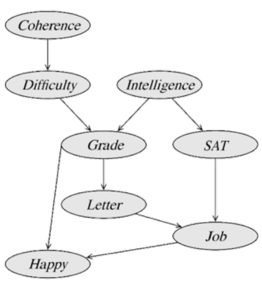
Sum-product algorithm

CS B553
Spring 2013

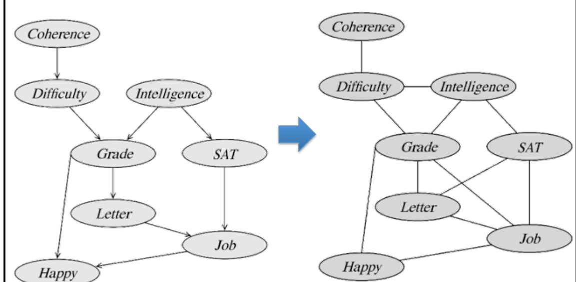
Announcements

- A2 (still) due next week

Recall we can convert a Bayes net to a Markov net…



Recall we can convert a Bayes net to a Markov net…



(Through moralization. But what do we lose in this process?)

We can use variable elimination to compute marginals…

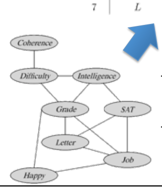
- Each step of VE "consumes" some factors, multiplies them together, marginalizes to eliminate a variable, and creates a new factor

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

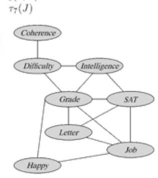
Execution of variable elimination defines an *induced graph*…

- Induced graph has an edge between X and Y iff X and Y appear in an intermediate factor during VE

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



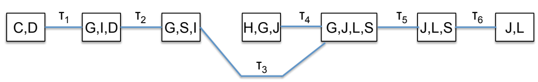
- Every maximal clique in the induced graph corresponds to an intermediate VE factor
- Every intermediate factor corresponds to a clique in the induced graph



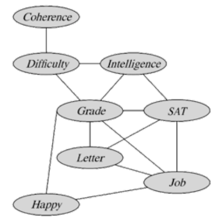
Execution of variable elimination also defines a *clique tree*…

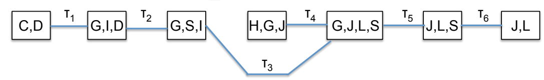
- Node for each subset of variables considered by VE
- Edge between X and Y if X is used to compute Y

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



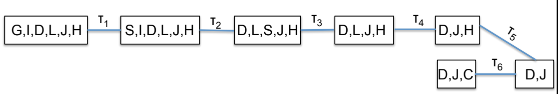
Nodes in the clique tree correspond to cliques in the induced graph…





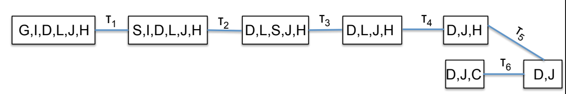
Different VE orderings yield different induced graphs and clique trees…

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



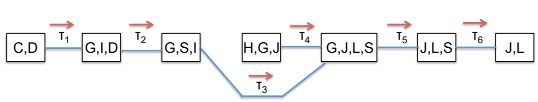
Important properties of clique trees

- A clique tree is always a tree (obviously)
- Clique trees exhibit the *running intersection property*
  - If variable X is part of node C and node D, then every node along the path from C to D also contains X
- If there is an edge between C and D, then the τ function "connecting" C and D has scope C ⋂ D



Inference through message passing

- Clique trees provide an alternative inference algorithm
  - Think of each node in clique tree as an "agent"
  - Each agent waits to receive its incoming message(s)
  - Once message(s) arrive, it computes a new message, based on incoming messages and its own factor, and sends the new message to its neighbor
  - Continue until all nodes have computed their message

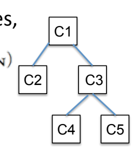


Inference through message passing

- Say you want to compute a marginal over variable X
  - Declare a node containing X to be the root of the clique tree
- Recall that Markov nets factor over cliques,

$$P(\mathbf{X}) = P(X_1, ..., X_N) = \frac{1}{Z}\phi_1(\mathbf{A_1}) \cdot \phi_2(\mathbf{A_2}) \cdot ... \cdot \phi_N(\mathbf{A_N})$$

  - We can assign each of these factors to one of the nodes in the clique tree,

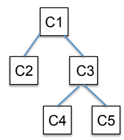


$$P(\mathbf{X}) = \frac{1}{Z}\prod_j \psi_j(\mathbf{C_j})$$

Sum-product algorithm

- First, each leaf C_i computes a message to send to its parent, C_j
  - This message has scope $\mathbf{S_{i,j}} = \mathbf{C_i} \cap \mathbf{C_j}$
  - In computing the message, we marginalize over the (single) variable in $\mathbf{C_i} - \mathbf{C_j}$
  - So the message is computed as,

$$\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{C_j}} \psi_i(\mathbf{C_i})$$

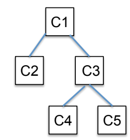


Sum-product algorithm

- Non-leaf nodes wait to receive messages from all their children
- Each node C_i then computes a message to send to its parent, C_j
  - This message (again) has scope $\mathbf{S_{i,j}} = \mathbf{C_i} \cap \mathbf{C_j}$
  - In computing the message, we (again) marginalize over the (single) variable in $\mathbf{C_i} - \mathbf{C_j}$
  - This message includes the messages (intermediate factors) from the children,

$$\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{C_j}} \psi_i(\mathbf{C_i}) \prod_{k\in\mathcal{N}(i)-\{j\}} \delta_{k\to i}(\mathbf{S_{k,i}})$$

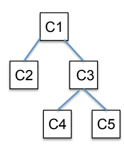
where $\mathcal{N}(i)$ is the set of neighbors of C_i



Sum-product algorithm

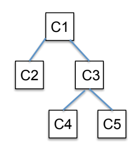
- Then, the root node computes a final message to send to its (imaginary) parent
  - This message, $\beta(\mathbf{C_r})$, gives us the marginal,

$$P(\mathbf{C_r}) = \frac{1}{Z}\beta(\mathbf{C_r})$$



Computing multiple marginals

- Often, we'll want to compute marginals over multiple variables of our graphical model
  - How to do this?

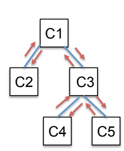


Sum-product belief propagation

- Instead of sending messages in one direction ("up" the tree), nodes send messages in all directions
  - Algorithm is almost exactly the same
  - Each node C_i sends a message to each of its neighbors C_j,

$$\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{C_j}} \psi_i(\mathbf{C_i}) \prod_{k\in\mathcal{N}(i)-\{j\}} \delta_{k\to i}(\mathbf{S_{k,i}})$$

  - Note that message sent to j does **not** use the message sent from j, to avoid double counting

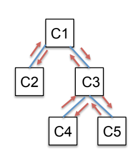


Sum-product belief propagation

- Once all messages have been exchanged, each node can compute its marginal,

$$P(\mathbf{C_i}) = \frac{1}{Z}\psi_i(\mathbf{C_i}) \prod_{k\in\mathcal{N}(i)} \delta_{k\to i}(\mathbf{S_{k,i}})$$

- Running time of sum-product BP?



Constructing a clique tree

- One way to find a clique tree is to choose a variable elimination ordering and "run" VE
- Another approach is to use a graph construction
  - If necessary, moralize to produce an undirected graph G.
  - Triangulate G to produce a chordal graph H. (Would like one with minimum clique size, but this is NP hard.)
  - Find maximal cliques in H. (Not NP hard for chordal graphs.)
  - Construct graph with nodes corresponding to max cliques in H, edges weighted according to degree of overlap. I.e. edge between C1 and C2 has weight $|C1 \cap C2|$
  - Find a maximum spanning tree on this graph to yield a clique tree.