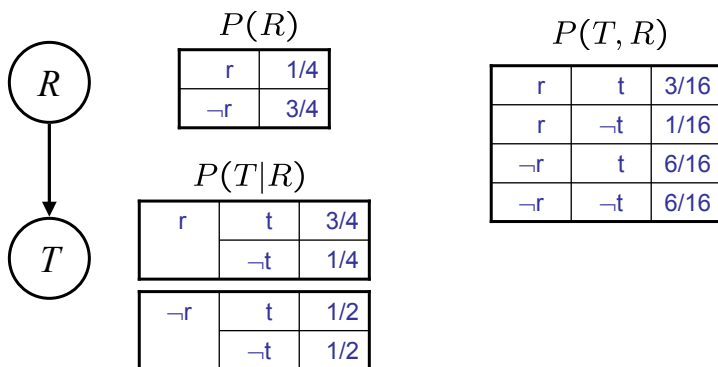


Causality?

- When Bayes' nets reflect the true causal patterns:
 - Often simpler (nodes have fewer parents)
 - Often easier to think about
 - Often easier to elicit from experts
- BNs need not actually be causal
 - Sometimes no causal net exists over the domain
 - E.g. consider the variables *Traffic* and *Drips*
 - End up with arrows that reflect correlation, not causation
- What do the arrows really mean?
 - Topology may happen to encode causal structure
 - **Topology only guaranteed to encode conditional independencies**

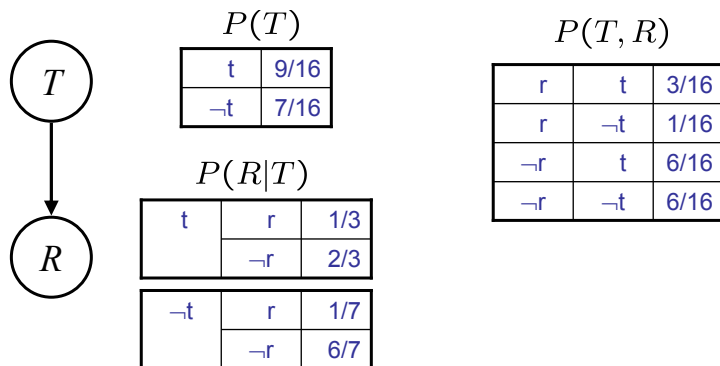
Example: Traffic

- Basic traffic net
- Let's multiply out the joint



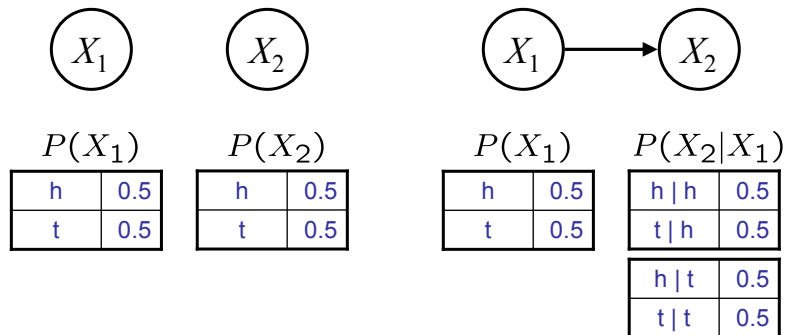
Example: Reverse Traffic

- Reverse causality?

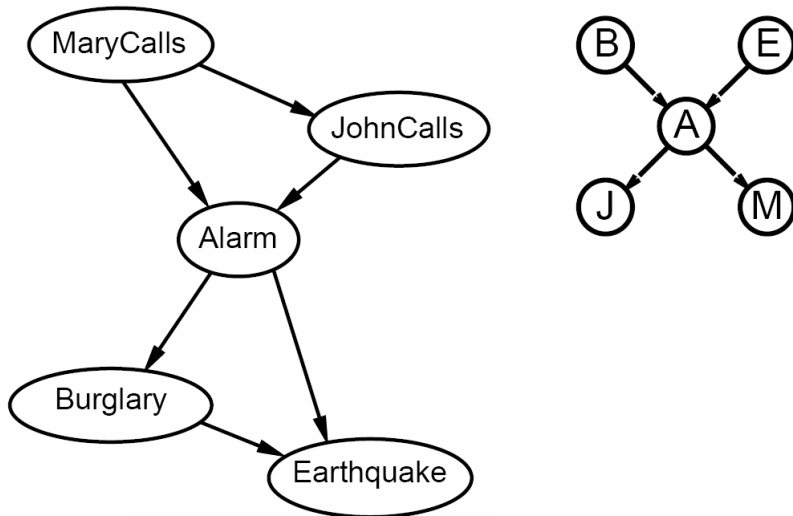


Example: Coins

- Extra arcs don't prevent representing independence, just allow non-independence



Alternate BNs



Summary

- Bayes nets compactly encode joint distributions
- Guaranteed independencies of distributions can be deduced from BN graph structure
- A Bayes' net may have other independencies that are not detectable until you inspect its specific distribution
- The Bayes' ball algorithm (aka d-separation) tells us when an observation of one variable can change belief about another variable

Inference

- Inference: calculating some statistic from a joint probability distribution

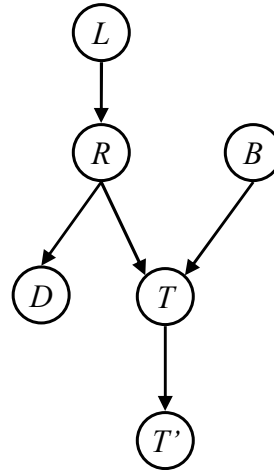
- Examples:

- Posterior probability:

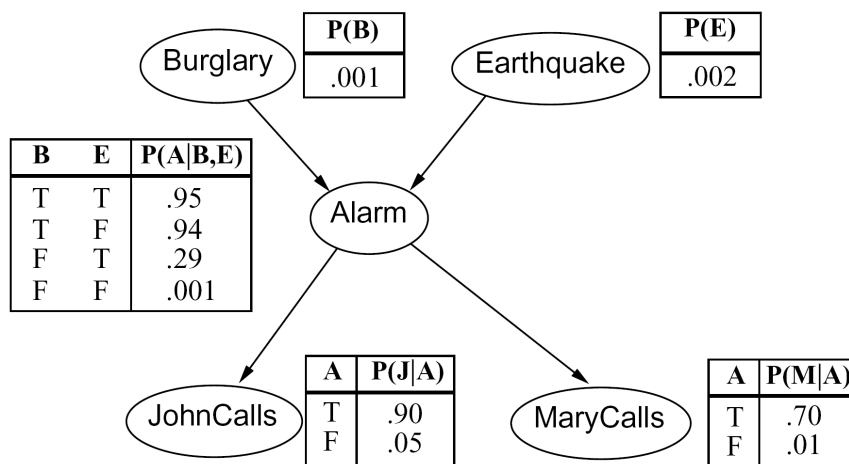
$$P(Q|E_1 = e_1, \dots, E_k = e_k)$$

- Most likely explanation:

$$\operatorname{argmax}_q P(Q = q|E_1 = e_1 \dots)$$



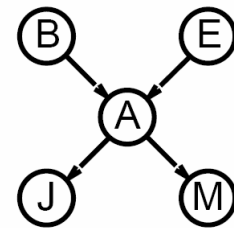
Reminder: Alarm Network



Inference by Enumeration

- Given unlimited time, inference in BNs is easy
- Recipe:
 - State the marginal probabilities you need
 - Figure out ALL the atomic probabilities you need
 - Calculate and combine them
- Example:

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)}$$

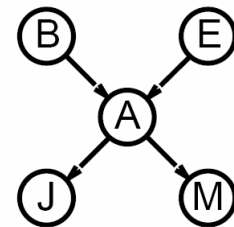


Example

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)}$$

$$P(b, j, m) = P(b, e, a, j, m) + P(b, \bar{e}, a, j, m) + P(b, e, \bar{a}, j, m) + P(b, \bar{e}, \bar{a}, j, m)$$

$$= \sum_{e, a} P(b, e, a, j, m)$$



Where did we use the BN structure?

We didn't!

Example

- In this simple method, we only need the BN to synthesize the joint entries

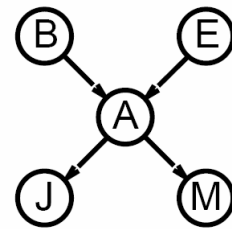
$$\begin{aligned}
 P(b, j, m) = & \\
 & P(b)P(e)P(a|b, e)P(j|a)P(m|a) + \\
 & P(b)P(e)P(\bar{a}|b, e)P(j|\bar{a})P(m|\bar{a}) + \\
 & P(b)P(\bar{e})P(a|b, \bar{e})P(j|a)P(m|a) + \\
 & P(b)P(\bar{e})P(\bar{a}|b, \bar{e})P(j|\bar{a})P(m|\bar{a})
 \end{aligned}$$

Normalization Trick

$$P(B|j, m) = \frac{P(B, j, m)}{P(j, m)}$$

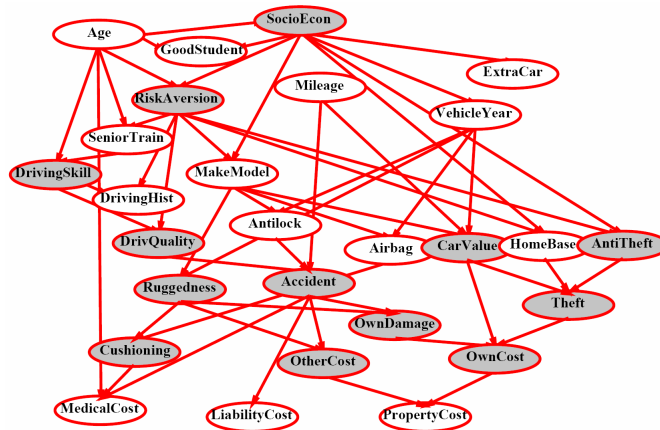
$$P(b, j, m) = \sum_{e, a} P(b, e, a, j, m)$$

$$P(\bar{b}, j, m) = \sum_{e, a} P(\bar{b}, e, a, j, m)$$



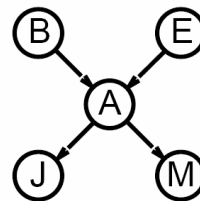
$$\begin{pmatrix} P(b, j, m) \\ P(\bar{b}, j, m) \end{pmatrix} \xrightarrow{\text{Normalize}} \begin{pmatrix} P(b|j, m) \\ P(\bar{b}|j, m) \end{pmatrix}$$

Inference by Enumeration?



Nesting Sums

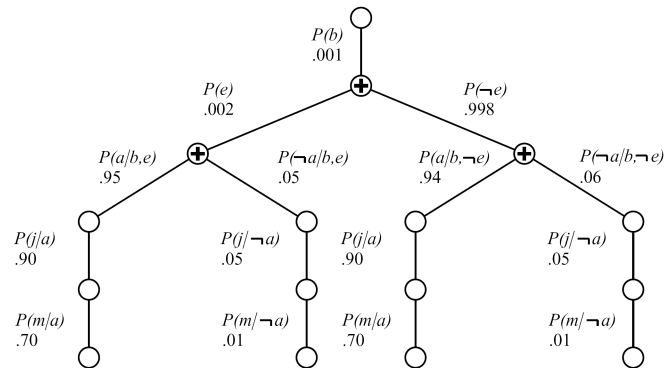
- Atomic inference is extremely slow!
- Slightly clever way to save work:
 - Move the sums as far right as possible
 - Example:



$$\begin{aligned}
 P(b, j, m) &= \sum_{e, a} P(b, e, a, j, m) \\
 &= \sum_{e, a} P(b)P(e)P(a|b, e)P(j|a)P(m|a) \\
 &= P(b) \sum_e P(e) \sum_a P(a|b, e)P(j|a)P(m|a)
 \end{aligned}$$

Evaluation Tree

- View the nested sums as a computation tree:

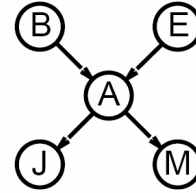


- Still repeated work: calculate $P(m|a)$ $P(j|a)$ twice, etc.

Variable Elimination: Idea

- Lots of redundant work in the computation tree
- We can save time if we cache all partial results
- This is the basic idea behind variable elimination

Basic Objects



- Track objects called **factors**
- Initial factors are local CPTs

$$\underbrace{P(B)}_{f_B(B)} \quad \underbrace{P(J|A)}_{f_J(A, J)} \quad \underbrace{P(A|B, E)}_{f_A(A, B, E)}$$

- During elimination, create new factors
- Anatomy of a factor:

4 numbers, one for each value of D and E

$$f_{A\bar{B}\bar{C}D}(D, E)$$

Variables introduced → A, \bar{B}, \bar{C}

Variables summed out → D, E

Argument variables, always non-evidence variables → (D, E)

Basic Operations

- First basic operation: **join factors**
- Combining two factors:
 - **Just like a database join**
 - Build a factor over the union of the domains
- Example:

$$f_1(A, B) \times f_2(B, C) \longrightarrow f_3(A, B, C)$$

$$f_3(a, b, c) = f_1(a, b) \cdot f_2(b, c)$$

$$"P(a, b|c) = P(a|b) \cdot P(b|c)"$$

Basic Operations

- Second basic operation: **marginalization**
- Take a factor and sum out a variable
 - Shrinks a factor to a smaller one
 - A **projection** operation
- Example:

$$f_{\bar{A}B}(b) = \sum_a f_{AB}(a, b)$$

$$"P(b) = \sum_a P(a, b)"$$

Example

$$P(b, j, m)$$

$$= \underbrace{P(b)}_B \sum_e \underbrace{P(e)}_E \sum_a \underbrace{P(a|b, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M$$

$$= f_B(b) \sum_e f_E(e) \sum_a f_A(a, b, e) f_J(a) f_M(a)$$

$$= f_B(b) \sum_e f_E(e) \sum_a f_{AJM}(a, b, e)$$

$$= f_B(b) \sum_e f_E(e) f_{\bar{A}JM}(b, e)$$

Example

$$\begin{aligned} P(b, j, m) &= f_B(b) \sum_e f_E(e) f_{\bar{A}JM}(b, e) \\ &= f_B(b) \sum_e f_{\bar{A}EJM}(b, e) \\ &= f_B(b) f_{\bar{A}\bar{E}JM}(b) \\ &= f_{\bar{A}B\bar{E}JM}(b) \end{aligned}$$

Variable Elimination

- What you need to know:
 - VE caches intermediate computations
 - Polynomial time for tree-structured graphs!
 - Saves time by marginalizing variables as soon as possible rather than at the end
- We will see special cases of VE later
 - You'll have to implement the special cases
- Approximations
 - Exact inference is slow, especially when you have a lot of hidden nodes
 - Approximate methods give you a (close) answer, faster