Factor Graphs and message passing

Carl Edward Rasmussen

October 28th, 2016

Key concepts

- Factor graphs are a class of graphical model
- A factor graph represents the product structure of a function, and contains factor nodes and variable nodes
- We can compute marginals and conditionals efficiently by passing messages on the factor graph, this is called the sum-product algorithm (a.k.a. belief propagation or factor-graph propagation)
- We can apply this to the True Skill graph
- But certain messages need to be approximated
- One approximation method based on moment matching is called Expectation Propagation (EP)

Factor Graphs

Factor graphs are a type of *probabilistic graphical model* (others are directed graphs, a.k.a. Bayesian networks, and undirected graphs, a.k.a. Markov networks).

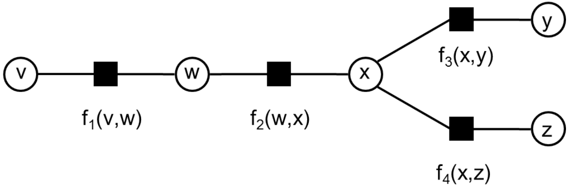
Factor graphs allow to represent the product structure of a function.

Example: consider the factorising probability distribution:

$$p(v, w, x, y, z) \;=\; f_1(v, w) f_2(w, x) f_3(x, y) f_4(x, z)$$

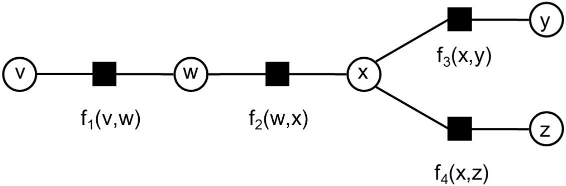
A factor graph is a bipartite graph with two types of nodes:

- Factor node: ■ Variable node: ○
- Edges represent the dependency of factors on variables.



Factor Graphs

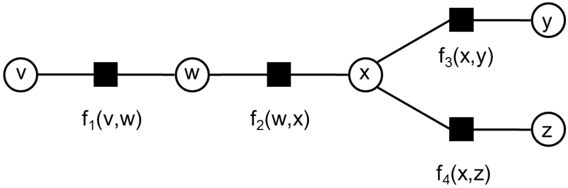
$$p(v, w, x, y, z) \; = \; f_1(v, w) f_2(w, x) f_3(x, y) f_4(x, z)$$



- What are the marginal distributions of the individual variables?
- What is $p(w)$?
- How do we compute conditional distributions, e.g. $p(w|y)$?

For now, we will focus on *tree-structured* factor graphs.

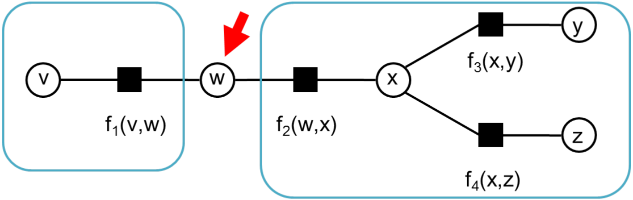
Factor trees: separation (1)



$$p(w) = \sum_{v}\sum_{x}\sum_{y}\sum_{z} f_1(v,w)f_2(w,x)f_3(x,y)f_4(x,z)$$

- If w, v, x, y and z take K values each, we have $\approx 3K^4$ products and $\approx K^4$ sums, for each value of w, i.e. total $\mathcal{O}(K^5)$.
- Multiplication is distributive: $ca + cb = c(a + b)$.
  The right hand side is more efficient!

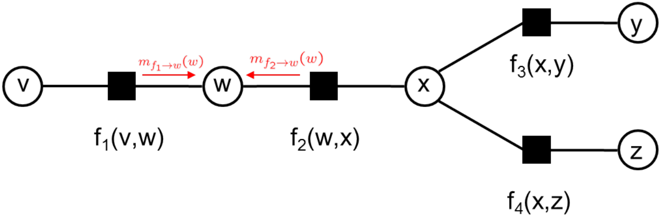
Factor trees: separation (2)



$$
\begin{aligned}
p(w) &= \sum_v \sum_x \sum_y \sum_z f_1(v,w) f_2(w,x) f_3(x,y) f_4(x,z) \\
&= \Big[\sum_v f_1(v,w) \Big] \cdot \Big[\sum_x \sum_y \sum_z f_2(w,x) f_3(x,y) f_4(x,z) \Big]
\end{aligned}
$$

- In a tree, each node separates the graph into disjoint parts.
- Grouping terms, we go from sums of products to products of sums.
- The complexity is now $\mathcal{O}(K^4)$.

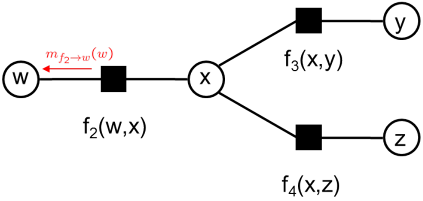
Factor trees: separation (3)



$$p(w) = \underbrace{\Big[\sum_{v} f_1(v, w)\Big]}_{m_{f_1 \to w}(w)} \cdot \underbrace{\Big[\sum_{x}\sum_{y}\sum_{z} f_2(w, x) f_3(x, y) f_4(x, z)\Big]}_{m_{f_2 \to w}(w)}$$

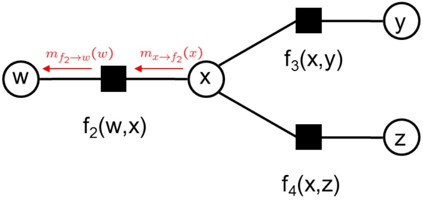
- Sums of products becomes products of sums of all messages from neighbouring factors to variable.

Messages: from factors to variables (1)



$$\color{red}{m_{f_2\to w}(w)} = \sum_x \sum_y \sum_z f_2(w,x) f_3(x,y) f_4(x,z)$$

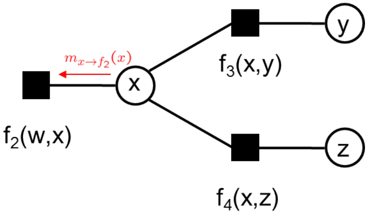
Messages: from factors to variables (2)



$$\begin{aligned} m_{f_2\to w}(w) &= \sum_x \sum_y \sum_z f_2(w,x) f_3(x,y) f_4(x,z) \\ &= \sum_x f_2(w,x) \cdot \underbrace{\Big[\sum_y \sum_z f_3(x,y) f_4(x,z)\Big]}_{m_{x\to f_2}(x)} \end{aligned}$$

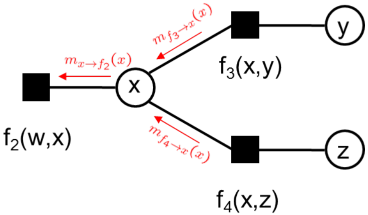
- Factors only need to sum out all their local variables.

Messages: from variables to factors (1)



$$m_{x \to f_2}(x) = \sum_y \sum_z f_3(x, y) f_4(x, z)$$

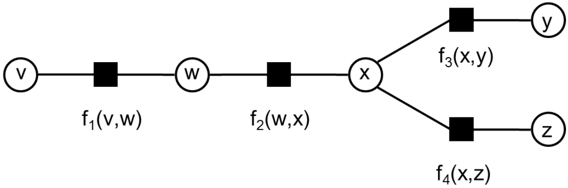
Messages: from variables to factors (2)



$$
\begin{aligned}
\color{red}{m_{x \to f_2}(x)} &= \sum_y \sum_z f_3(x,y) f_4(x,z) \\
&= \underbrace{\Big[\sum_y f_3(x,y)\Big]}_{\color{red}{m_{f_3 \to x}(x)}} \cdot \underbrace{\Big[\sum_z f_4(x,z)\Big]}_{\color{red}{m_{f_4 \to x}(x)}}
\end{aligned}
$$

- Variables pass on the product of all incoming messages.

Factor graph marginalisation: summary



$$p(w) = \sum_v \sum_x \sum_y \sum_z f_1(v,w) f_2(w,x) f_3(x,y) f_4(x,z)$$

$$= \underbrace{\Big[\sum_v f_1(v,w)\Big]}_{m_{f_1\to w}(w)} \cdot \underbrace{\Big[\sum_x f_2(w,x) \cdot \underbrace{\Big[\underbrace{\Big[\sum_y f_3(x,y)\Big]}_{m_{f_3\to x}(x)} \cdot \underbrace{\Big[\sum_z f_4(x,z)\Big]}_{m_{f_4\to x}(x)}\Big]}_{m_{x\to f_2}(x)}\Big]}_{m_{f_2\to w}(w)}$$

- The complexity is reduced from $\mathcal{O}(K^5)$ (naïve implementation) to $\mathcal{O}(K^2)$.

The sum-product algorithm

In summary, message passing involved three update equations:

- Marginals are the product of all incoming messages from neighbour factors

$$p(t) = \prod_{f \in F_t} m_{f \to t}(t)$$

- Messages from factors sum out all variables except the receiving one

$$m_{f \to t_1}(t_1) = \sum_{t_2} \sum_{t_3} \ldots \sum_{t_n} f(t_1, t_2, \ldots, t_n) \prod_{i \neq 1} m_{t_i \to f}(t_i)$$

- Messages from variables are the product of all incoming messages except the message from the receiving factor

$$m_{t \to f}(t) = \prod_{f_j \in F_t \setminus \{f\}} m_{f_j \to t}(t) = \frac{p(t)}{m_{f \to t}(t)}$$

Messages are results of partial computations. Computations are localised.