Exact Inference: Introduction

• Using a Bayesian network to compute probabilities is called inference
• In general, inference involves queries of the form:
  \[ P(X | E=e) \]
  - \( E \) = The evidence variable(s)
  - \( X \) = The query variable(s) (Assume a single variable for now)
Exact Inference: Introduction

- An example of a query would be:
  \[ P(\text{TastesGood} = \text{true} \mid \text{HasPepperoni} = \text{true}, \text{HasMushrooms} = \text{true}, \text{HasAnchovies} = \text{false}) \]
- Note: Even though \text{CookWashesHands} is in the Bayesian network, it is not given values in the query (i.e. they do not appear either as query variables or evidence variables)
- They are treated as unobserved variables

Recall that:

\[ P(X \mid E = e) = \alpha P(X, E = e) \]
\[ = \alpha \sum_{y} P(X, E = e, Y = y) \]

and \[ P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i)) \]

Enumeration-Ask algorithm:
Answer queries by computing sums of products of conditional probabilities from the network
Query: $P(A=true \mid B=true)$

How do you solve this? 2 steps:

1. Express it in terms of the joint probability distribution $P(A, B, C)$
2. Express the joint probability distribution in terms of the entries in the CPTs of the Bayes net

Whenever you see a conditional like $P(A=true \mid B=true)$, use the Chain Rule:

$$P(A = true \mid B = true) = \frac{P(A = true, B = true)}{P(B = true)} = \frac{P(A, B)}{P(B)}$$
**Exact Inference: Introduction**

Whenever you need to get a subset of the variables eg. $P(B, A)$ from the full joint distribution $P(A, B, C)$, use marginalization:

$$P(X) = \sum_y P(X, Y = y)$$

To express the joint probability distribution as the entries in the CPTs, use:

$$P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i | \text{Parents}(X_i))$$

\[
\sum_c P(A = \text{true}, B = \text{true}, C = c) \\
= \frac{\sum_c P(A = a, B = \text{true}, C = c)}{\sum_{a,c} P(A = a, B = \text{true}, C = c)} \\
= \frac{\sum_c P(C = c | A = \text{true})P(B = \text{true} | A = \text{true})P(A = \text{true})}{\sum_{a,c} P(C = c | A = a)P(B = \text{true} | A = a)P(A = a)}
\]
Exact Inference: Introduction

Take the probabilities that don’t depend on the terms in the summation and move them outside the summation.

\[
\sum_c p(C = c \mid A = \text{true}) p(B = \text{true} \mid A = \text{true}) p(A = \text{true}) \\
= \frac{\sum_{a,c} p(C = c \mid A = a) p(B = \text{true} \mid A = a) p(A = a)}{p(B = \text{true} \mid A = \text{true}) p(A = \text{true}) \sum_c p(C = c \mid A = a)}
\]

Simplify if possible.

\[
P(B = \text{true} \mid A = \text{true}) p(A = \text{true}) \sum_c p(C = c \mid A = \text{true}) \\
= \frac{\sum_a p(B = \text{true} \mid A = a) p(A = a) \sum_c p(C = c \mid A = a)}{p(B = \text{true} \mid A = a) p(A = a)}
\]
Exact Inference: Introduction

Exact Inference in graphical models is NP-hard
– Exponential time in worst case

Approximate inference is also NP-hard
– But this is in the worst case. In practice, it is much more efficient

Example #2 (Variable Elimination):

\[ P(B) = \sum_a P(A = a)P(B \mid A = a) \]

- **Note:** B is not instantiated with a value. We are computing the table \( P(B) \).
- If A has \( k \) values and B has \( k \) values, the number of arithmetic operations required is \( O(k^2) \)
- If the chain has \( n \) nodes, computing the joint probability \( P(X_1, ..., X_n) \) is \( O(nk^2) \)
- Naïve approach required \( O(k^n) \) operations
Exact Inference: Introduction

\[ P(D) = \sum_C \sum_B \sum_A P(A)P(B \mid A)P(C \mid B)P(D \mid C) \]

\[ = \sum_C P(D \mid C) \sum_B P(C \mid B) \sum_A P(A)P(B \mid A) \]

Use dynamic programming to work from the innermost summation outward.

\[ \psi_1(A, B) = P(A)P(B \mid A) \]
\[ \tau_1(B) = \sum_A \psi_1(A, B) \]
\[ \psi_2(B, C) = \tau_1(B)P(C \mid B) \]
\[ \tau_2(C) = \sum_B \psi_2(B, C) \]
Exact Inference: Introduction

Two key ideas to variable elimination:

1. Due to structure of BN, some subexpressions in the joint only depend on a small number of variables

2. Dynamic programming caches the intermediate results to avoid recomputing them exponentially many times

Variable Elimination
Variable Elimination

Recall:
• Let $\mathbf{X}$ be a set of random variables
• A factor $\phi$ is a function from $\text{Val}(\mathbf{X}) \rightarrow \mathbb{R}$
• The set of variables $\mathbf{X}$ is called the scope of the factor and denoted $\text{Scope}[\phi]$

We will be manipulating factors

Variable Elimination

Let $\mathbf{X}$ be a set of variables, and $Y \not\in \mathbf{X}$ a variable. Let $\phi(\mathbf{X}, Y)$ be a factor. We define the factor marginalization of $Y$ in $\phi$, denoted $\Sigma_Y \phi$ to be a factor $\psi$ over $\mathbf{X}$ such that:

$$\psi(\mathbf{X}) = \sum_Y \phi(\mathbf{X}, Y)$$

This operation is also called summing out of $Y$ in $\psi$
Variable Elimination

In a Bayesian network:
- Summing out all variables results in a factor with value 1

In a Markov network:
- Summing out all variables in the unnormalized distribution $\widetilde{P}_\phi$ defined by the product of factors in the Markov network results in the partition function

Note: we only sum up entries in the table where the values of X match up.
Variable Elimination

Recall: Let $X$, $Y$, and $Z$ be three disjoint sets of variables, and let $\phi_1(X,Y)$ and $\phi_2(Y,Z)$ be two factors. We define the factor product $\phi_1 \times \phi_2$ to be a factor $\psi$: $\text{Val}(X,Y,Z) \rightarrow \mathbb{R}$ as follows:

$$\psi(X,Y,Z) = \phi_1(X,Y) \phi_2(Y,Z)$$
Variable Elimination

Operations over factors:
• Addition is commutative: \( \sum_x \sum_y \phi = \sum_y \sum_x \phi \)

• Multiplication is commutative: \( \phi_1 \cdot \phi_2 = \phi_2 \cdot \phi_1 \)

• Products are associative: \( (\phi_1 \cdot \phi_2) \cdot \phi_3 = \phi_1 \cdot (\phi_2 \cdot \phi_3) \)

• Exchanging summations and products:

If \( X \not\in \text{Scope}[\phi_1] \), \( \sum_x (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_x \phi_2 \)  
(\( X \) is not in the terms of \( \phi_1 \))

Example:

\[
P(D) = \sum_A \sum_B \sum_C P(A, B, C, D)
\]

\[
= \sum_A \sum_B \sum_C \varphi_A \cdot \varphi_B \cdot \varphi_C \cdot \varphi_D
\]

\[
= \sum_C \sum_B \varphi_D \cdot \varphi_B \left( \sum_A \varphi_A \right)
\]

\[
= \varphi_D \sum_C \varphi_C \cdot \left( \sum_B \varphi_B \cdot \left( \sum_A \varphi_A \right) \right)
\]
Variable Elimination

The general problem involves a sum-product inference task:

\[ \sum_{Z} \prod_{\phi \in \Phi} \phi \]

Trick: Push in the summations as far as you can

Variable Elimination

Procedure Sum-Product-VE(
    \Phi, // A set of factors
    Z, // Set of variables to be eliminated
    < // Ordering on Z
)

1. Let Z₁, ..., Zₖ be an ordering of Z such that
2. Zᵢ < Zⱼ if and only if i < j
3. for i = 1, ..., k
4. \( \Phi \leftarrow \text{Sum-Product-Eliminate-Var}(\Phi, Zᵢ) \)
5. \( \phi^* \leftarrow \prod_{\phi \in \Phi} \)
6. Return \( \phi^* \)
Variable Elimination

Procedure Sum-Product-Eliminate-Var(
  \Phi, \quad // A set of factors
  Z, \quad // Variable to be eliminated
)
1. \Phi' \leftarrow \{ \phi \in \Phi : Z \in \text{Scope}[\phi] \}
2. \Phi'' \leftarrow \Phi - \Phi'
3. \psi \leftarrow \prod_{\phi \in \Phi'} \phi
4. \tau \leftarrow \sum_{z} \psi
5. Return \Phi'' \cup \{ \tau \}

Variable Elimination

Let X be some set of variables, and let \Phi be a set of factors such that for each \phi \in \Phi, \text{Scope}[\Phi] \subseteq X. Let Y \subseteq X be a set of query variables, and let Z = X – Y. Then for any ordering < over Z, \text{Sum-Product-VE}(\Phi, Z, <) returns a factor \phi^*(Y) such that

\phi^*(Y) = \sum_{Z} \prod_{\phi \in \Phi} \phi
Variable Elimination

Example: compute $P_B(Y)$ for Bayesian network $B$. Let:

- $\Phi = \{\phi_{X_i}\}_{i=1}^n$ where $\phi_{X_i} = P(X_i | Parents(X_i))$
- $Z = \{Z_1, \ldots, Z_m\} = \mathcal{X} - Y$ (eliminate all non-query variables)

Note: We can do the exact same thing on a Markov network except the final factor $\phi^*(Y)$ is unnormalized.

Variable Elimination

We will compute $P(J)$ using the elimination ordering $C, D, I, H, G, S, L$.

Note that:


$= \phi_C(C) \phi_D(D,C) \phi_I(I) \phi_G(G,I,D) \phi_S(S,I) \phi_L(L,G) \phi_J(J,L,S) \phi_H(H,G,J)$
Variable Elimination

Elimination ordering: C, D, I, H, G, S, L

1. Eliminating C:
   \[ \Psi_1(C, D) = \phi_C(C) \cdot \phi_D(D, C) \]
   \[ \tau_1(D) = \sum_C \Psi_1(C, D) \]

2. Eliminating D:
   \[ \Psi_2(G, I, D) = \phi_G(G, I, D) \cdot \tau_1(D) \]
   \[ \tau_2(G, I) = \sum_D \Psi_2(G, I, D) \]

3. Eliminating I:
   \[ \Psi_3(G, I, S) = \phi_I(I) \cdot \phi_S(S, I) \cdot \tau_2(G, I) \]
   \[ \tau_3(G, S) = \sum_I \Psi_3(G, I, S) \]

4. Eliminating H:
   \[ \Psi_4(G, J, H) = \phi_H(H, G, J) \]
   \[ \tau_4(G, J) = \sum_H \Psi_4(G, J, H) \]

Note: \( \tau_4 = 1 \) since \( \sum_H P(H|G, J) \). However, in this elimination ordering, you do need to generate this factor for the next step.
Variable Elimination

Elimination ordering: C, D, I, H, G, S, L

5. Eliminating G:
\[ \Psi_s(G, J, L, S) = \tau_4(G, J) \cdot \tau_5(G, S) \cdot \phi_L(L, G) \]
\[ \tau_5(J, L, S) = \sum_G \Psi_s(G, J, L, S) \]

6. Eliminating S:
\[ \Psi_s(J, L, S) = \tau_5(J, L, S) \cdot \phi_J(J, L, S) \]
\[ \tau_5(J, L) = \sum_S \Psi_s(J, L, S) \]

Variable Elimination

Elimination ordering: C, D, I, H, G, S, L

7. Eliminating L:
\[ \Psi_s(J, L) = \tau_5(J, L) \]
\[ \tau_5(J) = \sum_L \Psi_s(J, L) \]

Note: You can use any elimination ordering eg. G, I, S, L, H, C, D. This is a bad ordering because it produces factors with very large scope (see Table 9.2 pg 302)
Variable Elimination

Computing: $P(J)$

<table>
<thead>
<tr>
<th>Step</th>
<th>Variable Eliminated</th>
<th>Factors Used</th>
<th>Variables Involved</th>
<th>New Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>$\phi_C(C), \phi_D(D,C)$</td>
<td>C, D</td>
<td>$\tau_1(D)$</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>$\phi_G(G,I,D), \tau_1(D)$</td>
<td>G, I, D</td>
<td>$\tau_2(G,I)$</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$</td>
<td>G, S, I</td>
<td>$\tau_3(G,S)$</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
<td>$\phi_H(H,G,J)$</td>
<td>H, G, J</td>
<td>$\tau_4(G,J)$</td>
</tr>
<tr>
<td>5</td>
<td>G</td>
<td>$\tau_4(G,J), \tau_5(G,S), \phi_L(L,G)$</td>
<td>G, J, L, S</td>
<td>$\tau_5(J,L,S)$</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>$\tau_5(J,L,S), \phi_L(J, L, S)$</td>
<td>J, L, S</td>
<td>$\tau_6(J,L)$</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>$\tau_6(J,L)$</td>
<td>J, L</td>
<td>$\tau_7(J)$</td>
</tr>
</tbody>
</table>

Variable Elimination

How do we deal with evidence?
eg. $P(\ J = \text{true} \mid I = \text{high}, H = \text{false})$?

Note that:

$$P(J \mid I = \text{high}, H = \text{false}) = \frac{P(J, I = \text{high}, H = \text{false})}{P(I = \text{high}, H = \text{false})}$$
Variable Elimination

Recall Proposition 4.7:
Let $\mathcal{B}$ be a Bayesian network over $\mathcal{X}$ and $E = e$ an observation. Let $W = \mathcal{X} - E$. Then $P_{\mathcal{B}}(W | e)$ is a Gibbs distribution defined by the factors where

$$\phi_{X_i} = P_{\mathcal{B}}(X_i | \text{Parents}(X_i))[E = e]$$

The partition function for this Gibbs distribution is $P(e)$

This means we can sum out entries in the reduced factor $P[I=\text{high},H=\text{false}]$ then normalize with $P(I=\text{high},H=\text{false})$.

Variable Elimination

Procedure Cond-Prob-VE(
    $\mathcal{K}$, // A network over $\mathcal{X}$
    $Y$, // Set of query variables
    $E = e$ // Evidence
)
1. $\Phi \leftarrow$ Factors parameterizing K
2. Replace each $\phi \in \Phi$ by $\phi[\Theta=E=e]$
3. Select an elimination ordering $<$
4. $Z \leftarrow \mathcal{X} - Y - E$
5. $\phi^* \leftarrow \text{Sum-Product-VE}(\Phi, <, Z)$
6. $\alpha \leftarrow \sum_{y \in \text{Val}(Y)} \phi^*(y)$
7. return $\alpha, \phi^*$

Note: $\phi^*$ represents $P(Y,e)$ so divide $\phi^*$ by $\alpha$ to get $P(Y|e)$
# Variable Elimination

Computing: $P(J, I=\text{high}, H=\text{false})$

<table>
<thead>
<tr>
<th>Step</th>
<th>Variable Eliminated</th>
<th>Factors Used</th>
<th>Variables Involved</th>
<th>New Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1'</td>
<td>C</td>
<td>$\phi_C(C), \phi_D(D, C)$</td>
<td>C, D</td>
<td>$\tau_1'(D)$</td>
</tr>
<tr>
<td>2'</td>
<td>D</td>
<td>$\phi_G[I=\text{high}](G, D), \phi_1<a href="I">I=\text{high}</a>, \tau_1'(D)$</td>
<td>G, D</td>
<td>$\tau_2'(G)$</td>
</tr>
<tr>
<td>5'</td>
<td>G</td>
<td>$\tau_2'(G), \phi_L(L, G), \phi_H[H=\text{false}](G, J)$</td>
<td>G, L, J</td>
<td>$\tau_5'(J, L)$</td>
</tr>
<tr>
<td>6'</td>
<td>S</td>
<td>$\phi_S<a href="S">I=\text{high}</a>, \phi_J(J, L, S)$</td>
<td>J, L, S</td>
<td>$\tau_6'(J, L)$</td>
</tr>
<tr>
<td>7'</td>
<td>L</td>
<td>$\tau_6'(J, L), \tau_5'(J, L)$</td>
<td>J, L</td>
<td>$\tau_7'(J)$</td>
</tr>
</tbody>
</table>