



## Marginals on a chain

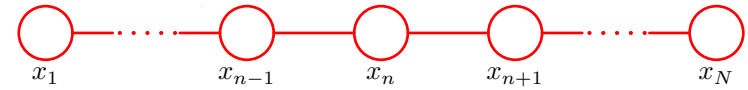
Recall   
 $p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2) \cdots p(x_{N-1}|x_{N-2})p(x_N|x_{N-1})$

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

Assume N discrete variables, with K values each.

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

## Marginals on a chain

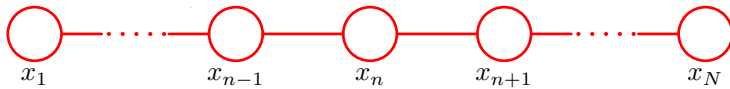


$$p(\mathbf{x}) = \psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3) \cdots \psi_{N-2,N-1}(x_{N-2}, x_{N-1})\psi_{N-1,N}(x_{N-1}, x_N)$$

Direct calculation of  $p(x_n)$

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

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



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

Main idea is to rearrange terms to exploit conditional independence.

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

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

and

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

## Computational Complexity

Suppose each variable has K values

What is the cost of evaluating the first factor?

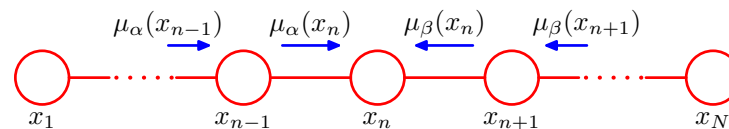
$$\sum_{x_{n+1}} \sum_{x_{n+2}} \dots \sum_{x_1} \prod_{i=1}^{n-1} \psi_{n-i, n-i+1}(x_{n-i}, x_{n-i+1}) = \left\{ \sum_{x_{n+1}} \psi_{n-1, n}(x_{n-1}, x_n) \dots \sum_{x_3} \psi_{3,4}(x_3, x_4) \left\{ \sum_{x_2} \psi_{2,3}(x_2, x_3) \left\{ \underbrace{\sum_{x_1} \psi_{1,2}(x_1, x_2)}_{\substack{\text{K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

The other factor is similar.

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

Much better than the naive computation where we had  $O(K^N)$ .

## Message passing interpretation

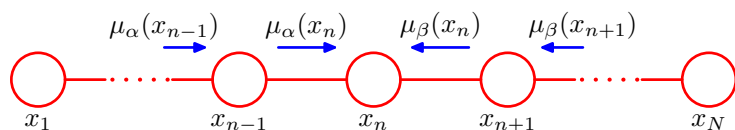


Define  $\mu_\alpha(x_n)$  as a message passed from node  $x_{n-1}$  to node  $x_n$ .

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

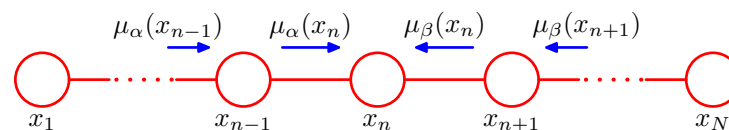
The messages will correspond to the computations which we see take and input message, and computes an output message using the potential.

## Message passing interpretation



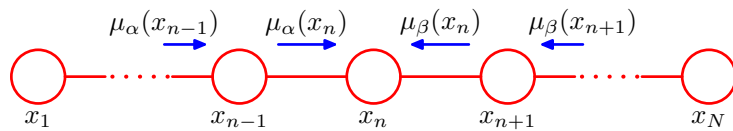
$$\sum_{x_{n+1}} \sum_{x_{n+2}} \dots \sum_{x_1} \prod_{i=1}^{n-1} \psi_{n-i, n-i+1}(x_{n-i}, x_{n-i+1}) = \left\{ \sum_{x_{n+1}} \psi_{n-1, n}(x_{n-1}, x_n) \dots \sum_{x_3} \psi_{3,4}(x_3, x_4) \left\{ \sum_{x_2} \psi_{2,3}(x_2, x_3) \left\{ \underbrace{\sum_{x_1} \psi_{1,2}(x_1, x_2)}_{\mu_\alpha(x_2)}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

## Message passing interpretation



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

## Message passing interpretation

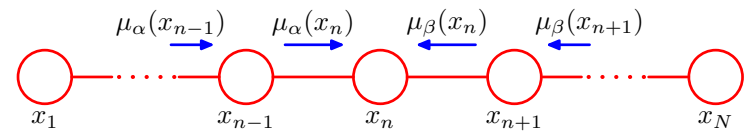


$$p(x_n) = \frac{1}{Z} \mu_a(x_n) \mu_b(x_n)$$

### 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



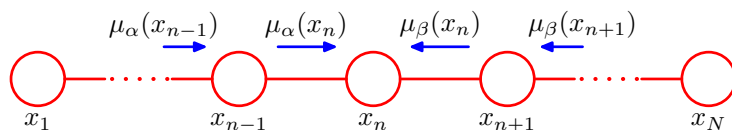
To compute all marginals, send a message from left to right, and right to left, storing the result.

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.

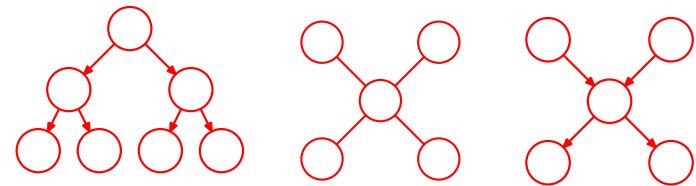
EG, if  $x_n$  is observed, then we do not need to marginalize over it.

## Trees

Directed graph. Root node has no parents, others have exactly one parent.

Undirected graph. Only one path between any pair of nodes.

A directed graph with only one path per pair is a polytree.



## Factor Graphs

Suppose  $p(\mathbf{x})$  factorizes as:

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

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,  $s$ .

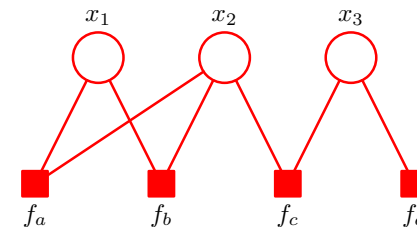
Note that the factorization formula means that we can convert **both** directed and undirected graphs to factor graphs.

## Factor Graph Example

Suppose  $p(\mathbf{x})$  factorizes as:

$$p(\mathbf{x}) = \prod_s f(x_s) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_4)$$

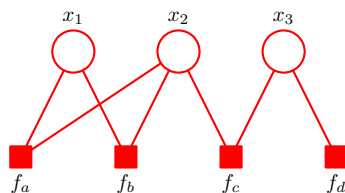
The graph is:



## Factor Graph Example (continued)

Suppose  $p(\mathbf{x})$  factorizes as:

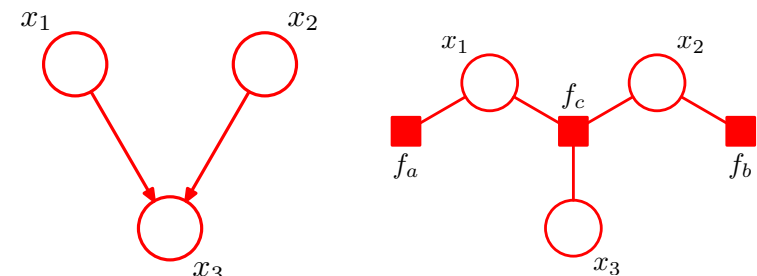
$$p(\mathbf{x}) = \prod_s f(x_s) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_4)$$



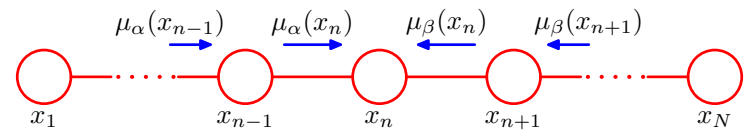
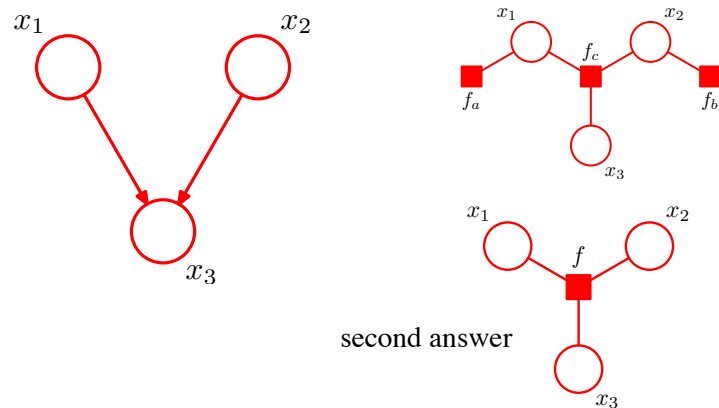
This layout emphasizes that factor graphs are *bipartite*.

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

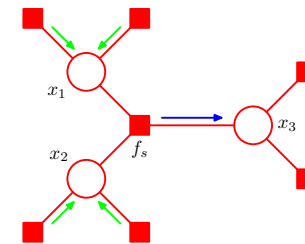
## Factor Graph Example (2)



## Factor Graph Example (2)



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



## Observations about factor graphs

Any node can be root

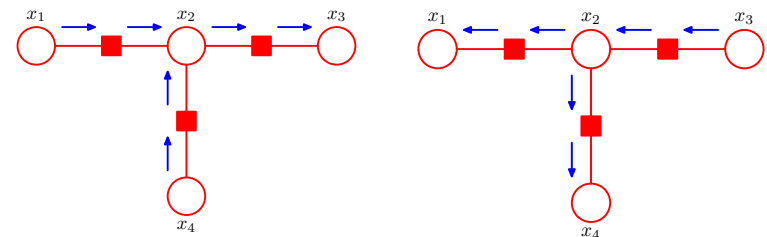
Any node with  $N$  links splits the graph into  $N$  subgraphs which do not share nodes.

Having passed messages from:

- 1) the leaves to a chosen root;
  - 2) the chosen root to the leaves,
- 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



## 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,  $x_n$ .

We defined two kinds of messages:

- 1) From nodes to factors.
- 2) From factors to nodes.

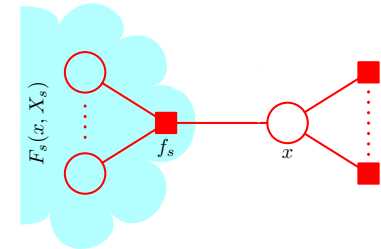
## Factor to node messages

The node  $x$  with  $N$  neighbors divides the graph into  $N$  subgraphs.

Define  $F(x, X_s)$  as the product of all factors involving  $x$  and nodes in the subgraph,  $X_s$ .

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

(joint distribution)



## Factor --> node messages

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

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

(recall our fancy formula)

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

## Factor --> node messages

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

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

(define)

$$\mu_{f_x \rightarrow x}(x) \equiv \sum_{X_s} F(x, X_s)$$