Bayesian networks

Chapter 14.1–3
Outline

♦ Syntax
♦ Semantics
♦ Parameterized distributions
Bayesian networks

A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

Syntax:
- a set of nodes, one per variable
- a directed, acyclic graph (link ≈ "directly influences")
- a conditional distribution for each node given its parents:
  \[ P(X_i|Parents(X_i)) \]

In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over \( X_i \) for each combination of parent values
Example

Topology of network encodes conditional independence assertions:

\[ \text{Weather} \quad \text{Cavity} \]
\[ \text{Toothache} \quad \text{Catch} \]

*Weather* is independent of the other variables

*Toothache* and *Catch* are conditionally independent given *Cavity*
Example

I’m at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn’t call. Sometimes it’s set off by minor earthquakes. Is there a burglar?

Variables: Burglar, Earthquake, Alarm, JohnCalls, MaryCalls

Network topology reflects “causal” knowledge:
- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call
Example contd.

\begin{itemize}
\item \textbf{Burglary}: \( P(B) = 0.001 \)
\item \textbf{Earthquake}: \( P(E) = 0.002 \)
\item \textbf{Alarm}: \( P(A|B,E) \) table:
\begin{tabular}{|c|c|c|}
\hline
B & E & P(A|B,E) \\
\hline
T & T & 0.95 \\
T & F & 0.94 \\
F & T & 0.29 \\
F & F & 0.001 \\
\hline
\end{tabular}
\item \textbf{JohnCalls}: \( P(J|A) \) table:
\begin{tabular}{|c|c|c|}
\hline
A & P(J|A) \\
\hline
T & 0.90 \\
F & 0.05 \\
\hline
\end{tabular}
\item \textbf{MaryCalls}: \( P(M|A) \) table:
\begin{tabular}{|c|c|c|}
\hline
A & P(M|A) \\
\hline
T & 0.70 \\
F & 0.01 \\
\hline
\end{tabular}
\end{itemize}
Compactness

A CPT for Boolean $X_i$ with $k$ Boolean parents has $2^k$ rows for the combinations of parent values.

Each row requires one number $p$ for $X_i = true$ (the number for $X_i = false$ is just $1 - p$).

If each variable has no more than $k$ parents, the complete network requires $O(n \cdot 2^k)$ numbers.

I.e., grows linearly with $n$, vs. $O(2^n)$ for the full joint distribution.

For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)
Global semantics

Global semantics defines the full joint distribution as the product of the local conditional distributions:

\[ P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i)) \]

e.g., \( P(j \land m \land a \land \neg b \land \neg e) \)
Global semantics

“Global” semantics defines the full joint distribution as the product of the local conditional distributions:

\[ P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i)) \]

e.g., \[ P(j \land m \land a \land \neg b \land \neg e) \]

\[ = P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \]

\[ = 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \]

\[ \approx 0.00063 \]
Local semantics: each node is conditionally independent of its nondescendants given its parents

Theorem: Local semantics $\iff$ global semantics
Markov blanket

Each node is conditionally independent of all others given its Markov blanket: parents + children + children’s parents
Constructing Bayesian networks

Need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics

1. Choose an ordering of variables $X_1, \ldots, X_n$
2. For $i = 1$ to $n$
   - add $X_i$ to the network
   - select parents from $X_1, \ldots, X_{i-1}$ such that
     \[ P(X_i | \text{Parents}(X_i)) = P(X_i | X_1, \ldots, X_{i-1}) \]

This choice of parents guarantees the global semantics:

\[
P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, \ldots, X_{i-1}) \quad \text{(chain rule)}
\]
\[
= \prod_{i=1}^{n} P(X_i | \text{Parents}(X_i)) \quad \text{(by construction)}
\]
Inference tasks

Simple queries: compute posterior marginal $P(X_i|E = e)$
  e.g., $P(\text{NoGas}|\text{Gauge} = \text{empty}, \text{Lights} = \text{on}, \text{Starts} = \text{false})$

 Conjunctive queries: $P(X_i, X_j|E = e) = P(X_i|E = e)P(X_j|X_i, E = e)$

 Optimal decisions: decision networks include utility information;
  probabilistic inference required for $P(\text{outcome}|\text{action}, \text{evidence})$

 Value of information: which evidence to seek next?

 Sensitivity analysis: which probability values are most critical?

 Explanation: why do I need a new starter motor?
Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:
\[ P(B|j,m) = \frac{P(B,j,m)}{P(j,m)} = \alpha P(B,j,m) = \alpha \sum_e \sum_a P(B,e,a,j,m) \]

Rewrite full joint entries using product of CPT entries:
\[ P(B|j,m) = \alpha \sum_e \sum_a P(B)P(e)P(a|B,e)P(j|a)P(m|a) = \alpha P(B) \sum_e P(e) \sum_a P(a|B,e)P(j|a)P(m|a) \]

Recursive depth-first enumeration: \( O(n) \) space, \( O(d^n) \) time
**Enumeration algorithm**

```plaintext
function Enumeration-Ask(\(X, e, bn\)) returns a distribution over \(X\)
inputs: \(X\), the query variable
\(e\), observed values for variables \(E\)
\(bn\), a Bayesian network with variables \(\{X\} \cup E \cup Y\)

\(Q(X) \leftarrow \) a distribution over \(X\), initially empty
for each value \(x_i\) of \(X\) do
    extend \(e\) with value \(x_i\) for \(X\)
    \(Q(x_i) \leftarrow\) Enumerate-All(Vars[\(bn\)], \(e\))
return Normalize(\(Q(X)\))
```

```plaintext
function Enumerate-All(\(vars, e\)) returns a real number
if Empty?(\(vars\)) then return 1.0

\(Y \leftarrow\) First(\(vars\))
if \(Y\) has value \(y\) in \(e\)
    then return \(P(y | Pa(Y)) \times\) Enumerate-All(Rest(\(vars\)), \(e\))
else return \(\sum_y P(y | Pa(Y)) \times\) Enumerate-All(Rest(\(vars\)), \(e_y\))
    where \(e_y\) is \(e\) extended with \(Y = y\)
```
Enumeration is inefficient: repeated computation

e.g., computes $P(j|a)P(m|a)$ for each value of $e$
Inference by variable elimination

Variable elimination: carry out summations right-to-left, storing intermediate results (factors) to avoid recomputation

\[
P(B|j, m) = \alpha \sum_e P(e) \sum_a P(a|B, e) P(j|a) P(m|a)
\]

\[
= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) f_M(a)
\]

\[
= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a)
\]

\[
= \alpha P(B) \sum_e P(e) f_{AJM}(b, e) \text{(sum out } A)\]

\[
= \alpha P(B) f_{EAJM}(b) \text{(sum out } E)\]

\[
= \alpha f_B(b) \times f_{EAJM}(b)
\]
Variable elimination: Basic operations

Summing out a variable from a product of factors:
move any constant factors outside the summation
add up submatrices in pointwise product of remaining factors

\[ \sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_X \]

assuming \( f_1, \ldots, f_i \) do not depend on \( X \)

Pointwise product of factors \( f_1 \) and \( f_2 \):

\[ f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l) = f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l) \]

E.g., \( f_1(a, b) \times f_2(b, c) = f(a, b, c) \)
Pointwise Product

- Pointwise multiplication of factors when variable is summed out or at last step
- Pointwise product of factors $f_1$ and $f_2$:
  \[ f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l) = f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l) \]
- E.g. $f_1(a, b) \times f_2(b, c) = f(a, b, c)$:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$f_1(a, b)$</th>
<th></th>
<th></th>
<th>$f_2(b, c)$</th>
<th></th>
<th></th>
<th>$f(a, b, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>0.3</td>
<td>T</td>
<td>T</td>
<td>0.2</td>
<td>T</td>
<td>T</td>
<td>0.3 * 0.2</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>0.7</td>
<td>T</td>
<td>F</td>
<td>0.8</td>
<td>T</td>
<td>T</td>
<td>0.3 * 0.8</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>0.9</td>
<td>F</td>
<td>T</td>
<td>0.6</td>
<td>T</td>
<td>F</td>
<td>0.7 * 0.6</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>0.1</td>
<td>F</td>
<td>F</td>
<td>0.4</td>
<td>T</td>
<td>F</td>
<td>0.7 * 0.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9 * 0.2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>0.9</td>
<td>F</td>
<td>T</td>
<td>0.8</td>
<td>F</td>
<td>F</td>
<td>0.9 * 0.8</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>0.1</td>
<td>F</td>
<td>T</td>
<td>0.6</td>
<td>F</td>
<td>F</td>
<td>0.1 * 0.6</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>0.1</td>
<td>F</td>
<td>F</td>
<td>0.4</td>
<td>F</td>
<td>F</td>
<td>0.1 * 0.4</td>
</tr>
</tbody>
</table>
Summing Out

- **Summing out**: a variable from a product of factors
- Move any constant factors outside the summation
- Add up sub-matrices in pointwise product of remaining factors

\[
\sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_l \times \left( \sum_x f_{l+1} \times \cdots \times f_k \right) = f_1 \times \cdots \times f_l \times f_{\bar{x}}
\]

- E.g. \( \sum_a f(a, b, c) = f_{\bar{a}}(b, c) \):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>f(a, b, c)</th>
<th></th>
<th></th>
<th>f_{\bar{a}}(b, c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>.3 * .2</td>
<td>T</td>
<td>T</td>
<td>.3 * .2 + .9 * .2</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>.3 * .8</td>
<td>T</td>
<td>F</td>
<td>.3 * .8 + .9 * .8</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>.7 * .6</td>
<td>F</td>
<td>T</td>
<td>.7 * .6 + .1 * .6</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>.7 * .4</td>
<td>F</td>
<td>F</td>
<td>.7 * .4 + .1 * .4</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>.9 * .2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>.9 * .8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>.1 * .6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>.1 * .4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Variable elimination algorithm**

function `Elimination-Ask(X, e, bn)` returns a distribution over `X`

**inputs:** `X`, the query variable

- `e`, evidence specified as an event
- `bn`, a belief network specifying joint distribution `P(X_1, \ldots, X_n)`

1. `factors ← [ ]; vars ← Reverse(VARS[bn])`
2. **for each** `var` **in** `vars` **do**
   - `factors ← [Make-Factor(var, e)|factors]`
   - **if** `var` **is** a hidden variable **then** `factors ← Sum-Out(var, factors)`
3. **return** `Normalize(Pointwise-Product(factors))`
Irrelevant variables

Consider the query $P(JohnCalls | Burglary = true)$

$$P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b,e) P(J|a) \sum_m P(m|a)$$

Sum over $m$ is identically 1; $M$ is irrelevant to the query

Thm 1: $Y$ is irrelevant unless $Y \in Ancestors(\{X\} \cup E)$

Here, $X = JohnCalls$, $E = \{Burglary\}$, and $Ancestors(\{X\} \cup E) = \{Alarm, Earthquake\}$ so $MaryCalls$ is irrelevant

(Compare this to backward chaining from the query in Horn clause KBs)
Complexity of exact inference

Singly connected networks (or polytrees):
- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:
- can reduce 3SAT to exact inference $\Rightarrow$ NP-hard
- equivalent to counting 3SAT models $\Rightarrow$ #P-complete

1. A $\lor$ B $\lor$ C
2. C $\lor$ D $\lor$ $\neg$A
3. B $\lor$ C $\lor$ $\neg$D

Chapter 14.4–5