Variable elimination is the dynamic programming variant of recursive conditioning.

Give a factorization, such as

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

it does the innermost sums first, constructing representations of the intermediate factors:

- \[\sum_A P(A)P(B \mid A)\] is a factor on \(B\); call it \(f_1(B)\).
- \[\sum_B P(C \mid B)f_1(B)\] is a factor on \(C\).

Lecture covers:

- Factors and factor arithmetic
- Variable elimination algorithm
A factor is a representation of a function from a tuple of random variables into a number.

We write factor $f$ on variables $X_1, \ldots, X_j$ as $f(X_1, \ldots, X_j)$.

You can assign some or all of the variables of a factor:

- $f(X_1 = v_1, X_2, \ldots, X_j)$, where $v_1 \in \text{domain}(X_1)$, is a factor on $X_2, \ldots, X_j$.
- $f(X_1 = v_1, X_2 = v_2, \ldots, X_j = v_j)$ is a number that is the value of $f$ when each $X_i$ has value $v_i$.

The former is also written as $f(X_1, X_2, \ldots, X_j)_{X_1 = v_1}$, etc.
Example factors

\[
\begin{array}{ccc|c}
X & Y & Z & \text{val} \\
\hline
\text{t} & \text{t} & \text{t} & 0.1 \\
\text{t} & \text{t} & \text{f} & 0.9 \\
\text{t} & \text{f} & \text{t} & 0.2 \\
\text{t} & \text{f} & \text{f} & 0.8 \\
\text{f} & \text{t} & \text{t} & 0.4 \\
\text{f} & \text{t} & \text{f} & 0.6 \\
\text{f} & \text{f} & \text{t} & 0.3 \\
\text{f} & \text{f} & \text{f} & 0.7 \\
\end{array}
\]

\[
r(X=t,Y,Z) = \begin{array}{ccc}
Y & Z & \text{val} \\
\hline
\text{t} & \text{t} & 0.1 \\
\text{t} & \text{f} & 0.9 \\
\text{f} & \text{t} & 0.2 \\
\text{f} & \text{f} & 0.8 \\
\end{array}
\]

\[
r(X=t,Y,Z) = \begin{array}{c}
Y \\
\hline
\text{t} & 0.9 \\
\text{f} & 0.8 \\
\end{array}
\]

\[
r(X=t,Y,Z) = 0.8
\]
The **product** of factor $f_1(X, Y)$ and $f_2(Y, Z)$, where $Y$ are the variables in common, is the factor $(f_1 * f_2)(X, Y, Z)$ defined by:

$$(f_1 * f_2)(X, Y, Z) = f_1(X, Y)f_2(Y, Z).$$
Multiplying factors example

\[ f_1: \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>0.1</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>0.9</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>0.2</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\[ f_2: \]

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>0.3</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>0.7</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>0.6</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>0.4</td>
</tr>
</tbody>
</table>

\[ f_1 * f_2: \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>0.03</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>f</td>
<td>0.07</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>0.54</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>0.36</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>t</td>
<td>0.06</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>0.14</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>t</td>
<td>0.48</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Summing out variables

We can sum out a variable, say $X_1$ with domain $\{v_1, \ldots, v_k\}$, from factor $f(X_1, \ldots, X_j)$, resulting in a factor on $X_2, \ldots, X_j$ defined by:

$$(\sum_{X_1} f)(X_2, \ldots, X_j)$$

$$= f(X_1 = v_1, \ldots, X_j) + \cdots + f(X_1 = v_k, \ldots, X_j)$$
Summing out a variable example

### $f_3$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>t t t</td>
<td>t t</td>
<td>t t t</td>
<td>0.03</td>
</tr>
<tr>
<td>t t f</td>
<td>t t f</td>
<td>t t f</td>
<td>0.07</td>
</tr>
<tr>
<td>t f t</td>
<td>t f t</td>
<td>t f t</td>
<td>0.54</td>
</tr>
<tr>
<td>t f f</td>
<td>t f f</td>
<td>t f f</td>
<td>0.36</td>
</tr>
<tr>
<td>f t t</td>
<td>f t t</td>
<td>f t t</td>
<td>0.06</td>
</tr>
<tr>
<td>f t f</td>
<td>f t f</td>
<td>f t f</td>
<td>0.14</td>
</tr>
<tr>
<td>f f t</td>
<td>f f t</td>
<td>f f t</td>
<td>0.48</td>
</tr>
<tr>
<td>f f f</td>
<td>f f f</td>
<td>f f f</td>
<td>0.32</td>
</tr>
</tbody>
</table>

### $\sum_B f_3$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$C$</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>t t</td>
<td>t t</td>
<td>0.57</td>
</tr>
<tr>
<td>t f</td>
<td>t f</td>
<td>0.43</td>
</tr>
<tr>
<td>f t</td>
<td>f t</td>
<td>0.54</td>
</tr>
<tr>
<td>f f</td>
<td>f f</td>
<td>0.46</td>
</tr>
</tbody>
</table>
To compute the posterior probability of query $Q$ given evidence $E = e$:

$$P(Q \mid E = e) = \frac{P(Q, E = e)}{P(E = e)} = \frac{P(Q, E = e)}{\sum_Q P(Q, E = e)}.$$

So the computation reduces to the probability of $P(Q, E = e)$ then normalize at the end.
Probability of a conjunction

- The variables of the belief network are $X_1, \ldots, X_n$.
- The evidence is $Y_1 = v_1, \ldots, Y_j = v_j$.
- To compute $P(Q, Y_1 = v_1, \ldots, Y_j = v_j)$:
  we add the other variables,
  $Z_1, \ldots, Z_k = \{X_1, \ldots, X_n\} - \{Q\} - \{Y_1, \ldots, Y_j\}$. and sum them out.
- We order the $Z_i$ into an elimination ordering.

$$P(Q, Y_1 = v_1, \ldots, Y_j = v_j)$$

$$= \sum_{Z_k} \cdots \sum_{Z_1} P(X_1, \ldots, X_n)_{Y_1 = v_1, \ldots, Y_j = v_j}.$$

$$= \sum_{Z_k} \cdots \sum_{Z_1} \prod_{i=1}^{n} P(X_i | parents(X_i))_{Y_1 = v_1, \ldots, Y_j = v_j}.$$
Computing sums of products

Computation in belief networks reduces to computing the sums of products.

- How can we compute $ab + ac$ efficiently?
- Distribute out $a$ giving $a(b + c)$
- How can we compute $\sum_{Z_1} \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i))$ efficiently?
- Distribute out those factors that don’t involve $Z_1$. 
Variable elimination algorithm

To compute $P(Q \mid Y_1 = v_1 \land \ldots \land Y_j = v_j)$:

- Construct a factor for each conditional probability.
- Set the observed variables to their observed values.
- Sum out each of the non-observed non-query variables (the \{Z_1, \ldots, Z_k\}) according to some elimination ordering.
- Multiply the remaining factors.
- Normalize by dividing the resulting factor $f(Q)$ by $\sum_Q f(Q)$. 
Summing out a variable

To sum out a variable $Z_j$ from a product $f_1, \ldots, f_k$ of factors:

- Partition the factors into
  - those that don’t contain $Z_j$, say $f_1, \ldots, f_i$,
  - those that contain $Z_j$, say $f_{i+1}, \ldots, f_k$

Then:

$$
\sum_{Z_j} f_1 \cdot \ldots \cdot f_k = f_1 \cdot \ldots \cdot f_i \left( \sum_{Z_j} f_{i+1} \cdot \ldots \cdot f_k \right).
$$

- Explicitly construct a representation of the rightmost factor.
  Replace the factors $f_{i+1}, \ldots, f_k$ by the new factor.
Example

\[
P(E | g) = \frac{P(E \land g)}{\sum_E P(E \land g)}
\]

\[
P(E \land g)
= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B | AC)P(C)P(D | C)P(E | B)P(F | E)P(g | ED)
= \left(\sum_F P(F | E)\right)
\]
\[
\sum_B P(E | B) \sum_C \left( P(C) \left( \sum_A P(A)P(B | AC) \right) \right)
\left( \sum_D P(D | C)P(g | ED) \right)
\]
Variable Elimination example

Query: $P(G \mid f)$; elimination ordering: $A, H, E, D, B, C$

$$P(G \mid f) \propto \sum_C \sum_B \sum_D \sum_E \sum_H \sum_A P(A)P(B \mid A)P(C \mid B)$$

$$P(D \mid C)P(E \mid D)P(f \mid E)P(G \mid C)P(H \mid E)$$

$$= \sum_C \left( \sum_B \left( \sum_A P(A)P(B \mid A) \right) P(C \mid B) \right) P(G \mid C)$$

$$\left( \sum_D P(D \mid C) \left( \sum_E P(E \mid D)P(f \mid E) \sum_H P(H \mid E) \right) \right)$$
Pruning Irrelevant Variables (Belief networks)

Suppose you want to compute $P(X \mid e_1 \ldots e_k)$:

- Prune any variables that have no observed or queried descendents.
- Connect the parents of any observed variable.
- Remove arc directions.
- Remove observed variables.
- Remove any variables not connected to $X$ in the resulting (undirected) graph.
Variable elimination is the dynamic programming variant of recursive conditioning.

Recursive Conditioning is the search variant of variable elimination.

They do the same additions and multiplications.

Space and time complexity \(O(nd^t)\), for \(n\) variables, domain size \(d\), and treewidth \(t\).

– treewidth is the number of variables in the smallest factor. It is a property of the graph and the elimination ordering.

Recursive conditioning never modifies or creates factors; it only evaluates them.