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

6 : Factor Graphs, Message Passing and Junction Trees

Lecturer: Kayhan Batmanghelich *Scribes: Sarthak Garg*

1 Factor Graphs

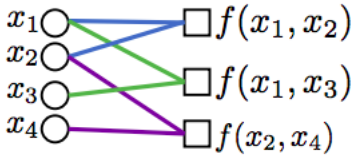
Factor Graphs are graphical representations which unify directed and undirected graphical models. Let G be a graphical model (directed/undirected) over a set of variables $\chi = \{x_1 \dots x_n\}$. For an undirected graphical model, the joint probability distribution can be written as

$$p(\boldsymbol{x}) = \frac{\prod_{i=1}^{m} \phi_i(\omega_i)}{Z} \text{ where } \omega_i \subseteq \chi \tag{1}$$

Each $\phi_i(\omega_i)$ is called a factor. The joint probability distribution for directed graphical models can also be written in this form with $m = n, Z = 1, \omega_i = \{x_i\} \cup \eta(x_i), \phi_i(\omega_i) = p(x_i|\eta(x_i))$ where $\eta(x)$ denotes the set of parents of variable x in the directed graph. From this factored representation, a factor graph $F(V, E)$ can be constructed. The set of vertices V of the factor graph consists for nodes for all the variables and all the factors, $V = \chi \cup \{\phi_1 \dots \phi_m\}$. The graph is undirected and bipartite, with edges only between the factors and variables. The edge set is given by, $\forall i, j, x_i \in \omega_j \rightarrow (x_i, \phi_j) \in E$.

Factor graphs can be used to describe Markov Random Fields, Conditional Random Fields and Bayesian Networks. For doing inference, i.e. computing conditional and/or marginal probability distributions over some subset of variables, the model should first be converted into a factor graph. If the factor graph is acyclic, then the Belief Propagation algorithm is used for exact inference. If the factor graph is cyclic, then the Junction Tree algorithm is used for inference.

Figure 1: Example of a factor graph



2 Message Passing/Belief Propagation

The underlying process of the message passing algorithm over an acyclic factor graph is pretty simple. Let $F(V, E)$ be a factor graph. Every node $v \in V$ sends exactly one message $\mu_{v \rightarrow w}$ to all its neighbours w. The message is a function defined over the variable x such that $v = x$ or $w = x$ (since the graph is bipartite, one of v, w will be a variable and the other one will be a factor). Once every node has received a message from all its neighbours, it forms a belief by combining all the received messages. A node v can send a message to its neighbour w, when it has received messages from all its neighbours other than w. The node v combines the

messages received from all the neighbours except w and sends it to w. It does this for each of its neighbour. The absence of any cycle ensures that this process terminates. The belief computation, combining process and the form of the message that a node sends out depends on whether the node is a variable or a factor.

2.1 Messages sent out by variables

Let x be a variable, $\{x_j\}$ be the set of values that x takes, $\Omega(x)$ be the set of neighbours of x, $w \in \Omega(x)$ be a neighbour to which x wants to send a message $\mu_{x\to w}(x_j)$ after combining all the messages $\{\mu_{k\to x}(x_j) \forall k \in \Omega \setminus w\}$.

$$\mu_{x\to w}(x_j) = \prod_{k\in\Omega(x)\setminus w} \mu_{k\to x}(x_j) \tag{2}$$

After receiving messages from all the neighbours, the following belief (function over values of the variable) is computed

$$b(x_j) = \prod_{k\in\Omega(x)} \mu_{k\to x}(x_j) \tag{3}$$

Figure 2: Example of a message sent out by a variable

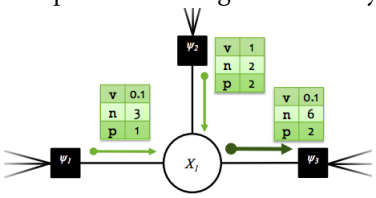
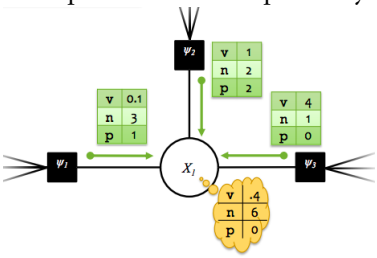


Figure 3: Example of belief computed by a variable



2.2 Messages sent out by factors

Let ϕ be a factor defined over variables Ω, $w \in \Omega$ be a variable which ϕ wants to send a message $\mu_{\phi\to w}$ after combining all the messages $\{\mu_{k\to x}(x_j) \forall k \in \Omega \setminus w\}$. Let I denote a valid assigment of values to all variables in Ω, i.e. $I \in \bigtimes_{x\in\Omega} d(x)$ where $d(x)$ represents domain of variable x. Let I_x denote the value of variable x in the assignment I.

$$\mu_{\phi \to w}(w_j) = \sum_{I \in \bigtimes_{x \in \Omega} d(x), I_w = w_j} \phi(I) \prod_{k \in \Omega \setminus w} \mu_{k \to \phi}(I_k) \tag{4}$$

After receiving messages from all the neighbours, the following belief (function over all possible assignments I) is computed as follows

$$b(I) = \phi(I) \prod_{k \in \Omega} \mu_{k \to \phi}(I_k) \tag{5}$$

Figure 4: Example of a message sent out by a factor

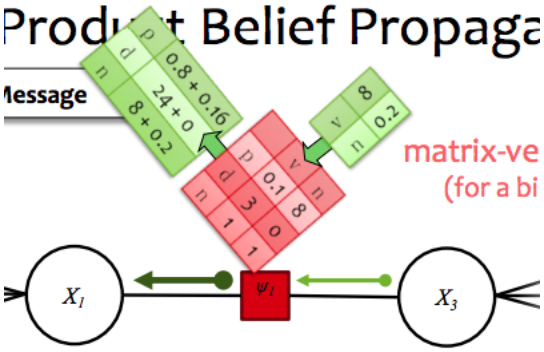
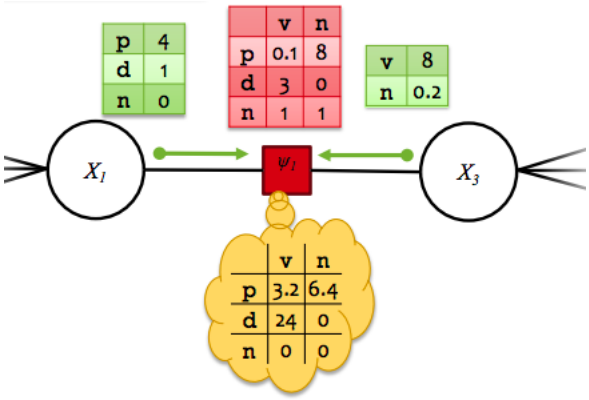


Figure 5: Example of belief computed by a factor



2.3 Sum-Product Belief Propagation for Acyclic Graphs

From the above sections, an algorithm for computing marginal probabilities over variables and subsets of variables for which factors are defined can be formulated as follows

To avoid numerical precision issues, all the messages should be stored and computed in log scale.

Algorithm 1: Sum-Product Belief Propagation

Data: a factor graph with no cycles
Result: exact marginals for each variable and factor
Initialize messages to uniform distribution;
$\mu_{w \to x}(x_i) = 1$;
Choose a root node;
Send messages from leaves up to the root;
Send messages from root down to the leaves;
Compute the beliefs (unnormalized marginals);
Normalize the beliefs and return the exact marginals;
$p_x(x_i) \propto b_x(x_i)$;
$p_{\Omega(\phi)}(I) \propto b_{\Omega(\phi)}(I)$;

3 Max-Product Belief Propagation for calculating most probable assignments

Consider a joint probability distribution defined over a set of variables. The maximal product assigment is an assigment of the variables that has the maximum joint probability density. Brute force search can take exponential time. A more efficient algorithm is a slight variation of the Sum-Product Belief Propagation algorithm.

- The message passed from factors to variables has the following formula. The notation is same as section 2.2

$$\mu_{\phi \to w}(w_j) = max_{I \in \bigtimes_{x \in \Omega} d(x), I_w = w_j} \phi(I) \prod_{k \in \Omega \setminus w} \mu_{k \to \phi}(I_k) \tag{6}$$

- Once a variable receives messages from all its neighbours, the maximal state of the variable is the one with the maximum belief. The notation is same as section 2.1

$$x^* = argmax_{x_j} \prod_{k \in \Omega(x)} \mu_{k \to x}(x_j) \tag{7}$$

4 Belief Propagation in loopy graphs

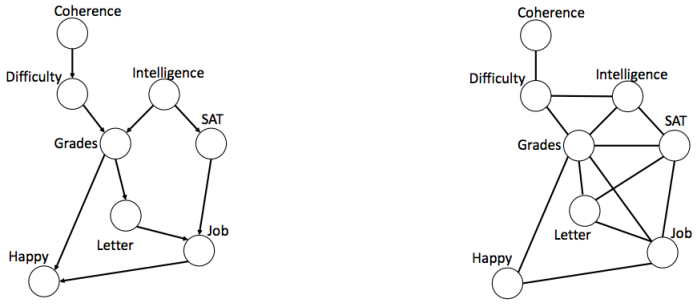
In the presence of cycles, the belief propagation algorithm does not compute the marginals. Even doing multiple rounds of belief propagation (unlike only one in case of no cycles) does not guarantee convergence to the correct marginals. The BP algorithm breaks down because doing variable elimination in graphs with cycles changes the graph structure. The junction tree algorithm deals with this by combining variables to make a singly connected graph which remains singly connected under variable elmination.

The junction tree algorithm has the following steps:

- Moralize the graph, if the graph is directed. This is because a node and its parents contribute to one factor (the CPD) in the joint distribution. Thus to convert it into an undirected graph, the parents should be connected to each other.

- Make the graph chordal. A chordal graph has a constraint that all cycles of 4 or more vertices should have a chord. This can be done by choosing an elimination ordering and performing variable elimination. During this process, some edges will be added in the intermediate graphs. Put all these edges in the original graph.
- Compute the maximal cliques in the chordal graphs. Build a clique graph with each node corresponding to one maximal clique.
- Compute the seperator sets for each pair of maximal cliques and compute a weighted clique graph. For each pair of maximal cliques (c_i, c_j) in the graph, add each edge between c_i and c_j with a weight $|c_1 \cap c_j|$
- Compute a maximum weighting spanning tree on the weighted clique graph to obtain a junction tree
- Run the sum-product belief propagation on the junction tree. The details of the message passing are described in the next section.

Figure 6: Example of moralization and triangulation of a DGM



5 Message Passing in Junction Trees

Let x be a node of a junction tree. Let $N(x)$ represents the neighbours of x. Let $w \in N(x)$ be a neighbour of x to which a message has to be sent. The message sent is

$$\mu_{x \to w}(u) = \sum_{v \in x \setminus w} \phi(u \cup v) \prod_{k \in N(x), k \neq w} \mu_{k \to x}(u \cup v) \tag{8}$$