Markov Random Fields

- Probabilistic models with symmetric dependences.
  - Typically models spatially varying quantities
    \[ P(x) \propto \prod_{c \in CL(x)} \phi_c(x_c) \]
    \( \phi_c(x_c) \) - A potential function (defined over factors)
  - If \( \phi_c(x_c) \) is strictly positive we can rewrite the definition as:
    \[ P(x) = \frac{1}{Z} \exp \left( - \sum_{c \in CL(x)} E_c(x_c) \right) \]  - Energy function
    \( Z = \sum_{x \in \{x\}} \exp \left( - \sum_{c \in CL(x)} E_c(x_c) \right) \) - A partition function
  - Gibbs (Boltzman) distribution
Graphical representation of MRFs

An undirected network (also called independence graph)

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

Example:
- variables A, B, ..H
- Assume the full joint of MRF

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

Markov random fields

- regular lattice (Ising model)
- Arbitrary graph
Markov random fields

• **Pairwise Markov property**
  – Two nodes in the network that are not directly connected can be made independent given all other nodes

• **Local Markov property**
  – A set of nodes (variables) can be made independent from the rest of nodes variables given its immediate neighbors

• **Global Markov property**
  – A vertex set A is independent of the vertex set B (A and B are disjoint) given set C if all chains in between elements in A and B intersect C

Types of Markov random fields

• **MRFs with discrete random variables**
  – Clique potentials can be defined by mapping all clique-variable instances to $\mathbb{R}$
  – Example: Assume two binary variables $A,B$ with values $\{a_1,a_2,a_3\}$ and $\{b_1,b_2\}$ are in the same clique $c$. Then:

\[
\phi_c(A, B) = \begin{bmatrix}
  a_1 & b_1 & 0.5 \\
  a_1 & b_2 & 0.2 \\
  a_2 & b_1 & 0.1 \\
  a_2 & b_2 & 0.3 \\
  a_3 & b_1 & 0.2 \\
  a_3 & b_2 & 0.4
\end{bmatrix}
\]
Types of Markov random fields

- Gaussian Markov Random Field
  \[ x \sim N(\mu, \Sigma) \]
  \[ p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right] \]

- Precision matrix \( \Sigma^{-1} \)
- Variables in \( x \) are connected in the network only if they have a nonzero entry in the precision matrix
  - All zero entries are not directly connected
  - Why?

Tree decomposition of the graph

- A tree decomposition of a graph \( G \):
  - A tree \( T \) with a vertex set associated to every node.
  - For all edges \( \{v, w\} \in G \) there is a set containing both \( v \) and \( w \) in \( T \).
  - For every \( v \in G \) the nodes in \( T \) that contain \( v \) form a connected subtree.
Tree decomposition of the graph

- **Another tree decomposition of a graph** $G$:
  - A tree $T$ with a vertex set associated to every node.
  - For all edges $\{v, w\} \in G$: there is a set containing both $v$ and $w$ in $T$.
  - For every $v \in G$: the nodes in $T$ that contain $v$ form a connected subtree.
Treewidth of the graph

- **Width** of the tree decomposition:
  \[ \max_{i \in I} |X_i| - 1 \]
- **Treewidth** of a graph \( G \):
  \( \text{tw}(G) = \) minimum width over all tree decompositions of \( G \).

Trees

Why do we like trees?
- Inference in trees structures can be done in time linear in the number of nodes
Clique tree

- Clique tree = a tree decomposition of the graph
- Can be constructed:
  - from the induced graph
    Built by running the variable elimination procedure
  - from the chordal graph
    Built by running the triangulation algorithm

- We have precompiled the clique tree.
- So how to take advantage of the clique tree to perform inferences?

VE on the Clique tree

- Variable Elimination on the clique tree
  - works on factors
- Makes factor a data structure
  - Sends and receives messages
- Cluster graph for set of factors, each node $i$ is associated with a subset (cluster) $C_i$.
  - Family-preserving: each factor’s variables are completely embedded in a cluster
Clique tree properties

- **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 \( X \)

Clique trees

- Initial potentials \( \pi^0 \):
  - Assign factors to cliques and multiply them.

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

• Query for $P(J)$
  – Eliminate $C$: 
    \[ \tau_1(D) = \sum_C \pi_1^0[C, D] \]

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

Message received at $[G,I,D]$ -- 

$[G,I,D]$ updates:

\[ \pi_2[G,I,D] = \tau_1(D) \times \pi_1^0[G,I,D] \]

---

Message Passing VE

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


Message received at $[G,S,I]$ -- 

$[G,S,I]$ updates:

\[ \pi_3[G,S,I] = \tau_2(G,I) \times \pi_1^0[G,S,I] \]
Message Passing VE

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

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

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

\[ \pi_4[G,J,S,L] = \tau_3(G,S) \times \pi_4^0[G,J,S,L] \]

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

Message Passing VE

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

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

\[ \pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \pi_4^0[G,J,S,L] \]

And ...
Message Passing VE

• Query for $P(J)$
  – Eliminate $K$: $\tau_5(S) = \sum_K \pi^0[S,K]

\[\pi_3[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \tau_5(S) \times \pi^0_4[G,J,S,L]\]

And calculate $P(J)$ from it by summing out $G,S,L$
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 \( \sum_{C_i \in S_y} \pi_i = \sum_{C_j \in S_y} \pi_j \)

• 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

\[ \delta_{i \rightarrow j} \]

Notation:
number the cliques and denote the messages

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

**Inverse of factor product**

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

Message Passing: BP

- 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

\[
\delta_{i \rightarrow j} = \frac{\sum \pi_j}{\delta_{j \rightarrow i}} = \sum_{c_i \in S_y} \prod_{k \in N(i)} \delta_{k \rightarrow i} = \sum_{c_i \in S_y} \prod_{k \in N(i) \setminus j} \delta_{k \rightarrow i}
\]

- 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.

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

\[ \mu_{2,3} = \delta_{2 \rightarrow 3} = \left( \sum_B \pi_2^0(B, C) \right) \quad \text{New message} \]
Message Passing: BP

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

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

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

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

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

The same as before

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

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

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
    \[ \pi_T = \prod_{i \in T} \pi_i(C_i) \]
    \[ \prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}^{-1}(S_{ij}) = P_F(X) \]
  - 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?

\[ \text{Advantage: tractable for large graphs} \]

- Sometimes converges
- If it converges it leads to an approximate solution

Loopy belief propagation

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

See papers: