Exact Inference Algorithms
Bucket-elimination

COMPSCI 276, Spring 2013
Class 5: Rina Dechter

(Reading: class notes chapter 4, Darwiche chapter 6)
Belief Updating

\[ P(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}) = ? \]
A Bayesian Network

Winter? (A)
Sprinkler? (B)
Rain? (C)
Wet Grass? (D)
Slippery Road? (E)

<table>
<thead>
<tr>
<th>A</th>
<th>Θ_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| A   | B   | Θ_B|A |
|-----|-----|----|
| true| true| .2 |
| true| false| .8 |
| false| true| .75 |
| false| false| .25 |

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Θ_D</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.95</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.05</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>.9</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>.1</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.8</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.2</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>E</th>
<th>Θ_E</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>.7</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>.3</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic Inference Tasks

- **Belief updating:** $E$ is a subset $\{X_1, \ldots, X_n\}$, $Y$ subset $X-E$, $P(Y=y|E=e)$
- **$P(e)$?** $\text{BEL}(X_i) = P(X_i = x_i | \text{evidence})$

Finding most probable explanation (MPE)

$$\bar{x}^* = \operatorname{arg max}_x P(x, e)$$

- **Finding maximum a-posteriori hypothesis**
  $$\begin{align*}
  & (a_1^*, \ldots, a_k^*) = \operatorname{arg max}_{\bar{a}} \sum_{X/A} P(\bar{x}, e) \\
  & A \subseteq X : \text{hypothesis variables}
  \end{align*}$$

- **Finding maximum-expected-utility (MEU) decision**
  $$\begin{align*}
  & (d_1^*, \ldots, d_k^*) = \operatorname{arg max}_d \sum_{X/D} P(\bar{x}, e)U(\bar{x}) \\
  & D \subseteq X : \text{decision variables}
  & U(\bar{x}) : \text{utility function}
  \end{align*}$$
Belief updating is NP-hard

- Each sat formula can be mapped to a Bayesian network query.
- Example: \((u, \neg v, w)\) and \((\neg u, \neg w, y)\) sat?
A simple network

Given:

- How can we compute $P(D)$, $P(D|A=0)$, $P(A|D=0)$?
- Brute force $O(k^4)$
- Maybe $O(4k^2)$
Elimination as a Basis for Inference

\[ A \rightarrow B \rightarrow C \]

<table>
<thead>
<tr>
<th></th>
<th>( \Theta_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

|   | \( \Theta_{B|A} \) |
|---|----------------|
| true true | .9 |
| true false | .1 |
| false true | .2 |
| false false | .8 |

|   | \( \Theta_{C|B} \) |
|---|----------------|
| true true | .3 |
| true false | .7 |
| false true | .5 |
| false false | .5 |

To compute the prior marginal on variable \( C, \Pr(C) \), we first eliminate variable \( A \) and then variable \( B \)
Elimination as a Basis for Inference

- There are two factors that mention variable $A$, $\Theta_A$ and $\Theta_{B|A}$.
- We multiply these factors first and then sum out variable $A$ from the resulting factor.
- Multiplying $\Theta_A$ and $\Theta_{B|A}$:

| $A$   | $B$    | $\Theta_A \Theta_{B|A}$ |
|-------|--------|-------------------------|
| true  | true   | .54                     |
| true  | false  | .06                     |
| false | true   | .08                     |
| false | false  | .32                     |

- Summing out variable $A$:

| $B$    | $\sum_A \Theta_A \Theta_{B|A}$ |
|--------|---------------------------------|
| true   | .62 = .54 + .08                |
| false  | .38 = .06 + .32                |
Elimination as a Basis for Inference

- We now have two factors, $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate variable $B$.

- Since $B$ appears in both factors, we must multiply them first and then sum out $B$ from the result.

- Multiplying:

  | $B$ | $C$ | $\Theta_{C|B} \sum_A \Theta_A \Theta_{B|A}$ |
  |-----|-----|------------------------------------------|
  | true| true| .186                                     |
  | true| false| .434                                     |
  | false| true| .190                                     |
  | false| false| .190                                     |

- Summing out:

  | $C$ | $\sum_B \Theta_{C|B} \sum_A \Theta_A \Theta_{B|A}$ |
  |-----|------------------------------------------|
  | true| .376                                     |
  | false| .624                                     |
Belief updating: \( P(X|\text{evidence}) = ? \)

\[
P(\text{a}e=0) \propto P(a,\text{e}=0) = \\
\sum_{e=0,d,c,b} P(\text{a})P(\text{b}|\text{a})P(\text{c}|\text{a})P(\text{d}|\text{b},\text{a})P(\text{e}|\text{b},\text{c}) = \\
P(\text{a})\sum_{e=0} \sum_{d} \sum_{c} P(\text{c}|\text{a}) \sum_{b} P(\text{b}|\text{a})P(\text{d}|\text{b},\text{a})P(\text{e}|\text{b},\text{c}) \\
\text{Variable Elimination} \\
\text{h}^B(a,d,c,e)
\]
Backwards Computation,

Ordering: a, e, d, c, b

\[ P(a, e = 0) = P(a) \sum_{e=0} \sum_d \sum_c P(c|a) \sum_b P(b|a) \]
\[ P(d|a, b) P(e|b, c) \]

\[ P(a) \sum_{e=0} \sum_d \sum_c P(c|a) \lambda_B(a, d, c, e) \]
\[ P(a) \sum_{e=0} \sum_d \lambda_C(a, d, e) \]
\[ P(a) \sum_{e=0} \lambda_D(a, e) \]
\[ P(a) \lambda_D(a, e = 0) \]
Backwards Computation,

Ordering: a, e, d, c, b
\[ P(a, e = 0) = P(a) \sum_{e=0} \sum_{d} \sum_{c} P(c|a) \sum_{b} P(b|a) \]
\[ P(d|a, b) P(e|b, c) \]

\[ P(a) \sum_{e=0} \sum_{d} \sum_{c} P(c|a) \lambda_B(a, d, c, e) \]
\[ P(a) \sum_{e=0} \sum_{d} \lambda_C(a, d, e) \]
\[ P(a) \sum_{e=0} \lambda_D(a, e) \]
\[ P(a) \lambda_D(a, e = 0) \]

The bucket elimination Process:

\[ bucket(B) = P(e|b, c), P(d|a, b), P(b|a) \]
\[ bucket(C) = P(c|a) \parallel \lambda_B(a, d, c, e) \]
\[ bucket(D) = \parallel \lambda_C(a, d, e) \]
\[ bucket(E) = e = 0 \parallel \lambda_D(a, e) \]
\[ bucket(A) = P(a) \parallel \lambda_D(a, e = 0) \]
Backwards Computation = Elimination

Using a different

\[
\text{Ordering: } a, b, c, d, e
\]
\[
P(a) \sum_b P(b|a) \sum_c P(c|a) \sum_d P(d|b, a) \sum_{e=0} P(e|b, c)
\]

\[
= P(a) \sum_b P(b|a) \sum_c P(c|a) P(e = 0|b, c) \sum_d P(d|b, a)
\]

\[
= P(a) \sum_b P(b|a) \lambda_D(a, b) \sum_c P(c|a) P(e = 0|b, c)
\]

\[
= P(a) \sum_b P(b|a) \lambda_D(a, b) \lambda_C(a, b)
\]

\[
= P(a) \lambda_B(a)
\]

The Bucket elimination process:

\[
\text{bucket}(E) = P(e|b, c), \hspace{1em} e = 0
\]

\[
\text{bucket}(D) = P(d|a, b)
\]

\[
\text{bucket}(C) = P(c|a)
\]

\[
\text{bucket}(B) = P(b|a)
\]

\[
\text{bucket}(A) = P(a)
\]
Bucket Elimination and Induced Width

Ordering: a, b, c, d, e
bucket(E) = \( P(e|b,c), \ e = 0 \)
bucket(D) = \( P(d|a,b) \)
bucket(C) = \( P(c|a) \ || \ P(e = 0|b,c) \)
bucket(B) = \( P(b|a) \ || \ \lambda_D(a,b), \lambda_C(b,c) \)
bucket(A) = \( P(a) \ || \ \lambda_B(a) \)

Ordering: a, e, d, c, b
bucket(B) = \( P(e|b,c), P(d|a,b), P(b|a) \)
bucket(C) = \( P(c|a) \ || \ \lambda_B(a,c,d,e) \)
bucket(D) = \( || \ \lambda_C(a,d,e) \)
bucket(E) = \( e = 0 \ || \ \lambda_D(a,c) \)
bucket(A) = \( P(a) \ || \ \lambda_E(a) \)
Factors: Sum-Out Operation

The result of summing out variable $X$ from factor $f(X)$ is another factor over variables $Y = X \setminus \{X\}$:

$$
\left( \sum_X f \right)(y) \overset{\text{def}}{=} \sum_x f(x, y)
$$

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.95</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.05</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>.9</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>.1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.8</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.2</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$\sum_D f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

$$
\sum_B \sum_C \sum_D f_1 = 4
$$
Factors: Sum-Out Operation

The sum-out operation is **commutative**

\[
\sum_Y \sum_X f = \sum_X \sum_Y f
\]

No need to specify the order in which variables are summed out.

If a factor \( f \) is defined over disjoint variables \( X \) and \( Y \)

then \( \sum_X f \) is said to **marginalize** variables \( X \)

If a factor \( f \) is defined over disjoint variables \( X \) and \( Y \)

then \( \sum_X f \) is called the result of **projecting** \( f \) on variables \( Y \)
## Factors: Multiplication Operation

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.95</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.05</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>.9</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>.1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.8</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.2</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D$</th>
<th>$E$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>0.448</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>0.192</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>0.112</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>0.248</td>
</tr>
</tbody>
</table>

The result of multiplying the above factors:

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$E$</th>
<th>$f_1(B, C, D)f_2(D, E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>$0.4256 = (.95)(.448)$</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>$0.1824 = (.95)(.192)$</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>$0.0056 = (.05)(.112)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>:</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>$0.2480 = (1)(.248)$</td>
</tr>
</tbody>
</table>
Factors: Multiplication Operation

The result of multiplying factors $f_1(X)$ and $f_2(Y)$ is another factor over variables $Z = X \cup Y$:

$$(f_1 f_2)(Z) \overset{\text{def}}{=} f_1(x) f_2(y),$$

where $x$ and $y$ are compatible with $z$; that is, $x \sim z$ and $y \sim z$.

Factor multiplication is commutative and associative.

It is meaningful to talk about multiplying a number of factors without specifying the order of this multiplication process.
A Bayesian network ordering: C,B,E,D,G

\[
P(a, g = 1) = \sum_{c,b,e,d,g=1} P(a, b, c, d, e, g) = \sum_{c,b,f,d,g=1} P(g|f)P(f|b, c)P(d|a, b)P(c|a)P(b|a)P(a).
\]

\[
P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b, c) \sum_d P(d|b, a) \sum_{g=1} P(g|f). \quad (4.1)
\]

\[
P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b, c) \lambda_G(f) \sum_{g=1} P(d|b, a). \quad (4.2)
\]

\[
P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \lambda_D(a, b) \sum_f P(f|b, c) \lambda_G(f) \quad (4.3)
\]

\[
P(a, g = 1) = P(a) \sum_c P(c|a) \sum_b P(b|a) \lambda_D(a, b) \lambda_F(b, c) \quad (4.4)
\]

\[
P(a, g = 1) = P(a) \sum_c P(c|a) \lambda_B(a, c) \quad (4.5)
\]

(a) Directed acyclic graph  
(b) Moral graph
A Bayesian network ordering: C, B, E, D, G

Figure 4.2: Bucket elimination along ordering $d_1 = A, C, B, F, D, G$. 
A different ordering

\[ P(a, g = 1) = P(a) \sum_f \sum_d \sum_c P(c|a) \sum_b P(b|a) \sum_{d_a,b} P(d|a,b) P(f|b, c) \sum_{g=1} P(g|f) \]
\[ = P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \sum_b P(b|a) P(d|a,b) P(f|b, c) \]
\[ = P(a) \sum_f \lambda_G(f) \sum_d \sum_c P(c|a) \lambda_B(a, d, c, f) \]
\[ = P(a) \sum_f \lambda_G(f) \sum_d \lambda_C(a, d, f) \]
\[ = P(a) \sum_f \lambda_G(f) \lambda_D(a, f) \]
\[ = P(a) \lambda_F(a) \]
A Bayesian network processed along 2 orderings

Figure 4.2: Bucket elimination along ordering $d_1 = A, C, B, F, D, G$. The bucket’s output when processing along $d_2 = A, F, D, C, B, G$. 
Bucket elimination

Algorithm BE-bel (Dechter 1996)

\[
\sum_b \prod
\]

Elimination operator

bucket B: \( P(b|a) \) \( P(d|b,a) \) \( P(e|b,c) \)

bucket C: \( P(c|a) \) \( h^B(a,d,c,e) \)

bucket D: \( h^C(a,d,e) \)

bucket E: \( e=0 \) \( h^D(a,e) \)

bucket A: \( P(a) \) \( h^E(a) \)

\( P(a|e=0) \)

\( W^* = 4 \)

"induced width" (max clique size)
Input: A belief network \( \{P_1, ..., P_n\} \), \(d,e\).
Output: belief of \(X_1\) given \(e\).

1. Initialize:
2. Process buckets from \(p = n\) to 1
   for matrices \(\lambda_1, \lambda_2, ..., \lambda_j\) in \(\text{bucket}_p\) do
   - If (observed variable) \(X_p = x_p\) assign
     \(X_p = x_p\) to each \(\lambda_i\).
   - Else, (multiply and sum)
     \(\lambda_p = \sum X_p \prod_{i=1}^{j} \lambda_i\).
     Add \(\lambda_p\) to its bucket.
3. Return \(Bel(x_1) = \alpha P(x_1) \cdot \prod_i \lambda_i(x_1)\)
Algorithm BE-bel

Input: A belief network $\mathcal{B} = \langle \mathcal{X}, \mathcal{D}, \mathcal{G}, \mathcal{P} \rangle$, an ordering $d = (x_1, \ldots, x_n)$; evidence $e$

Output: The belief $P(x_1|e)$ and probability of evidence $P(e)$

1. Partition the input functions (CPTs) into $\textit{bucket}_1, \ldots, \textit{bucket}_n$ as follows: for $i \leftarrow n$ downto 1, put in $\textit{bucket}_i$ all unplaced functions mentioning $x_i$. Put each observed variable in its bucket. Denote by $\psi_i$ the product of input functions in $\textit{bucket}_i$.

2. backward: for $p \leftarrow n$ downto 1 do

3. for all the functions $\psi_{s_0}, \lambda_{s_1}, \ldots, \lambda_{s_j}$ in $\textit{bucket}_p$ do

   If (observed variable) $X_p = x_p$ appears in $\textit{bucket}_p$,
   assign $X_p = x_p$ to each function in $\textit{bucket}_p$ and then
   put each resulting function in the bucket of the closest variable in its scope.
   else,

   4. $S_p \leftarrow \text{scope}(\psi_p) \cup \bigcup_{i=0}^p \text{scope}(\lambda_i) - \{X_p\}$
   5. $\lambda_p \leftarrow \sum_{X_p} \psi_p \cdot \prod_{i=1}^p \lambda_{s_i}$
   6. add $\lambda_p$ to the bucket of the latest variable in $S_p$,

8. return $P(e) = \alpha = \sum_{X_1} \psi_1 \cdot \prod_{\lambda \in \textit{bucket}_1} \lambda$

   return: $P(x_1|e) = \frac{1}{\alpha} \psi_1 \cdot \prod_{\lambda \in \textit{bucket}_1} \lambda$

Figure 4.4: BE-bel: a sum-product bucket-elimination algorithm
Student Network example

Difficulty

Intelligence

Grade

SAT

Letter

Apply

Job

P(J)?
Bucket Elimination and Induced Width

Ordering: a, b, c, d, e
- \text{bucket}(E) = P(e|b, c), \ e = 0
- \text{bucket}(D) = P(d|a, b)
- \text{bucket}(C) = P(c|a) \ || \ P(e = 0|b, c)
- \text{bucket}(B) = P(b|a) \ || \ \lambda_D(a, b), \lambda_C(b, c)
- \text{bucket}(A) = P(a) \ || \ \lambda_B(a)

Ordering: a, e, d, c, b
- \text{bucket}(B) = P(e|b, c), P(d|a, b), P(b|a)
- \text{bucket}(C) = P(c|a) \ || \ \lambda_B(a, c, d, e)
- \text{bucket}(D) = \ || \ \lambda_C(a, d, e)
- \text{bucket}(E) = e = 0 \ || \ \lambda_D(a, c)
- \text{bucket}(A) = P(a) \ || \ \lambda_E(a)
Complexity of elimination

\[ O(n \exp(w^*(d))) \]

\[ w^*(d) \] – the induced width of moral graph along ordering \( d \)

The effect of the ordering:

“Moral” graph

\[ w^*(d_1) = 4 \]

\[ w^*(d_2) = 2 \]
Complexity of bucket elimination

Theorem
Given a belief network having \( n \) variables, observations \( e \), the complexity of \( \text{BE-BEL} \) along \( d \), is time and space

\[
O(n \cdot \exp(w \ast (d))
\]

where \( w \ast (d) \) is the induced width of the moral graph whose edges connecting evidence to earlier nodes, were deleted.

More accurately: \( O(r \exp(w\ast(d)) \) where \( r \) is the number of cpts. For Bayesian networks \( r=n \). For Markov networks?
Handling Observations

Observing $b = 1$

Ordering: a, e, d, c, b

\[
\begin{align*}
\text{bucket}(B) &= P(e|b, c), P(d|a, b), P(b|a), b = 1 \\
\text{bucket}(C) &= P(c|a), \quad P(e|b = 1, c) \\
\text{bucket}(D) &= P(d|a, b = 1) \\
\text{bucket}(E) &= e = 0 \quad \lambda_C(e, a) \\
\text{bucket}(A) &= P(a), \quad P(b = 1|a) \quad \lambda_D(a), \lambda_E(e, a)
\end{align*}
\]

Ordering: a, b, c, d, e

\[
\begin{align*}
\text{bucket}(E) &= P(e|b, c), \quad e = 0 \\
\text{bucket}(D) &= P(d|a, b) \\
\text{bucket}(C) &= P(c|a) \quad \lambda_E(b, c) \\
\text{bucket}(B) &= P(b|a), \quad b = 1 \quad \lambda_D(a, b), \lambda_C(a, b) \\
\text{bucket}(A) &= P(a) \quad \lambda_B(a)
\end{align*}
\]
The impact of observations

(a) Ordered graph

(b) Induced graph

(c) Ordered conditioned graph

- (a) Directed acyclic graph
- (b) Moral graph
“Moral” graph

Irrelevant buckets for BE-BEL

Buckets that sum to 1 are irrelevant.
Identification: no evidence, no new functions.

Recursive recognition: \( \text{bel}(a|e) \)
- \( \text{bucket}(E) = P(e|b,c), \quad e = 0 \)
- \( \text{bucket}(D) = P(d|a,b), \text{...skipable bucket} \)
- \( \text{bucket}(C) = P(c|a) \)
- \( \text{bucket}(B) = P(b|a) \)
- \( \text{bucket}(A) = P(a) \)

Complexity: Use induced width in moral graph without irrelevant nodes, then update for evidence arcs.

Use the ancestral graph only
Pruning Nodes

Given a Bayesian network $\mathcal{N}$ and query $(Q, e)$

one can remove any leaf node (with its CPT) from the network as long as it does not belong to variables $Q \cup E$, yet not affect the ability of the network to answer the query correctly.

If $\mathcal{N}' = \text{pruneNodes}(\mathcal{N}, Q \cup E)$

then $Pr(Q, e) = Pr'(Q, e)$, where $Pr$ and $Pr'$ are the probability distributions induced by networks $\mathcal{N}$ and $\mathcal{N}'$, respectively.
Pruning Nodes: Example

network structure

joint on $B, E$

joint on $B$
Pruning Edges: Example

\[
\begin{array}{ccc}
A & B & \Theta_{B|A} \\
true & true & .2 \\
true & false & .8 \\
false & true & .75 \\
false & false & .25
\end{array}
\]

\[
\begin{array}{ccc}
A & C & \Theta_{C|A} \\
true & true & .8 \\
true & false & .2 \\
false & true & .1 \\
false & false & .9
\end{array}
\]

Evidence \( e : C = false \)
Pruning Nodes and Edges: Example

| $B$   | $\Theta'_B = \sum_A \Theta_{B|A}^A_{\text{true}}$ |
|-------|-----------------------------------------------|
| true  | .2                                            |
| false | .8                                            |

| $C$   | $\Theta'_C = \sum_A \Theta_{C|A}^A_{\text{true}}$ |
|-------|-----------------------------------------------|
| true  | .8                                            |
| false | .2                                            |

Query $Q = \{D\}$ and $e: A = \text{true}, C = \text{false}$
Probabilistic Inference Tasks

- **Belief updating:**

  \[ \text{BEL}(X_i) = P(X_i = x_i \mid \text{evidence}) \]

- Finding most probable explanation (MPE)

  \[ \bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e) \]

- Finding maximum a-posteriori hypothesis

  \[ (a_1^*, \ldots, a_k^*) = \arg\max_{\bar{a}} \sum_{X/A} P(\bar{x}, e) \quad A \subseteq X : \quad \text{hypothesis variables} \]
Finding \[ MPE = \max_{\bar{x}} P(\bar{x}) \]

Algorithm \textit{BE-mpe}

\[ \sum \text{is replaced by } \max: \]

\[ MPE = \max_{a,e,d,c,b} P(a)P(c \mid a)P(b \mid a)P(d \mid a,b)P(e \mid b, c) \]
Finding \( MPE = \max P(\overline{x}) \)

Algorithm \textit{elim-mpe} (Dechter 1996)

\[
MPE = \max_{a,e,d,c,b} P(a)P(c \mid a)P(b \mid a)P(d \mid a,b)P(e \mid b, c)
\]

\[
\sum \text{ is replaced by } \max : \]

bucket B: \( P(b \mid a) \quad P(d \mid b, a) \quad P(e \mid b, c) \)

bucket C: \( P(c \mid a) \quad h^B(a, d, c, e) \)

bucket D: \( h^C(a, d, e) \)

bucket E: \( e = 0 \quad h^D(a, e) \)

bucket A: \( P(a) \quad h^E(a) \)

\( MPE \)

“induced width” (max clique size)

\( W^* = 4 \)

Elimination operator
Generating the MPE-tuple

1. \( a' = \arg \max_a P(a) \cdot h^E (a) \)

2. \( e' = 0 \)

3. \( d' = \arg \max_d h^C (a', d, e') \)

4. \( c' = \arg \max_c P(c / a') \times h^B (a', d', c, e') \)

5. \( b' = \arg \max_b P(b / a') \times P(d' / b, a') \times P(e' / b, c') \)

Return \((a', b', c', d', e')\)
Algorithm BE-mpe

**Input:** A belief network $\mathcal{B} = \langle X, D, G, \mathcal{P} \rangle$, where $\mathcal{P} = \{P_1, ..., P_n\}$; an ordering of the variables, $d = X_1, ..., X_n$; observations $e$.

**Output:** The most probable assignment given the evidence.

1. **Initialize:** Generate an ordered partition of the conditional probability function, $\text{bucket}_1, ..., \text{bucket}_n$, where $\text{bucket}_i$ contains all functions whose highest variable is $X_i$. Put each observed variable in its bucket. Let $\psi_i$ be the input function in a bucket and let $h_i$ be the messages in the bucket.

2. **Backward:** For $p \leftarrow n$ downto 1, do
   
   for all the functions $h_1, h_2, ..., h_j$ in $\text{bucket}_p$, do
   
   - If (observed variable) $\text{bucket}_p$ contains $X_p = x_p$, assign $X_p = x_p$ to each function and put each in appropriate bucket.
   
   - else, $S_p \leftarrow \bigcup_{i=1}^{j} \text{scope}(h_i) \cup \text{scope}(\psi_p) - \{X_p\}$. Generate functions $h_p \leftarrow \max_{X_p} \psi_p \cdot \Pi_{i=1}^{j} h_i$ Add $h_p$ to the bucket of the largest-index variable in $S_p$.

3. **Forward:**
   
   - Generate the mpe cost by maximizing over $X_1$, the product in $\text{bucket}_1$.
   
   - (generate an mpe tuple)
     
     For $i = 1$ to $n$ along $d$ do: Given $\bar{x}_{i-1} = (x_1, ..., x_{i-1})$ Choose $x_i = \arg\max_{X_i} \psi_i \cdot \Pi_{\{h_j \in \text{bucket}_i\}} h_j(\bar{x}_{i-1})$
Finding MAP

Algorithm *BE-map*

\[ \sum \text{ and max :} \]

\[ MPE = \max_{a,c} P(a)P(c \mid a) \sum_{e,d,b} P(b \mid a)P(d \mid a,b)P(e \mid b,c) \]
Finding the MAP
(An optimization task)

Variables $A$ and $B$ are the hypothesis variables.

**Ordering:** $a, b, c, d, e$

\[
\max_{a,b} P(a, b, e = 0) = \max_{a,b} \sum_{c,d,e=0} P(a, b, c, d, e) \\
= \max_a P(a) \max_b P(b|a) \sum_c P(c|a) \sum_d P(d|b, a) \sum_{e=0} P(e|b, c)
\]

**Ordering:** $a, e, d, c, b$ .... illegal ordering

\[
\max_{a,b} P(a, e, e = 0) = \max_{a,b} \sum P(a, b, c, d, e) \\
\max_{a,b} P(a, b, e = 0) = \max_a P(a) \max_b P(b|a) \sum d \cdot \max_c P(c|a) P(d|a, b) P(e = 0|b, c)
\]
Algorithm BE-MAP

Variable ordering:
Restricted: Max buckets should Be processed after sum buckets

Theorem 4.2.3 Algorithm BE-map is complete for the map task for orderings started by the hypothesis variables. Its time and space complexity are are $O(r \cdot k w^*_a(E) + 1)$ and $O(n \cdot k w^*_a(E))$, respectively, where $n$ is the number of variables in graph, $k$ bounds the domain size and $w^*_a(E)$ is the conditioned induced width of its moral graph along $d$. (prove as an exercise.) \(\square\)
BE for Markov networks queries
Complexity of bucket elimination

Theorem
Given a belief network having \( n \) variables, observations \( e \), the complexity of elim-mpe, elim-bel, elim-map along \( d \), is time and space

\[
O(n^{\exp(w^*(d)+1)}) \text{ and } O(n^{\exp(w*)}), \text{ respectively}
\]

where \( w^*(d) \) is the induced width of the moral graph whose edges connecting evidence to earlier nodes, were deleted.

More accurately: \( O(r^{\exp(w^*(d))} \) where \( r \) is the number of cpts. For Bayesian networks \( r=n \). For Markov networks?
Finding small induced-width

- NP-complete
- A tree has induced-width of ?
- Greedy algorithms:
  - Min width
  - Min induced-width
  - Max-cardinality
  - Fill-in (thought as the best)
  - See anytime min-width (Gogate and Dechter)
Figure 5.1: (a) Hyper, (b) Primal, (c) Dual and (d) Join-tree of a graphical model having scopes ABC, AEF, CDE and ACE. (e) the factor graph
The induced width

Definition 5.2.1 (width) Given an undirected graph $G = (V, E)$, an ordered graph is a pair $(G, d)$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E$ is a set of arcs over $V$, and $d = (v_1, ..., v_n)$ is an ordering of the nodes. The nodes adjacent to $v$ that precede it in the ordering are called its parents. The width of a node in an ordered graph is its number of parents. The width of an ordering $d$ of $G$, denoted $w_d(G)$ (or $w_d$ for short) is the maximum width over all nodes. The width of a graph is the minimum width over all the orderings of the graph.

Definition 5.2.3 (induced width) The induced width of an ordered graph $(G, d)$, denoted $w^*_d$), is the width of the induced ordered graph along $d$ obtained as follows: nodes are processed from last to first; when node $v$ is processed, all its parents are connected. The induced width of a graph, denoted by $w^*$, is the minimal induced width over all its orderings. Formally

$$w^*(G) = \min_{d \in \text{orderings}} w^*_d(G)$$
Min-width ordering

MIN-WIDTH (MW)

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

output: A min-width ordering of the nodes $d = (v_1, ..., v_n)$.

1. for $j = n$ to 1 by -1 do
2. $r \leftarrow$ a node in $G$ with smallest degree.
3. put $r$ in position $j$ and $G \leftarrow G - r$.
   (Delete from $V$ node $r$ and from $E$ all its adjacent edges)
4. endfor

Proposition: algorithm min-width finds a min-width ordering of a graph
Greedy orderings heuristics

**MIN-INDUCED-WIDTH (MIW)**

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

output: An ordering of the nodes $d = (v_1, ..., v_n)$.

1. for $j = n$ to 1 by -1 do
2.   $r \leftarrow$ a node in $V$ with smallest degree.
3.   put $r$ in position $j$.
4. connect $r$’s neighbors: $E \leftarrow E \cup \{(v_i, v_j) | (v_i, r) \in E, (v_j, r) \in E\}$,
5. remove $r$ from the resulting graph: $V \leftarrow V - \{r\}$.

**Theorem:** A graph is a tree iff it has both width and induced-width of 1.

**MIN-FILL (MIN-FILL)**

input: a graph $G = (V, E)$, $V = \{v_1, ..., v_n\}$

output: An ordering of the nodes $d = (v_1, ..., v_n)$.

1. for $j = n$ to 1 by -1 do
2.   $r \leftarrow$ a node in $V$ with smallest fill edges for his parents.
3.   put $r$ in position $j$.
4. connect $r$’s neighbors: $E \leftarrow E \cup \{(v_i, v_j) | (v_i, r) \in E, (v_j, r) \in E\}$,
5. remove $r$ from the resulting graph: $V \leftarrow V - \{r\}$.
Different Induced-graphs
Induced-width for chordal graphs

- **Definition:** A graph is chordal if every cycle of length at least 4 has a chord.
- Finding \( w^* \) over chordal graph is easy using the **max-cardinality ordering:** order vertices from 1 to \( n \), always assigning the next number to the node connected to a largest set of previously numbered nodes. Let \( d \) be such an ordering.
- A graph along max-cardinality order has no fill-in edges iff it is chordal.
- On chordal graphs \( \text{width}=\text{induced-width} \).
Max-cardinality ordering

**MAX-CARDINALITY (MC)**

**input:** a graph \( G = (V, E) \), \( V = \{v_1, ..., v_n\} \)

**output:** An ordering of the nodes \( d = (v_1, ..., v_n) \).

1. Place an arbitrary node in position 0.
2. for \( j = 1 \) to \( n \) do
3. \( r \leftarrow \) a node in \( G \) that is connected to a largest subset of nodes in positions 1 to \( j - 1 \), breaking ties arbitrarily.
4. endfor

**Proposition 5.3.3** [56] Given a graph \( G = (V, E) \) the complexity of max-cardinality search is \( O(n + m) \) when \( |V| = n \) and \( |E| = m \).

What is the complexity of min-fill? Min-induced-width? \( O(n^3) \)
K-trees

Definition 5.3.4 (K-trees) A subclass of chordal graphs are k-trees. A k-tree is a chordal graph whose maximal cliques are of size $k + 1$, and it can be defined recursively as follows:

1. A complete graph with $k$ vertices is a k-tree.
2. A k-tree with $r$ vertices can be extended to $r + 1$ vertices by connecting the new vertex to all the vertices in any clique of size $k$. A partial k-tree is a k-tree having some of its arcs removed. Namely it will clique of size smaller than $k$. 

Which greedy algorithm is best?

- MinFill, prefers a node who add the least number of fill-in arcs.

- Empirically, fill-in is the best among the greedy algorithms (MW, MIW, MF, MC)

- Complexity of greedy orderings?
  - MW is $O(n^2)$...maybe $O(n\log n + m)$?
  - MIW: $O(\Theta(n^3))$,
  - MF ($O(n^3)$,
  - MC is $O(m+n)$, m edges.
Recent work in my group

- **Vibhav Gogate and Rina Dechter.** "A Complete Anytime Algorithm for Treewidth". *In UAI 2004.*

- **Andrew E. Gelfand, Kalev Kask, and Rina Dechter.** "Stopping Rules for Randomized Greedy Triangulation Schemes" in *Proceedings of AAAI 2011.*

- Potential project