Note on message-passing algorithm: 18 April 2019
CS-526 Learning Theory

An illustrative example

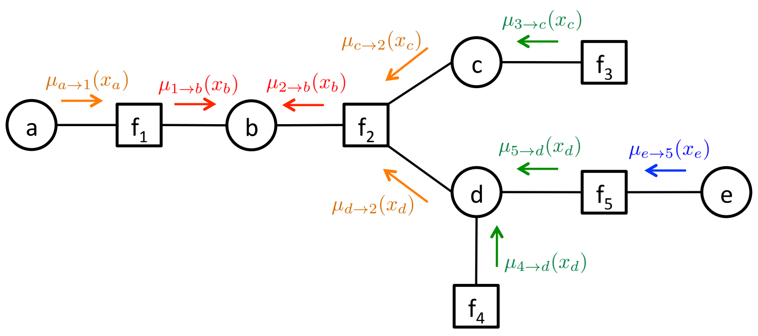


Figure 1: A factor graph for $p(\mathbf{x})$ and messages for computing the marginal $\tilde{p}(x_b)$

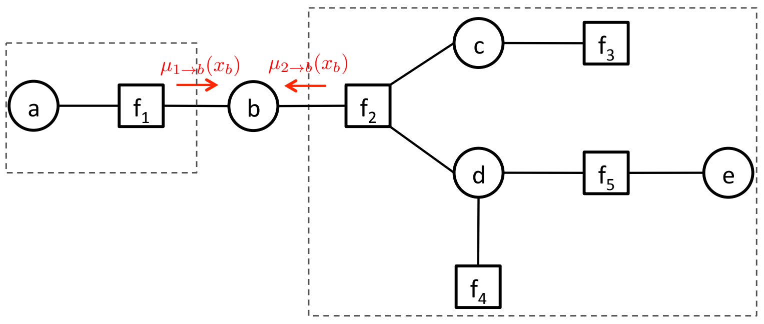
The probability $p(x_b) = \sum_{\mathbf{x}\setminus x_b} p(\mathbf{x})$ for the above factor graph is directly proportional to

$$\tilde{p}(x_b) = \sum_{x_a,x_c,x_d,x_e} f_1(x_a,x_b)f_2(x_b,x_c,x_d)f_3(x_c)f_4(x_d)f_5(x_d,x_e).$$

Suppose we want to compute $\tilde{p}(x_b)$. We can understand the computation as propagation of messages on a tree. First we can split the sum in two parts:

$$\tilde{p}(x_b) = \underbrace{\sum_{x_a} f_1(x_a,x_b)}_{\mu_{1\to b}(x_b)} \underbrace{\sum_{x_c,x_d,x_e} f_2(x_b,x_c,x_d)f_3(x_c)f_4(x_d)f_5(x_e)}_{\mu_{2\to b}(x_b)} = \mu_{1\to b}(x_b)\mu_{2\to b}(x_b)$$

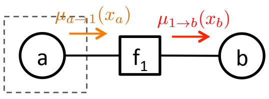
Graphically, we have split the tree in two parts around node b. We call $\mu_{1\to b}(x_b)$ the message from factor 1 to node b, and $\mu_{2\to b}(x_b)$ the message from factor 2 to node b. They correspond to the two red messages in Fig. 1.



The message $\mu_{1\to b}(x_b)$ can be written as

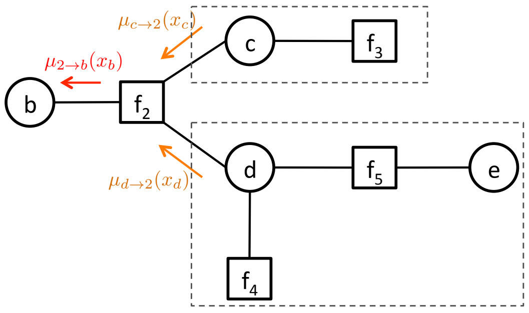
$$\mu_{1\to b}(x_b) = \sum_{x_a} f_1(x_a, x_b) \cdot \underbrace{1}_{\mu_{a\to 1}(x_a)} = \sum_{x_a} f_1(x_a, x_b)\mu_{a\to 1}(x_a)$$

where $\mu_{a\to 1}(x_a)$ is a message from the leaf node a to factor 1. This later message originating from the leaf variable node is initialized to 1.

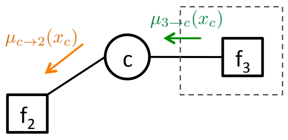


The same procedure is applied to $\mu_{2\to b}(x_b)$. We split the sum in two separable parts:

$$\begin{aligned}\mu_{2\to b}(x_b) &= \sum_{x_c,x_d,x_e} f_2(x_b, x_c, x_d) f_3(x_c) f_4(x_d) f_5(x_d, x_e) \\ &= \sum_{x_c,x_d} f_2(x_b, x_c, x_d) \underbrace{f_3(x_c)}_{\mu_{c\to 2}(x_c)} \underbrace{\sum_{x_e} f_4(x_d) f_5(x_d, x_e)}_{\mu_{d\to 2}(x_d)}\end{aligned}$$

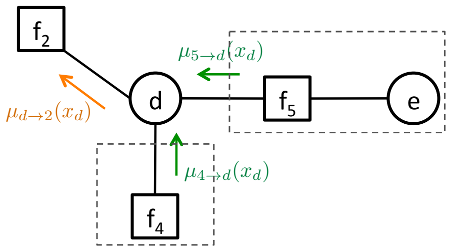


The message $\mu_{c\to 2}(x_c) = \mu_{3\to c}(x_c)$ directly propagates the message from the leaf factor 3 to node c with the initialization $\mu_{3\to c}(x_c) = f_3(x_c)$:



The other message $\mu_{d\to 2}(x_d)$ is split into

$$\mu_{d\to 2}(x_d) = \underbrace{f_4(x_d)}_{\mu_{4\to d}(x_d)} \underbrace{\sum_{x_e} f_5(x_d, x_e)}_{\mu_{5\to d}(x_d)}$$



Again, we initilialize the message from a leaf node $\mu_{e\to 5}(x_e) = 1$ and write

$$\mu_{5\to d}(x_d) = \sum_{x_e} f_5(x_d, x_e) \cdot \underbrace{1}_{\mu_{e\to 5}(x_e)} = \sum_{x_e} f_5(x_d, x_e)\mu_{e\to 5}(x_e).$$

Now, the messages are computed in decreasing order according to their distances from node b in Fig. 1 – namely, blue, green, orange, and red.

The general rules that apply to trees

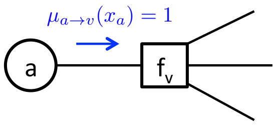
For any node b, assume we want to compute the marginal (the true marginal can be normalized at the end)

$$\tilde{p}(x_b) = \sum_{\mathbf{x}_{\sim b}} \prod_v f_v(\mathbf{x}_{\partial v}).$$

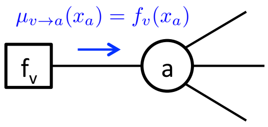
We can compute $\tilde{p}(x_b)$ with the following rules.

Initilialization:

- For every leaf node a we initilialize the message $\mu_{a\to v}(x_a) = 1$.



- For every leaf factor v we initilialize the message $\mu_{v\to a}(x_a) = f_v(x_a)$.

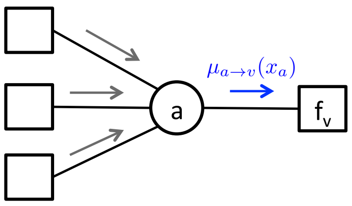


Propagation: Then the messages are computed in decreasing order according to their distances from node b.

- For non-leaf node a we compute the variable-to-factor message

$$\mu_{a\to v}(x_a) = \prod_{w\in\partial a\setminus v} \mu_{w\to a}(x_a). \tag{1}$$

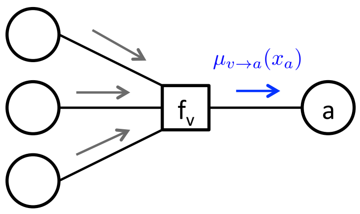
where ∂a denotes the set of neigboring factors of a. This is called the "product rule".



- For non-leaf factor v we compute the factor-to-variable message

$$\mu_{v\to a}(x_a) = \sum_{\mathbf{x}_{\partial v\setminus a}} f_v(x_a, x_{\partial v\setminus a}) \prod_{c\in\partial v\setminus a} \mu_{c\to v}(x_c). \tag{2}$$

  where ∂a denotes the set of neigboring factors of i. This is called the "sum rule".



Finally, we obtain $\tilde{p}(x_b)$ by taking products of all the incoming messages into node b:

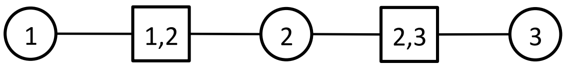
$$\tilde{p}(x_b) = \prod_{v\in\partial b} \mu_{v\to b}(x_b).$$

With $\tilde{p}(x_b)$ it is easy to compute the true marginal probability $p(x_b) = \tilde{p}(x_b)/\sum_{x_b}\tilde{p}(x_b)$.

Complexity

The computational complexity mainly comes from the sum rule (2). Let $x_c \in \mathcal{A}$. To compute $\mu_{v\to a}(x_a)$ the size of sum is $|\mathcal{A}|^{\deg(v)-1}$. When we consider the vector $\big(\mu_{v\to a}(x_a)\big)_{x_a\in\mathcal{A}}$ we have $|\mathcal{A}|^{\deg(v)}$ terms. Suppose the tree has $O(K)$ nodes. To compute $p(x_b)$ we have to propagate the messages over $O(K)$ edges so the complexity for this marginal is $O(K|\mathcal{A}|^{\deg(v)})$. In order to compute the marginals for all nodes a priori we would then have a complexity $O(K^2|\mathcal{A}|^{\deg(v)})$. But in fact we can propagate the messages from the leafs to node b and then from node b back to the leafs assuming we store them. Once we know the messages going in both directions for each edge we can compute all marginals so that the complexity can be reduced to $O(K|\mathcal{A}|^{\deg(v)})$.

Key concept: distributive law



To compute

$$\sum_{x_1,x_3\in\mathcal{A}} f_{1,2}(x_1,x_2) f_{2,3}(x_2,x_3)$$

there are $|\mathcal{A}|^2$ terms. If we use distributive law to write it as

$$\left(\sum_{x_1\in\mathcal{A}} f_{1,2}(x_1,x_2)\right)\left(\sum_{x_3\in\mathcal{A}} f_{2,3}(x_2,x_3)\right)$$

then each sum contains $|\mathcal{A}|$ terms. The complexity would be computing the two $|\mathcal{A}|$ terms plus a multiplcaition between the two sum.

Each time we have two operations that are distributive like $(+, \times)$ we can use this algorithm.

An example is (min, +). We have

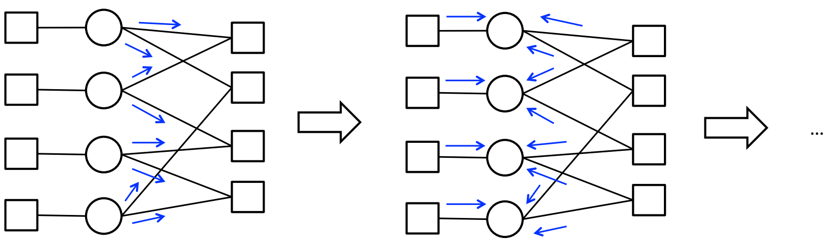
$$a + \min(b, c) = \min(a + b) + \min(a + c).$$

We can apply the same rules to the optimization problem

$$C(x_b) = \min_{\mathbf{x} \backslash x_b} \sum_s E_s(x_s)\,.$$

This gives the "min-sum" algorithm.

General loopy factor graph



For general loopy factor graph, we can use the same initialization if there are are leaf nodes (in this figure the leaf factor nodes often exist because in Bayesian inference we have a prior on the variables). Otherwise, if we do not have a prior (or if the prior is uniform) we can use $\mu_{v\to a} = 1$. Then we repeat the parallel schedule:

- Run from nodes to factors (left to right) using (1).
- Run from factors to nodes (right to left) using (2).

The update is stopped when messages have converged (if they do). This method is often good on locally-tree-like graphs (with large loops) but not always. The total complexity is again $O(K|\mathcal{A}|^{\deg(v)})$.