Exact Inference 3: Message Passing

Introduction
- We will cover the sum-product message passing algorithm
- Also known as belief propagation

Message Passing
- Message passing is exact when the graph has no (undirected) loops eg.
- If there are loops, you need to use loopy belief propagation (which is approximate)

Intuition (using a chain as an example)
- Each node maintains its current marginal $P(X_i)$ (also called its belief).
- Initially, the marginal doesn’t take the influence of the neighbors into account
- Note that a Node’s belief is affected by its neighbors
- Neighboring nodes send messages to each other
Introduction

Intuition (using a chain as an example)
• Node $X_2$ receives a message $m_{1\rightarrow 2}$ from Node $X_1$
• The message tells Node $X_2$ what state Node $X_1$ thinks Node $X_2$ should be in
• The higher the value of the message, the more likely Node $X_1$ thinks Node $X_2$ should be in that state
• Node $X_2$ updates its belief about $P(X_2)$

What if the graphical model isn’t a chain or a tree?
• Clump nodes into “mega-nodes” (ie. cliques) and treat the cliques like nodes
• This is where clique trees come in
Cluster Graph

In this section we are dealing with a product over factors:

\[ \widetilde{P}_\Phi(X) = \prod_{\phi_i \in \Phi} \phi_i(X_i) \]

- Normalized distribution for Bayesian networks since factors are CPDs
- Unnormalized distribution for Gibbs distributions

• Each factor \( \phi \in \Phi \) must be associated with a cluster \( C_i \), denoted \( \alpha(\phi) \), such that \( \text{Scope}[\phi] \subseteq C_i \).
• Each edge between a pair of clusters \( C_i \) and \( C_j \) is associated with a sepset \( S_{ij} \subseteq C_i \cap C_j \).
• A cluster graph is a generalization of a clique tree

Example of a cluster graph:

1: C, D
2: G, I, D
3: G, S, I
4: H, G, J
5: G, J, S, L
6: J, S, L
7: J, L

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

A new way to interpret variable elimination:
• (Recall: variable elimination defines a cluster graph)
• Factors \( \psi_i \) accept messages \( \tau_i \) from another factor \( \psi_j \)
• Factors \( \psi_i \) also send their own messages \( \tau_i \) to another factor
### Cluster Graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Variable Eliminated</th>
<th>Factors Used</th>
<th>New Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>$\phi(C), \phi(D,C)$</td>
<td>$\tau_1(D)$</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>$\phi(D), \phi(D,C), \phi(G,I,D), \phi(G,I)$</td>
<td>$\tau_2(G,J)$</td>
</tr>
<tr>
<td>3</td>
<td>I</td>
<td>$\phi(I), \phi(I,J), \phi(I,J,L), \phi(G,I)$</td>
<td>$\tau_3(G,J)$</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
<td>$\phi(H,G,J), \phi(H,G,J), \phi(H,G,J,L)$</td>
<td>$\tau_4(G,J)$</td>
</tr>
<tr>
<td>5</td>
<td>G</td>
<td>$\phi(G,J), \phi(G,J,L), \phi(G,J,L,S)$</td>
<td>$\tau_5(G,J)$</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>$\phi(J,L), \phi(J,L,S), \phi(J,L,S,J)$</td>
<td>$\tau_6(J,L)$</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>$\phi(J,L)$</td>
<td>$\tau_7(J,L)$</td>
</tr>
</tbody>
</table>

Note:
- Cluster graph produced by variable elimination is a tree.
- Each original factor $\phi$ is used only once to create cluster $\psi$.
- Execution of variable elimination causes messages to flow "up" to a "root" node.

---

### Cluster Graph

- T has the **running intersection property** if, whenever there is a variable X such that $X \in C_i$ and $X \in C_j$, then X is also in every cluster in the (unique) path in T between $C_i$ and $C_j$.
- Example: cluster tree below obeys the running intersection property (see G in $C_2$ and $C_4$).
- Running intersection property implies sepset $S_{ij} = C_i \cap C_j$.

- Theorem 10.1: Let T be a cluster tree induced by a variable elimination algorithm over some set of factors $\Phi$. Then T satisfies the running intersection property.
**Clique Tree**

- Let $\Phi$ be a set of factors over $X$. A cluster tree over $\Phi$ that satisfies the running intersection property is called a **clique tree** (aka junction tree or join tree).
- In the case of a clique tree, the clusters are also called cliques.

**Message Passing: Sum Product**

- Assume we are given a clique tree
- Note: can use the same clique tree to cache computations for multiple executions of variable elimination
- Cheaper than performing each variable elimination separately
Message Passing: Sum Product

1: (C,D)  
2: (G, I, D)  
3: (G, S, I)  
5: (G, J, S, L)  
4: (H, G, J)

Execute the following:
- In C₁: Eliminate C by $\sum_C \psi_1(C,D)$. Resulting factor has scope D. Send message $\delta_1 \rightarrow 2(D)$ to C₂.
- In C₂: Define $\beta_2(G,I,D) = \delta_1 \rightarrow 2(D) \cdot \psi_2(G,I,D)$. Eliminate D to get a factor $\delta_2 \rightarrow 3(G,I)$ which is sent to C₃.
- In C₃: Define $\beta_3(G,S,I) = \delta_2 \rightarrow 3(G,I) \cdot \psi_3(G,S,I)$. Eliminate I to get a factor $\delta_3 \rightarrow 5(G,S)$ which is sent to C₅.
- In C₄: Eliminate H by $\sum_H \psi_4(H,G,J)$. Send factor $\delta_4 \rightarrow 5(G,J)$ to C₅.

Clique is ready when it has received all of its incoming messages eg.
- C₄ ready at the start
- C₂ ready only after getting message from C₁

C₁, C₄, C₂, C₃, C₅ is a legal execution ordering for the tree rooted at C₅

C₂, C₁, C₅, C₃, C₅ is not a legal execution ordering
Message Passing: Sum Product

1. Initial potentials
   - Each factor \( \phi \in \Phi \) is assigned to some clique \( \alpha(\phi) \)
   - The initial potential of \( C_j \) is:
     \[
     \Psi_j(C_j) = \prod_{\phi : \alpha(\phi) = j} \phi
     \]
   - Since each factor is assigned to exactly one clique, we have:
     \[
     \prod_j \phi = \prod_j \Psi_j
     \]

Clique-Tree Message Passing

1. Set initial potentials
2. Pass messages to neighboring cliques, sending to root clique

Message Passing: Sum Product

1. Initial potentials
   - Each factor \( \phi \in \Phi \) is assigned to some clique \( \alpha(\phi) \)
   - The initial potential of \( C_j \) is:
     \[
     \Psi_j(C_j) = \prod_{\phi : \alpha(\phi) = j} \phi
     \]
   - Since each factor is assigned to exactly one clique, we have:
     \[
     \prod_j \phi = \prod_j \Psi_j
     \]

Message Passing: Sum Product

2. Message passing
   - Definitions:
     - \( C_r \) = root clique
     - \( Nb_c \) = indices of cliques that are neighbors of \( C_i \)
     - \( p(i) \) = upstream neighbor of \( i \) (the one on the path to the root clique \( r \))
   - Start with the leaves of the clique tree and move inward
   - Each clique \( C_j \) (except for the root) performs a message passing computation and sends message to upstream neighbor \( C_{p(i)} \)
Message Passing: Sum Product

Message from $C_i$ to $C_j$:

$$\delta_{i \rightarrow j} = \sum_{C_i \rightarrow S_{i,j}} \psi_{i,j} \prod_{k \in \{N_{ij} \setminus \{j\}\}} \delta_{k \rightarrow i}$$

- Clique $C_i$ multiplies incoming messages from its neighbors (except $j$) with its initial clique potential.
- Sums out all variables except those in the sepset between $C_i$ and $C_j$.
- Sends resulting factor to $C_j$.

Message Passing: Sum Product

• At the root, once all messages are received, it multiplies them with its own initial potential.
• Result is a factor called the beliefs $\beta_r(C_r)$, which represents

$$\tilde{P}_\Phi(C_r) = \sum_{X - C_r} \prod \phi$$

Message Passing: Sum Product

**Procedure** CTree-SP-Upward (  
\(\Phi\), // Set of factors \(T\), // Clique tree over \(\Phi\) \(\alpha\), // Initial assignment of factors to cliques \(C_r\) // Some selected root clique )

1. Initialize-Cliques()  
2. while \(C_r\) is not ready  
3. Let $C_i$ be a ready clique  
4. $\delta_{i \rightarrow (i)} (S_{i,a(i)}) \leftarrow SP-Message(i,p_{i}(i))$  
5. $\beta_r \leftarrow \psi_{r} \cdot \prod_{k \in \text{sepset}} \delta_{k \rightarrow r}$  
6. return $\beta_r$

Message Passing: Sum Product

**Procedure** Initialize-Cliques ()

1. for each clique $C_i$  
2. $\psi_{i}(C_i) \leftarrow \prod_{\phi \in \alpha(C_i) \setminus \{i\}} \phi$

**Procedure** SP-Message (  
i, // sending clique \(j\) // receiving clique )

1. $\psi(C_i) \leftarrow \psi_{i} \cdot \prod_{k \in \{N_{ij} \setminus \{j\}\}} \delta_{k \rightarrow i}$  
2. $\tau(S_{i,j}) \leftarrow \sum_{C_i \in S_{i,j}} \psi(C_i)$  
3. return $\tau(S_{i,j})$