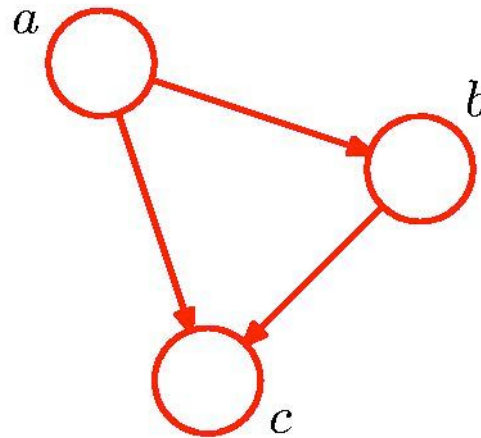

PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER 8: GRAPHICAL MODELS

Bayesian Networks

Directed Acyclic Graph (DAG)

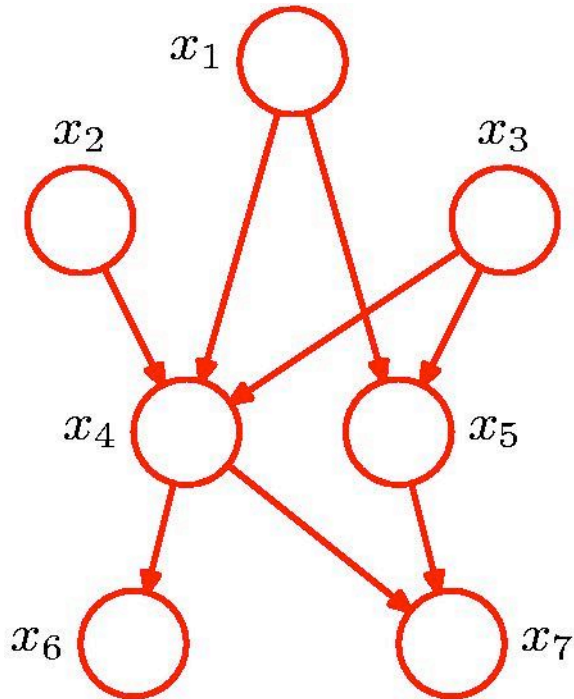


$$p(a, b, c) = p(c|a, b)p(a, b) = p(c|a, b)p(b|a)p(a)$$

$$p(x_1, \dots, x_K) = p(x_K|x_1, \dots, x_{K-1}) \dots p(x_2|x_1)p(x_1)$$

Bayesian Networks

$$p(x_1, \dots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\ p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$



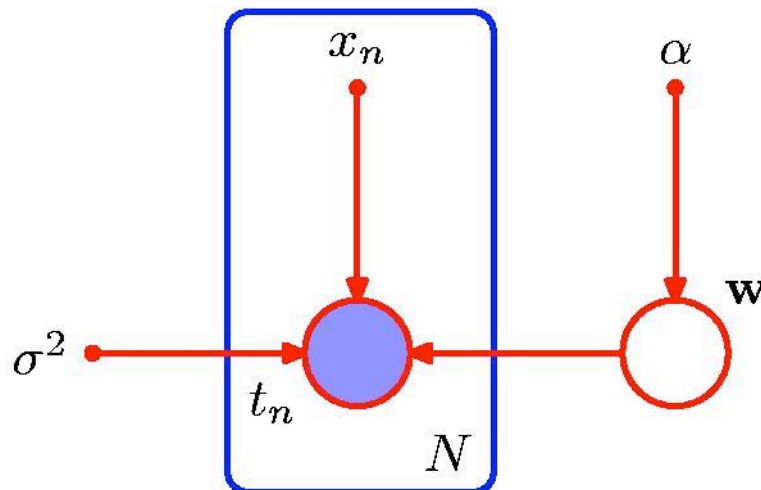
General Factorization

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

Bayesian Curve Fitting—Learning

Condition on data

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w}) \prod_{n=1}^N p(t_n|\mathbf{w})$$

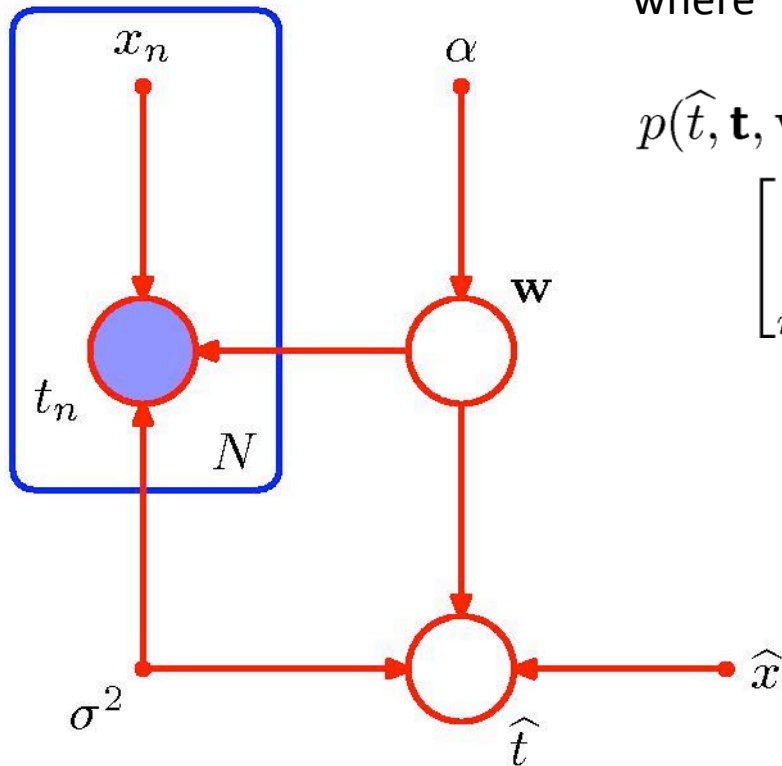


Bayesian Curve Fitting—Prediction

Predictive distribution: $p(\hat{t}|\hat{x}, \mathbf{x}, \mathbf{t}, \alpha, \sigma^2) \propto \int p(\hat{t}, \mathbf{t}, \mathbf{w}|\hat{x}, \mathbf{x}, \alpha, \sigma^2) d\mathbf{w}$

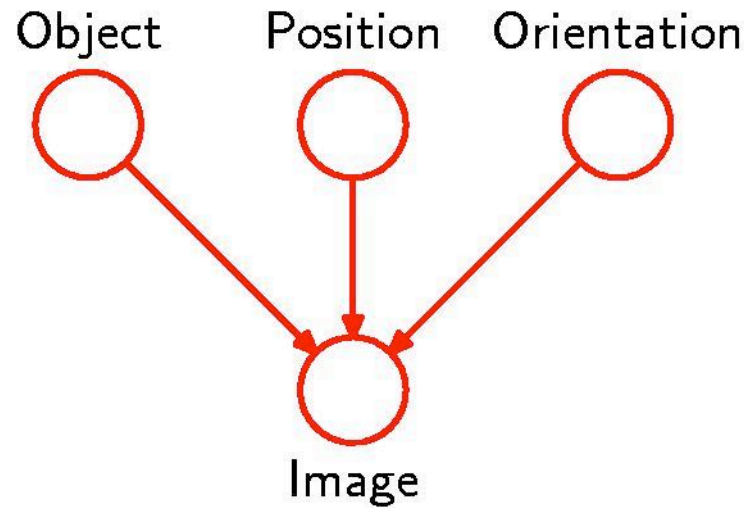
where

$$p(\hat{t}, \mathbf{t}, \mathbf{w}|\hat{x}, \mathbf{x}, \alpha, \sigma^2) = \left[\prod_{n=1}^N p(t_n|x_n, \mathbf{w}, \sigma^2) \right] p(\mathbf{w}|\alpha)p(\hat{t}|\hat{x}, \mathbf{w}, \sigma^2)$$



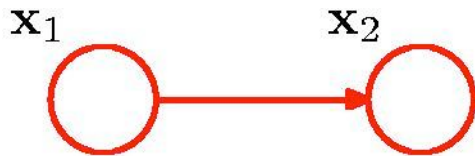
Generative Models

Causal process for generating images



Discrete Variables (1)

General joint distribution: $K^2 - 1$ parameters



$$p(\mathbf{x}_1, \mathbf{x}_2 | \boldsymbol{\mu}) = \prod_{k=1}^K \prod_{l=1}^K \mu_{kl}^{x_{1k} x_{2l}}$$

Independent joint distribution: $2(K - 1)$ parameters



$$\hat{p}(\mathbf{x}_1, \mathbf{x}_2 | \boldsymbol{\mu}) = \prod_{k=1}^K \mu_{1k}^{x_{1k}} \prod_{l=1}^K \mu_{2l}^{x_{2l}}$$

Discrete Variables (2)

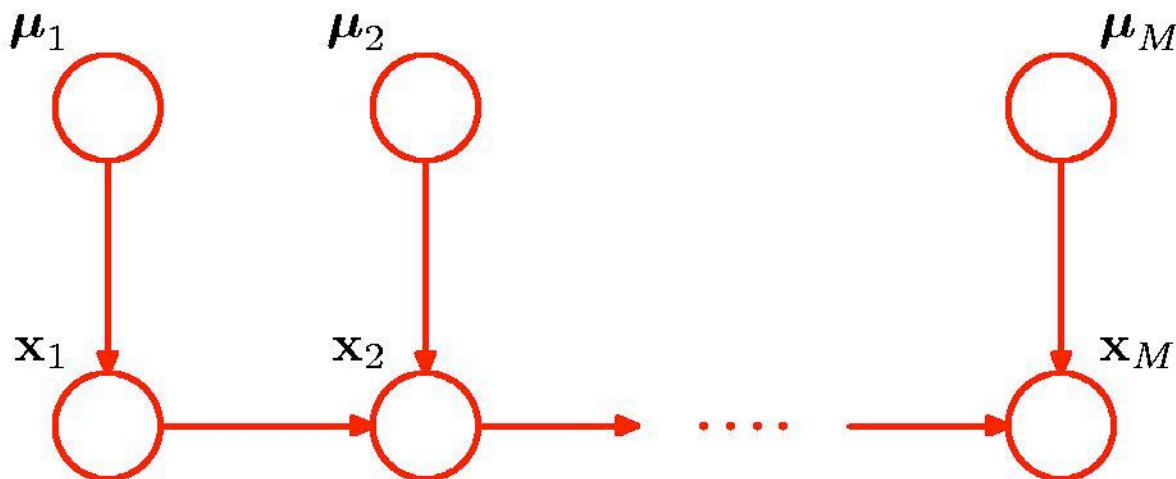
General joint distribution over M variables:

$K^M - 1$ parameters

M -node Markov chain: $K - 1 + (M - 1)K(K - 1)$
parameters



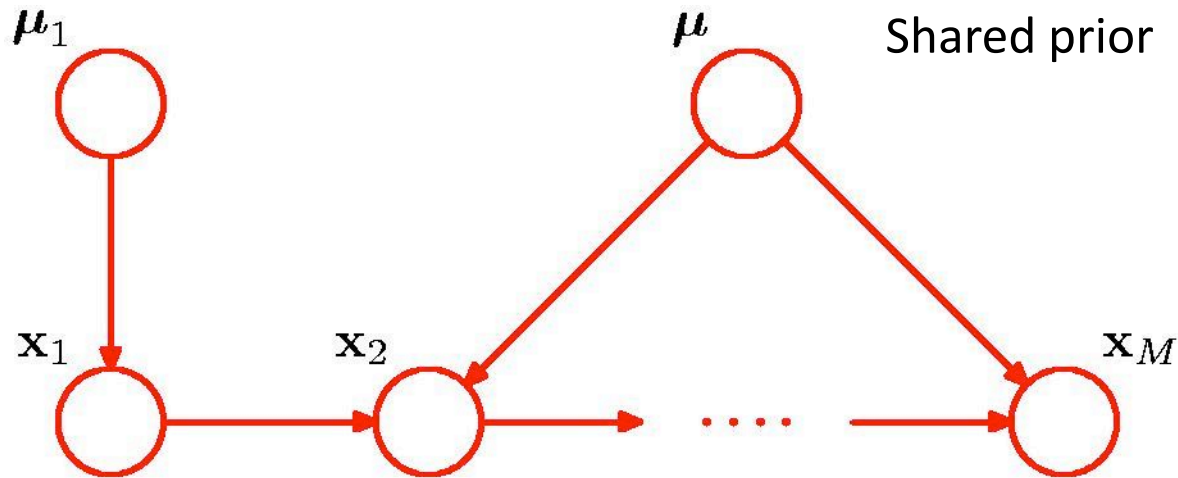
Discrete Variables: Bayesian Parameters (1)



$$p(\{\mathbf{x}_m, \boldsymbol{\mu}_m\}) = p(\mathbf{x}_1 | \boldsymbol{\mu}_1) p(\boldsymbol{\mu}_1) \prod_{m=2}^M p(\mathbf{x}_m | \mathbf{x}_{m-1}, \boldsymbol{\mu}_m) p(\boldsymbol{\mu}_m)$$

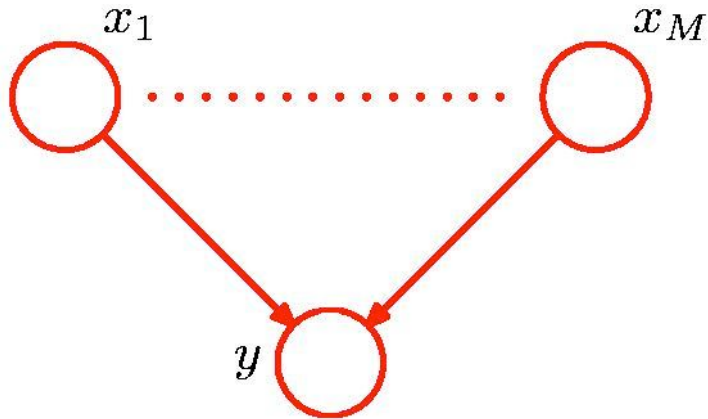
$$p(\boldsymbol{\mu}_m) = \text{Dir}(\boldsymbol{\mu}_m | \boldsymbol{\alpha}_m)$$

Discrete Variables: Bayesian Parameters (2)



$$p(\{x_m\}, \mu_1, \mu) = p(x_1 | \mu_1) p(\mu_1) \prod_{m=2}^M p(x_m | x_{m-1}, \mu) p(\mu)$$

Parameterized Conditional Distributions



If x_1, \dots, x_M are discrete, K -state variables, $p(y = 1|x_1, \dots, x_M)$ in general has $O(K^M)$ parameters.

The parameterized form

$$p(y = 1|x_1, \dots, x_M) = \sigma \left(w_0 + \sum_{i=1}^M w_i x_i \right) = \sigma(\mathbf{w}^T \mathbf{x})$$

requires only $M + 1$ parameters

Linear-Gaussian Models

Directed Graph

$$p(x_i | \text{pa}_i) = \mathcal{N} \left(x_i \mid \sum_{j \in \text{pa}_i} w_{ij} x_j + b_i, v_i \right)$$

Each node is Gaussian, the mean is a linear function of the parents.

Vector-valued Gaussian Nodes

$$p(\mathbf{x}_i | \text{pa}_i) = \mathcal{N} \left(\mathbf{x}_i \mid \sum_{j \in \text{pa}_i} \mathbf{W}_{ij} \mathbf{x}_j + \mathbf{b}_i, \mathbf{\Sigma}_i \right)$$

Conditional Independence

a is independent of b given c

$$p(a|b, c) = p(a|c)$$

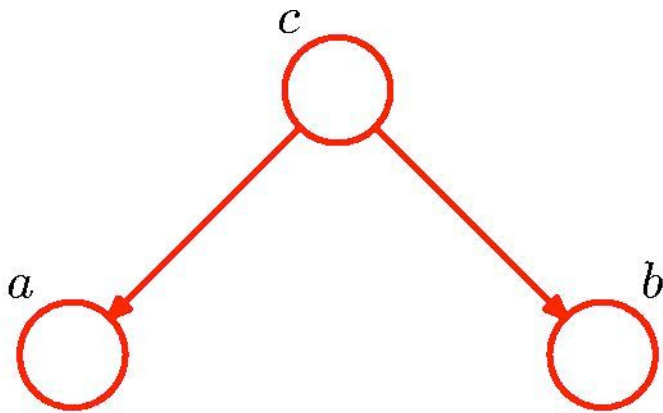
Equivalently

$$\begin{aligned} p(a, b|c) &= p(a|b, c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

Notation

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 1

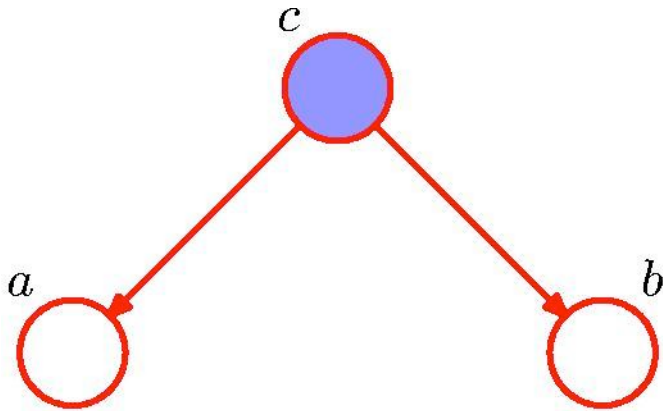


$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

$$p(a, b) = \sum_c p(a|c)p(b|c)p(c)$$

$$a \perp\!\!\!\perp b \mid \emptyset$$

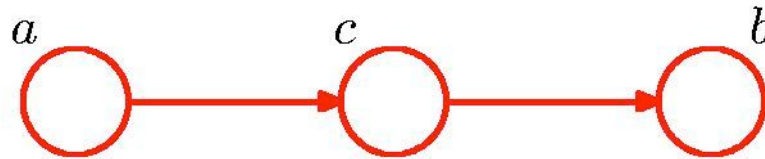
Conditional Independence: Example 1



$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 2

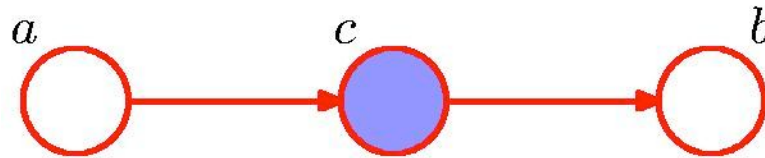


$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a) \sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp b \mid \emptyset$$

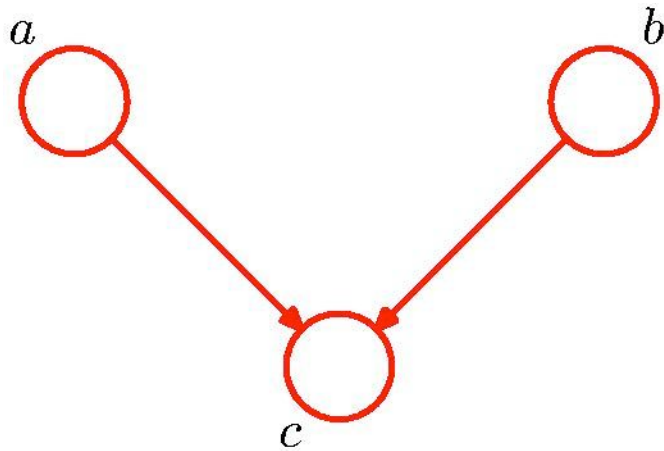
Conditional Independence: Example 2



$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 3



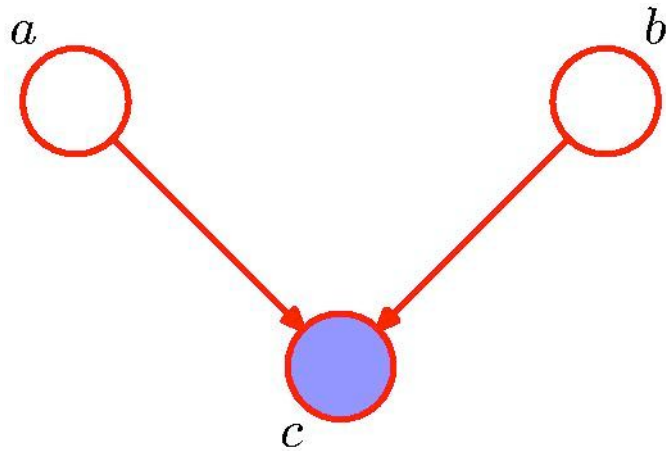
$$p(a, b, c) = p(a)p(b)p(c|a, b)$$

$$p(a, b) = p(a)p(b)$$

$$a \perp\!\!\!\perp b \mid \emptyset$$

Note: this is the opposite of Example 1, with c unobserved.

Conditional Independence: Example 3



$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(b)p(c|a, b)}{p(c)} \end{aligned}$$

$$a \not\perp b | c$$

Note: this is the opposite of Example 1, with c observed.

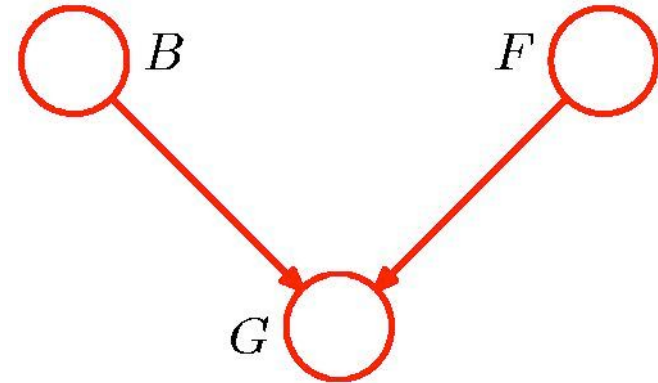
“Am I out of fuel?”

$$p(G = 1 | B = 1, F = 1) = 0.8$$

$$p(G = 1 | B = 1, F = 0) = 0.2$$

$$p(G = 1 | B = 0, F = 1) = 0.2$$

$$p(G = 1 | B = 0, F = 0) = 0.1$$



$$p(B = 1) = 0.9$$

$$p(F = 1) = 0.9$$

and hence

$$p(F = 0) = 0.1$$

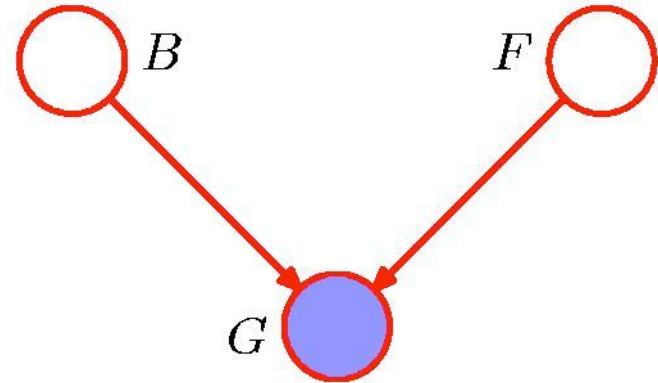
B = Battery (0=flat, 1=fully charged)

F = Fuel Tank (0=empty, 1=full)

G = Fuel Gauge Reading

(0=empty, 1=full)

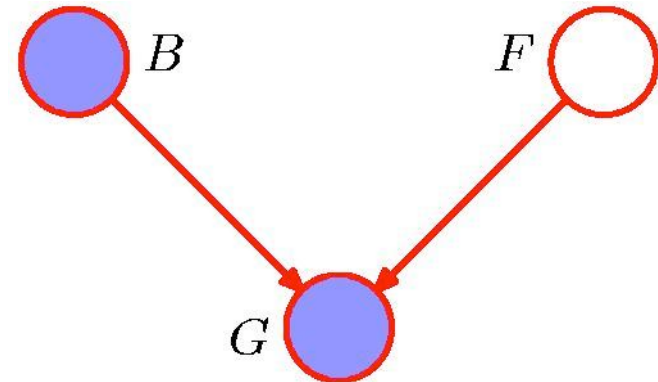
“Am I out of fuel?”



$$\begin{aligned} p(F = 0|G = 0) &= \frac{p(G = 0|F = 0)p(F = 0)}{p(G = 0)} \\ &\simeq 0.257 \end{aligned}$$

Probability of an empty tank increased by observing $G = 0$.

“Am I out of fuel?”



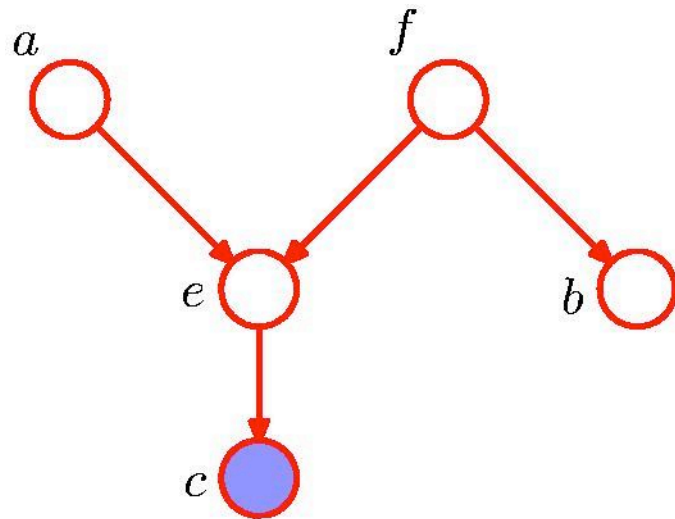
$$\begin{aligned} p(F = 0 | G = 0, B = 0) &= \frac{p(G = 0 | B = 0, F = 0)p(F = 0)}{\sum_{F \in \{0,1\}} p(G = 0 | B = 0, F)p(F)} \\ &\simeq 0.111 \end{aligned}$$

Probability of an empty tank reduced by observing $B = 0$.
This referred to as “explaining away”.

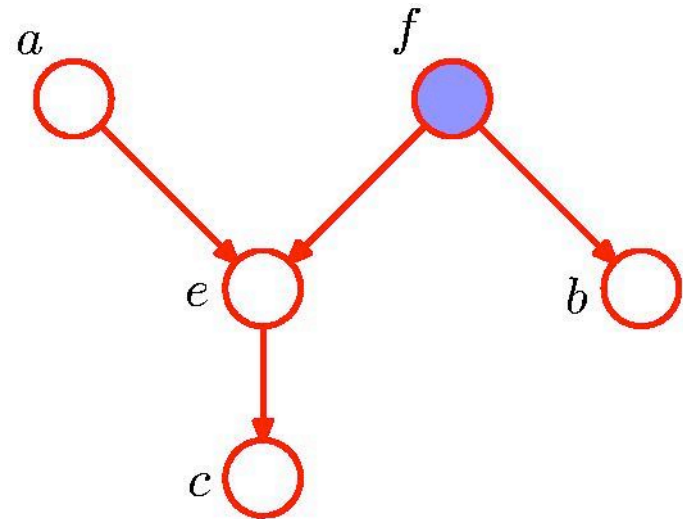
D-separation

- A , B , and C are non-intersecting subsets of nodes in a directed graph.
 - A path from A to B is blocked if it contains a node such that either
 - a) the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C , or
 - b) the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in the set C .
 - If all paths from A to B are blocked, A is said to be d-separated from B by C .
 - If A is d-separated from B by C , the joint distribution over all variables in the graph satisfies $A \perp\!\!\!\perp B \mid C$.
-

D-separation: Example

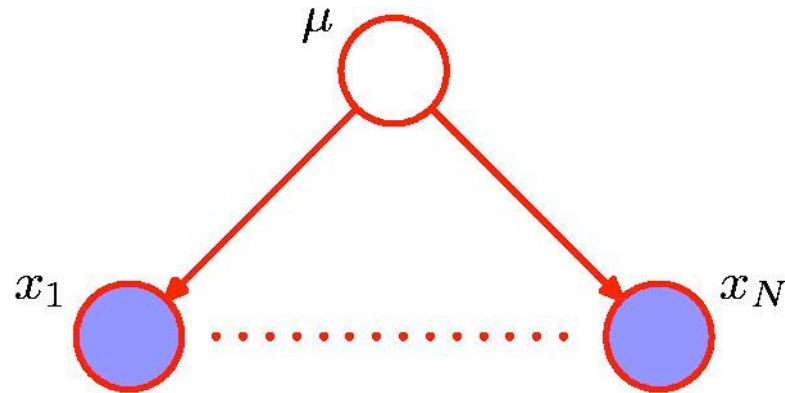


$a \not\perp b \mid c$



$a \perp b \mid f$

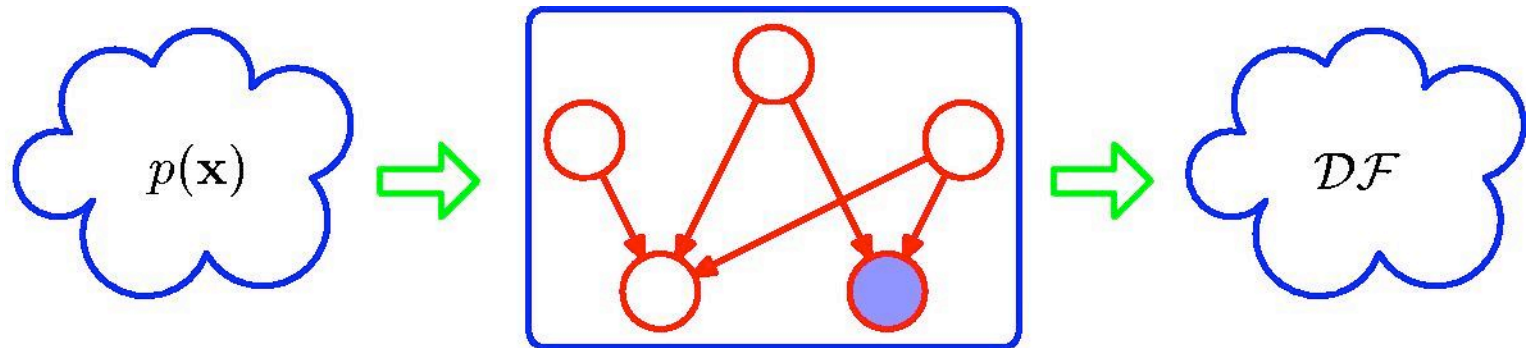
D-separation: I.I.D. Data



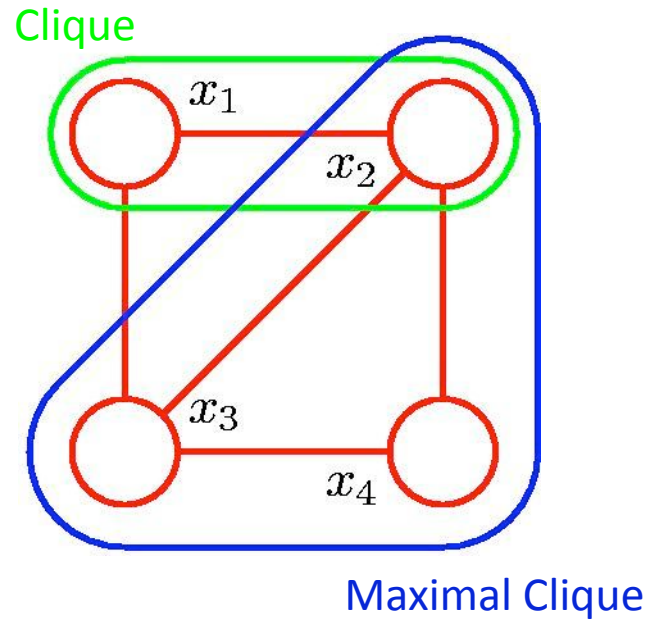
$$p(\mathcal{D}|\mu) = \prod_{n=1}^N p(x_n|\mu)$$

$$p(\mathcal{D}) = \int_{-\infty}^{\infty} p(\mathcal{D}|\mu)p(\mu) d\mu \neq \prod_{n=1}^N p(x_n)$$

Directed Graphs as Distribution Filters



Cliques and Maximal Cliques



Joint Distribution

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

where $\psi_C(\mathbf{x}_C)$ is the potential over clique C and

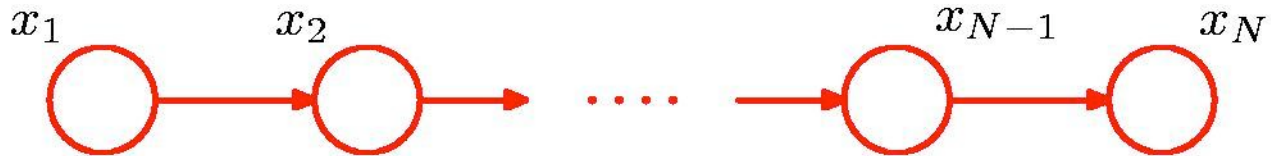
$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

is the normalization coefficient; note: M K -state variables $\rightarrow K^M$ terms in Z .

Energies and the Boltzmann distribution

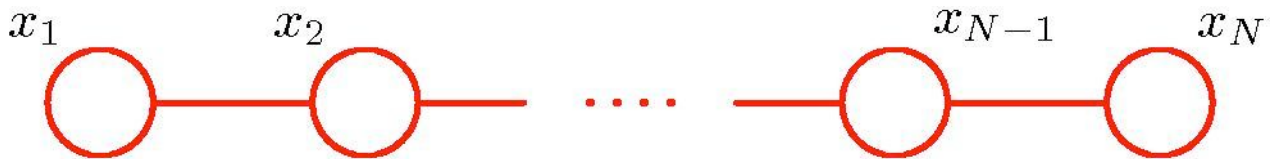
$$\psi_C(\mathbf{x}_C) = \exp \{-E(\mathbf{x}_C)\}$$

Converting Directed to Undirected Graphs (1)



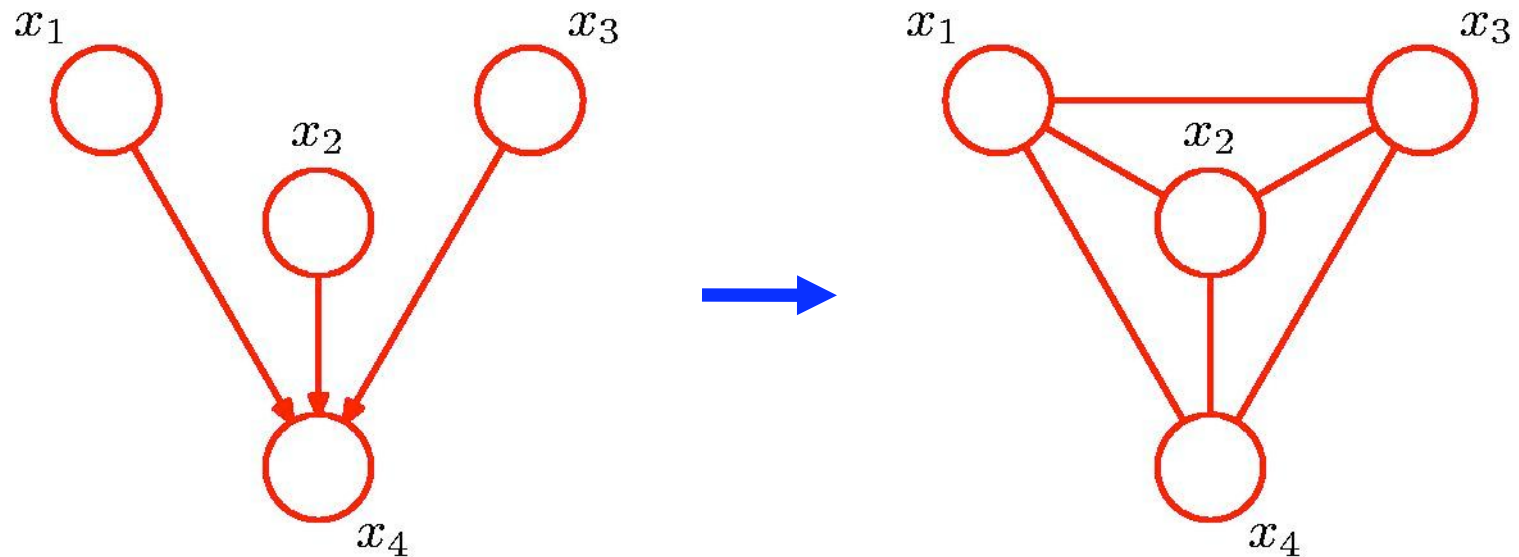
$$p(\mathbf{x}) = p(x_1) p(x_2|x_1) p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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



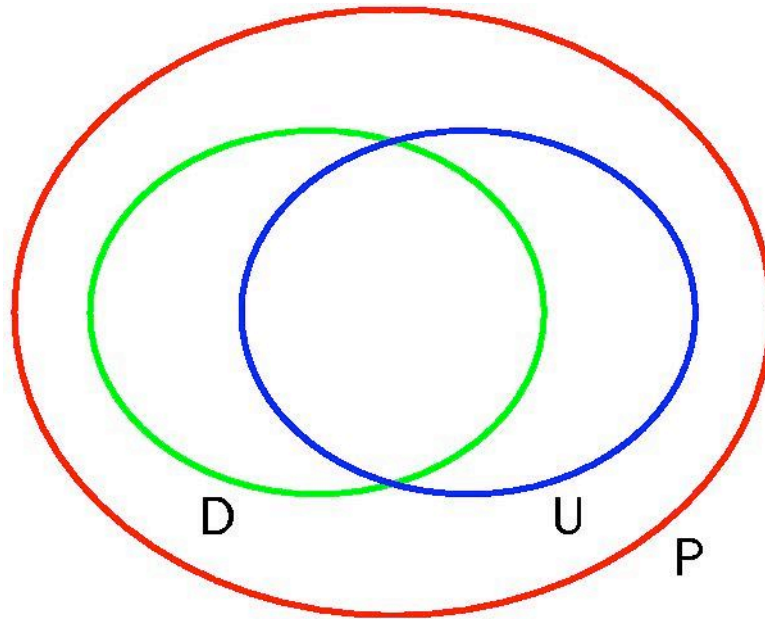
Converting Directed to Undirected Graphs (2)

Additional links

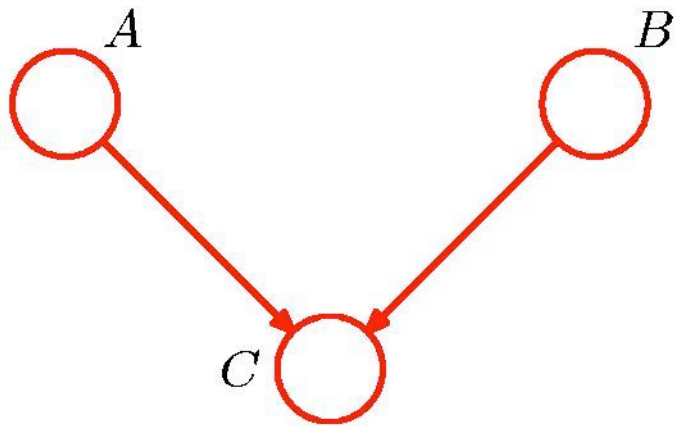


$$\begin{aligned} p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\ &= \frac{1}{Z} \psi_A(x_1, x_2, x_3) \psi_B(x_2, x_3, x_4) \psi_C(x_1, x_2, x_4) \end{aligned}$$

Directed vs. Undirected Graphs (1)

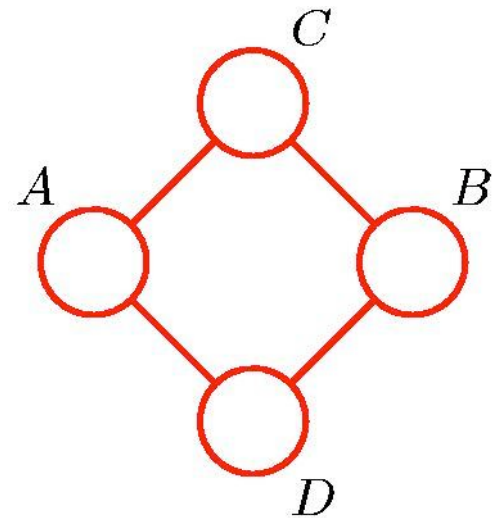


Directed vs. Undirected Graphs (2)



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

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

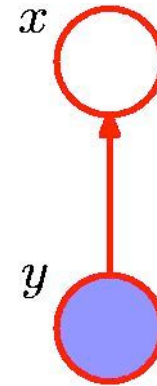
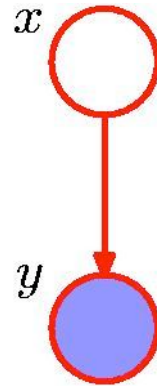
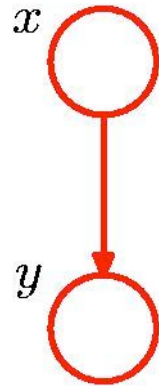


$$A \not\perp\!\!\!\perp B \mid \emptyset$$

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

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

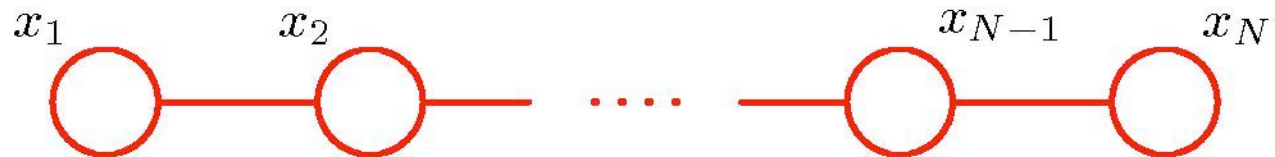
Inference in Graphical Models



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

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

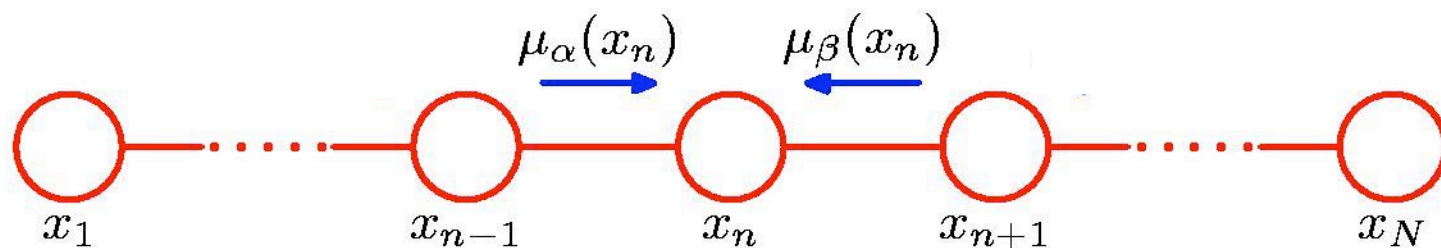
Inference on a Chain



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

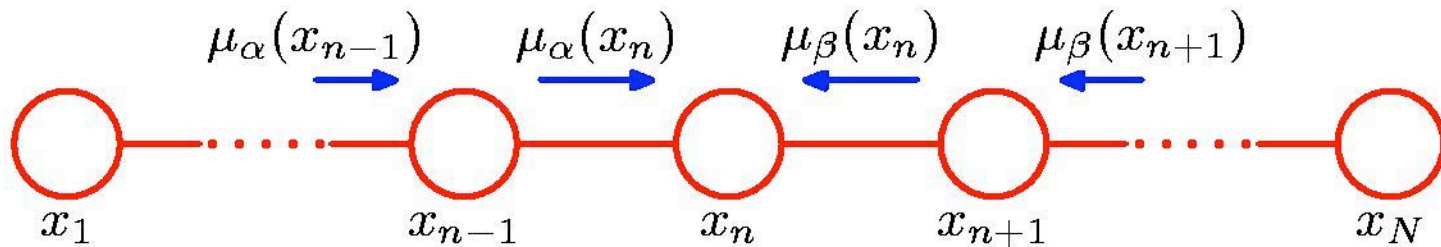
$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_N} p(\mathbf{x})$$

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



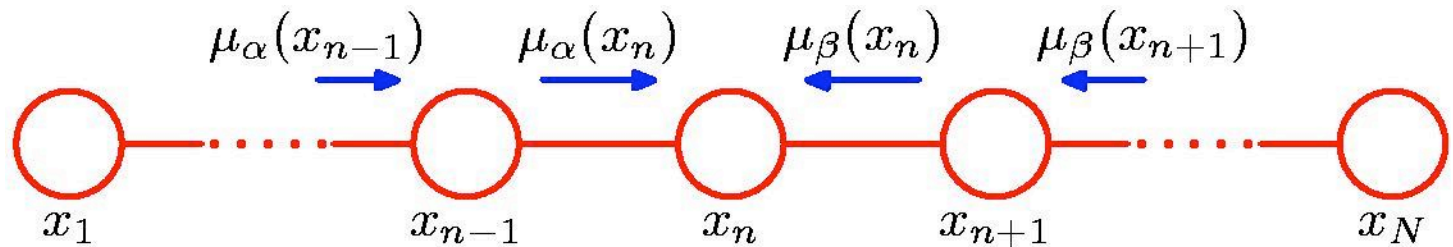
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$

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

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

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

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:

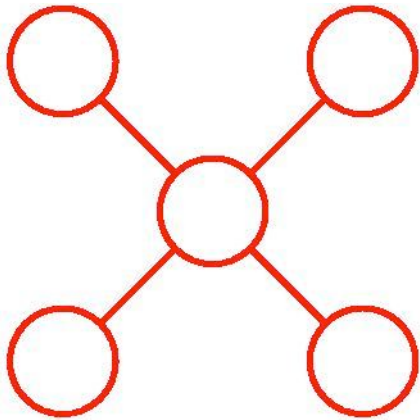
- Compute and store all forward messages, $\mu_\alpha(x_n)$.
- Compute and store all backward messages, $\mu_\beta(x_n)$.
- Compute Z at any node x_m
- Compute

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

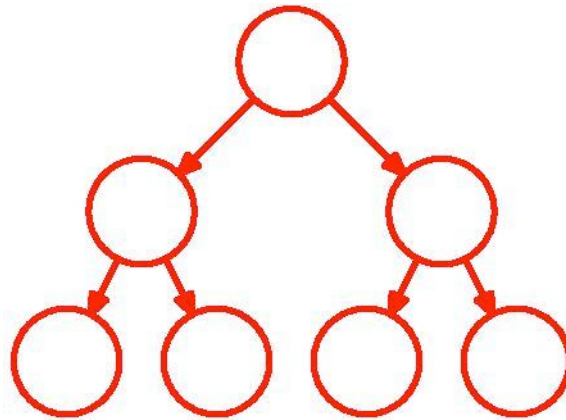
for all variables required.

Trees

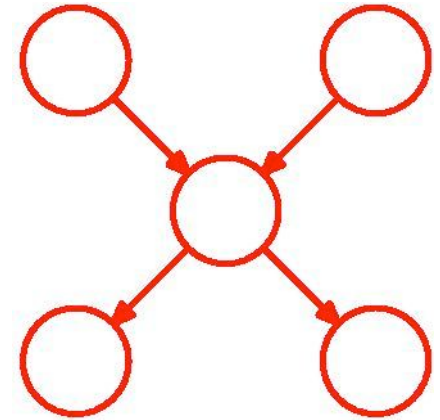
Undirected Tree



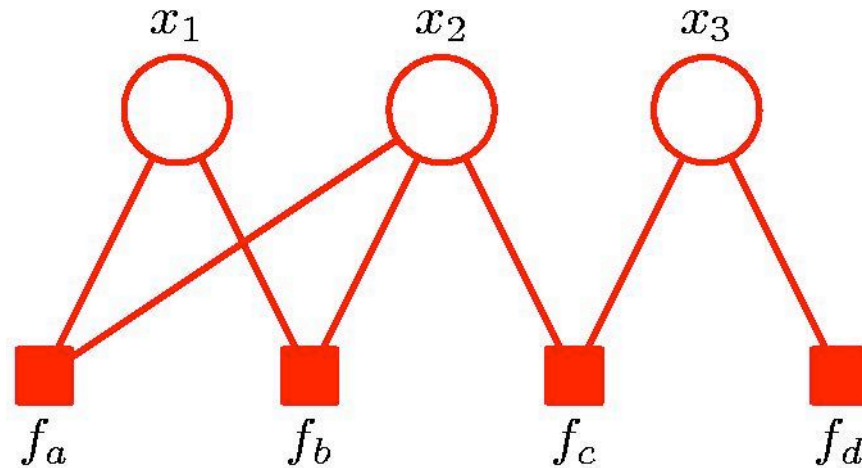
Directed Tree



Polytree



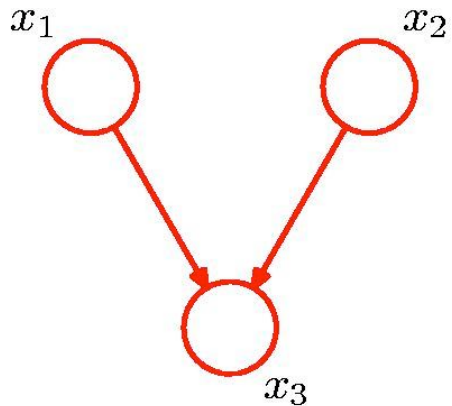
Factor Graphs



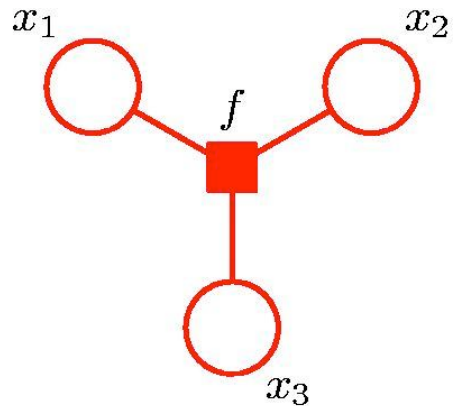
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

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

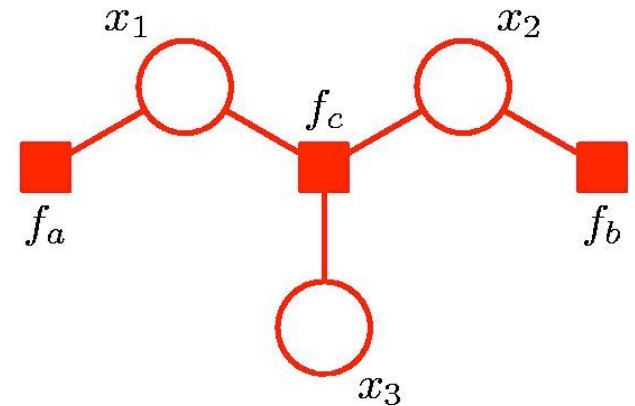
Factor Graphs from Directed Graphs



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



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

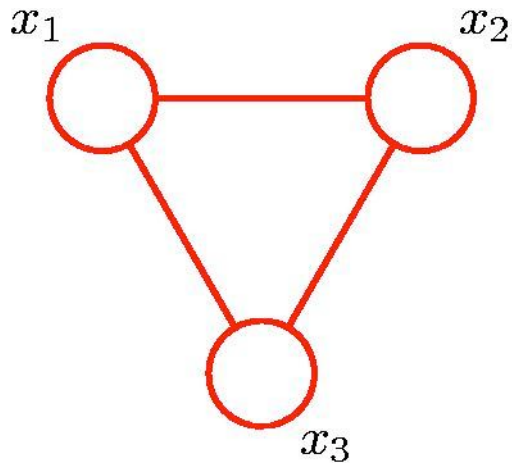


$$f_a(x_1) = p(x_1)$$

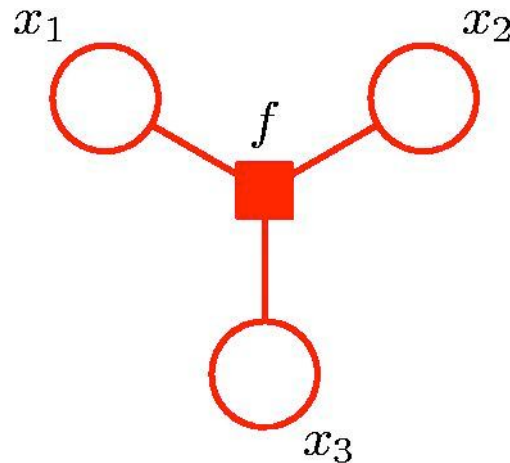
$$f_b(x_2) = p(x_2)$$

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

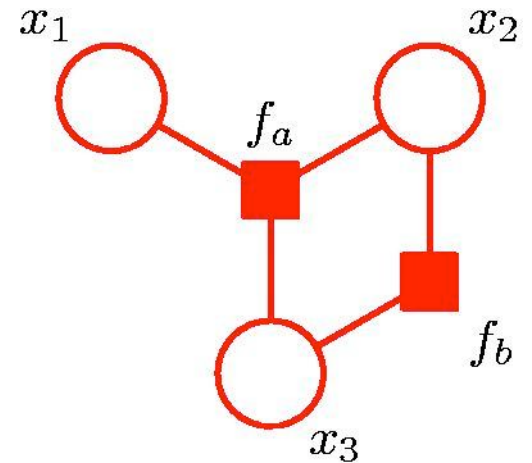
Factor Graphs from Undirected Graphs



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



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



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

The Sum-Product Algorithm (1)

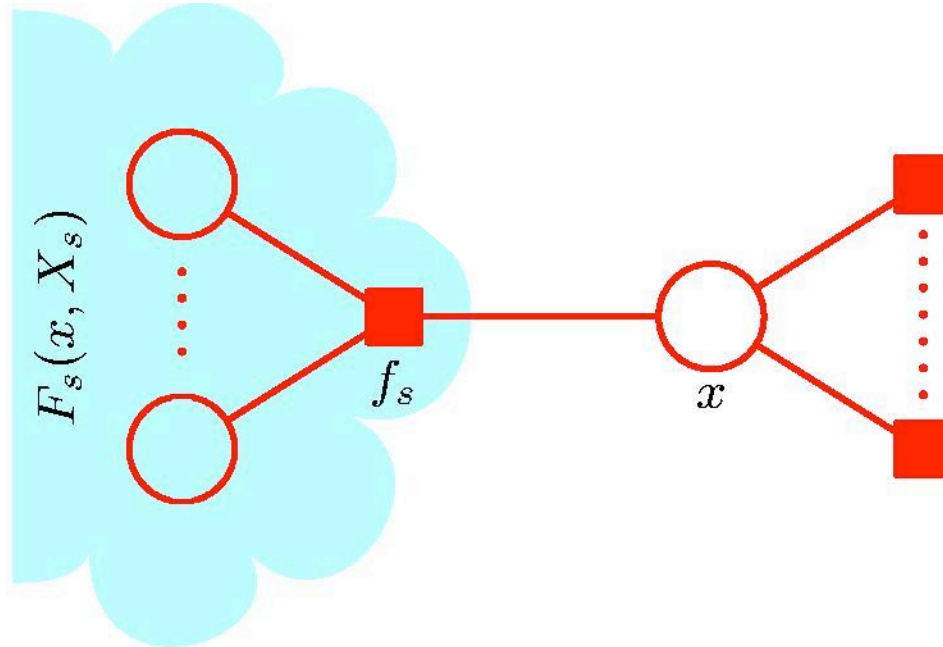
Objective:

- 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

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

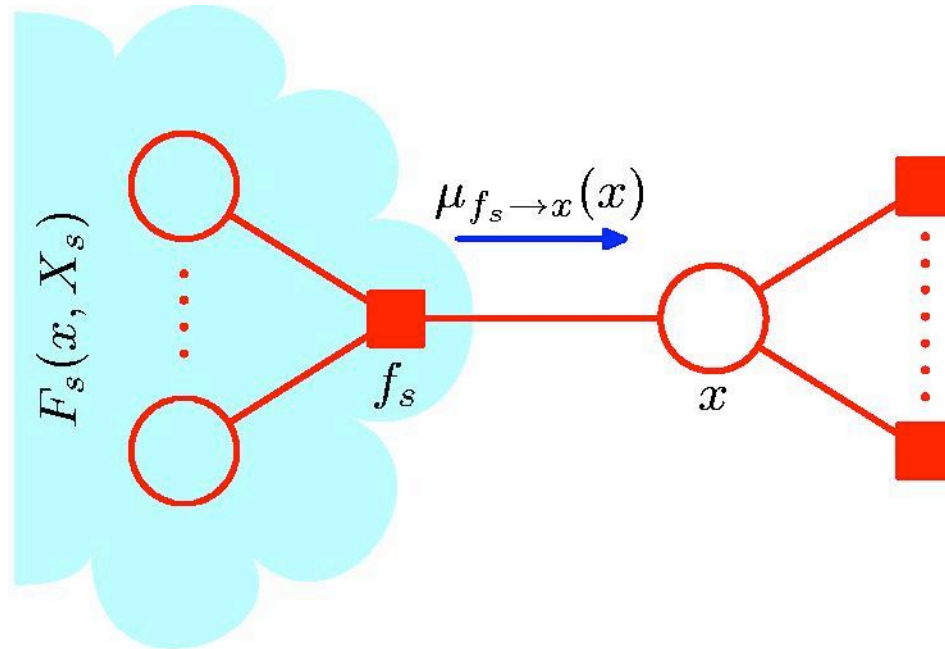
The Sum-Product Algorithm (2)



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

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

The Sum-Product Algorithm (3)

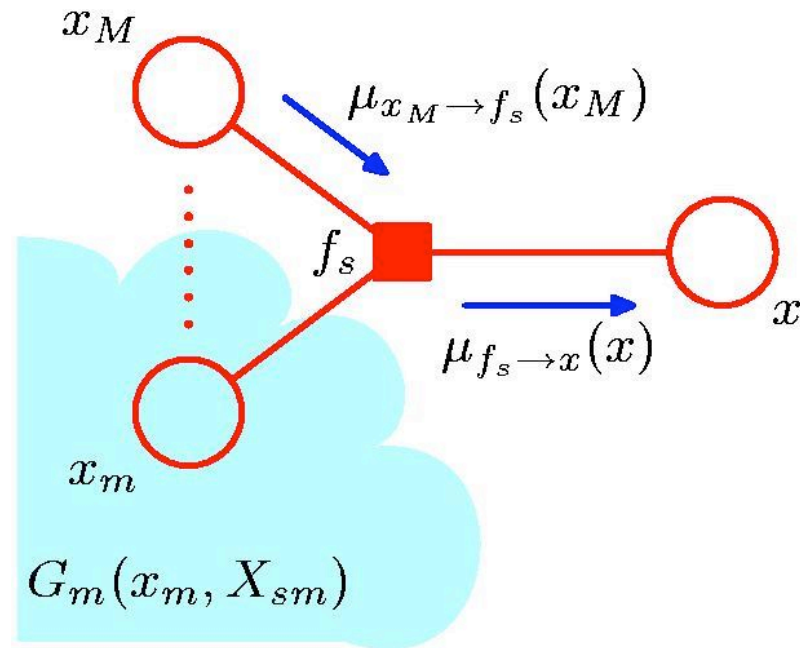


$$p(x) = \prod_{s \in \text{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right]$$

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

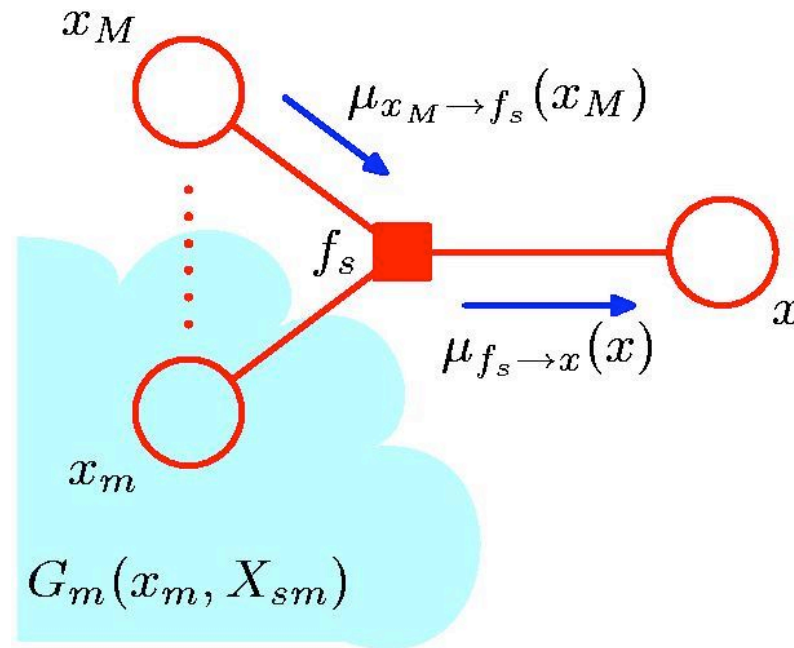
$$\mu_{f_s \rightarrow x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

The Sum-Product Algorithm (4)



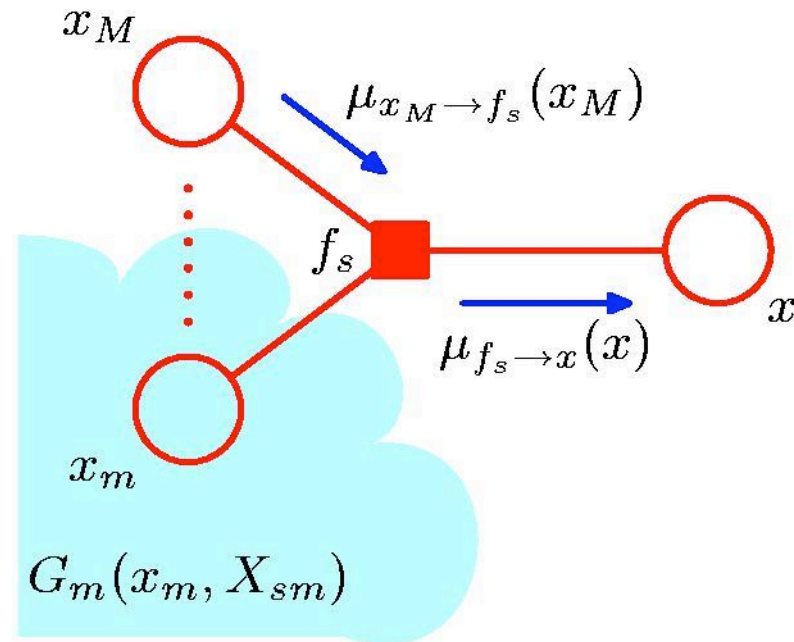
$$F_s(x, X_s) = f_s(x, x_1, \dots, x_M) G_1(x_1, X_{s1}) \dots G_M(x_M, X_{sM})$$

The Sum-Product Algorithm (5)



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

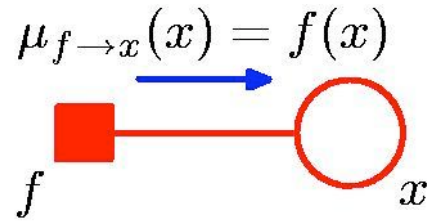
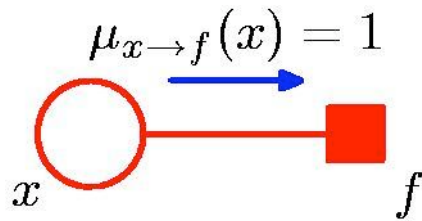
The Sum-Product Algorithm (6)



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

The Sum-Product Algorithm (7)

Initialization

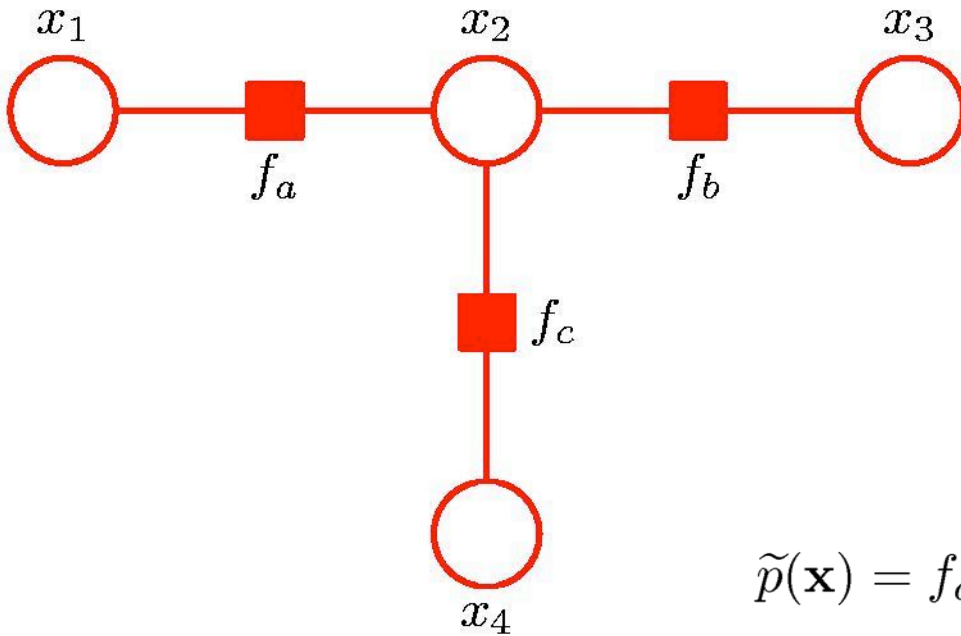


The Sum-Product Algorithm (8)

To compute local marginals:

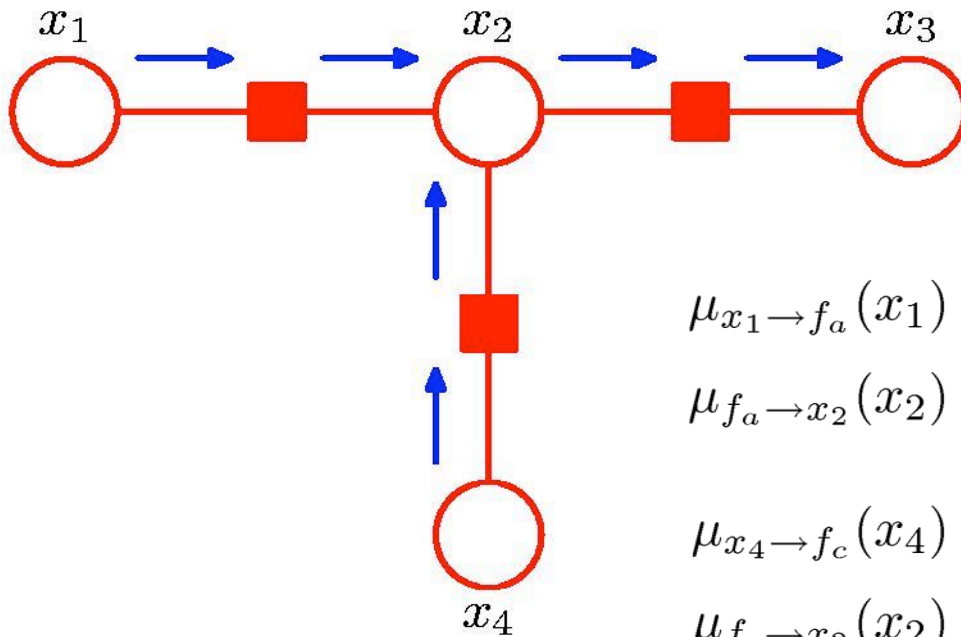
- Pick an arbitrary node as root
 - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
 - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
 - Compute the product of received messages at each 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)



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

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

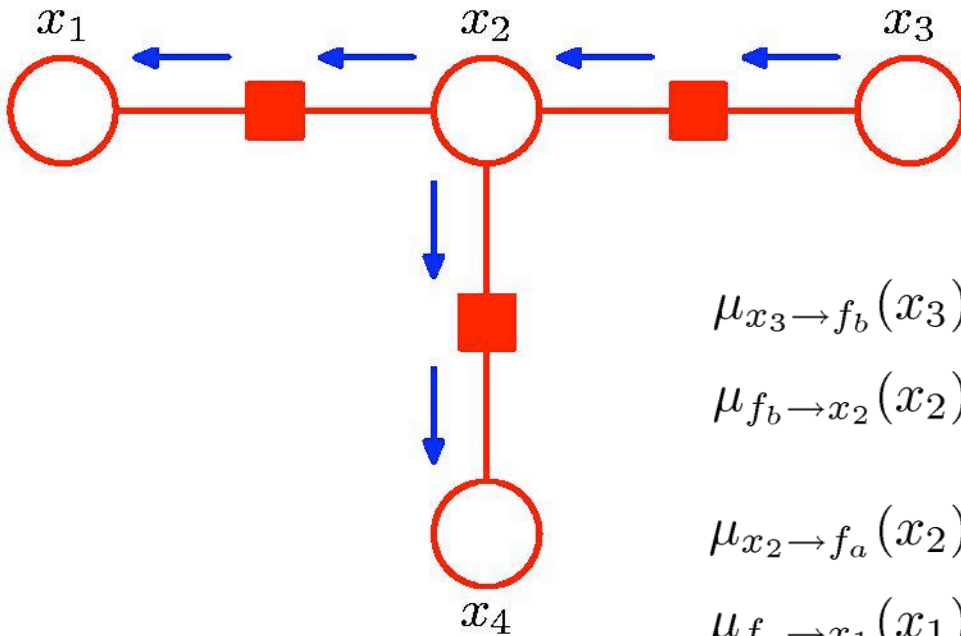
$$\mu_{x_4 \rightarrow f_c}(x_4) = 1$$

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

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

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

Sum-Product: Example (3)



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

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

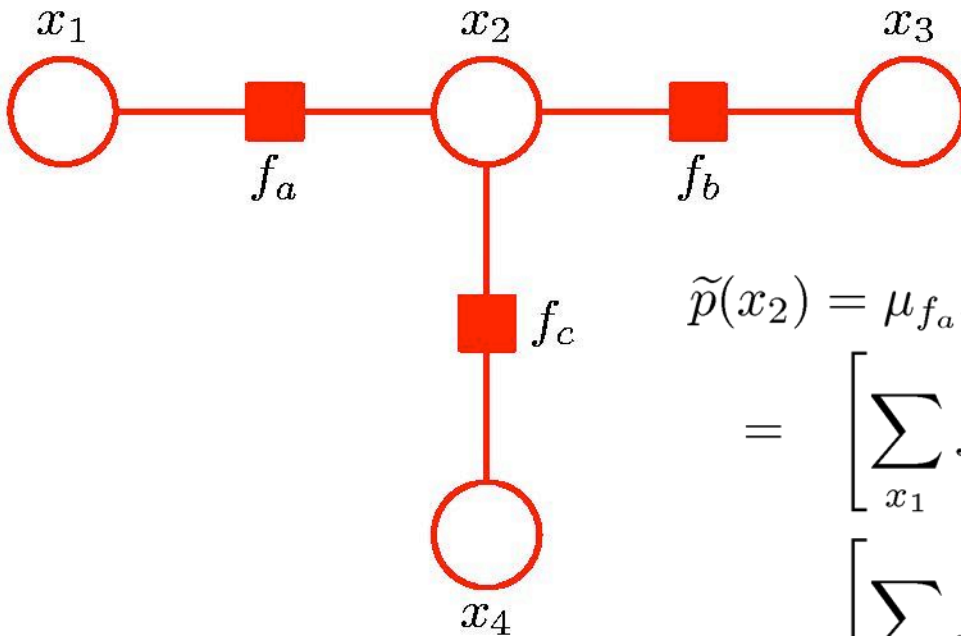
$$\mu_{x_2 \rightarrow f_a}(x_2) = \mu_{f_b \rightarrow x_2}(x_2) \mu_{f_c \rightarrow x_2}(x_2)$$

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

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

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

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 $p(\mathbf{x}^{\max})$.

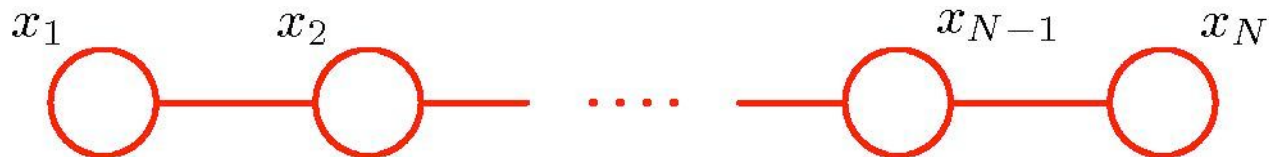
In general, maximum marginals \neq joint maximum.

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

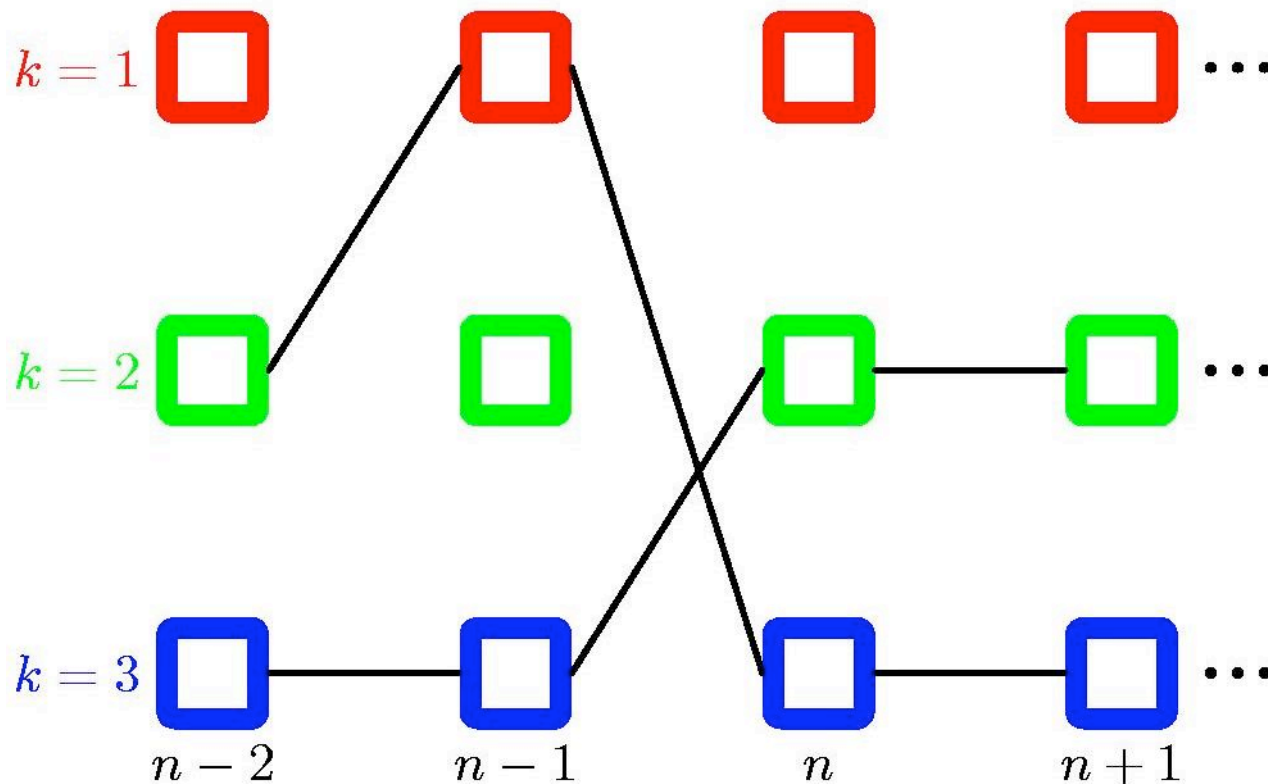
$$p^{\max} = \max_x \left[\sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right]$$
$$x^{\max} = \arg \max_x \left[\sum_{s \in \text{ne}(x)} \mu_{f_s \rightarrow x}(x) \right]$$

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.
-