Inference on graphs

- Given a graph and its conditionals or potentials compute

$p(\theta|e)$ (particular θ and e, marginalizing out other variables)

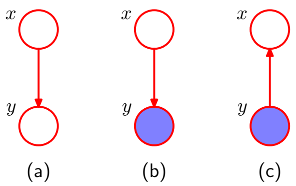
$p(X)$ (particular event, marginalizing out other variables)

argmax $p(\theta|e)$ (particular θ and e; marginalizing other variables)

argmax $p(\theta,\theta_N,e_N|e)$ (all variables, will nuisance / unobserved)

Inference on graphs

- Simplest example (Bayes' rule)
  - (a) model
  - (b) illustrates observed
  - (c) inference **reverses** the arrow

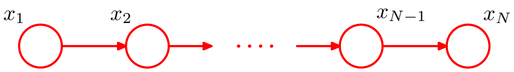


- Computationally

$$p(x|y)=\frac{p(y|x)p(x)}{\sum_{x'}p(y|x')p(x')}$$

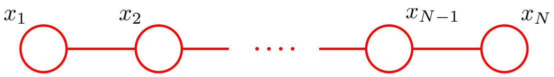
Marginals on a chain

Recall



$$p(\mathbf{x})=p(x_1)p(x_2|x_1)p(x_3|x_2)\ \cdots\ p(x_{N-1}|x_{N-2})p(x_N|x_{N-1})$$

Converted to

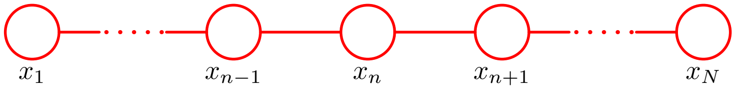


$$p(\mathbf{x})=\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ \ldots\ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

Assume N discrete variables, with K values each.

Compute the marginal of a node in the middle, $p(x_n)$

Marginals on a chain



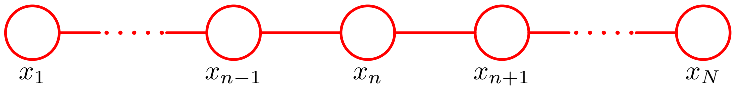
$$p(\mathbf{x})=\psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3)\ \ldots\ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

Direct calculation of $p(x_n)$

$$p(x_n)=\sum_{x_1}\sum_{x_2}\ \ldots\ \sum_{x_{n-1}}\sum_{x_{n+1}}\ \ldots\ \sum_{x_{N-1}}\sum_{x_N}p(\mathbf{x})$$

Computational complexity is O(K^N). Way too slow!

Computing marginals on a chain efficiently



$$p(\mathbf{x}) = \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3) \ \ \ldots \ \ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

Main idea is to rearrange terms to exploit conditional independence.

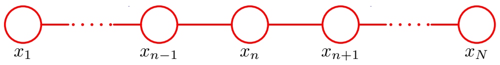
Fancy formulas from algebra

$$\underbrace{\sum_{x_1,\, x_2,\, \ldots,\, x_N}}_{\text{all values of each}} f(x_1, x_2, \ldots, x_N) = \underbrace{\sum_{x_1}\sum_{x_2}\ldots\sum_{x_N}}_{\text{any order you like}} f(x_1, x_2, \ldots, x_N)$$

(essentially a definition)

$$\left(\sum a_i\right)\left(\sum b_j\right) = \sum\sum a_i b_j$$

Back to marginals on a chain



$$p(\mathrm{x}) = \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3) \ \ \ldots \ \ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

$$p(x_n) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\underbrace{\psi_{1,2}(x_1,x_2)\ldots\psi_{n-1,n}(x_{n-1},x_n)}_{f_L(x_1,x_1,\ldots,x_n)}\underbrace{\psi_{n,n+1}(x_n,x_{n+1})\ldots\psi_{N-1,N}(x_{N-1},x_N)}_{f_R(x_n,x_{n+1},\ldots,x_N)}$$

$$= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N} f_L(x_1,x_1,\ldots,x_n) f_R(x_n,x_{n+1},\ldots,x_N)$$

$$= \left(\sum_{x_1}\sum_{x_2}\ldots\sum_{x_{n-1}} f_L(x_1,x_1,\ldots,x_n)\right)\left(\sum_{x_{n+1}}\ldots\sum_{x_{N-1}}\sum_{x_N} f_R(x_n,x_{n+1},\ldots,x_N)\right)$$

$$= \left(\sum_{x_1}\sum_{x_2}\ldots\sum_{x_{n-1}}\prod_{i=1}^{n-1}\psi_{i,i+1}(x_i,x_{i+1})\right)\left(\sum_{x_{n+1}}\ldots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})\right)$$

$$= \left(\sum_{x_{n-1}}\sum_{x_{n-2}}\ldots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})\right)\left(\sum_{x_{n+1}}\ldots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})\right)$$

$$p(x_n) = \left(\sum_{x_{n-1}}\sum_{x_{n-2}}\ldots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}(x_{n-i},x_{n-i+1})\right)\left(\sum_{x_{n+1}}\ldots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}(x_i,x_{i+1})\right)$$

$$\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}(x_{n-i},x_{n-i+1}) = \psi_{n-1,\,n}(x_{n-1},x_n)\psi_{n-2,\,n-1}(x_{n-2},x_{n-2}) \ \ldots \ \psi_{2,\,3}(x_2,x_3)\psi_{1,\,2}(x_1,x_2)$$

More warmup

$$\sum_{x_2}\sum_{x_1}\underbrace{\psi\left(x_2,x_3\right)}_{\substack{\text{No dependency on } x_1\\ \text{hence we can move}\\ \text{factor outside sum}\\ \text{over } x_1.}}\ \psi\left(x_1,x_2\right)=\sum_{x_2}\psi\left(x_2,x_3\right)\underbrace{\sum_{x_1}\psi\left(x_1,x_2\right)}_{\substack{\text{Vector of size over the}\\ \text{components of } x_2}}$$

(Recall the distibutive law: $ba+ca=a(b+c)$)

For example, K=2, the first component of the sum, x_3^1, is:

$$\begin{aligned}
&\sum_{x_2}\sum_{x_1}\psi\left(x_2,x_3^1\right)\psi\left(x_1,x_2\right)\\
&\quad=\psi\left(x_2^1,x_3^1\right)\psi\left(x_1^1,x_2^1\right)+\psi\left(x_2^1,x_3^1\right)\psi\left(x_1^2,x_2^1\right)+\psi\left(x_2^2,x_3^1\right)\psi\left(x_1^1,x_2^2\right)+\psi\left(x_2^2,x_3^1\right)\psi\left(x_1^2,x_2^2\right)\\
&\quad=\psi\left(x_2^1,x_3^1\right)\bullet\left(\psi\left(x_1^1,x_2^1\right)+\psi\left(x_1^2,x_2^1\right)\right)+\psi\left(x_2^2,x_3^1\right)\bullet\left(\psi\left(x_1^1,x_2^2\right)+\psi\left(x_1^2,x_2^2\right)\right)\\
&\quad=\psi\left(x_2^1,x_3^1\right)\left(\sum_{\{x_1^i\}}\psi\left(x_1^i,x_2^1\right)\right)+\psi\left(x_2^2,x_3^1\right)\left(\sum_{\{x_1^i\}}\psi\left(x_1^i,x_2^1\right)\right)\\
&\quad=\sum_{\{x_2^i\}}\psi\left(x_2^i,x_3^1\right)\sum_{\{x_1^i\}}\psi\left(x_1^i,x_2^1\right)
\end{aligned}$$

$$p\left(x_n\right)=\left(\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\right)\left(\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}\left(x_i,x_{i+1}\right)\right)$$

$$\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)=\psi_{n-1,\,n}\left(x_{n-1},x_n\right)\psi_{n-2,\,n-1}\left(x_{n-2},x_{n-2}\right)\ \ldots\ \psi_{2,\,3}\left(x_2,x_3\right)\psi_{1,\,2}\left(x_1,x_2\right)$$

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)=\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_2}\prod_{i=1}^{n-2}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\underbrace{\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}}_{\substack{\text{This has K elements, indexed}\\ \text{by } x_2}}$$

$$p\left(x_n\right)=\left(\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\right)\left(\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}\left(x_i,x_{i+1}\right)\right)$$

$$\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)=\psi_{n-1,\,n}\left(x_{n-1},x_n\right)\psi_{n-2,\,n-1}\left(x_{n-2},x_{n-2}\right)\ \ldots\ \psi_{2,\,3}\left(x_2,x_3\right)\psi_{1,\,2}\left(x_1,x_2\right)$$

$$\begin{aligned}
\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)&=\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_2}\prod_{i=1}^{n-2}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}\\
&=\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_3}\prod_{i=1}^{n-3}\psi_{n-i,\,n-i+1}\left(x_{n-i-1},x_{n-i}\right)\left\{\underbrace{\sum_{x_2}\psi_{2,3}\left(x_2,x_3\right)\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}}_{\text{Like a dot product for each } x_3\text{, with elements indexed by } x_2}\right\}
\end{aligned}$$

$$p\left(x_n\right)=\left(\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\right)\left(\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,\,i+1}\left(x_i,x_{i+1}\right)\right)$$

$$\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)=\psi_{n-1,\,n}\left(x_{n-1},x_n\right)\psi_{n-2,\,n-1}\left(x_{n-2},x_{n-2}\right)\ \ldots\ \psi_{2,\,3}\left(x_2,x_3\right)\psi_{1,\,2}\left(x_1,x_2\right)$$

$$\begin{aligned}
\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)&=\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_2}\prod_{i=1}^{n-2}\psi_{n-i,\,n-i+1}\left(x_{n-i},x_{n-i+1}\right)\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}\\
&=\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_3}\prod_{i=1}^{n-3}\psi_{n-i,\,n-i+1}\left(x_{n-i-1},x_{n-i}\right)\left\{\sum_{x_2}\psi_{2,3}\left(x_2,x_3\right)\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}\right\}\\
&\ \ \cdots\\
&=\left\{\sum_{x_{n-1}}\psi_{n-1,n}\left(x_{n-1},x_n\right)\cdots\left\{\sum_{x_3}\psi_{3,4}\left(x_3,x_4\right)\left\{\sum_{x_2}\psi_{2,3}\left(x_2,x_3\right)\left\{\sum_{x_1}\psi_{1,2}\left(x_1,x_2\right)\right\}\right\}\right\}\right\}
\end{aligned}$$

$$p(x_n)=\left(\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})\right)\left(\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})\right)$$

where

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})=\left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\cdots\left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\left\{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\sum_{x_1}\psi_{1,2}(x_1,x_2)\right\}\right\}\right\}\cdots\right\}$$

and

$$\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})=\left\{\sum_{x_{n+1}}\psi_{n,n+1}(x_n,x_{n+1})\cdots\left\{\sum_{x_{N-1}}\psi_{N-2,N-1}(x_{N-2},x_{N-1})\left\{\sum_{x_N}\psi_{N-1,N}(x_{N-1},x_N)\right\}\right\}\cdots\right\}$$

(Deriving the right factor is similar to doing the left one which we did in detail.)

Matrix interpretation (for two variables)

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})=\left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\cdots\left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\left\{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\sum_{x_1}\psi_{1,2}(x_1,x_2)\right\}\right\}\right\}\cdots\right\}$$

$\psi_{i,\,i+1}(x_i,x_{i+1})\Leftrightarrow Q_{i+1,\,i}$ (note transposition!)

$\sum_i\psi_{i,\,i+1}(x_i,x_{i+1})\Leftrightarrow$ sums columns to get a vector V_{i+1}

$$\sum_{x_{i+1}}\psi_{i+1,\,i+2}(x_{i+1},x_{i+2})\left\{\sum_{x_i}\psi_{i,\,i+1}(x_i,x_{i+1})\right\}\Leftrightarrow Q_{i+2,\,i+1}\bullet\mathrm{V}_i$$

Computational Complexity

Suppose each variable has K values
What is the cost of evaluating the first factor?

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})=\left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\cdots\left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\underbrace{\left\{\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\underbrace{\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\text{K sums of K values}}\right\}}_{\text{K evaluations of K products}}\right\}}_{\text{K sums of K values}}\right\}\cdots\right\}$$

The cost for computing the part shown in orange is $\mathrm{O}(K^2)$.

Computational Complexity

Suppose each variable has K values
What is the cost of evaluating the first factor?

$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})=\left\{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\cdots\left\{\sum_{x_3}\psi_{3,4}(x_3,x_4)\underbrace{\left\{\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\underbrace{\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\text{K sums of K values}}\right\}}_{\text{K evaluations of K products}}\right\}}_{\text{K sums of K values}}\right\}\cdots\right\}$$

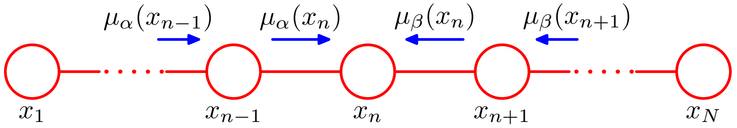
The cost of the left factor is $\mathrm{O}(N\cdot K^2)$.

The other factor is similar.

We see that the overall cost is $\mathrm{O}(N\cdot K^2)$.

Much better than the naive computation where we had $\mathrm{O}(K^N)$!

Message passing interpretation

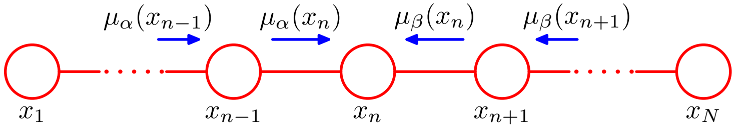


Define $\mu_\alpha(x_n)$ as a message passed from node x_{n-1} to node x_n.

Define $\mu_b(x_n)$ as a message passed from node x_{n+1} to node x_n.

Passing messages will correspond to the computation of taking input messages, and computing output messages.

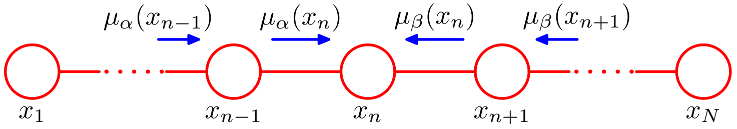
Message passing interpretation



$$\sum_{x_{n-1}}\sum_{x_{n-2}}\cdots\sum_{x_1}\prod_{i=1}^{n-1}\psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1})=$$

$$\left\{\underbrace{\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1},x_n)\cdots\left\{\underbrace{\sum_{x_3}\psi_{3,4}(x_3,x_4)\left\{\underbrace{\sum_{x_2}\psi_{2,3}(x_2,x_3)\left\{\underbrace{\sum_{x_1}\psi_{1,2}(x_1,x_2)}_{\mu_\alpha(x_2)}\right\}}_{\mu_\alpha(x_3)}\right\}}_{\mu_\alpha(x_4)}\right\}\cdots}_{\mu_\alpha(x_n)}\right\}$$

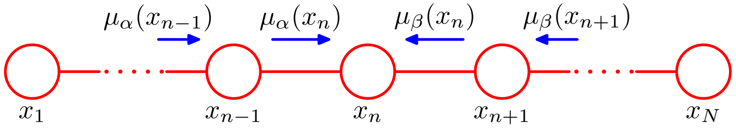
Message passing interpretation



$$\sum_{x_{n+1}}\cdots\sum_{x_{N-1}}\sum_{x_N}\prod_{i=n}^{N-1}\psi_{i,i+1}(x_i,x_{i+1})=$$

$$\left\{\underbrace{\sum_{x_{n+1}}\psi_{n,n+1}(x_n,x_{n+1})\cdots\left\{\underbrace{\sum_{x_{N-1}}\psi_{N-2,N-1}(x_{N-2},x_{N-1})\left\{\underbrace{\sum_{x_N}\psi_{N-1,N}(x_{N-1},x_N)}_{\mu_b(x_{N-1})}\right\}}_{\mu_b(x_{N-2})}\right\}\cdots}_{\mu_b(x_n)}\right\}$$

Message passing interpretation

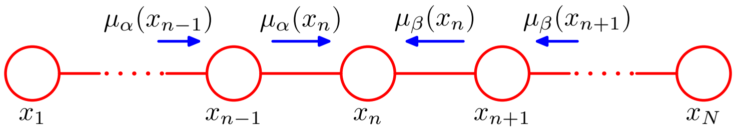


$$p(x_n)=\frac{1}{Z}\mu_a(x_n)\mu_b(x_n)$$

Algorithm

Send a message from x_1 to x_n.
Send a message from x_N to x_n.
Element wise multiply messages.
Normalize by sum (Z).

Computing all marginals



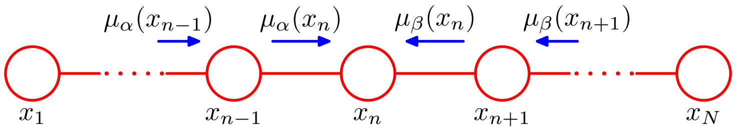
To compute all marginals, send a message from left to right, and right to left, storing the result.

Now compute any marginal as before.

This way, computing all marginals is only twice as expensive as computing one of them.

The normalization constant is easily computed using any convenient node.

What if a node is observed?



If a node is observed, then we do the obvious. Specifically, we clamp the values of variables to the particular case.

This means that messages flowing into it, do not affect messages flowing out, which are set to the "clamped" value.

Factor Graphs

Suppose p(**x**) factorizes as:

$$p(\mathbf{x}) = \prod_s f(x_s) \qquad \text{where } x_s \text{ are sets of of variables within } \mathbf{x}.$$

Make a node for each x_i as usual.

Now, make a different kind of node for *f()* (e.g., squares).

Draw edges between the factor nodes and the variables in the variable set, *s*.

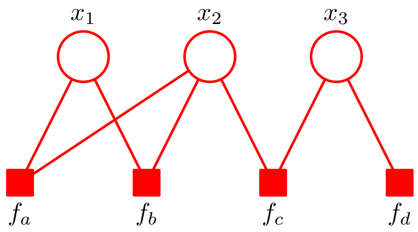
Note that the factorization formula means that we can convert **both** directed and undirected graphs to factor graphs.

Factor Graph Example

Suppose p(**x**) factorizes as:

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

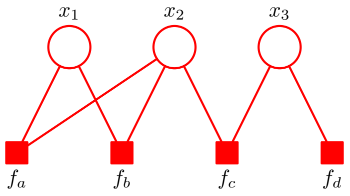
The graph is:



Factor Graph Example (continued)

Suppose p(**x**) factorizes as:

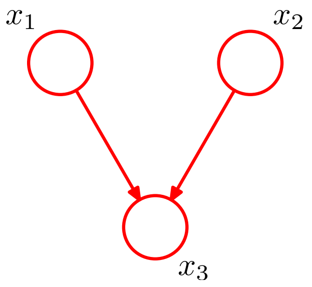
$$p(\mathbf{x}) = \prod_s f(x_s) = f_a(x_1,x_2) f_b(x_1,x_2) f_c(x_2,x_3) f_d(x_3)$$

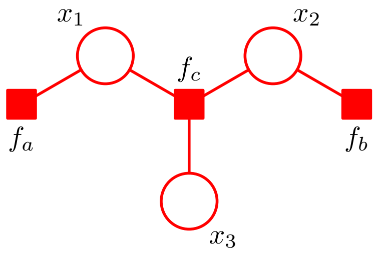


This layout emphasizes that factor graphs are *bipartite*.

Note two factors for the clique for 1 and 2, suggesting that factor graphs can preserve extra structure compared to undirected graphs.

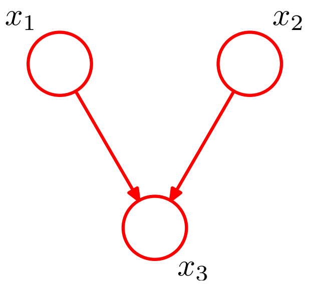
Factor Graph Example (2)

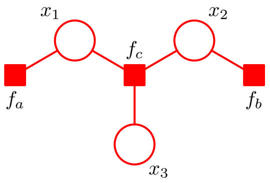


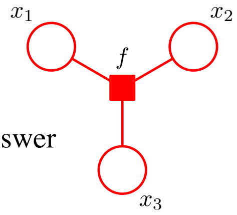


$$p(\mathbf{x}) = \underbrace{p(x_1)}_{f_a}\underbrace{p(x_2)}_{f_b}\underbrace{p(x_3|x_1,x_2)}_{f_c}$$

Factor Graph Example (2)







second answer

Factor Graph Summary

$p(\mathbf{x}) = \prod_s f(x_s)$ where x_s are sets of of variables within **x**.

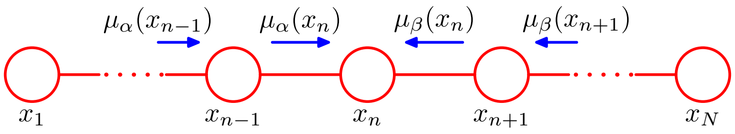
Denote variables by circles

Denote each factor by a square

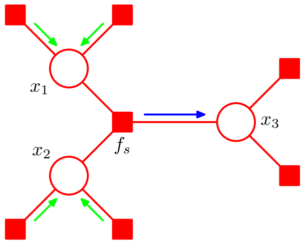
Draw links between squares and variables in the sets x_s.

Factor graphs are bipartite

Factor graph for a distribution is not necessarily unique.



Factor graphs conveniently represent the extended message passing needed for inference on trees/polytrees.

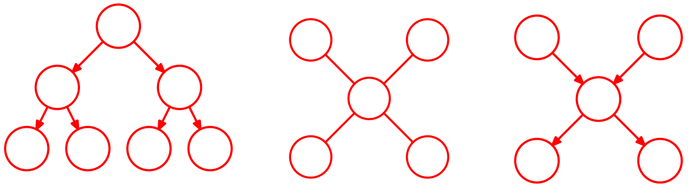


Trees/Polytrees

A directed graph is tree if the root node has no parents, others have exactly one parent.

An undirected graph is a tree if there is only one path between any pair of nodes.

A directed graph is a polytree if there is only one path per pair of nodes.



Factor Graphs and Trees

The factor graphs for directed trees, undirected trees, and directed polytrees are all trees.

(Recall definition for undirected trees---there is only one path between any two nodes).

This means that (variable) node, x_n, with K branches divides a tree into K subtrees whose factors do not share variables except x_n.

Observations about factor graphs for trees

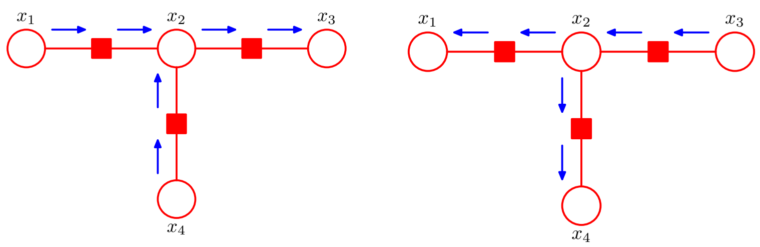
Any node can be root

Any node with K links splits the graph into K subgraphs which do not share nodes.

If we pass messages from:
1) the leaves to a chosen root;
2) the chosen root to the leaves,
then **all** messages that **can** be passed **have** been passed.

Further, the number of messages in 1 and 2 are the same.

Observations about factor graphs



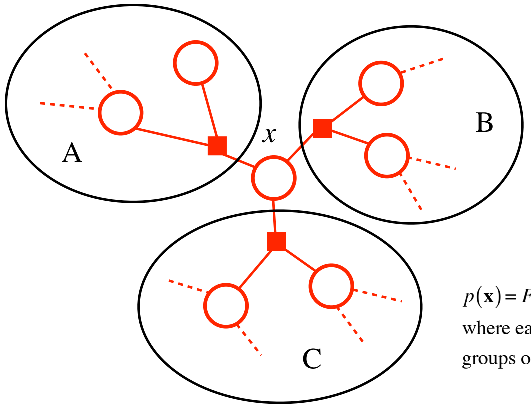
(x_3 is the root)

Sum-product algorithm

Generalizes what we did with chains.

Generalizes and simplifies an algorithm introduced as “belief propagation”.

As with chains, consider the problem of computing the marginal of a selected node, x_n.



x connects subgraphs with node sets A, B, C.

$p(\mathbf{x}) = F(x,X_A)F(x,X_B)F(x,X_C)$

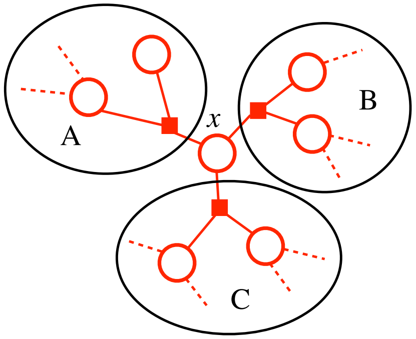
where each of these three factors are themselves groups of factors over x and the subgraphs.

More explicitly,

$F(x,X_A) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup A$

$F(x,X_B) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup B$

$F(x,X_C) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup C$



$p(\mathbf{x}) = F(x,X_A)F(x,X_B)F(x,X_C)$

where each of these three factors are themselves groups of factors over x and the subgraphs.

More explicitly,

$F(x,X_A) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup A$

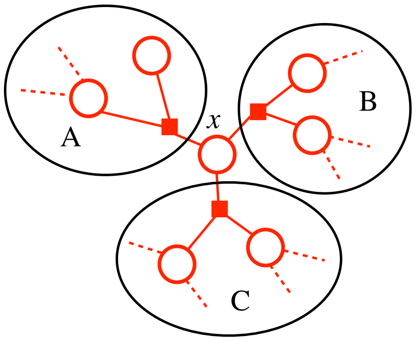
$F(x,X_B) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup B$

$F(x,X_C) = \prod_s f(X_s)$ with $X_s \subseteq \{x\} \cup C$

$$p(x) = \sum_{X\backslash\{x\}} p(\mathbf{x}) = \sum_{X\backslash\{x\}} F(x,X_A)F(x,X_B)F(x,X_C) = \left(\sum_A F(x,X_A)\right)\left(\sum_B F(x,X_B)\right)\left(\sum_C F(x,X_C)\right)$$

(recall our fancy formula)

$$\left(\sum a_i\right)\left(\sum b_j\right) = \sum\sum a_i b_j$$



$p(\mathbf{x}) = F(x,X_A)F(x,X_B)F(x,X_C)$
where each of these three factors are themselves groups of factors over x and the subgraphs.

More explicitly,

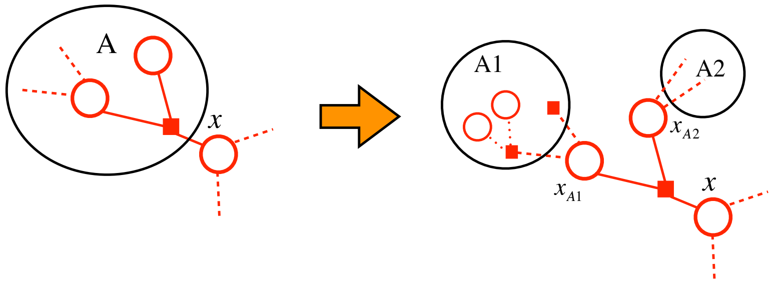
$$F(x,X_A) = \prod_s f(X_s) \quad \text{with } X_s \subseteq \{x\} \cup A$$

$$F(x,X_B) = \prod_s f(X_s) \quad \text{with } X_s \subseteq \{x\} \cup B$$

$$F(x,X_C) = \prod_s f(X_s) \quad \text{with } X_s \subseteq \{x\} \cup C$$

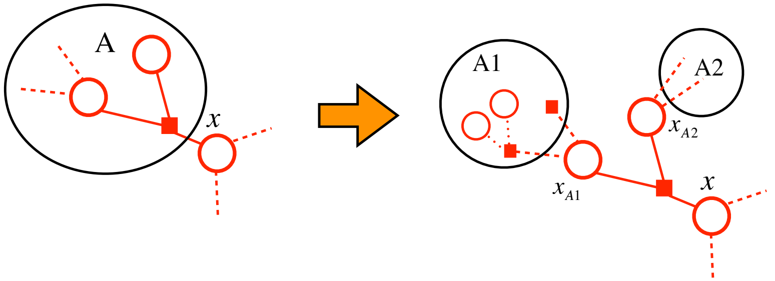
$$p(x) = \sum_{X \backslash \{x\}} p(\mathbf{x}) = \sum_{X \backslash \{x\}} F(x,X_A)F(x,X_B)F(x,X_C) = \left(\sum_A F(x,X_A)\right)\left(\sum_B F(x,X_B)\right)\left(\sum_C F(x,X_C)\right)$$

Consider the first factor



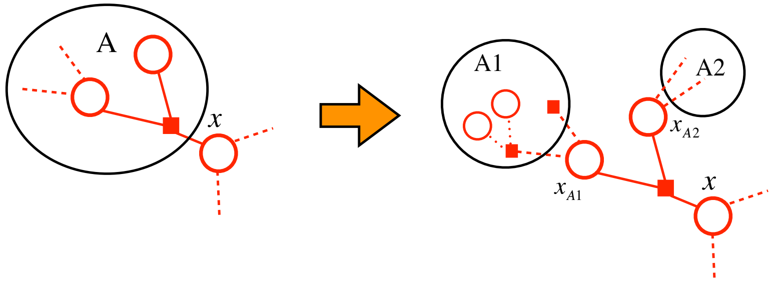
Considering the first factor in the product on the previous slide,

$$\sum_A F(x,X_A) = \sum_A f(x,x_{A1},x_{A2})F_{A1}(x_{A1},A1)F_{A2}(x_{A2},A2)$$
$$= \sum_{x_{A1},x_{A2}} f(x,x_{A1},x_{A2})\sum_{A1} F_{A1}(x_{A1},A1)\sum_{A2} F_{A2}(x_{A2},A2)$$



$$\sum_A F(x,X_A) = \sum_A f(x,x_{A1},x_{A2})F_{A1}(x_{A1},A1)F_{A2}(x_{A2},A2)$$
$$= \sum_{x_{A1},x_{A2}} f(x,x_{A1},x_{A2})\sum_{A1} F_{A1}(x_{A1},A1)\sum_{A2} F_{A2}(x_{A2},A2)$$

To continue the expansion,
consider the first factor



$$\sum_A F(x,X_A) = \sum_A f(x,x_{A1},x_{A2})F_{A1}(x_{A1},A1)F_{A2}(x_{A2},A2)$$
$$= \sum_{x_{A1},x_{A2}} f(x,x_{A1},x_{A2})\sum_{A1} F_{A1}(x_{A1},A1)\sum_{A2} F_{A2}(x_{A2},A2)$$

This factor expands to

$$\sum_{A1} F_{A1}(x_{A1},A1) = \prod_{ne(x_{A1}) \backslash f_{x,A}} F_{A1,i}(x_{A1},A1i)$$

Sum-product algorithm

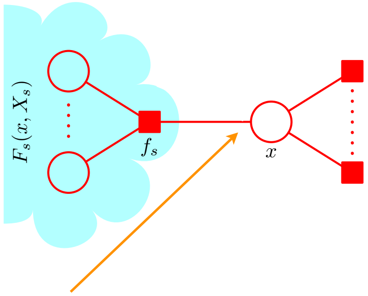
We could continue on recursively until we get to the leaf nodes, thereby computing *p(x)* via recursion.

However, a message passing implementation is simpler, and is better suited to computing all marginals at once.

We defined two kinds of messages:
1) From nodes to factors.
2) From factors to nodes.

Sum-product algorithm

We defined two kinds of messages:
1) From factors to nodes.
2) From nodes to factors.



In analogy with chains, factor-to-node messages provide marginal distributions for the node, for a subgraph. (In the chain case, we had the left side and the right side).

In the chain case we did not have factor nodes. This worked because the second kind of message (nodes-to-factor) is just “pass through” or “copy” in the case of only two links. So, we described it as simply passing messages from node to node.

Marginal distribution for a node *x*

$$p(x) = \sum_{\mathbf{x}/x} \prod_{s\in n(x)} F(x,X_s)$$ (marginalize)

$$= \prod_{s\in n(x)} \left\{ \sum_{X_s} F(x,X_s) \right\}$$ (interchange sums and products)

(recall our fancy formula)

$$\left(\sum a_i\right)\left(\sum b_j\right) = \sum\sum a_i b_j$$

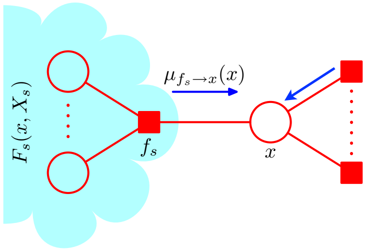
Note that each sum is simpler than what we started with because the variable sets are disjoint except for *x*.

Factor → node messages

$$p(x) = \sum_{\mathbf{x}/x} \prod_{s\in n(x)} F(x,X_s)$$

$$= \prod_{s\in n(x)} \left\{ \sum_{X_s} F(x,X_s) \right\}$$

$$= \prod_{s\in n(x)} \mu_{f_x\to x}(x)$$

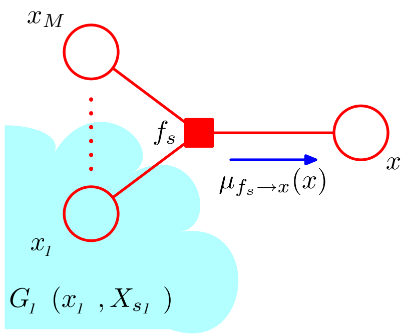


$$\mu_{f_x\to x}(x) \equiv \sum_{X_s} F(x,X_s)$$ (factor-to-node message)

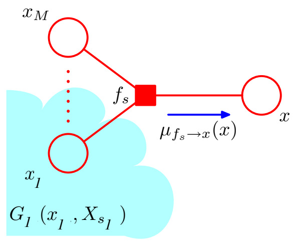
Computing the factor → node messages

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$ (sum removes all variables except x.)

Where $F_s(x,X_s) = f_s(x,x_1,x_2,\ldots,x_M)\underbrace{G_1(x_1,X_{s1})G_2(x_2,X_{s2})\ldots G_M(x_M,X_{sM})}_{\text{Collections of factors in the M sub-graphs}}$



Computing the factor → node messages



$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$ (sum removes all variables except x.)

Where $F_s(x,X_s) = f_s(x,x_1,x_2,\ldots,x_M)G_1(x_1,X_{s1})G_2(x_2,X_{s2})\ldots G_M(x_M,X_{sM})$

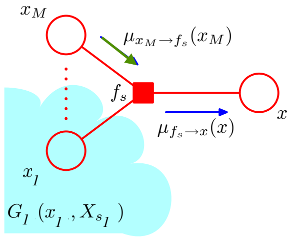
$$\sum_{X_s} F_s(x,X_s) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M} f_s(x,x_1,x_2,\ldots,x_M)\sum_{X_{x1}} G_1(x_1,X_{s1})\sum_{X_{x2}} G_2(x_2,X_{s2})\cdots\sum_{X_{xM}} G_M(x_M,X_{sM})$$

(moving products outside sums where possible)

So, $\sum_{X_s} F_s(x,X_s) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M} f_s(x,x_1,x_2,\ldots,x_M)\prod_{m\in ne(f_s)\backslash x}\ \sum_{X_{xm}} G_m(x_m,X_{sm})$

(interchanging sums and products)

Computing the factor → node messages



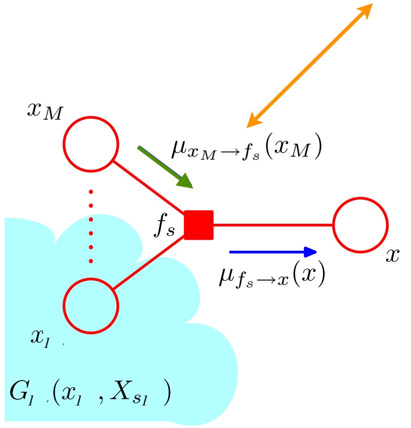
$$\begin{aligned}\sum_{X_s} F_s(x,X_s) &= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M} f_s(x,x_1,x_2,\ldots,x_M)\prod_{m\in ne(f_s)\backslash x}\ \sum_{X_{xm}} G_m(x_m,X_{sm})\\ &= \sum_{x_1}\sum_{x_2}\cdots\sum_{x_M} f_s(x,x_1,x_2,\ldots,x_M)\prod_{m\in ne(f_s)\backslash x}\ \mu_{x_m \to f_s}(x_m)\end{aligned}$$

where we define:

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{xm}} G_m(x_m,X_{sm})$$ (node → factor messages)

Summary of computation for factor → node message

$$\underbrace{\mu_{f_x \to x}(x)}_{\text{factor}\to\text{node}} = \sum_{x_1} \ldots \sum_{x_M} f(x,x_1,\ldots,x_M)\prod_{m\in n(f_s)\backslash x}\underbrace{\mu_{x_m \to f_s}(x_m)}_{\text{node}\to\text{factor}}$$

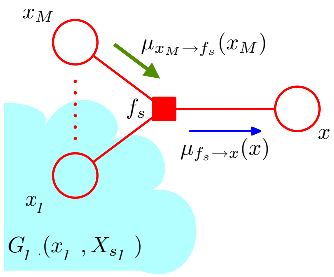


The node → factor message

(We have defined)

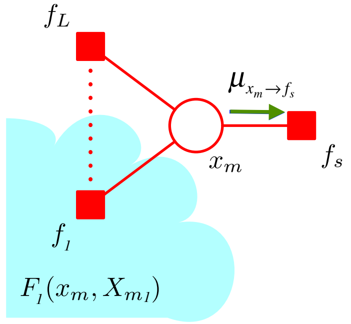
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{s_m}} G_m(x_m, X_{s_m})$$

(For a node x_m we send its distribution with the other variables in the subgraph marginalized out.)



Computing the node → factor message

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



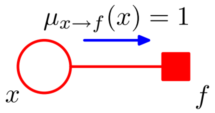
Nodes that only have two links just pass the message through (i.e., in the chain we skipped this part).

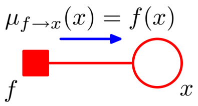
The sum-product algorithm (1)

We could implement what we have just described as recursion, but the local view of nodes getting and passing messages suggests:

Pass messages from leaves to root. If you just want more than one marginal or plan to do other computation, store the results.

Initialization: If leaf node is a variable node, then start with a unity message. If leaf node is factor, then start with the factor.





The sum-product algorithm (2)

We could implement what we have just described as recursion, but the local view of nodes getting and passing messages suggests:

Pass messages from leaves to root. If you just want more than one marginal or plan to do other computation, store the results.

Initialization: If leaf node is a variable node, then start with a unity message. If leave node is factor, then start with the factor.

Note that all needed messages for computation will arrive at each node eventually.

The root node can compute the needed marginal.

The sum-product algorithm (3)

To prepare for other computations (e.g, all marginals), pass messages from the root to the leaves.

Now every node has incoming messages on all its links, and can thus be considered the root.

Hence we can compute all marginals for twice the cost of computing one of them.

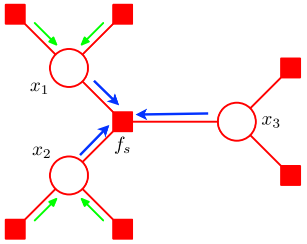
(From before, also note that all messages that can be passed, have now been passed).

The sum-product algorithm (4)

Another easy computation is the marginal for the group of variables in a factor.

Intuitively (and easily shown---homework) this is given by:

$$p(\mathbf{x}_s) = f(\mathbf{x}_s) \prod_{i \in n(f_s)} \mu_{x_i \to f_s}(x_i)$$



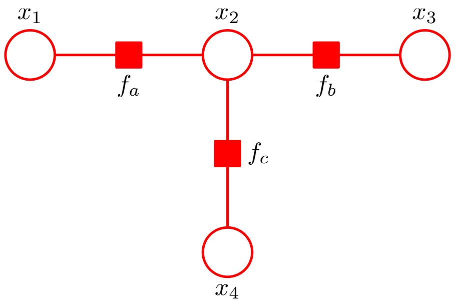
The sum-product algorithm (5)

If the factor graph came from a directed graph, then the expression for p(x) is already normalized.

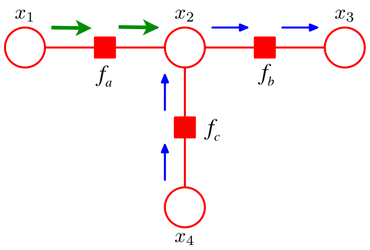
Otherwise (as was the case of the chain), we can determine the normalization constant by summing up one of the marginals (relatively inexpensive because only one variable is involved).

Sum-product algorithm example

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



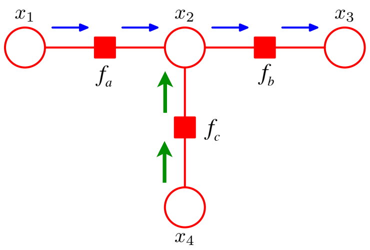
Declare x_3 as root node.



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

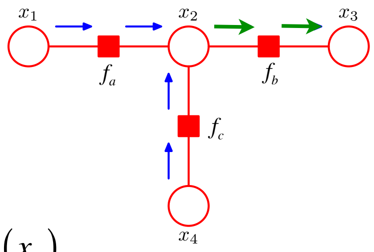
$$\mu_{f_a\to x_2}(x_2)=\sum_{x_1} f_a(x_1,x_2)$$

$$\underbrace{\mu_{f_x\to x}(x)}_{\text{factor}\to\text{node}}=\sum_{x_1}\dots\sum_{x_M} f(x,x_1,\dots,x_M)\prod_{m\in n(f_s)\backslash x}\underbrace{\mu_{x_m\to f_s}(x_m)}_{\text{node}\to\text{factor}}$$



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

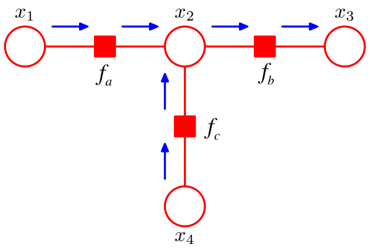
$$\mu_{f_c\to x_2}(x_2)=\sum_{x_4} f_c(x_2,x_4)$$



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

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

Summary of messages
from leaves to root



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

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

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

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

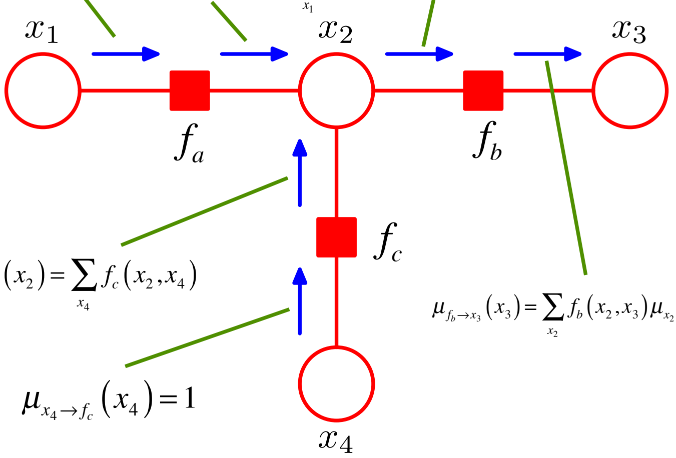
$$\mu_{x_2\to f_b}(x_2)=\mu_{f_a\to x_2}(x_2)\mu_{f_c\to x_2}(x_2)$$

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

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

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

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

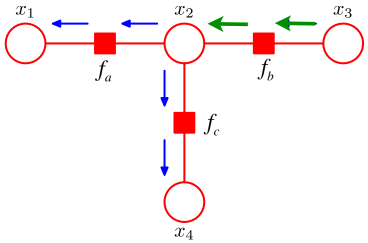


$$\mu_{f_c\to x_2}(x_2)=\sum_{x_4} f_c(x_2,x_4)$$

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

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

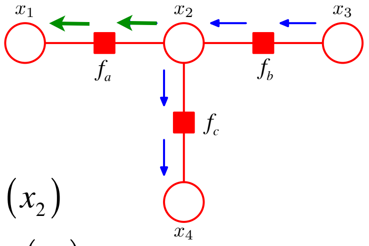
Next we pass messages from root to leaves.



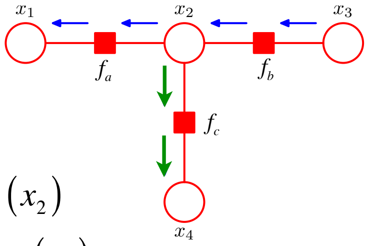
$$\mu_{x_3\to f_b}(x_3)=1$$
$$\mu_{f_b\to x_2}(x_2)=\sum_{x_3} f_b(x_2,x_3)$$

Candidate for third and fourth?

Lets go towards x_1 first.



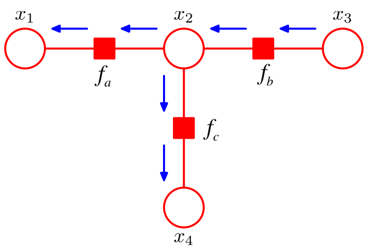
$$\mu_{x_2\to f_a}(x_2)=\mu_{f_b\to x_2}(x_2)\mu_{f_c\to x_2}(x_2)$$
$$\mu_{f_a\to x_1}(x_1)=\sum_{x_2} f_a(x_1,x_2)\mu_{x_2\to f_a}(x_2)$$



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

(similar to previous one)

Summary of messages from root to leaves.



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

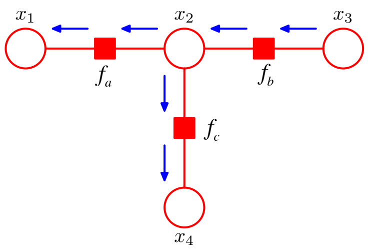
$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

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

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

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

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



An illustrative check

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

Handling observed variables

Usually we have observed variables (e.g., evidence). We simply clamp those variables to their observed values.

More formally, denote hidden variables by $\mathbf{h}$, and observed ones by $\mathbf{v}$. Denote the observed value as $\hat{\mathbf{v}}$. For each observed variable, v_i, with value $\hat{v}_i$, we can introduce factors into the graph

$$I(v_i, \hat{v}_i) = \begin{cases} 1 & \text{if } v_i = \hat{v}_i \\ 0 & \text{otherwise} \end{cases}$$

Then, $p(\mathbf{h}, \mathbf{v} = \hat{\mathbf{v}}) = p(\mathbf{h}, \mathbf{v}) \prod_i I(v_i, \hat{v}_i)$

(needs to be normalized to get $p(\mathbf{h}|\hat{\mathbf{v}})$, but this is easy since we are doing sum-product.)

Adds factor nodes, i.e., 🟥

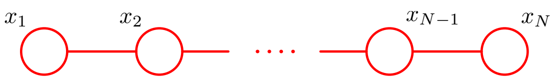
Max-sum algorithm

Method to compute.

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

$$i.e.,\ \ p\left(\mathbf{x}^{\max}\right) = \max_{\mathbf{x}} p(\mathbf{x})$$

Recall inference on chains



$$p(\mathrm{x}) = \psi_{1,2}(x_1,x_2)\psi_{2,3}(x_2,x_3) \ \ \ldots \ \ \psi_{N-2,N-1}(x_{N-2},x_{N-1})\psi_{N-1,N}(x_{N-1},x_N)$$

Naive compution of $\arg\max_{\mathbf{x}} p(\mathbf{x})$

would evauate the above for each value of x,

and take the max.

Too expensive!!

Recall speeding up marginalization

$$p(x_n) = \left(\sum_{x_{n-1}} \sum_{x_{n-2}} \cdots \sum_{x_1} \prod_{i=1}^{n-1} \psi_{n-i,\,n-i+1}(x_{n-i},x_{n-i+1}) \right) \left(\sum_{x_{n+1}} \cdots \sum_{x_{N-1}} \sum_{x_N} \prod_{i=n}^{N-1} \psi_{i,\,i+1}(x_i,x_{i+1}) \right)$$

$$\sum_{x_{n-1}} \sum_{x_{n-2}} \cdots \sum_{x_1} \prod_{i=1}^{n-1} \psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1}) = \left\{ \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1},x_n) \ldots \left\{ \sum_{x_3} \psi_{3,4}(x_3,x_4) \left\{ \sum_{x_2} \psi_{2,3}(x_2,x_3) \left\{ \sum_{x_1} \psi_{1,2}(x_1,x_2) \right\} \right\} \right\} \ldots \right\}$$

and

$$\sum_{x_{n+1}} \cdots \sum_{x_{N-1}} \sum_{x_N} \prod_{i=n}^{N-1} \psi_{i,\,i+1}(x_i,x_{i+1}) = \left\{ \sum_{x_{n+1}} \psi_{n,n+1}(x_n,x_{n+1}) \ldots \left\{ \sum_{x_{N-1}} \psi_{N-2,N-1}(x_{N-2},x_{N-1}) \left\{ \sum_{x_N} \psi_{N-1,\,N}(x_{N-1},x_N) \right\} \right\} \ldots \right\}$$

What if we could do with max() what we are doing with $\sum$?

Helpful facts

First. note that.

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \ \max_{x_2} \ \ldots \ \max_{x_M} p(\mathbf{x})$$

Second, note that.

$$\max(ab,ac) = a \ \max(b,c) \qquad \text{(for a} \geq 0)$$

Max on a chain

In analogy with marginals on a chain,

$$\max_{\mathbf{x}} p(\mathbf{x}) = \frac{1}{Z} \max_{x_n} \left\{ \left[\max_{x_{n-1}} \max_{x_{n-2}} \ldots \max_{x_1} \prod_{i=1}^{n-1} \psi_{n-i,\,n-i+1}(x_{n-i},x_{n-i+1}) \right]_{x_n} \bullet \left[\max_{x_{n+1}} \max_{x_{n+2}} \ldots \max_{x_N} \prod_{i=n}^{N-1} \psi_{i,\,i+1}(x_i,x_{i+1}) \right] \right\}$$

where,

$$\max_{x_{n-1}} \max_{x_{n-2}} \ldots \max_{x_1} \left(\prod_{i=1}^{n-1} \psi_{n-i,n-i+1}(x_{n-i},x_{n-i+1}) \right) =$$

$$\max_{x_{n-1}} \left(\psi_{n-1,n}(x_{n-1},x_n) \bullet \max \left(\ldots \max_{x_3} \left(\psi_{3,4}(x_3,x_4) \bullet \max_{x_2} \left(\psi_{2,3}(x_2,x_3) \bullet \max_{x_1} \left(\psi_{1,2}(x_1,x_2) \right) \right) \right) \right) \right)$$

and similarly for the part in red.

Max-sum algorithm

Two steps.

1) Compute the max while remembering certain computations
2) Compute a value of **x** that achieves the max

The message passing algorithm for step (1) is clear from the analog with the “sum-product” algorithm, except that it would then be called the “max-product” algorithm.

Computing long products looses precision (*), so we switch to log(), and call it the max-sum algorithm.

(*) Less of an issue with marginalization.

Max-sum algorithm (continued)

Note that

$$\ln\left(\max_{\mathbf{x}}\left(p\left(x\right)\right)\right)=\max_{\mathbf{x}}\left(\ln\left(p\left(x\right)\right)\right)$$

And we have

$$\begin{aligned}\ln\left(\max\left(ab,ac\right)\right)&=\max\left(\log\left(ab\right),\log\left(ac\right)\right)\\&=\max\left(\ln\left(a\right)+\ln\left(b\right),\ln\left(a\right)+\ln\left(c\right)\right)\\&=\ln\left(a\right)+\max\left(\ln\left(b\right)+\ln\left(c\right)\right)\end{aligned}$$

(In general, $\max\left(x+y,x+z\right)=x+\max(y,z)$)

Can also get this from taking logs of product version,
namely: $\max\left(ab,ac\right)=a\cdot\max\left(b,c\right)$, for $a\geq 0$)

Max-sum algorithm

Working now in analogy with the sum-product algorithm (*)

$$\mu_{f\to x}\left(x\right)=\max_{x_1,x_2,\,\ldots,\,x_M}\left[\ln f\left(x,\,x_1,\,x_2,\,\ldots,\,x_M\right)+\sum_{m\in n\left(f_s\right)\backslash x}\mu_{x_m\to f}\left(x_m\right)\right]$$

and

$$\mu_{x\to f}\left(x\right)=\sum_{l\in n(x)\backslash f}\mu_{f\to x_l}\left(x\right)$$

*Recall that in sum-product: $\underbrace{\mu_{f_x\to x}\left(x\right)}_{\text{factor}\to\text{node}}=\sum_{x_1}\ldots\sum_{x_M}f\left(x,x_1,\,\ldots\,,\,x_M\right)\prod_{m\in n\left(f_s\right)\backslash x}\underbrace{\mu_{x_m\to f_s}\left(x_m\right)}_{\text{node}\to\text{factor}}$

Max-sum algorithm

Working now in analogy with the sum-product algorithm

For initialization at leaf nodes

$$\mu_{f\to x}\left(x\right)=0$$

and

$$\mu_{x\to f}\left(x\right)=\ln\left(f\left(x\right)\right)$$

Max-sum algorithm

Working now in analogy with the sum-product algorithm

To compute the max using the choosen root node,

$$\ln\left(\mathrm{p}^{\max}\right) = \max_{\mathbf{x}}\left[\sum_{s\in n(x)} \mu_{s\to x_s}(x)\right]$$

Max-sum algorithm

Now we need to find an **x** where *p()* reaches the max.

This does not have an exact analogy in the sum-product algorithm.

Why we do not know x yet:

The factor-to-node messages takes a distribution for the maxima over the upstream variables, and multiplies it by the factor (sum using logs), and reports a new distribution.

We do not yet know which value in the new distribution will be part of the maximum (it is not necessarily argmax of the reported distribution).

Max-sum algorithm

Can passing messages backwards find the arg max?

At the root node, which is a product (sum in logs), when we find the maximum, we can easily record the argmax for that node's variable, and it will be a valid for a particular maximizing configuration.

In analogy with sum-product, we might be tempted to send messages backwards to "finish the job" to get values for the other nodes.

But this can fail if there is more than one maximal configuration.
You can get pieces of each one!
We are only sure that the value is part of **some** maximal configuration.
You could end up with an inconsistent set of values.

Max-sum algorithm

Adjustment to forward message passing for "backtracking:"

The factor-to-node operations store the dependencies for the various choices of x_i .

Then, once the node-to-factor backtracking expresses a choice, a consistent set of values for x_i for the max can be found.

For example, suppose the root node could choose either setting its variable to 2 or 3, 2. Then it sends to the incoming nodes, the value "2". Those nodes need to know how their incoming links have maximized to get the value for "2" passed to the root.

Max-sum algorithm (back-tracking)

In more detail, when we compute

$$\mu_{f\to x}(x) = \max_{x_1, x_2, \ldots, x_M} \left[\ln\left(f(x, x_1, x_2, \ldots, x_M) + \sum_{m \in n(f_s)\backslash x} \mu_{x_m \to f}(x_m) \right) \right]$$

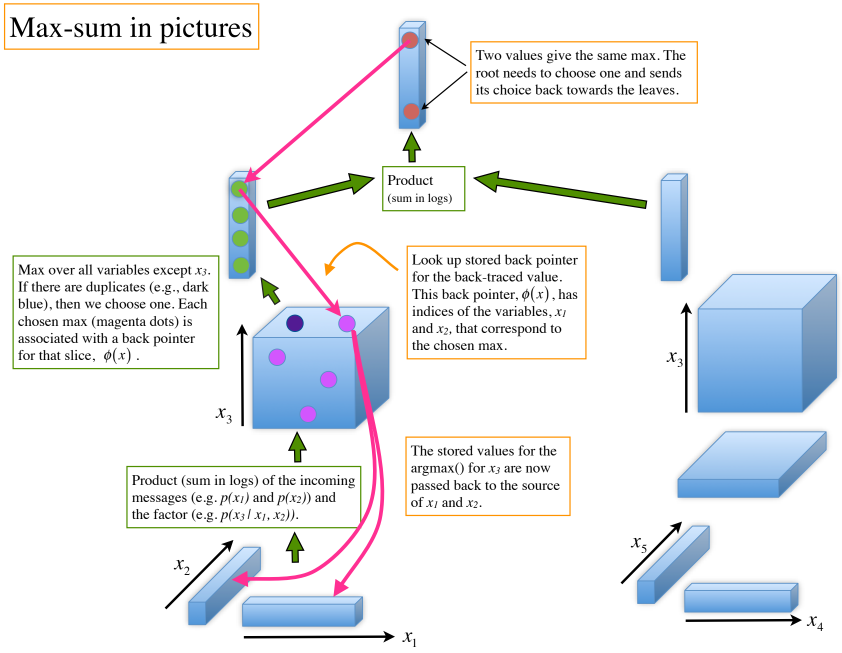
store

$$\phi(x) = \underset{x_1, x_2, \ldots, x_M}{\arg\max} \left[\ln f(x, x_1, x_2, \ldots, x_M) + \sum_{m \in n(f_s)\backslash x} \mu_{x_m \to f}(x_m) \right]$$

(This records the downstream choices for any upstream choice of x)

Then, once we know the overall max, we can recover a set of x_i that leads to it by backtracking.

Max-sum in pictures



Two values give the same max. The root needs to choose one and sends its choice back towards the leaves.

Product
(sum in logs)

Max over all variables except x_3. If there are duplicates (e.g., dark blue), then we choose one. Each chosen max (magenta dots) is associated with a back pointer for that slice, $\phi(x)$.

Look up stored back pointer for the back-traced value. This back pointer, $\phi(x)$, has indices of the variables, x_1 and x_2, that correspond to the chosen max.

Product (sum in logs) of the incoming messages (e.g. $p(x_1)$ and $p(x_2)$) and the factor (e.g. $p(x_3 \mid x_1, x_2)$).

The stored values for the argmax() for x_3 are now passed back to the source of x_1 and x_2.