Machine Learning

Chapter 8: Graphical Models

林嘉文 (Chia-Wen Lin)

清華大學電機系

cwlin@ee.nthu.edu.tw

Introduction

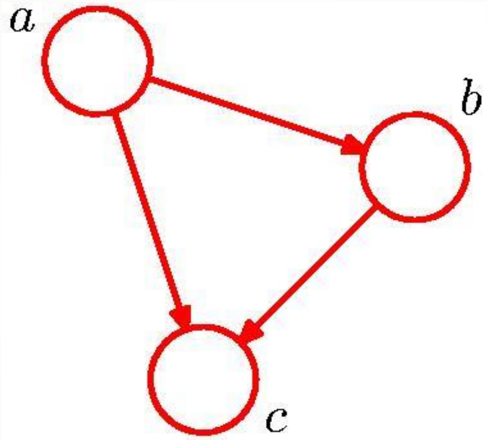
- Visualize the structure of a probabilistic model
- Design and motivate new models
- Insights into the model's properties, in particular **conditional independence** obtained by inspection
- Complex computations = graphical manipulations

Terminology

- Nodes (vertices) + links (arcs, edges)
- Node: a random variable
- Link: a probabilistic relationship
- **Directed graphical models** or Bayesian networks useful to express causal relationships between variables.
- **Undirected graphical models** or Markov random fields useful to express soft constraints between variables.
- **Factor graphs** convenient for solving inference problems

Bayesian Networks

- Directed Acyclic Graph (DAG)



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

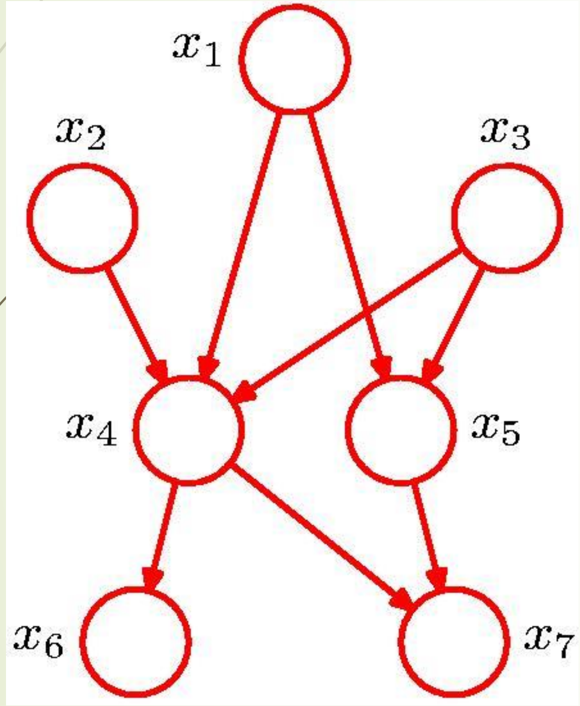
Generalization to K variables:

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

- The associated graph is fully connected.
- The absence of links conveys important information.

Bayesian Networks

$$
\begin{aligned}
p(x_1, \ldots, x_7) \quad = \quad & p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\
& p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)
\end{aligned}
$$



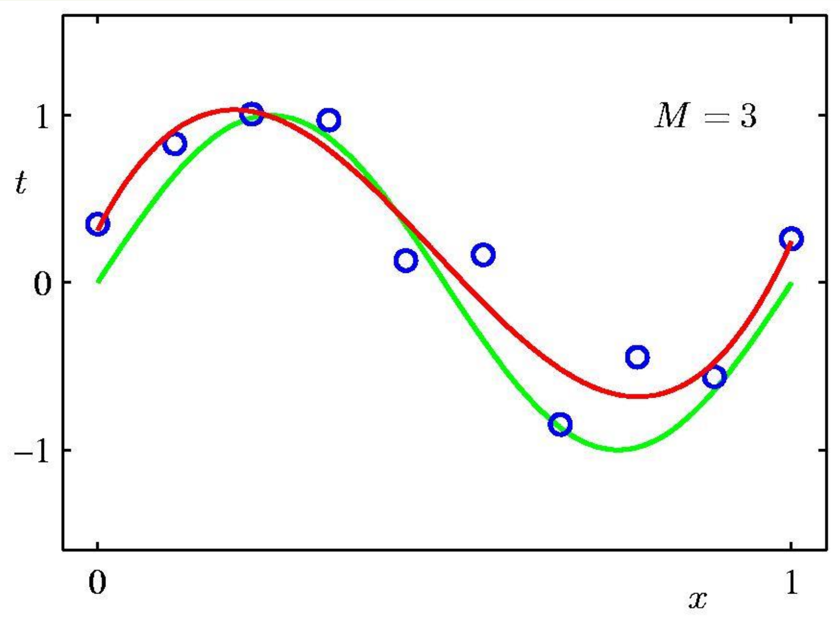
is not a fully connected graph

General Factorization

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

- This key equation expresses the factorization properties of the joint distribution.
- There must be no directed cycles
- These graphs are also called directed acyclic graphs (DAGs).

Bayesian Curve Fitting (1)



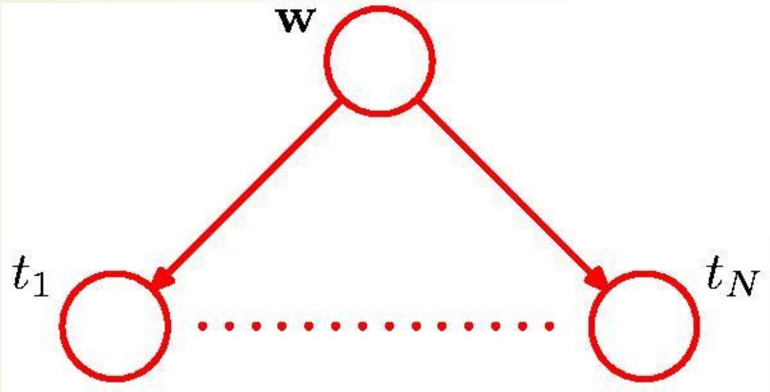
Polynomial

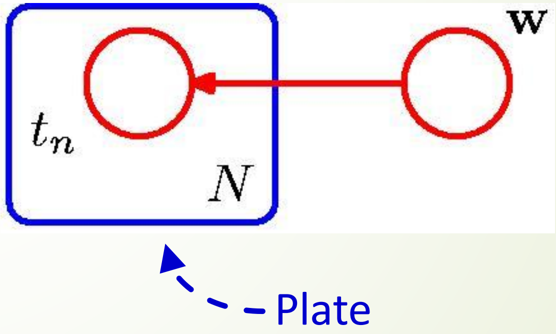
$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$$

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

Bayesian Curve Fitting (2)

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

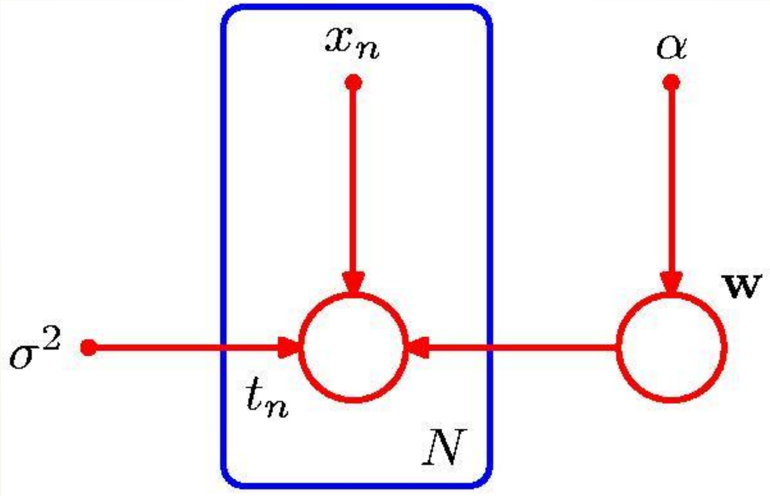




Bayesian Curve Fitting (3)

- Input variables and explicit hyperparameters

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

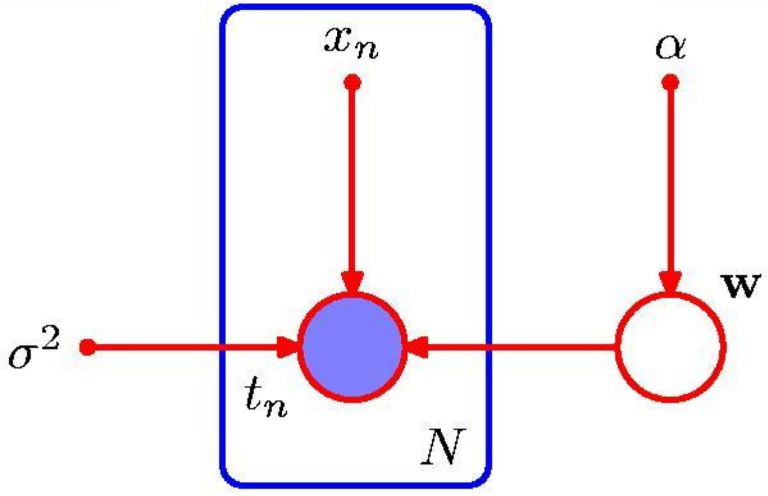


Deterministic parameters shown by small node

Bayesian Curve Fitting —Learning

- Condition on data

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

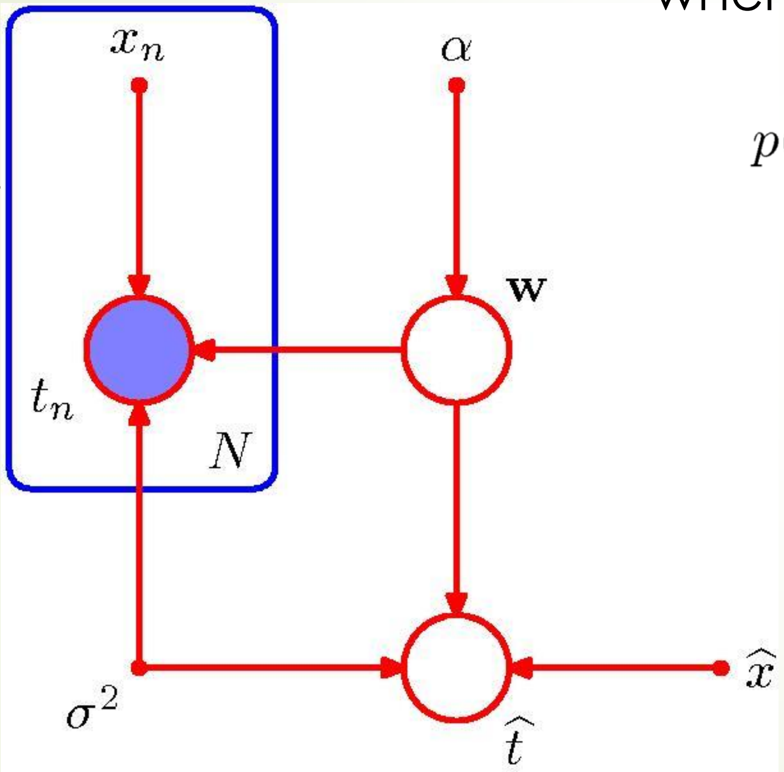


Latent (hidden) variable

Shaded nodes are set to observed values

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

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

Generative Models

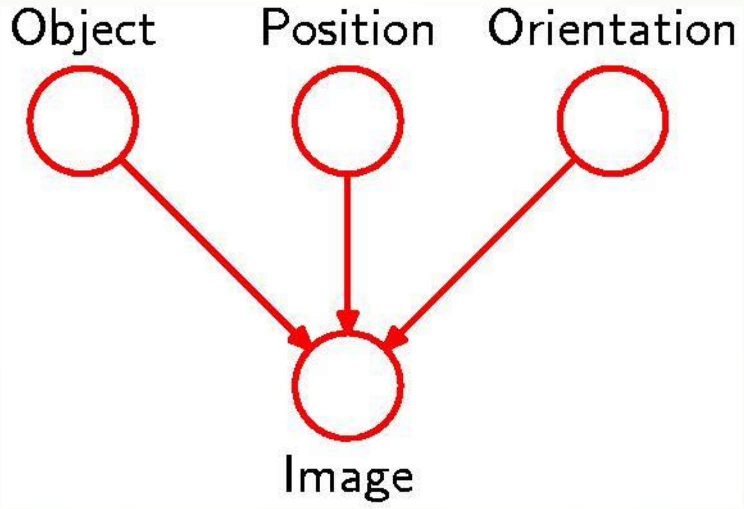
- Back to:

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

- Each node has a higher number than any of its parents.
- The factorization above corresponds to a DAG.
- Goal: draw a sample $\hat{x}_1, \cdots, \hat{x}_K$ from the joint distribution.
- Apply **ancestral sampling** starting from lower-numbered nodes, downwards through the graph's nodes.
- Generative graphical model captures the causal process that generated the observed data (object recognition example)

Generative Models

- Causal process for generating images



Discrete Variables (1)

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

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

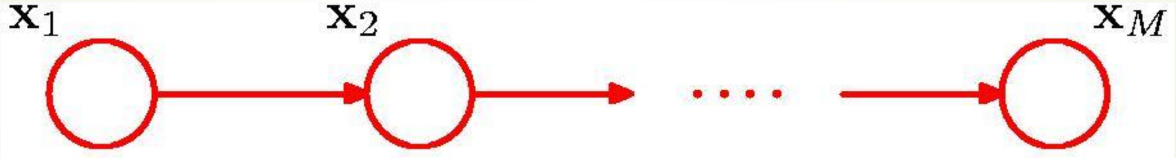
- Independent joint distribution: $2(K-1)$ parameters

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

Discrete Variables (2)

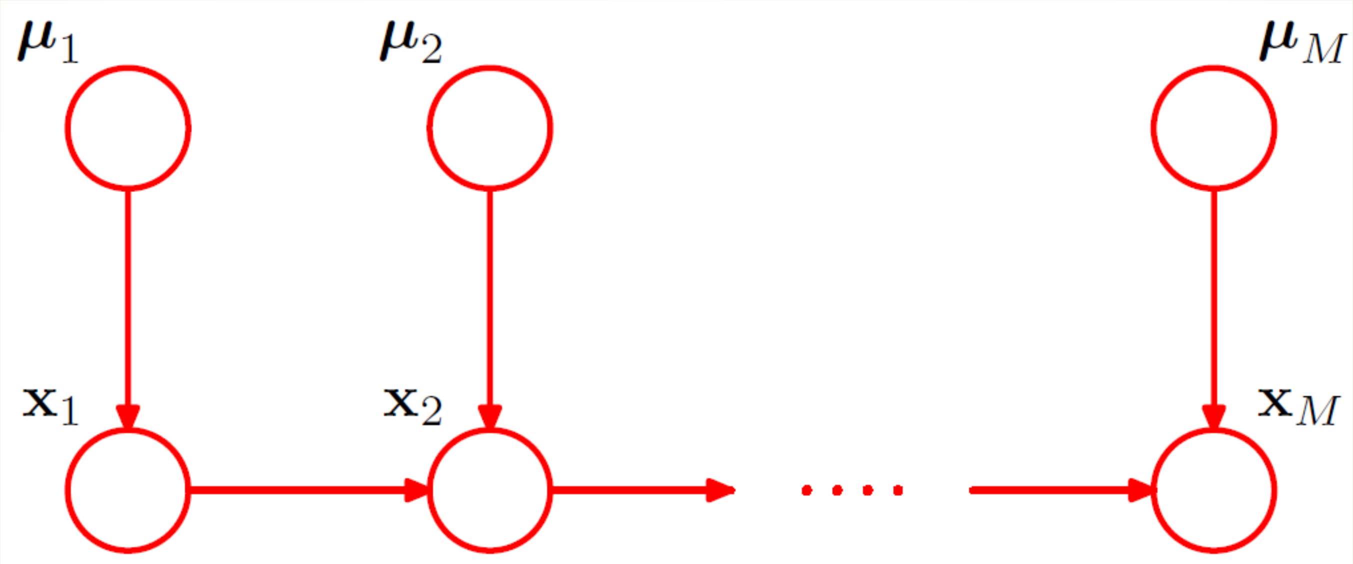
- General joint distribution over M variables: $K^M - 1$ parameters

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



$K - 1$ $\quad K(K - 1)$ $\quad K(K - 1)$

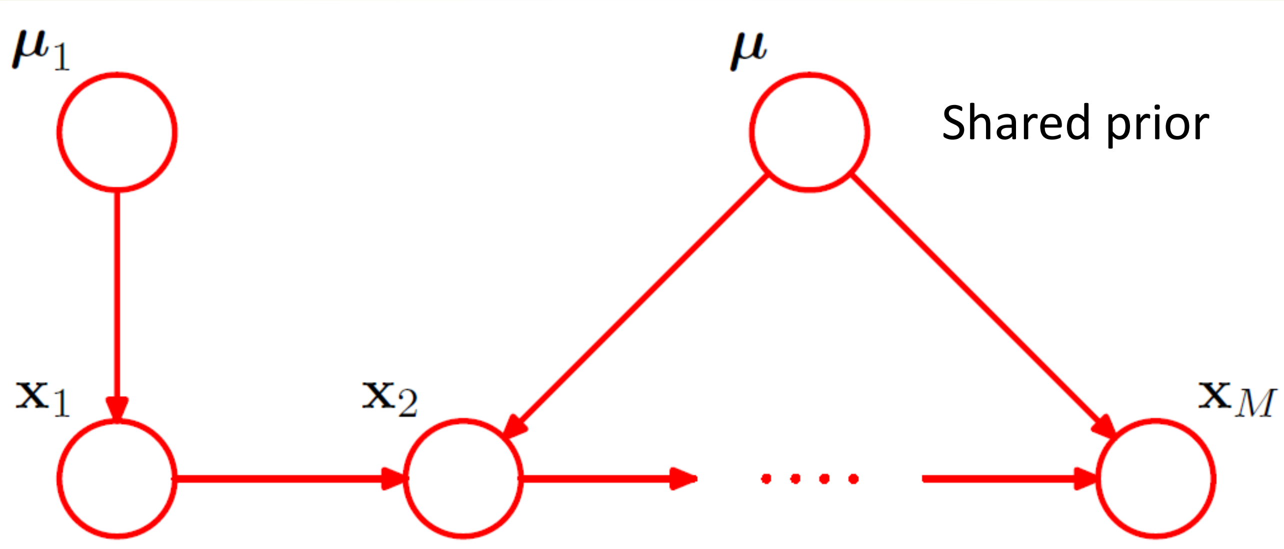
Discrete Variables: Bayesian Parameters (1)



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

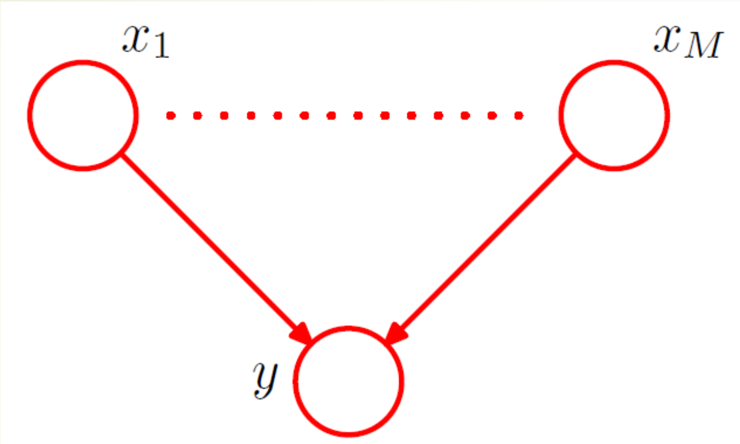
$$p(\boldsymbol{\mu}_m) = \mathrm{Dir}(\boldsymbol{\mu}_m | \boldsymbol{\alpha}_m)$$

Discrete Variables: Bayesian Parameters (2)



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

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

Linear-Gaussian Models

- Directed acyclic graph over D variables

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

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

$$\begin{aligned}\ln p(\mathbf{x}) &= \sum_{i=1}^{D} \ln p(x_i|\mathrm{pa}_i) \\ &= -\sum_{i=1}^{D} \frac{1}{2v_i}\left(x_i - \sum_{j\in\mathrm{pa}_i} w_{ij}x_j - b_i\right)^2 + \mathrm{const}\end{aligned}$$

$$x_i = \sum_{j\in\mathrm{pa}_i} w_{ij}x_j + b_i + \sqrt{v_i}\epsilon_i$$

where ϵ_i is a zero mean, unit variance Gaussian random variable

Linear-Gaussian Models

- Directed acyclic graph over D variables

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

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

- Vector-valued Gaussian Nodes

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

Conditional Independence

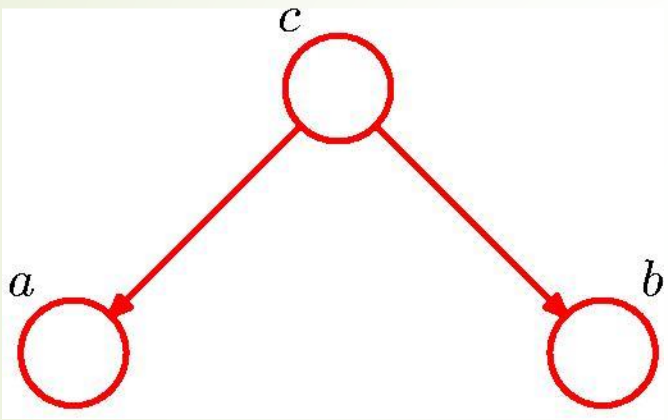
- a is conditionally independent of b given c

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

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

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

Conditional Independence: Example 1

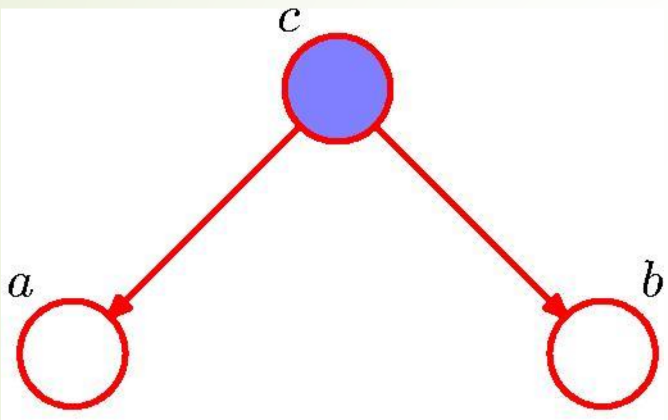


$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

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

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

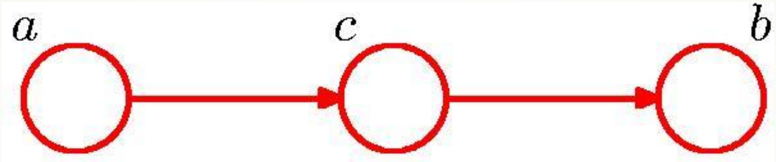
Conditional Independence: Example 1



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

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

Conditional Independence: Example 2

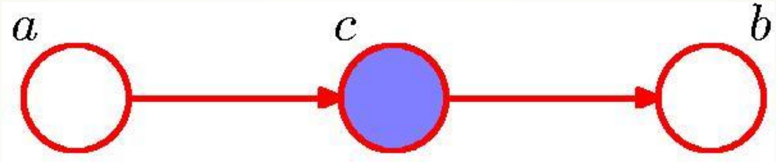


$$p(a,b,c) = p(a)p(c|a)p(b|c)$$

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

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

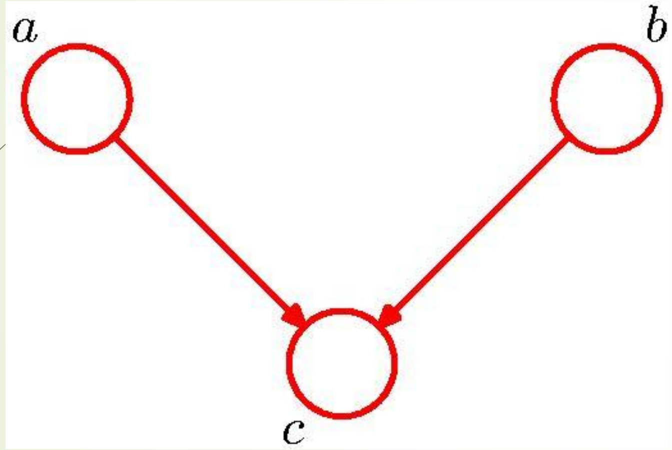
Conditional Independence: Example 2



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

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

Conditional Independence: Example 3



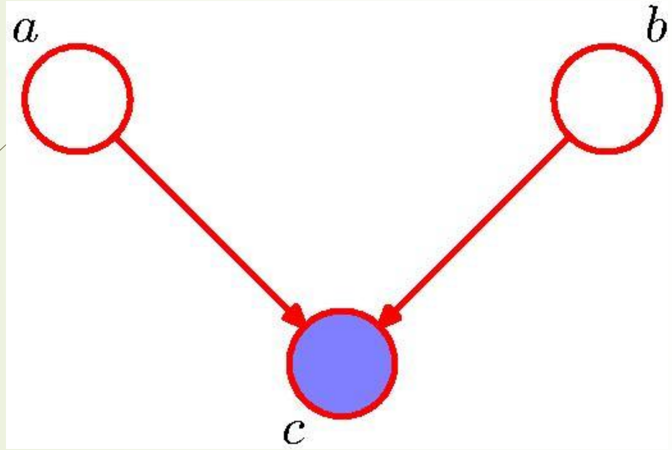
$$p(a, b, c) = p(a)p(b)p(c|a, b)$$

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

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

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

Conditional Independence: Example 3



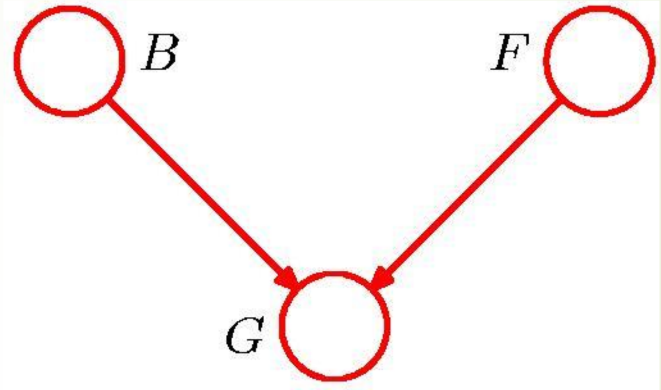
$$
\begin{aligned}
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= \frac{p(a)p(b)p(c|a, b)}{p(c)}
\end{aligned}
$$

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

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

"Am I out of fuel?"

$$
\begin{aligned}
p(G=1|B=1,F=1) &= 0.8 \\
p(G=1|B=1,F=0) &= 0.2 \\
p(G=1|B=0,F=1) &= 0.2 \\
p(G=1|B=0,F=0) &= 0.1
\end{aligned}
$$



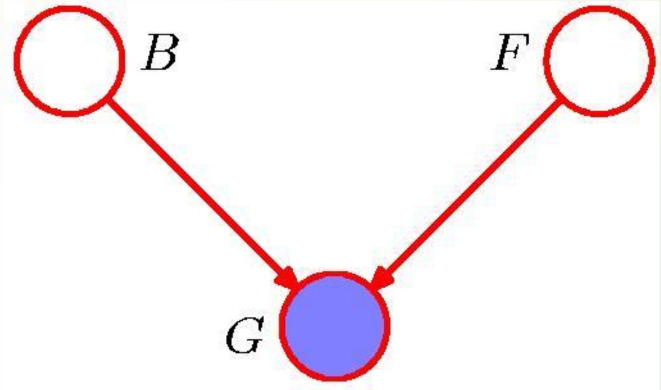
$$
\begin{aligned}
p(B=1) &= 0.9 \\
p(F=1) &= 0.9
\end{aligned}
$$

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)

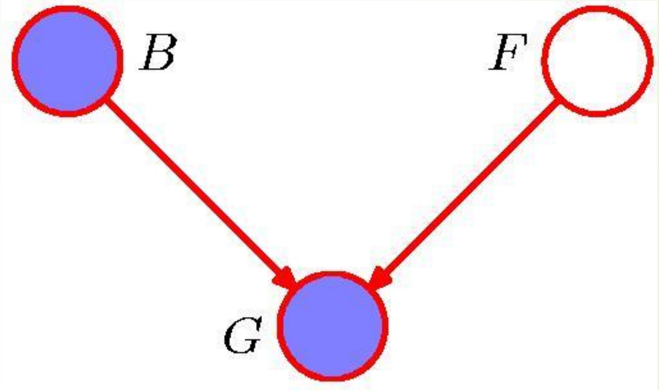
"Am I out of fuel?"



$$p(F=0|G=0) = \frac{p(G=0|F=0)p(F=0)}{p(G=0)} \simeq 0.257$$

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

“Am I out of fuel?”



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

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

D-separation

- d-separation (d connotes "directional") is a criterion for deciding, from a given causal graph, whether a set X of variables is independent of another set Y, given Z. The idea is to associate "dependence" with "connectedness" and "independence" with "unconnected-ness" or "separation" .
- Unconditional separation
  - x and y are d-connected if there is an unblocked path between them.
  - By "unblocked path" we mean a path that can be traced without traversing a pair of arrows that collide "head-to-head". In other words, arrows that meet head-to-head do not constitute a connection for the purpose of passing information, such a meeting will be called a "collider"

$$x \longrightarrow r \longrightarrow s \longrightarrow t \longleftarrow u \longleftarrow v \longrightarrow y$$

D-separation

- Unconditional separation
  - x and y are d-connected if there is an unblocked path between them.

$$x \longrightarrow r \longrightarrow s \longrightarrow t \longleftarrow u \longleftarrow v \longrightarrow y$$

  - This graph contains one collider, at t. The path x-r-s-t is unblocked, hence x and t are d-connected. So is also the path t-u-v-y, hence t and y are d-connected. However, x and y are not d-connected (i.e., d-separated); there is no way of tracing a path from x to y without traversing the collider at t. (The ramification is that the covariance terms corresponding to these pairs of variables will be zero, for every choice of model parameters)

D-separation

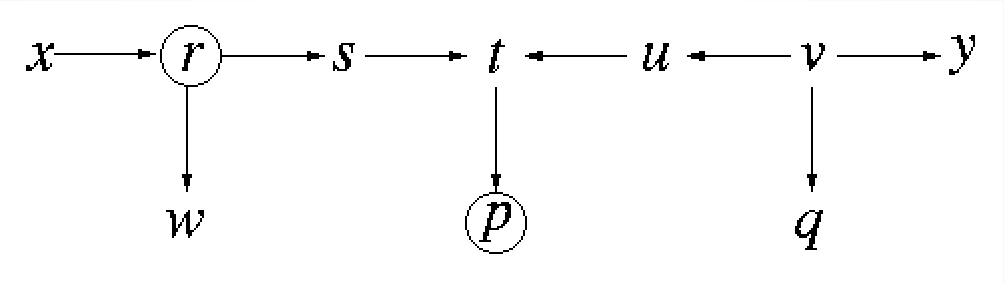
- Blocking by conditioning
  - x and y are *d*-connected, conditioned on a set Z of nodes, if there is a collider-free path between x and y that traverses no member of Z. If no such path exists, we say that x and y are *d*-separated by Z, We also say then that every path between x and y is "blocked" by Z.

$$x \to r \to s \to t \leftarrow u \leftarrow v \to y$$

  - Let $Z = \{r, v\}$ (marked by circles). x and y are d-separated by Z, and so are x and s, u and y, s and u etc. The path x-r-s is blocked by Z. The only pairs of unmeasured nodes that remain d-connected in this example, conditioned on Z, are s and t and u and t.

D-separation

- Conditioning on colliders
  - If a collider is a member of the conditioning set Z, or has a descendant in Z, then it no longer blocks any path that traces this collider.

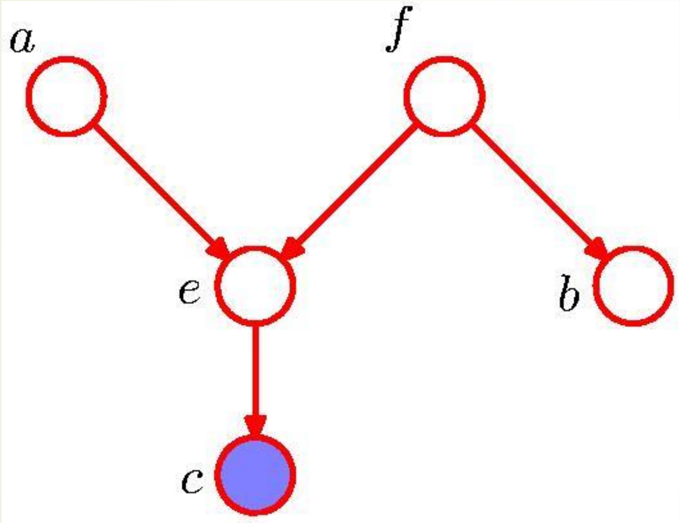


  - Let $Z = \{r, v\}$. s and y are d-connected by Z, because the collider at t has a descendant (p) in Z, which unblocks the path s-t-u-v-y. However, x and u are still d-separated by Z, because although the linkage at t is unblocked, the one at r is blocked since r is in Z.

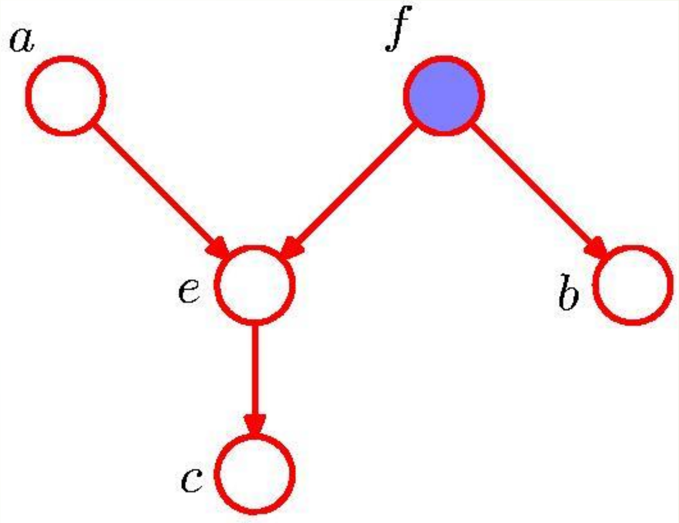
D-separation

- 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, is 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 $A \perp B | C$.

D-separation: Example

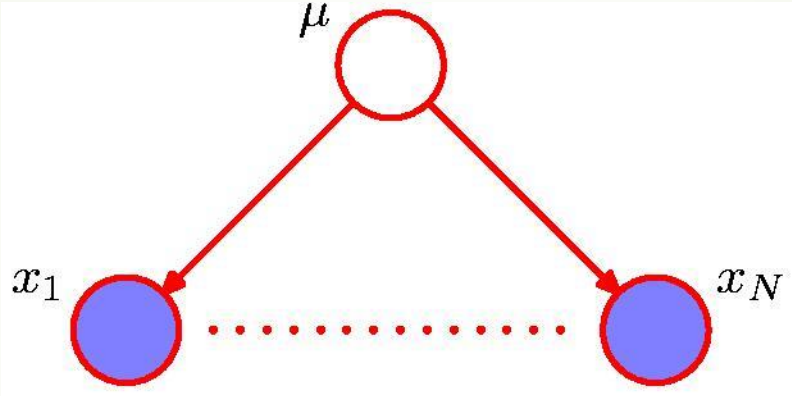


$a \not\perp\!\!\!\perp b \mid c$



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

D-separation: I.I.D. Data

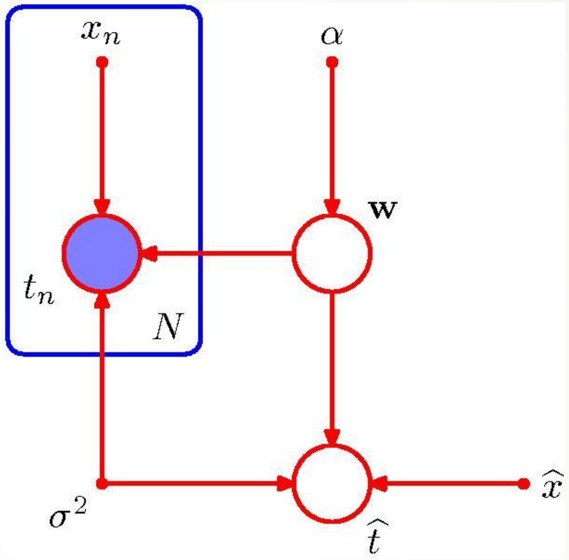


$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu)$$

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

Here μ is a latent variable, because its value is not observed

D-Separation: Bayesian Polynomial Regression



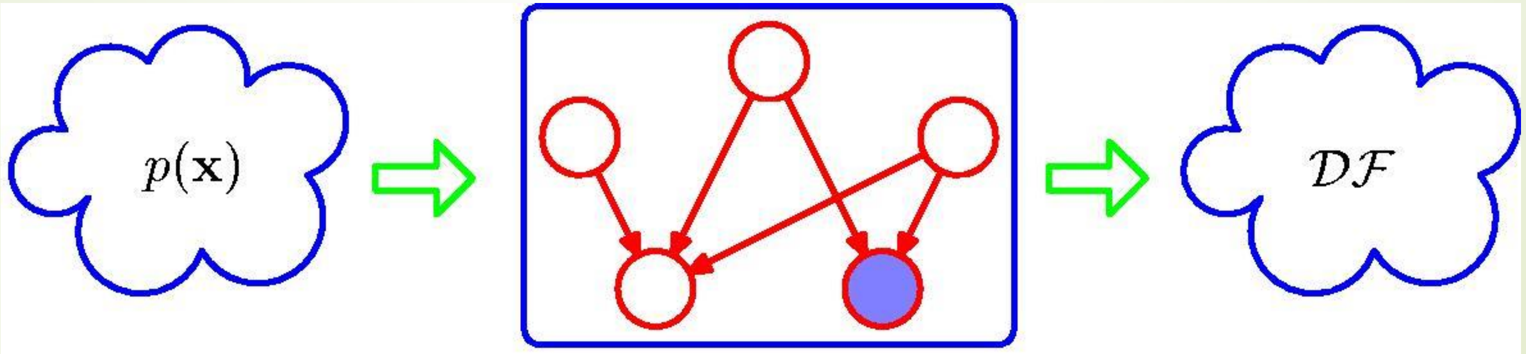
- $\mathbf{w}$ is a tail-to-tail node with respect to the path from $\hat{t}$ to any one of the nodes $\{t_n\}$.
- Hence $\hat{t} \perp t_n|\mathbf{w}$
- Interpretation:
  - First use the training data to determine the posterior distribution over $\mathbf{w}$
  - Discard $\{t_n\}$ and use posterior distribution for $\mathbf{w}$ to make predictions of $\hat{t}$ for new input observations $\hat{x}$

Interpretation as Filter

- Filter-I: allows a distribution to pass through if, and only if, it can be expressed in terms of the factorization implied by the graph

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

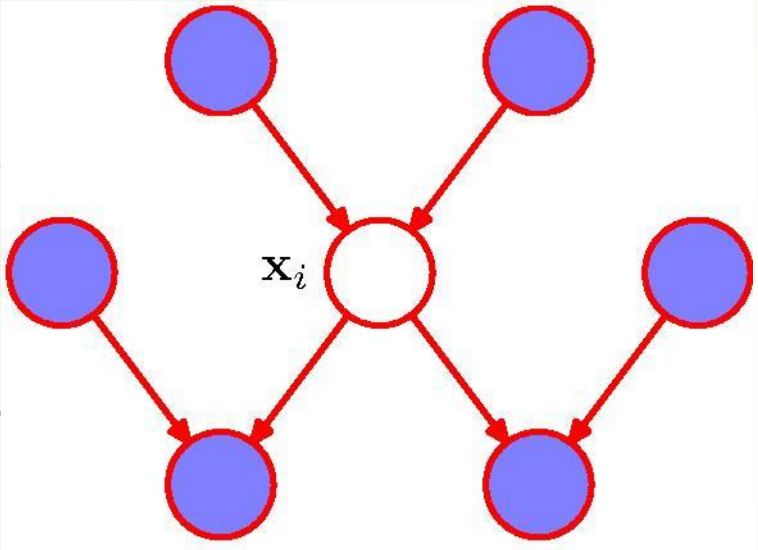
- Filter-II: allows distributions to pass according to whether they respect all of the conditional independencies implied by the d-separation properties of the graph
- The set of all possible probability distributions $p(\mathbf{x})$ that is passed by both the filters is precisely the same and are denoted by DF, for **directed factorization**



Directed Graphs: Summary

- Represents specific decomposition of a joint probability distribution into a product of conditional probabilities
- Expresses a set of conditional independence statements through d-separation criterion
- Distributions satisfying d-separation criterion are denoted as DF
- Extreme Cases: DF can contain all possible distributions in case of fully connected graph or product of marginals in case fully disconnected graphs

The Markov Blanket

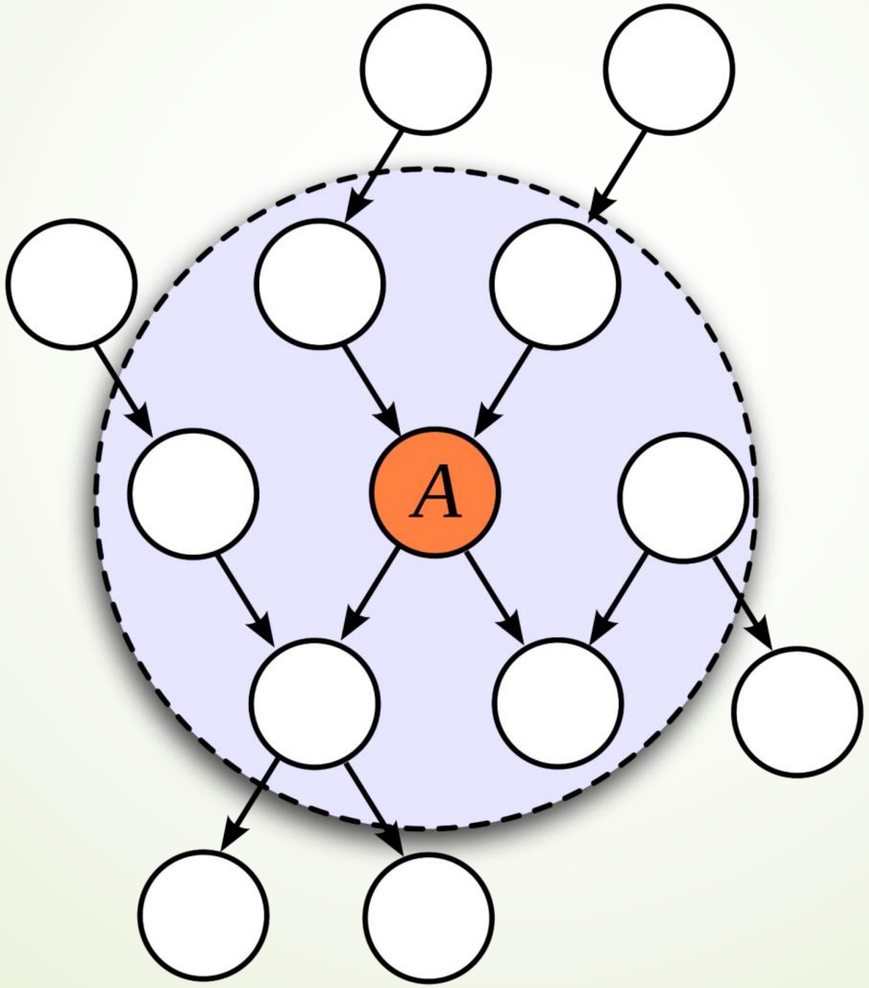


$$p(\mathbf{x}_i|\mathbf{x}_{\{j\neq i\}}) = \frac{p(\mathbf{x}_1,\ldots,\mathbf{x}_M)}{\int p(\mathbf{x}_1,\ldots,\mathbf{x}_M)\,\mathrm{d}\mathbf{x}_i} = \frac{\prod_k p(\mathbf{x}_k|\mathrm{pa}_k)}{\int \prod_k p(\mathbf{x}_k|\mathrm{pa}_k)\,\mathrm{d}\mathbf{x}_i}$$

- Factors independent of $\mathbf{x}_i$ cancel between numerator and denominator.
- Only factors remaining are
  - Parents and children $\mathbf{x}_i$
  - Also co-parents: corresponding to parents of node $\mathbf{x}_k$ (other than $\mathbf{x}_i$)

These remaining factors are referred to as The **Markov Blanket** of node $\mathbf{x}_i$

The Markov Blanket

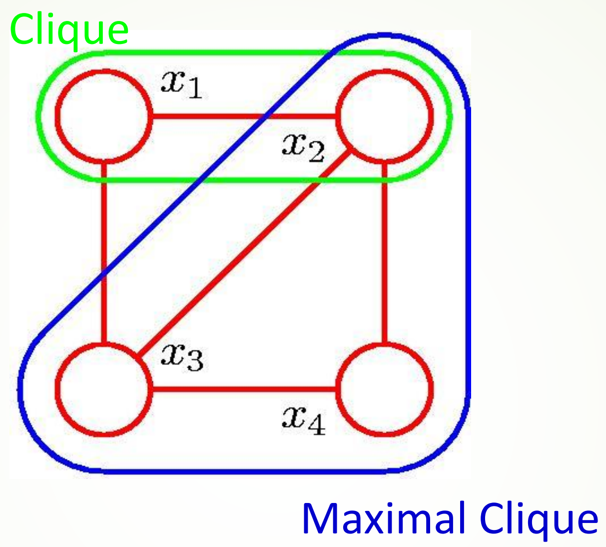


Factorization Properties

- Consider two nodes x_i and x_j that are not connected by a link then these are conditionally independent given all other nodes
- As there is no direct path between the nodes
- All other paths are blocked by nodes that are observed

$$p(x_i, x_j | \mathbf{x}_{\setminus\{i,j\}}) = p(x_i | \mathbf{x}_{\setminus\{i,j\}})\, p(x_j | \mathbf{x}_{\setminus\{i,j\}})$$

Cliques and Maximal Cliques



- **Clique**: A set of fully connected nodes
- **Maximal Clique**: clique in which it is not possible to include any other nodes without it ceasing to be a clique
- Joint distribution can thus be factored in terms of the functions of maximal cliques
- Functions defined on maximal cliques include the subsets of maximal cliques

Joint Distribution

For clique C and the set of variables in that clique $\mathbf{x}_C$, The 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 the partition function Z

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

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

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

- For evaluating local marginal probabilities the unnormalized joint distribution can be used

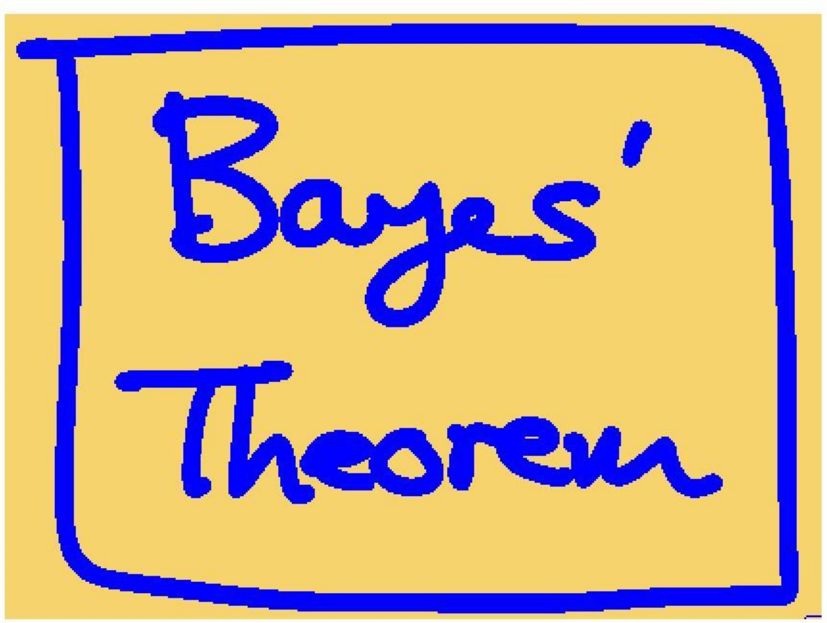
Hammersley and Clifford Theorem

- Using filter analogy
- UI: the set of distributions that are consistent with the set of conditional independence statements read from the graph using graph separation
- UF: the set of distributions that can be expressed as a factorization described with respect to the maximal cliques
- The Hammersley-Clifford theorem states that the sets UI and UF are identical if $\psi_C(\mathbf{x}_C)$ is strictly positive
- In such case

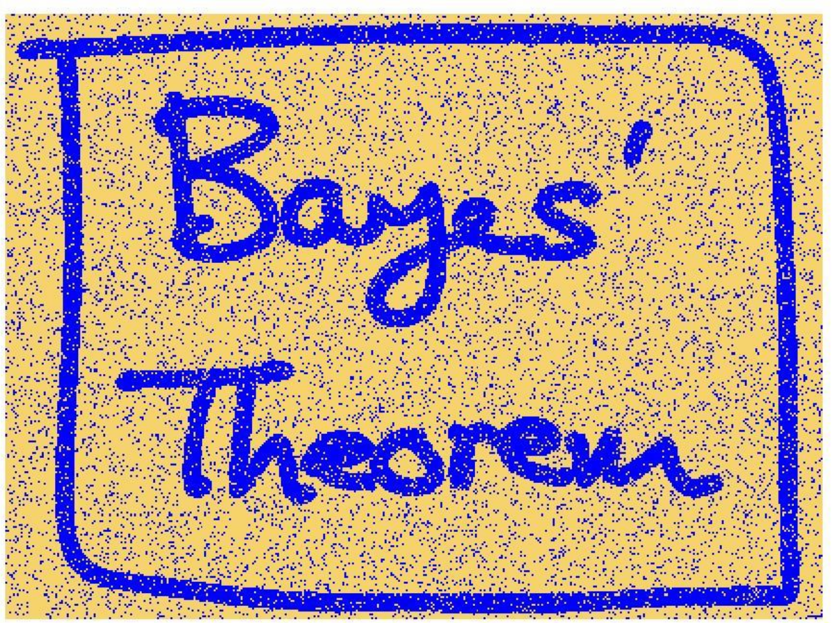
$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

where $E(\mathbf{x}_C)$ is called an energy function, and the exponential representation is called the Boltzmann distribution

Illustration: Image De-Noising (1)

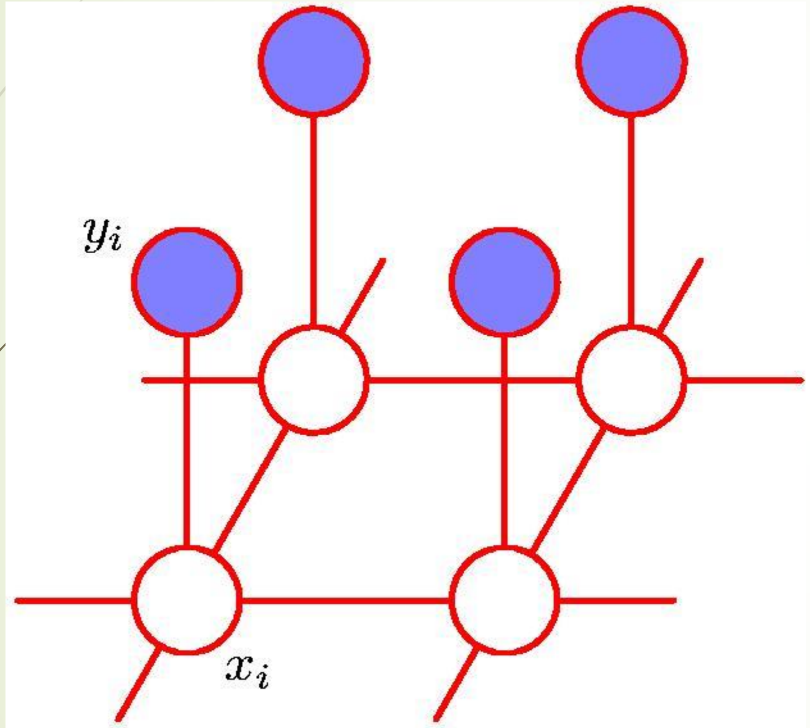


Original Image $\mathbf{x}$



Noisy Image $\mathbf{y}$ (10% noise)

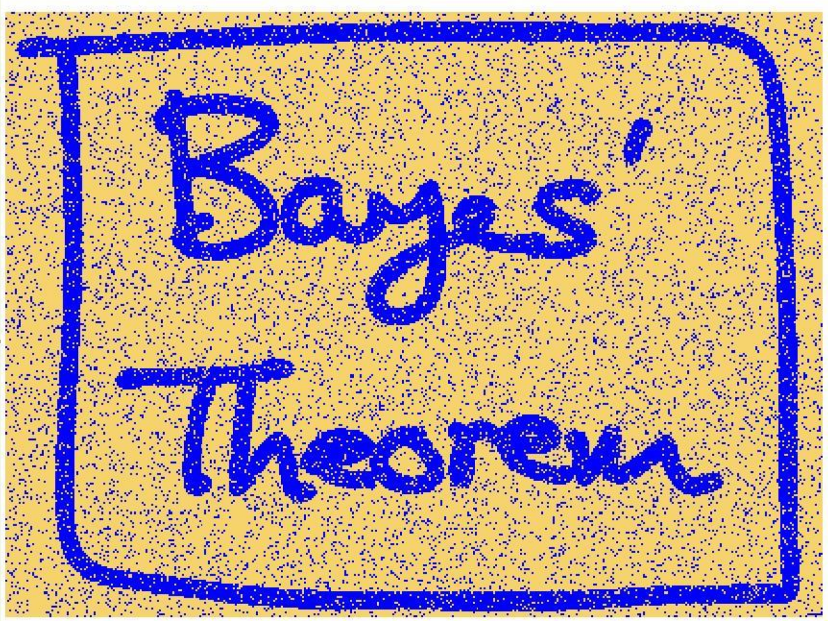
Illustration: Image De-Noising (2)



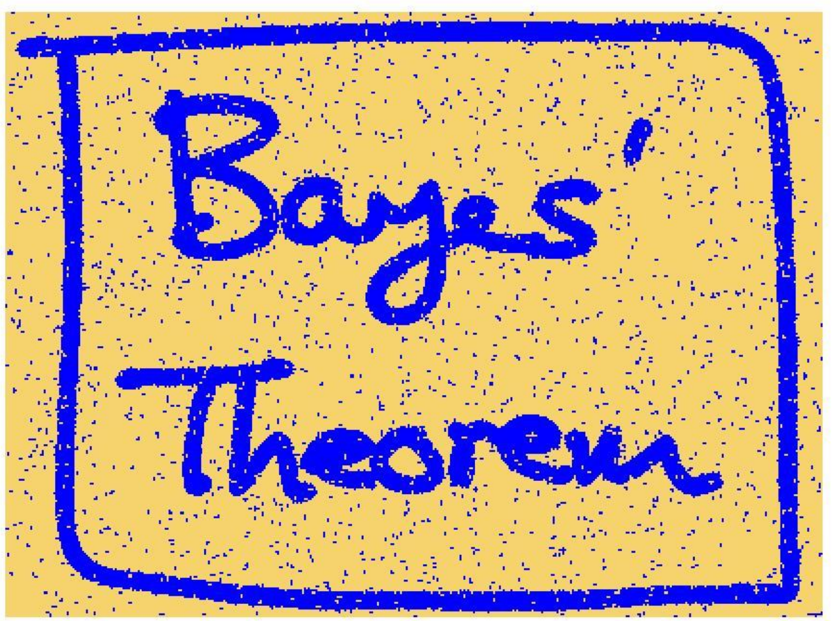
$$E(\mathbf{x},\mathbf{y}) = h\sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i$$

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

Illustration: Image De-Noising (3)

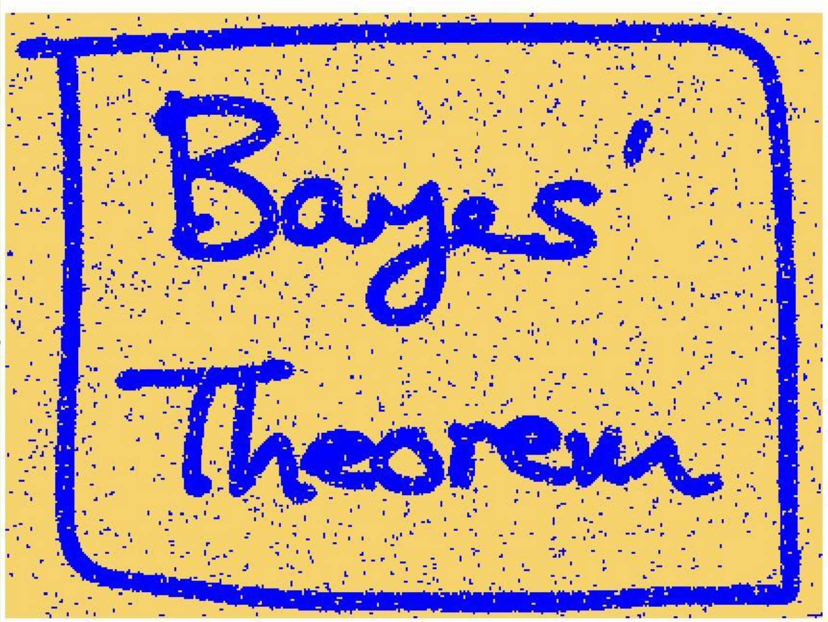


Noisy Image

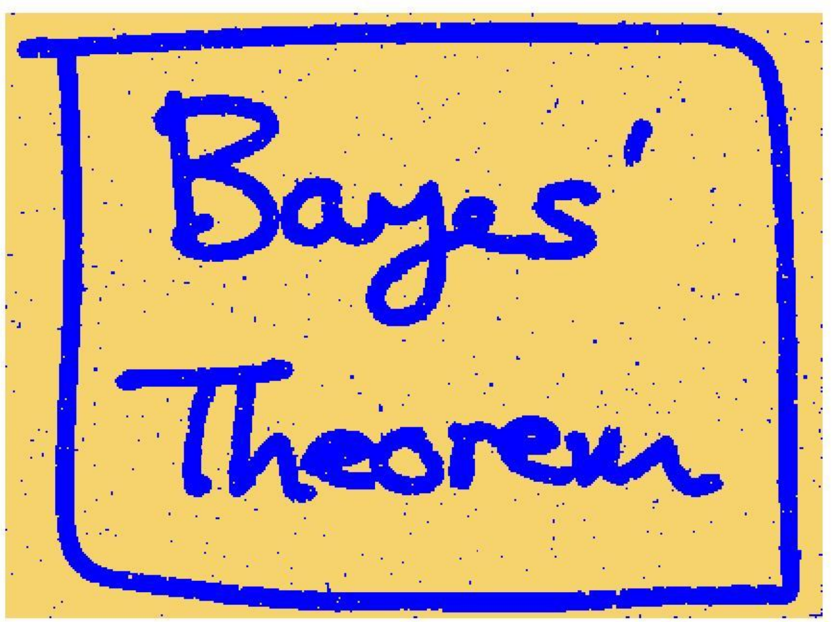


Restored Image
(Iterated Conditional Modes; ICM)

Illustration: Image De-Noising (4)



Restored Image (ICM)

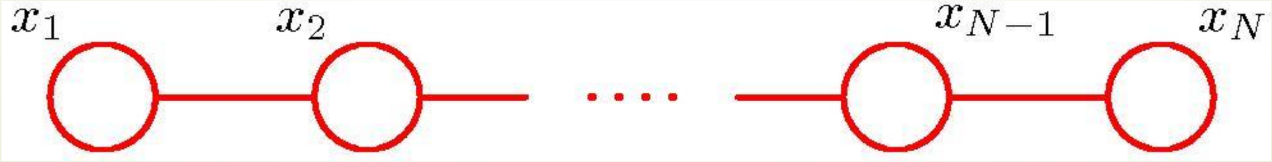


Restored Image (Graph cuts)

Converting Directed to Undirected Graphs (1)

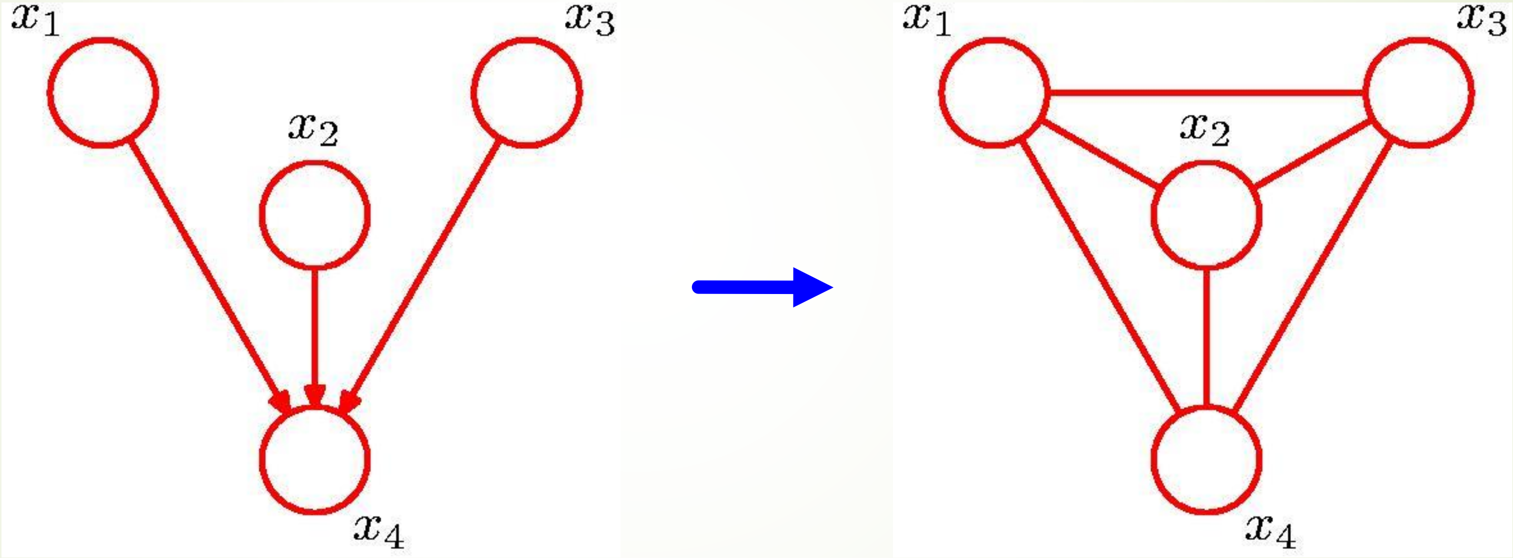
$$p(\mathbf{x}) = \underbrace{p(x_1)p(x_2|x_1)}\, p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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



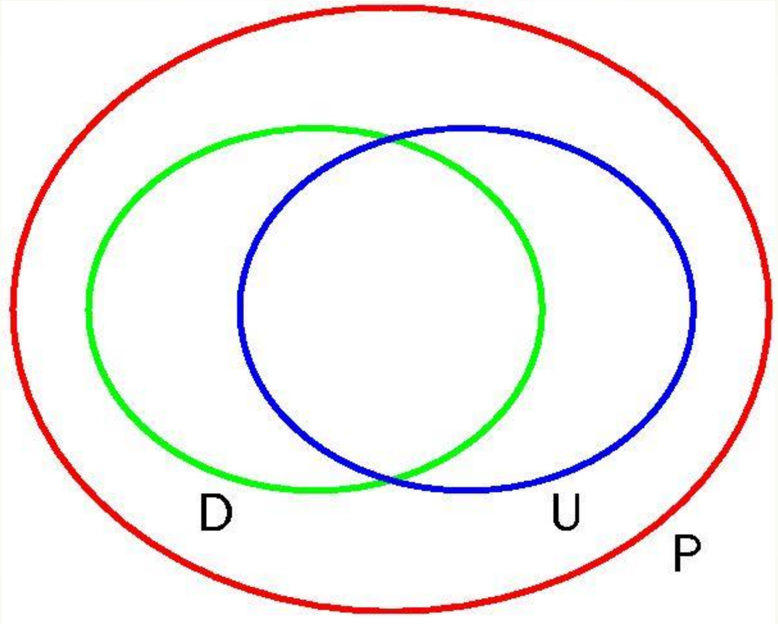
Converting Directed to Undirected Graphs (2)

- Additional links between parent nodes (moralization)



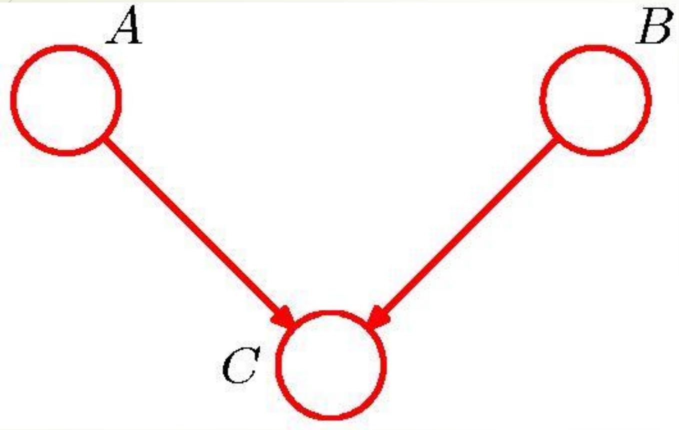
$$
\begin{aligned}
p(\mathbf{x}) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\
&= \frac{1}{Z}\psi_A(x_1, x_2, x_3)\psi_B(x_2, x_3, x_4)\psi_C(x_1, x_2, x_4)
\end{aligned}
$$

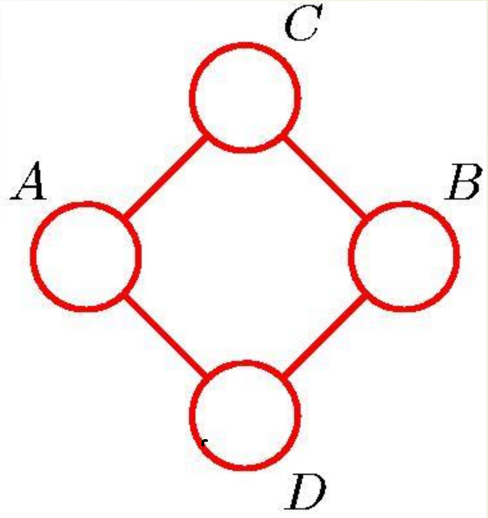
Directed vs. Undirected Graphs (1)



If every conditional independence property of the distribution is reflected in the graph, and vice versa, then the graph is said to be a perfect map for that distribution

Directed vs. Undirected Graphs (2)





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

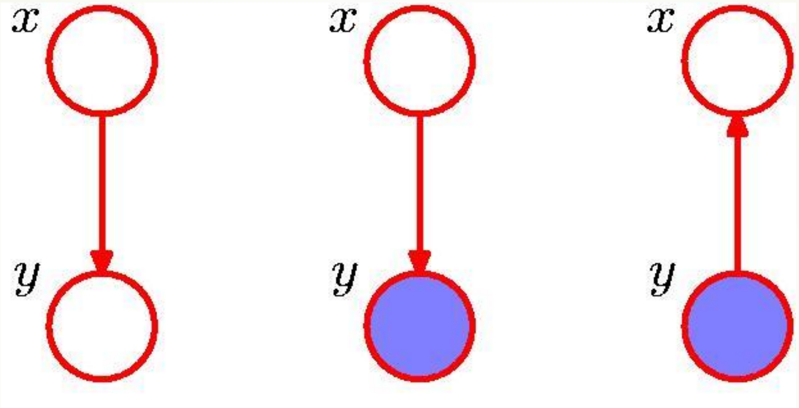
$$A \not\!\perp\!\!\!\perp B \mid C$$

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

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

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

Inference in Graphical Models



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

Inference on a Chain

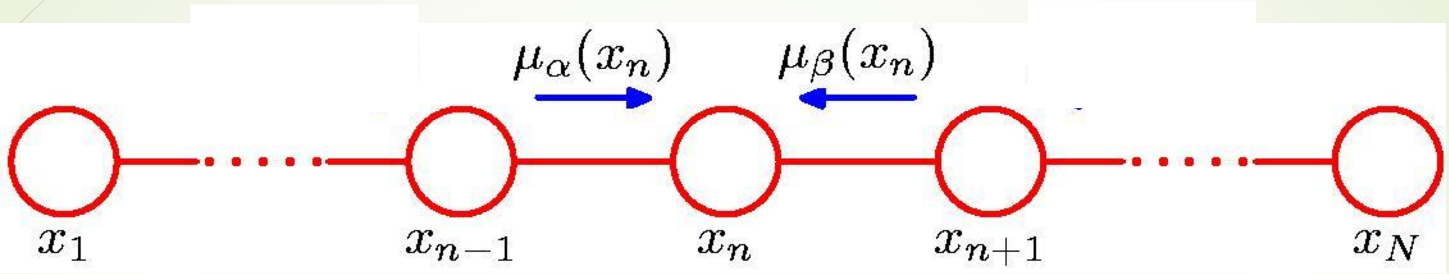
$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

Each discrete variables have K states, in which case each potential function $\psi_{n-1,n}(x_{n-1}, x_n)$ comprises a $K \times K$ table, and so the joint distribution has $(N - 1)K^2$ parameters

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

Complexity: $O(K^N)$

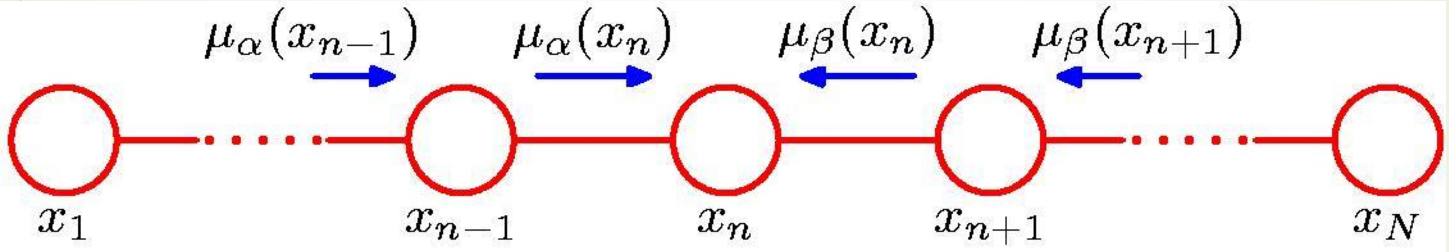
Inference on a Chain



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

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

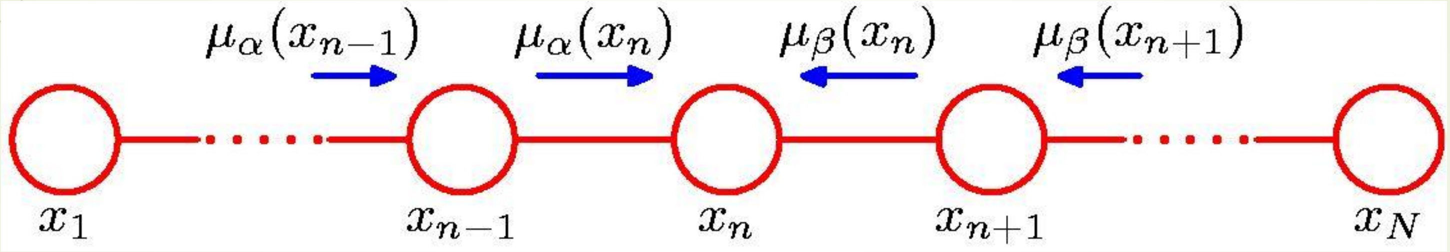
Inference on Markov Chain: Message Passing



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

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

Inference on Markov Chain: Message Passing



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

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

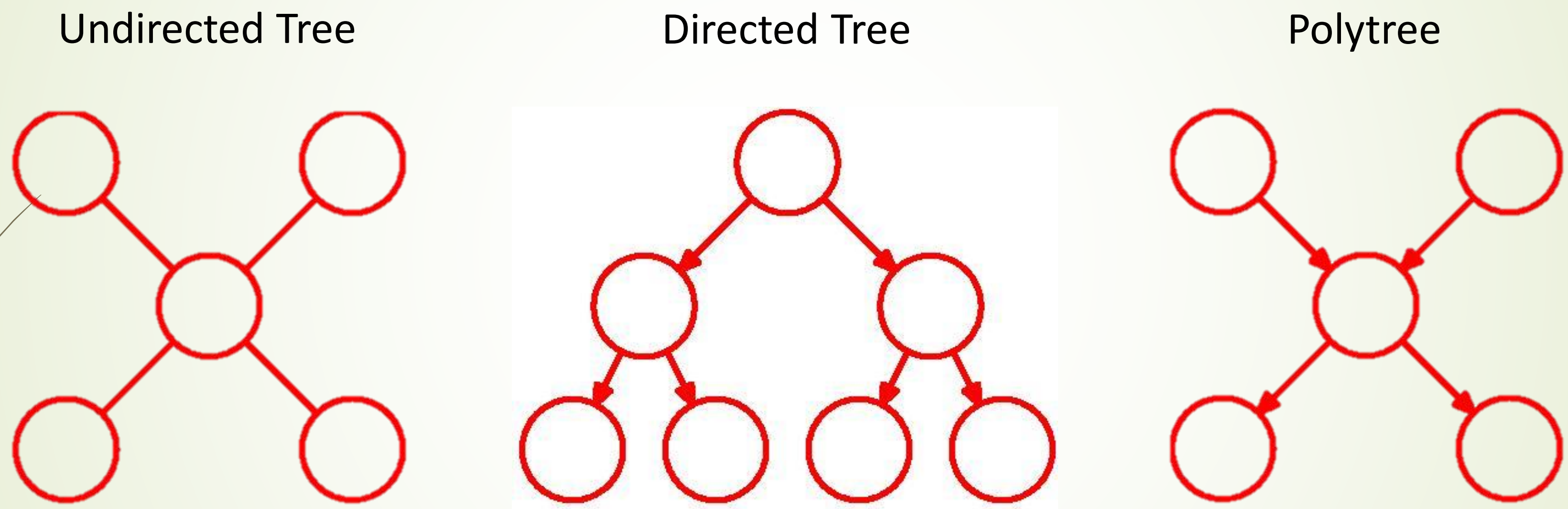
Linear-Complexity Inference on Markov Chain

To compute local marginals:

- Recursively compute and store all forward messages, $\mu_\alpha(x_n)$.
- Recursively compute and store all backward messages, $\mu_\beta(x_n)$.
- Compute $$Z = \sum_{x_n} \mu_\alpha(x_n)\mu_\beta(x_n)$$ Complexity: $O(K)$
- Compute $$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$$

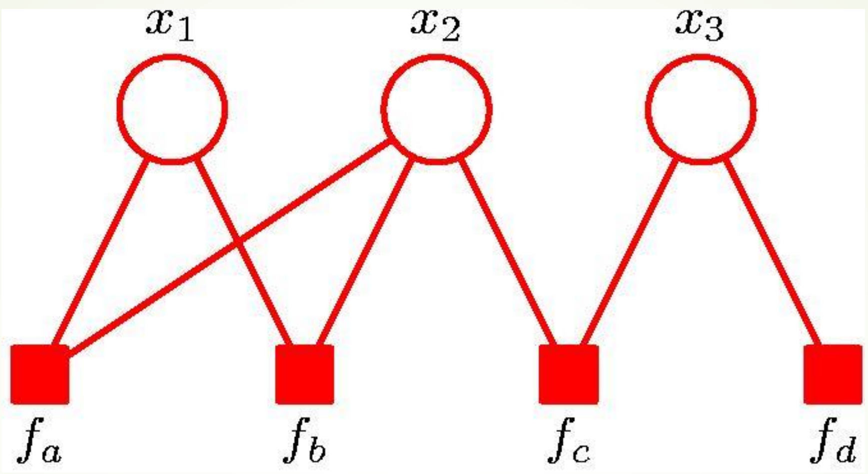
for all variables required.

Trees



The ***sum-product*** algorithm provides an efficient framework for exact inference in tree-structured graphs

Factor Graphs



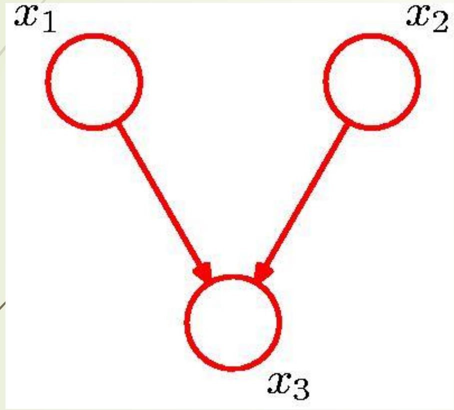
Factor graphs are said to be ***bipartite*** because they consist of two distinct kinds of nodes, and all links go between nodes of opposite type.

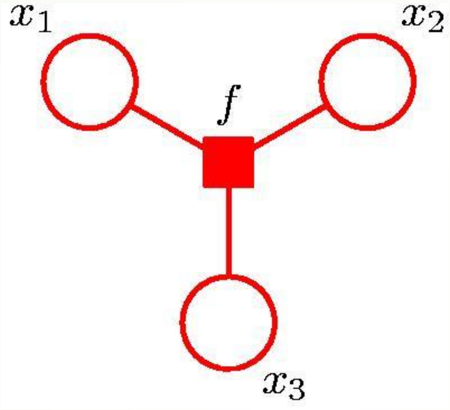
- Both directed and undirected graphs allow a global function of several variable to be expressed as a product of factors over subsets of those variables.
- Factor graphs make this decomposition explicit by introducing additional nodes for the factors themselves in addition to the nodes representing the variables.

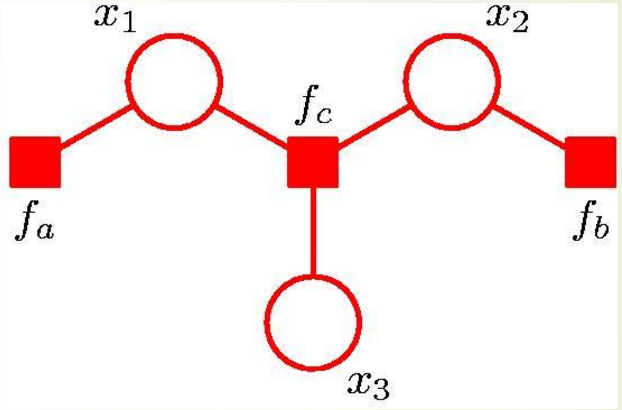
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

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

Factor Graphs from Directed Graphs





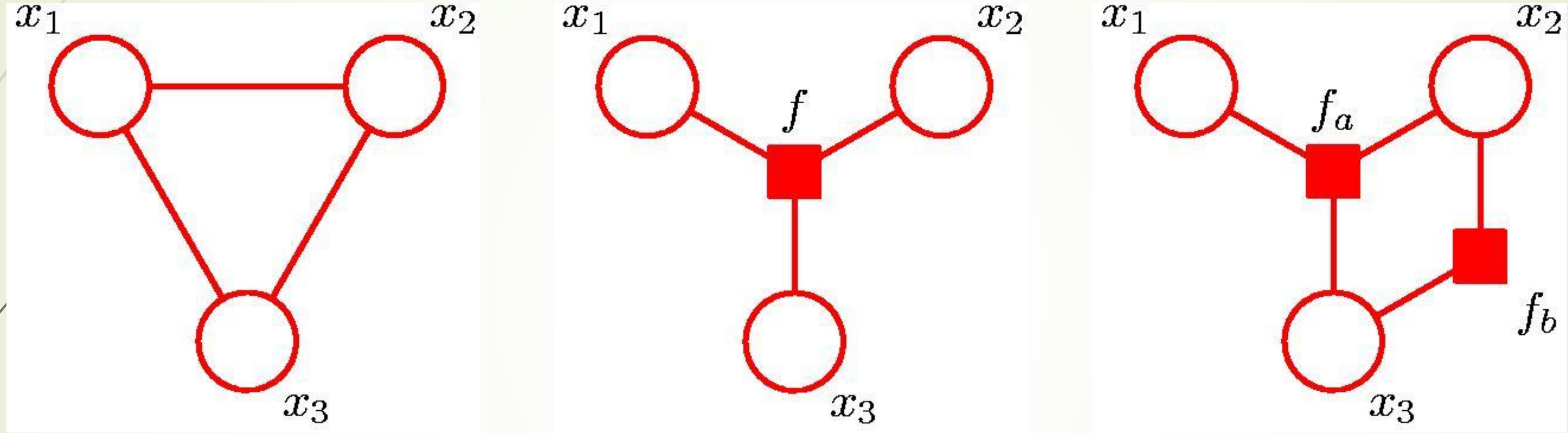


$$\begin{aligned} p(\mathbf{x}) &= p(x_1)p(x_2) \\ &\quad p(x_3|x_1,x_2) \end{aligned}$$

$$\begin{aligned} & f(x_1,x_2,x_3) = \\ & \quad p(x_1)p(x_2)p(_3|x_1,x_2) \end{aligned}$$

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

Factor Graphs from Undirected Graphs



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

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

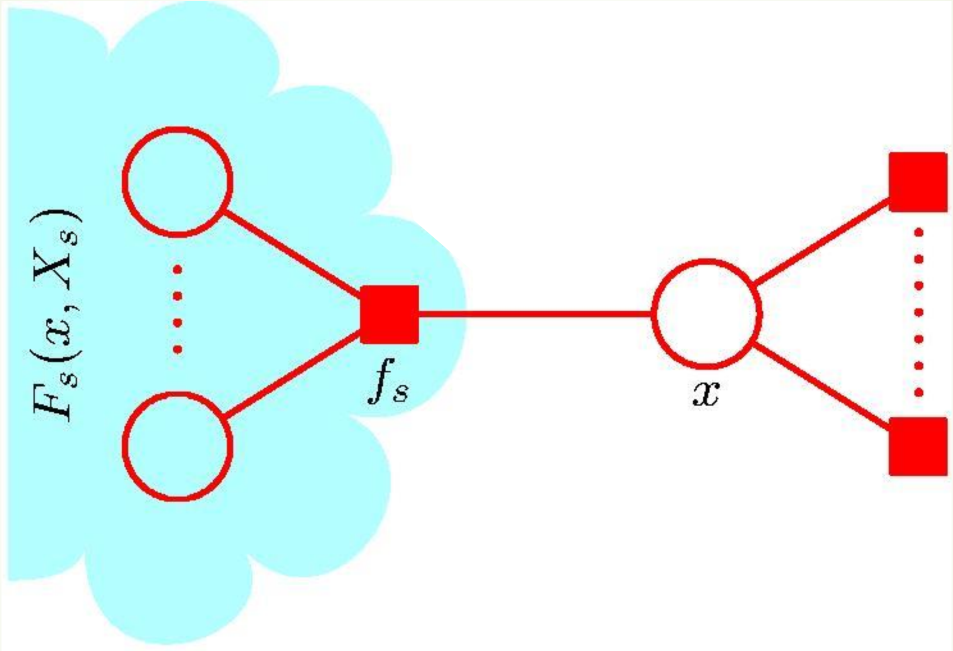
$$\begin{aligned} & f_a(x_1, x_2, x_3) f_b(x_2, x_3) \\ = \quad & \psi(x_1, x_2, x_3) \end{aligned}$$

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

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

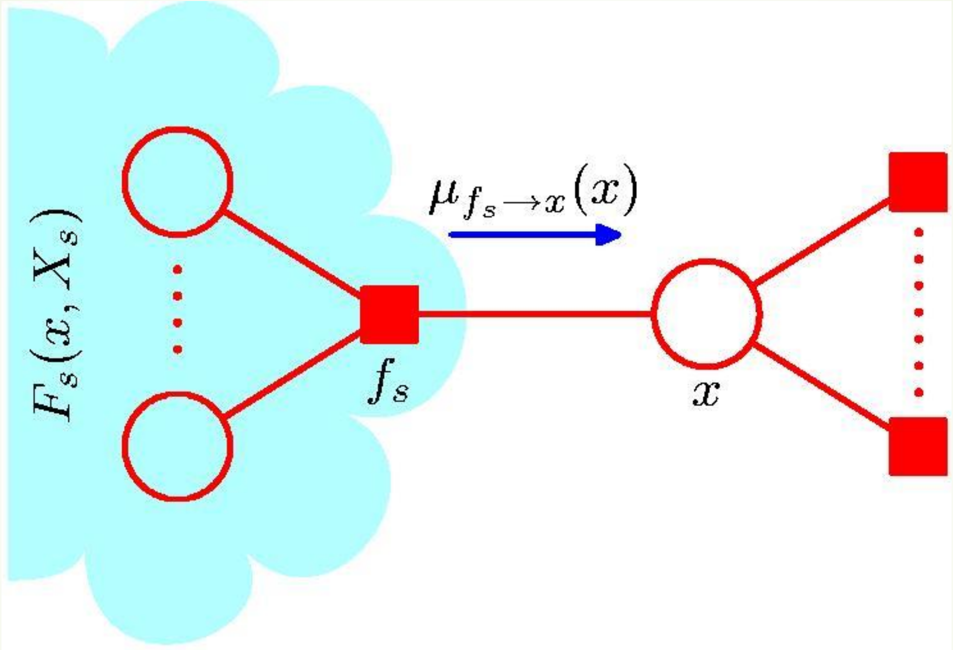
The Sum-Product Algorithm (2)



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

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

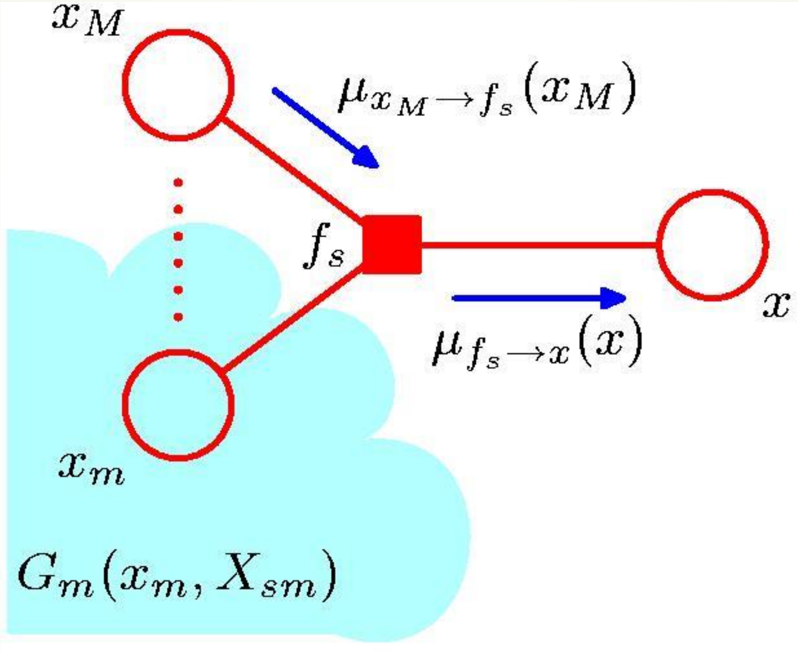
The Sum-Product Algorithm (3)



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

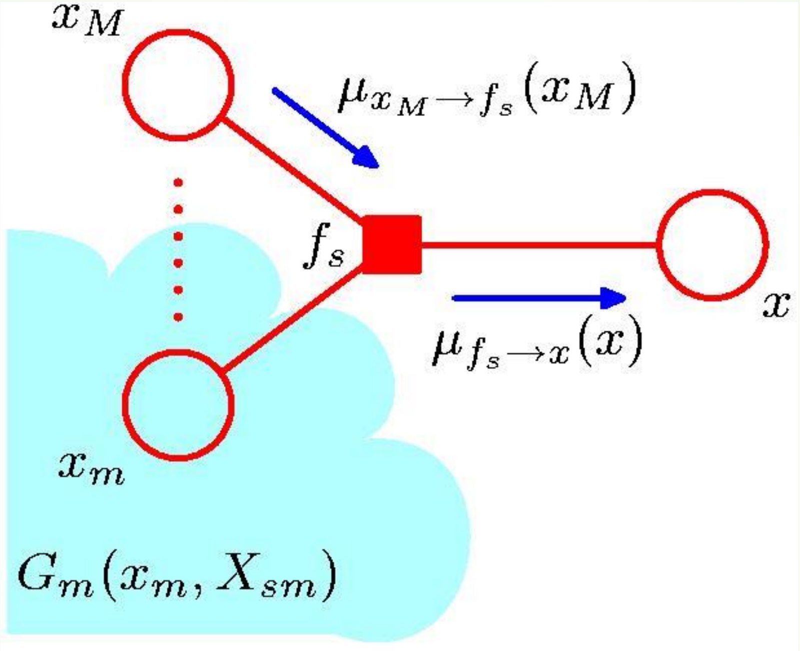
$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

The Sum-Product Algorithm (4)



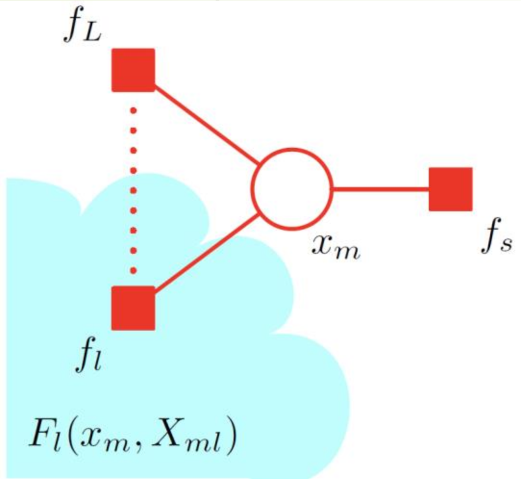
$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) G_1(x_1, X_{s1}) \ldots G_M(x_M, X_{sM})$$

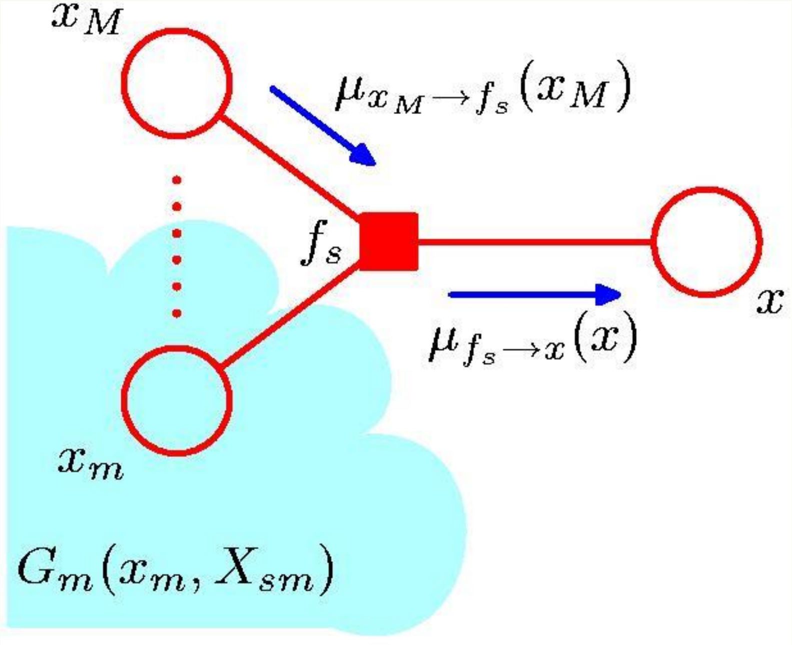
The Sum-Product Algorithm (5)



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

The Sum-Product Algorithm (6)



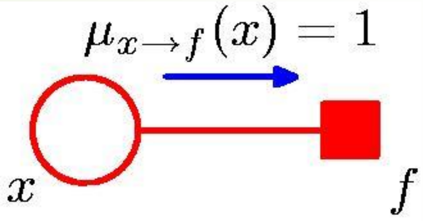


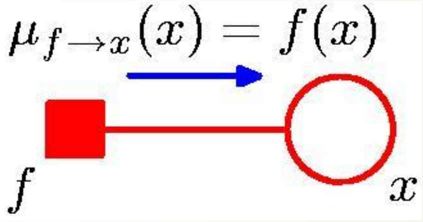
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

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

The Sum-Product Algorithm (7)

- Initialization

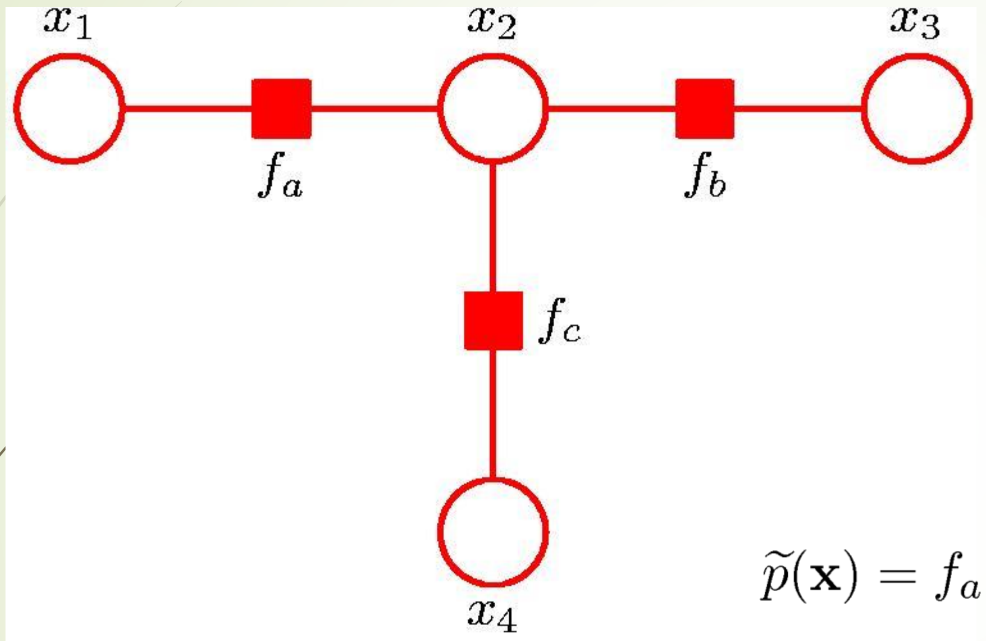




The Sum-Product Algorithm (8)

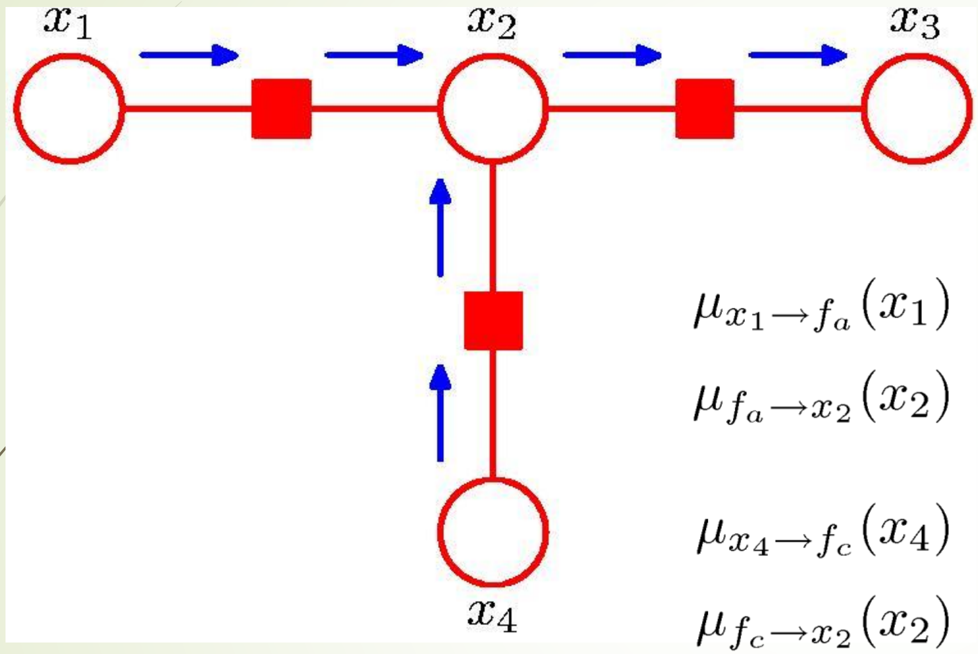
- To compute local marginals:
  - Pick an arbitrary node as root
  - Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  - Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  - Compute the product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

Sum-Product: Example (1)



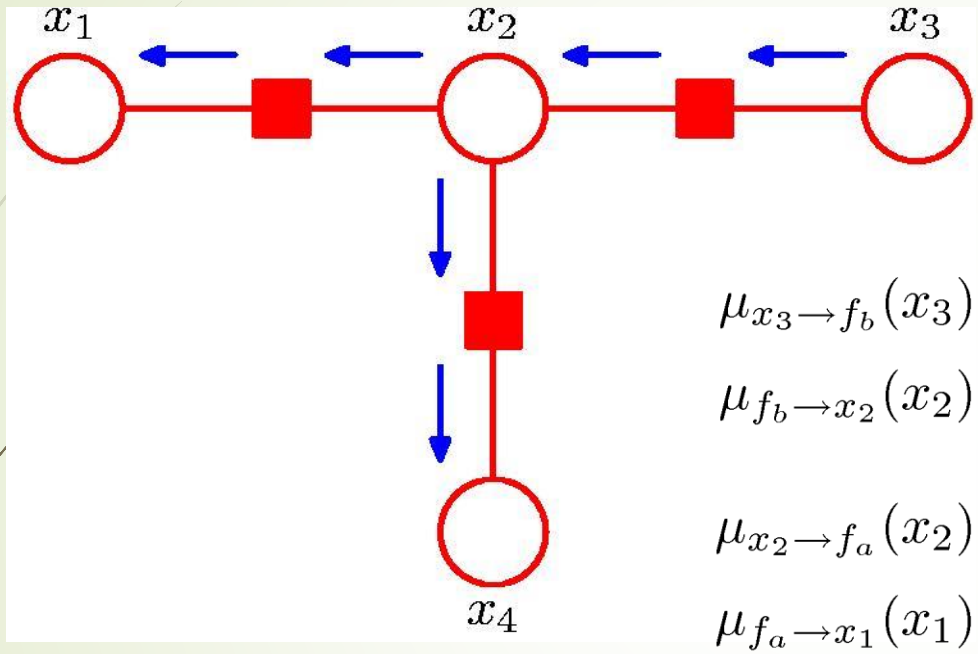
$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

Sum-Product: Example (2)



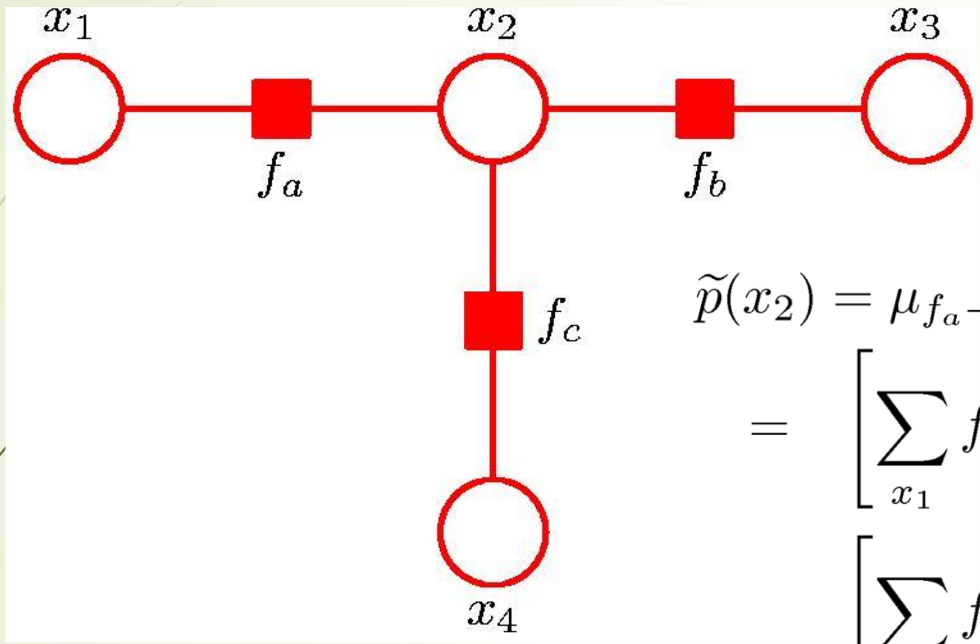
$$
\begin{aligned}
\mu_{x_1 \to f_a}(x_1) &= 1 \\
\mu_{f_a \to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\
\mu_{x_4 \to f_c}(x_4) &= 1 \\
\mu_{f_c \to x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\
\mu_{x_2 \to f_b}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_b \to x_3}(x_3) &= \sum_{x_2} f_b(x_2,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

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 $\mathbf{x}^{\max}$ 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)

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \ldots \max_{x_M} p(\mathbf{x})$$

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

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

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)

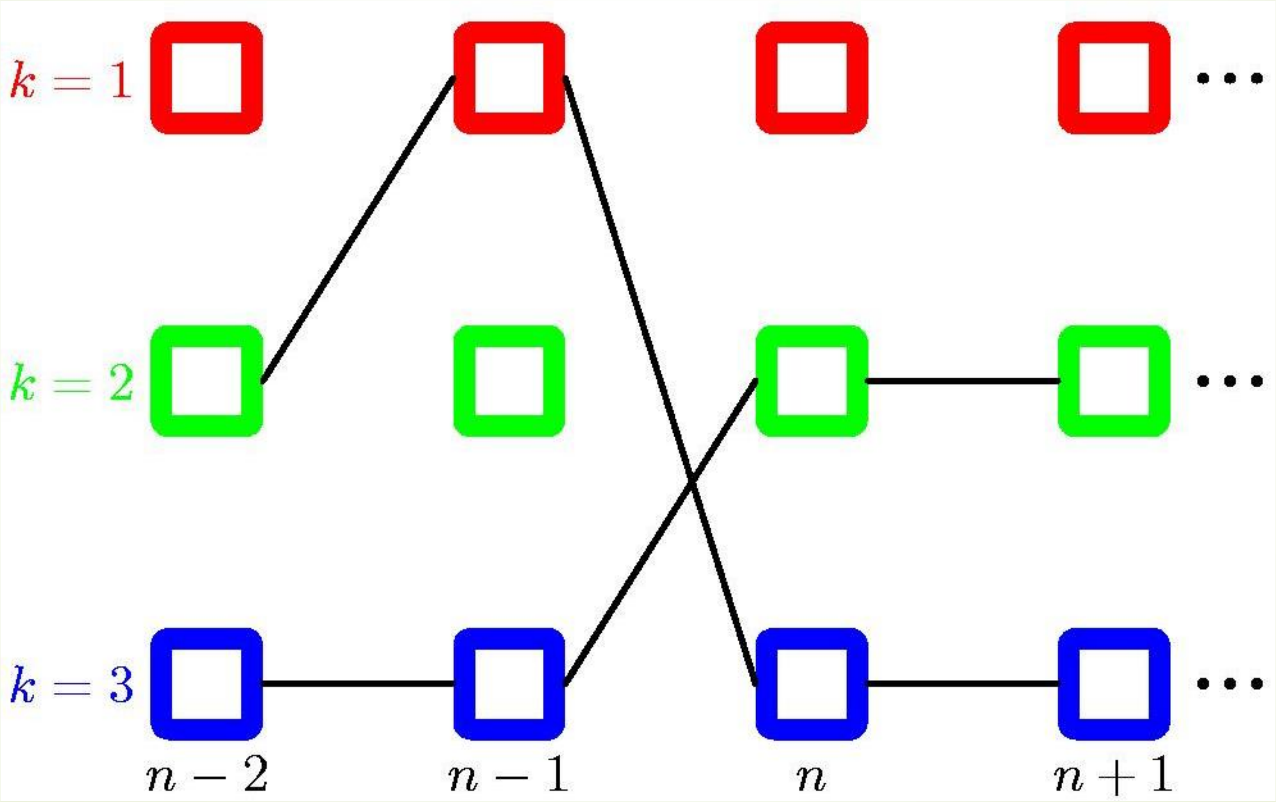
$$
\begin{aligned}
p^{\max} &= \max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right] \\
x^{\max} &= \arg\max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]
\end{aligned}
$$

Back-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.