Representing Knowledge

- battery age
- alternator broken
- fanbelt broken
- battery dead
- battery meter
- no charging
- battery flat
- no oil
- no gas
- fuel line blocked
- starter broken
- lights
- oil light
- gas gauge
- car won't start
- dipstick
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**

\[
A \perp W \quad \text{Yes}
\]

\[
A \perp W | R
\]

Diagram:

- Nodes: aliens, watch, late, report
- Edges showing dependencies

- Yes
Example

$L \perp T'|T$  Yes
$L \perp B$  Yes
$L \perp B|T$
$L \perp B|T'$
$L \perp B|T, R$  Yes

Example

- **Variables:**
  - R: Raining
  - T: Traffic
  - D: Roof drips
  - S: I’m sad

- **Questions:**
  - $T \perp D$
  - $T \perp D|R$  Yes
  - $T \perp D|R, S$
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

\[
\begin{array}{c|c|c}
\text{r} & \text{t} & P(R) \\
\hline
\text{r} & \frac{1}{4} \\
\text{¬r} & \frac{3}{4} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{r} & \text{t} & P(T|R) \\
\hline
\text{r} & \text{t} & \frac{3}{4} \\
\text{r} & \text{¬t} & \frac{1}{4} \\
\text{¬r} & \text{t} & \frac{1}{2} \\
\text{¬r} & \text{¬t} & \frac{1}{2} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{r} & \text{t} & P(T, R) \\
\hline
\text{r} & \text{t} & \frac{3}{16} \\
\text{r} & \text{¬t} & \frac{1}{16} \\
\text{¬r} & \text{t} & \frac{6}{16} \\
\text{¬r} & \text{¬t} & \frac{6}{16} \\
\end{array}
\]
Example: Reverse Traffic

- Reverse causality?

\[
\begin{array}{c|c|c}
T & P(T) & P(T, R) \\
\hline
\text{t} & \frac{9}{16} & \text{r} \\
\neg\text{t} & \frac{7}{16} & \neg\text{r} \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
P(R|T) & & \\
\hline
\text{t} & \text{r} & \frac{1}{3} \\
\neg\text{r} & \frac{2}{3} & \\
\text{r} & \neg\text{t} & \text{r} \\
\neg\text{t} & \frac{1}{7} & \frac{6}{7} \\
\hline
\end{array}
\]

Example: Coins

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

\[
\begin{array}{c|c|c|c|c}
P(X_1) & P(X_2) & P(X_1) & P(X_2|X_1) \\
\hline
\text{h} & \text{h} & \text{h} & \text{h} | \text{h} \\
\text{t} & \text{t} & \text{t} & \text{t} | \text{h} \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
P(X_1) & P(X_2|X_1) \\
\hline
\text{h} & \text{h} & \text{h} | \text{t} \\
\text{t} & \text{t} & \text{t} | \text{t} \\
\hline
\end{array}
\]
Alternate BNs

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:
    \[ P(Q|E_1 = e_1, \ldots, E_k = e_k) \]
  - Most likely explanation:
    \[ \text{argmax}_q P(Q = q|E_1 = e_1, \ldots) \]

Reminder: Alarm Network

| B | E | P(A|B,E) |
|---|---|---------|
| T | T | .95     |
| T | F | .94     |
| F | T | .29     |
| F | F | .001    |

| A | P(J|A) |
|---|-------|
| T | .90   |
| F | .05   |

| A | P(M|A) |
|---|-------|
| T | .70   |
| F | .01   |
**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**

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

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

\[= \sum_{e,a} P(b, e, a, j, m)\]
Example

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

\[ P(b, j, m) = \]
\[ P(b) P(e) P(a|b, e) P(j|a) P(m|a) + \]
\[ P(b) P(e) P(\overline{a}|b, e) P(j|\overline{a}) P(m|\overline{a}) + \]
\[ P(b) P(\overline{e}) P(a|b, \overline{e}) P(j|a) P(m|a) + \]
\[ P(b) P(\overline{e}) P(\overline{a}|b, \overline{e}) P(j|\overline{a}) P(m|\overline{a}) \]

Normalization Trick

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

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

\[ P(\overline{b}, j, m) = \sum_{e, a} P(\overline{b}, e, a, j, m) \]

\[
\begin{pmatrix}
P(b, j, m) \\
P(\overline{b}, j, m)
\end{pmatrix}
\xrightarrow{\text{Normalize}}
\begin{pmatrix}
P(b|j, m) \\
P(\overline{b}|j, m)
\end{pmatrix}
Inference by Enumeration?

Atomic inference is extremely slow!

Slightly clever way to save work:
- Move the sums as far right as possible
- Example:

\[ P(b, j, m) = \sum_{e, a} P(b, e, a, j, m) \]
\[ = \sum_{e, a} P(b)P(e)P(a|b, e)P(j|a)P(m|a) \]
\[ = P(b) \sum_{e} P(e) \sum_{a} P(a|b, e)P(j|a)P(m|a) \]
Evaluation Tree

- View the nested sums as a computation tree:

  ![Evaluation Tree Diagram]

- Still repeated work: calculate $P(m \mid a) P(j \mid a)$ 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
  
  \[
  \begin{align*}
  P(B) & \quad P(J|A) & \quad P(A|B, E) \\
  f_R(B) & \quad f_J(A, J) & \quad f_A(A, B, E)
  \end{align*}
  \]

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

  \[
  f_{ABCD}(D, E)
  \]

  - Variables introduced
  - Variables summed out

  4 numbers, one for each value of D and 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:

  \[
  f_1(A, B) \times f_2(B, C) \quad \rightarrow \quad f_3(A, B, C)
  \]

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

  \[
  "P(a, b|c) = P(a|b) \cdot P(b|c)"
  \]
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:

\[
f_{AB}(b) = \sum_a f_{AB}(a, b)
\]

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

Example

\[
P(b, j, m) = \frac{P(b)}{B} \sum_e P(e) \frac{\sum_a P(a|b, e) P(j|a) P(m|a)}{A \quad J \quad M}
\]

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

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

\[
= f_B(b) \sum_e f_E(e) f_{\overline{AJM}}(b, e)
\]
Example

\[ P(b, j, m) = f_B(b) \sum_e f_E(e) f_{\bar{A}JM}(b, e) \]

\[ = f_B(b) \sum_e f_{\bar{A}EJM}(b, e) \]

\[ = f_B(b) f_{\bar{A}EJM}(b) \]

\[ = f_{\bar{A}B\bar{E}JM}(b) \]

Variable Elimination

- **What you need to know:**
  - VE caches intermediate computations
  - Polynomial time for tree-structured graphs!
  - Saves time by marginalizing variables ask 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