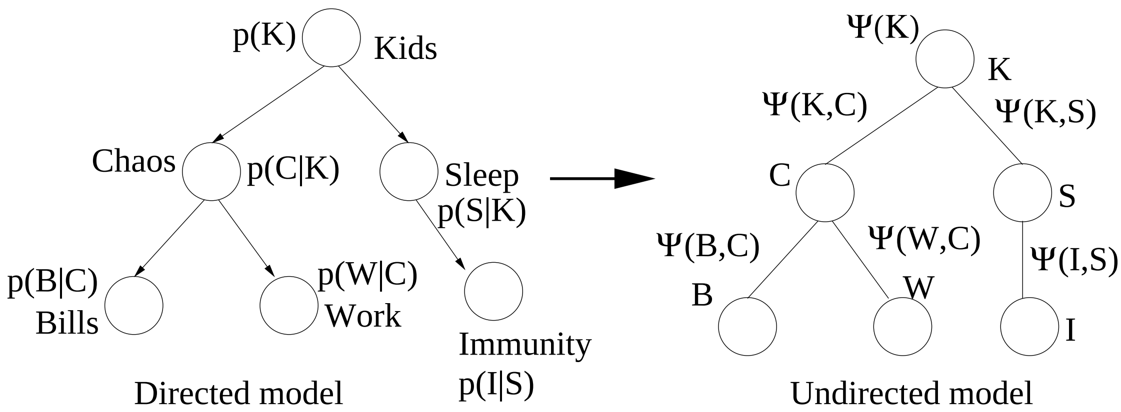
Lecture 8: Sum-product algorithm

- Directed and undirected trees
- Variable elimination on trees
- Sum-product algorithm (belief propagation) for trees
- MAP inference

Recall from last time

- We want to compute the probability of some query variables Y given values e for the evidence variables E
- Variable elimination is an algorithm that allows us to compute such probabilities exactly for general, directed or undirected graphical models
- Main ides is to re-arrange sums in order to (hopefully) compute more efficiently
- Today we discuss a more efficient algorithm, but which works *only for tree-structured models*
- Next time we put these two together

Directed trees



- Directed trees are such that their moral graph is a tree
- We can parameterize the corresponding undirected model by:

$$\Psi(\text{root}) = p(\text{root}) \text{ and } \Psi(x_j, x_i) = p(x_j | x_i)$$

for any nodes such that X_i is the parent of X_j

From undirected to directed trees

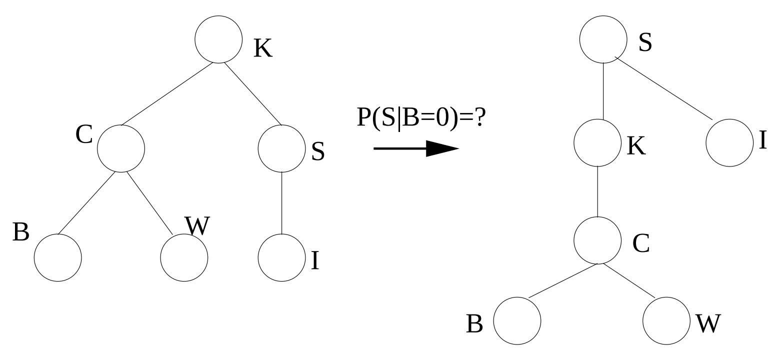
- Any undirected tree can be converted into a directed one by picking a root and directing arcs from there outwards
- We will parameterize an undirected tree by $\Psi(x_i)$, for all nodes i, and $\Psi(x_i, x_j)$, for all arcs (X_i, X_j)
- If we want to compute $P(Y|E)$, we introduce the evidence potential $\delta(x_i, \hat{x_i})$, for all evidence variables $X_i \in E$
- The potentials now become:

$$\psi^E(x_i) \begin{cases} \psi(x_i)\delta(x_i, \hat{x_i}) & \text{if } X_i \in E \\ \psi(x_i), & \text{otherwise} \end{cases}$$

Outline of variable elimination

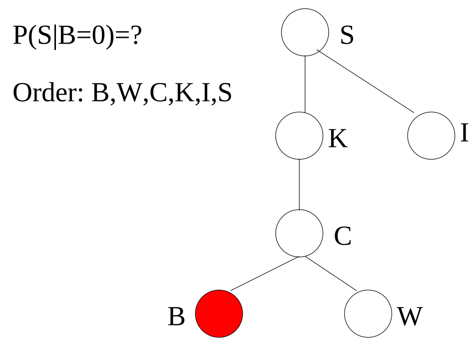
1. Pick a variable ordering with Y at the end of the list
2. Initialize the *active factor list*:
   - with the CPDs in a Bayes net
   - with the potentials in a Markov random field
3. Introduce the *evidence* by adding to the active factor list the evidence potentials $\delta(e, \hat{e})$, for all the variables in E
4. For $i = 1$ to n
   (a) Take the next variable X_i from the ordering.
   (b) Take all the factors that have X_i as an argument off the active factor list, and multiply them, then sum over all values of X_i, creating a new factor m_{X_i}
   (c) Put m_{X_i} on the active factor list

Variable elimination on undirected trees



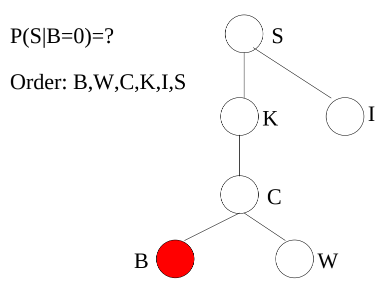
- The query node becomes root
- Traverse the resulting tree *depth-first*
- A node can only be eliminated *after all its children* have been eliminated
- What orderings arise in our example?

Intermediate factors



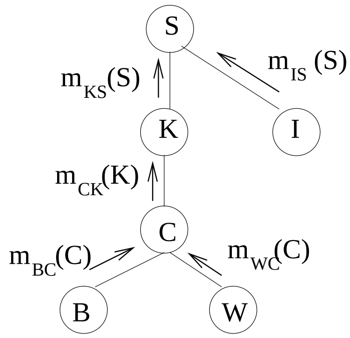
- Consider nodes C and K, which are connected
- C will be eliminated before K
- When we eliminate C, and create factor m_C, what potentials will get out of the active list? What variables will m_C depend on?

Intermediate factors



- $\psi(K, C)$ and $\psi^E(C)$ will have to be eliminated
- None of the factors that will be eliminated can reference B or W, since they would have been eliminated already
- None of the factors can reference I or S, (variables outside C's subtree), because of tree-ness
- So the factor that we create will be a *function of K only*!
- We can view this as a <u>*message*</u> computed by C and passed on to K. Call it $m_{CK}(K)$.

Message passing



The message passed by C to K will be:

$$m_{CK}(K) = \sum_{c} \left(\psi^E(c)\psi(c,k)m_{BC}(c)m_{WC}(c) \right)$$

where $m_{BC}(C)$ and $m_{WC}(C)$ are the messages from B and C.

Variable elimination for trees

- To eliminate node X_j, we have:

$$m_{ji}(x_i) = \sum_{x_j} \left(\psi^E(x_j)\psi(x_i,x_j) \prod_{k \in \text{neighbors}(x_j)-\{x_i\}} m_{kj}(x_j) \right)$$

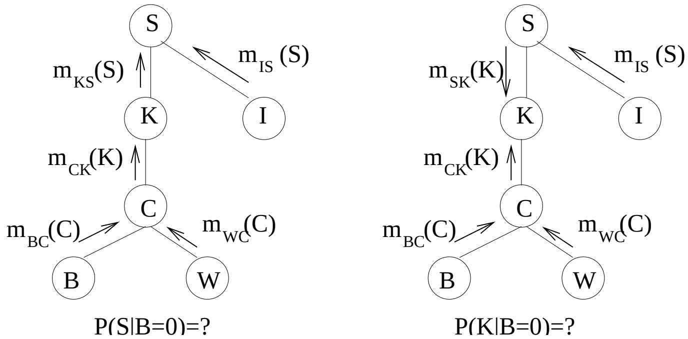
- The desired probability is computed as:

$$p(y|\hat{x_E}) \propto \psi^E(y) \prod_{k \in \text{neighbors}(Y)} m_{ky}(y)$$

- But what if we want to query *all* of the variables in the network?

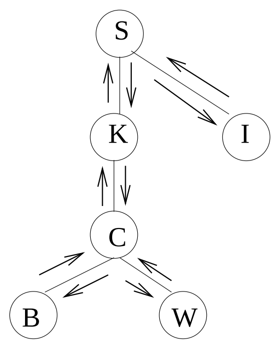
Abracadabra!

The same equations are sufficient to compute all probabilities!



Notice that when we ask for a different variable, almost all the messages needed are still the same

Computing all probabilities

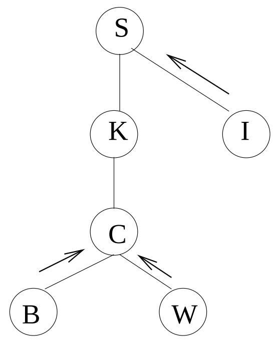


- Key idea: *messages can be re-used*!
- So we can compute all probabilities by computing all messages!
- We can use our previous equations to compute messages, but we need a protocol for when to compute them

Message-passing protocol

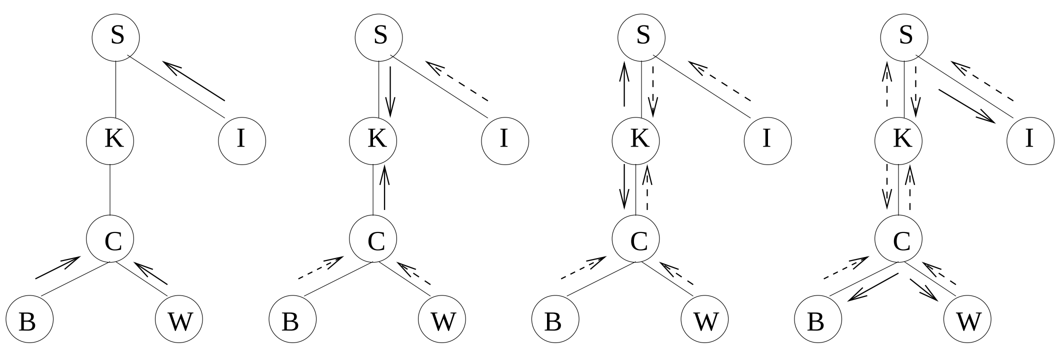
A node can send a message to a neighbor after it has received the messages from all its other neighbors.

Synchronous parallel implementation: any node with D neighbors sends a message after receiving messages on $d-1$ edges



What messages are sent next?

Message passing example



Past messages are dashed, current messages are in solid arrow.

Sequential implementation of the sum-product algorithm

1. Introduce the evidence (by putting in the evidence potentials)
2. Choose any node as root
3. Inward pass: Send all messages toward the root
4. Outward pass: Send all messages outward from the root
5. Compute the probabilities at all the nodes

Remarks

- A similar algorithm for inference in Bayes nets with v-structures but no undirected cycles (polytrees) is given by Pearl, called *belief propagation*
- The algorithm can be implemented even if the Bayes net has undirected cycles (there is more than one path from one node to another). This is called *loopy belief propagation*, and works very well in practice (though theoretical understanding is limited).

MAP inference

- We want to compute $\max_x p(x|E=e)$, or $\arg\max_x p(x|E=e)$
- To compute $\max_x p(x|E=e)$, we can use the same apparatus as for computing $p(x|E=e)$!
- This is because for our purposes, $\max$ behaves like a sum:

$$a \cdot b + a \cdot c = a \cdot (b + c) \qquad \max(a \cdot b, a \cdot c) = a \cdot (b, c)$$

- Hence, we can write down a variable elimination MAP inference algorithm and also a “max-product” algorithm

Finding the MAP configuration

- This can be a little tricky, because we can have ties, and we need to make sure that we don’t mix up the values from different assignments
- When we send message $m_{ji}^{\max}(x_i)$n we keep at node j and index list, $\delta_{ji}(x_i)$, with the indices of the x_j values that generated the $\max$.
- In the outward pass, we pick a value from $\delta_{ji}(x_i)$, x_j^*, and this remains set.