Markov Random Fields II

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Markov random fields

- **Probabilistic models with symmetric dependences.**
  - Typically models spatially varying quantities

\[
P(x) \propto \prod_{c \in \Omega(x)} \phi_c(x_c)
\]

\[\phi_c(x_c)\] - A potential function (defined over factors)

- If \( \phi_c(x_c) \) is strictly positive we can rewrite the definition in terms of a log-linear model:

\[
P(x) = \frac{1}{Z} \exp \left( - \sum_{c \in \Omega(x)} E_c(x_c) \right)
\]

- Energy function

- Gibbs (Boltzman) distribution

\[
Z = \sum_{x \in \Omega} \exp \left( - \sum_{c \in \Omega(x)} E_c(x_c) \right)
\]

- A partition function
Graphical representation of MRFs

An undirected network (also called independence graph)

- \( G = (S, E) \)
  - \( S = 1, 2, \ldots, N \) correspond to random variables
  - \( (i, j) \in E \iff \exists c : \{i, j\} \subset c \)
    or \( x_i \) and \( x_j \) appear within the same factor \( c \)

Example:
- variables \( A, B, \ldots, H \)
- Assume the full joint of MRF

\[
P(A, B, \ldots, H) \sim \phi_1(A, B, C)\phi_2(B, D, E)\phi_3(A, G)\phi_4(C, F)\phi_5(G, H)\phi_6(F, H)
\]

Converting BBNs to MRFs

Moral-graph \( H[G] \): of a Bayesian network over \( X \) is an undirected graph over \( X \) that contains an edge between \( x \) and \( y \) if:
- There exists a directed edge between them in \( G \).
- They are both parents of the same node in \( G \).
Moral Graphs

Why moralization?

\[ P(C, D, G, I, S, L, J, H) = \]

\[ = P(C)P(D \mid C)P(G \mid I, D)P(S \mid I)P(L \mid G)P(J \mid L, S)P(H \mid G, J) \]

\[ = \phi_1(C)\phi_2(D, C)\phi_3(G, I, D)\phi_4(S, I)\phi_5(L, G)\phi_6(J, L, S)\phi_7(H, G, J) \]

Chordal graphs

**Chordal Graph**: an undirected graph \( G \) whose minimum cycle contains 3 vertices.
Chordal Graphs

Properties:
- There exists an elimination ordering that adds no edges.
- The minimal induced treewidth of the graph is equal to the size of the largest clique - 1.

Triangulation

The process of converting a graph $G$ into a *chordal graph* is called Triangulation.

A new graph obtained via triangulation is:
1) Guaranteed to be chordal.
2) Not guaranteed to be (treewidth) optimal.

There exist exact algorithms for finding the *minimal chordal graphs*, and heuristic methods with a guaranteed upper bound.
Chordal Graphs

- Given a minimum triangulation for a graph $G$, we can carry out the variable-elimination algorithm in the minimum possible time.

- **Complexity** of the optimal triangulation:
  - Finding the minimal triangulation is **NP-Hard**.

- **The inference limit:**
  - Inference time is exponential in terms of the largest clique (factor) in $G$.

---

Inference: conclusions

- We cannot escape **exponential costs in the treewidth**.

- But in many graphs the treewidth is much smaller than the total number of variables

- Still a problem: Finding the optimal decomposition is hard
  - But, paying the cost up front may be worth it.
  - Triangulate once, query many times.
  - Real cost savings if not a bounded one.
Clique tree properties

- A clique tree:
  - a tree where nodes correspond to sets of variables
  - used for performing probabilistic inferences

- Sepset \( S_{ij} = C_i \cap C_j \)
  - separation set: Variables \( X \) on one side of sepset are separated from the variables \( Y \) on the other side in the factor graph given variables in \( S \)

- Running intersection property
  - if \( C_i \) and \( C_j \) both contain \( X \), then all cliques on the unique path between them also contain \( X \)

Inference in clique trees

Running intersection:
- E.g. Cliques involving \( S \) form a connected subtree.

Initial potentials \( \pi_0 \):
- Assign factors to cliques and multiply them.
Message Passing VE

- Query for $P(J)$
  - Eliminate $C$:
    $$\tau_1(D) = \sum_C \pi_0^0[C,D]$$

Message sent from $[C,D]$ to $[G,I,D]$  

$$\pi_2[G,I,D] = \tau_1(D) \times \pi_0^0[G,I,D]$$

---

Message Passing VE

- Query for $P(J)$
  - Eliminate $D$:
    $$\tau_2(G,I) = \sum_D \pi_2[G,I,D]$$


$$\pi_3[G,S,I] = \tau_2(G,I) \times \pi_0^0[G,S,I]$$
Message Passing VE

• Query for \( P(J) \)
  – Eliminate I:
    \[ \tau_1(G,S) = \sum_I \pi_3[G,S,I] \]

Message sent from \([G,S,I]\) to \([G,J,S,L]\)

Message received at \([G,J,S,L]\) -- \([G,J,S,L]\) updates:

\[ \pi_4[G,J,S,L] = \pi_3(G,S) \times \pi_4[H,G,J] \]

\([G,J,S,L]\) is not ready!

... And ...

Message Passing VE

• Query for \( P(J) \)
  – Eliminate H:
    \[ \tau_4(G,J) = \sum_I \pi_5[H,G,J] \]

Message sent from \([H,G,J]\) to \([G,J,S,L]\)

\[ \pi_4[G,J,S,L] = \pi_3(G,S) \times \pi_4(G,J) \times \pi_4[H,G,J] \]
Message Passing VE

- Query for $P(J)$
  - Eliminate $K$: $\tau_6(S) = \sum_K \pi_0[S,K]$ 

Message sent from $[S,K]$ to $[G,J,S,L]$ 

All messages received at $[G,J,S,L]$

$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \tau_6(S) \times \pi_0[G,J,S,L]$

And calculate $P(J)$ from it by summing out $G,S,L$

---

Message Passing VE

- $[G,J,S,L]$ clique potential
- … is used to finish the inference
Message passing VE

- Often, **many marginals are desired**
  - Inefficient to re-run each inference from scratch
  - One distinct message per edge & direction
- **Methods**:
  - Compute (unnormalized) marginals for any vertex (clique) of the tree
  - Results in a *calibrated clique tree* \( \sum_{C_i \rightarrow S_y} \pi_i = \sum_{C_j \rightarrow S_y} \pi_j \)
- Recap: three kinds of factor objects
  - Initial potentials, final potentials and messages

---

Two-pass message passing VE

- Chose the root clique, e.g. [S,K]
- Propagate messages to the root

---
Two-pass message passing VE

- Send messages back from the root

![Graphical representation of message passing]

Notation:
- number the cliques and denote the messages
- $\delta_{i \to j}$

Message Passing: BP

- Graphical model of a distribution
  - More edges = larger expressive power
  - Clique tree also a model of distribution
  - Message passing preserves the model but changes the parameterization
- Different but equivalent algorithm
## Factor division

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>0.5</th>
<th>A</th>
<th>B</th>
<th>0.5/0.4=1.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>0.5/0.4=1.25</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.4</td>
<td>1</td>
<td>2</td>
<td>0.4/0.4=1.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.8</td>
<td>2</td>
<td>1</td>
<td>0.8/0.4=2.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.2</td>
<td>2</td>
<td>2</td>
<td>0.2/0.4=2.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.6</td>
<td>3</td>
<td>1</td>
<td>0.6/0.5=1.2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.5</td>
<td>3</td>
<td>2</td>
<td>0.5/0.5=1.0</td>
</tr>
</tbody>
</table>

### Inverse of factor product

- Each node: multiply all the messages and divide by the one that is coming from node we are sending the message to
  - Clearly the same as VE

\[
\delta_{i \rightarrow j} = \frac{\sum_{j \in S_{\delta}} \pi_j}{\sum_{j \in S_{\delta}} \prod_{k \in N(i)} \delta_{k \rightarrow i}} = \frac{\sum_{j \in S_{\delta}} \prod_{k \in N(i)} \delta_{k \rightarrow i}}{\sum_{j \in S_{\delta}} \prod_{k \in N(i)} \delta_{k \rightarrow i}}
\]

- Initialize the messages on the edges to 1
Message Passing: BP

Store the last message on the edge and divide each passing message by the last stored.

\[
\pi_3(C,D) = \pi_3^0(C,D) \frac{\delta_{2\rightarrow3}}{\mu_{2,3}} = \pi_3^0(C,D) \sum_B \pi_2^0(B,C)
\]

\[
\mu_{2,3} = \delta_{2\rightarrow3} = \left( \sum_B \pi_1^0(B,C) \right)
\]

New message

Store the last message on the edge and divide each passing message by the last stored.

\[
\pi_3(C,D) = \pi_3^0(C,D) \sum_B \pi_2^0(B,C) = \pi_3^0(C,D) \mu_{2,3}
\]

\[
\delta_{3\rightarrow2} = \left( \sum_D \pi_3(C,D) \right)
\]

\[
\pi_2(B,C) = \pi_2^0(B,C) \frac{\delta_{3\rightarrow2}}{\mu_{2,3}(C)} = \pi_2^0(B,C) \sum_D \pi_3^0(C,D) \times \mu_{2,3}(C) = \pi_2^0(B,C) \sum_D \pi_3^0(C,D)
\]

\[
\mu_{2,3} = \delta_{3\rightarrow2} = \left( \sum_D \pi_3(C,D) \right) = \sum_D \pi_3^0(C,D) \sum_B \pi_2^0(B,C)
\]

New message
Message Passing: BP

\[
\mu_{2,3} = \sum_D \pi_3^0(C,D) \sum_B \pi_2^0(B,C)
\]

\[
\pi_3(C,D) = \pi_3^0(C,D) \sum_B \pi_2^0(B,C)
\]

\[
\delta_{3 \rightarrow 2} = \left( \sum_D \pi_3(C,D) \right)
\]

\[
\pi_2(B,C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)
\]

The same as before

\[
\pi_2(B,C) = \pi_2(B,C) \times \frac{\delta_{3 \rightarrow 2} \mu_{2,3}(C)}{\pi_3^0(C,D)} = \frac{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)}{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)} = \pi_2(B,C)
\]

Message Propagation: BP

- **Lauritzen-Spiegelhalter algorithm**
- Two kinds of objects: clique and sepset potentials
  - Initial potentials not kept
- Improved “stability” of asynchronous algorithm (repeated messages cancel out)
- **New distribution representation**
  - Clique tree potential
    \[
    \pi_T = \prod_{C_j \in T} \pi_1(C_j) \prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}(S_{ij}) = P_F(X)
    \]
  - Clique tree invariant = \( P_F \)
Loopy belief propagation

- The asynchronous BP algorithm works on clique trees
- What if we run the belief propagation algorithm on a non-tree structure?

- Sometimes converges
- If it converges it leads to an approximate solution
- **Advantage:** tractable for large graphs

---

Loopy belief propagation

- If the BP algorithm converges, it converges to an optimum of the Bethe free energy

See papers: