Lecture 2: Inference
Inference: An Ubiquitous Obstacle

• Decoding is inference.
• Subroutines for learning are inference.
• Learning is inference.

• Exact inference is #P-complete.
  – Even approximations within a given absolute or relative error are hard.
Probabilistic Inference Problems

Given values for some random variables \((X \subset V)\) ...

- **Most Probable Explanation**: what are the *most probable* values of the rest of the r.v.s \(V \setminus X\)?

(More generally ...)

- **Maximum A Posteriori (MAP)**: what are the most probable values of *some* other r.v.s, \(Y \subset (V \setminus X)\)?

- Random **sampling** from the posterior over values of \(Y\)
- Full **posterior** over values of \(Y\)
- **Marginal** probabilities from the posterior over \(Y\)

- **Minimum Bayes risk**: What is the \(Y\) with the lowest expected cost?
- **Cost-augmented decoding**: What is the most *dangerous* \(Y\)?
Approaches to Inference

- **Inference**
  - **Exact**
    - Variable elimination
    - ILP
    - Dynamic program’ng
    - MCMC
    - Gibbs
  - **Approximate**
    - Randomized search
    - Simulated annealing
    - Mean field
    - Variational propagation
    - Loopy belief propagation
    - LP relaxations
    - Dual decomp.
    - Beam search

Today

Lecture 6
Exact Marginal for Y

• This will be a generalization of algorithms you already know: the *forward* and *backward* algorithms.

• The general name is *variable elimination*.

• After we see it for the marginal, we’ll see how to use it for the MAP.
Simple Inference Example

• Goal: $P(D)$
Simple Inference Example

• Let’s calculate $P(B)$ from things we have.

<table>
<thead>
<tr>
<th>$P(B \mid A)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P(C \mid B)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P(D \mid C)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simple Inference Example

• Let’s calculate $P(B)$ from things we have.

\[ P(B) = \sum_{a \in \text{Val}(A)} P(A = a)P(B \mid A = a) \]
Simple Inference Example

• Let’s calculate $P(B)$ from things we have.

$$P(B) = \sum_{a \in \text{Val}(A)} P(A = a)P(B \mid A = a)$$

• Note that C and D do not matter.
Simple Inference Example

• Let’s calculate $P(B)$ from things we have.

$$P(B) = \sum_{a \in \text{Val}(A)} P(A = a)P(B \mid A = a)$$

|   | P(B | A) | 0 | 1 |
|---|--------|---|---|
| T |        |   |   |
| 0 | 0      |   |   |
| 1 | 1      |   |   |

$$= \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
Simple Inference Example

- We now have a Bayesian network for the marginal distribution \( P(B, C, D) \).
Simple Inference Example

• We can repeat the same process to calculate $P(C)$.

$$P(C) = \sum_{b \in \text{Val}(B)} P(B = b)P(C \mid B = b)$$

• We already have $P(B)$!
Simple Inference Example

• We can repeat the same process to calculate $P(C)$.

$$P(C) = \sum_{b \in \text{Val}(B)} P(B = b)P(C | B = b)$$
Simple Inference Example

• We now have $P(C, D)$.
• Marginalizing out A and B happened in two steps, and we are exploiting the Bayesian network structure.

| $P(D | C)$ | 0 | 1 |
|-----------|---|---|
| 0         |   |   |
| 1         |   |   |
Simple Inference Example

- Last step to get $P(D)$:

$$P(D) = \sum_{c \in \text{Val}(C)} P(C = c) P(D \mid C = c)$$

| P(D | C) | 0   | 1   |
|---------|-----|-----|
| 0       |     |     |
| 1       |     |     |

$$= \begin{pmatrix}
0 & 1
\end{pmatrix}$$
Simple Inference Example

• Notice that the same step happened for each random variable:
  
  – We created a new CPD over the variable and its “successor”
  – We summed out (marginalized) the variable.

\[
P(D) = \sum_{a \in \text{Val}(A)} \sum_{b \in \text{Val}(B)} \sum_{c \in \text{Val}(C)} P(A = a)P(B = b \mid A = a)P(C = c \mid B = b)P(D \mid C = c)
\]

\[
= \sum_{c \in \text{Val}(C)} P(D \mid C = c) \sum_{b \in \text{Val}(B)} P(C = c \mid B = b) \sum_{a \in \text{Val}(A)} P(A = a)P(B = b \mid A = a)
\]
That Was Variable Elimination

• We reused computation from previous steps and avoided doing the same work more than once.
  – Dynamic programming à la forward algorithm!
• We exploited the Bayesian network structure (each subexpression only depends on a small number of variables).
• Exponential blowup avoided!
What Remains

• Some machinery
• Variable elimination in general
• The maximization version (for MAP inference)
• A bit about approximate inference
Factor Graphs

• Variable nodes (circles)
• Factor nodes (squares)
  – Can be MN factors or BN conditional probability distributions!
• Edge between variable and factor if the factor depends on that variable.

• The graph is bipartite.
Products of Factors

• Given two factors with different scopes, we can calculate a new factor equal to their products.

\[ \phi_{product}(x \cup y) = \phi_1(x) \cdot \phi_2(y) \]
Products of Factors

• Given two factors with different scopes, we can calculate a new factor equal to their products.

\[
\begin{array}{c|c|c|c|c|c|c|c}
A & B & \varphi_1(A, B) & B & C & \varphi_2(B, C) & A & B & C & \varphi_3(A, B, C) \\
0 & 0 & 30 & 0 & 0 & 100 & 0 & 0 & 0 & 3000 \\
0 & 1 & 5 & 0 & 1 & 1 & 0 & 0 & 1 & 30 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 100 \\
1 & 1 & 10 & 1 & 1 & 100 & 1 & 1 & 0 & 10 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1000 \\
\end{array}
\]
Factor Marginalization

• Given $X$ and $Y$ ($Y \notin X$), we can turn a factor $\varphi(X, Y)$ into a factor $\psi(X)$ via marginalization:

$$\psi(X) = \sum_{y \in \text{Val}(Y)} \varphi(X, y)$$
Factor Marginalization

• Given \( X \) and \( Y (Y \notin X) \), we can turn a factor \( \varphi(X, Y) \) into a factor \( \psi(X) \) via marginalization:

\[
\psi(X) = \sum_{y \in \text{Val}(Y)} \varphi(X, y)
\]

<table>
<thead>
<tr>
<th>( P(C \mid A, B) )</th>
<th>0, 0</th>
<th>0, 1</th>
<th>1, 0</th>
<th>1,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.6</td>
<td>0.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

“summing out” \( B \)

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>( \psi(A, C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.7</td>
</tr>
</tbody>
</table>
Factor Marginalization

• Given \( X \) and \( Y \) \((Y \notin X)\), we can turn a factor \( \varphi(X, Y) \) into a factor \( \psi(X) \) via marginalization:

\[
\psi(\mathbf{X}) = \sum_{y \in \text{Val}(Y)} \varphi(\mathbf{X}, y)
\]

<table>
<thead>
<tr>
<th>( P(C \mid A, B) )</th>
<th>0, 0</th>
<th>0, 1</th>
<th>1, 0</th>
<th>1,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.6</td>
<td>0.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

“summing out” \( C \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( \psi(A, B) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Factor Marginalization

• Given $X$ and $Y$ ($Y \notin X$), we can turn a factor $\varphi(X, Y)$ into a factor $\psi(X)$ via marginalization:

$$\psi(X) = \sum_{y \in \text{Val}(Y)} \phi(X, y)$$

• We can refer to this new factor by $\sum_{Y} \varphi$. 
Marginalizing Everything?

• Take a Markov network’s “product factor” by multiplying all of its factors.

• Sum out all the variables (one by one).

• What do you get?
Factors Are Like Numbers

- Products are commutative: $\varphi_1 \cdot \varphi_2 = \varphi_2 \cdot \varphi_1$
- Products are associative:
  $$(\varphi_1 \cdot \varphi_2) \cdot \varphi_3 = \varphi_1 \cdot (\varphi_2 \cdot \varphi_3)$$
- Sums are commutative: $\sum_X \sum_Y \varphi = \sum_Y \sum_X \varphi$
- Distributivity of multiplication over summation:
  $$X \not\in \text{Scope}(\phi_1) \implies \sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2$$
Eliminating One Variable

Input: Set of factors $\Phi$, variable $Z$ to eliminate
Output: new set of factors $\Psi$

1. Let $\Phi' = \{ \varphi \in \Phi \mid Z \in \text{Scope}(\varphi) \}$
2. Let $\Psi = \{ \varphi \in \Phi \mid Z \notin \text{Scope}(\varphi) \}$
3. Let $\psi$ be $\sum_{Z} \prod_{\varphi \in \Phi'} \varphi$
4. Return $\Psi \cup \{ \psi \}$
Example

- Query: \( P(\text{Flu} \mid \text{runny nose}) \)
- Let’s eliminate H.
Example

• Query:
  \( P(\text{Flu} \mid \text{runny nose}) \)

• Let’s eliminate H.
Example

• Query: \( P(\text{Flu} \mid \text{runny nose}) \)

• Let’s eliminate H.
  1. \( \Phi' = \{ \varphi_{SH} \} \)
  2. \( \Psi = \{ \varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR} \} \)
  3. \( \psi = \sum_H \prod_{\varphi \in \Phi'} \varphi \)
  4. Return \( \Psi \cup \{ \psi \} \)
Example

- Query:
  \[ P(\text{Flu} \mid \text{runny nose}) \]

- Let’s eliminate H.
  1. \( \Phi' = \{\varphi_{SH}\} \)
  2. \( \Psi = \{\varphi_F, \varphi_A, \varphi_{FAS}, \varphi_{SR}\} \)
  3. \( \psi = \sum_H \varphi_{SH} \)
  4. Return \( \Psi \cup \{\psi\} \)
Example

- Query: \( P(\text{Flu} \mid \text{runny nose}) \)

- Let’s eliminate H.
  1. \( \Phi' = \{\phi_{SH}\} \)
  2. \( \Psi = \{\phi_{F}, \phi_{A}, \phi_{FAS}, \phi_{SR}\} \)
  3. \( \psi = \sum_{H} \phi_{SH} \)
  4. Return \( \Psi \cup \{\psi\} \)
Example

• Query: $P(\text{Flu} \mid \text{runny nose})$

• Let’s eliminate H.
  1. $\Phi' = \{\varphi_{\text{SH}}\}$
  2. $\Psi = \{\varphi_{\text{F}}, \varphi_{\text{A}}, \varphi_{\text{FAS}}, \varphi_{\text{SR}}\}$
  3. $\psi = \sum_{H} \varphi_{\text{SH}}$
  4. Return $\Psi \cup \{\psi\}$

<table>
<thead>
<tr>
<th>$P(H \mid S)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S$</th>
<th>$\psi(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Example

• Query: \( P(\text{Flu} \mid \text{runny nose}) \)

• Let’s eliminate H.

• We can actually ignore the new factor, equivalently just deleting H!
  – Why?
  – In some cases eliminating a variable is really easy!

<table>
<thead>
<tr>
<th>S</th>
<th>( \psi(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Variable Elimination

Input: Set of factors Φ, ordered list of variables Z to eliminate

Output: new factor ψ

1. For each $Z_i \in Z$ (in order):
   - Let $Φ = \text{Eliminate-One}(Φ, Z_i)$

2. Return $\prod_{ϕ \in Φ} ϕ$
Example

• Query: \( P(\text{Flu} \mid \text{runny nose}) \)

• H is already eliminated.

• Let’s now eliminate S.
Example

• Query:
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Eliminating S.
  1. \( \Phi' = \{ \varphi_{\text{SR}}, \varphi_{\text{FAS}} \} \)
  2. \( \Psi = \{ \varphi_{\text{F}}, \varphi_{\text{A}} \} \)
  3. \( \psi_{\text{FAR}} = \sum_S \prod_{\varphi \in \Phi'} \varphi \)
  4. Return \( \Psi \cup \{ \psi_{\text{FAR}} \} \)
Example

- Query: \( P(\text{Flu} \mid \text{runny nose}) \)

- Eliminating S.
  1. \( \Phi' = \{\varphi_{SR}, \varphi_{FAS}\} \)
  2. \( \Psi = \{\varphi_{F}, \varphi_{A}\} \)
  3. \( \psi_{\text{FAR}} = \sum_{S} \varphi_{SR} \cdot \varphi_{FAS} \)
  4. Return \( \Psi \cup \{\psi_{\text{FAR}}\} \)
Example

• Query:
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Eliminating \( S \).
  1. \( \Phi' = \{ \varphi_{SR}, \varphi_{FAS} \} \)
  2. \( \Psi = \{ \varphi_F, \varphi_A \} \)
  3. \( \psi_{FAR} = \sum_S \varphi_{SR} \cdot \varphi_{FAS} \)
  4. Return \( \Psi \cup \{ \psi_{FAR} \} \)
Example

• Query:
P(Flu | runny nose)

• Finally, eliminate A.
Example

• Query: 
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Eliminating A.
  1. \( \Phi' = \{\varphi_A, \varphi_{\text{FAR}}\} \)
  2. \( \Psi = \{\varphi_F\} \)
  3. \( \psi_{\text{FR}} = \sum_A \varphi_A \cdot \psi_{\text{FAR}} \)
  4. Return \( \Psi \cup \{\psi_{\text{FR}}\} \)
Example

• Query:
P(Flu | runny nose)

• Eliminating A.
1. $\Phi' = \{\varphi_A, \varphi_{FAR}\}$
2. $\Psi = \{\varphi_F\}$
3. $\psi_{FR} = \sum_A \varphi_A \cdot \psi_{FAR}$
4. Return $\Psi \cup \{\psi_{FR}\}$
Markov Chain, Again

• Earlier, we eliminated A, then B, then C.
Now let’s start by eliminating C.
Markov Chain, Again

- Now let’s start by eliminating C.

\[
\begin{array}{c|c|c}
   & 0 & 1 \\
\hline
P(C | B) & & \\
0 & & \\
1 & & \\
\end{array}
\]

\[
\begin{array}{c|c|c}
   & 0 & 1 \\
\hline
P(D | C) & & \\
0 & & \\
1 & & \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
   B & C & D & \varphi'(B, C, D) \\
\hline
0 & 0 & 0 & \\
0 & 0 & 1 & \\
0 & 1 & 0 & \\
0 & 1 & 1 & \\
1 & 0 & 0 & \\
1 & 0 & 1 & \\
1 & 1 & 0 & \\
1 & 1 & 1 & \\
\end{array}
\]
Markov Chain, Again

• Now let’s start by eliminating C.

\[
\sum_C A B C D \varphi'(B, C, D) = \begin{array}{|c|c|c|}
\hline
B & C & D \varphi'(B, C, D) \\
\hline
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\hline
\end{array}
= \begin{array}{|c|c|}
\hline
B & D \psi(B, D) \\
\hline
0 & 0 \\
0 & 1 \\
1 & 0 \\
1 & 1 \\
\hline
\end{array}
\]
Markov Chain, Again

- Eliminating B will be similarly complex.

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
<th>$\psi(B, D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Variable Elimination: Comments

• Can prune away all non-ancestors of the query variables.
• Ordering makes a difference!
• Works for Markov networks and Bayesian networks.
  – Factors need not be CPDs and, in general, new factors won’t be.
What about Evidence?

• So far, we’ve just considered the posterior/marginal \( P(Y) \).
• Next: conditional distribution \( P(Y | X = x) \).
• It’s almost the same: the additional step is to *reduce* factors to respect the evidence.
Example

• Query: \( P(\text{Flu} \mid \text{runny nose}) \)

• Let’s reduce to \( R = \text{true} \) (runny nose).

<table>
<thead>
<tr>
<th>( P(R \mid S) )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

• Query: \( P(\text{Flu} \mid \text{runny nose}) \)

• Let’s reduce to \( R = \text{true} \) (runny nose).
Example

• Query: $P(\text{Flu} \mid \text{runny nose})$

• Let’s reduce to $R = \text{true (runny nose)}$. 

| P(R | S) | 0 | 1 |
|--------|---|---|
| 0      |   |   |
| 1      |   |   |

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>$\varphi_{SR} (S, R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>$\varphi'_{S} (S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Example

- Query: $P(\text{Flu} \mid \text{runny nose})$
- Let’s reduce to $R = \text{true (runny nose)}$.

<table>
<thead>
<tr>
<th>$P(R \mid S)$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S$</th>
<th>$R$</th>
<th>$\varphi_{SR} (S, R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S$</th>
<th>$R$</th>
<th>$\varphi'_{S} (S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Example

• Query:
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Let’s reduce to \( R = \text{true} \) (runny nose).

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>( \varphi'_S(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Example

• Query:  
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Now run variable elimination all the way down to one factor (for F).

H can be pruned for the same reasons as before.
Example

• Query: P(Flu | runny nose)

• Now run variable elimination all the way down to one factor (for F).
Example

• Query: 
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Now run variable elimination all the way down to one factor (for F).

Eliminate A.
Example

• Query: 
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Now run variable elimination all the way down to one factor (for \( F \)).

Take final product.
Example

• Query:
  \[ P(\text{Flu} \mid \text{runny nose}) \]

• Now run variable elimination all the way down to one factor.
Variable Elimination for Conditional Probabilities

Input: Graphical model on $V$, set of query variables $Y$, evidence $X = x$

Output: factor $\varphi$ and scalar $\alpha$

1. $\Phi = \text{factors in the model}$
2. Reduce factors in $\Phi$ by $X = x$
3. Choose variable ordering on $Z = V \setminus Y \setminus X$
4. $\varphi = \text{Variable-Elimination}(\Phi, Z)$
5. $\alpha = \sum_{z \in \text{Val}(Z)} \varphi(z)$
6. Return $\varphi, \alpha$
Note

• For Bayesian networks, the final factor will be $P(Y, X = x)$ and the sum $\alpha = P(X = x)$.
• This equates to a Gibbs distribution with partition function $= \alpha$. 
Variable Elimination

• In general, exponential requirements in induced width corresponding to the ordering you choose.
• It’s NP-hard to find the best elimination ordering.
• If you can avoid “big” intermediate factors, you can make inference linear in the size of the original factors.
  – Chordal graphs
  – Polytrees
Additional Comments

• Runtime depends on the size of the *intermediate* factors.

• Hence, variable elimination ordering matters a lot.
  – But it’s NP-hard to find the best one.
  – For MNs, *chordal graphs* permit inference in time linear in the size of the original factors.
  – For BNs, *polytree* structures do the same.
Getting Back to NLP

• Traditional structured NLP models were sometimes subconsciously chosen for these properties.
  – HMMs, PCFGs (with a little work)
  – But not: IBM model 3

• Need MAP inference for decoding!

• Need approximate inference for complex models!
From Marginals to MAP

• Replace factor marginalization steps with maximization.
  – Add bookkeeping to keep track of the maximizing values.

• Add a traceback at the end to recover the solution.

• This is analogous to the connection between the forward algorithm and the Viterbi algorithm.
  – Ordering challenge is the same.
Factor Maximization

- Given $X$ and $Y$ ($Y \notin X$), we can turn a factor $\varphi(X, Y)$ into a factor $\psi(X)$ via maximization:

$$\psi(X) = \max_Y \phi(X, Y)$$

- We can refer to this new factor by $\max_Y \varphi$. 
Factor Maximization

• Given $X$ and $Y$ ($Y \notin X$), we can turn a factor $\varphi(X, Y)$ into a factor $\psi(X)$ via maximization:

$$\psi(X) = \max_Y \varphi(X, Y)$$
Distributive Property

• A useful property we exploited in variable elimination:

\[ X \notin \text{Scope}(\phi_1) \implies \sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2 \]

• Under the same conditions, factor multiplication distributes over max, too:

\[ \max_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \max_X \phi_2 \]
Traceback

Input: Sequence of factors with associated variables: \((\psi_Z, ..., \psi_Z^k)\)

Output: \(z^*\)

• Each \(\psi_Z\) is a factor with scope including \(Z\) and variables eliminated \(after\) \(Z\).

• Work backwards from \(i = k\) to \(1\):
  – Let \(z_i = \arg \max_z \psi_{Z_i}(z, z_{i+1}, z_{i+2}, ..., z_k)\)

• Return \(z\)
About the Traceback

• No extra (asymptotic) expense.
  – Linear traversal over the intermediate factors.

• The factor operations for both sum-product VE and max-product VE can be generalized.
  – Example: get the K most likely assignments
Eliminating One Variable
(Max-Product Version with Bookkeeping)

Input: Set of factors $\Phi$, variable $Z$ to eliminate
Output: new set of factors $\Psi$

1. Let $\Phi' = \{\varphi \in \Phi \mid Z \in \text{Scope}(\varphi)\}$
2. Let $\Psi = \{\varphi \in \Phi \mid Z \notin \text{Scope}(\varphi)\}$
3. Let $\tau$ be $\max_Z \prod_{\varphi \in \Phi'} \varphi$
   - Let $\psi$ be $\prod_{\varphi \in \Phi'} \varphi$ (bookkeeping)
4. Return $\Psi \cup \{\tau\}, \psi$
Variable Elimination
(Max-Product Version with Decoding)

Input: Set of factors $\Phi$, ordered list of variables $Z$ to eliminate

Output: new factor

1. For each $Z_i \subseteq Z$ (in order):
   - Let $(\Phi, \psi_{Z_i}) = \text{Eliminate-One}(\Phi, Z_i)$

2. Return $\prod_{\varphi \in \Phi} \varphi$, Traceback($\{\psi_{Z_i}\}$)
Variable Elimination Tips

• Any ordering will be correct.
• Most orderings will be too expensive.
• There are heuristics for choosing an ordering (you are welcome to find them and test them out).
(Rocket Science: True MAP)

- Evidence: $X = x$
- Query: $Y$
- Other variables: $Z = V \setminus X \setminus Y$

\[
y^* = \arg \max_{y \in \text{Val}(Y)} P(Y = y \mid X = x)
= \arg \max_{y \in \text{Val}(Y)} \sum_{z \in \text{Val}(Z)} P(Y = y, Z = z \mid X = x)
\]

- First, marginalize out $Z$, then do MAP inference over $Y$ given $X = x$
- This is not usually attempted in NLP, with some exceptions.
Sketch of Gibbs Sampling

- MCMC: design (on paper) a graph where each configuration from Val(V) is a node.
  - Transitions in the graph designed to give a Markov chain whose stationary distribution is the posterior.
- Simulate a random walk in the graph.
- If you walk long enough, your position is distributed according to P(V).
Transitions in Gibbs Sampling

• A transition in the Markov chain equates to changing a subset of the random variables.

• Gibbs: resample $V_i$’s value according to $P(V_i \mid V \setminus \{V_i\})$.
  – Only need the local factors that affect $V_i$: take product, marginalize, and randomly choose new value.

• Simply lock evidence variables $X$.
• Maximizing version gradually shifts sampler in favor of most probable value for $V_i$. 
Sketch of Mean Field Variational Inference

• Inference with our distribution $P$ is hard.
• Choose an “easier” distribution family, $Q$. Then find:

$$\arg \min_{Q \in \mathcal{Q}} D(Q \| P)$$

• Usually iterative methods are required to “fit” $Q$ to $P$.
  – These often resemble familiar learning algorithms like EM!
Energy Functional

\[
D(Q(Y)||P(Y \mid X = x)) = \mathbb{E}_Q[\log Q(Y)] - \mathbb{E}_Q[\log P(Y \mid X = x)] \\
= -H(Q(Y)) - (\mathbb{E}_Q[\log P(X = x, Y)] - \log P(X = x)) \\
= -H(Q(Y)) - \left( \sum_{\phi} \mathbb{E}_Q[\log \phi_{\mid x}] - \log P(X = x) \right)
\]

\[
\underbrace{\log P(X = x)}_{\text{constant}} = D(Q(Y)||P(Y \mid X = x)) + H(Q(Y)) + \sum_{\phi} \mathbb{E}_Q[\log \phi_{\mid x}] \\
\text{maximize this}
\]

- Expectations under simpler distribution family, \( Q \).
  - Every element of \( Q \) is an approximate solution.
  - We try to find the best one.
Variational Methods

• This is a simple example.
• For any $\lambda$ and any $x$:

\[-\ln(x) \geq -\lambda x + \ln(\lambda) + 1\]

family of functions $g_\lambda(x)$
Variational Methods

• This is a simple example.
• For any $\lambda$ and any $x$:
  
  $-\ln(x) \geq -\lambda x + \ln(\lambda) + 1$

• Further, for any $x$, there is some $\lambda$ where the bound is tight.
  
  $-\lambda$ is called a variational parameter.
Tangent: Variational Methods

• This is a simple example.
• For any $\lambda$ and any $x$:
  
  $-\ln(x) \geq -\lambda x + \ln(\lambda) + 1$

• Further, for any $x$, there is some $\lambda$ where the bound is tight.
  
  $-\lambda$ is called a variational parameter.
Tangent: Variational Methods

- This is a simple example.
- For any $\lambda$ and any $x$:
  $$-\ln(x) \geq -\lambda x + \ln(\lambda) + 1$$
- Further, for any $x$, there is some $\lambda$ where the bound is tight.
  - $\lambda$ is called a variational parameter.
- For us, $\log P(X = x)$ is like $-\ln(x)$, and $Q$ is like $\lambda$. 
Structured Variational Approach

• Maximize the energy functional over a family $Q$ that is well-defined.
  – A graphical model!
  – Probably not an I-map for $P$. (Bound isn’t tight.)

• Simpler structures lead to easier inference.
  – Mean field is the simplest:

$$Q(V) = \prod_i Q_i(V_i)$$
Parting Shots

• You will probably never implement the general variable elimination algorithm.
• You will rarely use exact inference.

• There is value in understanding the problem that approximation methods are trying to solve, and what an exact (if intractable) solution would look like!