

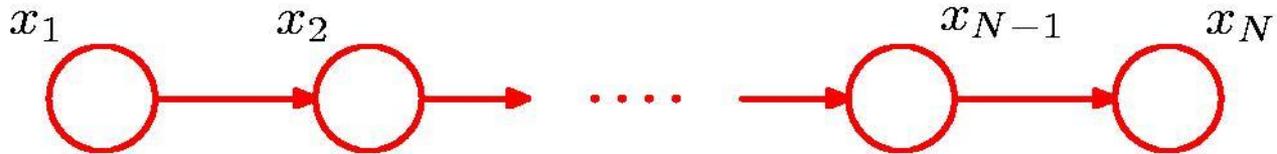
Part II

C. M. Bishop

**PATTERN RECOGNITION
AND MACHINE LEARNING**

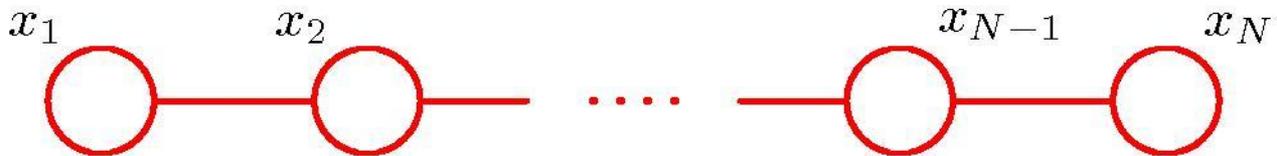
CHAPTER 8: GRAPHICAL MODELS

Converting Directed to Undirected Graphs (1)



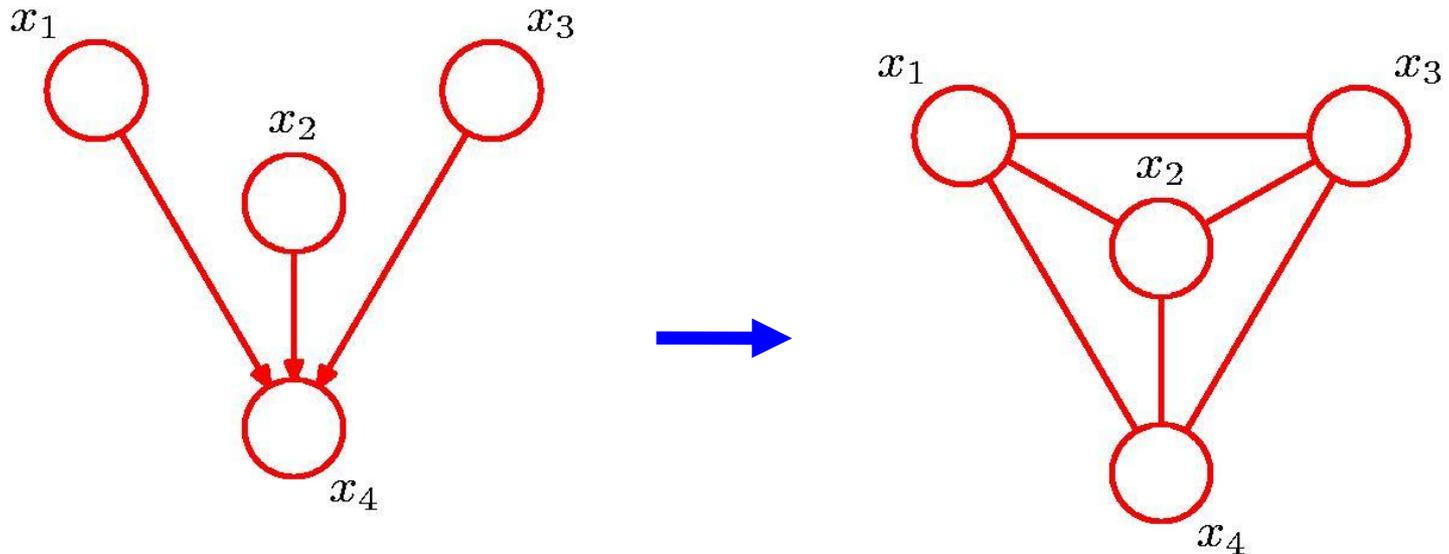
$$p(\mathbf{x}) = p(x_1) p(x_2|x_1) p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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



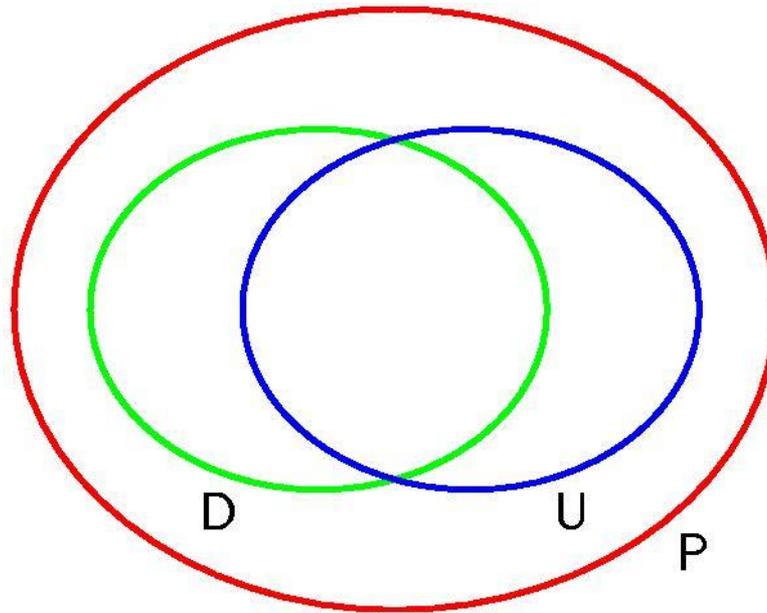
Converting Directed to Undirected Graphs (2)

- Add extra links between all pairs of parents : moralization – “marrying the parents”.
- The resulting undirected graph, after dropping the arrows, is called the moral graph.



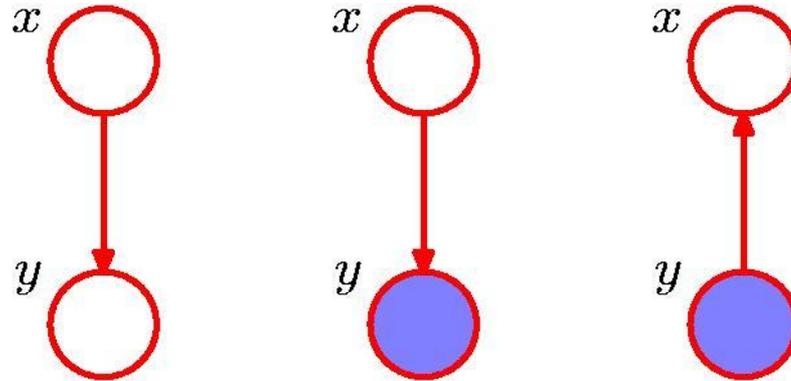
$$\begin{aligned} p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\ &= \frac{1}{Z} \psi_A(x_1, x_2, x_3) \psi_B(x_2, x_3, x_4) \psi_C(x_1, x_2, x_4) \end{aligned}$$

Directed vs. Undirected Graphs



Inference in Graphical Models

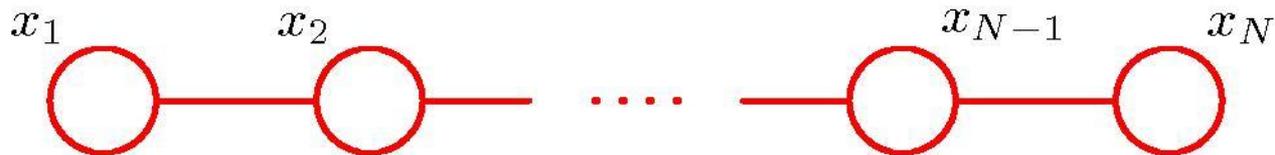
A graphical representation of Bayes' theorem.



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

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

Inference on a Chain



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

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



For N variables, each with K states, there are KN values for x , thus evaluation and storage of the joint distribution, as well as marginalization to obtain $p(x_n)$, all involve storage and computation that scale exponentially with the length N of the chain.

Efficient Computation(1)

- Rearrange the order of the summations and the multiplications allows the required marginal to be evaluated much more efficiently.

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

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_N} \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)$$

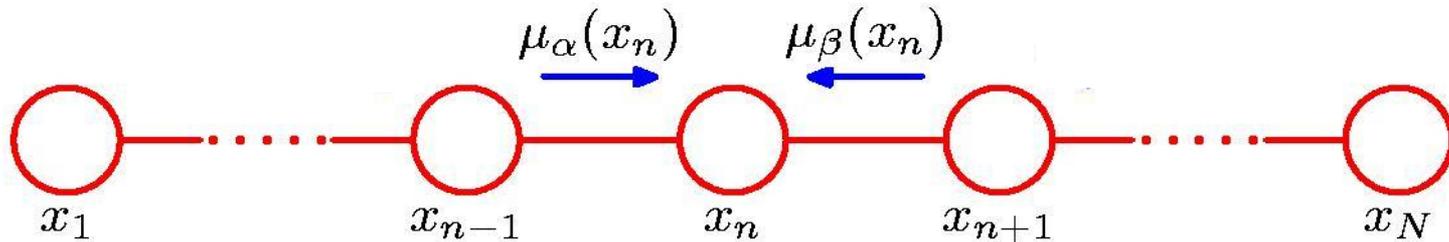
Doesn't depend on x_N

Efficient Computation(2)

$$p(x_n) = \frac{1}{Z} \left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \cdots \right]$$
$$\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]$$

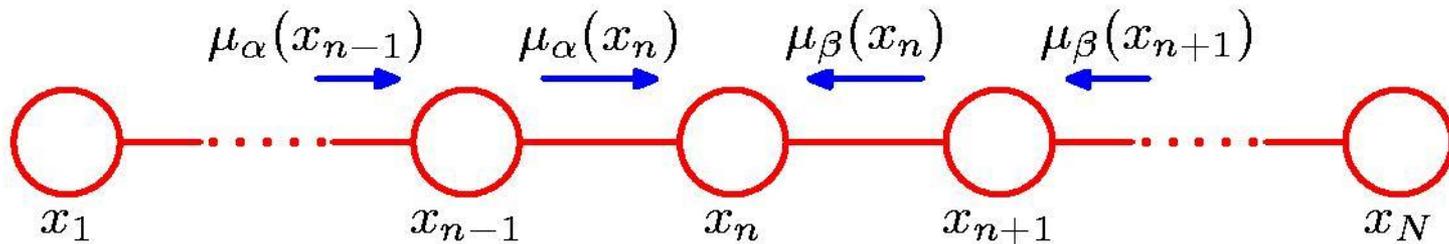
Evaluating the marginal is $O(NK^2)$. This is linear in the length of the chain, in contrast to the exponential cost of a naive approach.

Inference on a Chain



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

Inference on a Chain



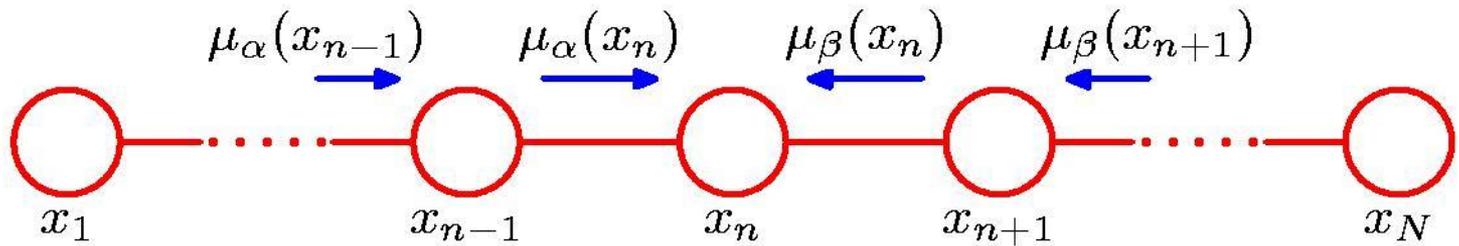
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$

$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}).$$

$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right]$$

$$= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_\beta(x_{n+1}).$$

Inference on a Chain



$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$$

$$\mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

$$Z = \sum_{x_n} \mu_\alpha(x_n) \mu_\beta(x_n)$$

Inference on a Chain

To compute local marginals:

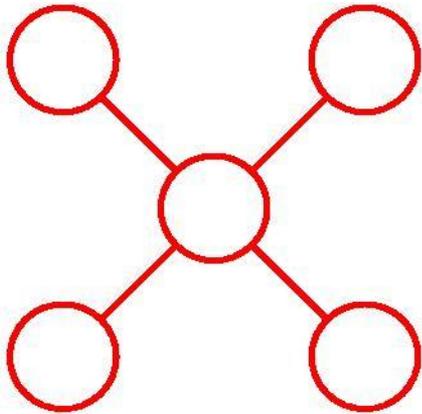
- Compute and store all forward messages, $\mu_\alpha(x_n)$.
- Compute and store all backward messages, $\mu_\beta(x_n)$.
- Compute Z at any node x_m
- Compute

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

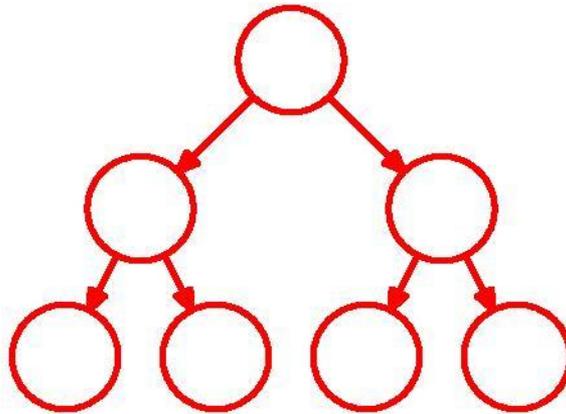
for all variables required.

Trees

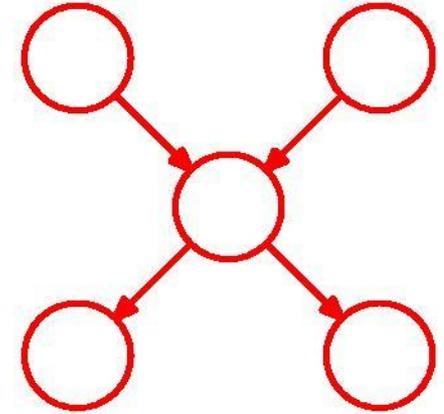
Undirected Tree



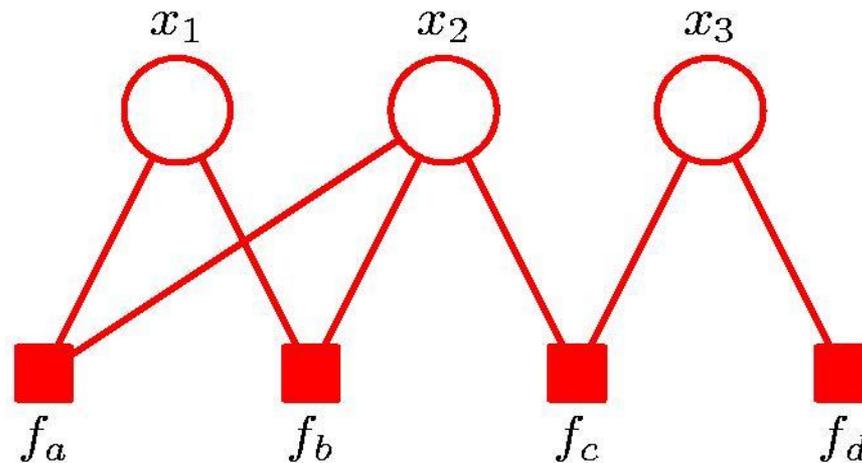
Directed Tree



Polytree



Factor Graphs



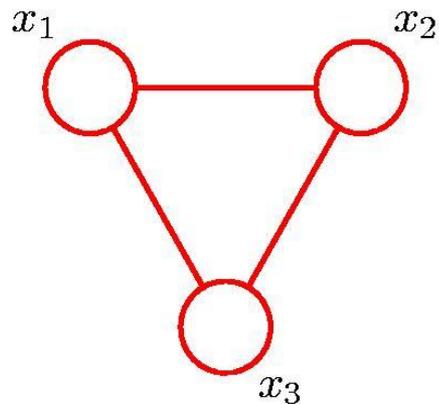
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

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

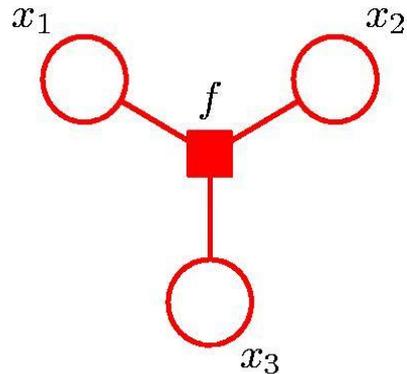
Factors in directed graphs are local conditional distributions.

Factors in undirected graphs are potential functions over the maximal cliques (the normalizing coefficient $1/Z$ can be viewed as a factor defined over the empty set of variables).

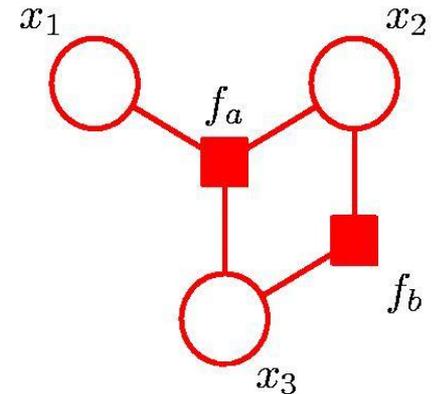
Factor Graphs from Undirected Graphs



$$\psi(x_1, x_2, x_3)$$



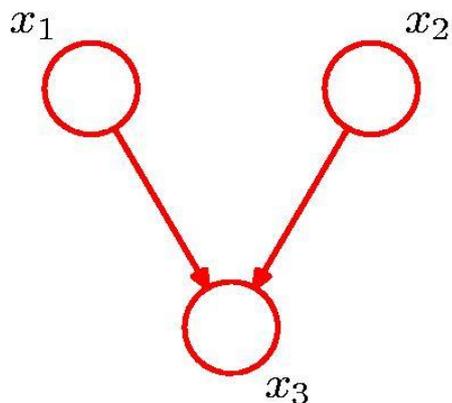
$$\begin{aligned} f(x_1, x_2, x_3) \\ = \psi(x_1, x_2, x_3) \end{aligned}$$



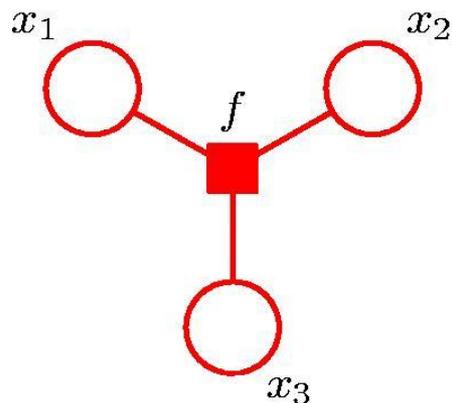
$$\begin{aligned} f_a(x_1, x_2, x_3) f_b(x_2, x_3) \\ = \psi(x_1, x_2, x_3) \end{aligned}$$

- ▶ Create variable nodes corresponding to the nodes in the original undirected graph
 - ▶ Create additional factor nodes corresponding to the maximal cliques x_s .
 - ▶ The factors $f_s(x_s)$ are then set equal to the clique potentials.
-

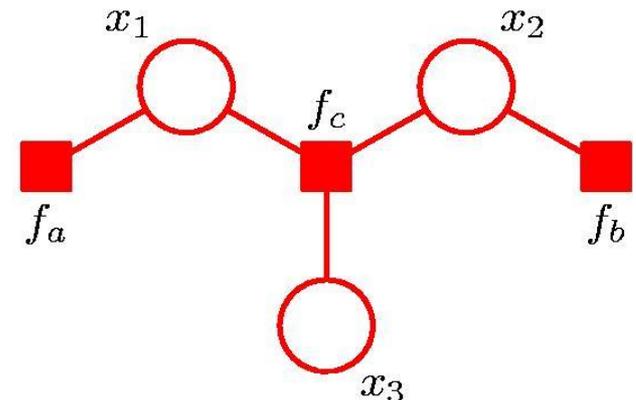
Factor Graphs from Directed Graphs



$$p(\mathbf{x}) = p(x_1)p(x_2) \\ p(x_3|x_1, x_2)$$



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



$$f_a(x_1) = p(x_1)$$

$$f_b(x_2) = p(x_2)$$

$$f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2)$$

- ▶ Create variable nodes in the factor graph corresponding to the nodes of the directed graph.
- ▶ Create factor nodes corresponding to the conditional distributions.
- ▶ Add the appropriate links.

The Sum-Product Algorithm (1)

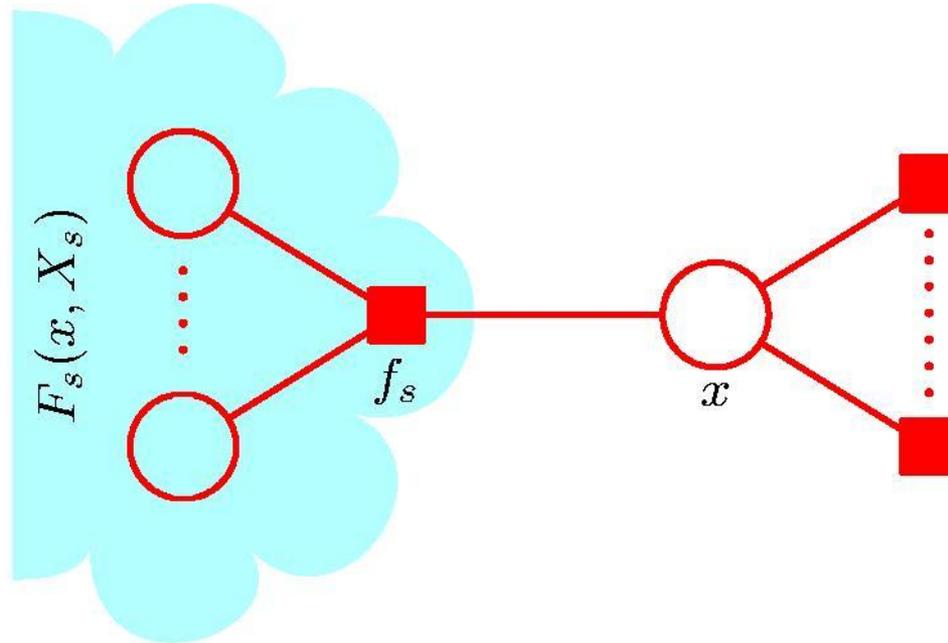
Objective:

- i. to obtain an efficient, exact inference algorithm for finding marginals;
- ii. in situations where several marginals are required, to allow computations to be shared efficiently.

Key idea: Distributive Law

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

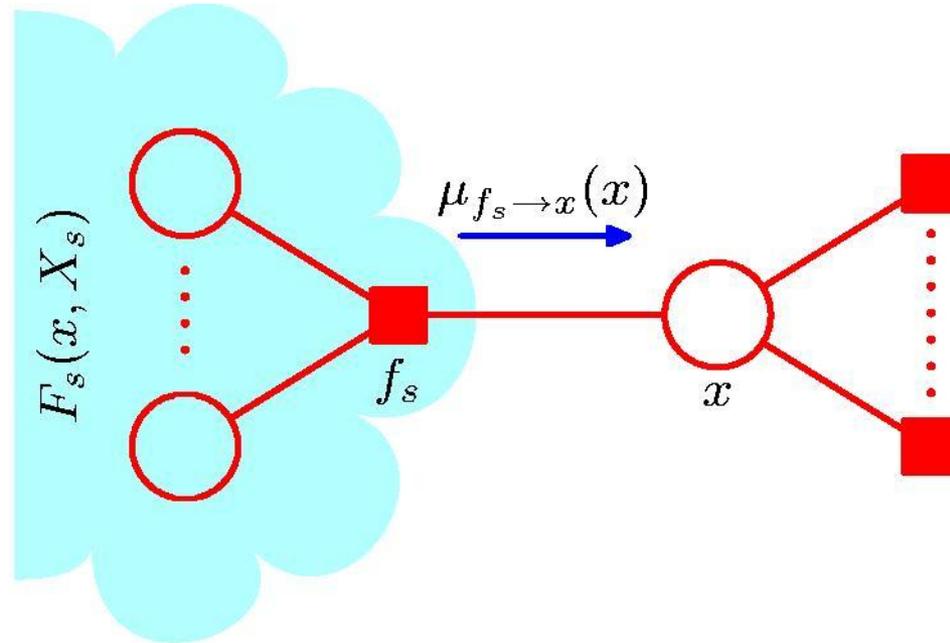
The Sum-Product Algorithm (2)



$$p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$$

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

The Sum-Product Algorithm (3)

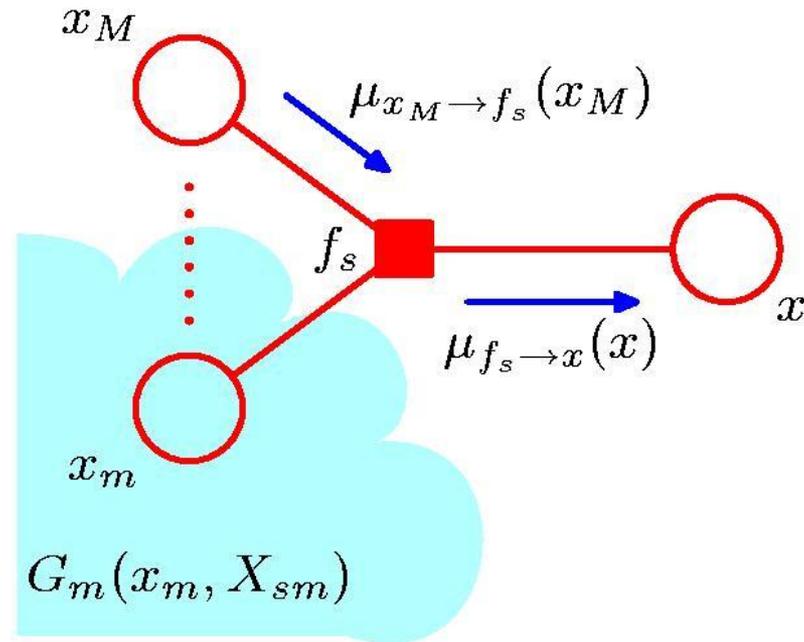


$$p(x) = \prod_{s \in \text{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right]$$

$$= \prod_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x).$$

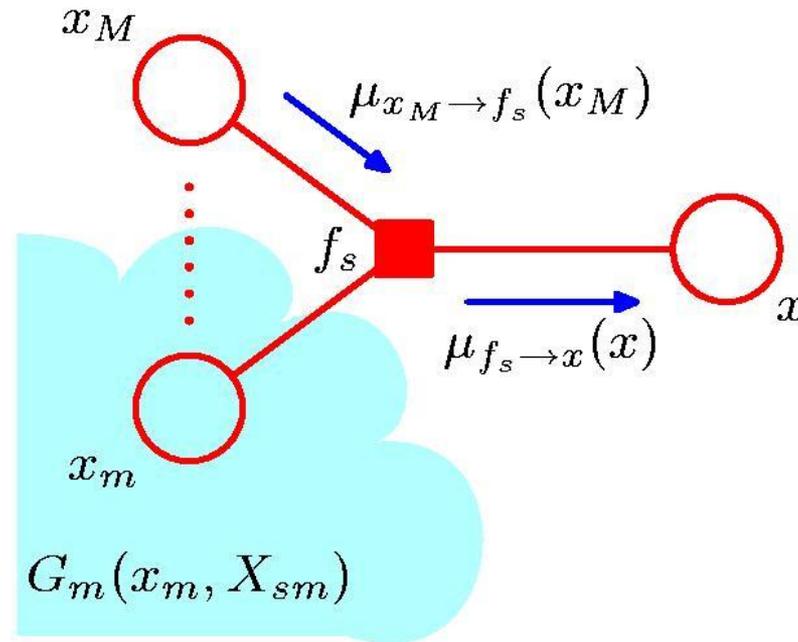
$$\mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

The Sum-Product Algorithm (4)



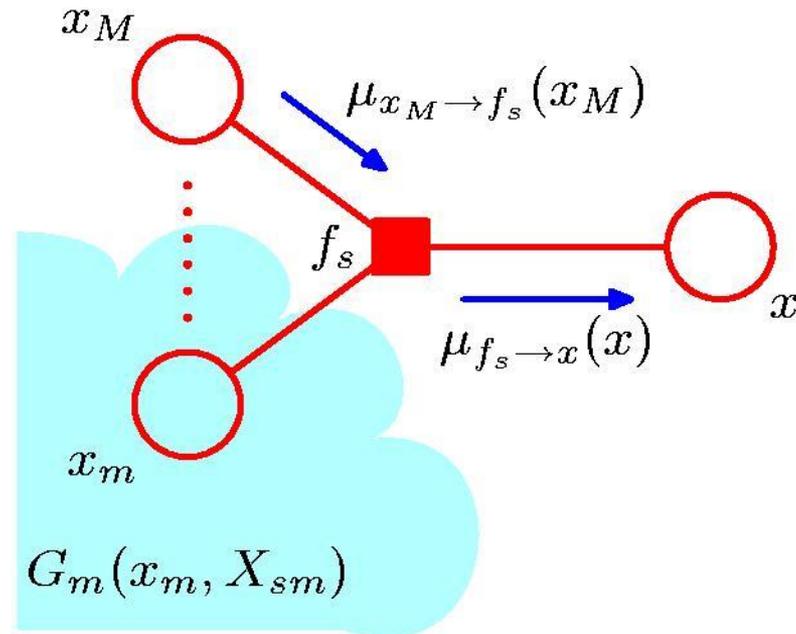
$$F_s(x, X_s) = f_s(x, x_1, \dots, x_M) G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM})$$

The Sum-Product Algorithm (5)



$$\begin{aligned} \mu_{f_s \rightarrow x}(x) &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\ &= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f_s}(x_m) \end{aligned}$$

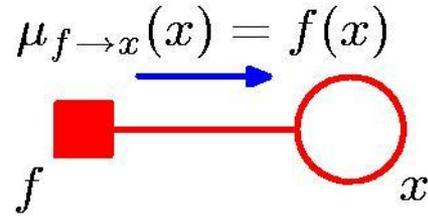
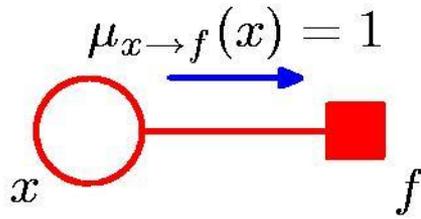
The Sum-Product Algorithm (6)



$$\begin{aligned} \mu_{x_m \rightarrow f_s}(x_m) &\equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \text{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml}) \\ &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m) \end{aligned}$$

The Sum-Product Algorithm (7)

Initialization

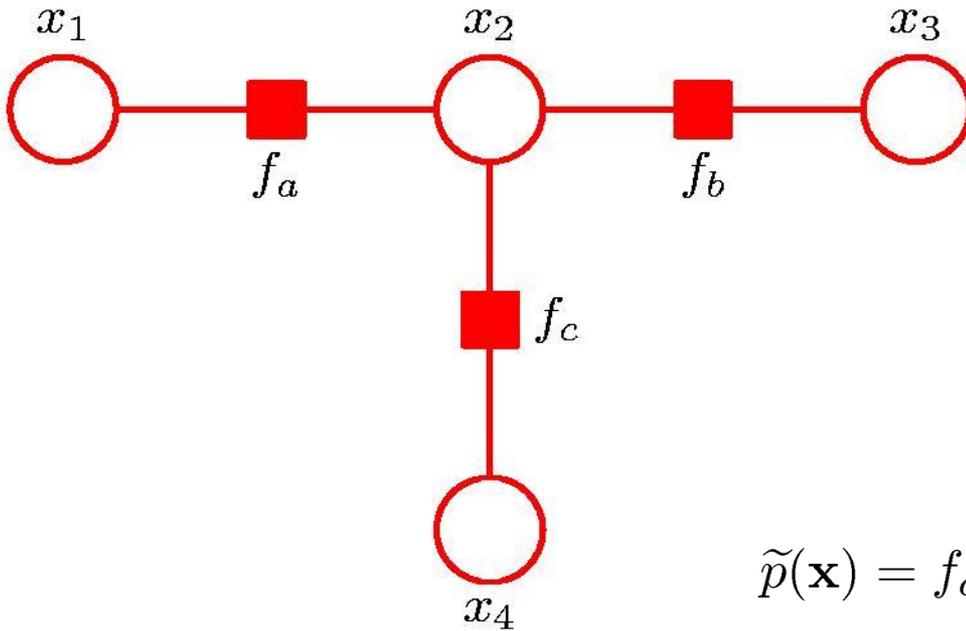


The Sum-Product Algorithm (8)

To compute local marginals:

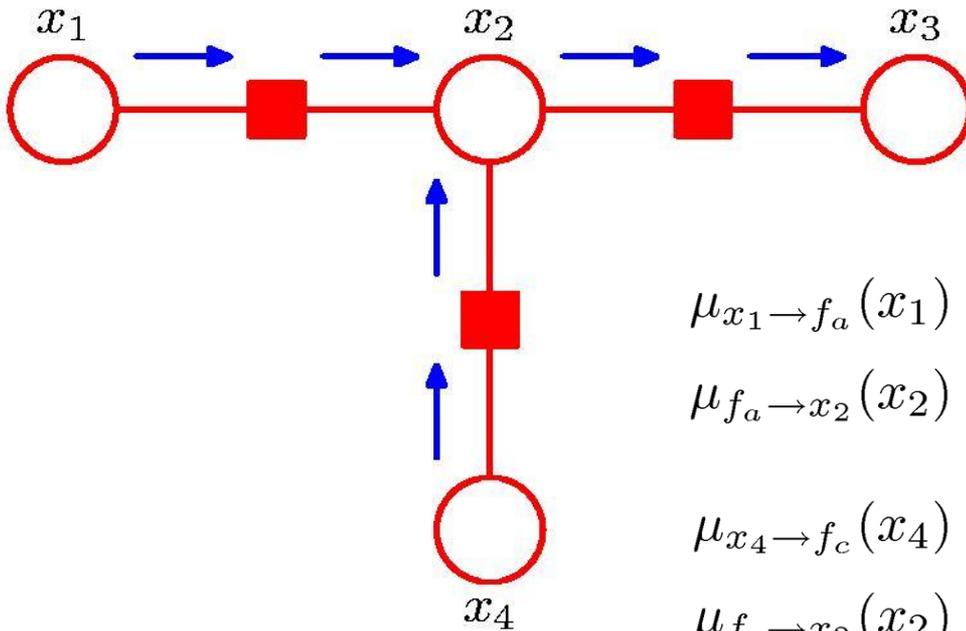
- Pick an arbitrary node as root
 - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
 - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
 - Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.
-

Sum-Product: Example (1)



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

Sum-Product: Example (2)



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

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

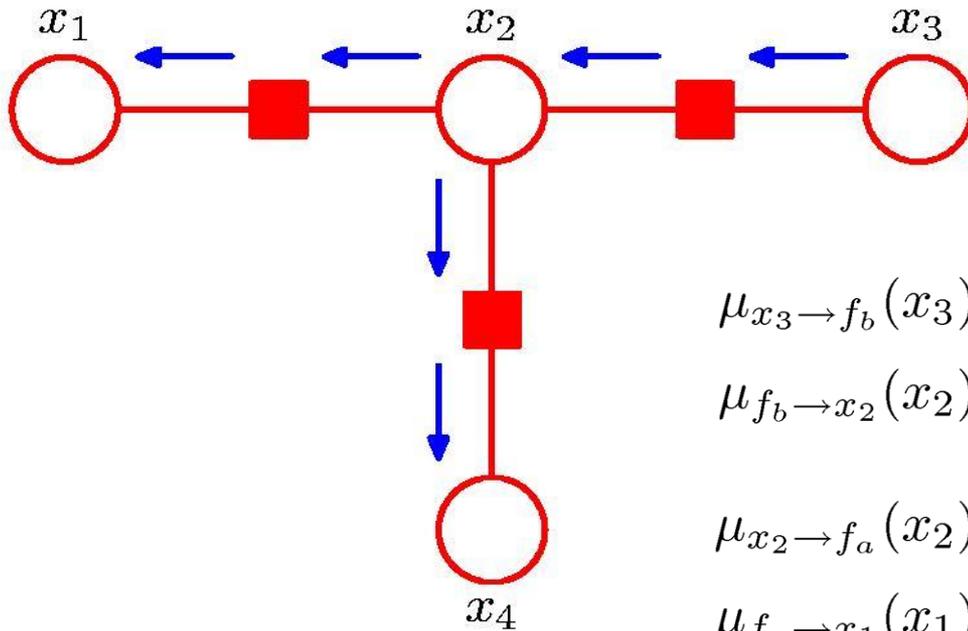
$$\mu_{x_4 \rightarrow f_c}(x_4) = 1$$

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

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

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

Sum-Product: Example (3)



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

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

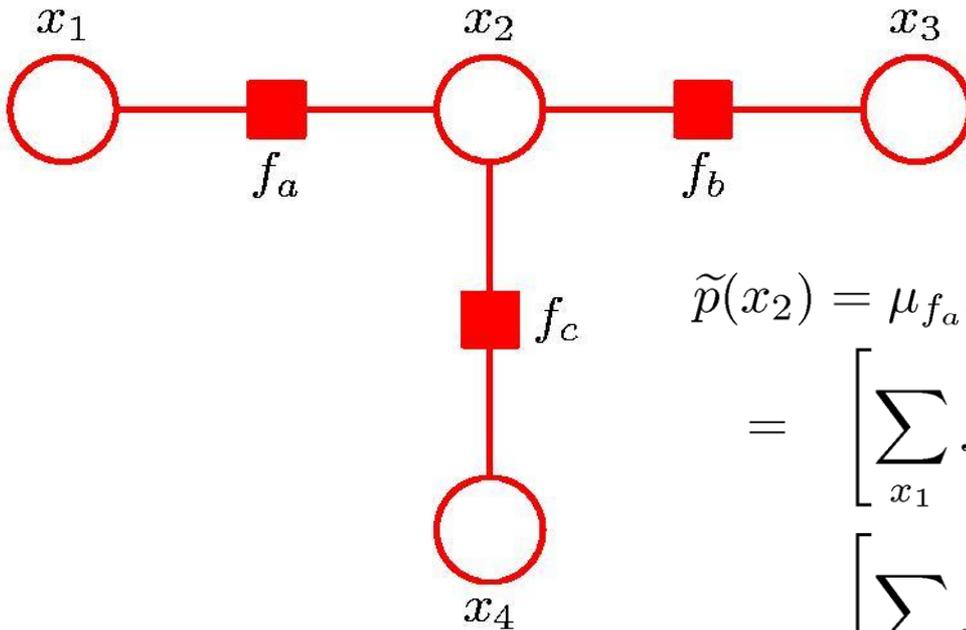
$$\mu_{x_2 \rightarrow f_a}(x_2) = \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2)$$

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

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

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

Sum-Product: Example (4)



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

The Max-Sum Algorithm (1)

Objective: an efficient algorithm for finding

- i. the value \mathbf{x}^{\max} that maximises $p(\mathbf{x})$;
- ii. the value of $p(\mathbf{x}^{\max})$.

In general, maximum marginals \neq joint maximum.

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

$$\arg \max_x p(x, y) = 1 \qquad \arg \max_x p(x) = 0$$

The Max-Sum Algorithm (2)

Maximizing over a chain (max-product)



$$\begin{aligned} p(\mathbf{x}^{\max}) &= \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \dots \max_{x_N} p(\mathbf{x}) \\ &= \frac{1}{Z} \max_{x_1} \dots \max_{x_N} [\psi_{1,2}(x_1, x_2) \dots \psi_{N-1,N}(x_{N-1}, x_N)] \\ &= \frac{1}{Z} \max_{x_1} \left[\max_{x_2} \left[\psi_{1,2}(x_1, x_2) \left[\dots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \dots \right] \right] \end{aligned}$$

The Max-Sum Algorithm (3)

Generalizes to tree-structured factor graph

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_n} \prod_{f_s \in \text{ne}(x_n)} \max_{X_s} f_s(x_n, X_s)$$

maximizing as close to the leaf nodes as possible

The Max-Sum Algorithm (4)

Max-Product \rightarrow Max-Sum

For numerical reasons, use

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

Again, use distributive law

$$\max(a + b, a + c) = a + \max(b, c).$$

The Max-Sum Algorithm (5)

Initialization (leaf nodes)

$$\mu_{x \rightarrow f}(x) = 0 \qquad \mu_{f \rightarrow x}(x) = \ln f(x)$$

Recursion

$$\mu_{f \rightarrow x}(x) = \max_{x_1, \dots, x_M} \left[\ln f(x, x_1, \dots, x_M) + \sum_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right]$$

$$\phi(x) = \arg \max_{x_1, \dots, x_M} \left[\ln f(x, x_1, \dots, x_M) + \sum_{m \in \text{ne}(f_s) \setminus x} \mu_{x_m \rightarrow f}(x_m) \right]$$

$$\mu_{x \rightarrow f}(x) = \sum_{l \in \text{ne}(x) \setminus f} \mu_{f_l \rightarrow x}(x)$$

The Max-Sum Algorithm (6)

Termination (root node)

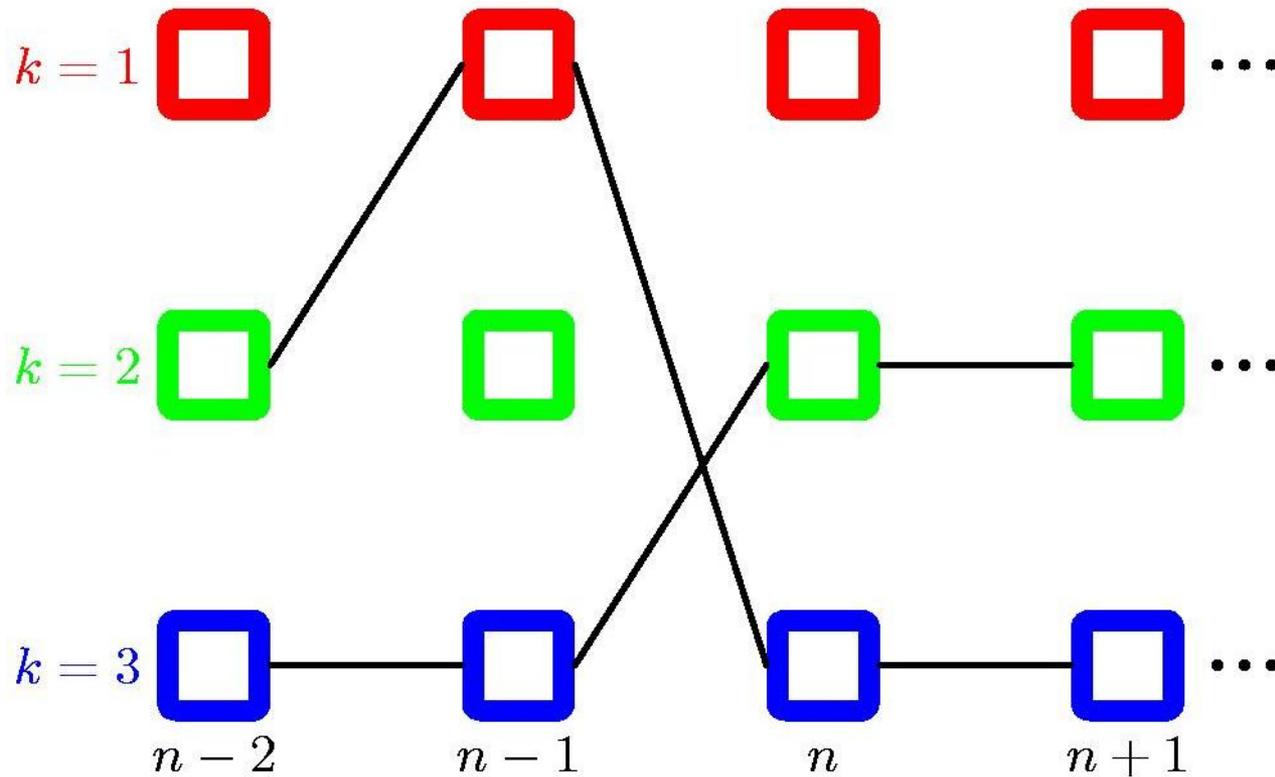
$$p^{\max} = \max_x \left[\sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right]$$
$$x^{\max} = \arg \max_x \left[\sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right]$$

Back-track, for all nodes i with l factor nodes to the root ($l=0$)

$$\mathbf{x}_l^{\max} = \phi(x_{i,l-1}^{\max})$$

The Max-Sum Algorithm (7)

Example: Markov chain



The Junction Tree Algorithm

- *Exact* inference on general graphs.
 - Works by turning the initial graph into a *junction tree* and then running a sum-product-like algorithm.
 - *Intractable* on graphs with large cliques.
-

Loopy Belief Propagation

- Sum-Product on general graphs.
 - Initial unit messages passed across all links, after which messages are passed around until convergence (not guaranteed!).
 - *Approximate* but *tractable* for large graphs.
 - Sometime works well, sometimes not at all.
-