10-708: Probabilistic Graphical Models 10-708, Spring 2018

Lecture 14: Loopy Belief Propagation

Lecturer: Kayhan Batmanghelich *Scribes: Kyle Xiong, Chaitanya Malaviya*

1 Inference Problems

Probabilistic inference tasks include computing the marginal likelihood of the data given a model, the marginal probability of a subset of variables, conditional posterior probabilities for disjoint subsets, or maximal assignment. We have learned exact inference methods to compute these tasks such as brute force marginalization, variable elimination, and message passing algorithms. Brute force algorithms require us to marginalize over all possible states of all variables in the distribution. This can be computational intractable even for a small number of variables. By contrast, variable elimination and message passing can be far more computationally efficient. A simple idea of message passing is illustrated in our soldier example in which they each pass and receive messages to each other to count the total number of soldiers in the graph.

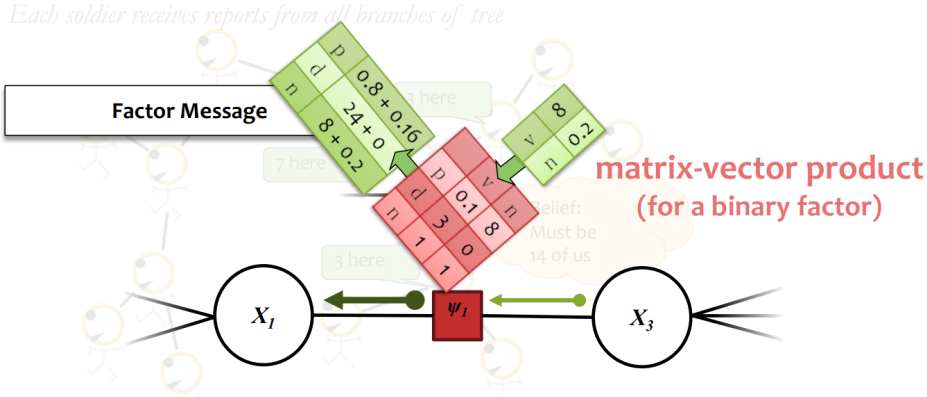

Figure 1: Example of message passing in an undirected graphical model using sum-product belief propagation. Factor ψ_1 receives a message from X_3 and passes it to X_1.

We have previously learned about sum-product belief propagation which operates on factor graphs to pass messages from nodes to factors and from factors to nodes. Illustrated in Figure 1 and by applying equation 1, a factor node receives messages from the neighboring node (X_3), applies its belief ψ_1 on the message, and propagates its resulting message to the target node X_1.

$$\mu_{\alpha \to i}(x_i) = \sum_{x_\alpha : x_\alpha[i] = x_i} \psi_\alpha(x_\alpha) \prod_{j \in \mathcal{N}(\alpha) \setminus i} \mu_{j \to \alpha}(x_\alpha[i]) \tag{1}$$

2 Junction Trees Revisited

Problems arise when a graph is not a tree and has some loops. When a graph has loops but the distribution does not have that many random variables, junction trees can be used to create a new singly connected graph. Given a graphical model, we can construct a junction tree by moralizing the graph, making a chordal graph, and then constructing the clique tree. However, while we can do exact inference, junction trees scale computationally with the size of the maximal cliques.

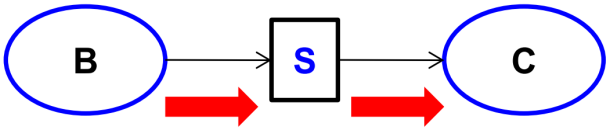

Figure 2: Message passing example in a clique tree model. Clique C receives a message from B. S is the separator between the two cliques.

Message passing on clique trees is described in Equations 2 and 3 where ϕ_S, ϕ_B, and ϕ_C are the potentials of the separator, clique B, and clique C respectively.

$$\widetilde{\phi}_S(x_S) \leftarrow \sum_{x_B \setminus S} \phi_B(x_B) \tag{2}$$

$$\phi_C(x_C) \leftarrow \frac{\widetilde{\phi}_S(x_S)}{\phi_S(x_S)} \phi_C(x_C) \tag{3}$$

Sometimes graphs can be both loopy and have a lot of random variables. In Figure 3, when factors are eliminated in the factor graph, new ones must also be introduced. For an $M \times M$ grid, the new factor would have 2^M entries. In such cases we cannot apply exact inference algorithms that only deal with small numbers of random variables. Therefore, we have to resort to approximate inferences.

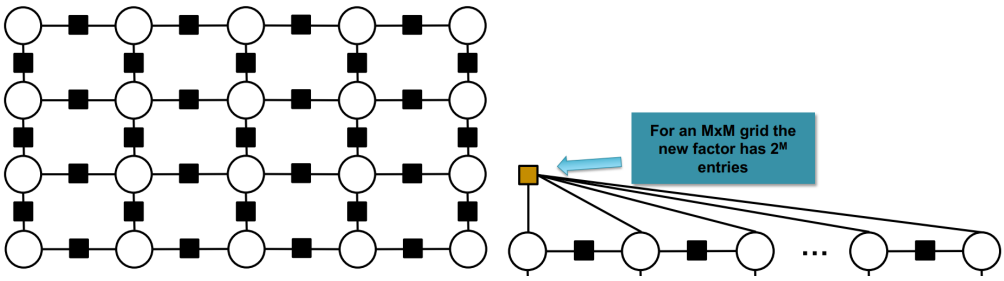

Figure 3: Factor graph variable elimination example. On the left is the original factor graph. On the right, the factor graph has had a portion of its factors eliminated.

3 Recap of Belief Propagation

3.1 Belief Propagation Without Factors

The message passed from a node k to node i in Figure 4 is computed using Equation 4. The sum terms before the product comprise the interaction term. Specifically, $\psi_{ij}(x_i, x_j)$ is the binary term which involves interactions between nodes i and j. $\psi_i(x_i)$ is the unary term which is the contribution of node i. When sending a message from i to j, i first receives messages from its neighbor nodes k (excluding node j). This is called the external evidence and is represented by the product $\prod_k M_{k\to i}(x_i)$ after the interaction term. This term summarizes the evidence from the rest of the random variables.

$$M_{k\to i} \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_k M_{k\to i}(x_i) \tag{4}$$

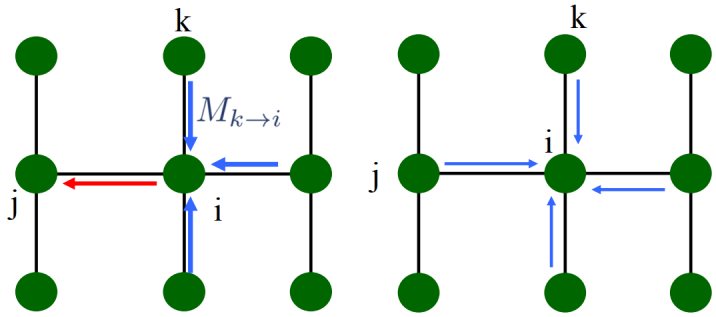

Figure 4: Message passing and receiving example.

After sending messages, they must also be passed back to form a belief according to Equation 5.

$$b_i(x_i) \propto \psi(x_i) \prod_k M_{k\to i}(x_i) \tag{5}$$

3.2 Belief Propagation in Factor Graphs

In factor graphs, there are two kinds of messages - those that are passed from nodes to factors and those from factors to nodes. For nodes, messages and beliefs are expressed in Equations 6 and 7. In Equation 6, i is the node and a is the factor the message is passed to. In Equation 7, the belief of node i is the product of all messages from its neighbors $N(i)$.

$$m_{i\to a}(x_i) = \prod_{c\in N(i)\setminus a} m_{c\to i}(x_i) \tag{6}$$

$$b(x_i) \propto \prod_{a\in N(i)} m_{a\to i}(x_i) \tag{7}$$

For factors, messages and beliefs are expressed in Equations 8 and 9. In Equation 8, the message from the factor a sums over all random variables except for x_i. f_a is the factor associated with factor a. For both

nodes and factors, the normalized beliefs are the marginal probabilities of the random variable.

$$m_{a\to i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a)\setminus i} m_{j\to a}(x_j) \tag{8}$$

$$b_a(X_a) \propto f_a(X_a) \prod_{i \in N(a)} m_{i\to a}(x_i) \tag{9}$$

In a factor graph with no cycles, it's feasible to pick any node to serve as the root, begin sending messages from the leaves to the root, and passing messages back to the leaves from the root. However, if there are cycles then a tree cannot be constructed from the graph.

In a loopy graph, we can start with random initialization of messages and beliefs. While the graph is not converged, we can continually update the beliefs and messages using the equations we have already defined. Termination conditions include: stopping after a fixed number of iterations, stopping when we observe no significant change in beliefs, or if the solution is not oscillatory. In the worst case scenario, the graph may not converge or may converge to a wrong solution, but in practice it converges most of the time.

To measure how close the approximation is to the actual distribution P where $P = \frac{1}{Z}\prod_{f_a \in F} f_a(X_a)$, we should find a distribution Q that minimizes the KL-divergence (Equation 10) between P and Q.

$$KL(P||Q) = \sum_X P(X) \log\left(\frac{P(X)}{Q(X)}\right) \tag{10}$$

KL-divergence is non-negative, but it is also not symmetric and does not measure distance. A new distance metric $D(Q_1, Q_2) = KL(Q_1||Q_2) + KL(Q_2||Q_1)$ is a symmetric variant of KL-divergence, but does not satisfy the triangular property ($D(Q_1, Q_2) \leq D(Q_1, Q_3) + D(Q_2, Q_3)$). It is not a real distance, but it's still useful. We use KL-divergence instead of real distance because the form of KL-divergence makes calculations easier.

4 Loopy Belief Propagation

4.1 Approximate Parameter Estimation

Belief Propagation can also be viewed as solving an optimization problem for a cost function called the Bethe free energy over possible marginal distributions. We will discuss the proof for this below.

Given a undirected graphical model, we assume a Gibbs distribution on a set of variables X in the graph, $P(X) = \frac{1}{Z}\prod_{f_a \in F} f_a(X_a)$. We aim to find a distribution Q that is a close approximation to P. We can then attempt to minimize the KL-divergence between P and Q. We use the KL-divergence as the distance metric since it is equivalent to minimizing an expectation of the real distribution and the negative entropy.

$$\begin{aligned} KL(Q||P) &= \sum_X Q(X) \log \frac{Q(X)}{P(X)} \\ &= \sum_X Q(X) \log Q(X) - \sum_X Q(X) \log P(X) \\ &= -H_Q(X) - \mathbf{E}_Q[\log P(X)] \end{aligned}$$

The above form allows us to estimate $\log P(X)$ without performing inference on P by sampling from $Q(X)$.

$$\begin{aligned} KL(Q||P) &= -H_Q(X) - \mathbf{E}_Q[\log \frac{1}{Z} \prod_{f_a \in F} f_a(X_a)] \\ &= -H_Q(X) - \sum_{f_a \in F} \mathbf{E}_Q \log f_a(x_a) + \log Z \end{aligned}$$

We refer to the sum of the first two terms in the above equation as the *free energy* $F(P, Q)$, where $F(P, Q) >= F(P, P)$.

$$F(P, Q) = -H_Q(X) - \sum_{f_a \in F} \mathbf{E}_Q[\log f_a(x_a)]$$

We aim to find a family of distributions $\mathcal{Q}$ that allows us to compute $F(P, Q)$ easily and is easy to sample from.

$$Q^* = \underset{Q \in \mathcal{Q}}{\arg\min}\, F(P, Q)$$

4.2 Bethe Approximation

So far, we have modified the original inference problem on $P(Q)$ to an optimization problem. Let us work out how to calculate the free energy $F(P, Q)$ for a general tree, given in Figure 5.

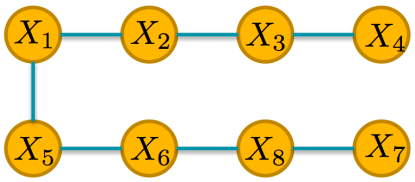

Figure 5: A general tree in the form of a chain

We can write the joint distribution of the graph in Figure 5 as a product of marginals over pairs of variables divided by the marginals over shared variables.

$$\begin{aligned} P(X_1 ... X_8) &= \frac{1}{Z} \phi_{43}(X_4, X_3) \phi_{32}(X_3, X_2) \phi_{21}(X_2, X_1) .. \phi_{87}(X_8, X_7) \\ &= \frac{P(X_3, X_4) P(X_2, X_3) P(X_2, X_1) ... P(X_8, X_7)}{P(X_3) P(X_2) P(X_1) ... P(X_8)} \end{aligned}$$

The entropy term in the free energy can be written as,

$$H(X_1, ... X_8) = -\sum_{a \in num} \mathbf{E}[\log p(X_a)] + \sum_{i \in den} \mathbf{E}[\log p(x_i)] \tag{11}$$

$$F(X_1, ... X_8) = -\sum_{a \in num} \mathbf{E}\left[\log \frac{p(X_a)}{f(X_a)}\right] + \sum_{i \in den} \mathbf{E}\left[\log \frac{p(x_i)}{f(X_i)}\right] \tag{12}$$

For a general tree, the joint probability is given by,

$$b(X) = \prod_a b_a(x_a) \prod_i b_i(x_i)^{1-d_i} \tag{13}$$

Here, d_i is the degree of node x_i. The Bethe Free Entropy H_{Bethe} and Bethe Free Energy F_{Bethe} for a general tree can be written as,

$$H_{Bethe} = \sum_a \sum_{x_a} b_a(x_a) \ln b_a(x_a) + \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{14}$$

$$F_{Bethe} = \sum_a \sum_{x_a} b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{15}$$

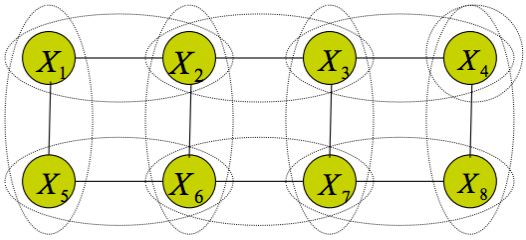

Figure 6: A loopy graph.

Lets extend this to the case of a general graph. In a general graph, we cannot compute the free energy F_{Bethe} exactly since there may be loops in the graph. If we choose the free energy $F(\hat{P}, Q)$ to be minimized as F_{Bethe} by pretending that our graph is a tree, we arrive at the *Bethe approximation*. For instance, the free energy F_{Bethe} for the loopy graph in Figure 6 can be written as,

$$F_{Bethe} = F_{12} + F_{23} + .. + F_{67} + F_{78} - F_1 - F_5 - 2F_2 - 2F_6 ... - F_8 \tag{16}$$

This approximation is easy to compute as the entropy term only involves a sum over pairwise and shared variables, as opposed to a sum over possible values. However, we must note that $F(\hat{P}, Q) = F_{Bethe}$ may not be well connected to the actual $F(P, Q)$.

4.3 Constrained Minimization

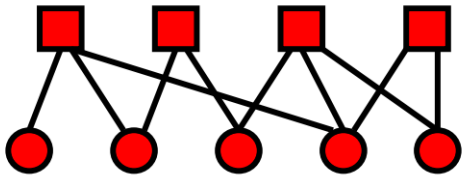

Figure 7: A factor graph.

Consider a factor graph with discrete random variables as shown in Figure 7. If x_a and x_i correspond to the factors and variables in this graph respectively, the Bethe Free Energy for this graph can be written as,

$$F_{Bethe} = \sum_a \sum_{x_a} b_a(x_a) \ln \frac{b_a(x_a)}{f_a(x_a)} + \sum_i (1 - d_i) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{17}$$

We would now like to minimize F_{Bethe} with respect to $b_a(x_a)$ and $b_i(x_i)$, subject to certain constraints.

$$\min_{b_a(x_a), b_i(x_i)} F_{Bethe}, \text{ subject to} \tag{18}$$

1. $\sum_{x_i} b_i(x_i) = 1$ and $\sum_{x_a} b_i(x_a) = 1$
2. $b_i(x_i) \geq 0$ and $b_a(x_a) \geq 0$
3. $\sum_{x_i \setminus x_j} b_i(x_i) = b_j(x_j)$ and $\sum_{x_a \setminus x_j} b_a(x_a) = b_j(x_j)$

We can solve the above constrained optimization problem by formulating it as a Lagrangian.

$$L = F_{Bethe} + \sum_i \gamma_i \left\{ \sum_{x_i} b_i(x_i) - 1 \right\} + \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{ai}(x_i) \left\{ \sum_{X_a \setminus x_i} b_a(X_a) - b_i(x_i) \right\} \tag{19}$$

Setting the derivatives of L with respect to the factors and variables to 0,

$$b_i(x_i) \propto \exp\left\{ \frac{1}{d_i - 1} \sum_{a \in N(i)} \lambda_{ai}(x(i)) \right\}$$

$$b_a(x_a) \propto \exp\left\{ -\log f_a(X_a) + \sum_{i \in N(a)} \lambda_{ai}(x(i)) \right\}$$

If we substitute the Lagrangian parameter $\lambda_{ai}(x_i) = \log(m_{i \to a(x_i)})$, where $m_{i \to a}(x_i)$ is the message from variable x_i to factor x_a, we obtain the original belief propagation equations.

$$b_i(x_i) \propto f_i(x_i) \prod_{a \in N(i)} m_{a \to i}(x_i)$$

$$b_a(x_a) \propto f_a(X_a) \prod_{i \in N(a)} \prod_{c \in N(a) \setminus a} m_{c \to i}(x_i)$$

This suggests that loopy belief propagation is in fact minimizing the Bethe Free Energy of the general graph.

To recap, we formulated the original inference problem on P as an optimization problem by introducing another distribution Q and minimizing the distance between these two distributions. For a complex graph, we know that computing marginal probabilities over arbitrary sets of random variables in Q is intractable. Using the Bethe energy function, which is formulated only in terms of unary and pairwise marginals (Eq 17), we can make Q tractable. In other words, the Bethe approximation allows us to reduce from exponential to polynomial constraints in the optimization problem. The above procedure is a variational method that allows us to make Q tractable by relaxing constraints on the original problem.

Graphical Intuition: For a set of discrete random variables, the space of probability distributions is a set of linear equations which all intersect with each other, forming a feasible set in the form of a polyhedron. Optimizing the Bethe Free Energy gives us a relaxed feasible set us that is the exterior of this original feasible set and is easier to optimize.