Trees review

Definition A *tree* is a graph in there is one, and only one, path between any pair of nodes.

- Trees do not have loops
- Moralization of a directed tree results in an undirected tree

* Exact Inference is possible on trees using an algorithm much like that presented for chains. Sum-product or B. P.

* Graphs with loops are more complicated - loopy B.P.

Before sum-product:

Factor Graphs

- Directed & undirected graphs express a global function as the product of factors over subsets of those vars
  - Factors make explicit nodes for the factors in the product (in addition to the variables nodes)

Joint can be written as

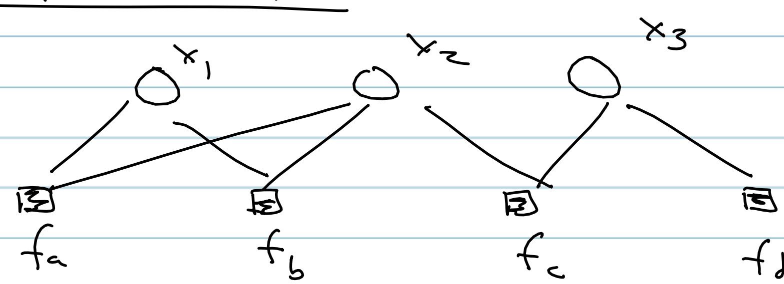
$$p(\vec{x}) = \prod_s f_s(\vec{x}_s)$$

where $\vec{x}_s$ is a subset of the variables

Individual var's are x_i, x_i may be complicated ie. matrix, vector, etc.

Where did the normalizing const. go?
- factor over an empty set of vars.

Factor graph example



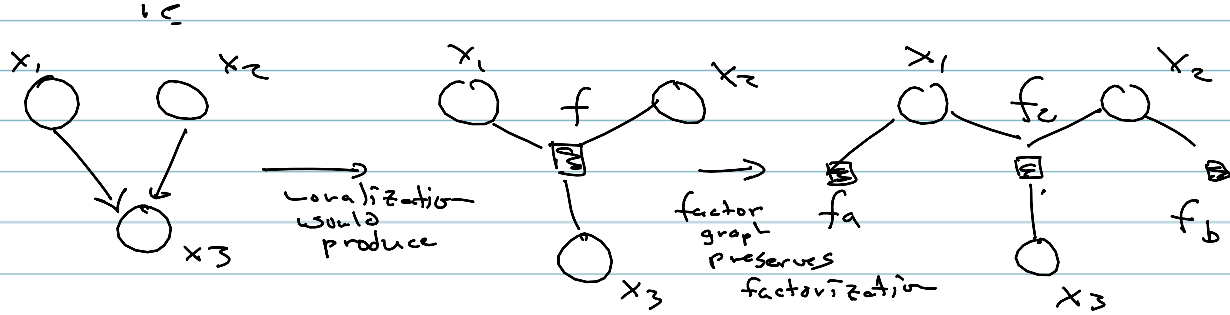
$$p(\vec{x}) = f_a(x_1, x_2)\, f_b(x_1, x_2)\, f_c(x_2, x_3)\, f_d(x_3)$$

A factor graph consists of

1 circle for each variable
1 square for each factor

A factor graph is a more explicit characterization of the factorization of the joint distribution, keeping separate terms that might be lumped together in an undirected g.m. for instance

ie



$p(\vec{x}) = p(x_1)\, p(x_2)\, p(x_3 \mid x_1, x_2)$

$f = p(x_1) p(x_2) p(x_3 \mid x_1, x_2)$

- can be reasons to combine factors in this way

$f_a = p(x_1)$
$f_b = p(x_2)$
$f_c = p(x_1, x_2, x_3)$

The Sum-product algorithm (Belief prop.)

Powerful algorithm for inference on tree-structured factor graphs.

Assumption all nodes discrete (could be linear-Gaussian however)

Recipe

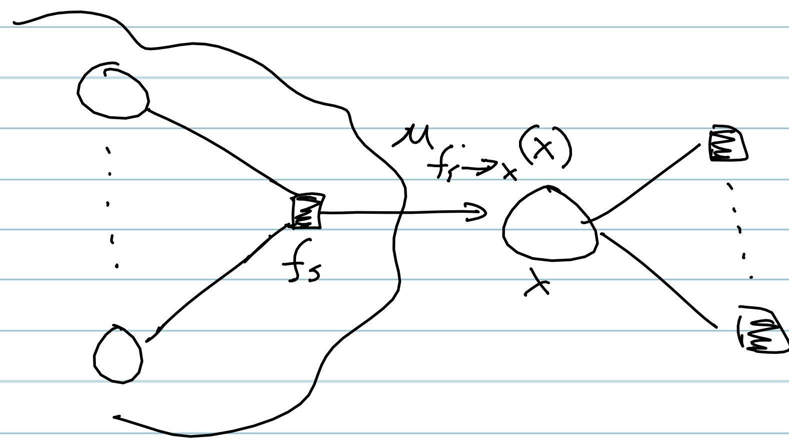
- Convert graph to factor graph.
- Pass messages
  - two kinds of messages in sum-prod. algorithm

Start as before with finding a single marginal dist:

$$p(x) = \sum_{\vec{x} \setminus x} p(\vec{x}) \qquad \vec{x} \setminus x \text{ is set w/o } x$$

Considering:

$$F_s(x, X_s)$$



One can directly see that

$$p(\vec{x}) = \prod_{s \in ne(x)} F_s(x, X_s)$$

where $ne(x)$ are the factor node neighbors of x and X_s denotes the set of all vars in the subtree connected to x through f_s.

$$F_s(x, X_s) = f_s(x, x_1 \ldots x_M)\, G(x_1, X_{s1}) \cdots G_M(x_M, X_{sM})$$

is the prod of all factors in group associated w/ factor f_s

x centric computation

$$p(x) = \sum_{\vec{X} \setminus x} p(\vec{X})$$

$$= \sum_{\vec{X} \setminus x} \prod_{s \in ne(x)} F_s(x, X_s)$$

$$= \prod_{s \in ne(x)} \sum_{X_s} F_s(x, X_s)$$

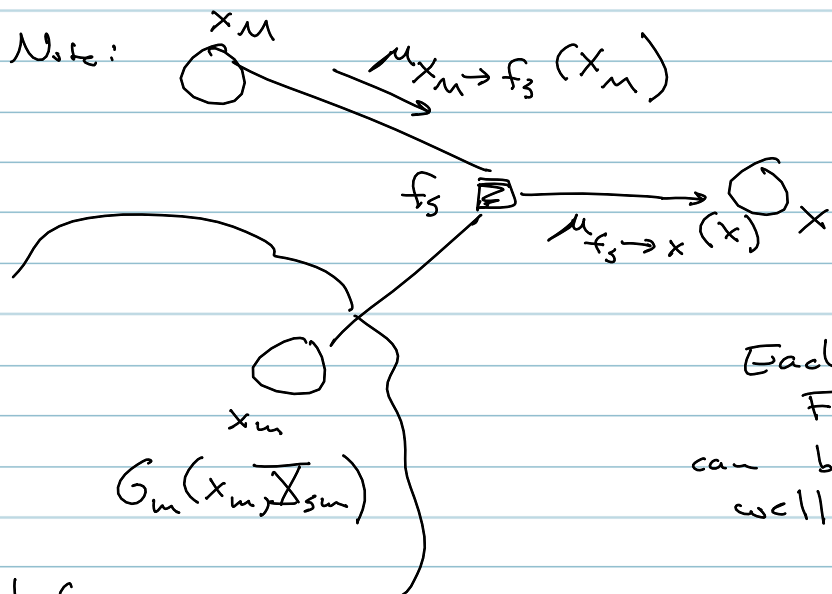
define $\mu_{f_s \to x}(x) = \sum_{X_s} F_s(x, X_s)$

$$= \prod_{s \in ne(x)} \mu_{f_s \to x}(x)$$

call these messages and prod's. of messages

How do we evaluate these messages, they are a big exponential sum?

Note:



Each factor $F_s(x, X_s)$ can be factorized as well!

As before

$$F_s(x, X_s) = f_s(x, x_1 \dots x_M)\, G_1(x_1, X_{s1}) \cdots G_M(x_M, X_{sM})$$

We exploit this factorization to compute

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

$$= \sum_{X_s} f_s(x, x_1, \ldots, x_M)\, G_1(x_1, X_{s1}) \cdots G_M(x_M, X_{sM})$$

$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right]$$

$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

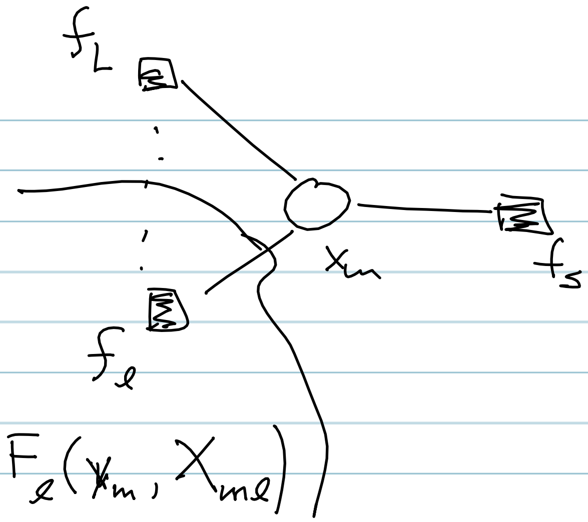
$m \in ne(f_s) \setminus x$ all variables in factor f_s besides x

Implicitly defined a new message

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm})$$

which is a message from a <u>var</u> node to a <u>factor</u> node.

How do we compute $\mu_{x_m \to f_s}(x_m)$?



From this we can see that $G_m(x_m, X_{sm})$ can be written as another product of factors

$$G_m(x_m, X_{sm}) = \prod_{\ell \in ne(x_m) \setminus f_s} F_\ell(x_m, X_{m\ell})$$

So we can write

$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} \prod_{\ell \in ne(x_m) \setminus f_s} F_\ell(x_m, X_{m\ell})$$

$$= \prod_{\ell \in ne(x_m) \setminus f_s} \left[\sum_{X_{m\ell}} F_\ell(x_m, X_{m\ell}) \right]$$

$$= \prod_{\ell \in ne(x_m) \setminus f_s} \mu_{f_\ell \to x_m}(x_m)$$

- Collecting up bits yields message passing algorithm

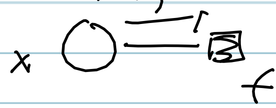
Message Types

$\blacksquare \to \bigcirc$ $$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

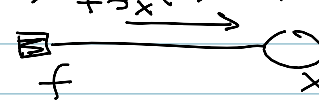
$\bigcirc \to \blacksquare$ $$\mu_{x_m \to f_s}(x_m) = \prod_{\ell \in ne(x_m) \setminus f_s} \mu_{f_\ell \to x_m}(x_m)$$

To start, messages can be sent from leaf nodes

$\mu_{x \to f}(x) = 1$



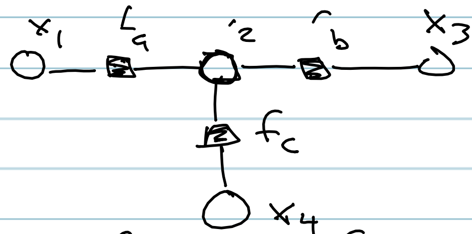
$\mu_{f \to x}(x) = f(x)$



Recap

- Message passing for evaluating marginal $p(x)$
- Pick x as root, pass messages from leaves to root.
- Once root receives all messages, pass all messages back to root
- Easy to see that this yields valid algorithm with enough messages always available j.i.t.

Example



$$\tilde{p}(\vec{x}) = f_a(x_1, x_2)\, f_b(x_2, x_3)\, f_c(x_2, x_4)$$

Designate x_3 as root and process messages from leaves

$$\mu_{x_1 \to f_a}(x_1) = 1$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2)$$

$$\mu_{x_4 \to f_c}(x_4) = 1$$

$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4)$$

$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\, \mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\, \mu_{x_2 \to f_b}$$

Now other way

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

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\, \mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2)\, \mu_{x_2 \to f_a}(x_2)$$

$$\mu_{x_2 \to f_c}(x_2) = \mu_{f_b \to x_2}(x_2)\, \mu_{f_a \to x_2}(x_2)$$

$$\mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4)\, \mu_{x_2 \to f_c}(x_2)$$

To evaluate $\tilde{p}(x_2)$ for instance we use

$$p(x) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x)$$

$$\begin{aligned}
\tilde{p}(x_2) &= \mu_{f_a \to x_2}(x_2)\, \mu_{f_b \to x_2}(x_2)\, \mu_{f_c \to x_2}(x_2) \\
&= \left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right]\left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_4} f_a(x_1, x_2)\, f_b(x_2, x_3)\, f_c(x_2, x_4)
\end{aligned}$$

Conditioning on observed values yields the same kind of indicator function approach.

- Discrete variable assumption not strictly necessary. Technique extends to continuous real val'ed vars, etc.

Loopy B.P.

What abouts graphs with loops?

Loopy BP is BP run on graph with loops until "convergence."