CS281A/Stat241A: Statistical Learning Theory

Elimination, Moralization and Sum-Product Algorithm (09/14/04)

*L*ecturer: Michael I. Jordan Scribes: Parvez Ahammad and Ying Zhang

1 Elimination for Undirected Graphical Models

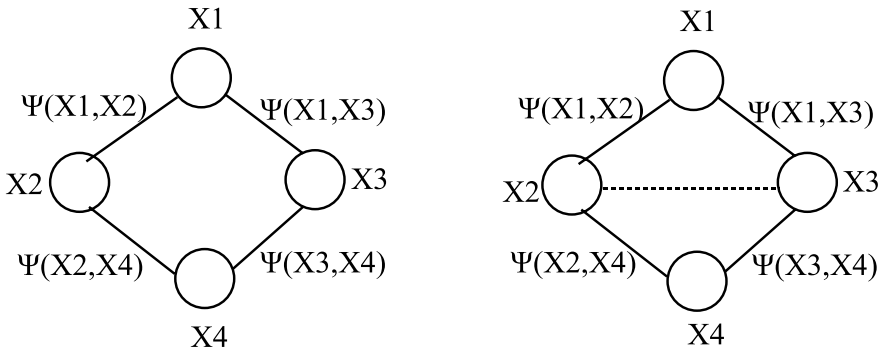
For an arbitrary undirected graphical model, the joint probability will factorize in the following way for the set of cliques $\mathcal{C}$:

$$\begin{aligned} p(x) &= \frac{1}{Z}\prod_{C\in\mathcal{C}}\psi_C(x_C) \\ p(x) &\propto \prod_{C\in\mathcal{C}}\psi_C(x_C) \end{aligned}$$

Compute $p(x)$ via elimination on $\prod_{C\in\mathcal{C}}\psi_C(x_C)$

$$\begin{aligned} p(x_i) &= \frac{m_j(x_i)}{\sum_{x_i} m_j(x_i)} \\ \text{where } m_j(x_i)) &= \sum_{x_{V\setminus i}}\prod_{C\in\mathcal{C}}\psi_C(x_C) \\ \text{and } \sum_{x_i} m_j(x_i) &= \sum_{x_V}\prod_{C\in\mathcal{C}}\psi_C(x_C) \equiv Z \end{aligned}$$

Example:



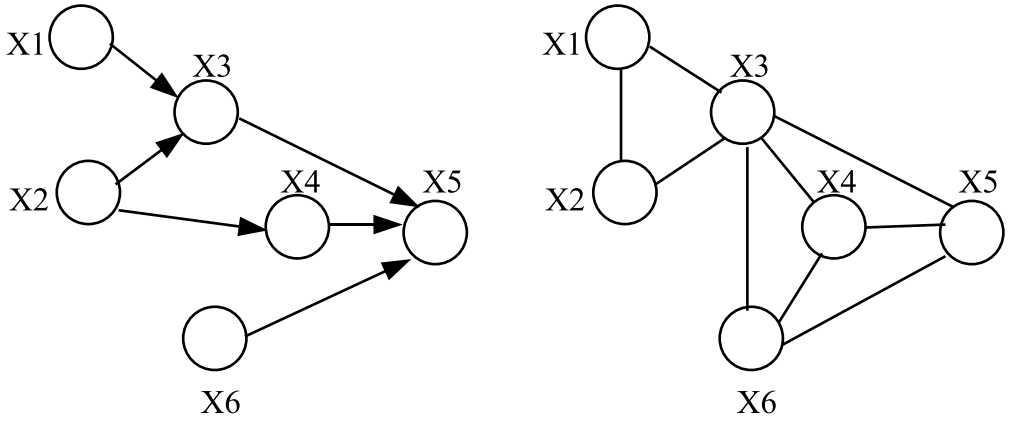
(Figure 1)

$$
\begin{aligned}
p(x_1) &\propto \sum_{x_2,x_3,x_4} \psi(x_1,x_2)\psi(x_1,x_3)\psi(x_2,x_4)\psi(x_1,x_4) \\
&= \sum_{x_2,x_3} \psi(x_1,x_2)\psi(x_1,x_3)m_4(x_2,x_3) \\
&= \sum_{x_2} \psi(x_1,x_2) \sum_{x_3} \psi(x_1,x_3)m_4(x_2,x_3) \\
&= \sum_{x_2} \psi(x_1,x_2)m_3(x_1,x_2) \\
&= m_2(x_1) \longrightarrow \textit{last message} \\
\text{so, } p(x_1) &= \frac{m_2(x_1)}{\sum_{x_1} m_2(x_1)}
\end{aligned}
$$

During elimination process, a new edge between x_2 and x_3 is created (i.e., a message containing both x_2 and x_3), which is shown in figure 1. Note that in undirected graphical models, we do not need to compute the normalization term explicitly; instead we can use the final message to quickly compute the term Z for all marginal probabilities.

2 Moralization

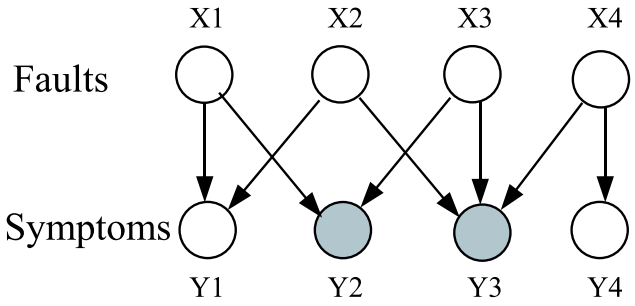
Moralization is a transformation from directed graphical model to an undirected graphical model. In brief, for each node in a directed graph, add an undirected edge between all pairs of parents of that node that are not already connected. Then change all edges in the graph from directed to undirected, as in figure 2.



(Figure 2)

Moralization captures the concept that we need to pay the expense of computing the term involving the parents of a child node while doing elimination.

Example: Fault Networks



(Figure 3)

We want to compute: $p(x_i|\bar{y})$ (where the top row of nodes denote faults x and the bottom row of nodes denote symptoms y, some of which are observed)

If we are using elimination process, we would have to do the whole computation all over again for each x_i as i varies. This is a waste of computations, especially since a lot of the intermediate steps are similar.

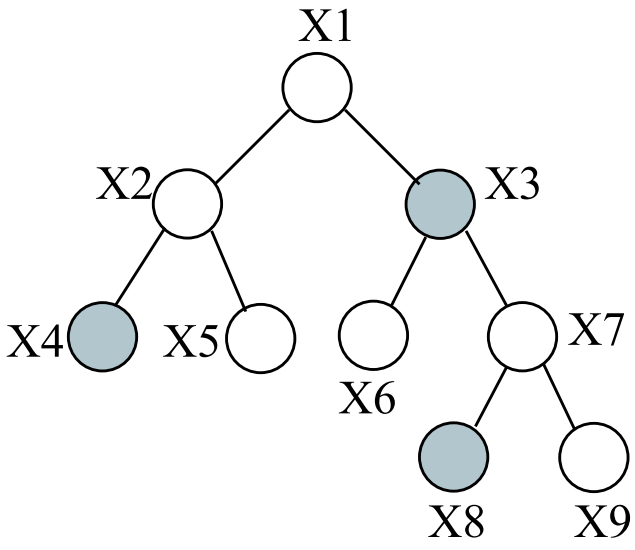
Ideally, we want to organize things in a way such that we do as little work as possible in a global sense (in terms of overall number of computations).

3 Sum-Product Algorithm

This algorithm gives us all of the singleton marginals, not just one. Also, it is only correct for trees.

Example: Loopy belief propagation algorithm (the sum-product algorithm on a graph with loops) in the error control coding domain is a good example. This algorithm finds an approximate answer, even though this is graph is not a tree (figure 3).

3.1 Sum-Product algorithm on Undirected Trees:



(Figure 4)

$$p(x_V) = \frac{1}{Z}\prod_{i\in\mathcal{V}}\psi_i(x_i)\prod_{(i,j)\in\mathcal{E}}\psi(x_i,x_j)$$

Conditioning: Let $x_i = \bar{x}_i$ for $i \in E$ (Evidence, or the shaded nodes in the graph whose values we have observed).

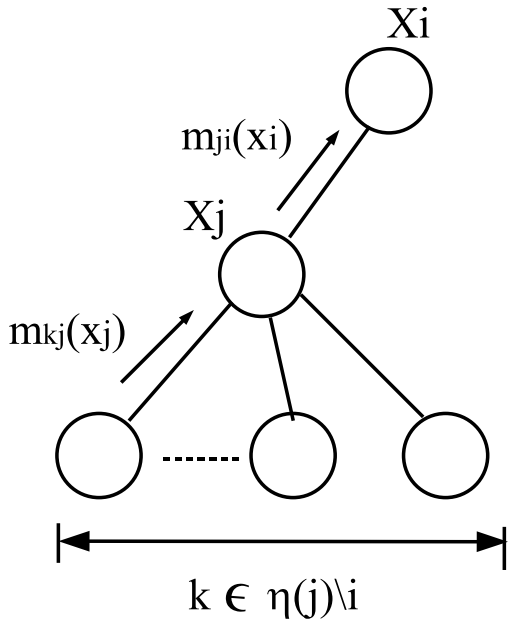
Evidence potentials are denoted as $\delta(x_i, \bar{x}_i)$. Using this notation, the unnormalized representation of conditional probability $p(x|\bar{x}_E)$ can be defined using the following definition,

$$\psi_i^E(x_i) = \begin{cases} \psi_i(x_i)\delta(x_i,\bar{x}_i),\ i \in E \\ \psi_i(x_i),\ i \notin E \end{cases}$$

(where E is the set of nodes with evidence), then the conditional probability becomes:

$$p(x|\bar{x}_E) \propto \prod_{i\in\mathcal{V}}\psi_i^E(x_i)\prod_{(i,j)\in\mathcal{E}}\psi(x_i,x_j)$$

Define a set of **messages** $m_{ij}(x_j)\ \forall(i,j) \in E$ that are updated according to the following rule:

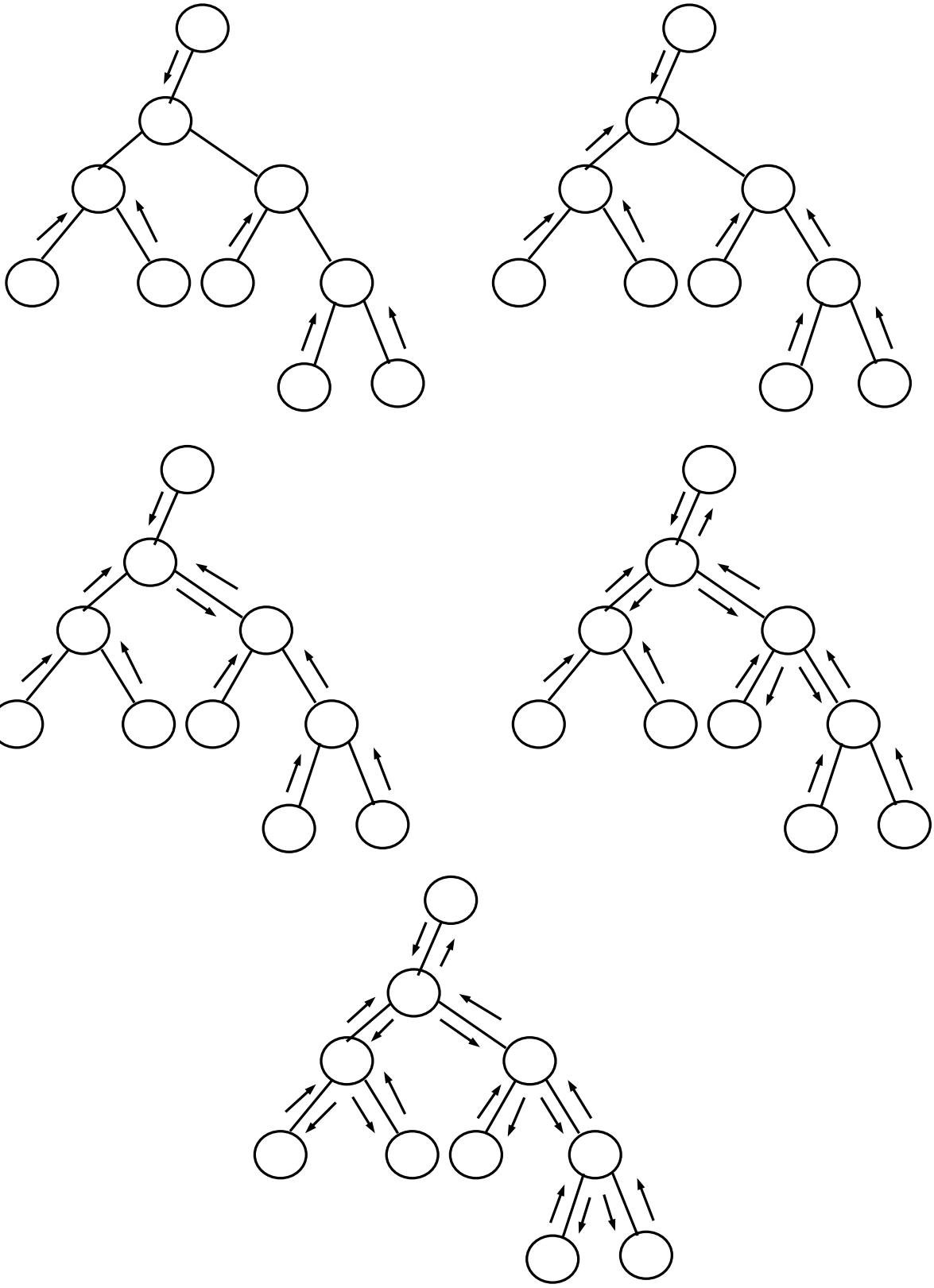


(Figure 5)

$$m_{ji}(x_i) = \sum_{x_j}\psi(x_i,x_j)\psi^E(x_j)\prod_{k\in\mathcal{N}(j)\setminus i}m_{kj}(x_j)$$
$$\text{where}\quad \psi(x_i,x_j) \equiv \psi(x_j,x_i)$$

If x_j has m states, then the algorithm has $O(m^2)$ complexity.

Protocol for sending messages: send a message from a node j to i only after j has received messages from all of its other neighbors, which is shown in figure 6.



(Figure 6)

The final step is:

$$p(x_i|\bar{x}_E) \quad \propto \quad \psi_i^E(x_i) \prod_{j \in \mathcal{N}(j) \setminus i} m_{ji}(x_i)$$

This has $O(2n)$ complexity where n is the number of edges.

We will show that $p(x_i|x_E)$ is defined as the marginal of node i according to the joint probability $p(x_V|x_E)$.

It's simple to pick a one-way flow based on a specific ordering, and use elimination algorithm to see that the $m_j(x_i)$ being computed is exactly the same as the Sum-Product algorithm.

$$m_j(x_i) \quad = \quad \sum_{x_j} \psi(x_i, x_j)\psi^E(x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_k(x_j)$$

Sum-Product algorithm is basically doing elimination algorithm in all possible ways in this undirected tree graph.

Question: How do we apply this to an arbitrary graph?

$\longrightarrow$ focus on the node we want, and pretend that we are actually looking at a part of the tree (ignoring the rest of the graph).

—*End of the Lecture*—