Probabilistic Graphical Models

Brown University CSCI 2950-P, Spring 2013
Prof. Erik Sudderth

Lecture 10:
Triangulation and Junction Tree Algorithms

Some figures courtesy Michael Jordan’s draft textbook,
An Introduction to Probabilistic Graphical Models
Inference in Undirected Graphs

\[
p(x_1, \bar{x}_6) = \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \psi(x_1, x_2) \psi(x_1, x_3) \psi(x_2, x_4) \psi(x_3, x_5) \psi(x_2, x_5, x_6) \delta(x_6, \bar{x}_6)
\]

\[
= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) \sum_{x_6} \psi(x_2, x_5, x_6) \delta(x_6, \bar{x}_6)
\]

\[
= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) m_6(x_2, x_5)
\]

\[
= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \sum_{x_4} \psi(x_2, x_4)
\]

\[
= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3)
\]

\[
= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) m_3(x_1, x_2) = \frac{1}{Z} m_2(x_1)
\]
A Graph Elimination Algorithm

Algebraic Marginalization Operations

• Marginalize out the variable associated with sum node
• Compute a new potential table involving all other variables which depend on the just-marginalized variable

Graph Manipulation Operations

• Remove, or eliminate, a single node from the graph
• Add edges (if they don’t already exist) between all pairs of nodes who were neighbors of the just-removed node

A Graph Elimination Algorithm

• Choose an elimination ordering (query nodes should be last)
• Eliminate a node, remove its incoming edges, add edges between all pairs of its neighbors
• Iterate until all non-query nodes are eliminated
Graph Elimination Example

Elimination Order: (6,5,4,3,2,1)
Graph Elimination Example

*Elimination Order: (6, 5, 4, 3, 2, 1)*
The *clique tree* contains the cliques (fully connected subsets) which are generated as elimination executes.

The *separator sets* contain the variables which are shared among each linked pair of cliques.
Elimination for Trees

\[
p_1(x_1) = \sum_{x_2, x_3, x_4} \psi_1(x_1) \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4)
\]

\[
= \psi_1(x_1) \sum_{x_2, x_3, x_4} \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4)
\]
Elimination for Trees

\[ p_1(x_1) = \sum_{x_2, x_3, x_4} \psi_1(x_1) \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]

\[ = \psi_1(x_1) \sum_{x_2, x_3, x_4} \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]

\[ = \psi_1(x_1) \sum_{x_2} \psi_{12}(x_1, x_2) \psi_2(x_2) \sum_{x_3, x_4} \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]
Elimination for Trees

\[ p_1(x_1) = \sum_{x_2, x_3, x_4} \psi_1(x_1) \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]

\[ = \psi_1(x_1) \sum_{x_2, x_3, x_4} \psi_{12}(x_1, x_2) \psi_2(x_2) \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]

\[ = \psi_1(x_1) \sum_{x_2} \psi_{12}(x_1, x_2) \psi_2(x_2) \sum_{x_3, x_4} \psi_{23}(x_2, x_3) \psi_3(x_3) \psi_{24}(x_2, x_4) \psi_4(x_4) \]

\[ = \psi_1(x_1) \sum_{x_2} \psi_{12}(x_1, x_2) \psi_2(x_2) \left[ \sum_{x_3} \psi_{23}(x_2, x_3) \psi_3(x_3) \right] \cdot \left[ \sum_{x_4} \psi_{24}(x_2, x_4) \psi_4(x_4) \right] \]

\[ m_{21}(x_1) = \sum_{x_2} \psi_{12}(x_1, x_2) \psi_2(x_2) m_{32}(x_2) m_{42}(x_2) \]

\[ m_{32}(x_2) \]

\[ m_{42}(x_2) \]
Belief Propagation (Sum-Product)

**BELIEFS:** Posterior marginals

\[
\hat{p}_t(x_t) \propto \psi_t(x_t) \prod_{u \in \Gamma(t)} m_{ut}(x_t)
\]

\(\Gamma(t)\) → neighborhood of node t (adjacent nodes)

**MESSAGES:** Sufficient statistics

\[
m_{ts}(x_s) \propto \sum_{x_t} \psi_{st}(x_s, x_t) \psi_t(x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t)
\]

I) Message Product

II) Message Propagation
### Undirected Inference Algorithms

#### One Marginal
- **Tree:** Elimination applied to leaves of the tree
- **Graph:** Elimination algorithm

#### All Marginals
- **Tree:** Belief propagation or sum-product algorithm
- **Graph:** Junction tree algorithm: belief propagation on a junction tree

- For directed models, first convert to undirected factor graph form (moralization)
- A *junction tree* is a clique tree with special properties
Undirected Graphical Models

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \]

- Parameterization exactly captures those non-degenerate distributions which are Markov with respect to this graph
- For now, we will assume that potentials are restricted to maximal cliques

\[ C \rightarrow \text{set of maximal cliques (fully connected subsets) of nodes} \]
\[ E \rightarrow \text{set of undirected edges } (s,t) \text{ linking pairs of nodes} \]
\[ V \rightarrow \text{set of } N \text{ nodes or vertices, } \{1, 2, \ldots, N\} \]
\[ Z \rightarrow \text{normalization constant (partition function)} \]
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \]

\[ z_c = \{ x_s \mid s \in c \}, \ c \in C \]

\[ p(z) \propto \prod_{c \in C} \psi_c(z_c) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates joint configurations of dependent variables
- Does this define an equivalent joint distribution?

**PROBLEM:** We have defined multiple copies of the variables in the true model, but not enforced any relationships among them.
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \]
\[ z_c = \{ x_s \mid s \in c \}, \quad c \in C \]
\[ p(z) \propto \prod_{c \in C} \psi_c(z_c) \prod_{d \neq c} \psi_{cd}(z_c, z_d) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates joint configurations of dependent variables
- Add potentials enforcing consistency between all pairs of clique variables which share one of the original variables:
\[ \psi_{cd}(z_c, z_d) = \begin{cases} 1 & z_c = z_d \text{ for all } x_s, s \in c \cap d \\ 0 & \text{otherwise} \end{cases} \]

**PROBLEM:** The graph may have a large number of pairwise consistency constraints, and inference will be difficult
Clique-Based Inference Algorithms

\[ p(x) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \]

\[ z_c = \{x_s \mid s \in c\}, \ c \in C \]

\[ p(z) \propto \prod_{c \in C} \psi_c(z_c) \prod_{(c,d) \in E(C)} \psi_{cd}(z_c, z_d) \]

- For each clique \( c \), define a variable \( z_c \) which enumerates \textit{joint} configurations of dependent variables
- Add potentials enforcing consistency between some subset of pairs of cliques, taking advantage of transitivity of equality:

\[ x_a = x_b, \ x_b = x_c \rightarrow x_a = x_c \]

**Question:** How many edges are needed for global consistency? When can we build a tree-structured clique graph?
Clique Trees and Junction Trees

- This clique tree has the *junction tree* property: the clique nodes containing any variable from the original model form a *connected* subtree.
- We can exactly represent the distribution *ignoring redundant constraints*.
- Note that not all clique trees are junction trees:

\[ X_1 \ X_2 \quad X_1 \ X_2 \ X_3 \quad X_2 \ X_3 \ X_5 \quad X_2 \ X_4 \quad X_2 \ X_5 \ X_6 \quad X_2 \ X_3 \ X_5 \]
Finding a Junction Tree

The junction tree property. A clique tree possesses the junction tree property if for every pair of cliques $V$ and $W$, all cliques on the (unique) path between $V$ and $W$ contain $V \cap W$.

- Given a set of cliques, how can we efficiently find a clique tree with the junction tree (running intersection) property?
- How can we be sure that at least one junction tree exists?
- Strategy: Augment the graph with additional edges
  - Cliques of original graph are always subsets of cliques of the augmented graph, so original distribution still factorizes appropriately
  - As cliques grow, will eventually be able to construct a junction tree

**Question:** Which undirected graphs have junction trees?
The junction tree property. A clique tree possesses the junction tree property if for every pair of cliques $V$ and $W$, all cliques on the (unique) path between $V$ and $W$ contain $V \cap W$.

- A chord is an edge connecting two non-adjacent nodes in some cycle
- A cycle is chordless if it contains no chords
- A graph is triangulated if it contains no chordless cycles

**Theorem:** The maximal cliques of a graph have a corresponding junction tree if and only if that undirected graph is triangulated

**Lemma 2** Let $G = (V, E)$ be a noncomplete triangulated graph with at least three nodes. Then there exists a decomposition of $V$ into disjoint sets $A$, $B$ and $S$ such that $S$ separates $A$ and $B$ and $S$ is complete.

- Key induction argument in constructing junction tree from triangulation
- Implies existence of elimination ordering which introduces no new edges
Constructing a Junction Tree

**Theorem:** A clique tree is a junction tree if and only if it is a maximal spanning tree of the weighted clique intersection graph

- **Graph:** Fully connected with nodes corresponding to maximal cliques
- **Edge weights:** Cardinality of separator set (intersection) of cliques
- **Computational complexity:** Quadratic in number of maximal cliques

**Junction Tree Algorithms for General-Purpose Inference**

1. Triangulate the target undirected graphical model
   - Any elimination ordering generates a valid triangulation
   - Optimal triangulation is NP-hard (in multiple ways)
2. Arrange triangulated cliques into a junction tree
3. Execute variant of sum-product algorithm on junction tree
Sum-Product for Junction Trees

\[ m_{ts}(x_s) \propto \sum_{x_t} \psi_{st}(x_s, x_t) \psi_t(x_t) \prod_{u \in \Gamma(t) \setminus s} m_{ut}(x_t) \]

Consider a junction tree linking a set of cliques, with pairwise equality constraints among intersections:

Messages are functions of the separating sets (variables shared among cliques):

\[ \mu_{ji}(x_{S_{ji}}) \propto \sum_{x_{R_j}} \psi_{C_j}(x_{C_j}) \prod_{k \neq j} \mu_{kj}(x_{S_{kj}}) \]

\[ R_j = C_j \setminus S_{ij} \]

Shafer-Shenoy Junction Tree Algorithm
Undirected Graphical Models

\[ p(x \mid \theta) = \frac{1}{Z(\theta)} \prod_{f \in \mathcal{F}} \psi_f(x_f \mid \theta_f) \]

\[ Z(\theta) = \sum_{x} \prod_{f \in \mathcal{F}} \psi_f(x_f \mid \theta_f) \]

\[ \mathcal{F} \rightarrow \text{set of hyperedges linking subsets of nodes } f \subseteq \mathcal{V} \]

\[ \mathcal{V} \rightarrow \text{set of } N \text{ nodes or vertices, } \{1, 2, \ldots, N\} \]

- Assume an exponential family representation of each factor:

\[ p(x \mid \theta) = \exp \left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_f) - A(\theta) \right\} \]

\[ \psi_f(x_f \mid \theta_f) = \exp \{ \theta_f^T \phi_f(x_f) \} \quad A(\theta) = \log Z(\theta) \]

- Partition function \textit{globally} couples the local factor parameters
Learning for Undirected Models

- Undirected graph encodes dependencies within a single training example:
  \[
p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} \frac{1}{Z(\theta)} \prod_{f \in \mathcal{F}} \psi_f(x_f,n \mid \theta_f) \quad \mathcal{D} = \{x_\mathcal{V},1, \ldots, x_\mathcal{V},N\}
\]

- Given N independent, identically distributed, completely observed samples:
  \[
\log p(\mathcal{D} \mid \theta) = \left[ \sum_{n=1}^{N} \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_f,n) \right] - NA(\theta)
\]

\[
p(x \mid \theta) = \exp \left\{ \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_f) - A(\theta) \right\}
\]

\[
\psi_f(x_f \mid \theta_f) = \exp\{\theta_f^T \phi_f(x_f)\} \quad A(\theta) = \log Z(\theta)
\]

- Partition function *globally* couples the local factor parameters
Learning for Undirected Models

- Undirected graph encodes dependencies within a single training example:

\[ p(\mathcal{D} \mid \theta) = \prod_{n=1}^{N} \frac{1}{Z(\theta)} \prod_{f \in \mathcal{F}} \psi_f(x_{f,n} \mid \theta_f) \] 
\[ \mathcal{D} = \{x_{\mathcal{V},1}, \ldots, x_{\mathcal{V},N}\} \]

- Given N independent, identically distributed, completely observed samples:

\[ \log p(\mathcal{D} \mid \theta) = \left[ \sum_{n=1}^{N} \sum_{f \in \mathcal{F}} \theta_f^T \phi_f(x_{f,n}) \right] - N A(\theta) \]

- Take gradient with respect to parameters for a single factor:

\[ \nabla_{\theta_f} \log p(\mathcal{D} \mid \theta) = \left[ \sum_{n=1}^{N} \phi_f(x_{f,n}) \right] - N \mathbb{E}_{\theta}[\phi_f(x_f)] \]

- Must be able to compute marginal distributions for factors in current model:
  - Tractable for tree-structured factor graphs via sum-product
  - For general graphs, use the junction tree algorithm to compute