

CS 3750 Machine Learning
Lecture 8

Graphical models
Exact inference

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

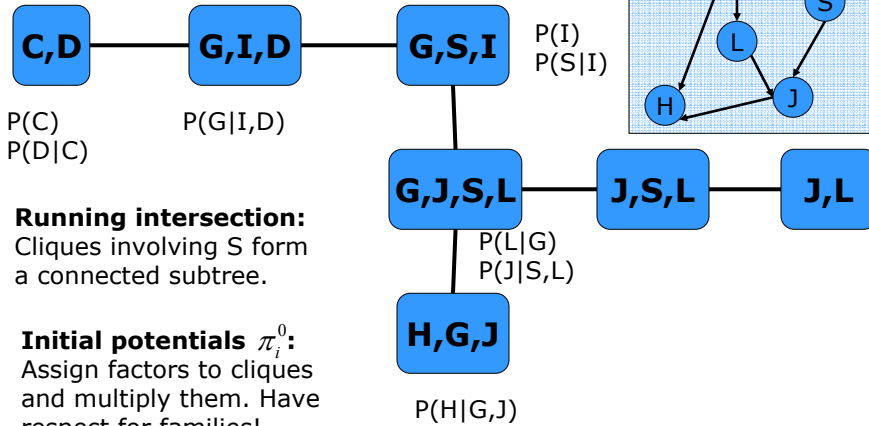
CS 3750 Advanced Machine Learning

Clique tree

- Clique tree = a tree decomposition of the graph
- Can be constructed:
 - from the induced graph
Built by running the variable elimination procedure
 - from the chordal graph
Built by running the triangulation algorithm
- We have precompiled the clique tree.
- So how to take advantage of the clique tree to perform inferences?

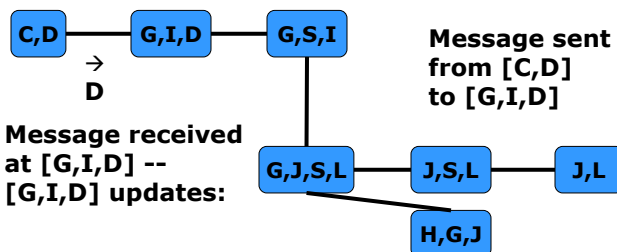
CS 3750 Advanced Machine Learning

Clique trees

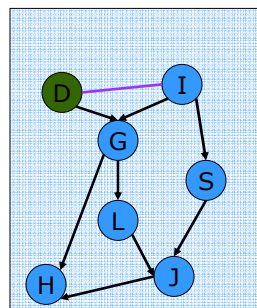
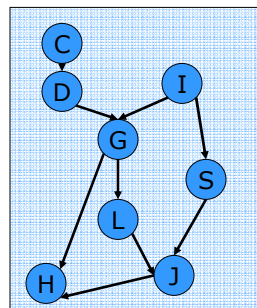


Message Passing VE

- Query for $P(J)$
 - Eliminate C: $\tau_1(D) = \sum_C \pi_1^0[C,D]$

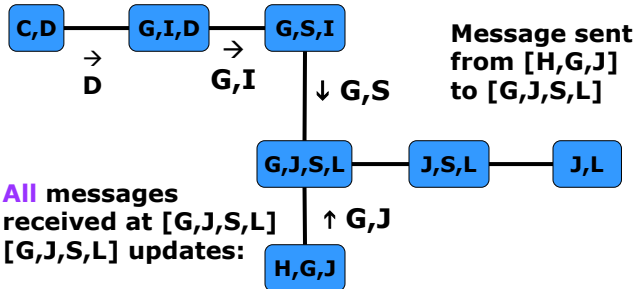


$$\pi_2[G,I,D] = \tau_1(D) \times \pi_2^0[G,I,D]$$



Message Passing VE

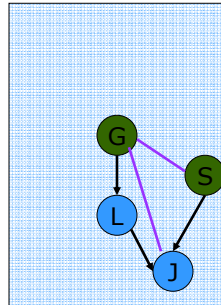
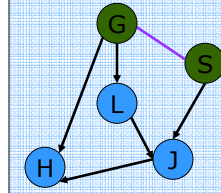
- Query for $P(J)$
 - Eliminate H: $\tau_4(G,J) = \sum_H \pi_5[H,G,J]$



All messages received at $[G,J,S,L]$
 $[G,J,S,L]$ updates:

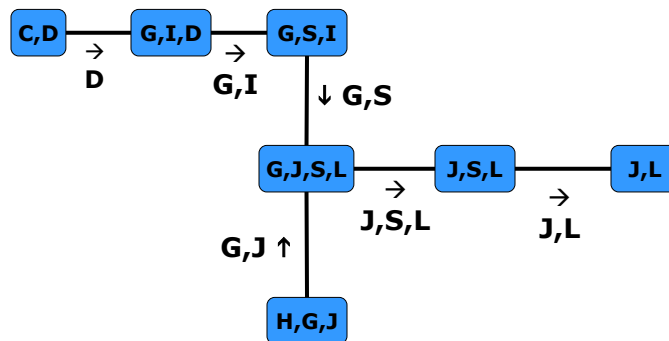
$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \pi_4^0[G,J,S,L]$$

And so on...



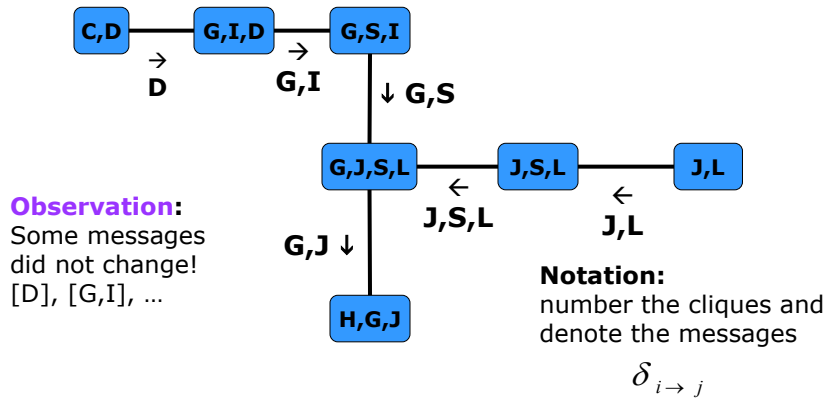
Message Passing VE

- Chose $[J,L]$ as the root clique
- ... and finish the inference



Message Passing VE

- Choose [H,G,J] as the root clique



CS 3750 Advanced Machine Learning

Message passing VE

- Often, many marginals desired
 - Inefficient to re-run inference from scratch
 - One distinct message per edge & direction
- Methods :**
 - Compute marginals for any node Y of the tree
 - Results in a *calibrated clique tree*

$$\sum_{C_i - S_{ij}} \pi_i = \sum_{C_j - S_{ij}} \pi_j$$

- Recap: three kinds of factor objects
 - Initial potentials, final potentials and messages

CS 3750 Advanced Machine Learning

Message passing VE

- **Shafer-Shenoy algorithm**

- Define a root
- asynchronous implementation of two passes: upward (send towards the root) and downward (send from the root)

Asynchronously do:

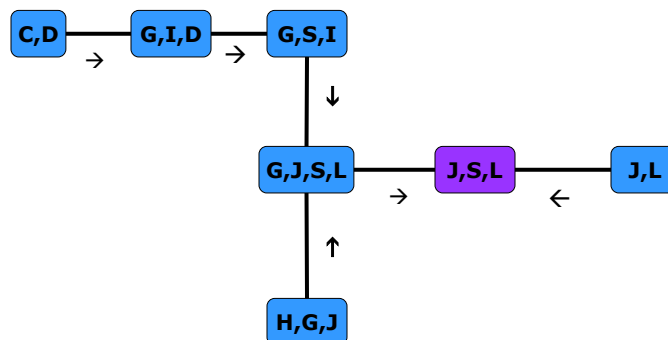
- node i ready to send m to node j when it has received a message from all other nodes

- Send message
$$\delta_{i \rightarrow j} = \sum_{C_i - S_{ij}} \pi_i \times \prod_{k \in N(i) - j} \delta_{k \rightarrow i}$$

- Marginalize root clique's ancillary vars

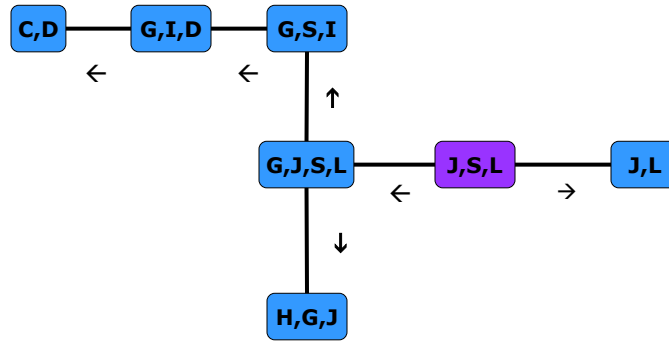
Shafer-Shenoy algorithm

- Choose $[J,S,L]$ as the root clique
- Pass 1: Send messages to the root



Shafer-Shenoy algorithm

- Choose [J,S,L] as the root clique
- Pass 2: Send messages from the root



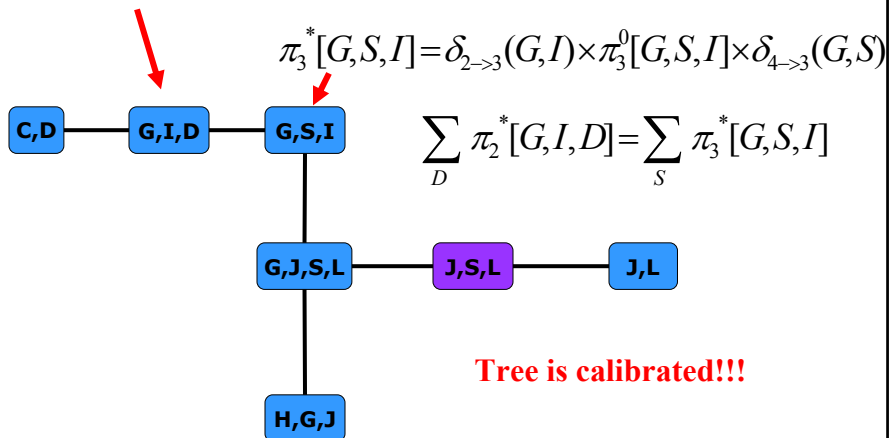
CS 3750 Advanced Machine Learning

Shafer-Shenoy algorithm

- After messages were received:

$$\pi_2^*[G,I,D] = \delta_{1 \rightarrow 2}(C) \times \pi_2^0[G,I,D] \times \delta_{3 \rightarrow 2}(G,I)$$

$$\pi_3^*[G,S,I] = \delta_{2 \rightarrow 3}(G,I) \times \pi_3^0[G,S,I] \times \delta_{4 \rightarrow 3}(G,S)$$

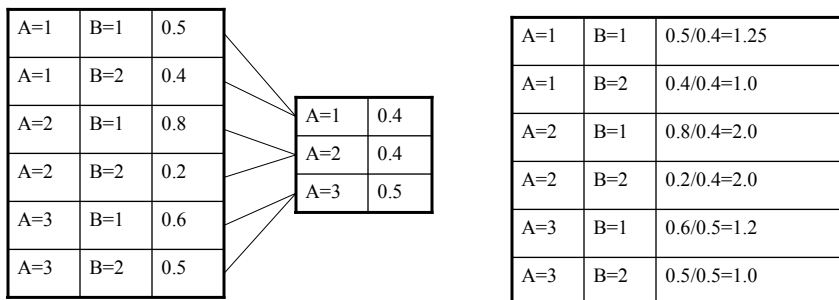


CS 3750 Advanced Machine Learning

Message passing: Belief propagation (BP)

- A slightly different (but equivalent) algorithm than Shafer-Shenoy
- Distributed algorithm
- Also leads to a new model of a **distribution**
 - More edges = larger expressive power
 - Clique tree becomes a model of distribution
 - Message passing preserves the model but changes the parameterization

Factor division



Inverse of factor product

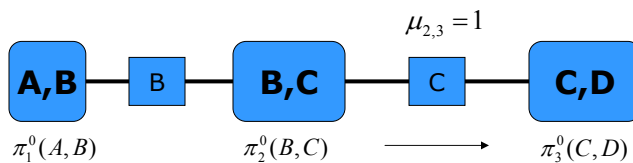
Message Passing: BP

- Each node: multiply all the messages and divide by the one that is coming from node we are sending the message to
 - Clearly the same as variable elimination

$$\delta_{i \rightarrow j} = \frac{\sum_{C_i - S_{ij}} \pi_i}{\delta_{j \rightarrow i}} = \frac{\sum_{C_i - S_{ij}} \prod_{k \in N(i)} \delta_{k \rightarrow i}}{\delta_{j \rightarrow i}} = \sum_{C_i - S_{ij}} \prod_{k \in N(i) \setminus j} \delta_{k \rightarrow i}$$

- Initialize the messages on the edges to 1

Message Passing: belief propagation



Store the last message on the edge and divide each passing message by the last stored.

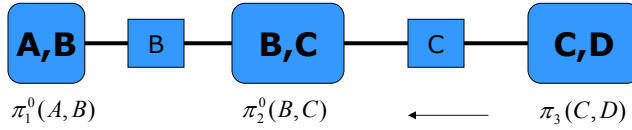
$$\delta_{2 \rightarrow 3} = \left(\sum_B \pi_2^0(B, C) \right)$$

$$\pi_3(C, D) = \pi_3^0(C, D) \frac{\delta_{2 \rightarrow 3}}{\mu_{2,3}} = \pi_3^0(C, D) \sum_B \pi_2^0(B, C)$$

$$\mu_{2,3} = \delta_{2 \rightarrow 3} = \left(\sum_B \pi_2^0(B, C) \right) \quad \text{New message}$$

Message Passing: BP

$$\mu_{2,3} = \left(\sum_B \pi_2^0(B, C) \right)$$



Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C, D) = \pi_3^0(C, D) \sum_B \pi_2^0(B, C) = \pi_3^0(C, D) \mu_{2,3}$$

$$\delta_{3 \rightarrow 2} = \left(\sum_D \pi_3(C, D) \right)$$

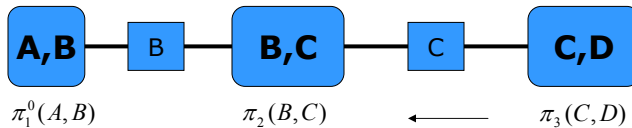
$$\pi_2(B, C) = \pi_2^0(B, C) \frac{\delta_{3 \rightarrow 2}}{\mu_{2,3}(C)} = \frac{\pi_2^0(B, C)}{\mu_{2,3}(C)} \times \sum_D \pi_3^0(C, D) \times \mu_{2,3}(C) = \pi_2^0(B, C) \times \sum_D \pi_3^0(C, D)$$

$$\mu_{2,3} = \delta_{3 \rightarrow 2} = \left(\sum_D \pi_3(C, D) \right) = \sum_D \pi_3^0(C, D) \sum_B \pi_2^0(B, C) \quad \text{New message}$$

CS 3750 Advanced Machine Learning

Message Passing: BP

$$\mu_{2,3} = \sum_D \pi_3^0(C, D) \sum_B \pi_2^0(B, C)$$



Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C, D) = \pi_3^0(C, D) \sum_B \pi_2^0(B, C)$$

$$\delta_{3 \rightarrow 2} = \left(\sum_D \pi_3(C, D) \right)$$

$$\pi_2(B, C) = \pi_2^0(B, C) \times \sum_D \pi_3^0(C, D)$$

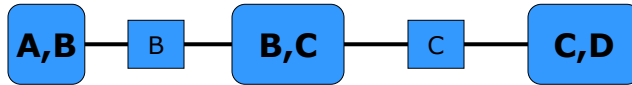
The same as before

$$\pi_2(B, C) = \pi_2^0(B, C) \frac{\delta_{3 \rightarrow 2}}{\mu_{2,3}(C)} = \pi_2^0(B, C) \times \frac{\sum_D \pi_3^0(C, D) \times \sum_B \pi_2^0(B, C)}{\sum_D \pi_3^0(C, D) \times \sum_B \pi_2^0(B, C)} = \pi_2^0(B, C)$$

CS 3750 Advanced Machine Learning

Message Passing: belief propagation

- When is the tree calibrated?
- No changes in clique or sepset potentials in the tree



- How to initialize sepsets?
- Initialize the messages on the edges to 1

Message Propagation: Belief propagation

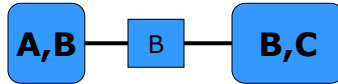
- **Lauritzen-Spiegelhalter algorithm**
- Two kinds of objects: clique and sepset potentials
 - Initial potentials are not kept!!!!
- Improved “stability” of asynchronous algorithm (repeated messages cancel out)
- **New distribution representation**
 - clique tree potential

$$\pi_T = \frac{\prod_{C_i \in T} \pi_i(C_i)}{\prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}(S_{ij})} = P_F(X)$$

- **It is the same at any step of belief propagation**

Message propagation: Belief propagation

Why?



$$P_F(A, B, C) = P_F(A, B)P_F(C | B, A) = P_F(A, B)P_F(C | B)$$

$$P_F(C | B) = \frac{\pi_2(C, B)}{P_F(B)}$$

$$P_F(A, B, C) = P_F(A, B) \frac{\pi_2(C, B)}{P_F(B)} =$$

$$\frac{\pi_1(A, B)\pi_2(C, B)}{P_F(B)} = \frac{\pi_1(A, B)\pi_2(C, B)}{\mu_{12}(B)}$$

CS 3750 Advanced Machine Learning

Message Propagation: Belief propagation

- **New distribution representation**

- clique tree potential

$$\pi_T = \frac{\prod_{C_i \in T} \pi_i(C_i)}{\prod_{(C_i \leftrightarrow C_j) \in T} \mu_{ij}(S_{ij})} = P_F(X)$$

- **It is the same at any step of belief propagation**

CS 3750 Advanced Machine Learning

Multiple queries

- Much caching possible over a clique tree
- Example: compute $P(X,Y)$, for each $X, Y \in \mathfrak{X}$
- Dynamic programming
 - Base case, X and Y are in neighbor cliques

$$P(C_i | C_j) = \frac{\pi_i(C_i)}{\mu_{ij}(C_i \cap C_j)} \quad P(C_i) \propto \pi_i$$

- Take advantage of conditional independence:

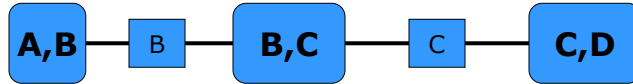
$$\exists I: C_i \perp C_j | C_l \quad P(C_i, C_j) = \sum_{C_l - C_j} P(C_i, C_l) P(C_l | C_j)$$

Incremental updates

- Fully-informed: all neighbors have sent their messages
- Calibrated -- messages and cliques agree on marginals:
 - fixed point of MP $\sum_{C_i - S_{ij}} \pi_i = \sum_{C_j - S_{ij}} \pi_j = \mu_{ij}$
- Evidence available in pieces
 - Re-running inference inefficient
- Express evidence in indicator vector
 - multiply it into some clique C_i
 - Example: Clique [A,B,C] and we want to set A=T, entries for the factor consistent with T are multiplied by 1, else 0
 - run one pass away from C_i to inform the rest
 - works for soft evidence as well

Out-of-clique queries

- I want $P(B, D)$, no clique with both B and D!
 - Build a new clique tree – expensive, or
 - Do variable elimination over *calibrated* tree



$$\begin{aligned} P(B, D) &= \sum_C P(B, C, D) \\ &= \sum_C \frac{\pi_2(B, C) \pi_3(C, D)}{\mu_{23}(C)} \\ &= \sum_C P(B | C) P(C, D) \end{aligned}$$

This is back to VE, we save if variables of interest are close in the clique tree.