JASON PACHECO

FACTOR GRAPH REVIEW:

Recall, factor graphs model the implicit Markov properties of a function.

EXAMPLE:

Let,

$$g(x_1, x_2, x_3, x_4, x_5) : S \longmapsto \mathbb{R}$$

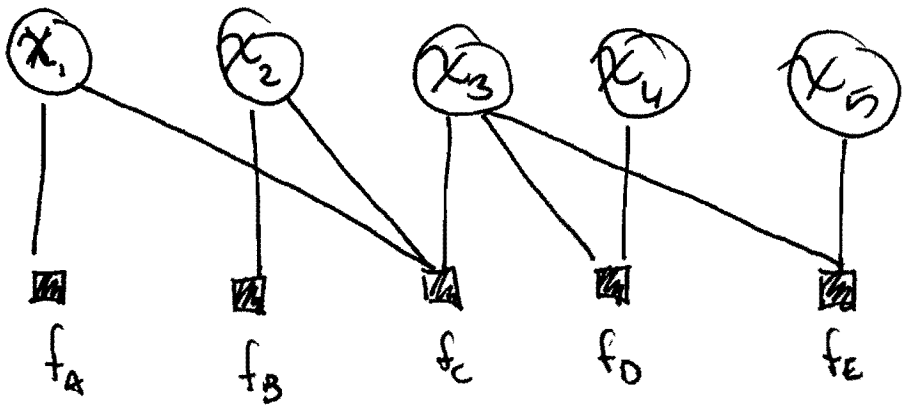
Where

$$S = A_1 \times A_2 \times \cdots \times A_5$$

Assume the "global" function $g(\cdot)$ can be expressed as a product of factors,

$$g(x_1, \cdots, x_5) = f_A(x_1)\, f_B(x_2)\, f_C(x_1, x_2, x_3)\, f_D(x_3, x_4)\, f_E(x_3, x_5)$$

We can express this factorization with a bipartite factor graph as,



Associated with the global function $g(x_1, \cdots, x_5)$ are the marginal functions $g_i(x_i)$ $\forall_i \in \{1, \cdots, 5\}$. We wish to develope an algorithm to compute these marginals that exploits the Markov properties of $g(\cdot)$.

As marginalization is key we introduce the following notation,

$$\sum_{\sim\{x_i\}} h(x_1, x_2, x_3) \triangleq \sum_{x_2 \in A_2} \sum_{x_3 \in A_3} h(x_1, x_2, x_3)$$

known as the "summary" operator.

Example:

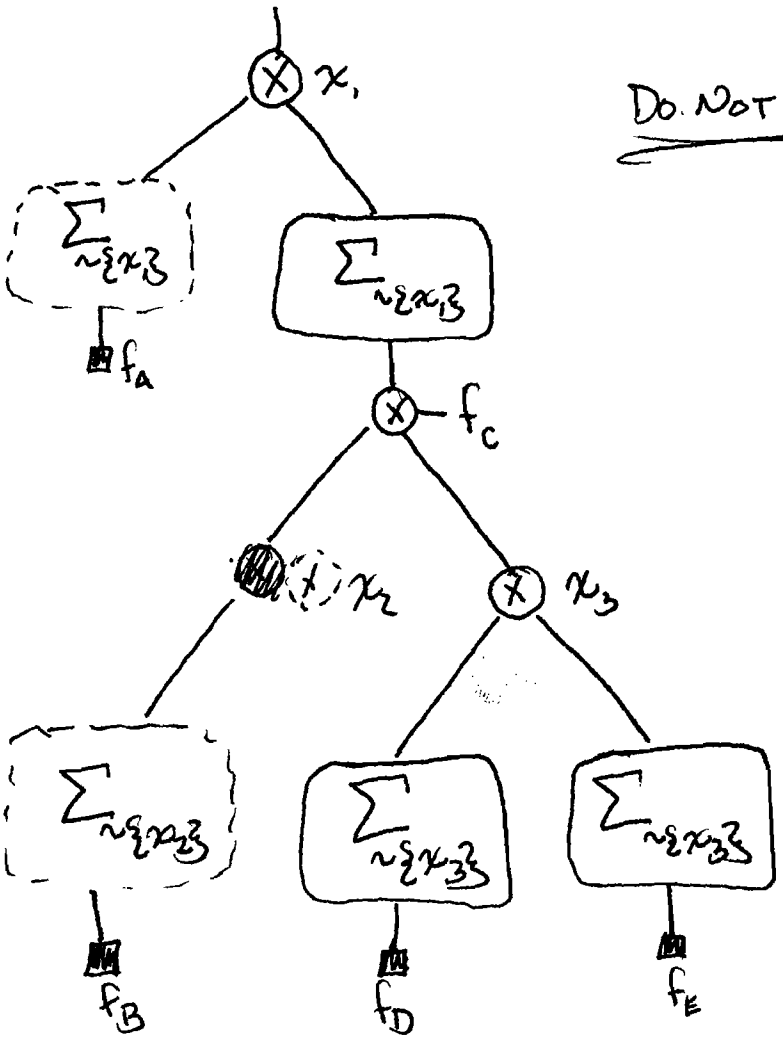
We can compute the marginal $g_1(x_1)$ as,

$$g_1(x_1) = f_A(x_1)\Big(\sum_{x_2} f_B(x_2)\Big(\sum_{x_3} f_C(x_1, x_2, x_3) * \cdots$$
$$* \Big(\sum_{x_4} f_D(x_3, x_4)\Big)\Big(\sum_{x_5} f_E(x_3, x_5)\Big)\Big)\Big)$$

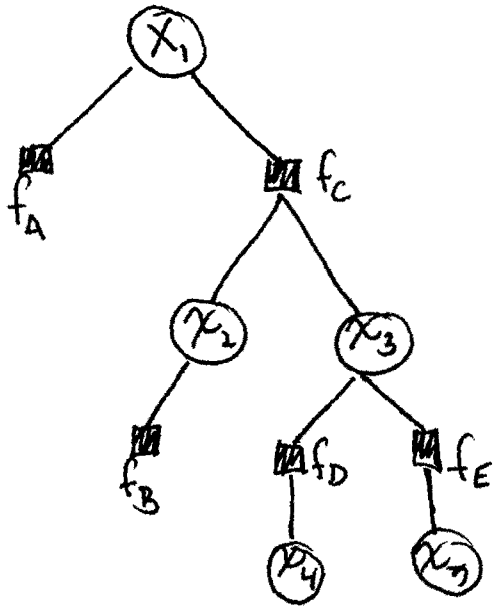
(1 for summary notation)

$$g_1(x_1) = f_A(x_1) * \sum_{\sim\{x_1\}} \Big(f_B(x_2)\, f_C(x_1, x_2, x_3) * \cdots * \Big(\sum_{\sim\{x_3\}} f_D(x_3, x_4) \Big) \Big(\sum_{\sim\{x_3\}} f_E(x_3, x_5) \Big) \Big)$$

We can represent this expression as a tree aptly named an expression tree.



... described to convert this to
the equivalent factor graph)



This mapping from expression tree → factor graph is distinct and covered in the paper.

So we now have the mapping

~~expre~~ marginal expression ⟺ expression tree ⟺ factor graph

NOTE: The mapping factor graph ⇒ expression tree is in Appendix A.

From these mappings we infer that the factor graph not only ~~shows~~ models explicit factorization but also explains the algorithm for computing marginal functions.

We can think of the single-i sum-product algorithm as, ~~"executing" the ax.~~

1) build expression tree rooted at x_i
2) "execute" expression tree bottom-up
3) marginal $g_i(x_i)$ is product of all incoming messages at root node i

The "message" passed on an edge is simply the result of computation from the subtree, or in other words a "summary" of local functions in subtree.

Computing All Marginals (Sum-Prod.)

IDEA:
We could simply repeat the single-i algorithm for all variables. This wastes computation since many messages are computed multiple times.

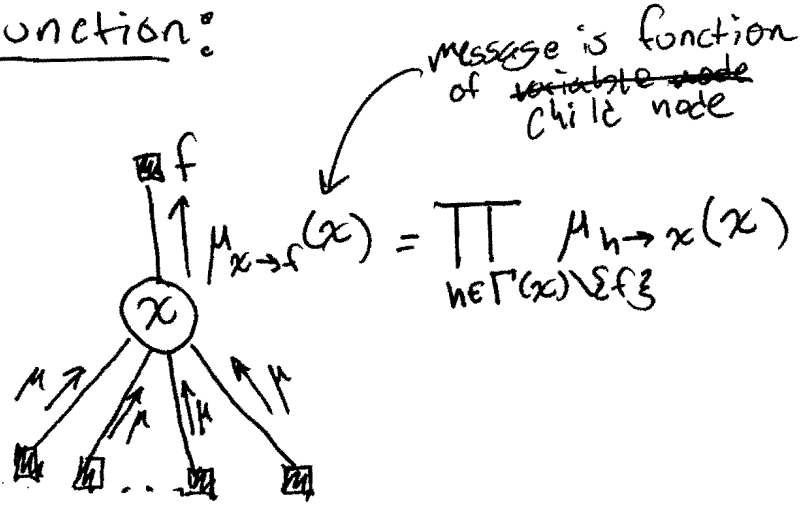
Instead we do the following updates beginning at leaf nodes,

LEAF: Pass local function, or 1 if variable

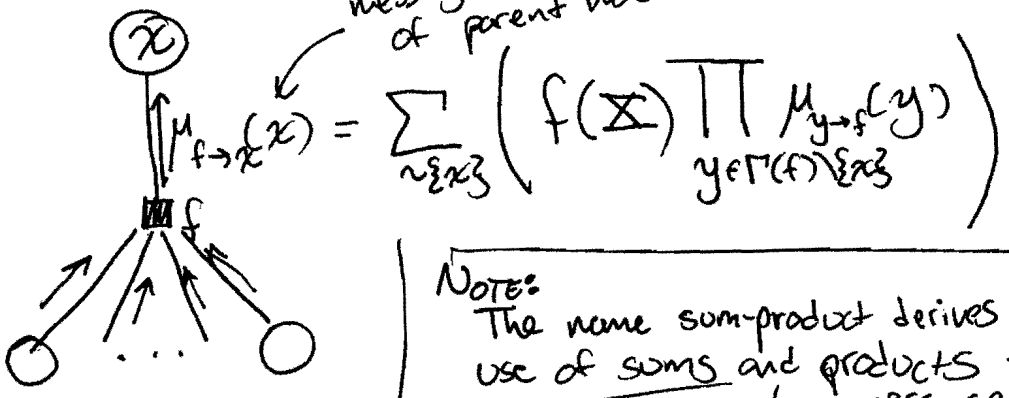
INTERNAL: ~~Perform sum-prod update rules~~ Wait until ~~all~~ messages arrive on all-but-one edge, then perform update rules

Sum-product update rules fall in two categories,

Variable-to-function:

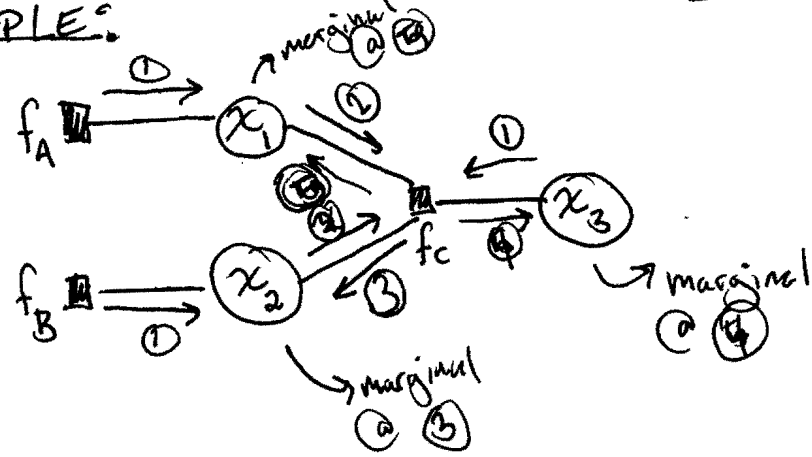


function-to-variable



NOTE:
The name sum-product derives from use of sums and products to comp. messages

EXAMPLE:



STEP 1:

$\mu_{f_A \to x_1}(x_1) = f_A(x_1)$

$\mu_{f_B \to x_2}(x_2) = f_B(x_2)$

$\mu_{x_3 \to f_c}(x_3) = 1$

STEP 2:

$\mu_{x_1 \to f_c}(x_1) = f_A(x_1)$

~~$\mu_{f_c}(x_2) = f_B(x_2)$~~

$\mu_{x_2 \to f_c}(x_2) = f_B(x_2)$

STEP 3:

$$\mu_{f_C \to x_2}(x_2) = \sum_{\sim\{x_2\}} f_C(x_1, x_2, x_3)\, \mu_{x_3 \to f_C}(x_3)\, \mu_{x_1 \to f_C}(x_1)$$

$$g_2(x_2) = \mu_{f_B \to x_2}(x_2)\, \mu_{f_C \to x_2}(x_2)$$

STEP 4:

$$\mu_{f_C \to x_3}(x_3) = \sum_{\sim\{x_3\}} f_C(x_1, x_2, x_3)\, \mu_{x_2 \to f_C}(x_2)\, \mu_{x_1 \to f_C}(x_1)$$

$$g_3(x_3) = \mu_{f_C \to x_3}(x_3)$$

STEP 5:

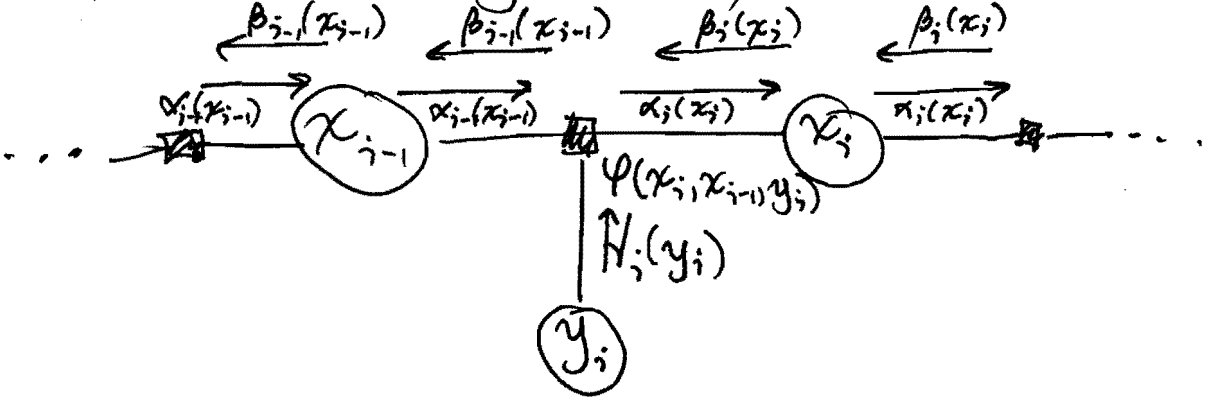
$$\mu_{f_C \to x_1}(x_1) = \sum_{\sim\{x_1\}} f_C(x_1, x_2, x_3)\, \mu_{x_3 \to f_C}(x_3)\, \mu_{x_2 \to f_C}(x_2)$$

$$g_1(x_1) = \mu_{f_C \to x_1}(x_1)$$

RELATED ALGORITHMS:

Forward/Backward Algorithm

Sum-product on an HMM yields the well-known forward/backward algorithm,



In a probabilistic setting the factors are typically defined as,

$$\psi_i(x_{i-1}, x_i, y_i) \propto p(x_i|x_{i-1})\, p(y_i|x_i)$$

Applying the sum-product update rules we get the following ~~definition of the~~ message updates,

$$\alpha_i(x_i) = \sum_{\sim\{x_i\}} \psi_i(x_{i-1}, x_i, y_i)\, \alpha_{i-1}(x_{i-1})\, \nu_i(y_i)$$

$$\beta_i(x_i) = \sum_{\sim\{x_i\}} \psi_{i+1}(x_i, x_{i+1}, y_{i+1})\, \beta_{i+1}(x_{i+1})\, \nu_{i+1}(y_{i+1})$$

The messages $\alpha(\cdot)$ and $\beta(\cdot)$ have a well-defined probabilistic interpretation,

$$\alpha_i(x_i) \propto p(y_1, \cdots, y_i, x_i) = p(x_i)\, p(y_1, \cdots, y_i|x_i)$$

$$\beta_i(x_i) \propto p(y_{i+1}, \cdots, y_N|x_i)$$

So the marginal (posterior) probability is,

$$p(x_i|y_1, \cdots, y_N) \propto \alpha_i(x_i)\, \beta_i(x_i)$$

Other Related Algorithms

- Replacing the summation w/ maximization yields max-product algorithm
- max-product on HMM yields Viterbi algorithm
- Sum-product on LDS yields Kalman filter
- Modifying update rules so all "computation" is at variable nodes yield Belief Prop.