Exact Inference 4: Message Passing

1

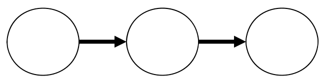
Introduction

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

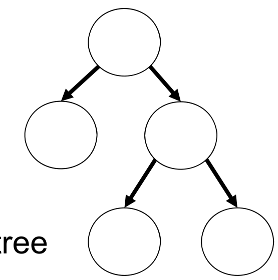
2

Introduction

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



A chain



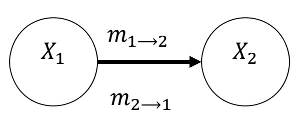
A tree

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

Message Passing

Intuition (using a chain as an example)

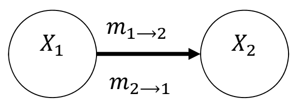
- Each node maintains its current marginal $P(X_i)$ (also called its belief).
- Initially, the marginal doesn't take the influence of the neighbors into account
- Note that a Node's belief is affected by its neighbors
- Neighboring nodes send messages to each other



Introduction

Intuition (using a chain as an example)

- Node X_2 receives a message $m_{1\to2}$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 $P(X_2)$

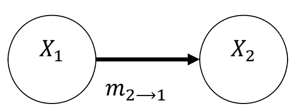


5

Introduction

Intuition (using a chain as an example)

- At convergence, the belief at a Node X_i is the marginal probability $P(X_i)$
- This is equivalent to a dynamic programming approach (very efficient!)



6

Introduction

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:

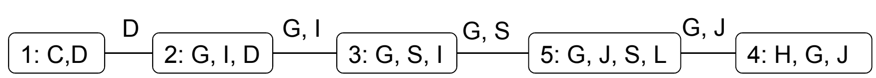
$$\tilde{P}_\Phi(\mathcal{X}) = \prod_{\phi_i \in \Phi} \phi_i(\boldsymbol{X}_i)$$

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

Cluster Graph

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

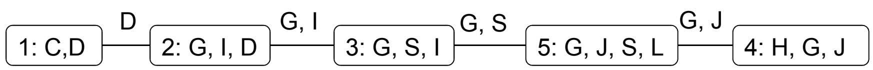
Example of a cluster graph



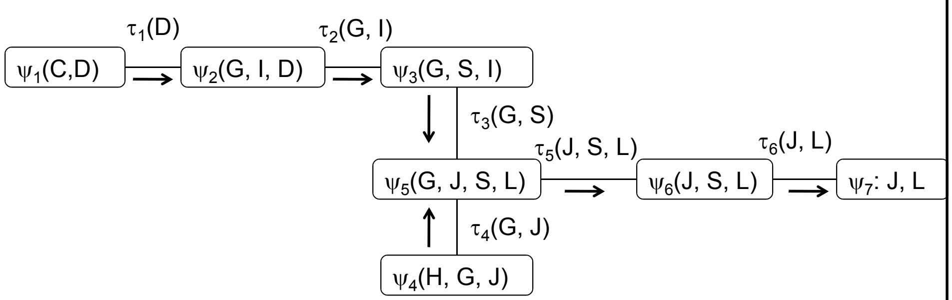
Cluster Graph

- Each factor $\phi \in \Phi$ must be associated with a cluster $\boldsymbol{C}$, denoted $\alpha(\phi)$, such that $Scope[\phi] \subseteq \boldsymbol{C}_i$.
- Each edge between a pair of clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ is associated with a sepset $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$.
- A cluster graph is a generalization of a clique tree

Example of a cluster graph

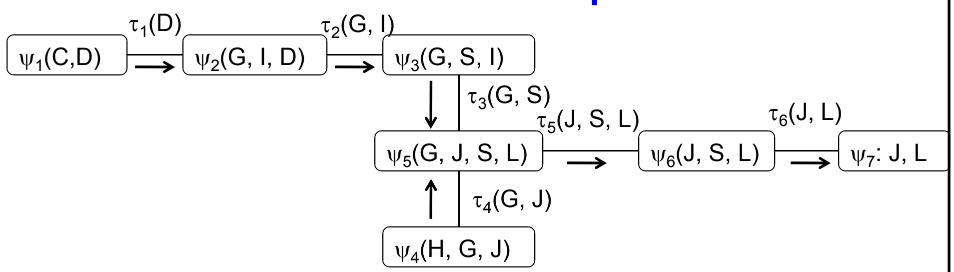


Cluster Graph



A new way to interpret variable elimination:
- (Recall: variable elimination defines a cluster graph)
- Factors ψ_i accept messages τ_j from another factor ψ_j
- Factors ψ_i also send their own messages τ_i to another factor

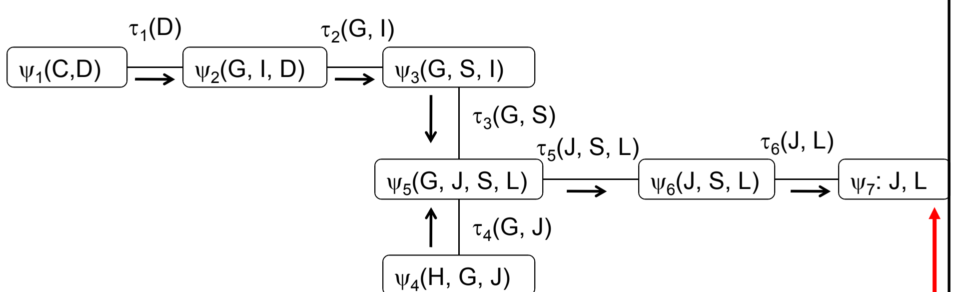
Cluster Graph



Step	Variable Eliminated	Factors Used	New Factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J, L, S)$	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	$\tau_7(J)$

13

Cluster Graph



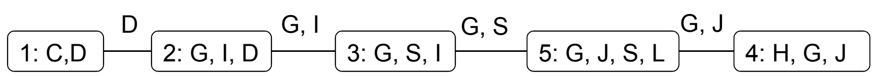
Note:

- 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

14

Cluster Graph

- T has the running intersection property if, whenever there is a variable X such that X $\in$ $\mathbf{C}_i$ and X $\in$ $\mathbf{C}_j$, then X is also in every cluster in the (unique) path in T between $\mathbf{C}_i$ and $\mathbf{C}_j$.
- Example: cluster tree below obeys the running intersection property (see G in C_2 and C_4)



- Running intersection property implies sepset $\mathbf{S}_{i,j} = \mathbf{C}_i \cap \mathbf{C}_j$.

15

Cluster Graph

- 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.

16

Clique Tree

- Let Φ be a set of factors over X. A cluster tree over Φ 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.

17

Message Passing: Sum Product

18

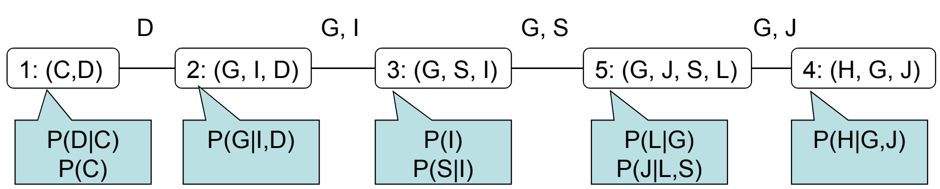
Message Passing: Sum Product

- Assume we are <u>given</u> 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

19

Message Passing: Sum Product

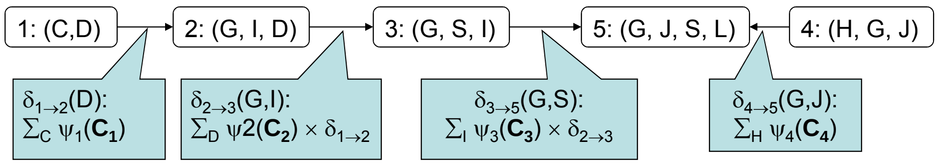
Example: Simplified Extended Student Clique tree



- First step: generate a set of initial potentials $\psi_i(\mathbf{C}_i)$ 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. $\mathbf{C}_5$.

20

Message Passing: Sum Product

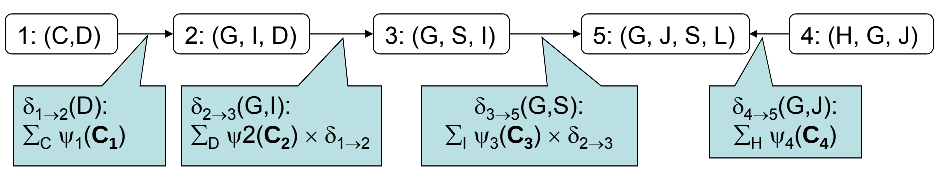


Execute the following:

- In $\mathbf{C}_1$: Eliminate C by $\sum_C \psi_1(C,D)$. Resulting factor has scope D. Send message $\delta_{1\to2}(D)$ to $\mathbf{C}_2$
- In $\mathbf{C}_2$: Define $\beta_2(G,I,D) = \delta_{1\to2}(D)\cdot\psi_2(G,I,D)$. Eliminate D to get a factor $\delta_{2\to3}(G,I)$ which is sent to $\mathbf{C}_3$.
- In $\mathbf{C}_3$: Define $\beta_3(G,S,I) = \delta_{2\to3}(G,I)\cdot\psi_3(G,S,I)$. Eliminate I to get a factor $\delta_{3\to5}(G,S)$ which is sent to $\mathbf{C}_5$.

21

Message Passing: Sum Product

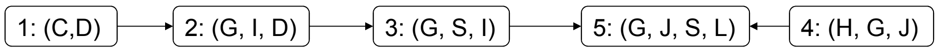


Execute the following:

- In $\mathbf{C}_4$: Eliminate H by $\sum_H \psi_4(H,G,J)$. Send factor $\delta_{4\to5}(G,J)$ to $\mathbf{C}_5$.
- In $\mathbf{C}_5$: Define $\beta_5(G,J,S,L)= \delta_{3\to5}(G,S) \cdot \delta_{4\to5}(G,J) \cdot \psi_5(G,J,S,L)$
- Sum out G, L, and S from β_5 to get P(J)

22

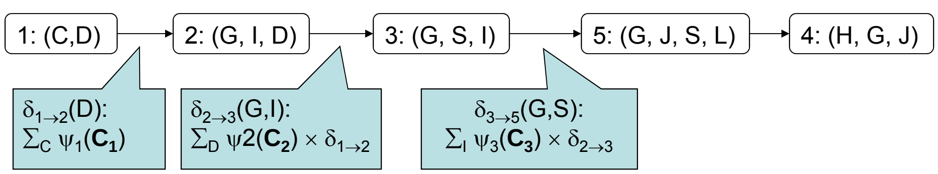
Message Passing: Sum Product



- Clique is ready when it has received all of its incoming messages eg.
  - $\mathbf{C}_4$ ready at the start
  - $\mathbf{C}_2$ ready only after getting message from $\mathbf{C}_1$
- $\mathbf{C}_1$, $\mathbf{C}_4$, $\mathbf{C}_2$, $\mathbf{C}_3$, $\mathbf{C}_5$ is a legal execution ordering for the tree rooted at $\mathbf{C}_5$
- $\mathbf{C}_2$, $\mathbf{C}_1$, $\mathbf{C}_4$, $\mathbf{C}_3$, $\mathbf{C}_5$ is not a legal execution ordering

23

Message Passing: Sum Product

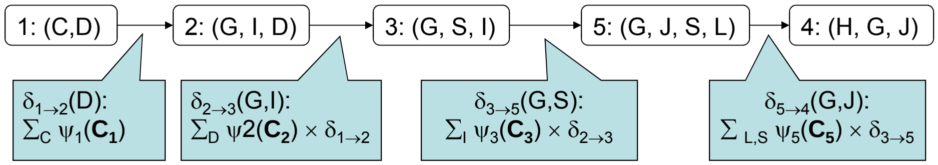


Could also define $\mathbf{C}_4$ as the root

- In $\mathbf{C}_1$: computation and message unchanged
- In $\mathbf{C}_2$: computation and message unchanged
- In $\mathbf{C}_3$: computation and message unchanged

24

Message Passing: Sum Product



$\mathbf{C}_4$ as the root

- In $\mathbf{C}_5$: Define $\beta_5(G,J,S,L) = \delta_{3\to5}(G,S)\cdot\psi_5(G,J,S,L)$. Eliminate S and L. Send out factor $\delta_{5\to4}(G,J)$ to $\mathbf{C}_4$.
- In $\mathbf{C}_4$: Define $\beta_4(H,G,J) = \delta_{5\to4}(G,S)\cdot\psi_4(H,G,J)$.
- Eliminate H and G from $\beta_4(H,G,J)$ to get P(J)

25

Message Passing: Sum Product

Clique-Tree Message Passing

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

26

Message Passing: Sum Product

1. Initial potentials
   - Each factor $\phi \epsilon \Phi$ is assigned to some clique $\alpha(\phi)$
   - The initial potential of $\mathbf{C}_j$ is:

$$\Psi_j(\boldsymbol{C_j}) = \prod_{\phi:\alpha(\phi)=j} \phi$$

   - Since each factor is assigned to exactly one clique, we have:

$$\prod_{\phi} \phi = \prod_{j} \Psi_j$$

27

Message Passing: Sum Product

2. Message passing
   - Definitions:
     - $\mathbf{C}_r$ = root clique
     - Nb_i = indices of cliques that are neighbors of $\mathbf{C}_i$
     - $p_r(i)$ = 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 $\mathbf{C}_i$ (except for the root) performs a message passing computation and sends message to upstream neighbor $\mathbf{C}_{pr(i)}$

28

Message Passing: Sum Product

Message from $\mathbf{C}_i$ to $\mathbf{C}_j$:

$$\delta_{i \to j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i}$$

Clique $\mathbf{C}_i$ multiplies incoming messages from its neighbors (except *j*) with its initial clique potential

Sums out all variables except those in the sepset between $\mathbf{C}_i$ and $\mathbf{C}_j$

Sends resulting factor to C_j

29

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 $\beta_r(\mathbf{C}_r)$, which represents

$$\tilde{P}_\Phi(\boldsymbol{C}_r) = \sum_{X - \boldsymbol{C}_r} \prod_{\phi} \phi$$

30

Message Passing: Sum Product

Procedure CTree-SP-Upward (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  $\mathbf{C}_r$ // Some selected root clique
)

1. Initialize-Cliques()
2. while $\mathbf{C}_r$ is not ready
3. Let $\mathbf{C}_i$ be a ready clique
4. $\delta_{i \to pr(i)}(S_{i,pr(i)}) \leftarrow$ SP-Message(i,p_r(i))
5. $\beta_r \leftarrow \psi_r \cdot \prod_{k \in Nb_{C_r}} \delta_{k \to r}$
6. return β_r

31

Message Passing: Sum Product

Procedure Initialize-Cliques ()

1. **for** each clique $\mathbf{C}_i$

2. $\psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j : \alpha(\phi_j) = i} \phi$

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

1. $\psi(\boldsymbol{C}_i) \leftarrow \psi_i \cdot \prod_{k \in (Nb_i - \{j\})} \delta_{k \to i}$
2. $\tau(\boldsymbol{S}_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(\boldsymbol{C}_i)$
3. return $\tau(\mathbf{S}_{i,j})$

32