Statistical Machine Learning

Christian Walder

Machine Learning Research Group
CSIRO Data61

and

College of Engineering and Computer Science
The Australian National University

Canberra
Semester One, 2020.

(Many figures from C. M. Bishop, "Pattern Recognition and Machine Learning")

Outlines

Part XIX

Probabilistic Graphical Models 3

Factor Graphs

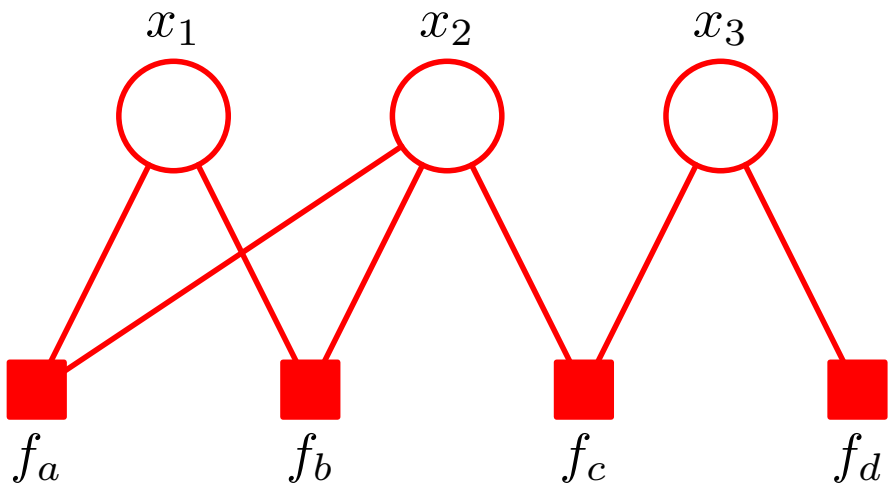
- Write $p(\mathbf{x})$ in the form of a product of factors

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

where $\mathbf{x}_s$ denotes a subset of variables.

- Example

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

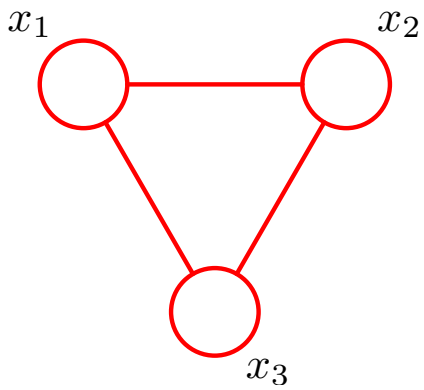


- More information than in MRF, because there $f_a(x_1, x_2) f_b(x_1, x_2)$ would be in one potential function.

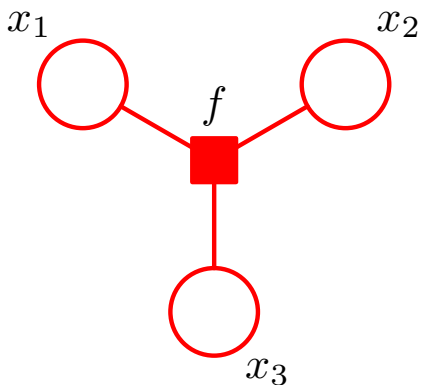
Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Factor Graphs - MRF example

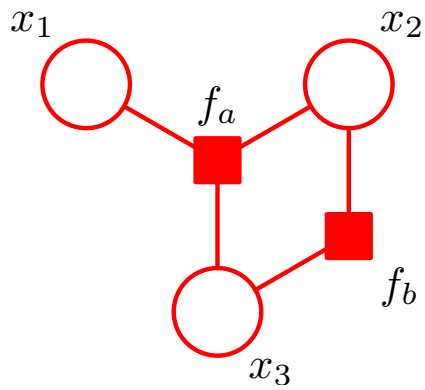
Example of factor graphs representing the same distribution



Undirected graph
single clique
potential
$\psi(x_1, x_2, x_3)$



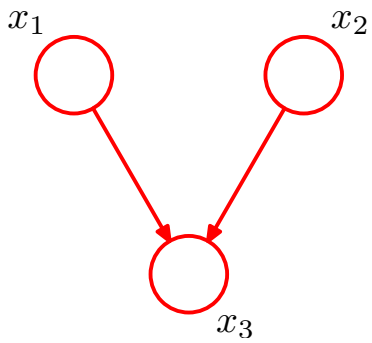
Factor graph
$f(x_1, x_2, x_3)$
$=$
$\psi(x_1, x_2, x_3)$



Factor graph
factors satisfy
$f_a(x_1, x_2, x_3) f_b(x_2, x_3)$
$=$
$\psi(x_1, x_2, x_3)$

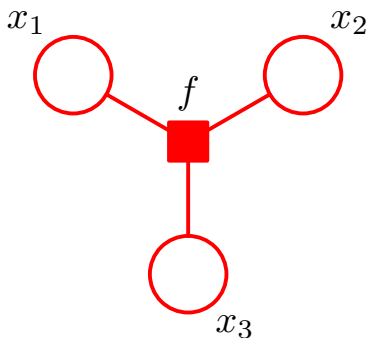
Factor Graphs - BN example

Example of factor graphs representing the same distribution



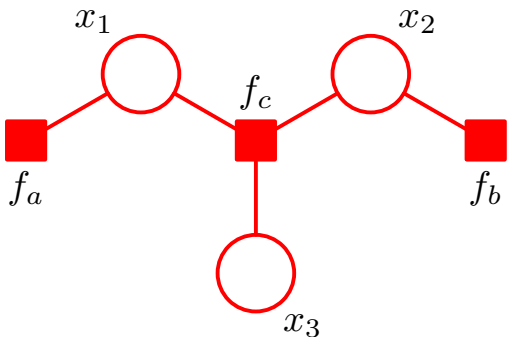
Directed graph

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



Factor graph

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



Factor graph factors satisfy

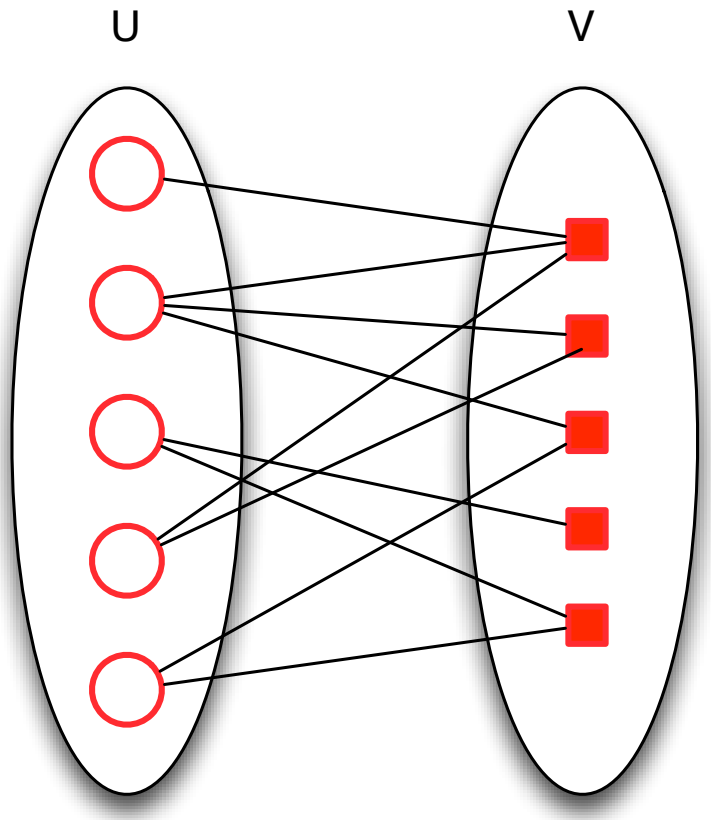
$$f_a(x_1) = p(x_1)$$
$$f_b(x_2) = p(x_2)$$
$$f_c(x_1, x_2, x_3) = p(x_3 \mid x_1, x_2)$$

Factor Graph - Bipartite Graph

- Factor Graphs are bipartite graphs.

Definition (Bipartite Graph)

A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.



Markov Random Field → Factor Graph

1. Create variable nodes for each node in the original graph
2. Create factor nodes for each maximal clique $\mathbf{x}_s$
3. Set the factors $f_s(\mathbf{x}_s)$ to the clique potentials

Note: There may be several different factor graphs corresponding to the same undirected graph.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Bayesian Network → Factor Graph

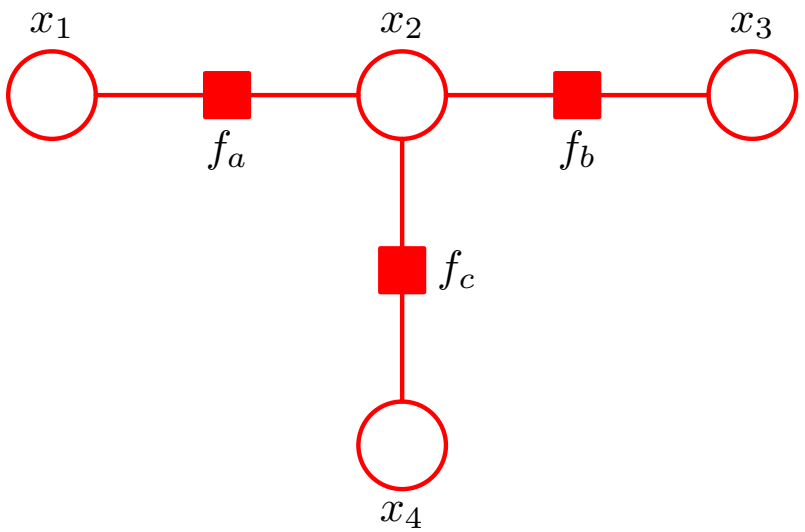
1. Create variable nodes for each node in the original graph.
2. Create factor nodes corresponding to the conditional distributions.
3. Add appropriate links.

Note: There may be several different factor graphs corresponding to the same directed graph.

The Sum-Product Algorithm - Motivation

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

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



Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

The Sum-Product Algorithm in a Nutshell

Kschischang & Frey, 2001.

The message sent from a node v on an edge e is the product of the local function at v (or the unit function if v is a variable node) with all messages received at v on edges *other* than e, summarized for the variable associated with e.

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

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

the marginals are then

$$p(x_m) = \prod_{l \in \mathrm{ne}(x_m)} \mu_{f_l \to x_m}(x_m)$$

The Sum-Product Algorithm - Example

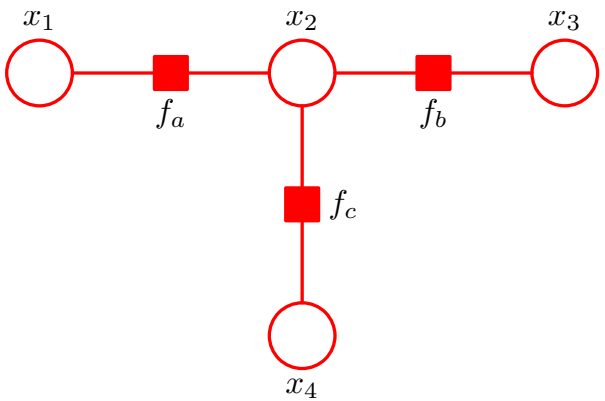
- Nodes send the following on each adjacent edge after receiving on other edges (leaves $\rightarrow$ arbitrary root $\rightarrow$ leaves)
  - factors nodes: marginalised own factor times product of incoming messages
  - variable nodes: product of incoming messages

$$\mu_{x_1 \to f_a}(x_1) = 1 \qquad \mu_{x_4 \to f_c}(x_4) = 1$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2) \qquad \mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4)$$

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

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



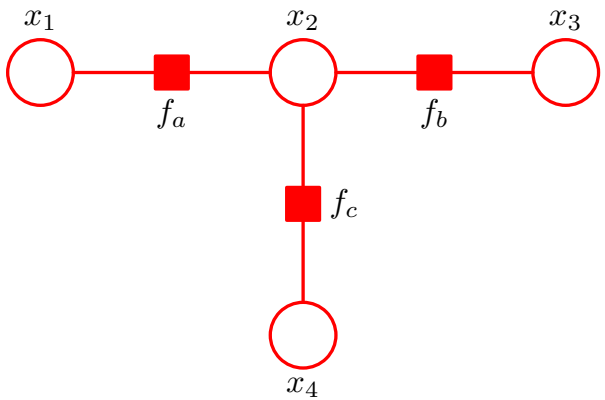
The Sum-Product Algorithm - Example

- Propagating back from root

$$\mu_{x_3 \to f_b}(x_3) = 1 \qquad \mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

- Marginals are products of messages at a variable node:

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



- Works for any tree structured factor graph.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

The Sum-Product Algorithm - Overview

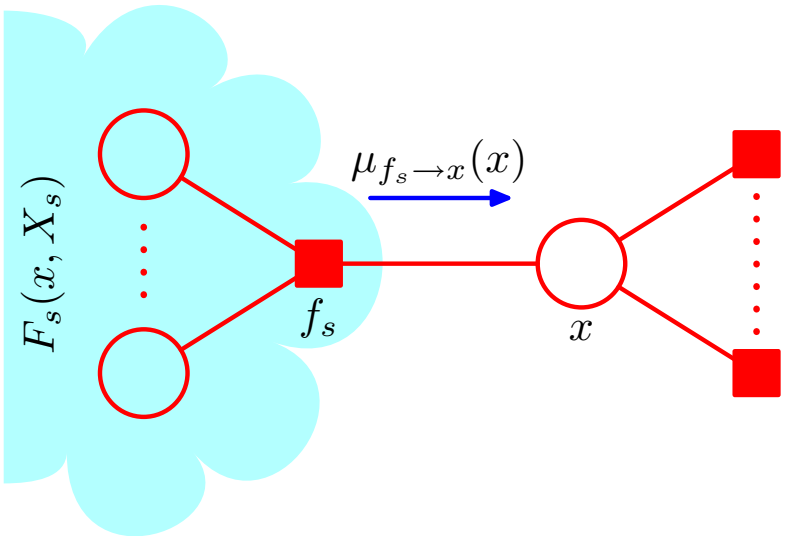
- Find some marginal $p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$
- Idea: interchange summations and products:
  - Push sums as far into products as possible
  - View partial sums as messages
  - Partial sums are coupled by factors
  - Deal with coupling by propagating messages
  - Messages are functions of some variable
  - Exploit the tree structure of the factor graph to divide and conquer
- The tree structure is crucial.
- Generally incorrect for loopy graphs.
- Approach is also known as
  - Belief propagation
  - Message passing
  - Sum product algorithm

The Sum-Product Algorithm - In Detail

- Partition the factors in the joint distribution into groups, with one group associated with each factor of the nodes that is a neighbour of the variable node x.

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

  - $\mathrm{ne}(x)$ is the set of factor nodes which are neighbours of x
  - X_s is the set of all variables in the sub-tree connected to the variable node x via the factor node f_s
  - $F_s(x, X_s)$ is the product of all the factors in the group associated with factor f_s.



Data61 | CSIRO
The Australian National University

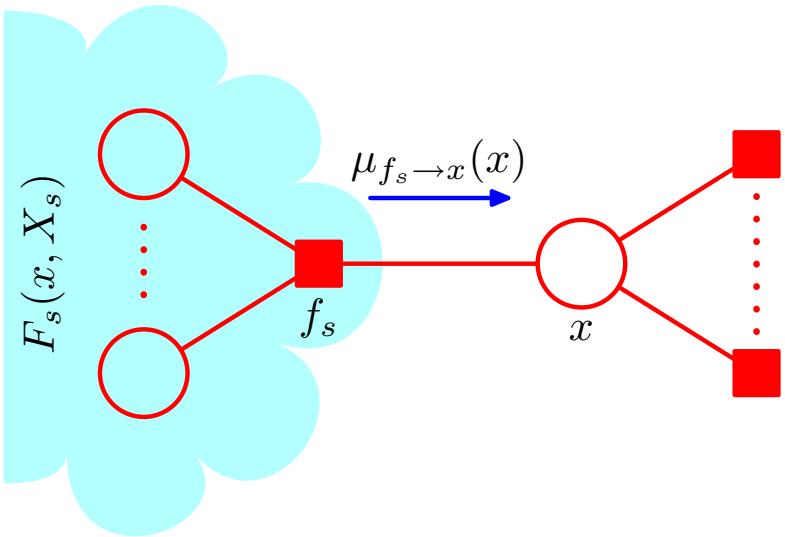
The Sum-Product Algorithm - In Detail

- Goal: Marginal distribution $p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$
- via joint distribution

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

- resulting in

$$p(x) = \sum_{\mathbf{x} \setminus x} \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) = \prod_{s \in \mathrm{ne}(x)} \underbrace{\sum_{X_s} F_s(x, X_s)}_{\doteq \mu_{f_s \to x}(x)}$$



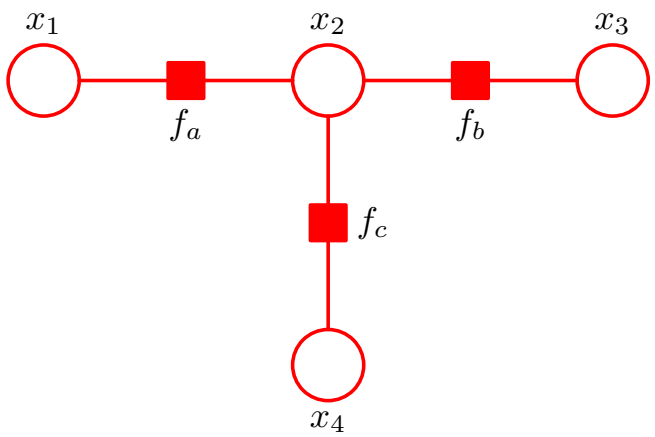
Example for Exchanging Sum and Product 1/2

- Goal: Marginal distribution $p(x_2) = \sum_{\mathbf{x} \setminus x_2} p(\mathbf{x})$
- via joint distribution

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

e.g.

$$p(x_1, x_2, x_3, x_4) = \prod_{s \in \mathrm{ne}(x_2)} F_s(x_2, X_s) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$



Example for Exchanging Sum and Product 2/2

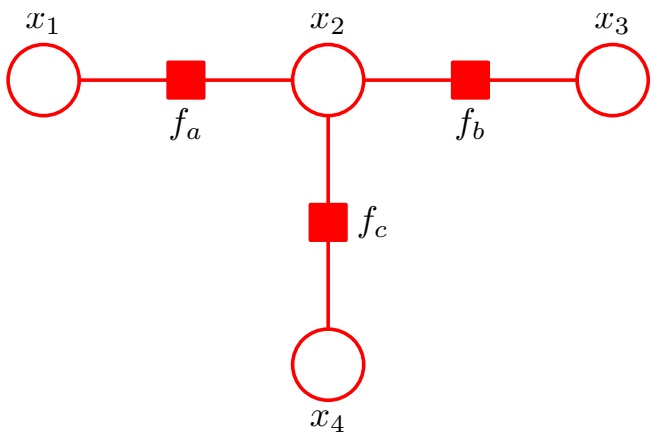
- Goal: Marginal distribution $p(x)$

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

e.g.

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

$$= \underbrace{\left(1 \cdot \sum_{x_1} f_a(x_1,x_2)\right)}_{\doteq \mu_{f_a\to x_2}(x_2)} \underbrace{\left(1 \cdot \sum_{x_3} f_b(x_2,x_3)\right)}_{\doteq \mu_{f_b\to x_2}(x_2)} \underbrace{\left(1 \cdot \sum_{x_4} f_c(x_2,x_4)\right)}_{\doteq \mu_{f_c\to x_2}(x_2)}$$

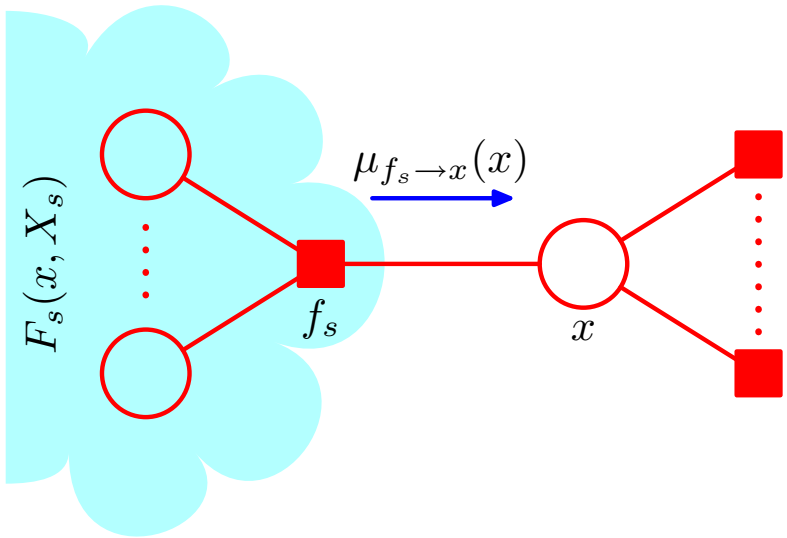


The Sum-Product Algorithm - In Detail (recap)

$$p(x) = \sum_{\mathbf{x}\setminus x} \prod_{s\in \mathrm{ne}(x)} F_s(x, X_s) = \prod_{s\in \mathrm{ne}(x)} \sum_{X_s} F_s(x, X_s)$$
$$= \prod_{s\in \mathrm{ne}(x)} \mu_{f_s\to x}(x)$$

messages are functions of the variable they are passed to:

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

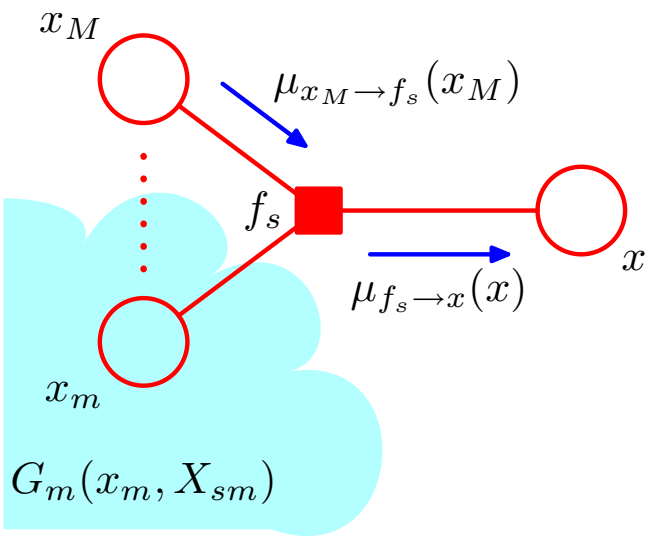


Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

How to evaluate the messages $\mu_{f_s \to x}(x)$?

- Each factor $F_s(x, X_s)$ consists of a subgraph and can therefore be written as

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



- $\{x, x_1, \ldots, x_M\}$ is the set of variables on which f_s depends
- X_{sm} are the variables in the subtree connected to f_s via x_m
- beware the variable notation (here and in the textbook)

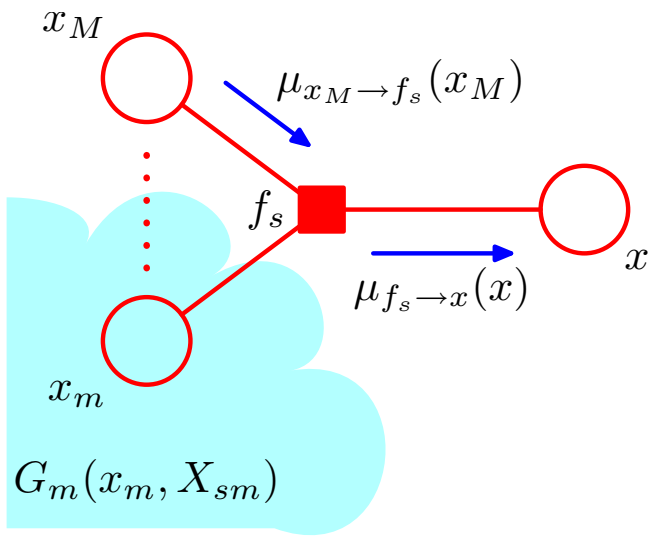
How to evaluate the messages $\mu_{f_s \to x}(x)$?

- Push the sums over X_{sm} into the product:

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

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

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



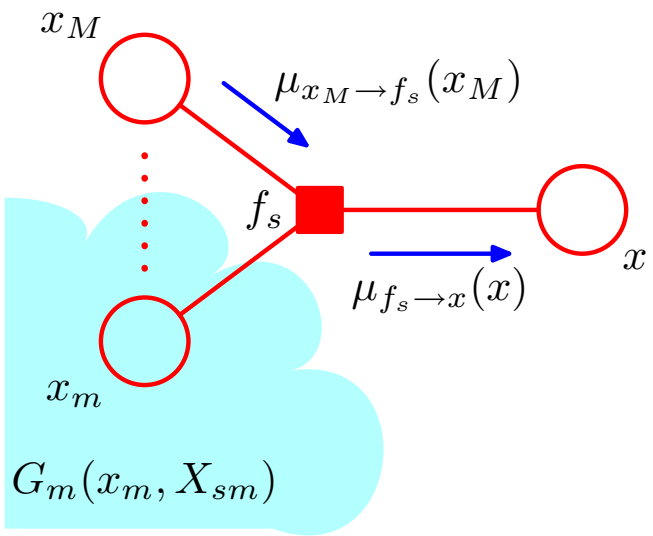
Two kind of messages

- factor nodes $\rightarrow$ variable nodes

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

- variable nodes $\rightarrow$ factor nodes

$$\mu_{x_m \rightarrow f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm})$$



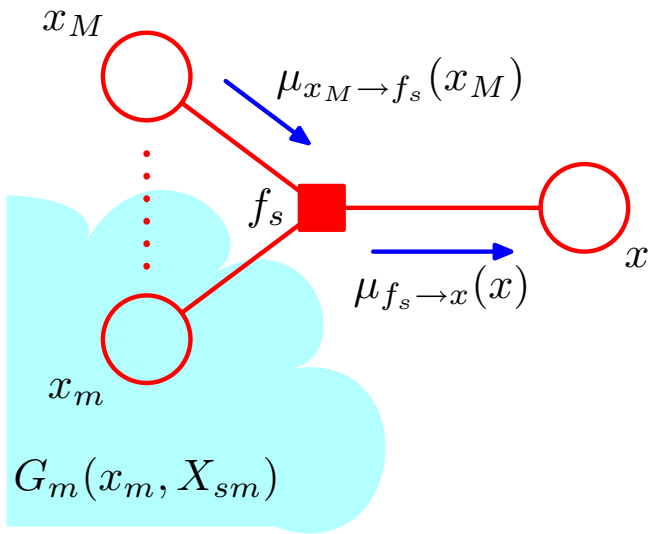
Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

factor nodes → variable nodes

- We already have a formula to calculate messages from factor nodes to variable nodes

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

- Take the product of all incoming messages, multiply with the factor associated with the node and marginalise over all variables associated with the incoming messages.



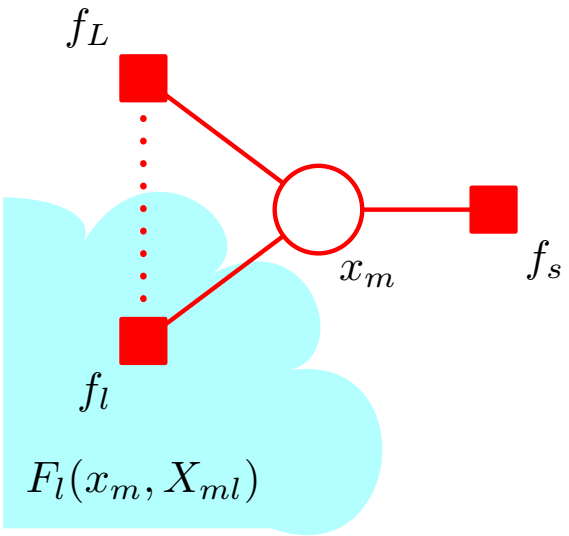
Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

variable nodes → factor nodes (1/2)

- $G_m(x_m, X_{sm})$ is a product of terms $F_l(x_m, X_{ml})$

$$G_m(x_m, X_{sm}) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

where the product is taken over all neighbours of node x_m except for node f_s.

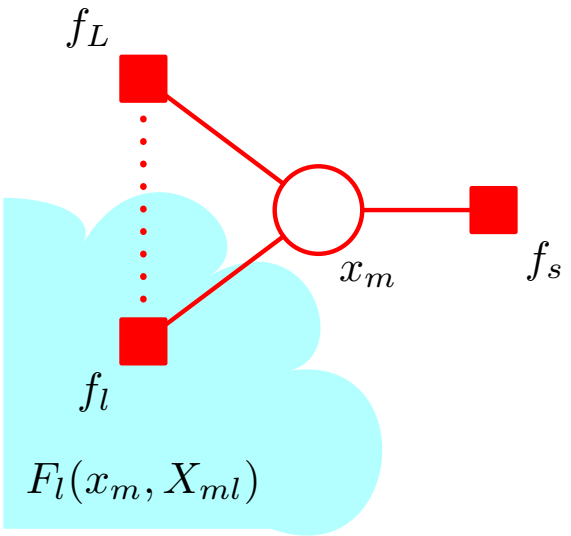


Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

variable nodes → factor nodes (2/2)

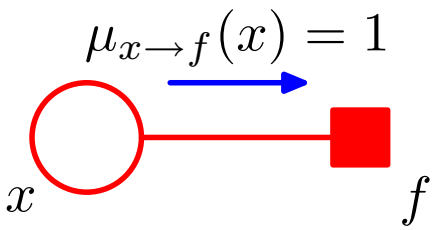
$$\begin{aligned}\mu_{x_m \to f_s}(x_m) &= \sum_{X_{sm}} G_m(x_m, X_{sm}) \\ &= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m, X_{ml}) \right] \\ &= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m).\end{aligned}$$

- Evaluate the message sent by a variable to an adjacent factor by taking the product of incoming messages.

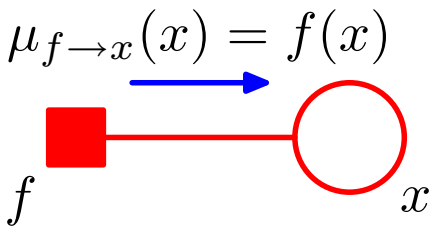


How to start? How to normalise?

- Consider node x as the root node of the factor graph.
- Start at the leaf nodes.
  - If the leaf node is a variable node then



  - If the leaf node is a factor node then



- These initialisation rules follow from the general formulae.
- Normalisation (if the factors are un-normalised):
  - Calculate $\tilde{p}(x)$ by message passing as before
  - Then $Z = \sum_x \tilde{p}(x)$ and $p = \frac{1}{Z}\tilde{p}$

Data61 | CSIRO
The Australian National University

Marginals for ALL variable nodes in the graph

- Brute force: run the above algorithm for each node.
- More efficient
  1. Arbitrarily choose one root in the graph.
  2. Propagate all messages from leaves to root.
  3. Propagate all messages from root to leaves.
  4. Local messages give marginals.
- Only doubles the number of computations.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Generalising Sum-Product: Distributive Property

- For the sum product algorithm, we used the property that multiplication is distributive over addition:

$$ab + ac = a(b + c)$$

- Abstract theory
  A set with two associative operations '$+$' and '$\times$' satisfying the distributive property is called a semi-ring.
- Generalized Distributive Law
  In principle, one can replace 'sum' and 'product' in the previous algorithm with other operations.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Generalising Sum-Product: Max-Product

- $ab + ac = a(b + c) \rightarrow \max(ab, ac) = a \max(b, c)$
- Replace '$+$' by $\max$, and '$\times$' by $\sum$
- Max Product algorithm: compute $\text{argmax}_x \, p(\mathbf{x})$
- For Hidden Markov Models, this is the Viterbi algorithm.

Ong & Walder & Webers
Data61 | CSIRO
The Australian National University

Similar Algorithms

- Junction Tree Algorithm: exact inference in general undirected graphs. (For directed graphs, first convert to moral graph.) Computational cost is determined by the number of variables in the largest clique of the graph. Grows exponentially with this number for discrete variables.
- Loopy Belief Propagation: try sum-product on graphs which are not tree-structured. Graph has cycles and therefore information flows several times through the graph. Initialise by assuming a unit message has been sent over each link in each direction. Convergence is not longer guaranteed in general.

Learning the Graph Structure

- Define a space of possible graph structures.
- Define a measure to score each of the graph structures.
- Bayesian viewpoint: Compute the posterior distribution over graph structures.
- If we have a prior $p(m)$ over graphs indexed by m, the posterior is given via Bayes' theorem as

$$p(m \mid \mathcal{D}) \propto p(m)\, p(\mathcal{D} \mid m)$$

  where $\mathcal{D}$ is the observed data set, and the model evidence $p(\mathcal{D} \mid m)$ provides the score for each model.
- Challenge 1: Evaluation of the model evidence involves marginalisation over the latent variables and is computationally very demanding.
- Challenge 2: The number of different graph structures grows exponentially with the number of nodes. Need to resort to heuristics to find good candidates.