# Notes

Andrew Kontaxis, Chris White

October 23, 2017

## 1 Background

### 1.1 Sum Product Algorithm

Suppose we have a graphical model which is a connected undirected tree; in that case we can choose an arbitrary ordering for the nodes and write:

$$p(x_1, x_2, .., x_n) = \prod_i \psi_i(x_i) \prod_{i,j \in E} \psi_{i,j}(x_i, x_j)$$

Suspend disbelief about probabilistic interpretations for a moment and suppose we simply want to compute the quantity

$$p(x_s) := \sum_{i \neq s} p(x_1, x_2, .., x_s, ...x_n)$$

where we interpret the sum as being over the *state space* of the corresponding variables. We can write

$$\begin{aligned} p(x_s) &= \psi_s(x_s) \prod_{i \neq s} \psi_i(x_i) \prod_{i,j \in E} \psi_{i,j}(x_i, x_j) \\ &= \psi_s(x_s) \prod_{i \in \mathcal{N}(s)} \psi_i(x_i) \psi_{i,s}(x_i, x_s) \omega(T_i) \end{aligned}$$

where $\mathcal{N}(s)$ denotes the set of *neighbors* of node $s$ and $\omega(T_i)$ is a *weighting* of the subtree containing node $i$ formed by removing node $s$; this weighting is a function of all variables in $T_i$. Let us focus our attention on a single $i \in \mathcal{N}(s)$ for a moment, and imagine marginalizing out only the nodes in $T_i$ first:

$$\sum_{j \in T_i} \psi_s(x_s) \prod_{k \in \mathcal{N}(s)} \psi_k(x_k) \psi_{k,s}(x_k, x_s) \omega(T_k) = \kappa(x_s, x_{V \setminus T_i}) \sum_{j \in T_i} \psi_k(x_i) \psi_{i,s}(x_i, x_s) \omega(T_i)$$

We should now see that in order to complete this computation, the weighting need only be given to us as a function of $x_i$ alone. I.e., we can write

$$\kappa(x_s, x_{V \setminus T_i}) \sum_{x_i} \psi_i(x_i) \psi_{i,s}(x_i, x_s) \omega(x_i)$$

where $\omega(x_i)$ is given by

$$\omega(x_i) := \sum_{T_i \setminus i} \prod_{k \in \mathcal{N}(i) \setminus s} \psi_k(x_k) \psi_{k,s}(x_k, x_i) \omega(T_k)$$

and now we begin to see the recursive nature of our task. We can then proceed to marginalize out the variables in each of the other subtrees resulting in an expression of the form

$$\psi_s(x_s) \prod_{i \in \mathcal{N}(s)} \left( \sum_{x_i} \psi_i(x_i) \psi_{i,s}(x_i, x_s) \omega(x_i) \right)$$

Consequently, to compute the marginal $p(x_s)$, each neighbor $i$ of $s$ needs to pass a “message” to node $s$ which is a function purely of $x_s$, specifying the ”weighting” of the subtree $T_i$ *conditional on* the value $x_s$:

$$\mu_{i \to s}(x_s) := \sum_{x_i} \psi_i(x_i) \psi_{i,s}(x_i, x_s) \prod_{k \in \mathcal{N}(i) \setminus s} \mu_{k \to i}(x_i)$$

Note that the messages $\mu$ are proxies for the "true" weightings $\omega$ and that whenever $\mu \equiv \omega$ we have a fixed point and can compute

$$p(x_s) = \psi_s(x_s) \prod_{i \in \mathcal{N}(s)} \mu_{i \to s}(x_s).$$

This distinction is particularly important in the case when we want to apply this algorithm to graphs with cycles; otherwise we can iteratively updating the messages by beginning at the leaves with

$$\mu_{\ell \to q}(x_q) = \sum_{x_\ell} \psi_\ell(x_\ell) \psi_{\ell,q}(x_\ell, x_q)$$

### 1.1.1 An Example