Exact Inference 4: Message Passing

Introduction

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

Introduction

• Message passing is exact when the graph has no (undirected) loops eg.

A chain

A tree

• If there are loops, you need to use loopy belief propagation (which is approximate)

Message Passing

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

Clique Trees
Cluster Graph

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

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

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

Example of a cluster graph

A cluster graph \( \mathcal{U} \) for a set of factors \( \Phi \) over \( X \) is an undirected graph, each of whose nodes \( i \) is associated with a subset \( \mathcal{C}_i \subseteq X \).

Example of a cluster graph

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

Example of a cluster graph

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

- T has the running intersection property if, whenever there is a variable X such that X ∈ Ci and X ∈ Cj, then X is also in every cluster in the (unique) path in T between Ci and Cj.
- Example: cluster tree below obeys the running intersection property (see G in C2 and C4)

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

Running intersection property implies sepset S_{i,j} = Ci ∩ Cj.

Theorem 10.1: Let T be a cluster tree induced by a variable elimination algorithm over some set of factors Φ. Then T satisfies the running intersection property.

Note:
- Cluster graph produced by variable elimination is a tree
- Each original factor ϕ is used only once to create cluster ψ
- Execution of variable elimination causes messages to flow “up” to a “root” node
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

Example: Simplified Extended Student Clique tree

$P(D|C)$
$P(C)$
$P(G|I,D)$
$P(I)$
$P(S|I)$
$P(L|G)$
$P(J|L,S)$
$P(H|G,J)$

- First step: generate a set of initial potentials $\psi(C_i)$ with each clique eg. by multiplying the initial factors
  - For instance, $\psi_5(J,L,G,S) = \phi_l(L,G) \cdot \phi_j(J,L,S)$
- Suppose we have to compute $P(J)$:
  - Select a root clique that does contain $J$ eg. $C_5$. 
Message Passing: Sum Product

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

δ_{1→2}(D): \sum_C \psi_1(C) \delta_1 \delta_{1→2}
δ_{2→3}(G,I): \sum_C \psi_2(C) \times \delta_{1→2}
δ_{1→2}(G,S): \sum_C \psi_3(C_3) \times \delta_{1→3}
δ_{2→3}(G,J): \sum_C \psi_4(C_4)

Execute the following:

- In C₁: Eliminate C by \(\sum_C \psi_1(C,D)\). Resulting factor has scope D. Send message \(\delta_{1→2}(D)\) to C₂.
- In C₂: Define \(\beta_2(G,I,D) = \delta_{1→2}(D) \psi_2(G,I,D)\). Eliminate D to get a factor \(\delta_{2→3}(G,I)\) which is sent to C₃.
- In C₃: Define \(\beta_3(G,S,I) = \delta_{2→3}(G,I) \psi_3(G,S,I)\). Eliminate I to get a factor \(\delta_{3→5}(G,S)\) which is sent to C₅.

Message Passing: Sum Product

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

δ_{1→2}(D): \sum_C \psi_1(C_1) \delta_{1→2}
δ_{2→3}(G,I): \sum_C \psi_2(C_2) \times \delta_{1→2}
δ_{2→3}(G,S): \sum_C \psi_3(C_3) \times \delta_{1→3}
δ_{2→3}(G,J): \sum_C \psi_4(C_4)

Execute the following:

- In C₂: Eliminate H by \(\sum_C \psi_4(H,G,J)\). Send factor \(\delta_{4→5}(G,J)\) to C₅.
- In C₃: Define \(\beta_3(G,J,S,L) = \delta_{3→5}(G,S) \cdot \delta_{4→5}(G,J) \cdot \psi_5(G,J,S,L)\)
- Sum out G, L, and S from \(\beta_5\) to get P(J).

Message Passing: Sum Product

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

δ_{1→2}(D): \sum_C \psi_1(C_1) \delta_{1→2}
δ_{2→3}(G,I): \sum_C \psi_2(C_2) \times \delta_{1→2}
δ_{2→3}(G,S): \sum_C \psi_3(C_3) \times \delta_{1→3}

Could also define C₄ as the root

- In C₄: computation and message unchanged
- In C₂: computation and message unchanged
- In C₅: computation and message unchanged

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 $C_i$.
   - The initial potential of $C_i$ is:
     $$\psi_j(C_j) = \prod_{\phi : \alpha(\phi) = j} \phi$$
   - Since each factor is assigned to exactly one clique, we have:
     $$\prod_{\phi} \phi = \prod_{j} \psi_j$$

2. Message passing
   - Definitions:
     - $C_r$ = root clique
     - $N_{b_i}$ = indices of cliques that are neighbors of $C_i$
     - $p_i(i)$ = 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 $C_i$ (except for the root) performs a message passing computation and sends message to upstream neighbor $C_{p_i(i)}$

Clique-Tree Message Passing
1. Set initial potentials
2. Pass messages to neighboring cliques, sending to root clique
Message Passing: Sum Product

Message from $C_i$ to $C_j$:

$$\delta_{i \rightarrow j} = \sum_{C_i - S_{i,j}} \psi_i \cdot \prod_{k \in (N_{j} \setminus \{j\})} \delta_{k \rightarrow i}$$

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:

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

---

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. $\delta_{i \rightarrow r}(S_{i,r}(\alpha)) \leftarrow$ SP-Message($i$, $r$)
5. $\beta_r \leftarrow \psi_r \cdot \prod_{k \in N_{r}} \delta_{k \rightarrow r}$
6. return $\beta_r$

---

Message Passing: Sum Product

**Procedure** Initialize-Cliques ()

1. for each clique $C_i$
2. $\psi_i(C_i) \leftarrow \prod_{\phi_i(\Phi) \neq i} \phi$

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

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