COS 513: Foundations of Probabilistic Modeling

Young-suk Lee

Lecture 5

1 Administrative

- Midterm report is due Oct. 29^{th}.
- Recitation is at 4:26pm in Friend 108.
- R is a computer language for statistical computing and graphics, and is highly recommended for this class. *RSeek* is a good search engine for R. [URL: www.r-project.org]

2 Project Ideas in Probabilistic Modeling

Super Topics:

1. Model Checking
2. Hierarchical Modeling (used in Sociology)
3. Information Geometry
4. Structural Learning
5. Online Learning/Estimation
6. Generative vs. Discriminative Modeling
7. Information theory and Statistics (such as code and data compression)
8. Application of X to Y
9. Graph Theory and Graphical Models

Resources:

1. Journals

   [JMLR] Journal of Machine Learning Research

   [MLJ] Machine Learning Journal

2. Conferences

   [NIPS] Neural Information Processing Systems

   [ICML] International Conference on Machine Learning

   [UAI] Uncertainty in Artificial Intelligence

   [AISTATS] Artificial Intelligence and Statistics

   [KDD] Knowledge Discovery and Data Mining

   [EMNLP] Empirical Methods in Natural Language Processing

   [SIGIR] Special Interest Group on Information Retrieval

3. Statistic Journals

   [JASA] Journal of the American Statistical Association

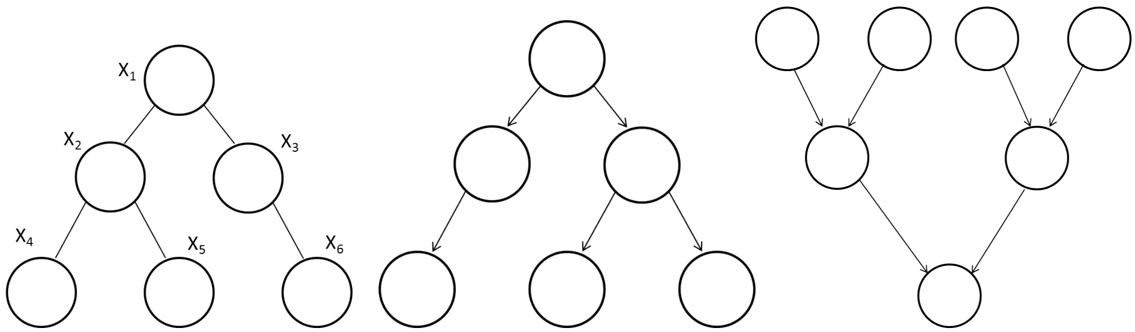
   [AAS] Annals of Applied Statistics

   [BA] Bayesian Analysis

   [AoS] Annals of Statistics (more theoritical)

4. Books

   - The Elements of Statistical Learning: Data Mining, Inference, and Prediction by Trevor Hastie, Robert Tibshirani, Jerome Friedman
     [URL: http://www-stat.stanford.edu/ tibs/ElemStatLearn/]
   - Bayesian Data Analysis by Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin
     [URL: http://www.stat.columbia.edu/ gelman/book/]
   - Pattern Recognition and Machine Learning by Christopher Bishop
     [URL: http://research.microsoft.com/en-us/um/people/cmbishop/PRML/]
   - Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman
     [URL: http://pgm.stanford.edu/]
   - Information Theory, Inference, and Learning Algorithms by David MacKay
     [URL: http://www.inference.phy.cam.ac.uk/mackay/itila/



(a) A undirected tree (b) A directed tree (c) Neither

Figure 1: Examples

3 Probability Propagation on Trees and The Sum-Product Algorithm

The *sum-product* algorithm (also known as the *belief propagation* algorithm) is a general inference algorithm for graphical models that are trees and that can compute all singe-node marginals. Although this algorithm does not apply to arbitrary graphs but only to trees, we study this algorithm for the following reasons:

1. Trees consist of a significant fraction of classical graphical models such as the hidden Markov model and the state-space model.
2. This algorithm provides insight to the completely general inference algorithm, the *junction tree* algorithm.
3. Later, we will see this algorithm as the basis for *approximate* inference with belief propagation.

3.1 Definition of Trees

Undirected Tree A undirected graph in which there is only one path between any pair of nodes. See Figure 1a.

Directed Tree Any graph whose moralized graph is an undirected tree. See Figure 1b.

3.2 Parameterization

We first consider the parameterization of probability distributions on undirected trees. Since the cliques are single nodes and pair of nodes, we get:

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

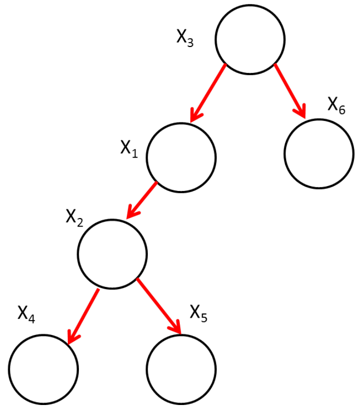


Figure 2: The tree in Figure 1a rooted at X_3. Note that the red edges do not denote the edges of a directed tree.

for a tree $T(V, E)$ with nodes V and edges E. In the directed case, we get:

$$p(x) = p(x_r) \prod_{(i,j) \in E} p(x_j | x_i), \tag{2}$$

where (i, j) is a directed edge such that i is the *unique* parent of j. Note that the following potential functions $\psi(x)$ and $\psi(x_i, x_j)$ shows that a directed tree is a *special case* of undirected trees and so we will only consider undirected trees:

$$\psi(x_r) = p(x_r), \tag{3}$$

$$\psi(x_i) = 1 if i \neq r, \tag{4}$$

$$\psi(x_i, x_j) = p(x_j | x_i), \tag{5}$$

$$Z = 1. \tag{6}$$

3.3 Evidence

Given the evidence $\overline{E}$, we define:

$$\psi_i^{\overline{E}}(x_i) = \begin{cases} \psi(x_i)\delta(x_i, xi) & i \in \overline{E}, \\ \psi(x_i) & i \notin \overline{E}. \end{cases} \tag{7}$$

Now we rewrite the conditional probability,

$$p(x | x^{\overline{E}}) = \frac{1}{Z^{\overline{E}}} \prod_{i \in V} \psi^{\overline{E}}(x_i) \prod_{(i,j) \in E} \psi^{\overline{E}}(x_i, x_j) \tag{8}$$

which has exactly the same form as $p(x)$. Therefore, we also do not pay special attention to evidence.

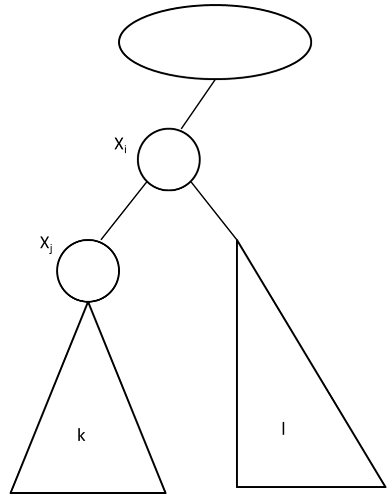


Figure 3: A undirected tree where k denote the descendants of node j and l denotes the sibling nodes and its descendants.

3.4 Undirected Eliminate

Recall the Elimination algorithm:

1. Choose an elimination ordering I such that query node f is last.
2. Place all potential functions on the active list.
3. Eliminate each node i by removing all potential functions referencing node i from the active list, taking the product over those functions referencing i, summing over x_i, and putting the resulting intermediate function back on the active list.

Similarly, on a tree, we treat f as the root of the tree, direct all edges to point away from f (not as a directed graphical model), and consider an ordering where each node is eliminated after its children. For example, given the tree in Figure 1a, if X_3 is our query node, we root the tree at X_3 and direct (in red) the edges away from X_3 (see Figure 2). There can be multiple elimination orderings. One possible elimination ordering I is: $\{X_5, X_4, X_2, X_1, X_6\}$, and another is: $\{X_6, X_5, X_4, X_2, X_1\}$. Notice that the graph from this preliminary step is in fact the *reconstituted* graph, and that the greatest clique size is 2. Since all elimination cliques are of size 2, the elimination algorithm is efficient for not only a particular query but also for any query.

3.5 More on Elimination Step

Consider X_i, X_j where X_i is closer to the root (see Figure 3). What fact is created when X_j is eliminated? We get the product over the following functions:

- $\psi(x_j)\psi(x_i, x_j)$

- no functions including node k (the descendants of node j)
- no functions including node l (the sibling nodes and its descendants)
- other functions of x_j

Once X_j is eliminated, the resulting factor is a function of x_i, which we call the *message* from node j to node i, or $m_{ji}(x_j)$. Thus, two equations follow:

$$\begin{aligned} m_{ji}(x_i) &= \sum_{x_j} \psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \backslash i} m_{kj}(x_j) && (9) \\ p(x_f | x_{\overline{E}}) &\propto \psi(x_f) \prod_{e \in N(f)} m_{ef}(x_f) && (10) \end{aligned}$$

where $N(i)$ is the set of neighbors of i. Note that in equation (10), we see no pairwise potential function because f has no parents.

3.6 Some Examples of Probability Inference

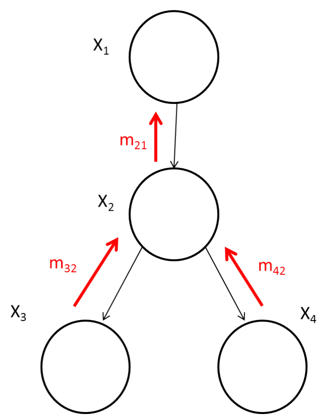
Let us try on some examples to understand the key insight in the *sum-product* algorithm. In Figure 4a, we wish to infer on X_1. Given the elimination ordering $I = \{3, 4, 2\}$, we compute:

$$\begin{aligned} m_{32} &= \sum_{x_3} \psi(x_3)\psi(x_3, x_2) && (11) \\ m_{42} &= \sum_{x_4} \psi(x_4)\psi(x_4, x_2) && (12) \\ m_{21} &= \sum_{x_2} \psi(x_2)\psi(x_2, x_1) m_{42}(x_2) m_{32}(x_2) && (13) \\ p(x_1) &\propto \psi(x_1) m_{21}(x_1) && (14) \end{aligned}$$

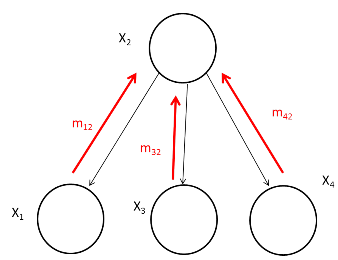
Likewise, we infer on X_2, but notice that we do not have to *recomputed* $m_{32}(x_2)$ and $m_{42}(x_2)$ (see Figure 4b). Thus, we only need to compute $m_{12}(x_2)$:

$$m_{12} = \sum_{x_1} \psi(x_1)\psi(x_1, x_2). \tag{15}$$

This *message* redundancy is the key insight in the *sum-product* algorithm which leads to a *message passing protocol* that will be discussed in more detail.



(a) rooted at X_1



(b) rooted at X_2

Figure 4: Both trees are identical graphs with different roots.