EECS 545: Machine Learning

Lecture 14. Markov Networks

Honglak Lee

2/23/2011

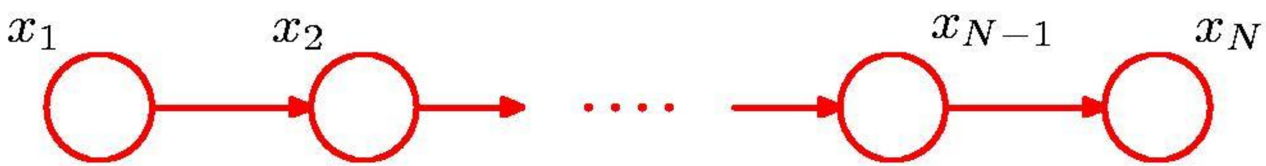
Midterm Student Feedback

- Ending classes on time!
- More high-level intuitions and applications
  – I will try to incorporate as much as possible. However, it is important to point out that the goal of this course is to provide you sufficient depth.
- More interactions
  – I will try to incorporate some short (1 min) quizes that include discussions between pair of students and the instructor.

Outline

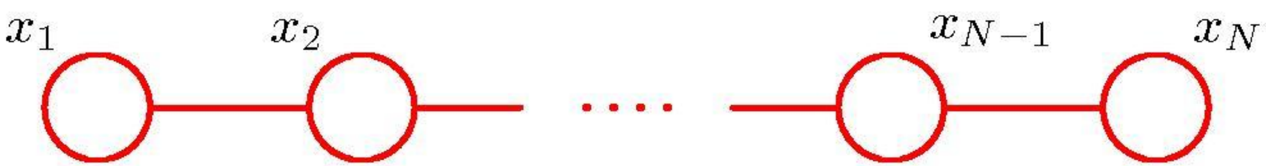
- Directed vs Undirected graphical models
- Inference in graphical models

Converting Directed to Undirected Graphs (1)



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

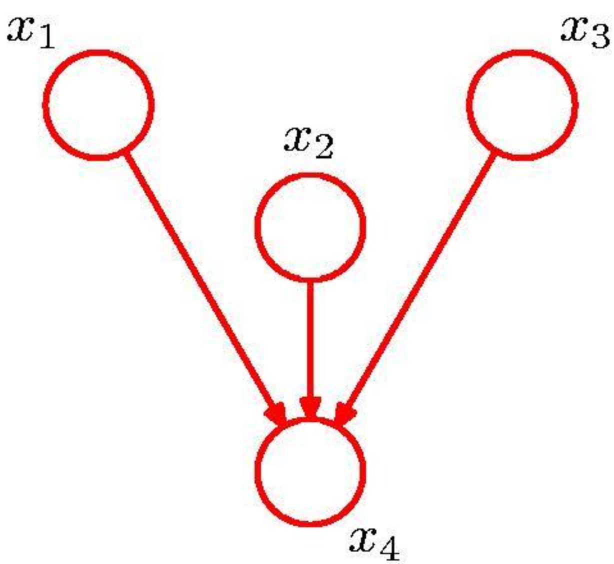
$$p(\mathbf{x}) = \frac{1}{Z}\, \psi_{1,2}(x_1, x_2)\, \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)$$

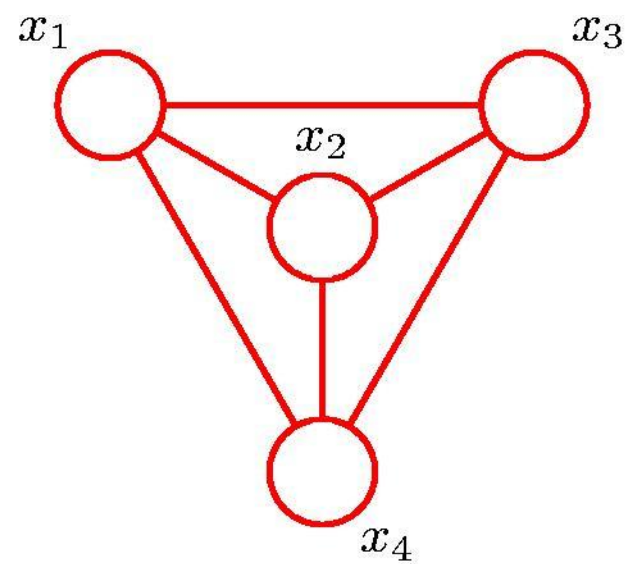


Converting Directed to Undirected Graphs (2)

- Additional links

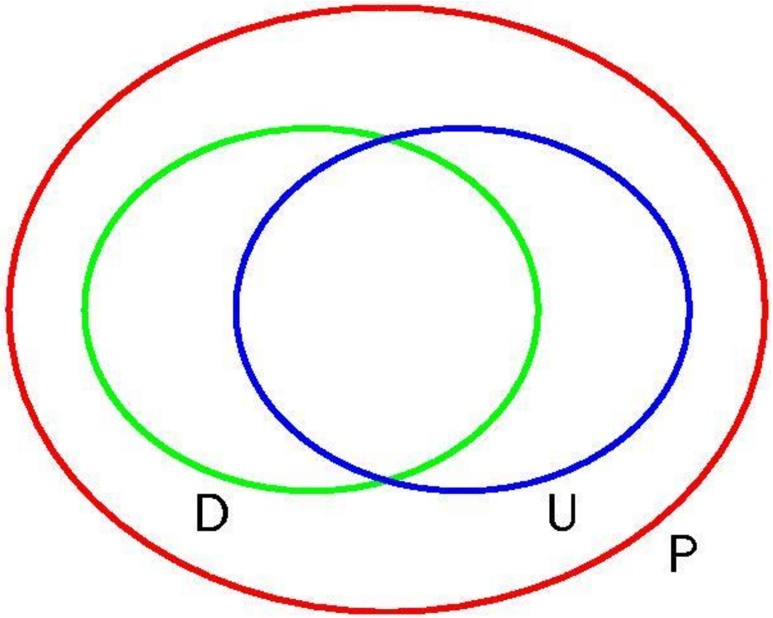
Moralizing: “Moral Graph”





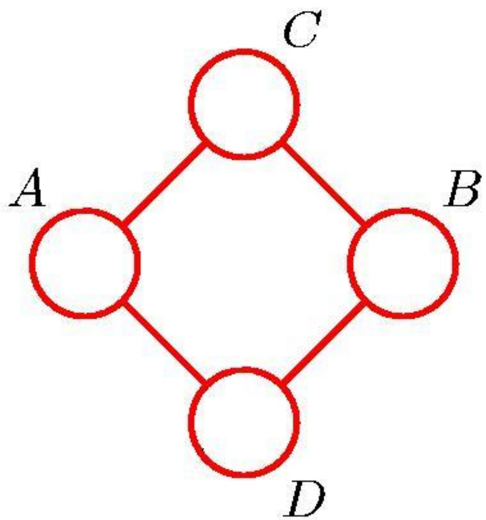
$$
\begin{aligned}
p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3) \\
&= \frac{1}{Z}\psi_A(x_1,x_2,x_3)\psi_B(x_2,x_3,x_4)\psi_C(x_1,x_2,x_4)
\end{aligned}
$$

Directed vs. Undirected Graphs (1)



Directed vs. Undirected Graphs (2)

E.g., Markov Network, but cannot be represented by Bayesian Network



$$A \not\perp\!\!\!\perp B \mid \emptyset$$

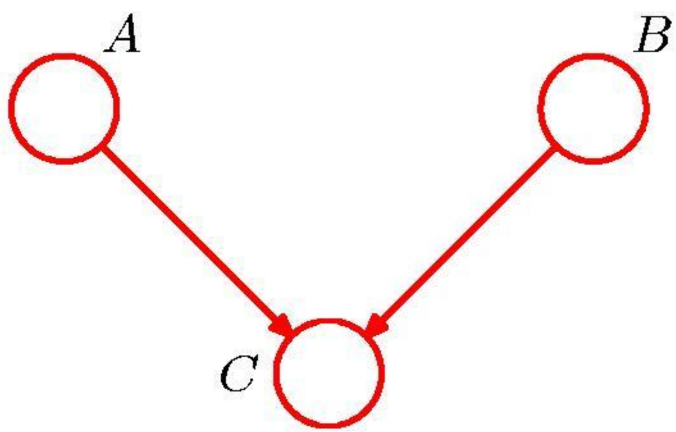
$$A \perp\!\!\!\perp B \mid C \cup D$$

$$C \perp\!\!\!\perp D \mid A \cup B$$

Q. Can this graph be converted into an equivalent directed graph? If not, why?

Directed vs. Undirected Graphs (3)

E.g., Bayesian Network, but cannot be represented by Markov Network



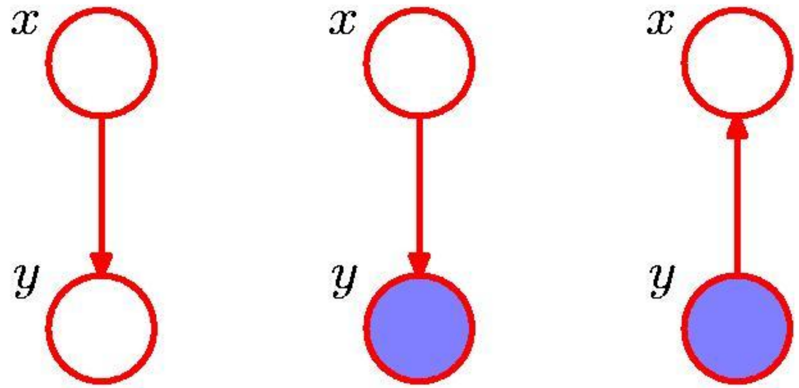
$$A \perp\!\!\!\perp B \mid \emptyset$$

$$A \not\!\perp\!\!\!\perp B \mid C$$

Q. Can this graph be converted into an equivalent undirected graph? If not, why?

Inference in graphical models

Inference in Graphical Models



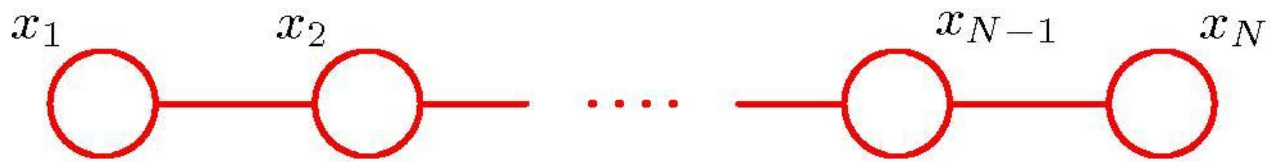
Marginal probability

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

Posterior probability

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

Inference on a Chain



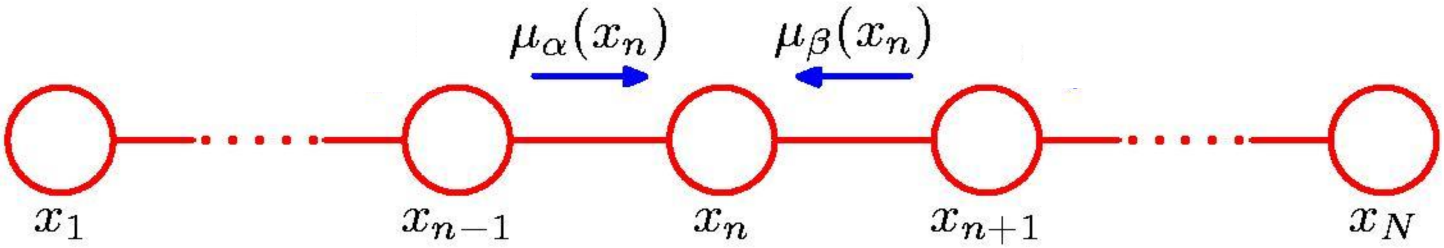
$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

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

$$\|$$

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

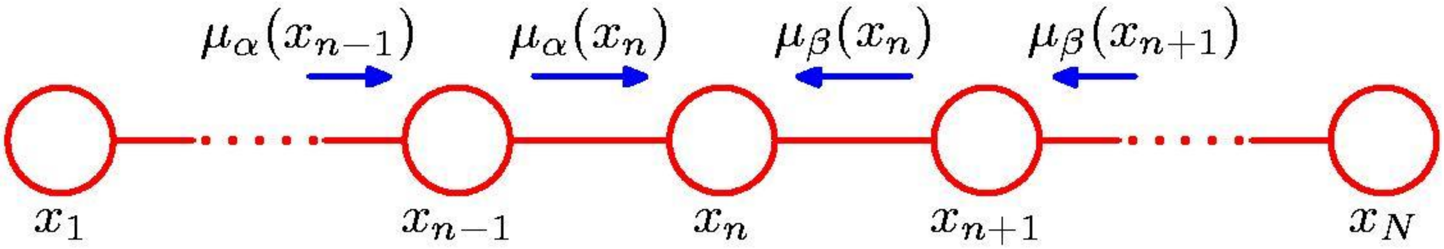
Inference on a Chain



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

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

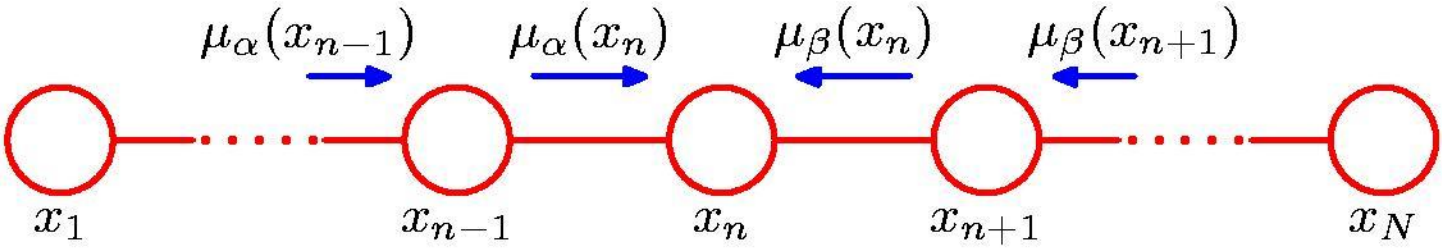
Inference on a Chain



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

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

Inference on a Chain



$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \qquad \mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

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

Q. Can you understand why three recursion rules hold?

Inference on a Chain

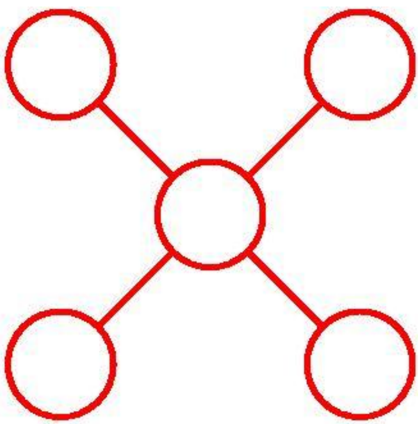
- To compute local marginals:
  - Compute and store all forward messages, $\mu_\alpha(x_n)$.
  - Compute and store all backward messages, $\mu_\beta(x_n)$.
  - Compute Z at any node x_m
  - Compute

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

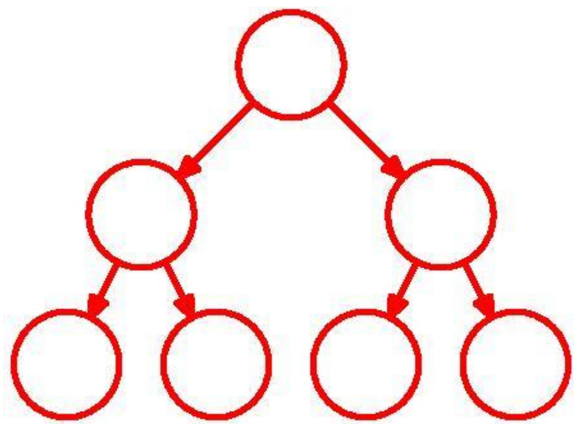
for all variables required.

Trees

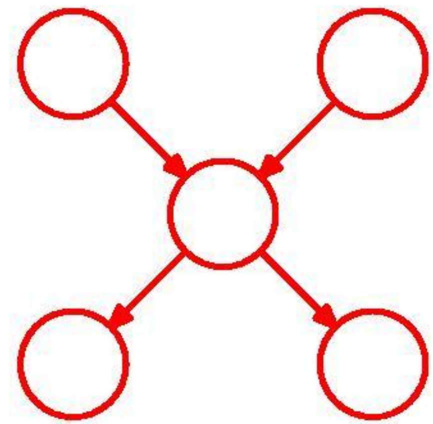
Undirected Tree



Directed Tree

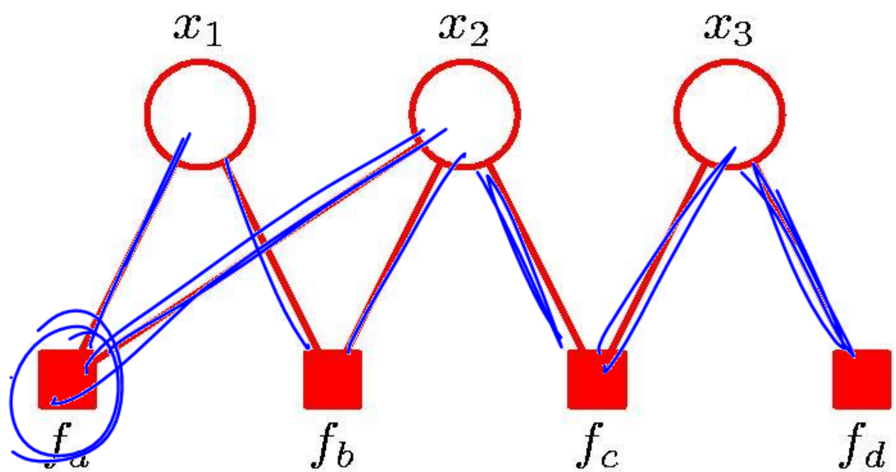


Polytree



Factor Graphs

variable Nodes: $X_1, \cdots, X_N$



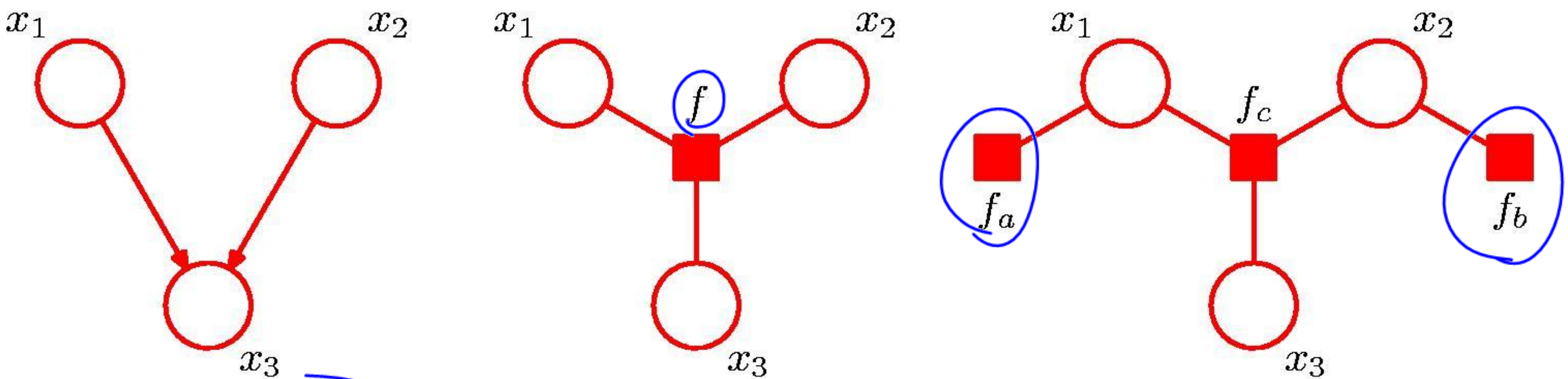
factor nodes

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

$\frac{1}{Z} \prod \psi_c(X_c)$

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

Factor Graphs from Directed Graphs

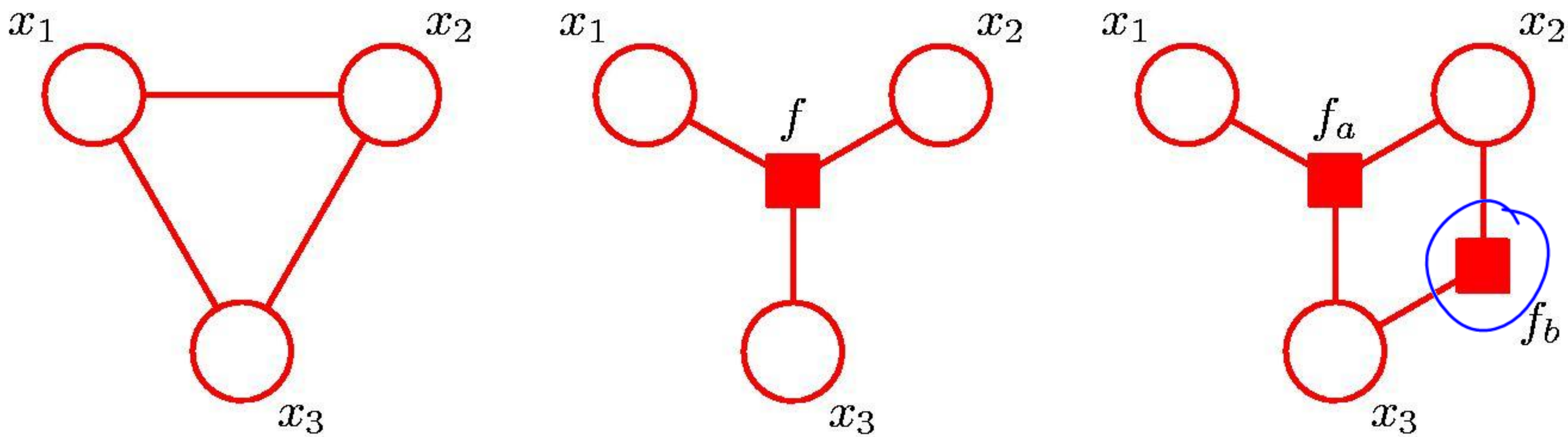


$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3|x_1,x_2)$$

$$f(x_1, x_2, x_3) = p(x_1)p(x_2)p(_3|x_1, x_2)$$

$$\begin{aligned} f_a(x_1) &= p(x_1) \\ f_b(x_2) &= p(x_2) \\ f_c(x_1, x_2, x_3) &= p(x_3|x_1, x_2) \end{aligned}$$

Factor Graphs from Undirected Graphs



$\psi(x_1, x_2, x_3)$

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

$f_a(x_1, x_2, x_3) f_b(x_2, x_3)$
$= \psi(x_1, x_2, x_3)$

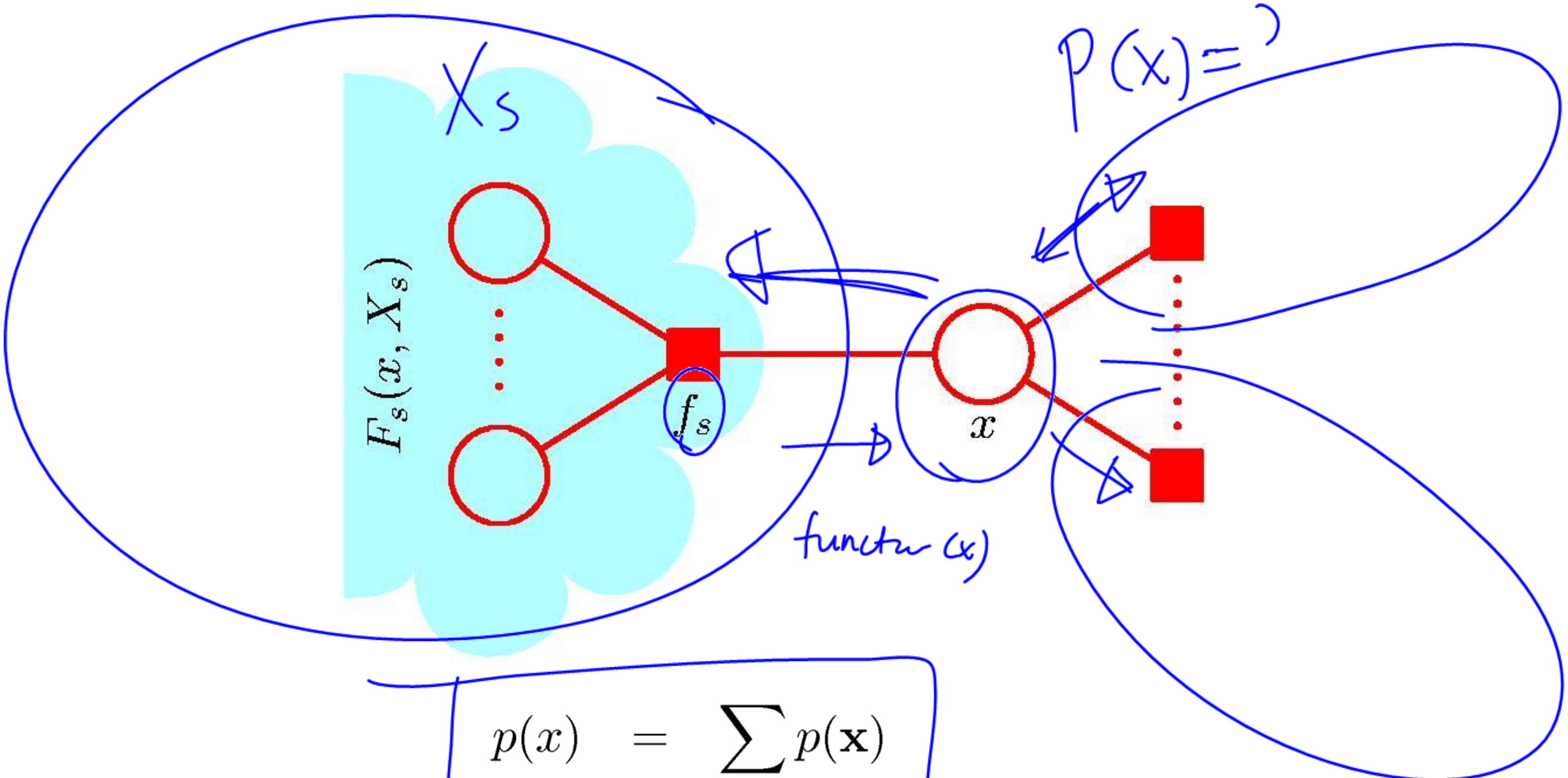
$\frac{1}{Z} \psi(x_1, x_2) \psi(x_2, x_3) \psi(x_3, x_1)$

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

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

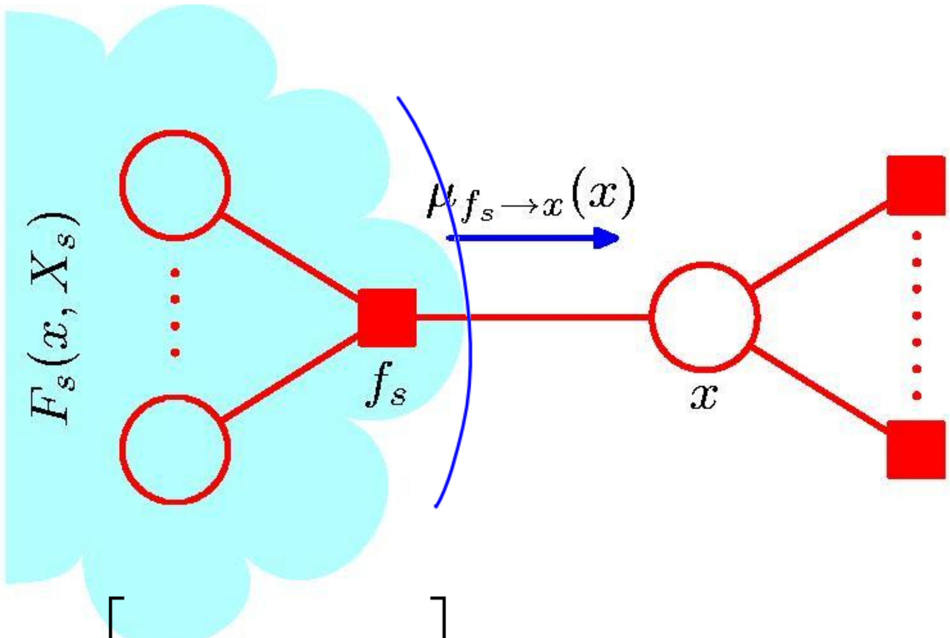
The Sum-Product Algorithm (2)



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

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

The Sum-Product Algorithm (3)

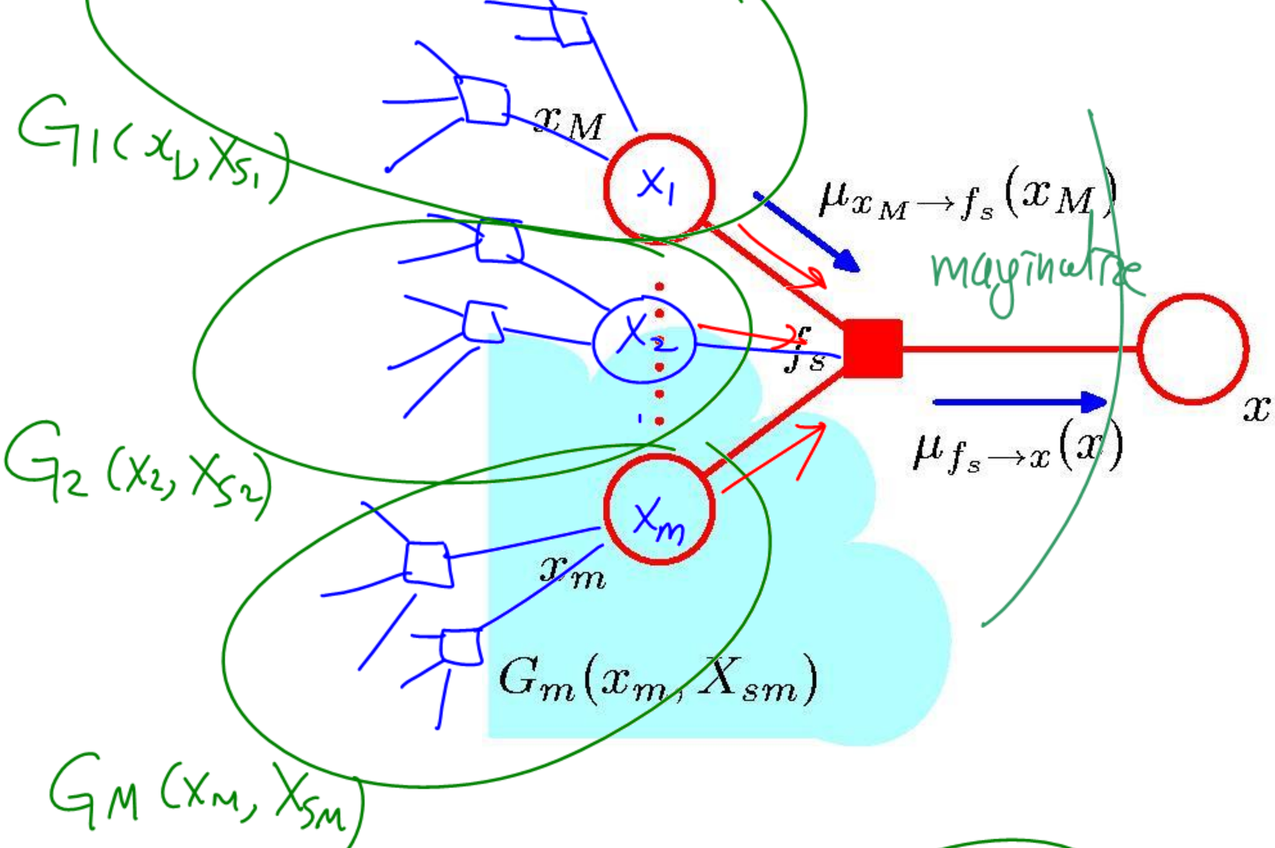


$$
\begin{aligned}
p(x) &= \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] \\
&= \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x).
\end{aligned}
$$

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

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

The Sum-Product Algorithm (4)



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

$$\mu_{f_s \to x}(x) = \sum_{x_1, \cdots x_m} \sum_{X_{s_1}} \sum_{X_{s_2}} \cdots \sum_{X_{sM}} F_s(x, X_s)$$

$G_M(x_M, X_{s_M})$

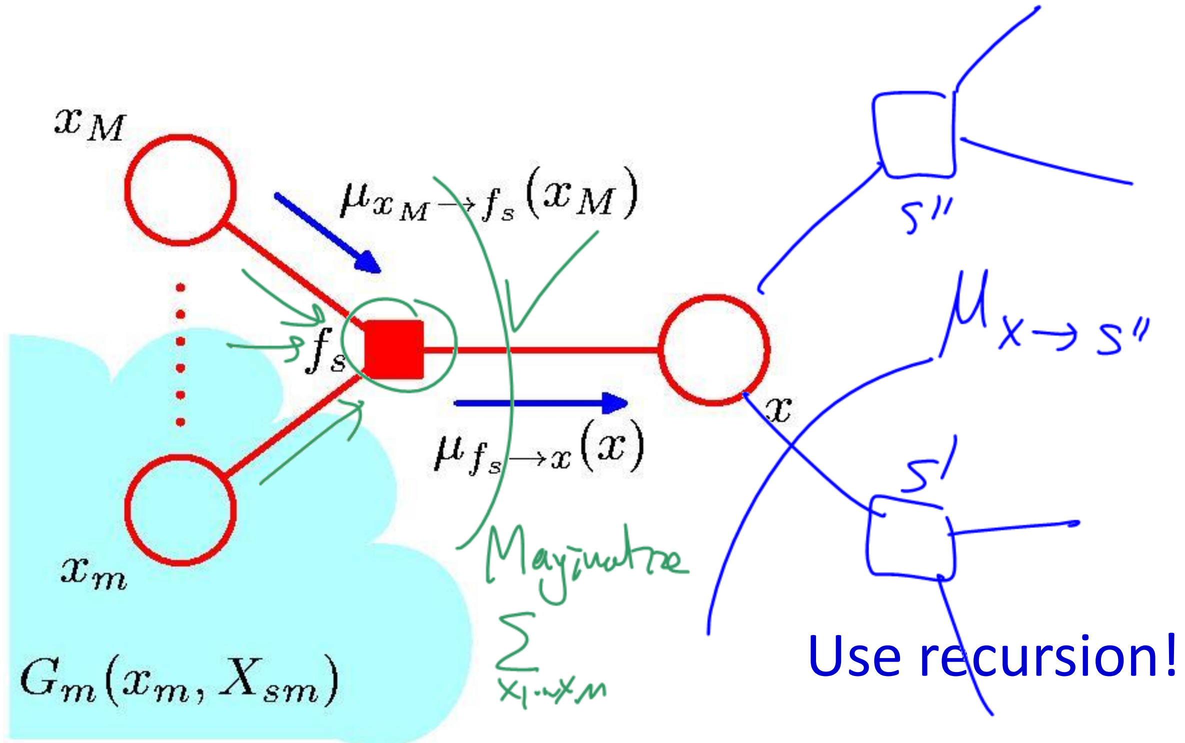
$$\mu_{f_s \to x}(x) = \sum_{x_1, \cdots x_m} \sum_{X_{s_1}} \sum_{X_{s_2}} \cdots \sum_{X_{s_M}} F_s(x, X_s)$$

$$= \quad f_s(x, x_1 \cdots x_m)\, G_1(x_1, X_{s_1}) \cdots G_M(x_M, X_{s_M})$$

$$= \sum_{x_1 \cdots x_m} f_s(x_1 \cdots x_m) \sum_{X_{s_1}} G_1(x_1, X_{s_1}) \sum_{X_{s_2}} G_2(x_2, X_{s_2}) \cdots \sum_{X_{s_M}} G(x_M, X_{s_M})$$

$$\sum_{X_{s_1}} G_1(x_1, X_{s_1}) = \mu_{x_1 \to f_s}(x_1), \quad \sum_{X_{s_2}} G_2(x_2, X_{s_2}) = \mu_{x_2 \to f_s}(x_2), \quad \sum_{X_{s_M}} G(x_M, X_{s_M}) = \mu_{x_M \to f_s}(x_M)$$

The Sum-Product Algorithm (5)



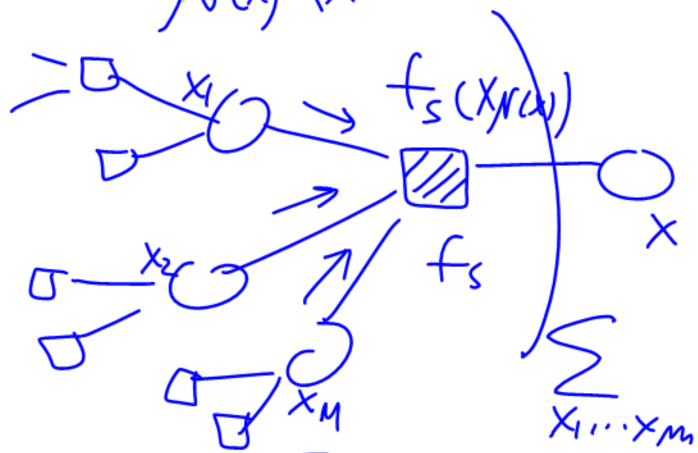
Use recursion!!

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

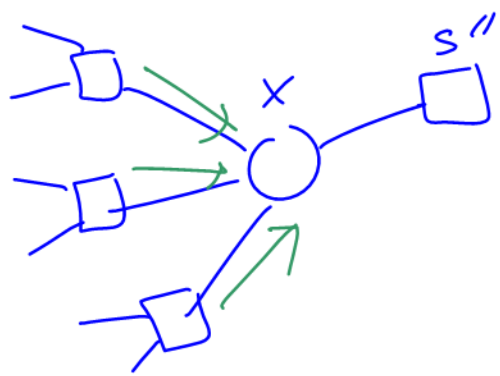
$$= \sum_{x_1} \dots \sum_{x_M} f_s(x, x_1, \dots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

$$\mu_{x \to s''}(x) = \mu_{f_s \to x}(x) \cdot \mu_{f_{s'} \to x}(x) = \prod_{f : N(x) \setminus f_s} \mu_{f \to x}(x)$$

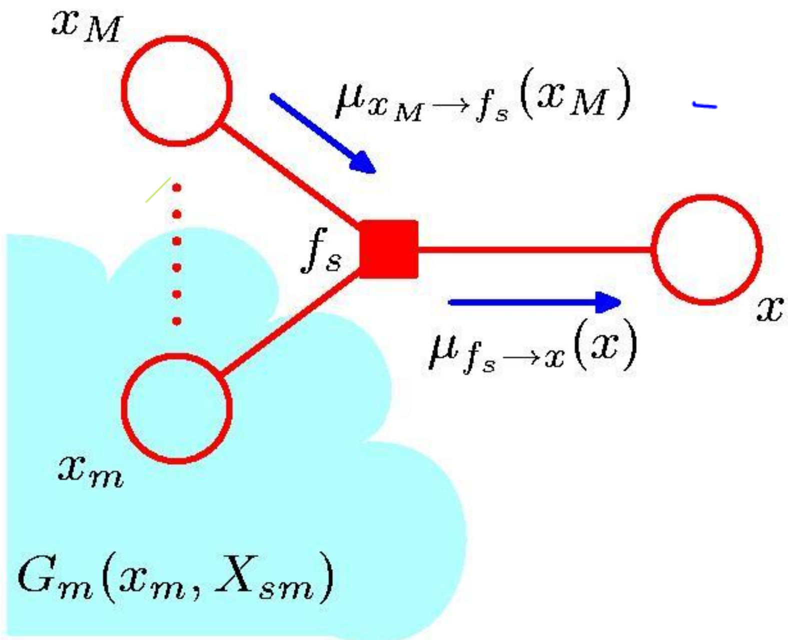
$$\mu_{f_s \to x} = \sum_{N(x) \setminus x} f_s(x_{N(x)}) \prod_{j \in N(f_s) \setminus x} \mu_{x_j \to f_s}(x_j)$$



$$\mu_{x \to f_{s''}} = \prod_{f : N(x) \setminus f_{s''}} \mu_{f \to x}(x)$$



The Sum-Product Algorithm (6)

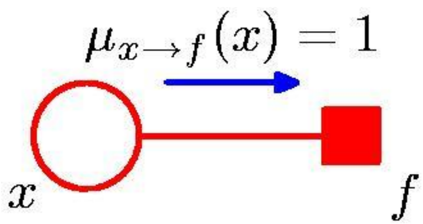


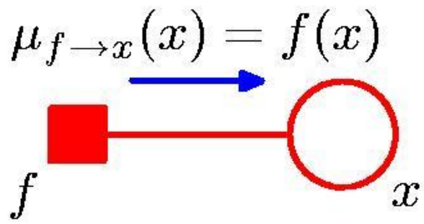
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

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

The Sum-Product Algorithm (7)

- Initialization

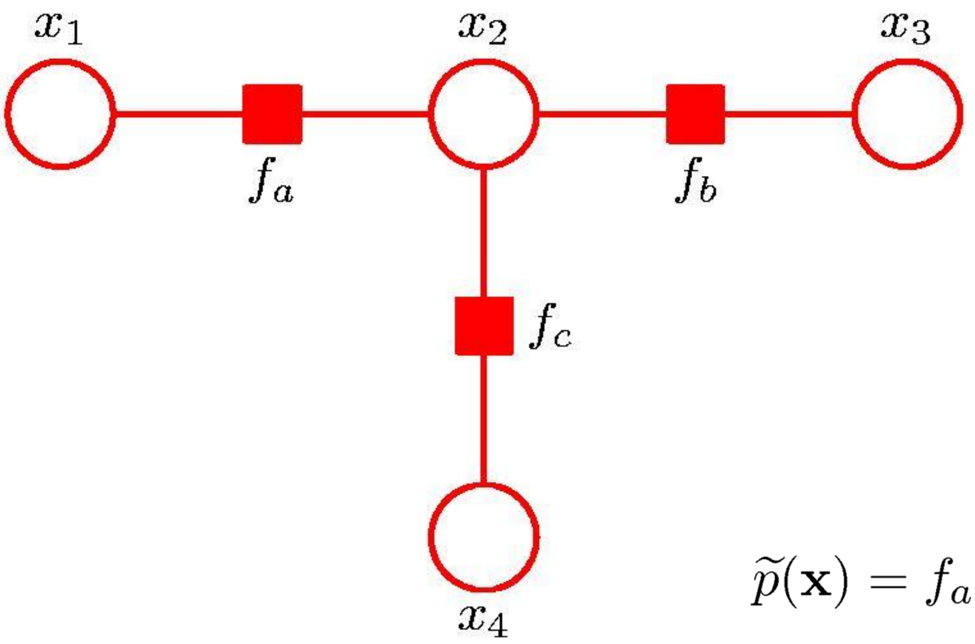




The Sum-Product Algorithm (8)

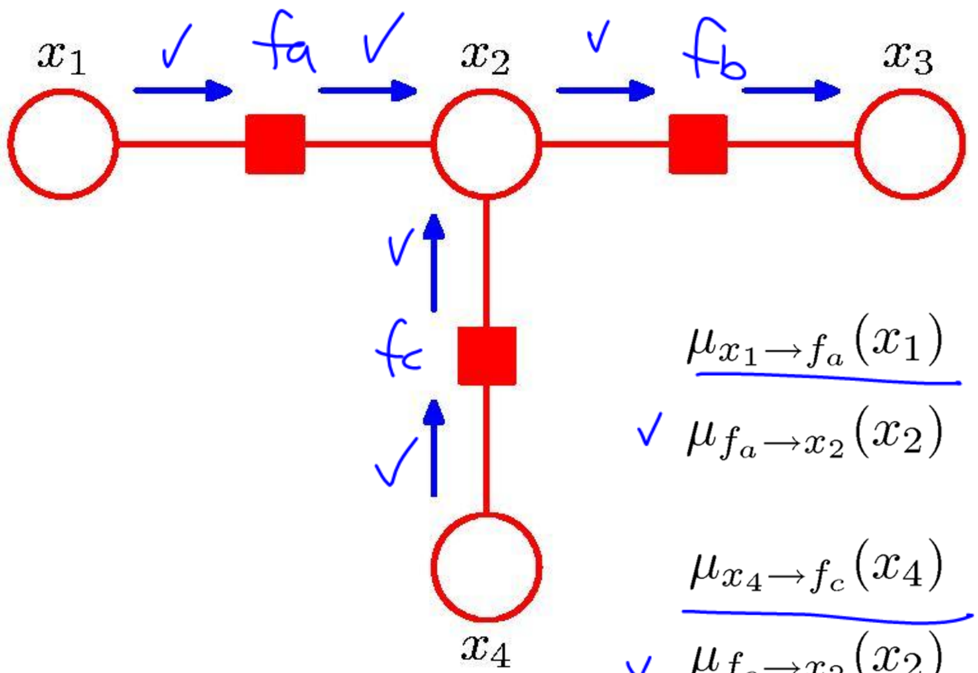
- To compute local marginals:
  - Pick an arbitrary node as root
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product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

Sum-Product: Example (1)



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

Sum-Product: Example (2)



$P(X_3)$

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

$$✓\ \mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2) \underbrace{\mu_{x_1 \to f_a}(x_1)}_{=1} = \sum_{x_1} f_a(x_1, x_2)$$

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

$$✓\ \mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4) \underbrace{\mu_{x_4 \to f_c}(x_4)}_{=1} = \sum_{x_4} f_c(x_2, x_4)$$

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

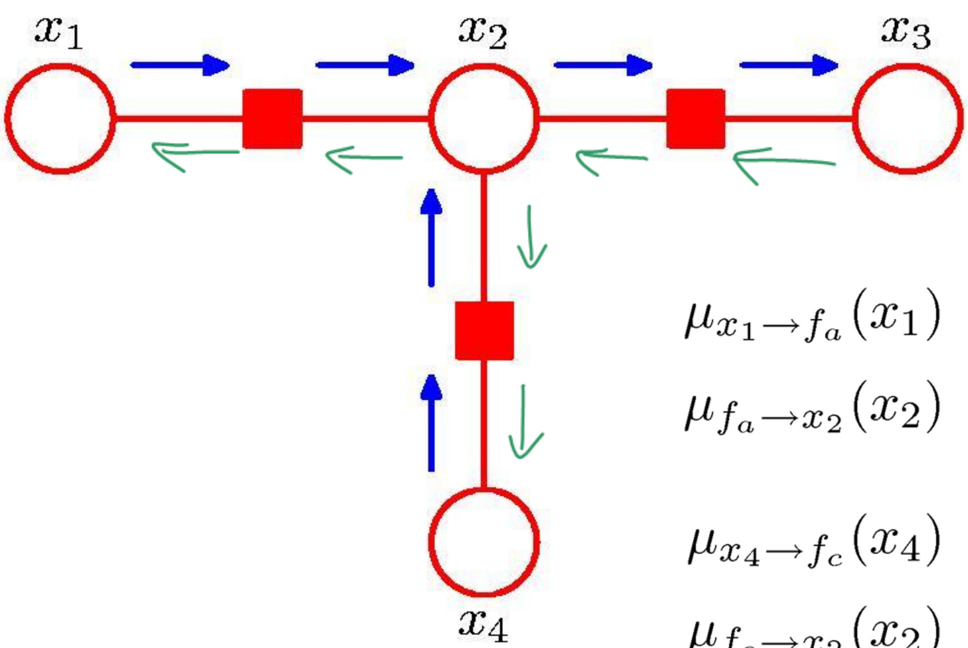
$$\boxed{\mu_{f_b \to x_3}(x_3)} = \sum_{x_2} f_b(x_2, x_3)\, \mu_{x_2 \to f_b}(x_2)$$

$$\|$$

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

Q. Can you fill this out?

Sum-Product: Example (2)



P(x₁), P(X₂), P(X₃), P(X₄)

M nodes.

O(M)

each factor has size of 2.

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

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

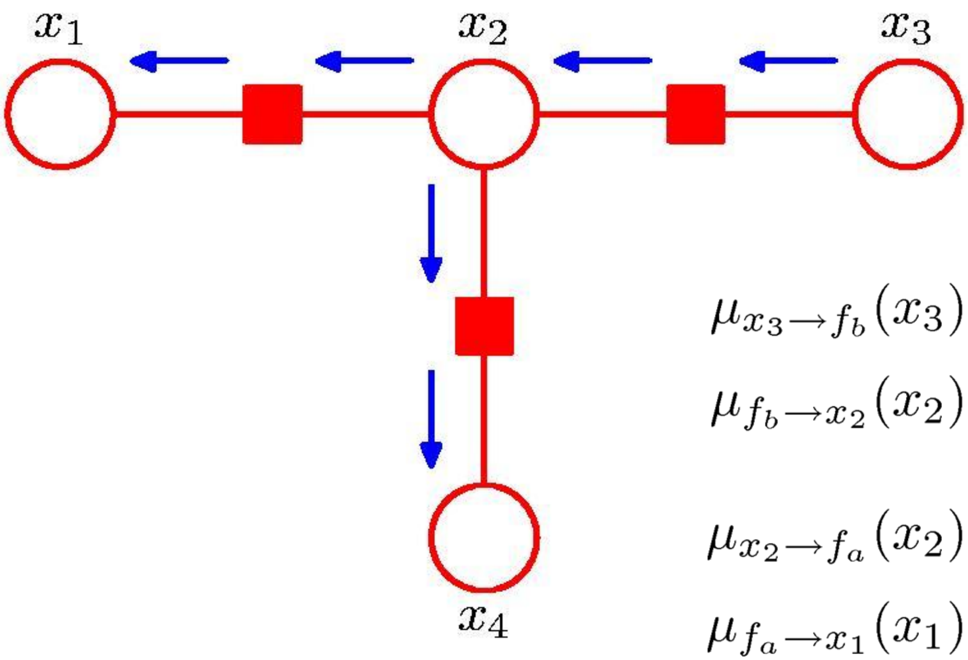
$$\mu_{x_4 \to f_c}(x_4) = 1$$

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

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

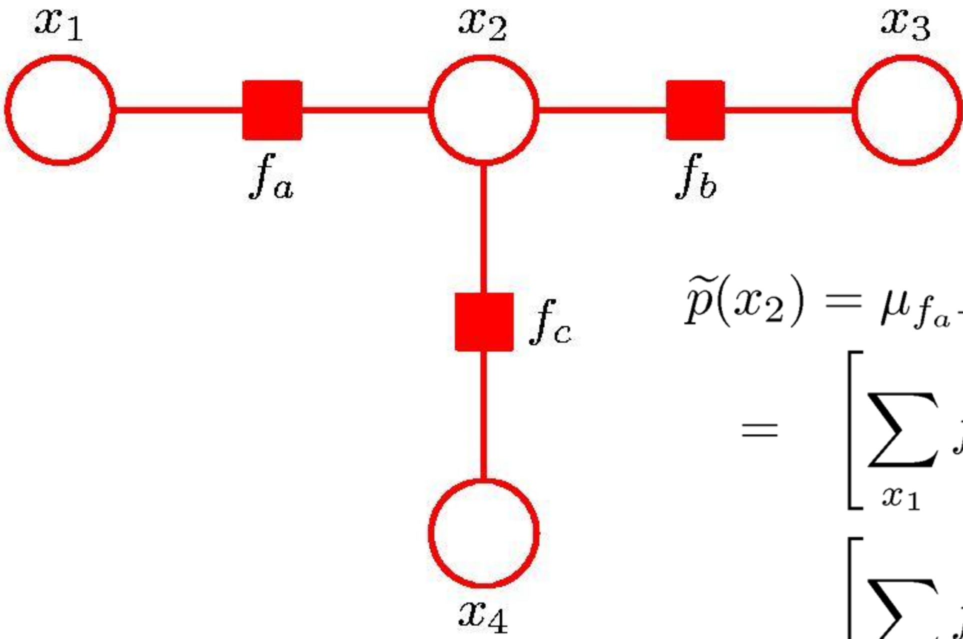
$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}(x_2)$$

Sum-Product: Example (3)



$$\begin{aligned}
\mu_{x_3 \to f_b}(x_3) &= 1 \\
\mu_{f_b \to x_2}(x_2) &= \sum_{x_3} f_b(x_2, x_3) \\
\mu_{x_2 \to f_a}(x_2) &= \mu_{f_b \to x_2}(x_2) \mu_{f_c \to x_2}(x_2) \\
\mu_{f_a \to x_1}(x_1) &= \sum_{x_2} f_a(x_1, x_2) \mu_{x_2 \to f_a}(x_2) \\
\mu_{x_2 \to f_c}(x_2) &= \mu_{f_a \to x_2}(x_2) \mu_{f_b \to x_2}(x_2) \\
\mu_{f_c \to x_4}(x_4) &= \sum_{x_2} f_c(x_2, x_4) \mu_{x_2 \to f_c}(x_2)
\end{aligned}$$

Sum-Product: Example (4)



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

The Max-Sum Algorithm (1)

- Objective: an efficient algorithm for finding
  i. the value x^{max} that maximises $p(x)$;
  ii. the value of $p(x^{max})$.

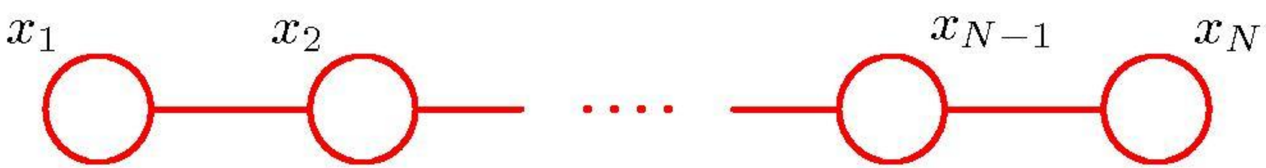
- In general, maximum marginals $\neq$ joint maximum.

	$x = 0$	$x = 1$
$y = 0$	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)



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

The Max-Sum Algorithm (3)

- Generalizes to tree-structured factor graph

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

- 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)

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

- Recursion

$$\begin{aligned}
\mu_{f \to x}(x) &= \max_{x_1, \ldots, x_M} \left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f}(x_m) \right] \\
\phi(x) &= \arg\max_{x_1, \ldots, x_M} \left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f}(x_m) \right] \\
\mu_{x \to f}(x) &= \sum_{l \in \mathrm{ne}(x) \setminus f} \mu_{f_l \to x}(x)
\end{aligned}$$

The Max-Sum Algorithm (6)

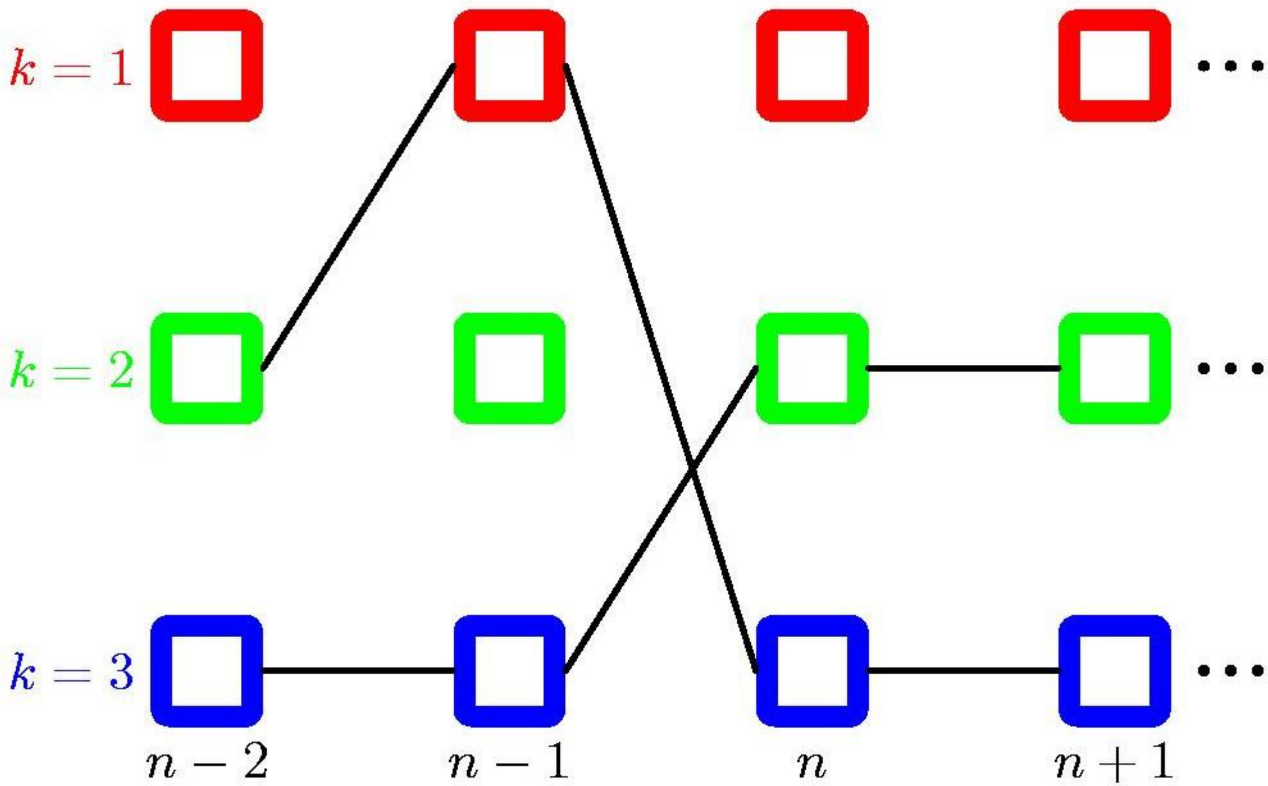
- Termination (root node)

$$
\begin{aligned}
p^{\max} &= \max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right] \\
x^{\max} &= \arg\max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]
\end{aligned}
$$

- Back-tracking to get the full assignment.

The Max-Sum Algorithm (7)

- Example: Markov chain



The Junction Tree Algorithm (sketch)

- *Exact* inference on general graphs.
- Works by turning the initial graph into a *junction tree* and then running a sum-product-like algorithm.
  1. Convert to undirected graph
  2. Triangulate the graph
  3. Construct a junction tree (where the nodes are cliques of the triangulated graph)
  4. Run belief propagation (e.g., sum-product)
- *Intractable* on graphs with large cliques.

Loopy Belief Propagation (sketch)

- 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.
- Read the Bishop book.

Next class

- Learning in graphical models
  – Maximum likelihood for fully observed variables
  – EM for partially observed variables