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)

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

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

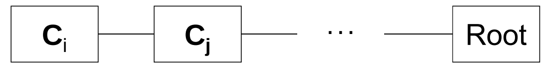
- The expensive way:
  - Run clique tree inference for each node
  - Cost is *O(c * number of nodes)*
- A little less expensive:
  - Make each clique the root and run inference
  - Cost is *O(c * number of cliques)*

Where *c* = 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



- 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 $\mathbf{C}_i$ is ready to transmit to a neighbor $\mathbf{C}_j$ when $\mathbf{C}_i$ has messages from all of its neighbors except from $\mathbf{C}_j$
- When $\mathbf{C}_i$ is ready to transmit to $\mathbf{C}_j$, is computes $\delta_{i\rightarrow j}(\mathbf{S}_{i,j})$ from all incoming messages (except from $\mathbf{C}_j$).
- Then eliminating the variables in $\mathbf{C}_i - \mathbf{S}_{i.j}$
- Use dynamic programming to avoid recomputing the same message multiple times

Clique Tree Calibration

<u>Sum-Product Belief Propagation</u>

Procedure CTree-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)

1. Initialize-Cliques
2. **while** exist *i*, *j* such that *i* is ready to transmit to *j*
3. $\delta_{i,j}(\mathbf{S}_{i,j}) \leftarrow$ 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** { β_i }

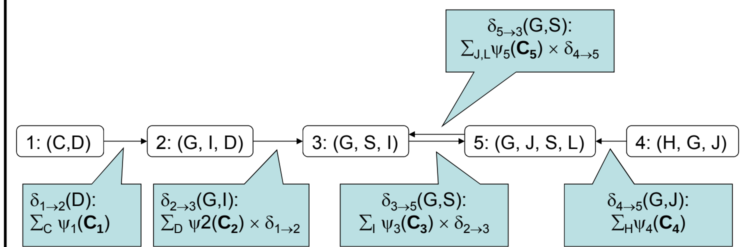
Clique Tree Calibration

- 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

9

Message Passing: Sum Product

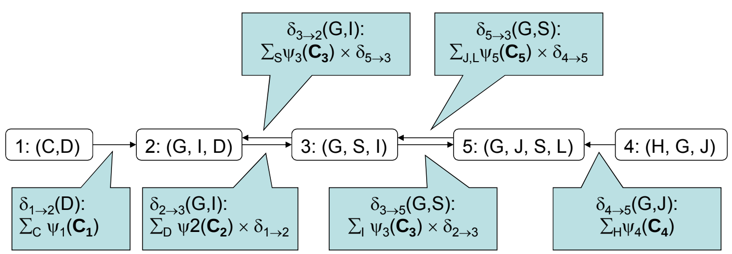
Example of a downward pass in the Student network:



10

Message Passing: Sum Product

Example of a downward pass in the Student network:



11

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*, β_i is computed as in the Sum-Product Belief Propagation algorithm. Then

$$\beta_i(\mathbf{C}_i) = \sum_{X-\mathbf{C}_i} \tilde{P}_\Phi(X)$$

12

Clique Tree Calibration

- $\mathbf{C}_i$ computes the message to a neighboring clique $\mathbf{C}_j$ based on its initial potential ψ_i (not its modified potential β_i)
- Modified potential already integrates information from $\mathbf{C}_j$ (would be double-counting factors in $\mathbf{C}_j$)

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 $\mathbf{C}_i$ and $\mathbf{C}_j$ are said to be calibrated if

$$\sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \sum_{\mathbf{C}_j - \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j)$$

Clique Tree Calibration

A clique $\mathcal{T}$ is calibrated if all pairs of adjacent cliques are calibrated. For a calibrated clique tree, we use the term clique beliefs for $\beta_i(\mathbf{C}_i)$ and sepset beliefs for

$$\mu_{i,j}(\mathbf{S}_{i,j}) = \sum_{\mathbf{C}_i - \mathbf{S}_{i,j}} \beta_i(\mathbf{C}_i) = \sum_{\mathbf{C}_j - \mathbf{S}_{i,j}} \beta_j(\mathbf{C}_j)$$

Calibrated Clique Trees as a Distribution

17

Calibrated Clique Trees as a Distribution

- Recall that the unnormalized measure:

$$\tilde{P}_\Phi(\mathcal{X}) = \prod_{\phi_i \in \Phi} \phi_i(\boldsymbol{X}_i)$$

- We will reparameterize the above as:

$$\tilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i \in V_T} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(\boldsymbol{S}_{i,j})}$$

This is called the clique tree invariant

- Why? Useful for an alternate version of message passing

18

Calibrated Clique Trees as a Distribution

To see this, note that at calibration we have:

- Clique beliefs:

$$\beta_i = \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i}$$

- Sepset beliefs:

$$\mu_{i,j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_i - S_{i,j}} \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i}$$

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

$$= \delta_{j \to i} \delta_{i \to j}$$

19

Calibrated Clique Trees as a Distribution

Using the clique beliefs and sepset beliefs,

$$\tilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i \in V_T} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(\boldsymbol{S}_{i,j})} = \frac{\prod_{i \in V_T} \psi_i(C_i) \prod_{k \in Nb_i} \delta_{k \to i}}{\prod_{(i-j) \in E_T} \delta_{i \to j} \delta_{j \to i}}$$

Each message $\delta_{i \to j}$ appears once in the numerator and once in the denominator:

$$\tilde{P}_\Phi(\mathcal{X}) = \prod_{i \in V_T} \psi_i(\boldsymbol{C}_i)$$

20

Calibrated Clique Trees as a Distribution

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

$$Q_T = \frac{\prod_{i \in V_T} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in E_T} \mu_{i,j}(\boldsymbol{S}_{i,j})}$$

where

$$\mu_{i,j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_i(\boldsymbol{C}_i)$$

21

Calibrated Clique Trees as a Distribution

Theorem 10.4: Let $\mathcal{T}$ be a clique tree over Φ, and let $\beta_i(\boldsymbol{C}_i)$ be a set of calibrated potentials for $\mathcal{T}$. Then, $\tilde{P}_\Phi(\mathcal{X}) \propto Q_\mathcal{T}$ if and only if, for each $i \in V_\mathcal{T}$, we have that

$$\beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)$$

(Proof Omitted)

This alternate representation of the joint measure directly reveals the clique marginals $\beta_i(\mathbf{C}_i)$

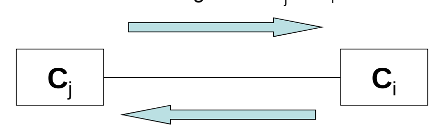
22

Message Passing: Belief Update

23

Message Passing: Belief Update

First message from $\mathbf{C}_j$ to $\mathbf{C}_i$



Second message from $\mathbf{C}_i$ to $\mathbf{C}_j$ (once it receives messages from all neighbors except j)

- Previously: final potential (β_i) not used in message to $\mathbf{C}_j$ (would double count information from $\mathbf{C}_j$)
- Different approach: multiply all messages together and divide resulting factor by $\delta_{j \to i}$ (removes $\mathbf{C}_j$'s contribution)

24

Message Passing: Belief Update

- Let **X** and **Y** be disjoint sets of variables, and let ϕ_1(**X**,**Y**) and ϕ_2(**Y**) be two factors.
- We define the factor division ϕ_1/ϕ_2 to be a factor ψ of scope X, Y defined as follows:

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

Where we define 0/0 = 0. The operation not well defined if denominator is 0 and numerator isn't

Message Passing: Belief Update

A	B	ϕ_1(A,B)
0	0	0.5
0	1	0.2
1	0	0
1	1	0
2	0	0.3
2	1	0.45

A	ϕ_2(A)
0	0.8
1	0
2	0.6

A	B	ψ(A,B)
0	0	0.5/0.8=0.625
0	1	0.2/0.8=0.25
1	0	0
1	1	0
2	0	0.3/0.6=0.5
2	1	0.45/0.6=0.75

Message Passing: Belief Update

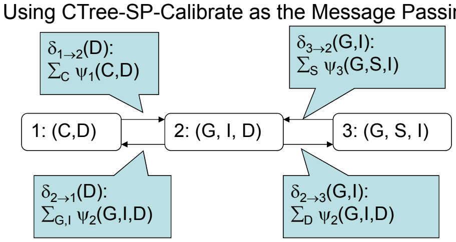
New version of message passing:

$$\beta_i = \psi_i \cdot \prod_{k \in Nb_i} \delta_{k \to i} \quad \text{(As before)}$$

$$\delta_{i \to j} = \frac{\sum_{C_i - S_{i,j}} \beta_i}{\delta_{j \to i}} \quad \text{Note the division}$$

Message Passing: Belief Update

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



Notice that:
$$\frac{\sum_{G,I} \beta_2(G,I,D)}{\delta_{1\to2}(D)} = \sum_{G,I} \frac{\psi_2(G,I,D)\cdot\delta_{1\to2}(D)\cdot\delta_{3\to2}(G,I)}{\delta_{1\to2}(D)}$$

$$= \sum_{G,I} \psi_2(G,I,D)\cdot\delta_{3\to2}(G,I)$$

(Approaches are equivalent)

Message Passing: Belief Update

Belief-update Message Passing Algorithm

Procedure CTree-BU-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)
1. Initialize-CTree
2. **while** exists an uninformed clique in $\mathcal{T}$
3. Select (i—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

29

Message Passing: Belief Update

Procedure Initialize-CTree ()
1. **for** each clique $\mathbf{C}_i$
2. $\beta_i \leftarrow \prod_{\phi:\alpha(\phi)=i} \phi$
3. **for** each edge(i—j) $\in E_{\mathcal{T}}$
4. $\mu_{i,j} \leftarrow 1$

Procedure BU-Message (
  i, // sending clique
  j // receiving clique
)
1. $\sigma_{i \to j} \leftarrow \sum_{C_i - S_{i,j}} \beta_i$ // marginalize clique over the sepset
2. $\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}}$ Divides out the previous message (prevents double counting)
3. $\mu_{i,j} \leftarrow \sigma_{i \to j}$ Remembers the current message as the new previous message

30

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 $\mathbf{C}_r$.

31

Constructing a Clique Tree

32

Constructing a Clique Tree

How do we construct a clique tree?

1. Through executing Variable Elimination
   - A clique $\mathbf{C}_i$ corresponds to a factor ψ_i
   - Undirected edge connects $\mathbf{C}_i$ and $\mathbf{C}_j$ when τ_i is used directly in the computation of ψ_j (or vice versa)
   - Cliques in clique tree are maximal cliques in the induced graph

33

Constructing a Clique Tree

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

34

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

35

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 $|\mathbf{C}_i \cap \mathbf{C}_j|$

36