Trees Review

Definition: A tree is a graph in which 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
* Graphs with loops are more complicated - loopy BP.

Before sum-product:

Factor Graphs

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

Joint can be written as

\[ p(\mathbf{x}) = \prod_{s} f_s(\mathbf{x}_s) \]

where \( \mathbf{x}_s \) is a subset of the variables

Individual vars are \( x_i \), \( x_i \) may be complicated, i.e. matrix, vector, etc.

Where did the normalizing constant go?
- factor over an empty set of vars.
Factor graph example

\[
p(x) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)
\]

A factor graph consists of:
1. A circle for each variable
2. A 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 graph, for instance:

\[
p(x) = p(x_1) p(x_2) p(x_3 | x_1, x_2)
\]

- 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 linear)

**Recipe:**
- Convert graph to factor graph.
- Pass messages
  - Two kinds of messages in sum-product algorithm

Start as before with finding a single marginal dist:

\[
p(x) = \sum_{\tilde{x} \setminus x} p(\tilde{x})
\]

\(\tilde{x} \setminus x\) is set w/o \(x\)

Considering:

\[F_s(x, \bar{X}_s)\]

One can directly see that

\[
p(x) = \prod_{s \in \text{nc}(x)} F_s(x, \bar{X}_s)
\]

where \(\text{nc}(x)\) are the factor node neighbors of \(x\) and \(X_s\) denotes the set of all
variables in the subtree connected to \(x\) through factors.

\[F_s(x, \bar{X}_s) = f_s(x, x, \ldots, x_m)G(x, \bar{X}_{s3}) \ldots G_m(x, \bar{X}_{sm})\]

is the prod of all factors in group associated with \(f_s\).
\( P(x) = \sum_{x \in \mathcal{X}} P(x) \)

\[ \sum_{x \in \mathcal{X}} \prod_{s \in \text{ne}(x)} F_s(x, \overline{x}_s) \]

\[ \prod_{s \in \text{ne}(x)} \sum_{\overline{x}_s} F_s(x, \overline{x}_s) \]

\[ \text{define } M_{f_s \rightarrow x}(x) = \sum_{\overline{x}_s} F_s(x, \overline{x}_s) \]

\[ \delta = \prod_{s \in \text{ne}(x)} M_{f_s \rightarrow x}(x) \]

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

Note: \( M_{f_s \rightarrow x}(x) \)

Each factor \( F_s(x, \overline{x}_s) \) can be factorized as well!

As before:

\[ F_s(x, \overline{x}_s) = f_s(x_1, x_2, \ldots, x_n) G_1(x_1, \overline{x}_{s_1}) \cdots G_n(x_n, \overline{x}_{s_n}) \]
We exploit this factorization to compute

\[ \mu_{f_3 \rightarrow x}(x) \equiv \sum_{X_3} \mathcal{F}_3(x, X_3) \]

\[ = \sum_{X_3} f_3(x, x_1, \ldots, x_m) \prod_{i \in \text{ne}(f_3) \setminus x} \sum_{X_{5i}} g_i(x_i, X_{5i}) \cdots \sum_{X_{5m}} g_m(x_m, X_{5m}) \]

\[ = \sum_{x_1} \cdots \sum_{x_m} f_3(x, x_1, \ldots, x_m) \prod_{i \in \text{ne}(f_3) \setminus x} \mu_{x_i \rightarrow f_3}(x_i) \]

\[ m \in \text{ne}(f_3) \setminus x \quad \text{all variables in factor } f_3 \quad \text{besides } x \]

Implicitly defined a new message

\[ \mu_{x_m \rightarrow f_3}(x_m) \equiv \sum_{X_{5m}} g_m(x_m, X_{5m}) \]

which is a message from a variable node to a factor node.

How do we compute \( \mu_{x_m \rightarrow f_3}(x_m) \)?
From this we can see that $G_m(x_m, \overline{x}_m)$ can be written as another product of factors:

$$G_m(x_m, \overline{x}_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} F_{\xi_l}(x_m, \overline{x}_m)$$

So we can write:

$$M_{x_m \rightarrow f_s(x_m)} = \sum_{\overline{x}_m} \prod_{l \in \text{ne}(x_m) \setminus f_s} F_{\xi_l}(x_m, \overline{x}_m)$$

$$= \prod_{l \in \text{ne}(x_m) \setminus f_s} \left[ \sum_{\overline{x}_m} F_{\xi_l}(x_m, \overline{x}_m) \right]$$

$$= \prod_{l \in \text{ne}(x_m) \setminus f_s} M_{f_s \rightarrow x_m}(x_m)$$

Collecting optimization yields message passing algorithm.
Message Types

\[ M_{f_s \rightarrow x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} M_{x_m \rightarrow f_s}(x_m) \]

To start, messages can be sent from leaf nodes

\[ M_{x \rightarrow f}(x) = 2 \quad M_{f \rightarrow x}(x) = f(x) \]

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.
Example 1:

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

Designate \( x_3 \) as root and process messages from leaves:

\[
\begin{align*}
\mu_{x_1} &\rightarrow f_a(x_1) = 1 \\
\mu_{f_a} &\rightarrow x_2(x_2) = \sum_{x_1} f_a(x_1, x_2) \\
\mu_{x_4} &\rightarrow f_c(x_4) = 1 \\
\mu_{f_c} &\rightarrow x_2(x_2) = \sum_{x_4} f_c(x_2, x_4) \\
\mu_{x_2} &\rightarrow f_b(x_2) = \mu_{f_a} \rightarrow x_2(x_2) \mu_{f_c} \rightarrow x_2(x_2) \\
\mu_{f_b} &\rightarrow x_3(x_3) = \sum_{x_2} f_b(x_2, x_3) \mu_{x_2} \rightarrow f_b
\end{align*}
\]

Now another way:

\[
\begin{align*}
\mu_{x_3} &\rightarrow f_b(x_3) = 1 \\
\mu_{f_b} &\rightarrow x_2(x_2) = \sum_{x_3} f_b(x_2, x_3) \\
\mu_{x_2} &\rightarrow f_a(x_2) = \mu_{f_b} \rightarrow x_2(x_2) \mu_{f_c} \rightarrow x_2(x_2) \\
\mu_{f_a} &\rightarrow x_1(x_1) = \sum_{x_2} f_a(x_1, x_2) \mu_{x_2} \rightarrow f_a(x_2) \\
\mu_{x_2} &\rightarrow f_c(x_2) = \mu_{f_b} \rightarrow x_2 \mu_{f_a} \rightarrow x_2(x_2) \\
\mu_{f_c} &\rightarrow x_4(x_4) = \sum_{x_2} f_c(x_2, x_4) \mu_{x_2} \rightarrow f_c(x_2)
\end{align*}
\]
To evaluate $\mathcal{P}(x_2)$ for instance we use

$$p(x) = \prod_{s \in \text{ne(x)}} m_{fs \rightarrow x}(x)$$

$$\mathcal{P}(x_2) = m_{f_a \rightarrow x_2}(x_2) \cdot m_{f_b \rightarrow x_2}(x_2) \cdot m_{f_c \rightarrow x_2}(x_2)$$

$$= \left[ \sum_{x_1} f_a(x_1, x_2) \right] \left[ \sum_{x_3} f_b(x_2, x_3) \right] \left[ \sum_{x_4} f_c(x_2, x_4) \right]$$

$$= \sum_{x_1} \sum_{x_2} \sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

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

- Discrete variable assumption not strictly necessary. Technique extends to continuous real valued var, etc.

Loopy B. P.

What about graphs with loops?

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