Exact Inference 5: Clique Trees

Clique Tree Calibration

• In the previous lecture, we used the clique tree to compute the probability of a single variable eg. $P(J)$
• Root clique must contain $J$
• Messages passed upstream (toward root)
Clique Tree Calibration

• But we often want to compute the probability of a large number of variables eg. P(J), P(C), P(H)
• What if we wanted to compute the probability of every random variable in the network?

Clique Tree Calibration

• The expensive way:
  – Run clique tree inference for each node
  – Cost is $O(c \times \text{number of nodes})$
• A little less expensive:
  – Make each clique the root and run inference
  – Cost is $O(c \times \text{number of cliques})$

Where $c = \text{cost of running clique tree inference}$
Clique Tree Calibration

• The smart way:
  – Notice that you end up calculating the same messages over and over again
  – Cache these result and reuse them in a clever way! \(=>\) dynamic programming
  – Results in a cost of \(2c\)

Clique Tree Calibration

\[
\begin{array}{ccc}
\mathbf{C_i} & \mathbf{C_j} & \cdots & \text{Root} \\
\end{array}
\]

• As long as the root clique is on the \(\mathbf{C_j}\) side, exactly the same message is sent from \(\mathbf{C_i}\) to \(\mathbf{C_j}\) (regardless of which clique is the root)
• Same thing applies if the root is on the \(\mathbf{C_i}\) side
• For any given clique tree, each edge has two messages associated with it – one for each direction
• If there are \(c\) cliques, there are \((c-1)\) edges and \(2(c-1)\) messages to compute
Clique Tree Calibration

• Let $\mathcal{T}$ be a clique tree. We say that $C_i$ is ready to transmit to a neighbor $C_j$ when $C_i$ has messages from all of its neighbors except from $C_j$.
• When $C_i$ is ready to transmit to $C_j$, it computes $\delta_{i\rightarrow j}(S_{i,j})$ from all incoming messages (except from $C_j$).
• Then eliminating the variables in $C_i - S_{i,j}$
• Use dynamic programming to avoid recomputing the same message multiple times

Clique Tree Calibration

**Sum-Product Belief Propagation**

**Procedure** CTREE-SP-CALIBRATE ($\Phi$, $\mathcal{T}$)

1. Initialize-Cliques
2. while exist $i, j$ such that $i$ is ready to transmit to $j$
3. $\delta_{i\rightarrow j}(S_{i,j}) \leftarrow$ SP-Message($i,j$)
4. for each clique $i$
5. $\beta_i \leftarrow \psi_i \cdot \prod_{k \in Nb_i} \delta_{k\rightarrow i}$
6. return { $\beta_i$ }

Sum-Product Belief Propagation
Clique Tree Calibration

- **Upward pass**: pick a root, send messages to root
- **Downward pass**: then send messages to the leaves
- In asynchronous version, each clique sends message as soon as it is ready

Message Passing: Sum Product

Example of a downward pass in the Student network:

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

\[
\delta_{5 \rightarrow 3}(G, S) = \sum_{L} \psi_{5}(C_{5}) \times \delta_{4 \rightarrow 5}
\]

\[
\delta_{1 \rightarrow 3}(D) = \sum_{C} \psi_{1}(C_{1})
\]

\[
\delta_{2 \rightarrow 3}(G, I) = \sum_{D} \psi_{2}(C_{2}) \times \delta_{1 \rightarrow 2}
\]

\[
\delta_{3 \rightarrow 5}(G, S) = \sum_{I} \psi_{3}(C_{3}) \times \delta_{2 \rightarrow 3}
\]

\[
\delta_{4 \rightarrow 5}(G, J) = \sum_{L} \psi_{4}(C_{4})
\]
Message Passing: Sum Product

Example of a downward pass in the Student network:

Clique Tree Calibration

- At the end, compute beliefs for all cliques in the tree by multiplying initial potential with each of the incoming messages
- Corollary 10.2: Assume that, for each clique $i$, $\beta_i$ is computed as in the Sum-Product Belief Propagation algorithm. Then

$$\beta_i(C_i) = \sum_{X \cap C_i} \tilde{P}_\phi(X)$$
Clique Tree Calibration

• \( C_i \) computes the message to a neighboring clique \( C_j \) based on its initial potential \( \psi_i \) (not its modified potential \( \beta_i \))

• Modified potential already integrates information from \( C_j \) (would be double-counting factors in \( C_j \))

Clique Tree Calibration

• At the end, each clique contains the marginal (unnormalized) probability over the variables in its scope

• Can compute marginal probability of \( X \) by selecting the clique whose scope contains \( X \) and eliminating the redundant variables in the clique
  – If \( X \) appears in two cliques, we can pick either one
  – Both must agree on the marginal
Clique Tree Calibration

Two adjacent cliques $C_i$ and $C_j$ are said to be calibrated if
\[
\sum_{c_{i-S_{i,j}}} \beta_i(C_i) = \sum_{c_{j-S_{i,j}}} \beta_j(C_j)
\]

Clique Tree Calibration

A clique $T$ is calibrated if all pairs of adjacent cliques are calibrated. For a calibrated clique tree, we use the term clique beliefs for $\beta_i(C_i)$ and sepset beliefs for
\[
\mu_{i,j}(S_{i,j}) = \sum_{c_{i-S_{i,j}}} \beta_i(C_i) = \sum_{c_{j-S_{i,j}}} \beta_j(C_j)
\]
Calibrated Clique Trees as a Distribution

- Recall that the unnormalized measure:
  \[ \tilde{P}_\Phi(X) = \prod_{\phi \in \Phi} \phi(X_i) \]

- We will reparameterize the above as:
  \[
  \tilde{P}_\Phi(X) = \frac{\prod_{i \in V} \beta_i(C_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(S_{i,j})}
  \]

- Why? Useful for an alternate version of message passing

This is called the clique tree invariant
Calibrated Clique Trees as a Distribution

To see this, note that at calibration we have:

- **Clique beliefs:**
  \[
  \beta_i = \psi_i \cdot \prod_{k \in N_{b_k}} \delta_{k \rightarrow i}
  \]

- **Sepset beliefs:**
  \[
  \mu_{i,j}(S_{i,j}) = \sum_{C_i \in S_{i,j}} \beta_i(C_i) = \sum_{C_i \in S_{i,j}} \psi_i \cdot \prod_{k \in N_{b_k}} \delta_{k \rightarrow i}
  \]
  \[
  = \sum_{C_i \in S_{i,j}} \psi_i \cdot \delta_{j \rightarrow i} \prod_{k \in (N_{b_i} \setminus \{j\})} \delta_{k \rightarrow i} = \delta_{j \rightarrow i} \sum_{C_i \in S_{i,j}} \psi_i \cdot \prod_{k \in (N_{b_i} \setminus \{j\})} \delta_{k \rightarrow i}
  \]
  \[
  = \delta_{j \rightarrow i} \delta_{i \rightarrow j}
  \]

Using the clique beliefs and sepset beliefs,

\[
\tilde{P}_\theta(X) = \prod_{i \in V_T} \beta_i(C_i) \prod_{i \in V_T} \psi_i(C_i) \prod_{k \in N_{b_k}} \delta_{k \rightarrow i}
\]

\[
= \prod_{(i \leftarrow j) \in E_T} \mu_{i,j}(S_{i,j}) \prod_{(i \leftarrow j) \in E_T} \delta_{i \rightarrow j} \delta_{j \rightarrow i}
\]

Each message \(\delta_{i \rightarrow j}\) appears once in the numerator and once in the denominator:

\[
\tilde{P}_\theta(X) = \prod_{i \in V_T} \psi_i(C_i)
\]
Calibrated Clique Trees as a Distribution

The measure induced by a calibrated tree $\mathcal{T}$ is defined as:

$$Q_T = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(S_{i,j})}$$

where

$$\mu_{i,j} = \sum_{C_i \neq S_{i,j}} \beta_i(C_i) = \sum_{C_j \neq S_{i,j}} \beta_i(C_i)$$

Calibrated Clique Trees as a Distribution

**Theorem 10.4:** Let $\mathcal{T}$ be a clique tree over $\Phi$, and let $\beta_j(C_j)$ be a set of calibrated potentials for $\mathcal{T}$. Then, $\tilde{P}_\phi(X) \propto Q_T$ if and only if, for each $i \in V_T$, we have that

$$\beta_i(C_i) \propto \tilde{P}_\phi(C_i)$$

(Proof Omitted)

This alternate representation of the joint measure directly reveals the clique marginals $\beta_i(C_i)$
Message Passing: Belief Update

First message from $C_j$ to $C_i$

Second message from $C_i$ to $C_j$ (once it receives messages from all neighbors except $j$)

- Previously: final potential ($\beta_i$) not used in message to $C_j$ (would double count information from $C_j$)
- Different approach: multiply all messages together and divide resulting factor by $\delta_{j-i}$ (removes $C_j$’s contribution)
Message Passing: Belief Update

- Let $X$ and $Y$ be disjoint sets of variables, and let $\phi_1(X,Y)$ and $\phi_2(Y)$ be two factors.
- We define the factor division $\phi_1/\phi_2$ to be a factor $\psi$ of scope $X, Y$ defined as follows:

$$\psi(X,Y) = \frac{\phi_1(X,Y)}{\phi_2(Y)}$$

Where we define $0/0 = 0$. The operation not well defined if denominator is 0 and numerator isn’t.
Message Passing: Belief Update

New version of message passing:

\[ \beta_i = \psi_i \cdot \prod_{k \in Nh_i} \delta_{k \rightarrow i} \]  
(As before)

\[ \delta_{i \rightarrow j} = \frac{\sum \beta_i}{\delta_{j \rightarrow i}} \]  
Note the division

Example: Using CTree-SP-Calibrate as the Message Passing algorithm:

Notice that:

\[ \sum_{G,I} \beta_2(G,I,D) \cdot \delta_{1 \rightarrow 2}(D) \cdot \delta_{3 \rightarrow 2}(G,I) = \sum_{G,I} \psi_2(G,I,D) \cdot \delta_{1 \rightarrow 2}(D) \cdot \delta_{3 \rightarrow 2}(G,I) \]  
(Approaches are equivalent)
**Message Passing: Belief Update**

Belief-update Message Passing Algorithm

**Procedure** CTree-BU-Calibrate (  
\( \Phi \), // Set of factors  
\( \mathcal{T} \), // Clique tree over \( \Phi \)  
)

1. Initialize-CTree
2. **while** exists an uninformed clique in \( \mathcal{T} \)
3. Select \((i\rightarrow j) \in E_{\mathcal{T}}\)
4. BU-Message(i,j)
5. **return** \{\(\beta_i\)\}

Note: any arbitrary pair can be chosen without violating the correctness of the algorithm

---

**Message Passing: Belief Update**

**Procedure** Initialize-CTree ()

1. **for** each clique \( C_i \)
2. \( \beta_i \leftarrow \prod_{\phi: \phi(i)=i} \phi \)
3. **for** each edge \((i\rightarrow j) \in E_{\mathcal{T}}\)
4. \( \mu_{ij} \leftarrow 1 \)

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

1. \( \sigma_{i\rightarrow j} \leftarrow \sum_{C \subseteq S_i} \beta_i \) // marginalize clique over the sepset
2. \( \beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i\rightarrow j}}{\mu_{ij}} \) // Divides out the previous message (prevents double counting)
3. \( \mu_{ij} \leftarrow \sigma_{i\rightarrow j} \) // Remembers the current message as the new previous message
Message Passing: Belief Update

The following are the implications (stated without proof here):

- Sum-Product and Belief-Update message passing are equivalent
- Belief-update message passing guaranteed to converge to the correct marginals
- Message schedule that guarantees convergence to the correct clique marginals in two passes:
  - Follow upward-downward pass schedule using any arbitrarily chosen root clique $C_r$.

Constructing a Clique Tree
Constructing a Clique Tree

How do we construct a clique tree?
1. Through executing Variable Elimination
   • A clique $C_i$ corresponds to a factor $\psi_i$
   • Undirected edge connects $C_i$ and $C_j$ when $\tau_i$ is used directly in the computation of $\psi_j$ (or vice versa)
   • Cliques in clique tree are maximal cliques in the induced graph

2. Manipulating the graph directly
   1. Given a set of factors, construct the undirected graph $\mathcal{H}_\Phi$
   2. Triangulate $\mathcal{H}_\Phi$ to construct a chordal graph $\mathcal{H}^*$
   3. Find cliques in $\mathcal{H}^*$, and make each one a node in a cluster graph
   4. Run the maximum spanning tree algorithm on the cluster graph to construct a tree
Constructing a Clique Tree

• Triangulation: constructing a chordal graph that subsumes an existing graph $\mathcal{H}$

• Minimum triangulation: largest clique in the resulting chordal graph has minimum size

• Finding the minimum triangulation is NP-hard – need to resort to heuristics

Constructing a Clique Tree

• Finding the maximal clique in a general graph is NP-hard
  – But for chordal graphs, this is easy (number of possible approaches)

• Finding edges in clique tree
  – Use maximum spanning tree algorithm
  – Nodes are the maximal cliques, edges have weight equal to $|C_i \cap C_j|$