Workshop report

1. Daniels report is on website
2. Don't expect to write it based on listening to one project (we had 6 only 2 was sufficient quality)
3. I suggest writing it on one presentation.
4. Include figures (from a related paper or their presentation)
5. Include references

May 8, **CODY** Machine Learning for finding oil, focusing on 1) robust seismic denoising/interpolation using structured matrix approximation 2) seismic image clustering and classification, using t-SNE(t-distributed stochastic neighbor embedding) and CNN. **Weichang Li,** Goup Leader Aramco, Houston.

May 10, **Class HW First distribution of final projects**. Ocean acoustic source tracking. Final projects. Final project is the main goal in last month. Bishop Ch 9 Mixture models

May 15, **CODY** Seismology and Machine Learning, Daniel Trugman (half class), ch 8 Graphical models

May 17, **Class HW** ch 8

May 22, Dictionary learning, Mike Bianco (half class), Bishop Ch 13

May 24, **Class HW** Bishop Ch 13

MAY 30 CODY

May 31, No Class. Workshop, Big Data and The Earth Sciences: Grand Challenges Workshop

June 5, Discuss workshop, ch13**. Spiess Hall open for project discussion 11am-.**

June 7, **Workshop report. No class**

June 12 Spiess Hall open for project discussion 9-11:30am and 2-7pm

June 16 Final report delivered. Beer time

For final project discussion every afternoon Mark and I will be available

Chapter 13 Sequential data

Final Report

Ocean source tracking X

~~Re-implement Source Localization in an Ocean Waveguide using Supervised Machine Learning~~

X-ray spectrum absorption interpretation using NN

Neural decoding

Plankton

Transfer learning and deep feature extraction for planktonic image data sets

Speaker tagger

Coral

Resturant

Amazon rainforest (Kaggle)

Myshake Seismic

High-precision indoor positioning framework for most wifi-enabled devices

Please ask questions

Mark and I available all afternoons. Just come or email for time slots.

Spiess hall 330 is open Monday 5 and 12 June. If interested I can book it at other times

Report

Rather concise than long.

Larger group can do more.

Start with some very simple example. To show your idea and that it is working.

End with showing the advanced abilities

Several figures.

Equations are nice.

Delivery Zip file (Friday 16)

Main code (not all). It should be able to run.

Report (pdf preferred).

PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER 8: GRAPHICAL MODELS

10.1.4 Graph terminology

Before we continue, we must define a few basic terms, most of which are very intuitive.

A **graph** $G = (\mathcal{V}, \mathcal{E})$ consists of a set of **nodes** or **vertices**, $\mathcal{V} = \{1, \ldots, V\}$, and a set of **edges**, $\mathcal{E} = \{(s,t) : s,t \in \mathcal{V}\}$. We can represent the graph by its **adjacency matrix**, in which we write $G(s,t) = 1$ to denote $(s,t) \in \mathcal{E}$, that is, if $s \to t$ is an edge in the graph. If $G(s,t) = 1$ iff $G(t,s) = 1$, we say the graph is **undirected**, otherwise it is **directed**. We usually assume $G(s,s) = 0$, which means there are no **self loops**.

Here are some other terms we will commonly use:

- **Parent** For a directed graph, the **parents** of a node is the set of all nodes that feed into it: $\mathrm{pa}(s) \triangleq \{t : G(t,s) = 1\}$.
- **Child** For a directed graph, the **children** of a node is the set of all nodes that feed out of it: $\mathrm{ch}(s) \triangleq \{t : G(s,t) = 1\}$.
- **Family** For a directed graph, the **family** of a node is the node and its parents, $\mathrm{fam}(s) = \{s\} \cup \mathrm{pa}(s)$.
- **Root** For a directed graph, a **root** is a node with no parents.
- **Leaf** For a directed graph, a **leaf** is a node with no children.
- **Ancestors** For a directed graph, the **ancestors** are the parents, grand-parents, etc of a node. That is, the ancestors of t is the set of nodes that connect to t via a trail: $\mathrm{anc}(t) \triangleq \{s : s \rightsquigarrow t\}$.
- **Descendants** For a directed graph, the **descendants** are the children, grand-children, etc of a node. That is, the descendants of s is the set of nodes that can be reached via trails from s: $\mathrm{desc}(s) \triangleq \{t : s \rightsquigarrow t\}$.
- **Neighbors** For any graph, we define the **neighbors** of a node as the set of all immediately connected nodes, $\mathrm{nbr}(s) \triangleq \{t : G(s,t) = 1 \vee G(t,s) = 1\}$. For an undirected graph, we

write $s \sim t$ to indicate that s and t are neighbors (so $(s,t) \in \mathcal{E}$ is an edge in the graph).

- **Degree** The **degree** of a node is the number of neighbors. For directed graphs, we speak of the **in-degree** and **out-degree**, which count the number of parents and children.
- **Cycle or loop** For any graph, we define a **cycle** or **loop** to be a series of nodes such that we can get back to where we started by following edges, $s_1 - s_2 \cdots - s_n - s_1$, $n \geq 2$. If the graph is directed, we may speak of a directed cycle. For example, in Figure 10.1(a), there are no directed cycles, but $1 \to 2 \to 4 \to 3 \to 1$ is an undirected cycle.
- **DAG** A **directed acyclic graph** or **DAG** is a directed graph with no directed cycles. See Figure 10.1(a) for an example.
- **Topological ordering** For a DAG, a **topological ordering** or **total ordering** is a numbering of the nodes such that parents have lower numbers than their children. For example, in Figure 10.1(a), we can use $(1,2,3,4,5)$, or $(1,3,2,5,4)$, etc.
- **Path or trail** A **path** or **trail** $s \rightsquigarrow t$ is a series of directed edges leading from s to t.
- **Tree** An undirected **tree** is an undirectecd graph with no cycles. A directed tree is a DAG in which there are no directed cycles. If we allow a node to have multiple parents, we call it a **polytree**, otherwise we call it a moral directed tree.
- **Forest** A **forest** is a set of trees.
- **Subgraph** A (node-induced) **subgraph** G_A is the graph created by using the nodes in A and their corresponding edges, $G_A = (\mathcal{V}_A, \mathcal{E}_A)$.
- **Clique** For an undirected graph, a **clique** is a set of nodes that are all neighbors of each other. A **maximal clique** is a clique which cannot be made any larger without losing the clique property. For example, in Figure 10.1(b), $\{1,2\}$ is a clique but it is not maximal, since we can add 3 and still maintain the clique property. In fact, the maximal cliques are as follows: $\{1,2,3\}$, $\{2,3,4\}$, $\{3,5\}$.

Three types of graphical model

Directed graphs

- useful for designing models

Undirected graphs

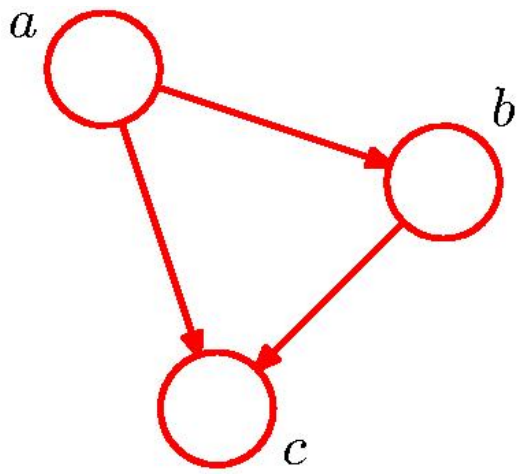
- good for some domains, e.g. computer vision

Factor graphs

- useful for inference and learning

Bayesian Networks (Bayes Nets) or Directed graphical model (DGM)

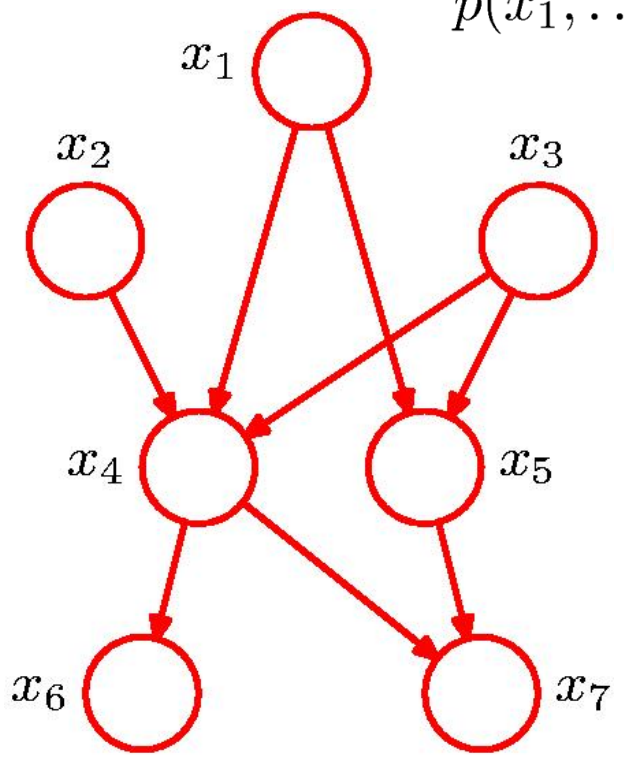
Decomposition



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

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

Directed Graphs or Bayesian Networks

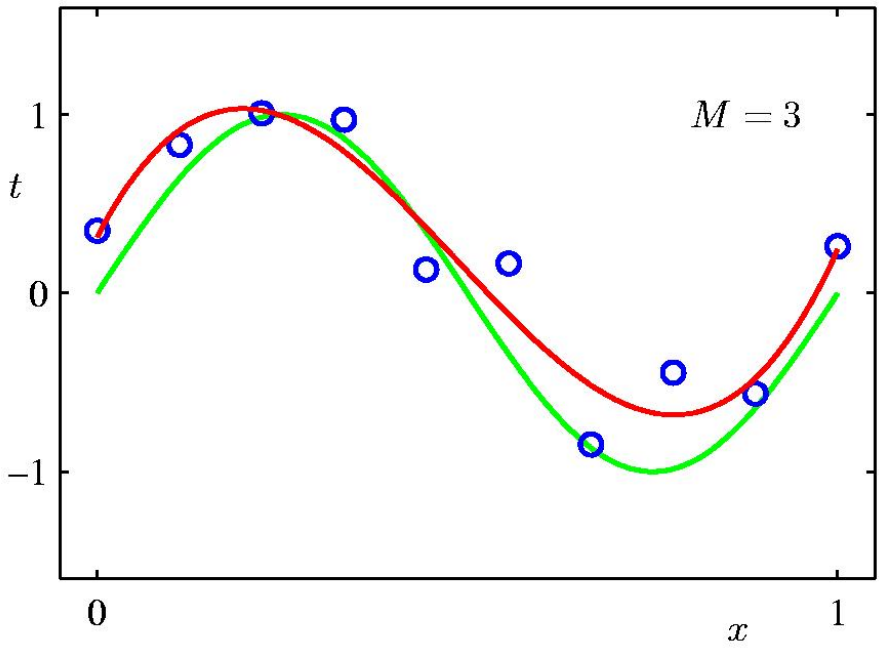


$$p(x_1,\ldots,x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)$$
$$p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5)$$

General Factorization

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

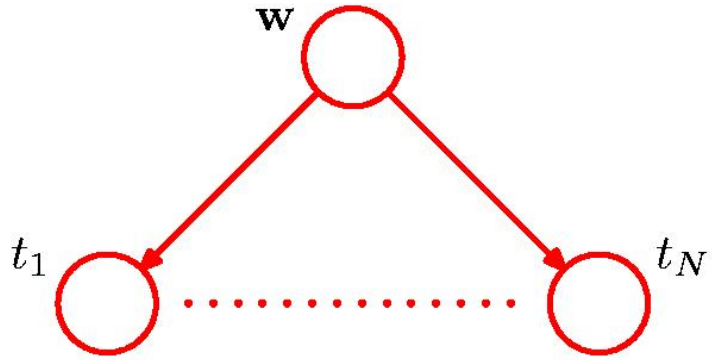
Bayesian Curve Fitting (1)

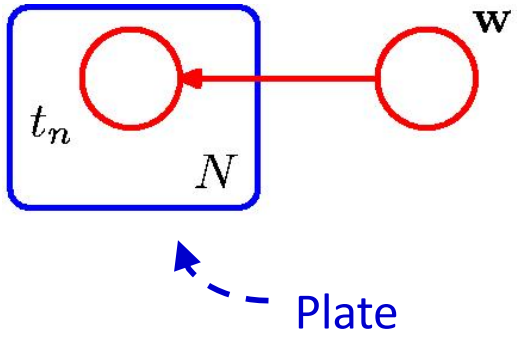


Polynomial

$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$$

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

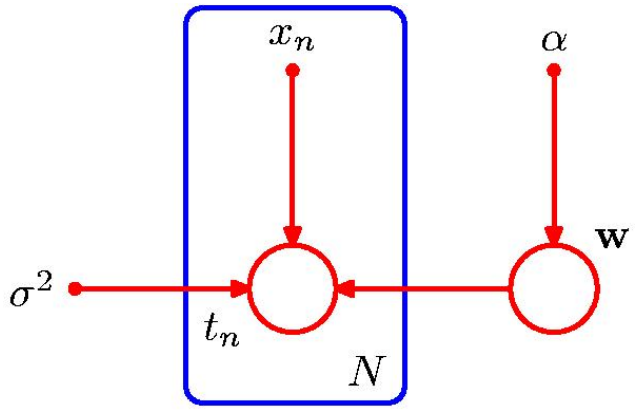




Bayesian Curve Fitting (3)

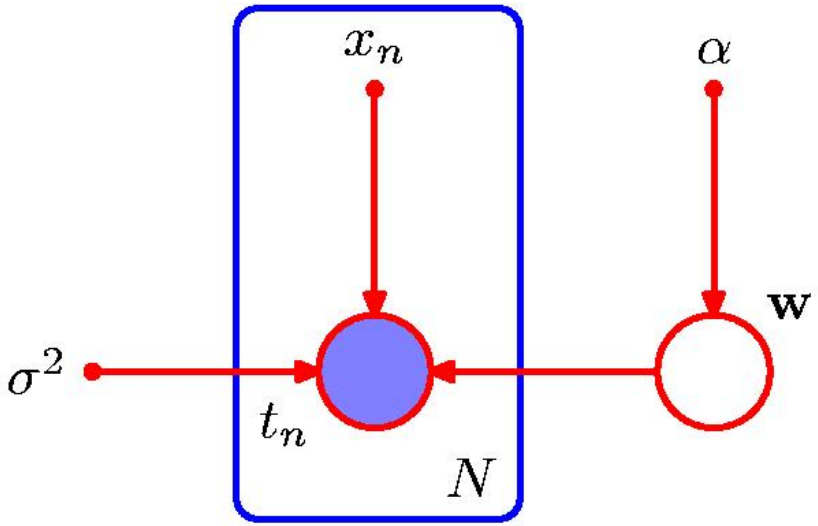
Input variables and explicit hyperparameters

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



Condition on data

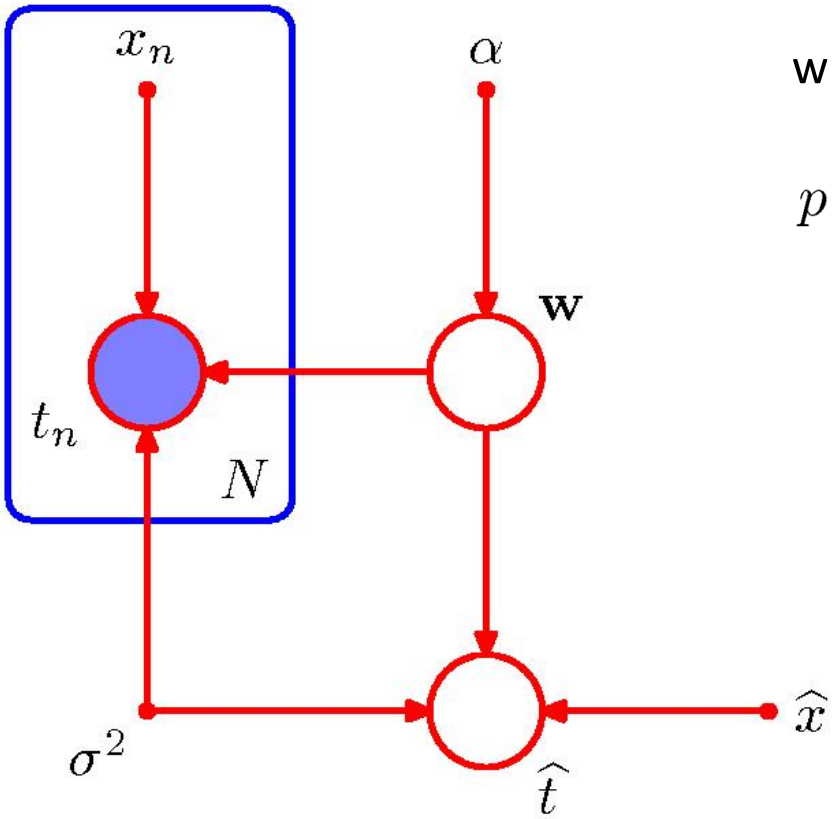
$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|\mathbf{w})$$



Bayesian Curve Fitting —Prediction

Predictive distribution:

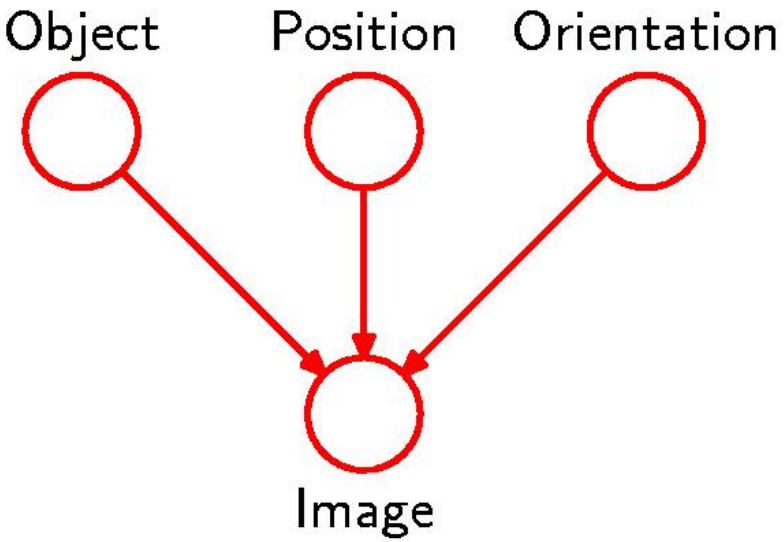
$$p(\widehat{t}|\widehat{x}, \mathbf{x}, \mathbf{t}, \alpha, \sigma^2) \propto \int p(\widehat{t}, \mathbf{t}, \mathbf{w}|\widehat{x}, \mathbf{x}, \alpha, \sigma^2)\, \mathrm{d}\mathbf{w}$$



where

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

Causal process for generating images



Discrete Variables (1)

General joint distribution: K^2-1 parameters

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

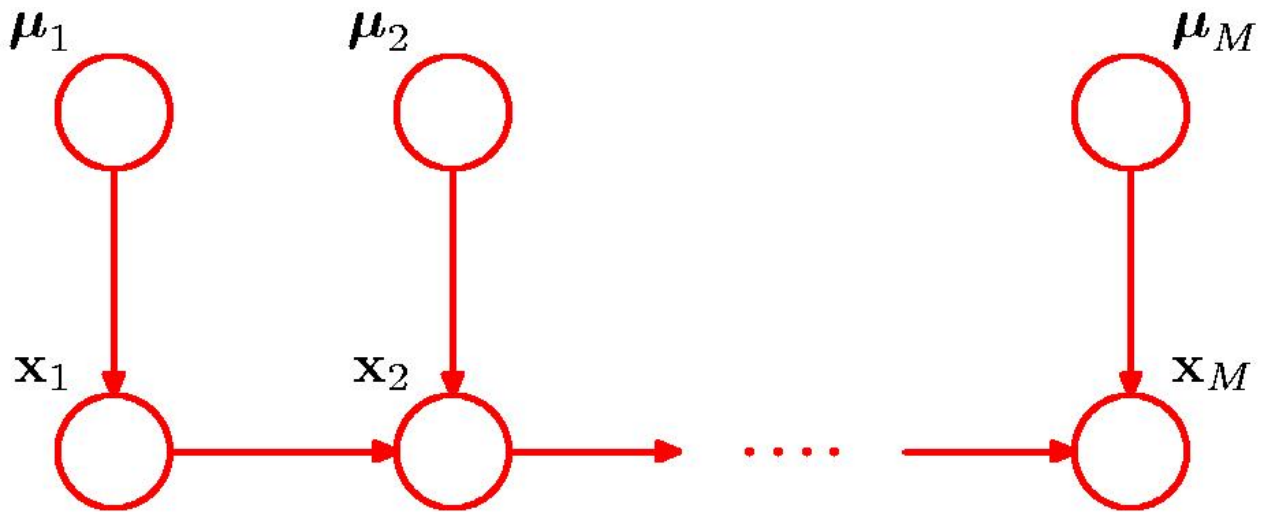
Independent joint distribution: 2(K-1) parameters

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

General joint distribution over M variables: K^M -1parameters

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

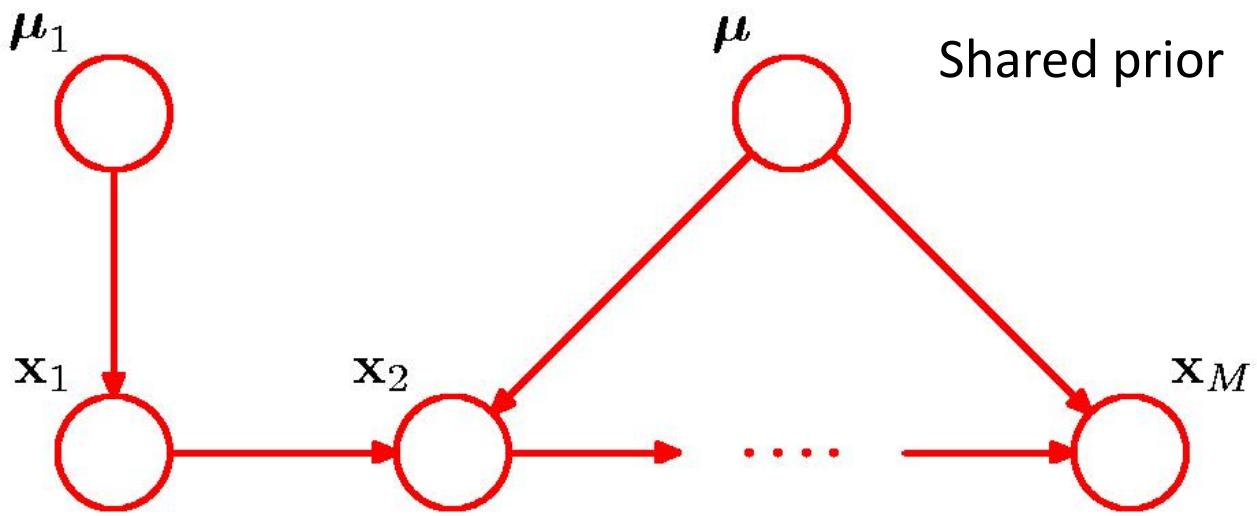
Discrete Variables: Bayesian Parameters (1)



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

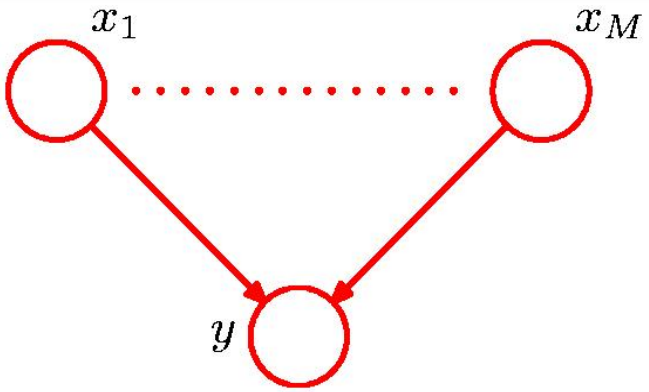
$$p(\boldsymbol{\mu}_m) = \mathrm{Dir}(\boldsymbol{\mu}_m|\boldsymbol{\alpha}_m)$$

Discrete Variables: Bayesian Parameters (2)



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

Parameterized Conditional Distributions



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

The parameterized form

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

requires only $M + 1$ parameters

Conditional Independence

a is independent of b given c

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

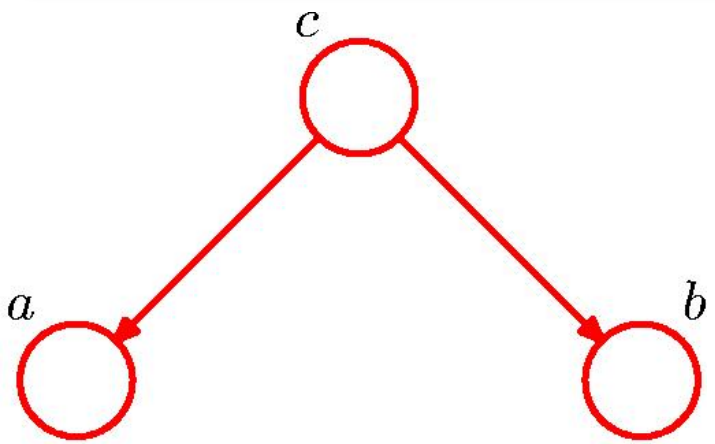
Equivalently

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

Notation

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

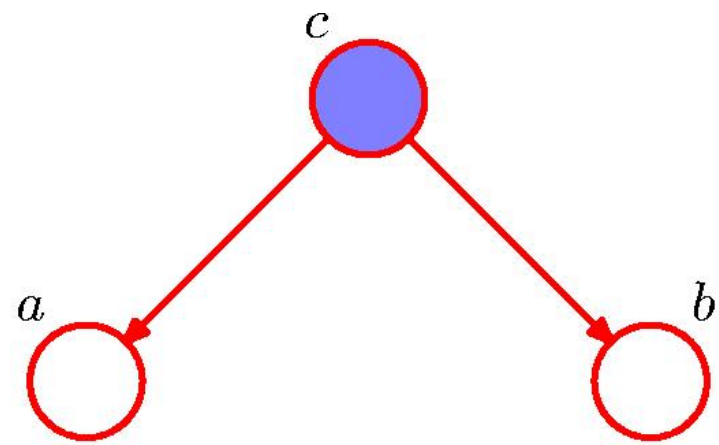
Conditional Independence: Example 1



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

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

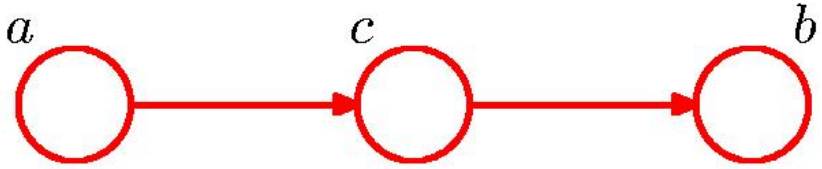
$$a \not\perp\!\!\!\perp b \mid \emptyset$$



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

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

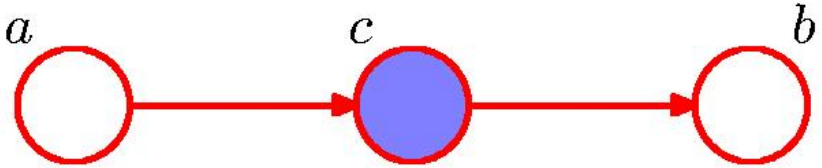
Conditional Independence: Example 2



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

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

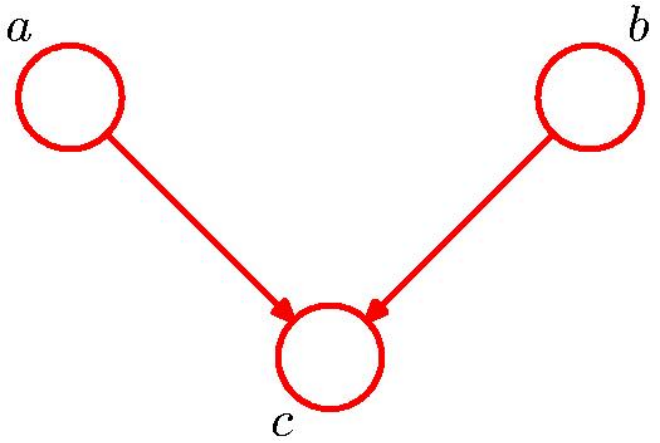
$$a \not\perp\!\!\!\perp b \mid \emptyset$$



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

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

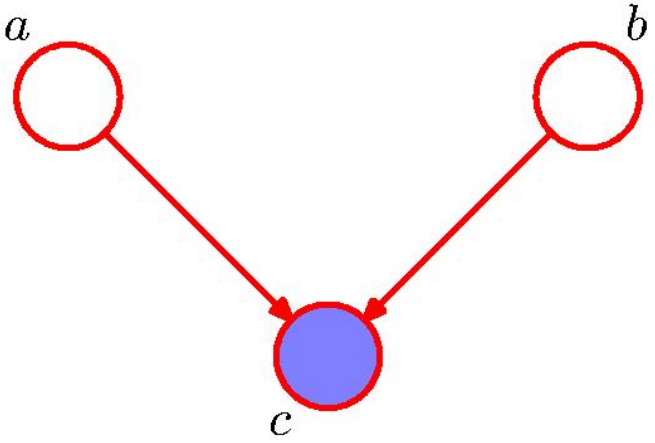
Conditional Independence: Example 3



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

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

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



$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(b)p(c|a, b)}{p(c)}$$

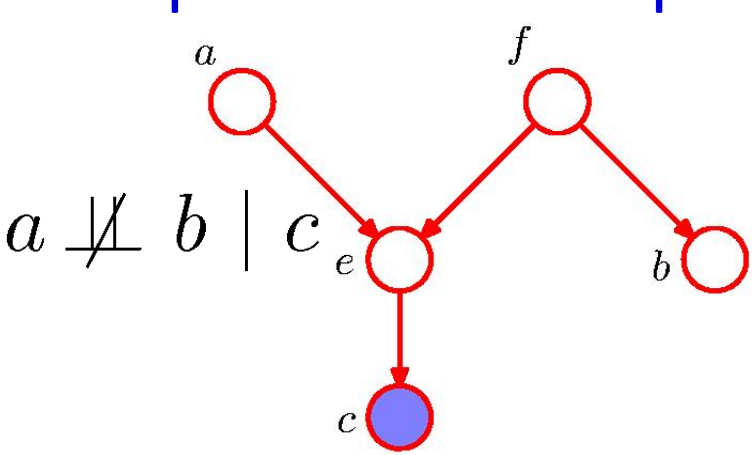
$$a \not\perp\!\!\!\perp b \mid c$$

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

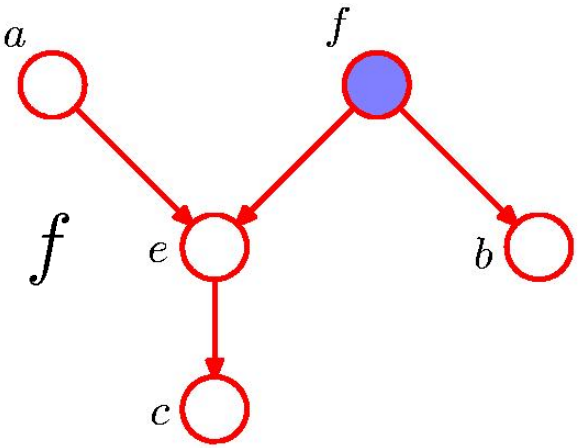
D-separation

- A, B, and C are non-intersecting subsets of nodes in a directed graph.
- 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. Then the joint distribution over all variables satisfies $A \perp\!\!\!\perp B \mid C$

D-separation: Example

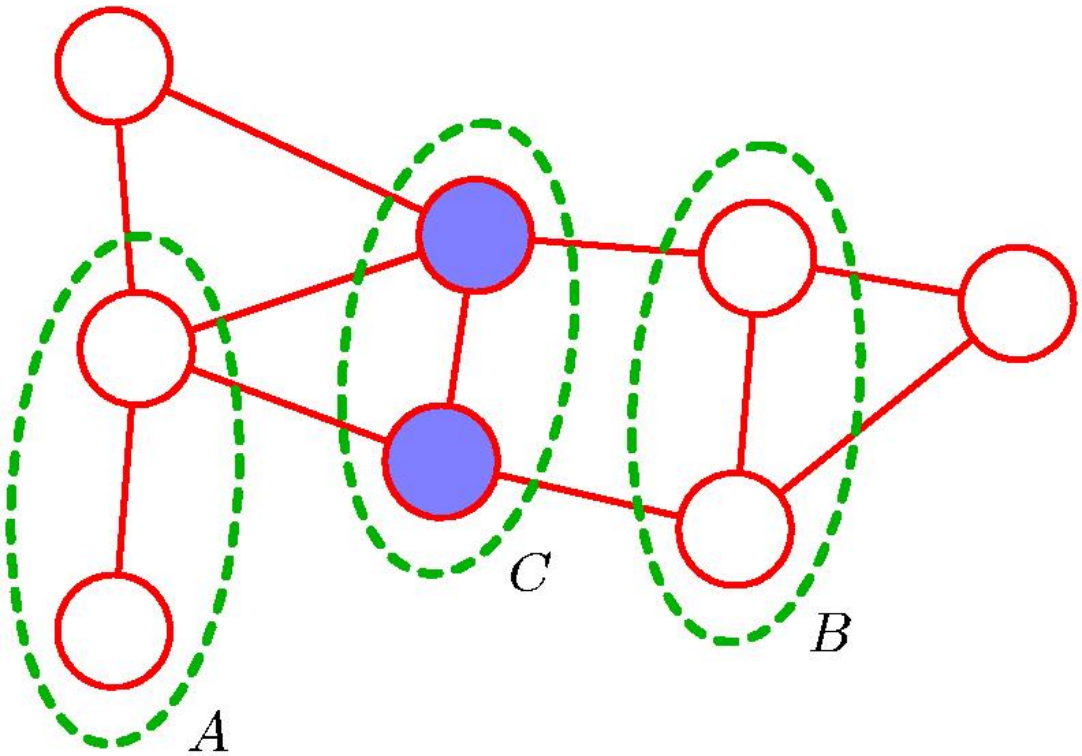


$a \not\perp\!\!\!\perp b \mid c$



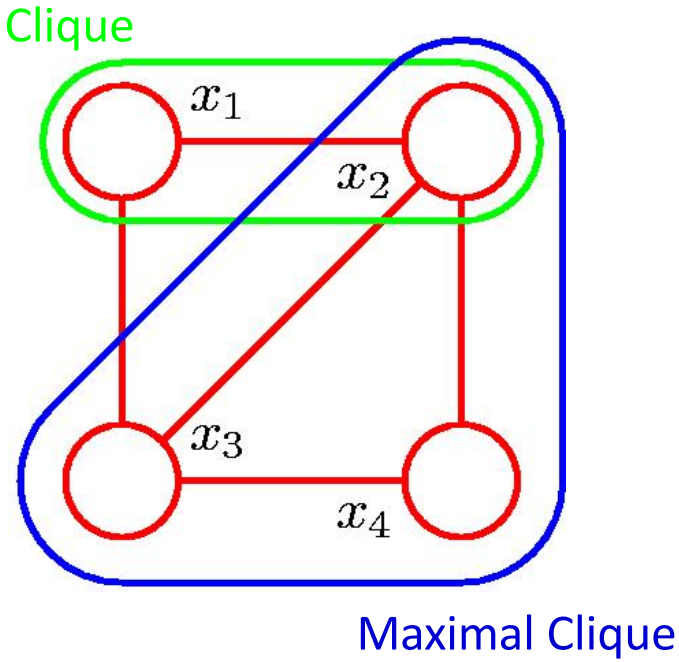
$a \perp\!\!\!\perp b \mid f$

Markov Random Fields or Undirected Graphs



$$A \perp\!\!\!\perp B | C$$

Cliques and Maximal Cliques



Joint Distribution

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

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

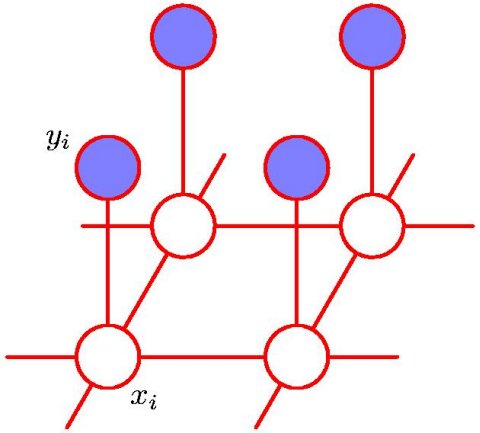
is the normalization coefficient; note: M K-state variables $\rightarrow \mathrm{K}^{\mathrm{M}}$ terms in Z.

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

Energies and the Boltzmann distribution

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

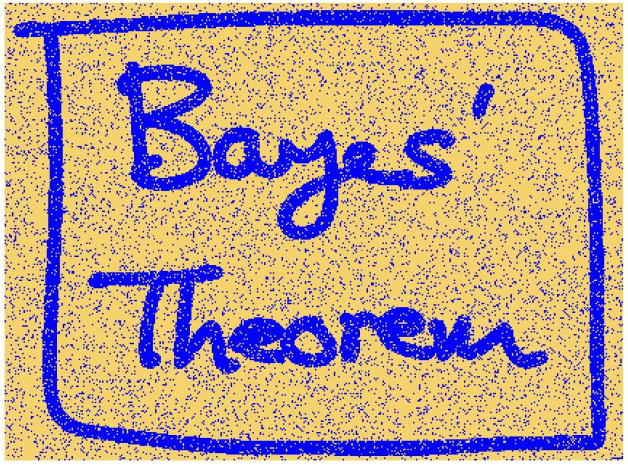
Illustration: Image De-Noising



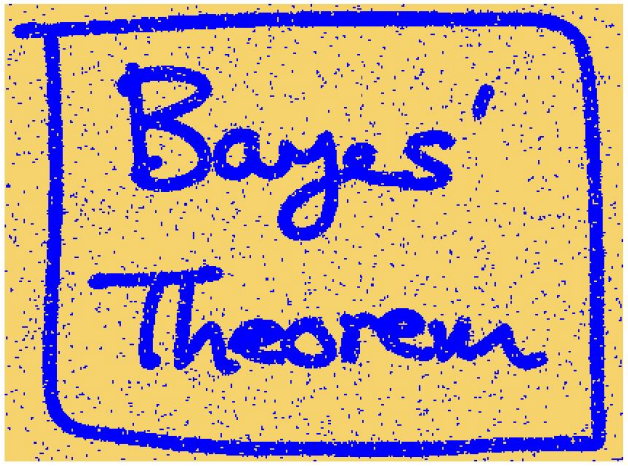
$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i$$

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}$$

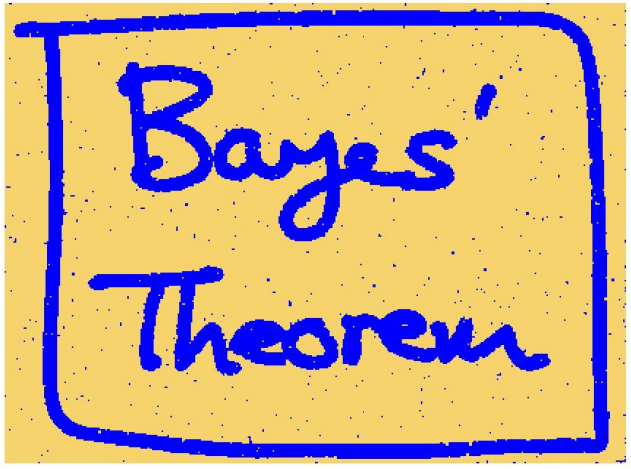
Noisy Image



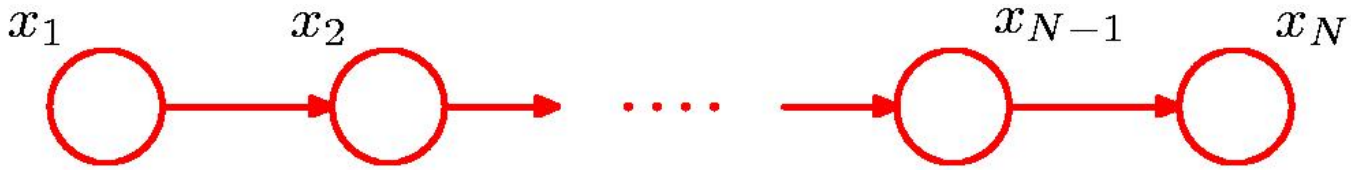
Restored Image (ICM)



Restored Image (Graph cuts)

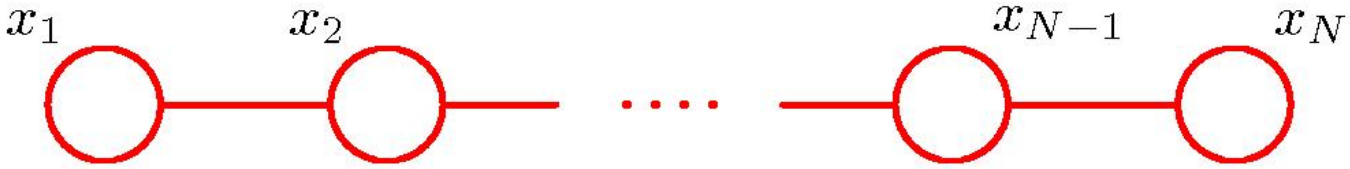


Converting Directed to Undirected Graphs (1)



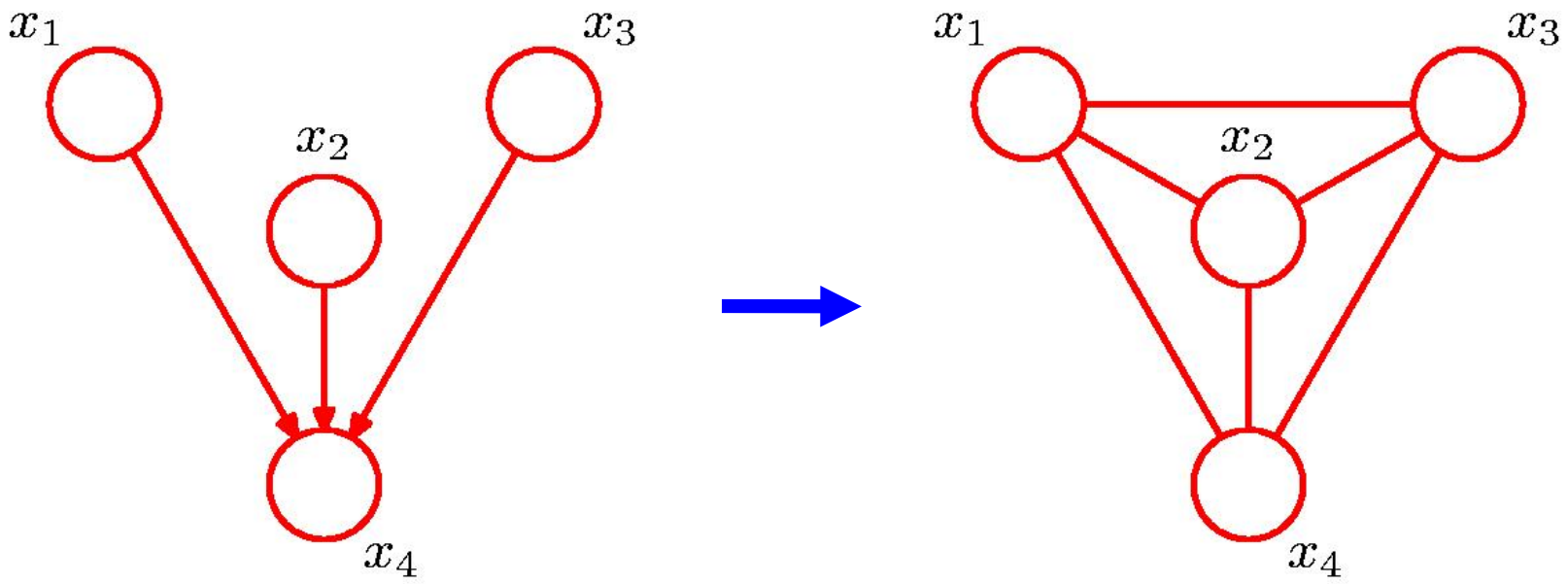
$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)\; p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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



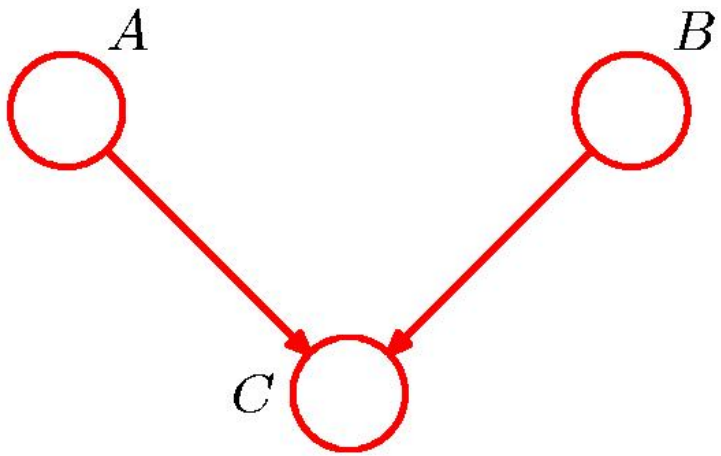
Converting Directed to Undirected Graphs (2)

Additional links



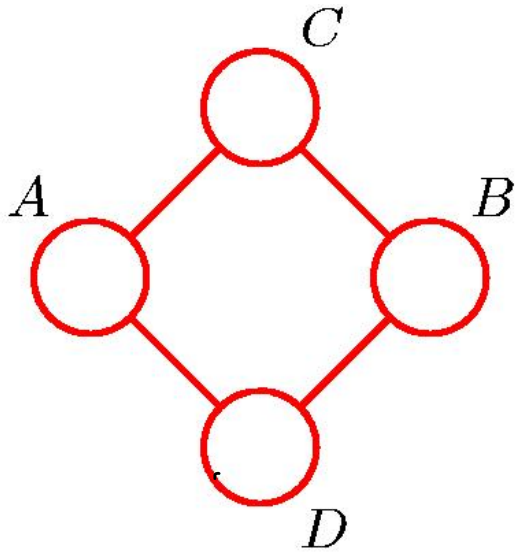
$$
\begin{aligned}
p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3) \\
&= \frac{1}{Z}\psi_A(x_1,x_2,x_3)\psi_B(x_2,x_3,x_4)\psi_C(x_1,x_2,x_4)
\end{aligned}
$$

Directed vs. Undirected Graphs (2)



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

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

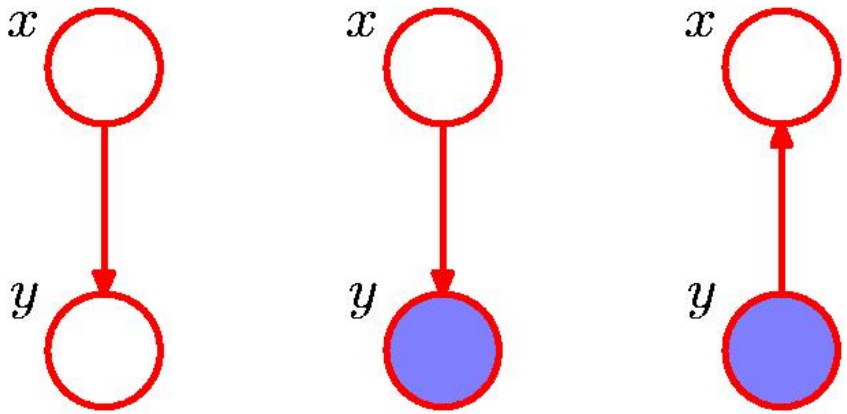


$$A \not\perp\!\!\!\perp B \mid \emptyset$$

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

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

Inference in Graphical Models



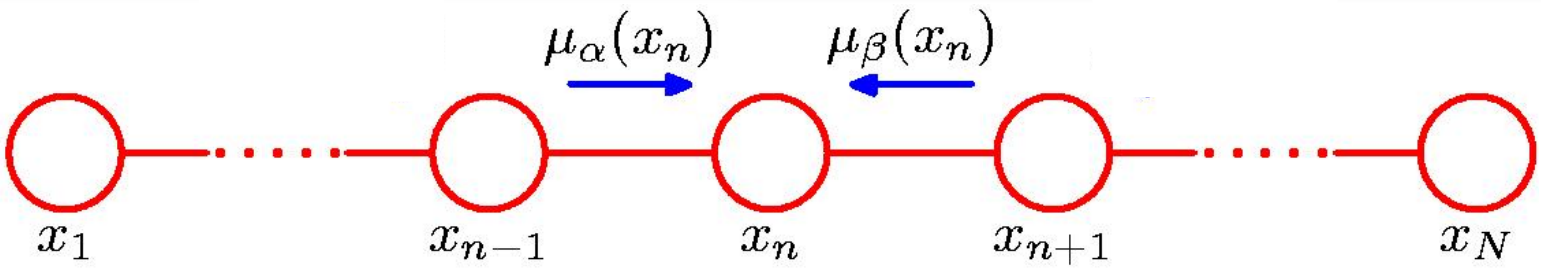
$$p(y) = \sum_{x'} p(y|x')p(x') \qquad p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Inference on a Chain

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

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

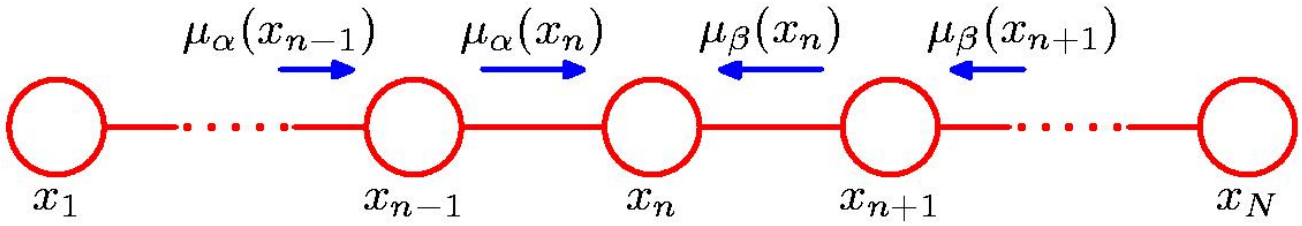
Inference on a Chain



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

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

Inference on a Chain



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

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

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

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

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

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

Inference on a Chain

To compute local marginals:

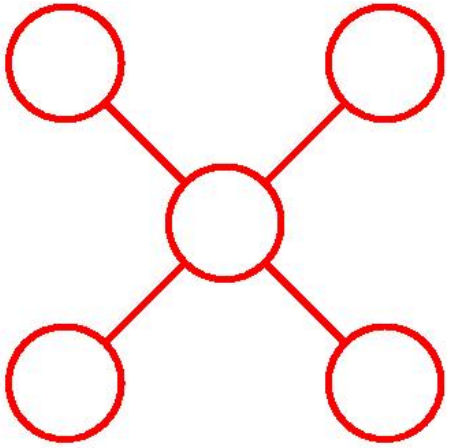
- Compute and store all forward messages, $\mu_\alpha(x_n)$
- Compute and store all backward messages, $\mu_\beta(x_n)$
- Compute Z at any node x_m
- Compute

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

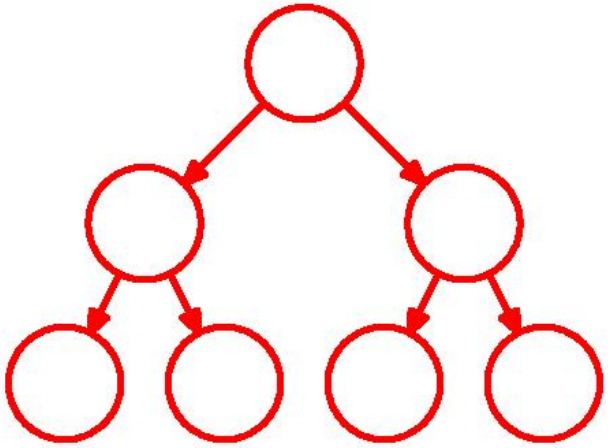
for all variables required.

Trees

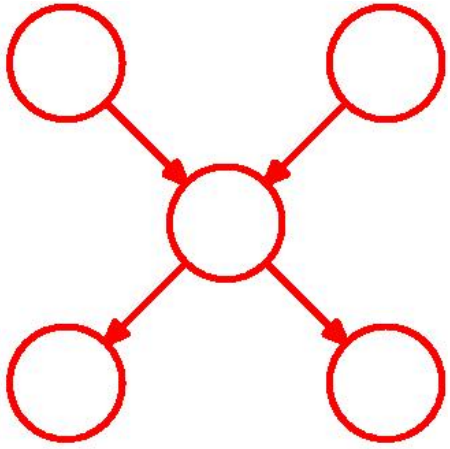
Undirected Tree



Directed Tree



Polytree



Factorization

Directed graphs:

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

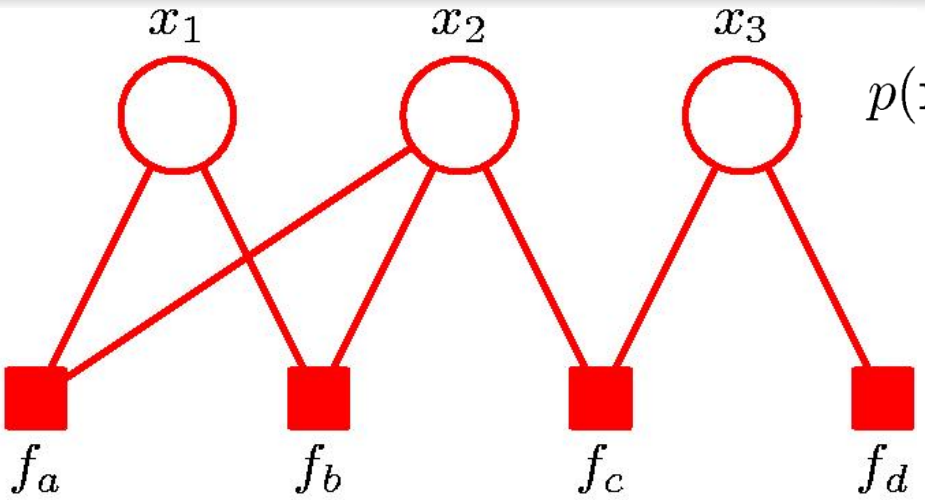
Undirected graphs:

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

Both have the form of products of factors:

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

Factor Graphs

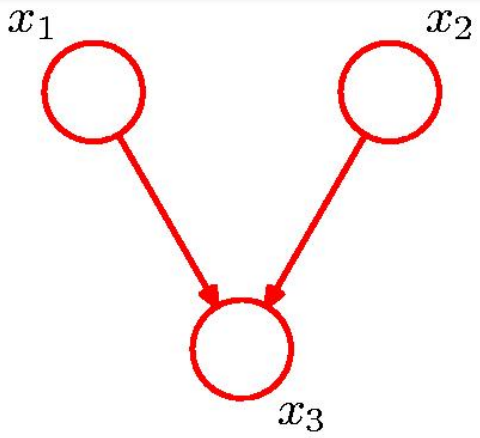


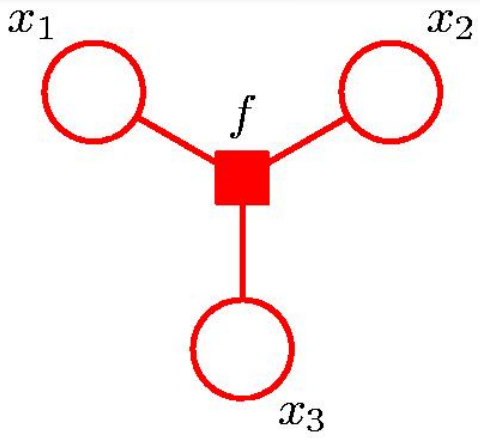
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

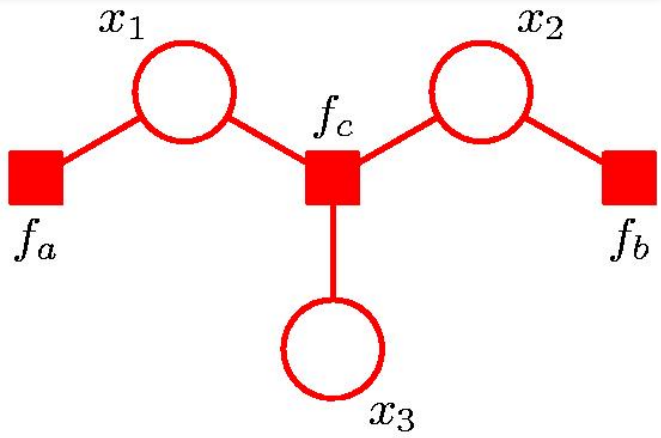
$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

More verbose!

from Directed Graphs to Factor Graphs



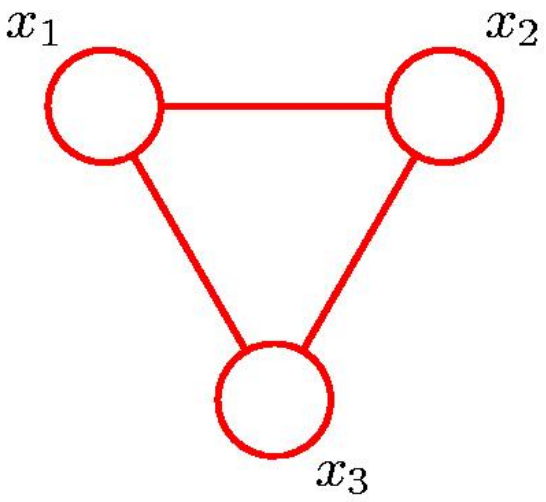




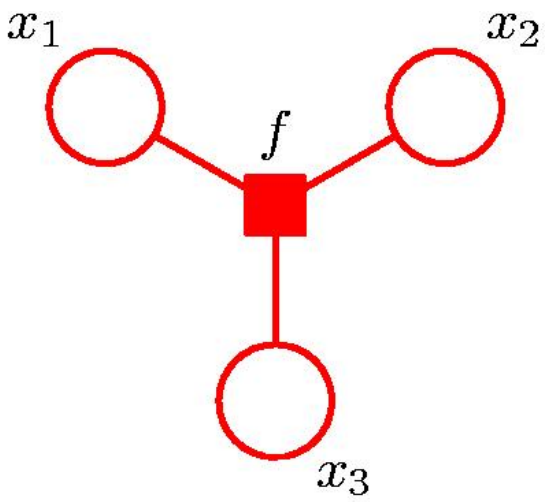
$$p(\mathbf{x}) = p(x_1)p(x_2) \quad f(x_1, x_2, x_3) =$$
$$p(x_3|x_1, x_2) \quad p(x_1)p(x_2)p(_3|x_1, x_2)$$

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

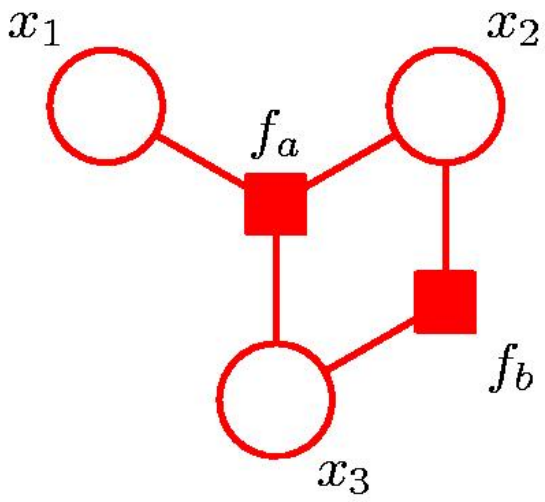
Factor Graphs from **Undirected** Graphs



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



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



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

INFERENCE

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.

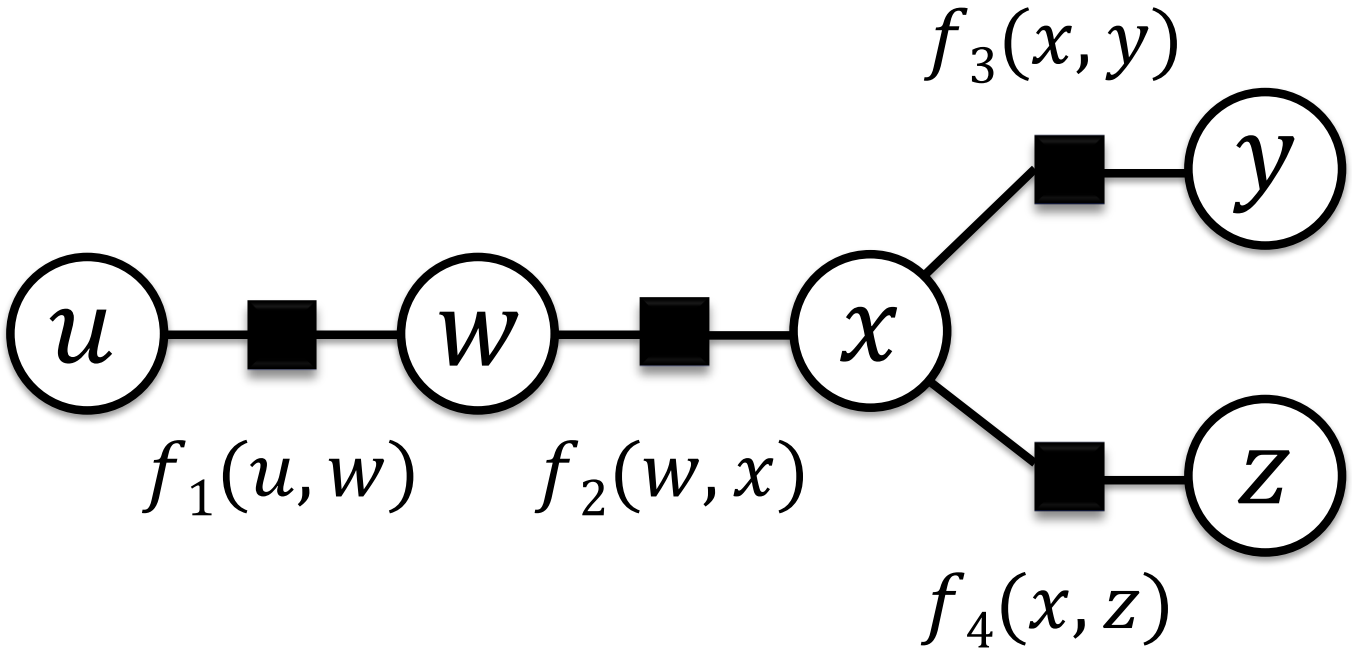
Key idea: Distributive Law

$$\sum_x \sum_y xy = x_1y_1 + x_2y_1 + x_1y_2 + x_2y_2$$

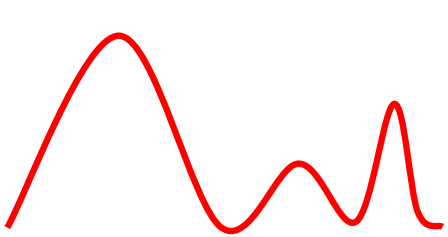
$$= (x_1 + x_2)(y_1 + y_2)$$

7 versus 3 operations

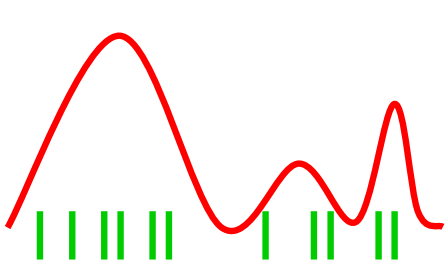
The Sum-Product Algorithm



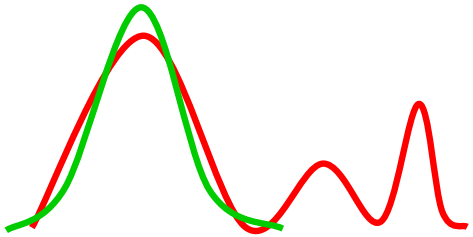
What if the messages are intractable?



True distribution



Monte Carlo

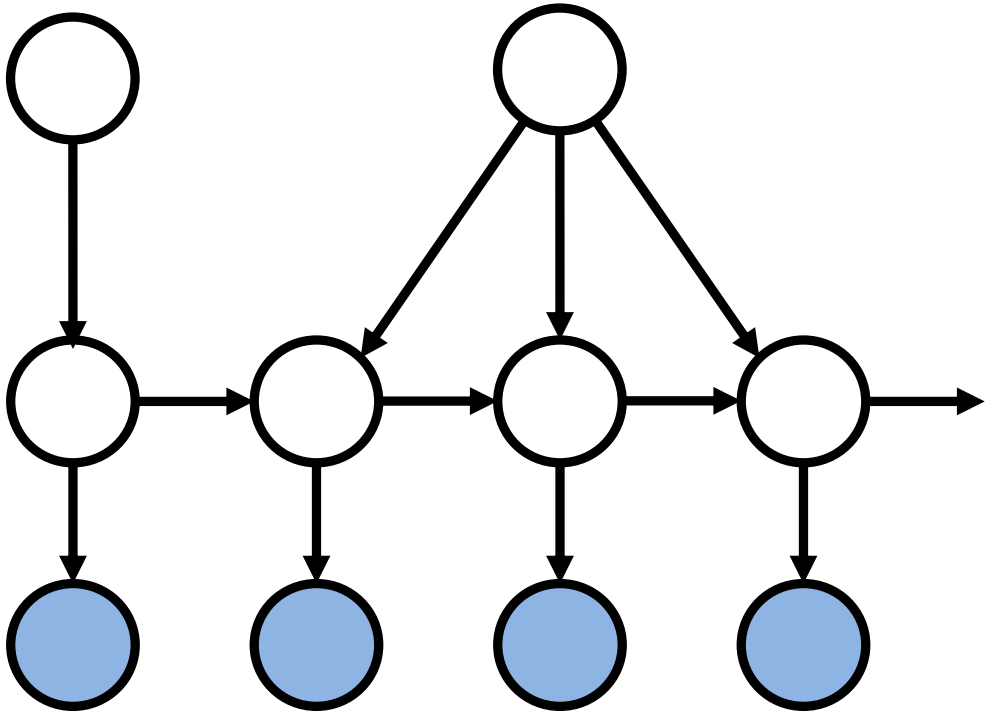


Variational Message Passing

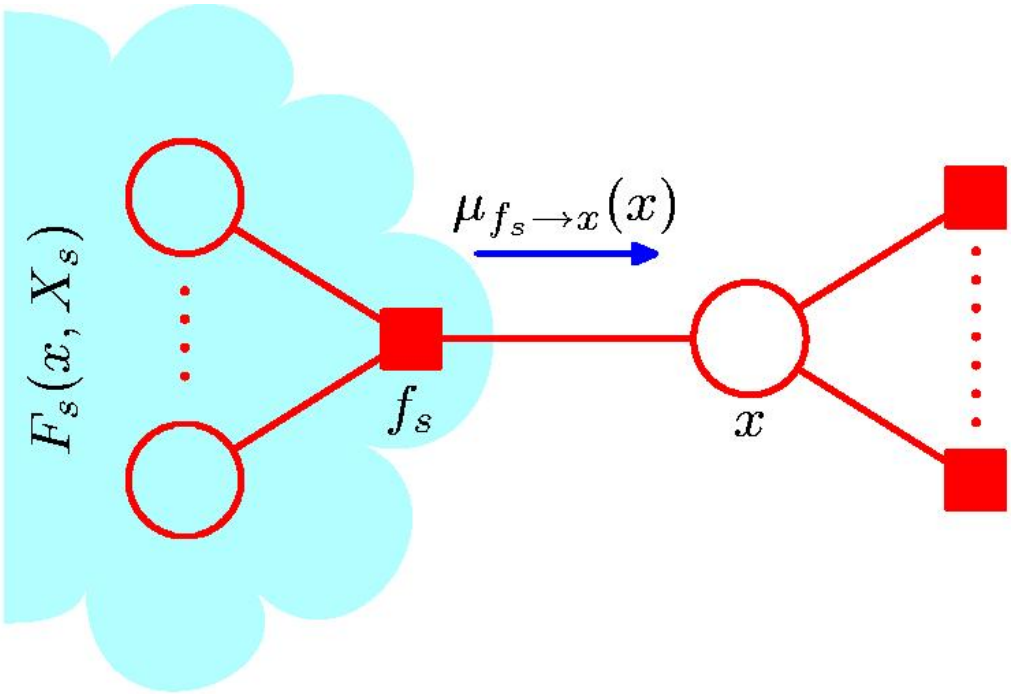
Expectation propagation

⋮

Learning is just inference!



The Sum-Product Algorithm (2)



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

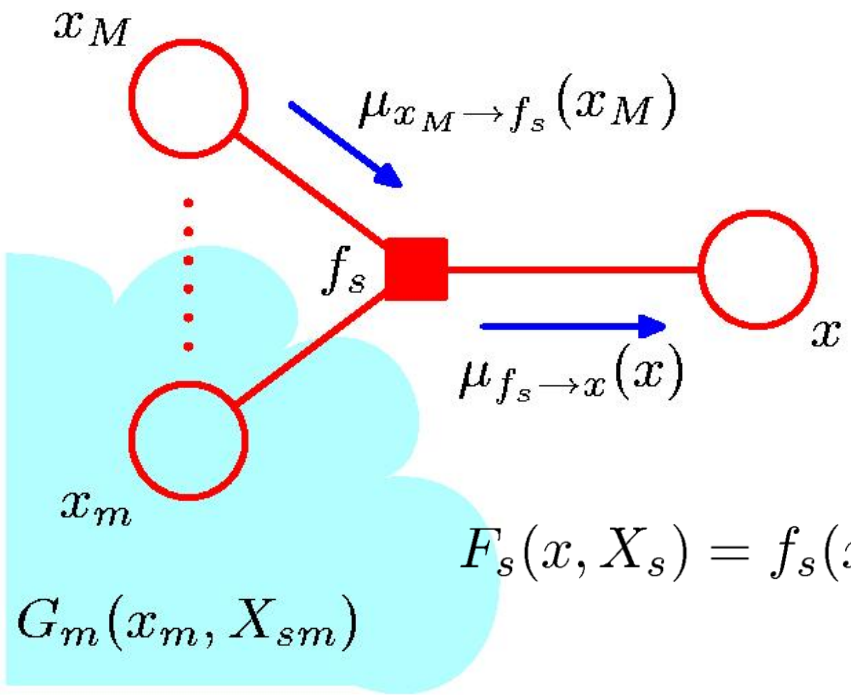
$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$$

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

$$= \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x).$$

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

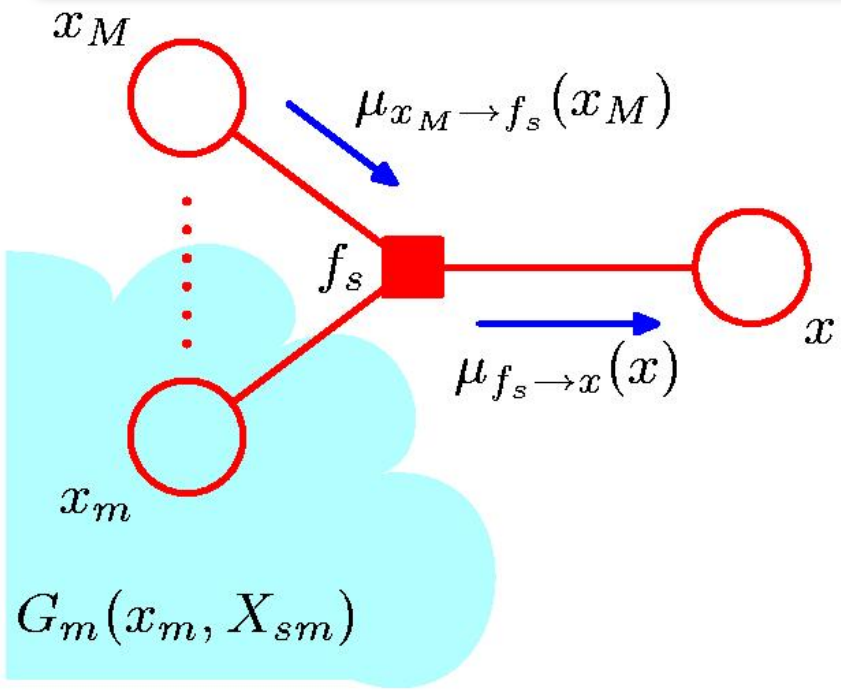
The Sum-Product Algorithm (3)



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

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

The Sum-Product Algorithm (4)

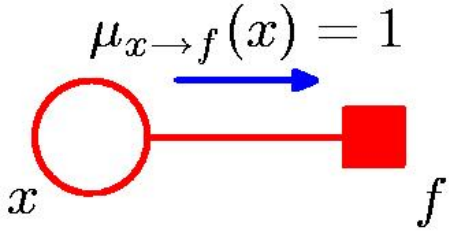


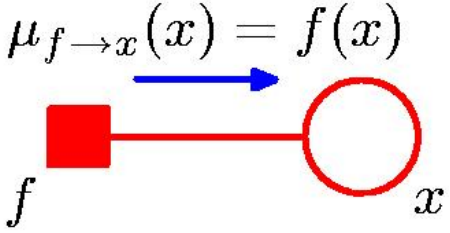
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

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

The Sum-Product Algorithm (5)

Initialization

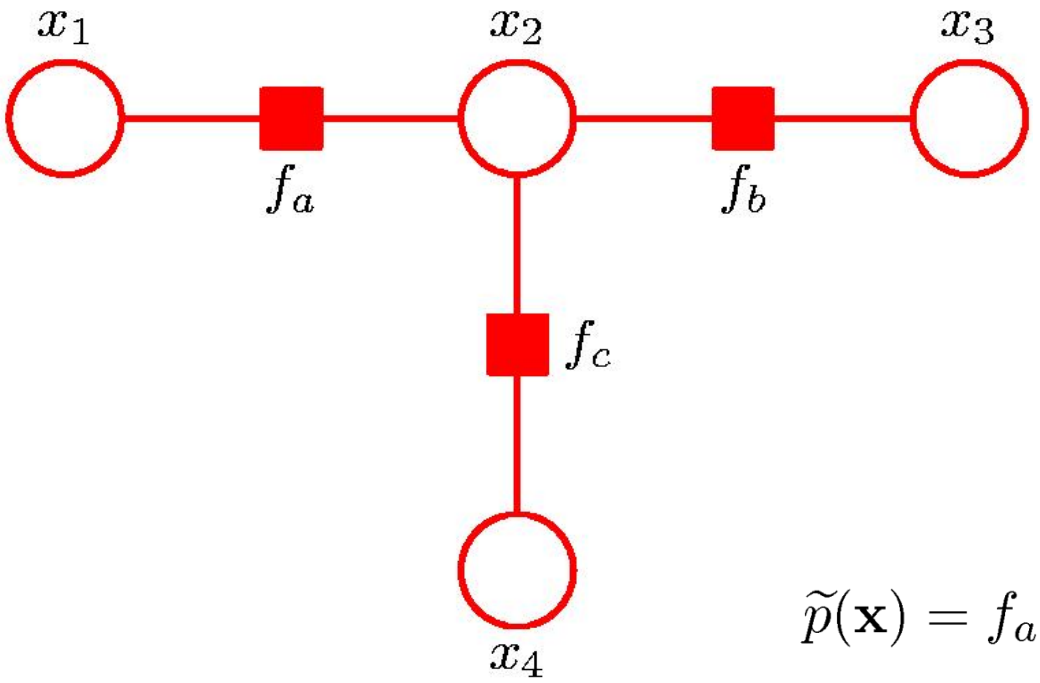




To compute local marginals:

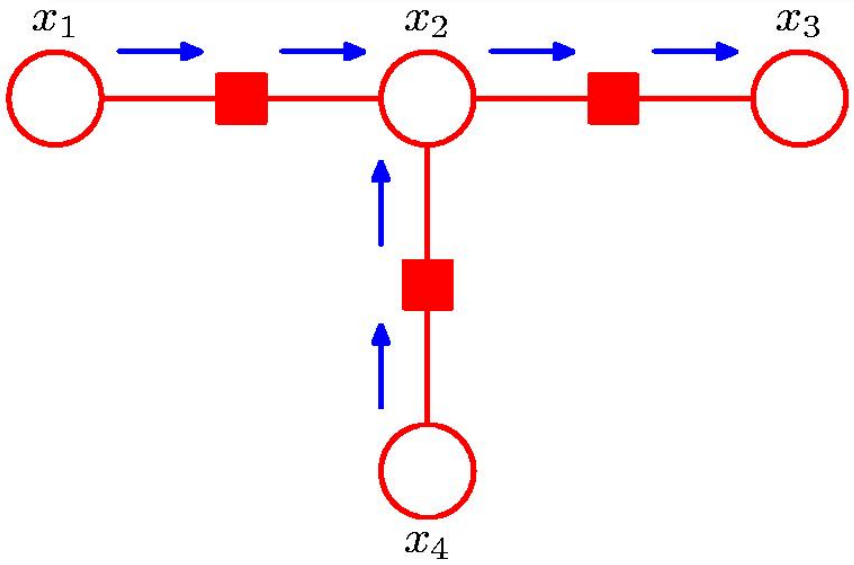
- Pick an arbitrary node as root
- Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
- Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
- Compute the product of received messages at each node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

Sum-Product: Example (1)

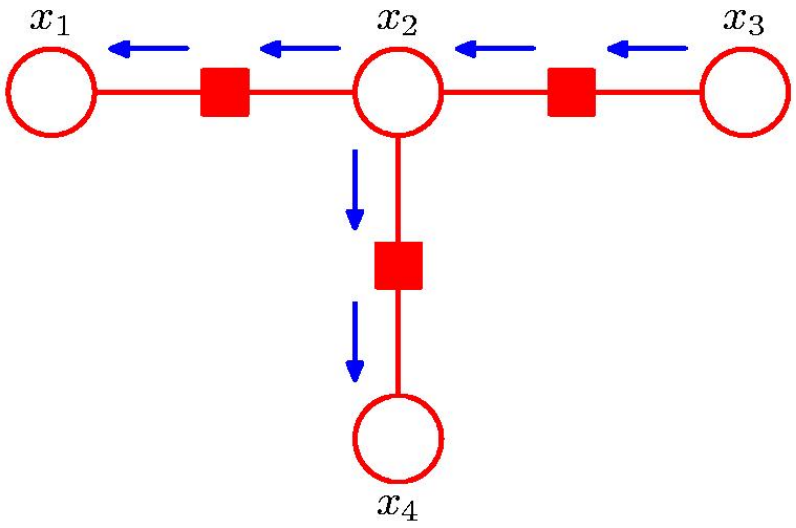


$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

Sum-Product: Example (2) and (3)



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



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