Exact Inference: Variable Elimination

40-956 Advanced Topics in AI: Probabilistic Graphical Models
Sharif University of Technology

Soleymani
Spring 2015
Why we need inference

- If we know the graphical model, we use the inference to find marginal or conditional distributions efficiently.

- We also need inference in the learning when we try to find a model from incomplete data or when the learning approach is Bayesian (as we will see in the next lectures).
Inference query

- **Likelihood: probability of evidence**
  \[
P(e) = \sum_{X} P(X, e)
  \]
  Nodes: \( X = \{X_1, \ldots, X_n\} \)
  \( e \): evidence on a set variables \( E \)
  \( X = X - E \)
  \( Y = X - Z \)

- **Marginal probability:**
  \[
P(X) = \sum_{X-X} P(X)
  \]

- **Conditional probability (a posteriori belief):**
  \[
P(X|e) = \frac{P(X, e)}{\sum_{x} P(X, e)}
  \]

- **Marginalized conditional probability:**
  \[
P(Y|e) = \frac{\sum_{Z} P(Y, Z, e)}{\sum_{Y} \sum_{Z} P(Y, Z, e)} \quad (X = Y \cup Z)
  \]
Marginal probability: Enumeration

- Marginal probability: $P(Y|e) \propto P(Y, e)$

- $P(Y, e) = \sum_Z P(Y, e, Z)$

Marginal probability: exponential computation is required in general
- #P-complete problem (enumeration intractable)
  - Even in the graph of polynomial size it can be exponential
- We cannot find a general procedure that works efficiently for arbitrary GMs
  - However, it can usually be improved by re-using calculations
- For special graph structure, provably efficient algorithms (avoiding exponential cost) are available
Most Probable Assignment (MPA)

Most probable assignment for some variables of interest given an evidence \( E = e \)

\[
y^*|e = \arg\max_Y P(Y|e)
\]

Maximum a posteriori configuration of \( Y \)

Applications of MPA
- Classification
  - find most likely label, given the evidence
- Explanation
  - what is the most likely scenario, given the evidence
Exact inference

- **Exact inference:**
  - **Variable elimination** algorithm
    - general graph
    - one query
  - **Belief propagation**, sum-product on factor graphs
    - Tree
    - marginal probability on all nodes
  - **Junction tree** algorithm
    - general graph
    - marginal probability on all clique nodes
Inference on a chain

\[ P(d) = \sum_a \sum_b \sum_c P(a, b, c, d) \]

\[ P(d) = \sum_a \sum_b \sum_c P(a)P(b|a)P(c|b)P(d|c) \]

- A naïve summation needs to enumerate over an exponential number of terms
Inference on a chain: marginalization and elimination

\[ P(d) = \sum_a \sum_b \sum_c P(a)P(b|a)P(c|b)P(d|c) \]

\[ = \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c) \]

\[ = \sum_c P(d|c) \sum_b P(c|b) \sum_a P(a)P(b|a) \]

\[ P(b) \]

\[ P(c) \]

\[ P(d) \]

- In a chain of \( n \) nodes each having \( k \) values, \( O(nk^2) \) instead of \( O(k^n) \)
Inference on a chain

- In both directed and undirected graphical models, the joint probability is a factored expression over subsets of the variables

\[
P(x) = \frac{1}{\mathcal{Z}} \phi_{1,2}(x_1, x_2) \phi_{2,3}(x_2, x_3) \ldots \phi_{N-1,N}(x_{N-1}, x_N) \quad \text{undirected}
\]

\[
P(x_i) = \frac{1}{\mathcal{Z}} \sum_{x_1} \ldots \sum_{x_{i-1}} \sum_{x_{i+1}} \ldots \sum_{x_N} \phi_{1,2}(x_1, x_2) \ldots \phi_{N-1,N}(x_{N-1}, x_N)
\]

\[
P(x_i) = \left[ \sum_{x_{i-1}} \phi(x_{i-1}, x_i) \sum_{x_{i-2}} \phi(x_{i-2}, x_{i-1}) \ldots \sum_{x_1} \phi(x_1, x_2) \right] \times \left[ \sum_{x_{i+1}} \phi(x_i, x_{i+1}) \sum_{x_{i+2}} \phi(x_{i+1}, x_{i+2}) \ldots \sum_{x_N} \phi(x_{N-1}, x_N) \right]
\]

\[O(|Val(X_j)| \times |Val(X_{j+1})|)\] operations in each elimination
Inference on a chain: improvement reasons

- Computing an expression of the form (sum-product inference):
  \[ \sum_{Z} \prod_{\phi \in \Phi} \phi \]
  \( \Phi \): the set of factors

- We used the structure of BN to factorize the joint distribution and thus the scope of the resulted factors will be limited.

- Distributive law: If \( X \not\in \text{Scope}(\phi_1) \) then \( \sum_{X} \phi_1 \cdot \phi_2 = \phi_1 \cdot \sum_{X} \phi_2 \)
  - Performing the summations over the product of only a subset of factors

- We find sub-expressions that can be computed once and then we save and reuse them in later computations
  - Instead of computing them exponentially many times
Variable elimination algorithm for sum-product inference

- Sum out each variable one at a time
  - all factors containing that variable are (removed from the set of factors and) multiplied to generate a product factor
  - The variable is summed out from the generated product factor and a new factor is obtained
  - The new factor is added to the set of the available factors

The resulted factor does not necessarily correspond to any probability or conditional probability in the network
**Procedure** Sum-Product-VE (Z, // Set of variables to be eliminated \( \mathcal{K} \), // network over \( X \))

\( \Phi \leftarrow \text{the factors parametrizing} \ \mathcal{K} \)

Select an elimination order \( Z_1, \ldots, Z_K \) for \( Z \) (i.e., \( Z_i < Z_j \) iff \( i < j \))

for \( i = 1, \ldots, K \)

\( \Phi \leftarrow \text{Sum-Product-Elim-Var}(\Phi, Z_i) \)

\( \phi^* \leftarrow \prod_{\phi \in \Phi} \phi \)

\( \alpha \leftarrow \sum_{x \in \text{Val}(X)} \phi^*(x) \)

Return \( \alpha, \phi^* \)

---

**Procedure** Sum-Product-Elim_Var(\( \Phi \), // the set of factors \( Z \) // the variable to be eliminated)

\( \Phi' \leftarrow \{\phi \in \Phi : Z \in \text{Scope}(\phi)\} \)

\( \Phi'' \leftarrow \Phi - \Phi' \)

\( m \leftarrow \sum_{Z} \prod_{\phi \in \Phi'} \phi \)

return \( \Phi'' \cup \{m\} \)

- Move all irrelevant factors (to the variable that must be eliminated now) outside of the summation
- Perform sum, getting a new term
- Insert the new term into the product
Procedure Cond-Prob-VE (\( \mathcal{K}, // \) the network over \( X \n Y, // \) Set of query variables \( E = e, // \) evidence)

\( \Phi \leftarrow \) the factors parametrizing \( \mathcal{K} \)

Replace each \( \phi \in \Phi \) by \( \phi[E = e] \)

Select an elimination order \( Z_1, \ldots, Z_K \) for \( Z = X - Y - E \) (i.e., \( Z_i < Z_j \) iff \( i < j \))

for \( i = 1, \ldots, k \)

\( \Phi \leftarrow \) Sum-Product-Elim-Var(\( \Phi, Z_i \))

\( \phi^* \leftarrow \prod_{\phi \in \Phi} \phi \)

\( \alpha \leftarrow \sum_{y \in Val(Y)} \phi^*(y) \)

Return \( \alpha, \phi^* \)
Directed example

- Query: \( P(X_2 | X_7 = \bar{x}_7) \)

- \( P(X_2 | \bar{x}_7) \propto P(X_2, \bar{x}_7) \)

\[
P(x_2, \bar{x}_7) = \sum_{x_1} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \sum_{x_8} P(x_1, x_2, x_3, x_4, x_5, x_6, \bar{x}_7, x_8)
\]

Consider the elimination order \( X_1, X_3, X_4, X_5, X_6, X_8 \)

\[
P(x_2, \bar{x}_7) = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} \sum_{x_1} P(x_1)P(x_2)P(x_3 | x_1, x_2)P(x_4 | x_3)P(x_5 | x_2)P(x_6 | x_3, \bar{x}_7)P(\bar{x}_7 | x_4, x_5)P(x_8 | \bar{x}_7)
\]
\[ P(x_2, \bar{x}_7) = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} P(x_2)P(x_4|x_3)P(x_5|x_2)P(x_6|x_3, \bar{x}_7)P(\bar{x}_7|x_4, x_5)P(x_8|\bar{x}_7) \sum_{x_1} P(x_1)P(x_3|x_1, x_2) \]

\[ = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} P(x_2)P(x_4|x_3)P(x_5|x_2)P(x_6|x_3, \bar{x}_7)P(\bar{x}_7|x_4, x_5)P(x_8|\bar{x}_7) m_1(x_2, x_3) \]

\[ = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} P(x_2)P(x_5|x_2)P(\bar{x}_7|x_4, x_5)P(x_8|\bar{x}_7) \sum_{x_3} P(x_4|x_3)P(x_6|x_3, \bar{x}_7) m_1(x_2, x_3) \]

\[ = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} P(x_2)P(x_5|x_2)P(\bar{x}_7|x_4, x_5)P(x_8|\bar{x}_7) m_3(x_2, x_6, x_4) \]

\[ = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} P(x_2)P(x_5|x_2)P(x_8|\bar{x}_7) \sum_{x_4} P(\bar{x}_7|x_4, x_5) m_3(x_2, x_6, x_4) \]

\[ = \sum_{x_8} \sum_{x_6} \sum_{x_5} P(x_2)P(x_5|x_2)P(x_8|\bar{x}_7) m_4(x_2, x_5, x_6) \]

\[ = \sum_{x_8} \sum_{x_6} P(x_2)P(x_8|\bar{x}_7) \sum_{x_5} P(x_5|x_2) m_4(x_2, x_5, x_6) \]

\[ = \sum_{x_8} \sum_{x_6} P(x_2)P(x_8|\bar{x}_7) m_5(x_2, x_6) \]

\[ = \sum_{x_8} P(x_2)P(x_8|\bar{x}_7) \sum_{x_6} m_5(x_2, x_6) \]

\[ = \left( \sum_{x_8} P(x_2)P(x_8|\bar{x}_7) \right) m_6(x_2) = m_8(x_2)m_6(x_2) \]
Conditional probability

\[
P(x_2 | \bar{x}_7) = \frac{m_8(x_2)m_6(x_2)}{\sum_{x_2} m_8(x_2)m_6(x_2)}
\]
Undirected example

- Query: \( P(X_2|X_7 = \bar{x}_7) \)

- \( P(X_2|\bar{x}_7) \propto P(X_2, \bar{x}_7) \)

\[
P(x_2, \bar{x}_7) = \sum_{x_1} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \sum_{x_8} P(x_1, x_2, x_3, x_4, x_5, x_6, \bar{x}_7, x_8)
\]

Consider the elimination order \( X_1, X_3, X_4, X_5, X_6, X_8 \)

\[
P(x_2, \bar{x}_7) = \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} \sum_{x_1} \phi(x_3, x_4)\phi(x_2, x_5)\phi(x_3, x_6, \bar{x}_7)\phi(x_4, x_5, \bar{x}_7)\phi(\bar{x}_7, x_8)\phi(x_1, x_2, x_3)
\]
\[ \begin{align*}
P(x_2, \bar{x}_7) &= \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} \phi(x_3, x_4) \phi(x_2, x_5) \phi(x_3, x_6, \bar{x}_7) \phi(x_4, x_5, \bar{x}_7) \phi(\bar{x}_7, x_8) \sum_{x_1} \phi(x_1, x_2, x_3) \\
&= \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \sum_{x_3} \phi(x_3, x_4) \phi(x_2, x_5) \phi(x_3, x_6, \bar{x}_7) \phi(x_4, x_5, \bar{x}_7) \phi(\bar{x}_7, x_8) m_1(x_2, x_3) \\
&= \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \phi(x_2, x_5) \phi(x_4, x_5, \bar{x}_7) \phi(\bar{x}_7, x_8) \sum_{x_3} \phi(x_3, x_4) \phi(x_3, x_6, \bar{x}_7) m_1(x_2, x_3) \\
&= \sum_{x_8} \sum_{x_6} \sum_{x_5} \sum_{x_4} \phi(x_2, x_5) \phi(x_4, x_5, \bar{x}_7) \phi(\bar{x}_7, x_8) m_3(x_2, x_6, x_4) \\
&= \sum_{x_8} \sum_{x_6} \sum_{x_5} \phi(x_2, x_5) \phi(\bar{x}_7, x_8) \sum_{x_4} \phi(x_4, x_5, \bar{x}_7) m_3(x_2, x_6, x_4) \\
&= \sum_{x_8} \sum_{x_6} \sum_{x_5} \phi(x_2, x_5) \phi(\bar{x}_7, x_8) m_4(x_2, x_5, x_6) \\
&= \sum_{x_8} \sum_{x_6} \phi(\bar{x}_7, x_8) \sum_{x_5} \phi(x_2, x_5) m_4(x_2, x_5, x_6) \\
&= \sum_{x_8} \sum_{x_6} \phi(\bar{x}_7, x_8) m_5(x_2, x_6) \\
&= \sum_{x_8} \phi(\bar{x}_7, x_8) \sum_{x_6} m_5(x_2, x_6) \\
&= \left( \sum_{x_8} \phi(\bar{x}_7, x_8) \right) m_6(x_2)
\end{align*} \]
Complexity of variable elimination algorithm

- In each elimination step:
  - \( f(x, x_1, \ldots, x_k) = \prod_{i=1}^{M} g_i(x, x_{c_i}) \)
  - \( \sum_x f(x, x_1, \ldots, x_k) \)

- We need:
  - \( M \times |Val(X)| \times \prod_{i=1}^{k} |Val(X_i)| \) multiplications
    - For each tuple \( x, x_1, \ldots, x_k \), we need \( M - 1 \) multiplications
  - \( |Val(X)| \times \prod_{i=1}^{k} |Val(X_i)| \) additions
    - For each tuple \( x_1, \ldots, x_k \), we need \( Val(X) \) additions

Complexity is exponential in number of variables in the intermediate factor

Size of the created factors is the dominant quantity in the complexity of VE
Graph elimination

- Graph elimination is a simple unified treatment of inference algorithms in both directed and undirected models
  - convert directed models to undirected ones (moralization)

- Graph-theoretic property: the factors resulted during variable elimination are captured by recording the elimination clique

- The computational complexity of the Eliminate algorithm can be reduced to purely graph-theoretic considerations
Graph elimination

- Begin with the undirected GM or moralized BN
- Choose an elimination ordering (query nodes should be last)
- Eliminate a node from the graph and add edges (called fill edges) between all pairs of its neighbors
- Iterate until all non-query nodes are eliminated
Graph elimination

Summation $\Leftrightarrow$ elimination
Intermediate term $\Leftrightarrow$ elimination clique

Removing a node from the graph and connecting the remaining neighbors
Graph elimination: elimination cliques

- Induced dependency during marginalization is captured in elimination cliques

- A correspondence between maximal cliques in the induced graph and maximal factors generated in VE algorithm
  - The complexity depends on the number of variables in the largest elimination clique

- The size of the maximal elimination clique in the induced graph depends on the elimination ordering
Elimination order: example

Ordering $\prec$: $X_1, X_3, X_4, X_5, X_6$

Ordering $\prec$: $X_4, X_3, X_5, X_6, X_1$
Elimination order

- Finding the best elimination ordering is NP-hard
  - Equivalent to finding the tree-width in the graph that is NP-hard
    - **Tree-width**: one less than the smallest achievable size of the largest elimination clique, ranging over all possible elimination ordering

- Good elimination orderings lead to **small cliques** and thus reduce complexity

- What is the optimal order for trees?
Heuristics for finding an ordering

- How can we find an ordering that induces a “small” graph?
- Some heuristics to select the next node for elimination:
  - Min-neighbors
    - The cost of a vertex is the number of neighbors it has in the current graph
  - Min-fill
    - The cost of a vertex is the number of edges that need to be added to the graph due to its elimination.
  - Min-weight
  - Weighted-min-fill
Elimination algorithm: summary

- Elimination algorithm computes the marginal probability for one query.

- It uses the factorization properties and distributive law to compute the marginal probabilities more efficiently.
  - reorder computations
  - save intermediate terms

- Elimination order affects the computational complexity. However, finding the best order in general is NP-hard.