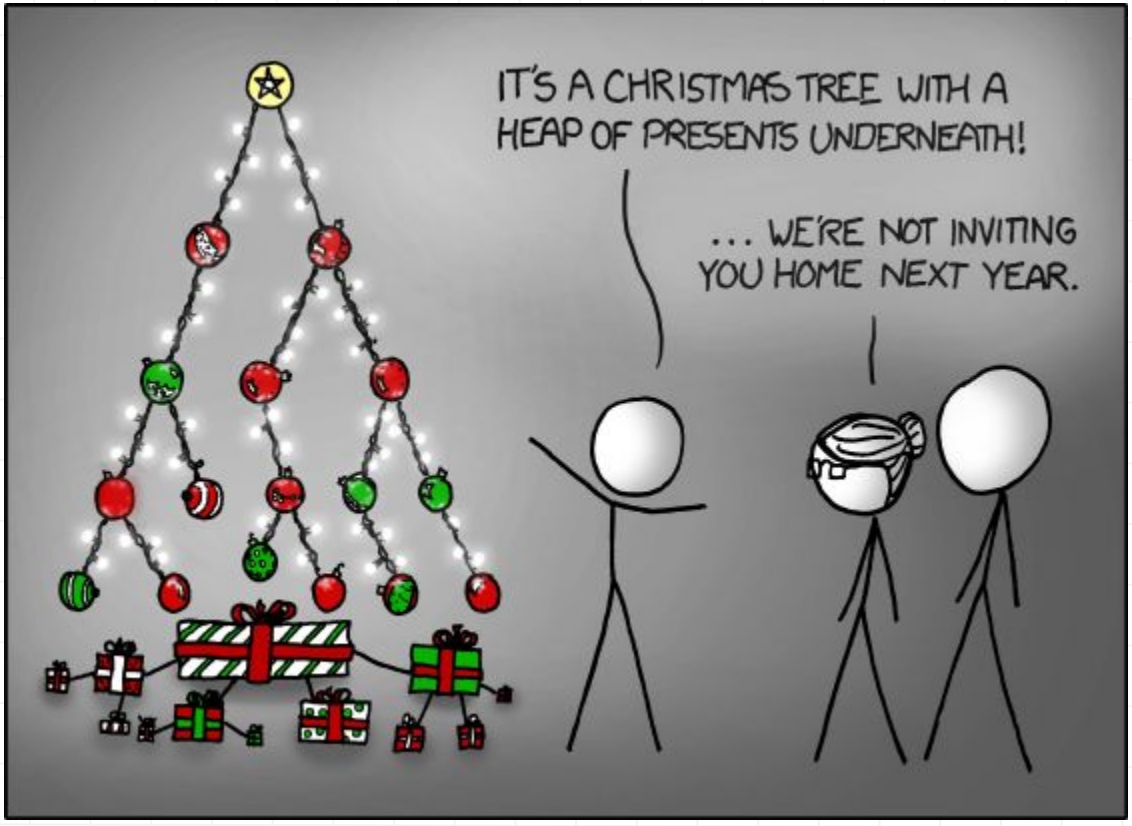
https://xkcd.com/835/



Announcements:

- Guest lecture this Wed
  PHYS T + Teams
  ML for cyber security
  See you here!

- Two graphical model tutorials: this week and next week.

- Hope you're going well in assignment 2
- Final exam Fri 3 June
  5:40 - 8:40 pm Canberra
  - On Wattle
  - Self-invigilated, video recording needed
  - Instructions will be available

Graphical model inference

Bishop 8.4.3, 8.4.4

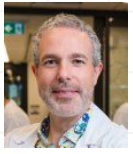
Factor graphs

The sum-product algorithm

Other algorithms

High-level goal:
One representation for both directed and undirected graphical models.
Use this representation to facilitate inference.

[Frey, 1998, Kschischnang et al 2001]



Factor graphs and the sum-product algorithm 7780 2001
FR Kschischang, BJ Frey, HA Loeliger
IEEE Transactions on information theory 47 (2), 498-519

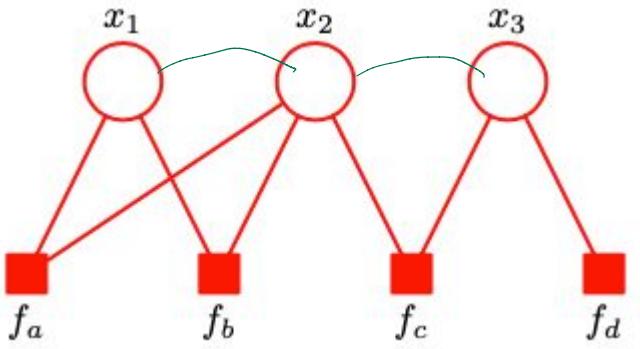
Factor graphs

undirected / directed graphical model.

s: a subset of variables

Joint distribution → a product of factors

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) \qquad (8.59)$$



$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3). \qquad (8.60)$$

Contains more info than MRF, because $f_a(x_1, x_2)$ and $f_b(x_1, x_2)$ would be in one potential function.

$\psi_{12}(x_1, x_2)$

e.g. f_a related humidity

f_b " temperature.

Directed graph: conditionals → factors

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k | \mathrm{pa}_k) \qquad (8.5)$$

Undirected graph: potential functions over max cliques → factors

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C). \qquad (8.39)$$

factors are "unnormalised"

Z is a factor too, over an empty set of variables.

Example: factor graphs representing the same distribution

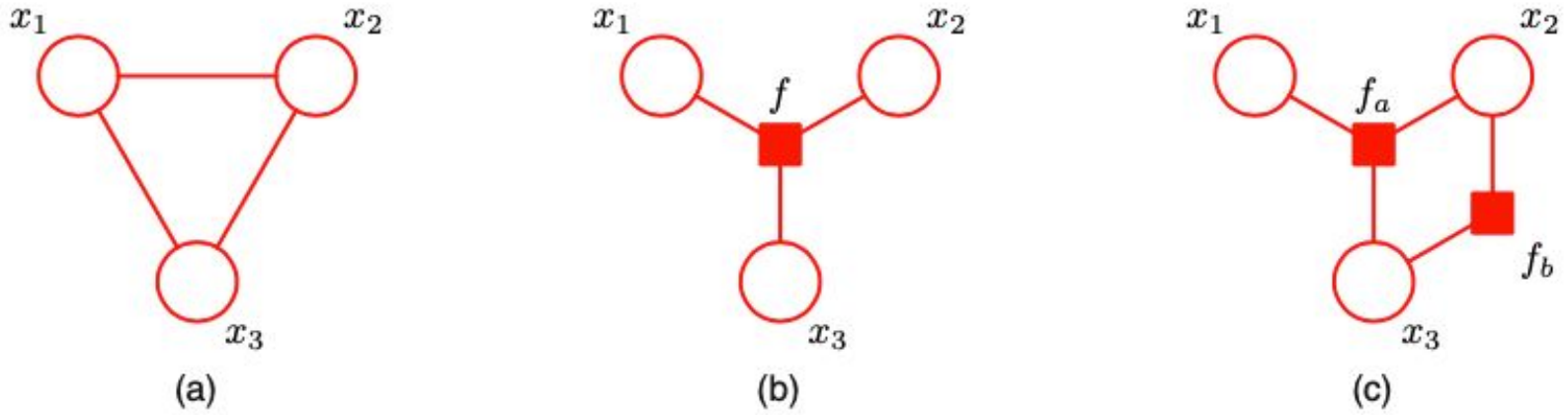


Figure 8.41 (a) An undirected graph with a single clique potential $\psi(x_1, x_2, x_3)$. (b) A factor graph with factor $f(x_1, x_2, x_3) = \psi(x_1, x_2, x_3)$ representing the same distribution as the undirected graph. (c) A different factor graph representing the same distribution, whose factors satisfy $f_a(x_1, x_2, x_3) f_b(x_1, x_2) = \psi(x_1, x_2, x_3)$.

Example: factor graphs representing the same distribution

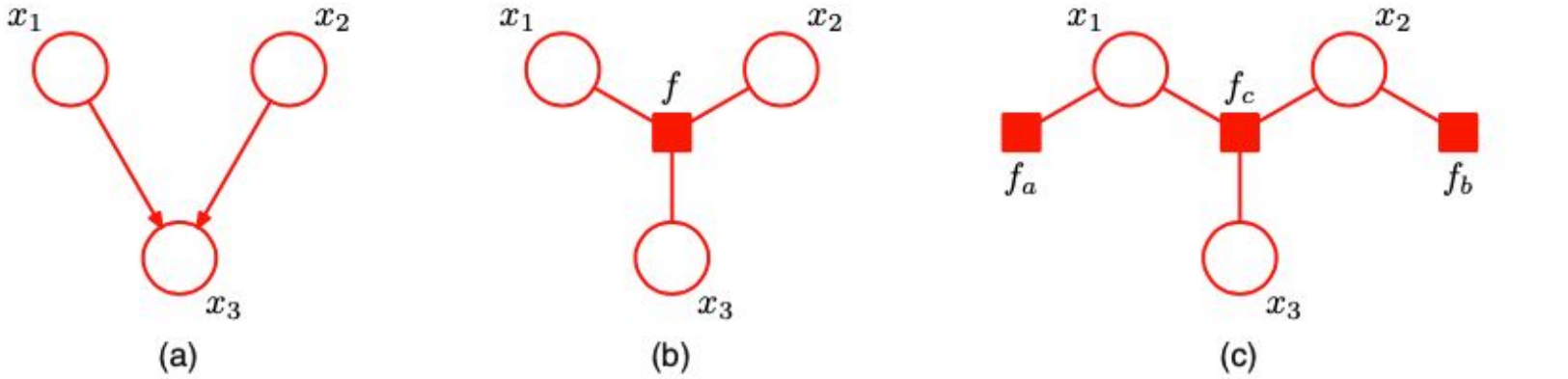
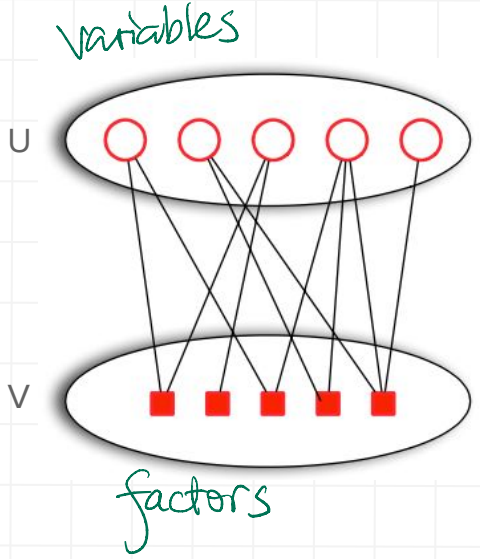


Figure 8.42 (a) A directed graph with the factorization $p(x_1)p(x_2)p(x_3|x_1, x_2)$. (b) A factor graph representing the same distribution as the directed graph, whose factor satisfies $f(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$. (c) A different factor graph representing the same distribution with factors $f_a(x_1) = p(x_1)$, $f_b(x_2) = p(x_2)$ and $f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2)$.

Factor graphs are bipartite

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.



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.

BN —moralisation→ MRF → FG

BN+MRF:
intuitive to think about variable dependencies

FG:
better for constructing inference

Factor graph for a poly-tree

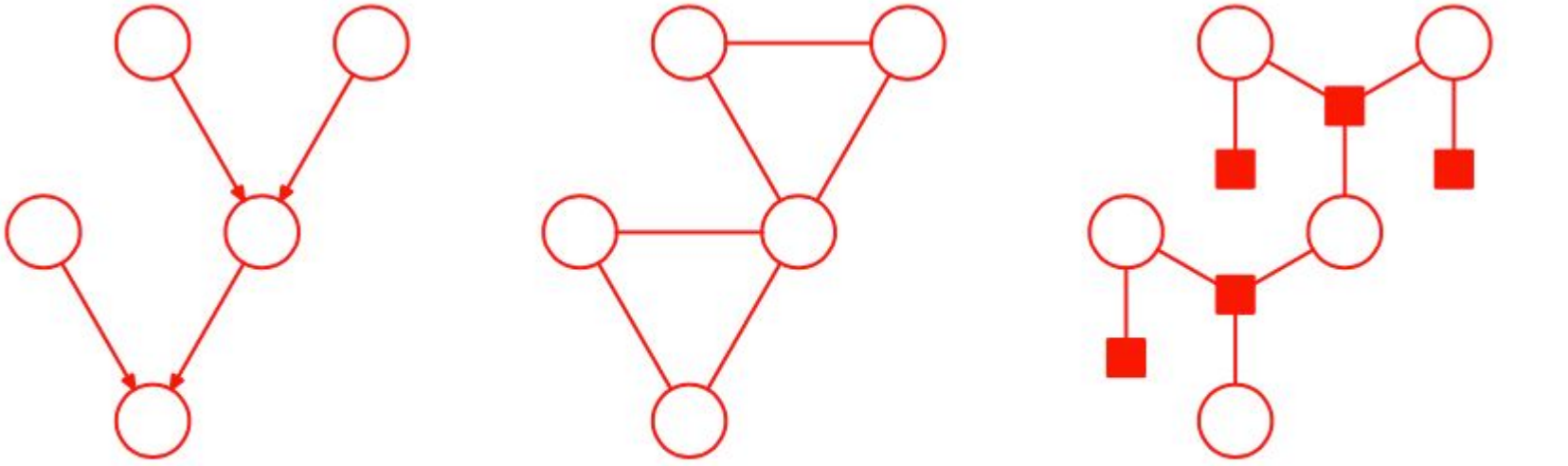
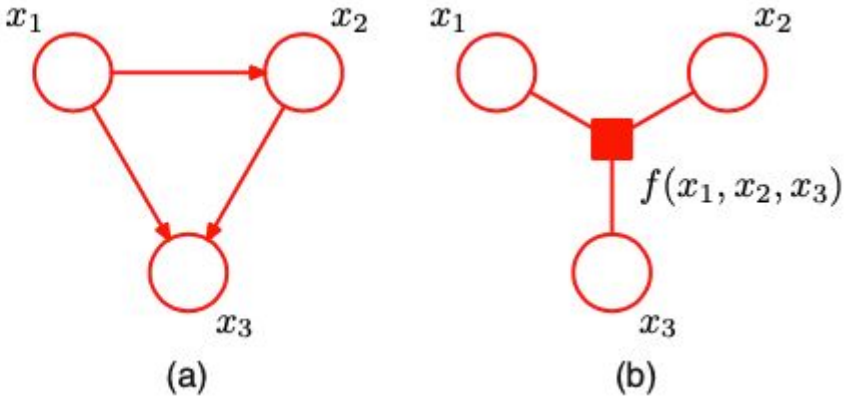


Figure 8.43 (a) A directed polytree. (b) The result of converting the polytree into an undirected graph showing the creation of loops. (c) The result of converting the polytree into a factor graph, which retains the tree structure.

Factor graphs for directed/undirected trees, and directed poly trees retain a tree structure.

Figure 8.44 (a) A fragment of a directed graph having a local cycle. (b) Conversion to a fragment of a factor graph having a tree structure, in which $f(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$.



BN ⊂ fully general factorisation

MRF : ψ_{123}

↓

FG $f(x_1, x_2, x_3)$

Multiple factor graphs representing the same (undirected) graph

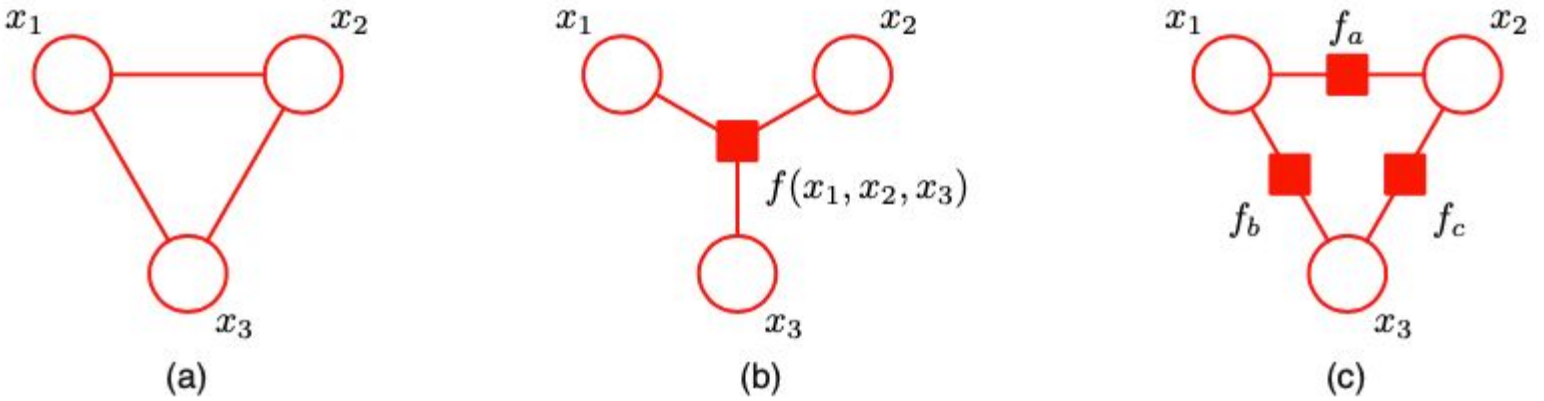


Figure 8.45 (a) A fully connected undirected graph. (b) and (c) Two factor graphs each of which corresponds to the undirected graph in (a).

Factorization in (c) does not correspond to any conditional independence properties.

FG with loops : "bad" for inference.

The sum-product algorithm

Goal: evaluate local marginals over nodes or subsets of nodes.
Variant: find the most probable state → max sum algorithm.

Assume (for instructional convenience): all variables discrete. Marginalization = sums
For continuous variables - perform integration, e.g. linear dynamic systems (e.g. Chap 13.3)

Historic note:
belief propagation (Pearl, 1988; Lauritzen and Spiegelhalter, 1988) perform exact inference on DAGs. It is a special case of the sum product algorithm (Frey, 1998; Kschischnang et al., 2001)

Recall from last lecture ...

$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}. \qquad (8.52)$$

$$p(x_n) = \frac{1}{Z} \mu_\alpha(x_n) \mu_\beta(x_n). \qquad (8.54)$$

O(NK^2)

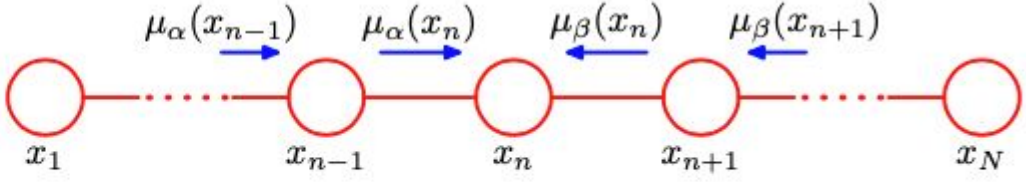


Figure 8.38 The marginal distribution $p(x_n)$ for a node x_n along the chain is obtained by multiplying the two messages $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$, and then normalizing. These messages can themselves be evaluated recursively by passing messages from both ends of the chain towards node x_n.

The sum-product algorithm

① do summation first (when we can)
② multiply less 2 → 1

$$ab + ac = a(b + c) \tag{8.53}$$

one var want marginal dist.

$$p(x) = \sum_{\mathbf{x}\setminus x} p(\mathbf{x}) = \sum_{\mathbf{x}\setminus x} \prod_s f_s(\mathbf{x}_s) \tag{8.61}$$

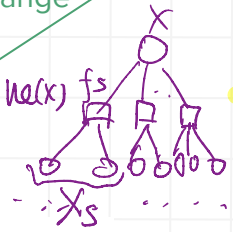
$O(K^{N-1})$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) \tag{8.59}$$

rearrange

product of $\{f_s\}$

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s) \tag{8.62}$$



s: a subset of nodes
ne(x) : *neighbouring* factor nodes (bipartite)
X_s: the set of all variables in the subtree connected to the variable node x via the factor node fs
$F_s(x,X_s)$: the product of all the factors in the group associated with factor f_s.

$$p(x) = \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x). \tag{8.63}$$

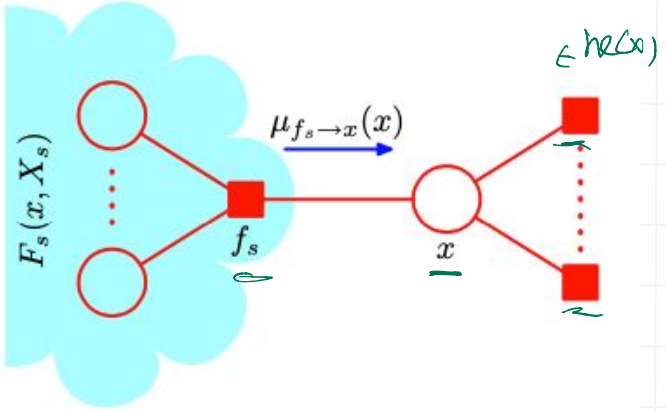
rename

$\bigcup_{s \in \mathrm{ne}(x)} X_s = \vec{X} \setminus x$

$X_s \cap X_{s'} = \phi$

where $\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$ (8.64)

Fig 8.46



A first example of sum-product

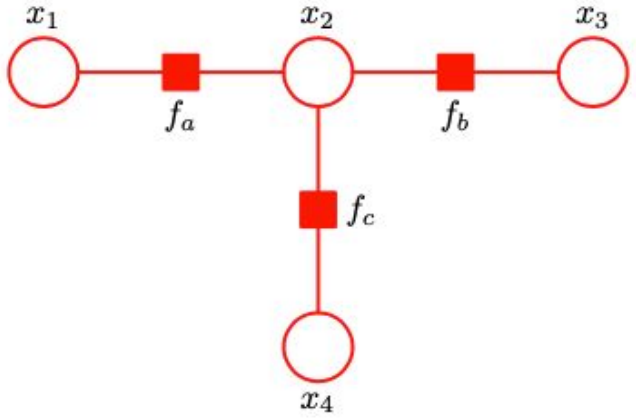
$$p(x) = \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x). \quad (8.63)$$

where $$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s) \quad (8.64)$$

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

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

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



What about $F_s(x, X_s)$?

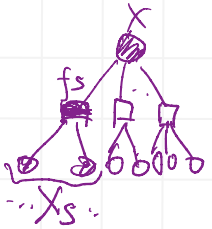
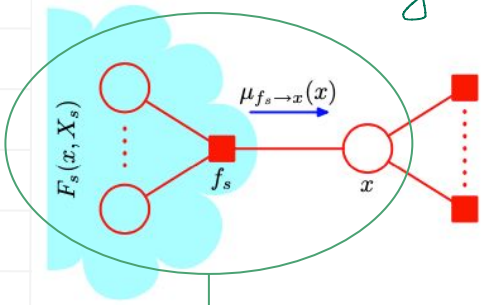


Fig 8.46



$F_s(x,X_s)$: the product of all the factors in the group associated with factor f_s.
X_s: the set of all variables in the subtree connected to the variable node x via the factor node fs

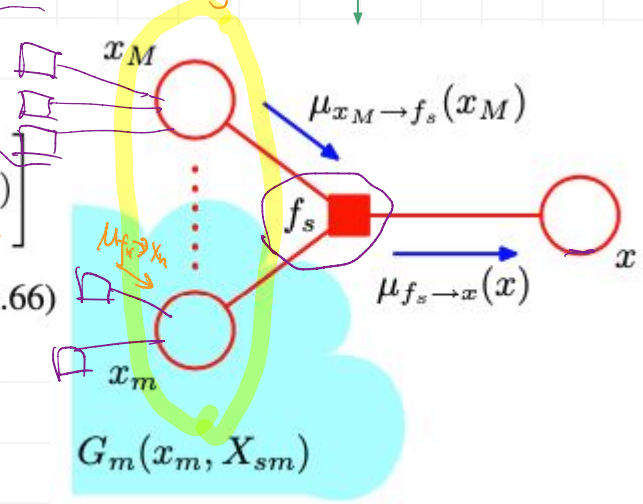
Factorise this factor subgraph:

$\mathbf{X}_s = \{x, x_1, \ldots, x_M\}$

grandchildren of x

$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) G_1(x_1, X_{s1}) \ldots G_M(x_M, X_{sM}) \quad (8.65)$$

Fig 8.47



ne(f_s) : *neighbours of* factor node f_s
ne(f_s)\x : *neighbours of* factor node f_s except *x*

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

X_{xm} ← grandchildren of x_m

$$= \sum_{x_1} \ldots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \quad (8.66)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}). \quad (8.67)$$

node-to-factor messages.

recap

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

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

Two different kinds of messages:

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s) \qquad (8.64)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}). \qquad (8.67)$$

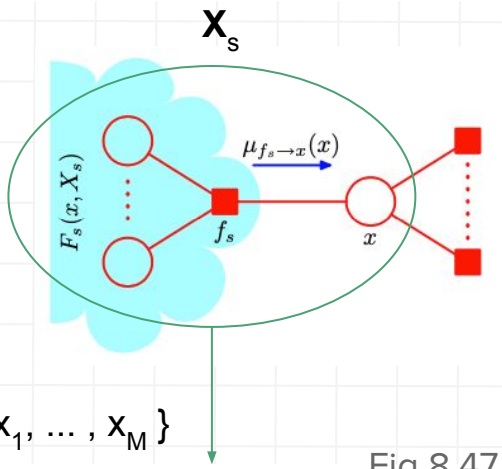
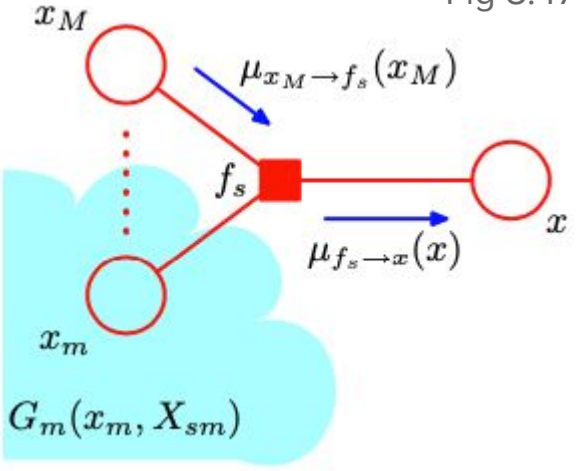


Fig 8.47

Computing factor-to-variable-message

1. take the product of the incoming messages along all other links coming into the factor node
2. multiply by the factor associated with that node
3. marginalize over all of the variables associated with the incoming messages

can send a message once the factor node has received incoming messages from all other neighbouring variable nodes



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

$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \qquad (8.66)$$

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.

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

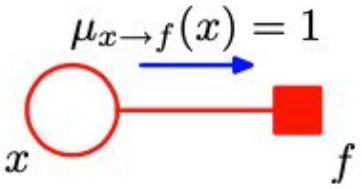
$$\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

the marginals are then

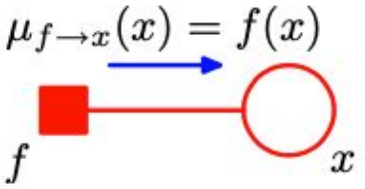
$$p(x_m) = \prod_{l \in \mathrm{ne}(x_m)} \mu_{f_l \to x_m}(x_m)$$

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



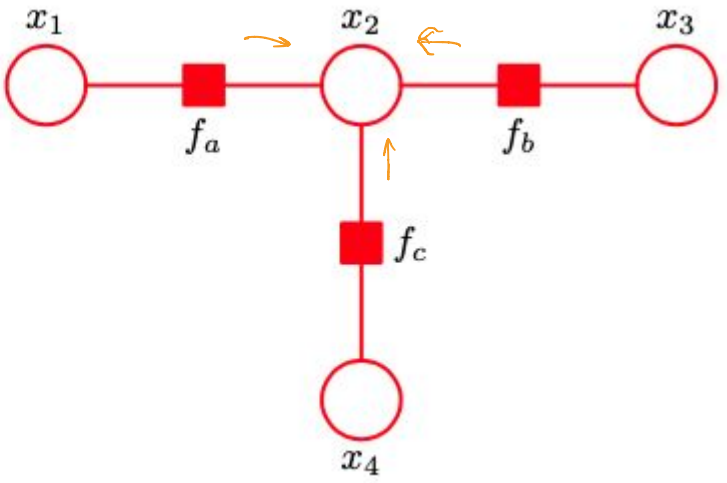
  - If the leaf node is a factor node then



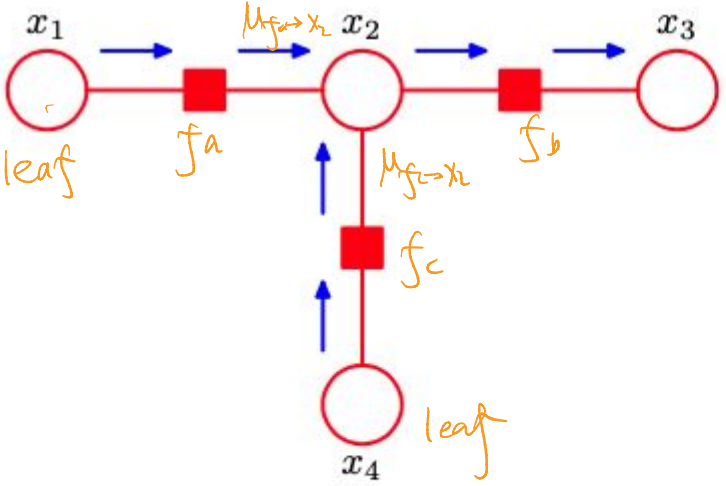
- These initialisation rules follow from the general formulae.
- Normalisation (if the factors are un-normalised):
  - Calculate $\tilde{p}(x)$ by message passing as before
  - Then $Z = \sum_x \tilde{p}(x)$ and $p = \frac{1}{Z}\tilde{p}$

Example

Figure 8.51 A simple factor graph used to illustrate the sum-product algorithm.



$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4). \tag{8.73}$$



$$\mu_{x_1 \to f_a}(x_1) = 1 \tag{8.74}$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2) \tag{8.75}$$

$$\mu_{x_4 \to f_c}(x_4) = 1 \tag{8.76}$$

$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4) \tag{8.77}$$

$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \tag{8.78}$$

$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}(x_2). \tag{8.79}$$

Figure 8.52 Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes x_1 and x_4 towards the root node x_3. (b) From the root node towards the leaf nodes.

$$\mu_{x_3 \to f_b}(x_3) = 1 \tag{8.80}$$

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3) \tag{8.81}$$

$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \tag{8.82}$$

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2)\mu_{x_2 \to f_a}(x_2) \tag{8.83}$$

$$\mu_{x_2 \to f_c}(x_2) = \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2) \tag{8.84}$$

$$\mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4)\mu_{x_2 \to f_c}(x_2). \tag{8.85}$$

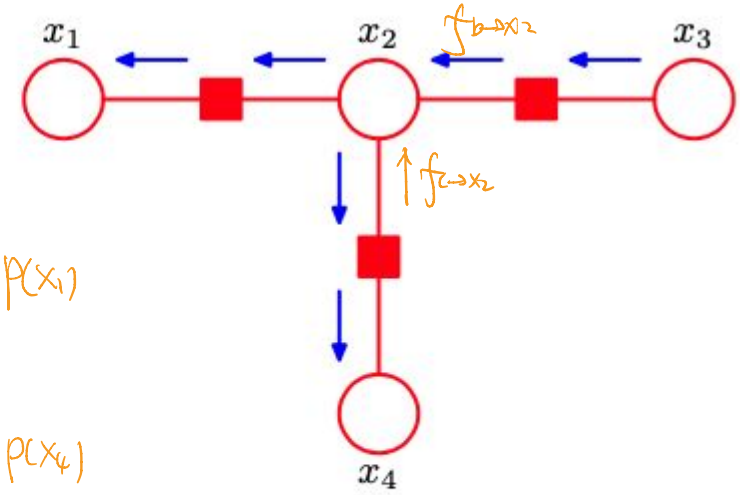


Figure 8.52 Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes x_1 and x_4 towards the root node x_3. (b) From the root node towards the leaf nodes.

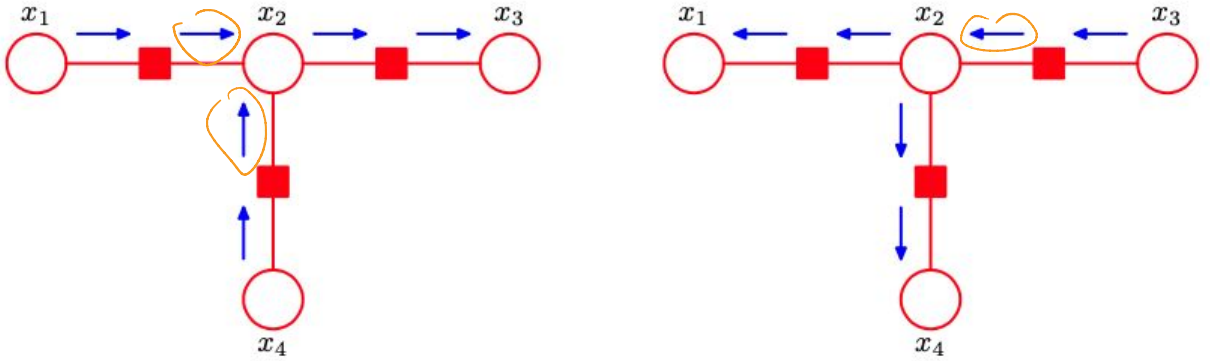


Figure 8.52 Flow of messages for the sum-product algorithm applied to the example graph in Figure 8.51. (a) From the leaf nodes x_1 and x_4 towards the root node x_3. (b) From the root node towards the leaf nodes.

$$
\begin{aligned}
\widetilde{p}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
&= \left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right]\left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
&= \sum_{x_1}\sum_{x_3}\sum_{x_4} \widetilde{p}(\mathbf{x}) \qquad (8.86)
\end{aligned}
$$

- 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

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.

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.

$a > 0$

$\max(ab, ac) = a \max(b, c)$

Aji, S.M.; McEliece, R.J. (Mar 2000). "The generalized distributive law". IEEE Transactions on Information Theory. 46 (2): 325–343. doi:10.1109/18.825794.

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 $\operatorname{argmax}_x p(\mathbf{x})$
- For Hidden Markov Models, this is the Viterbi algorithm.

"decoding"

+ speech reco.

Sound → word → sentences

+ language model

+ "wireless" radio.

What was the 0/1 message originally sent?

tree of cliques.

- 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.

then "make cliques"

- 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.

Graphical model inference

Factor graphs

The sum-product algorithm

Other algorithms