GRAPHICAL MODELS

J. Elder

CSE 6390/PSYC 6225 Computational Modeling of Visual Perception

Credits

- These slides were modified from:
  - Christopher Bishop, Microsoft UK

PART 1
DIRECTED GRAPHICAL MODELS (BAYES NETS)

Directed Graphical Models and the Role of Causality

- Bayes nets are directed acyclic graphs in which each node represents a random variable.
- Arcs signify the existence of direct causal influences between linked variables.
- Strengths of influences are quantified by conditional probabilities

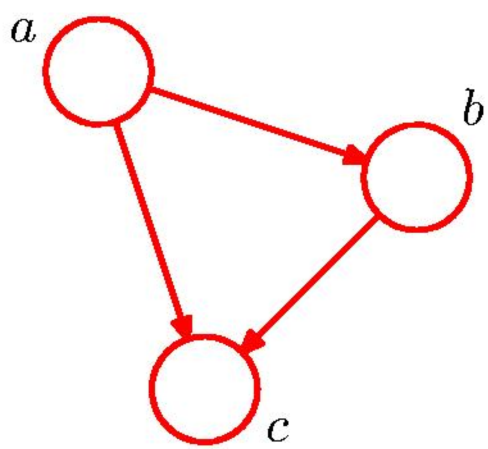
$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

where pa_k is the set of 'parent' nodes of node k.

- NB: For this to hold it is critical that the graph be acyclic.

Bayesian Networks

- Directed Acyclic Graph (DAG)



From the definition of conditional probabilities (product rule):

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

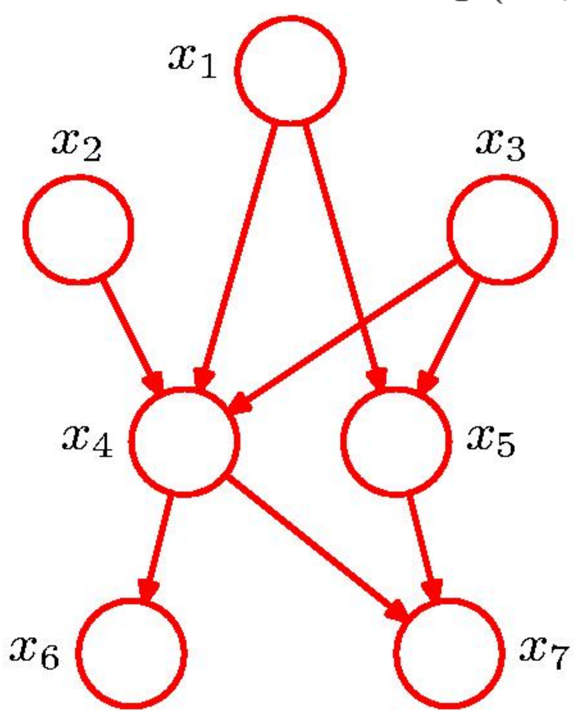
In general:

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

This corresponds to a complete graph.

Bayesian Networks

- However, many systems have sparser causal relationships between their variables.



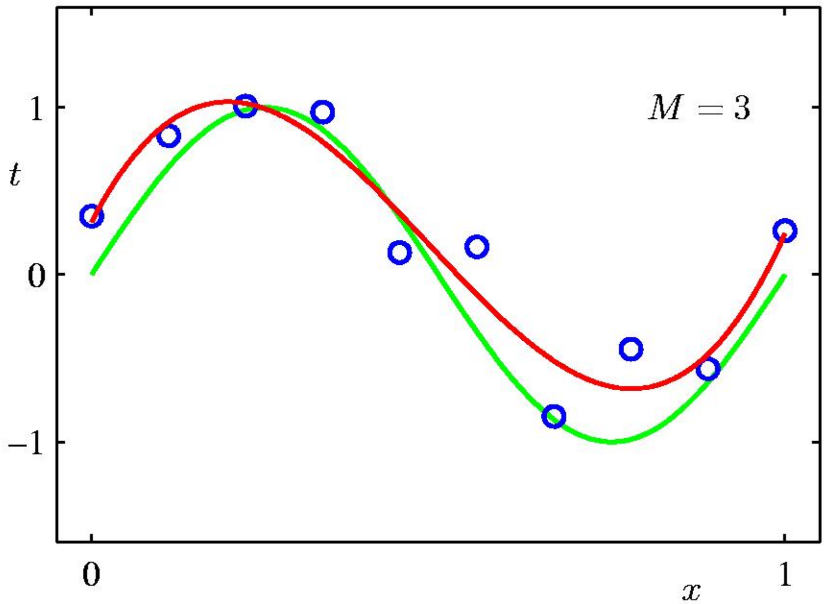
$$
\begin{aligned}
p(x_1, \ldots, x_7) \quad = \quad & p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\
& p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)
\end{aligned}
$$

General Factorization

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

Examples of Bayesian Networks

Example: Bayesian Curve Fitting



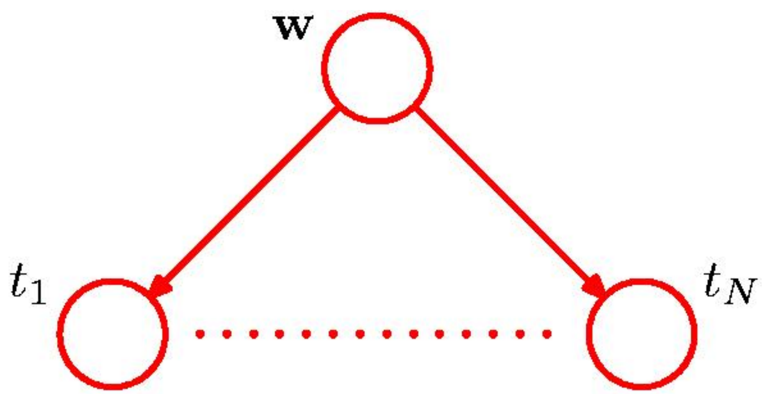
Polynomial

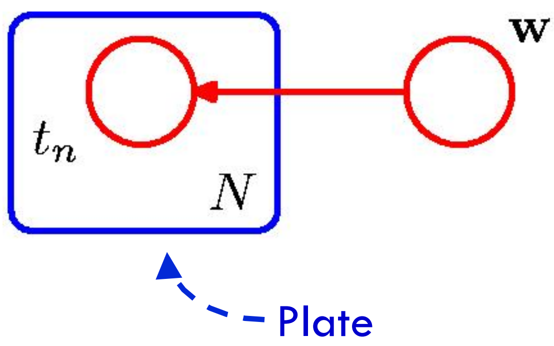
$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$$

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

Bayesian Curve Fitting

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

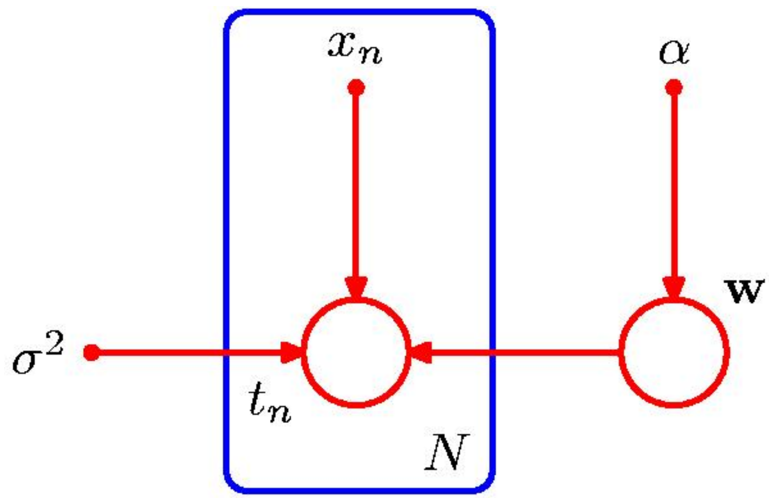




Bayesian Curve Fitting

- Input variables and explicit hyperparameters

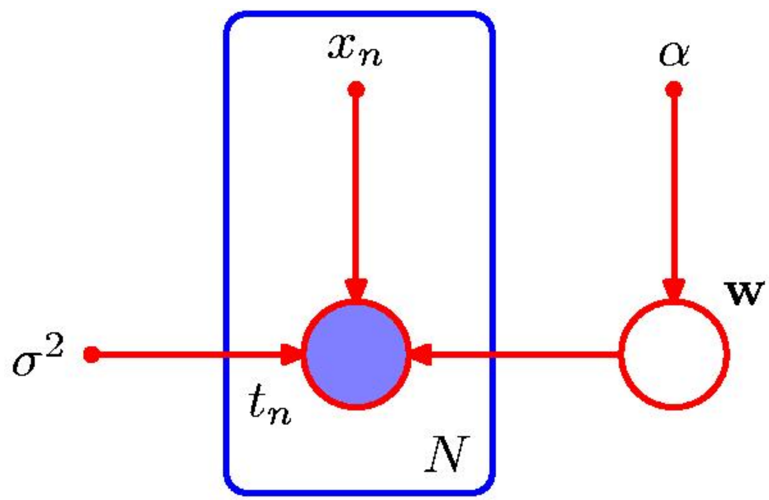
$$p(\mathbf{t}, \mathbf{w}|\mathbf{x}, \alpha, \sigma^2) = p(\mathbf{w}|\alpha) \prod_{n=1}^{N} p(t_n|\mathbf{w}, x_n, \sigma^2).$$



Bayesian Curve Fitting —Learning

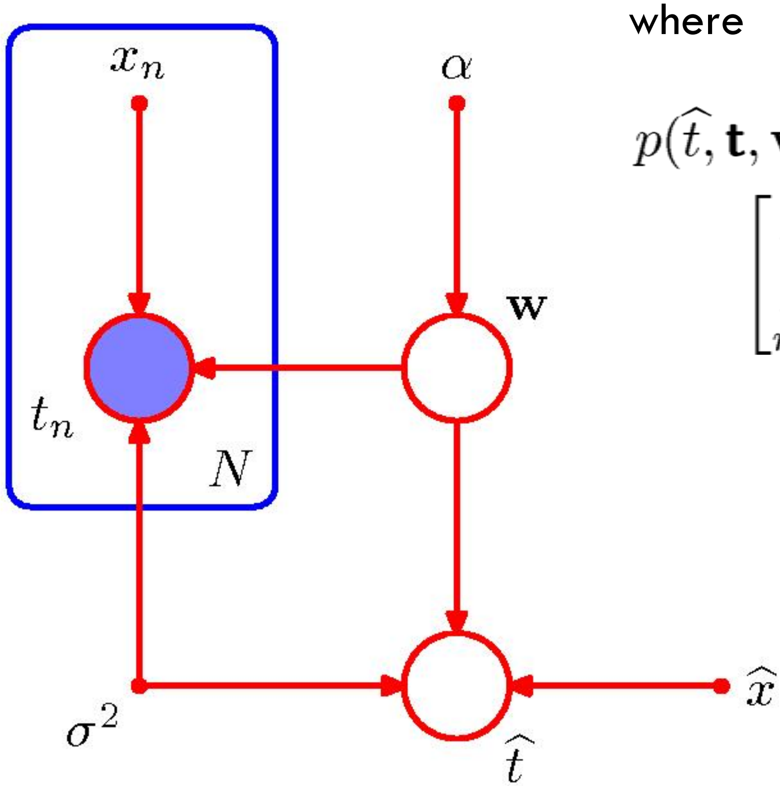
- Conditioning on training data: we represent the fact that a variable has been observed (and is therefore fixed) by shading the corresponding node.

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



Bayesian Curve Fitting - Prediction

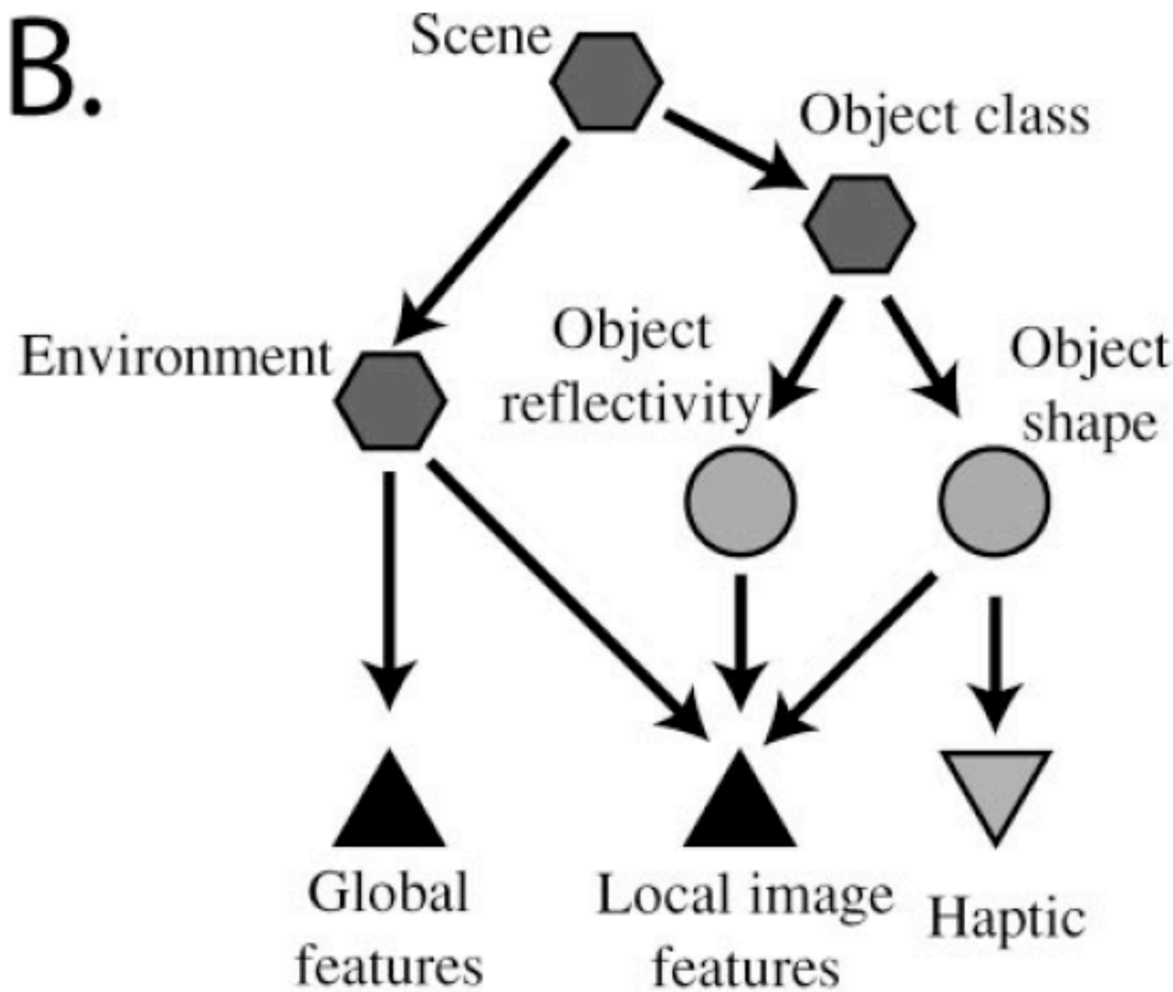
Predictive distribution: $p(\widehat{t}|\widehat{x}, \mathbf{x}, \mathbf{t}, \alpha, \sigma^2) \propto \int p(\widehat{t}, \mathbf{t}, \mathbf{w}|\widehat{x}, \mathbf{x}, \alpha, \sigma^2)\,\mathrm{d}\mathbf{w}$



where

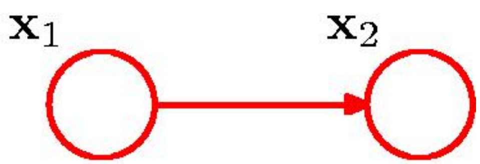
$$p(\widehat{t}, \mathbf{t}, \mathbf{w}|\widehat{x}, \mathbf{x}, \alpha, \sigma^2) = \left[\prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}, \sigma^2)\right] p(\mathbf{w}|\alpha)p(\widehat{t}|\widehat{x}, \mathbf{w}, \sigma^2)$$

Generative Models of Perception



Discrete Variables

- 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

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

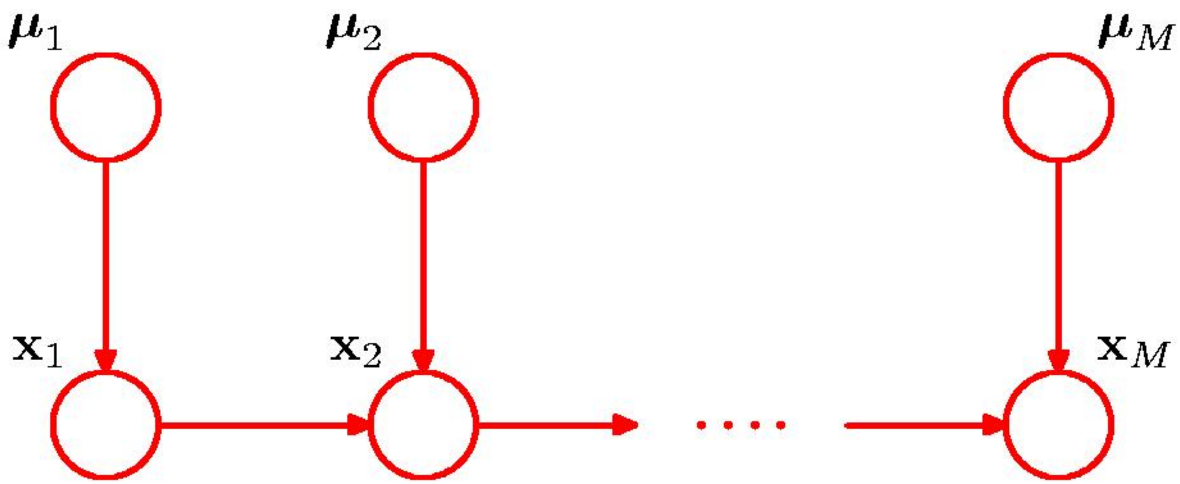
Discrete Variables

- General distributions require many parameters.
- General joint distribution over M variables:
  K^M -1 parameters
- It is thus extremely important to identify structure in the system that corresponds to a sparser graphical model and hence fewer parameters.

Discrete Variables

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

Discrete Variables: Bayesian Parameters

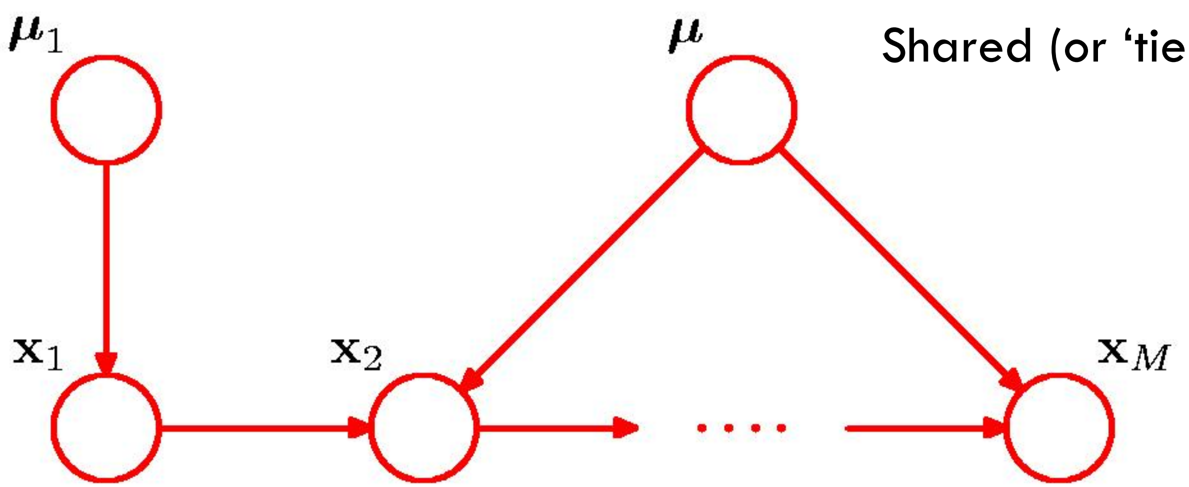


$$p\left(\{\mathbf{x}_m, \boldsymbol{\mu}_m\}\right) = p\left(\mathbf{x}_1 \,|\boldsymbol{\mu}_1\,\right) p\left(\boldsymbol{\mu}_1\right) \prod_{m=2}^{M} p\left(\mathbf{x}_m | \mathbf{x}_{m-1}, \boldsymbol{\mu}_m\right) p\left(\boldsymbol{\mu}_m\right)$$

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

Discrete Variables: Bayesian Parameters

- The number of parameters can also be reduced if parameters can be shared, or 'tied':

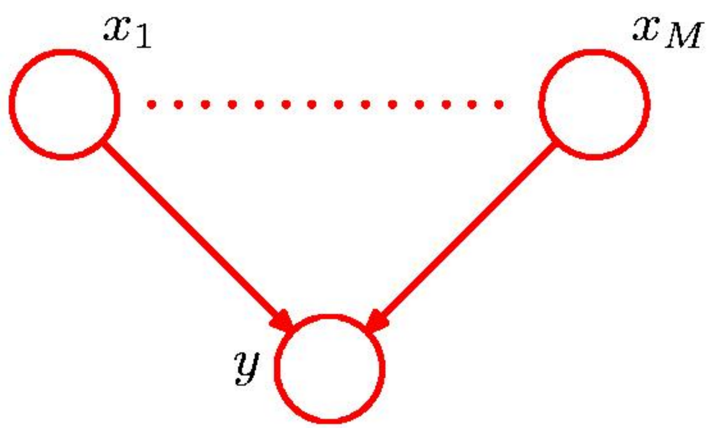


Shared (or 'tied') prior

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

Parameterized Conditional Distributions

- The number of parameters can also be reduced by restricting the generality of conditional distributions.



If x_i and y are binary random variables,

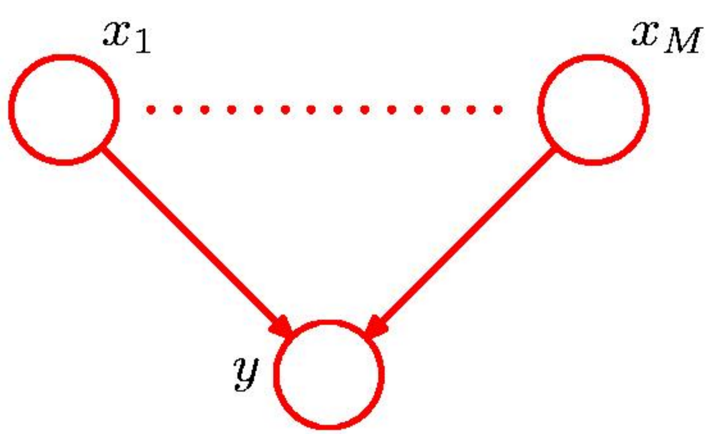
then the general form of $p(y|x_1 \ldots x_M)$ has 2^M parameters.

Parameterized Conditional Distributions

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

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

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

- Can find the mean and covariance of the joint Gaussian distribution recursively:

$$x_i = \sum_{j\in pa_i} w_{ij}x_j + b_i + \sqrt{v_i}\varepsilon_i$$

$$E\left[x_i\right] = \sum_{j\in pa_i} w_{ij}E\left[x_j\right] + b_i \qquad \mathrm{cov}\left[x_i, x_j\right] = \sum_{k\in pa_j} w_{jk}\,\mathrm{cov}\left[x_i, x_k\right] + I_{ij}v_j$$

Linear-Gaussian Models

- Vector variables

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

PART 2. CONDITIONAL INDEPENDENCE

Conditional Independence

- *a* is independent of *b* given *c*

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

- Equivalently

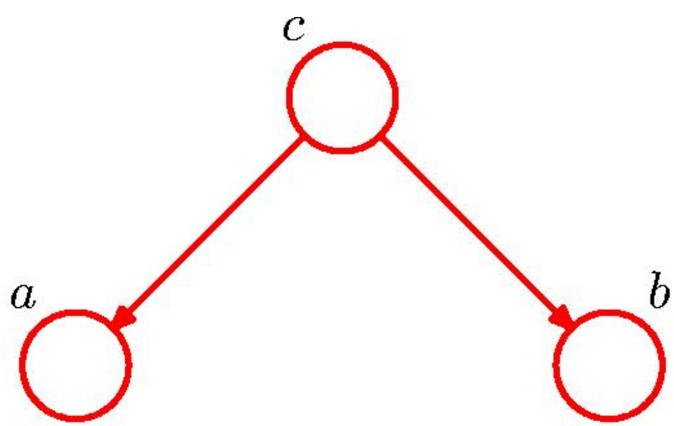
$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

- Notation

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

Conditional Independence: Example 1

- In this system, *a* is not directly causal on *b*, and *b* is not directly causal on *a*.
- Yet *a* and *b* are not, in general, independent.



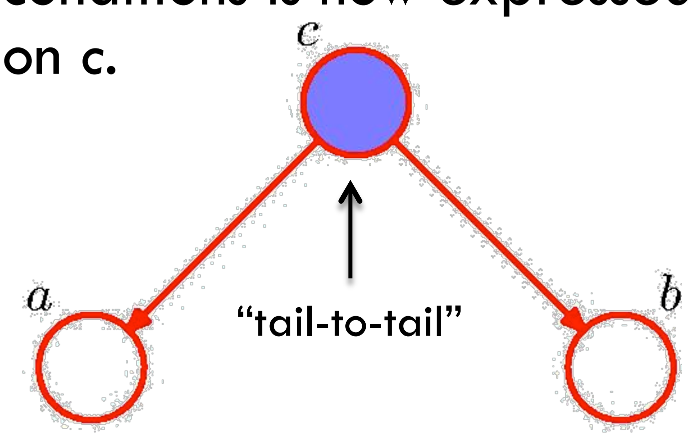
$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

$$p(a,b) = \sum_c p(a \mid c)p(b \mid c)p(c) \neq p(a)p(b)$$

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

Conditional Independence: Example 1

- We can also consider the statistical relationship between a and *b* once c has been observed.
- In this case, c is no longer a random variable – it has a fixed value.
- The statistical relationship between *a* and *b* under these conditions is now expressed as the joint distibution, conditioned on c.

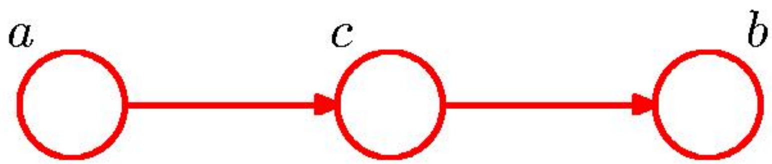


$$\begin{aligned} p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

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

- Thus observation of c blocks the statistical relationship between *a* and *b*.

Conditional Independence: Example 2



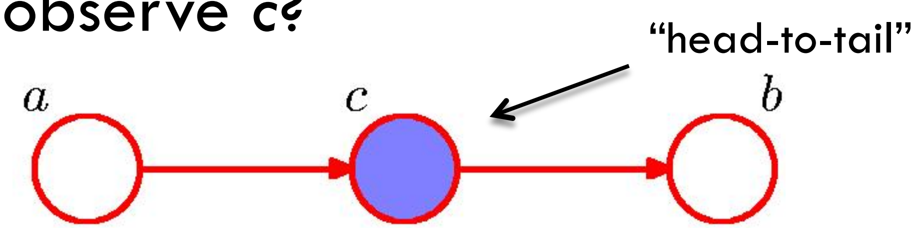
$$p(a,b,c) = p(a)p(c|a)p(b|c)$$

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

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

Conditional Independence: Example 2

- What if we observe c?

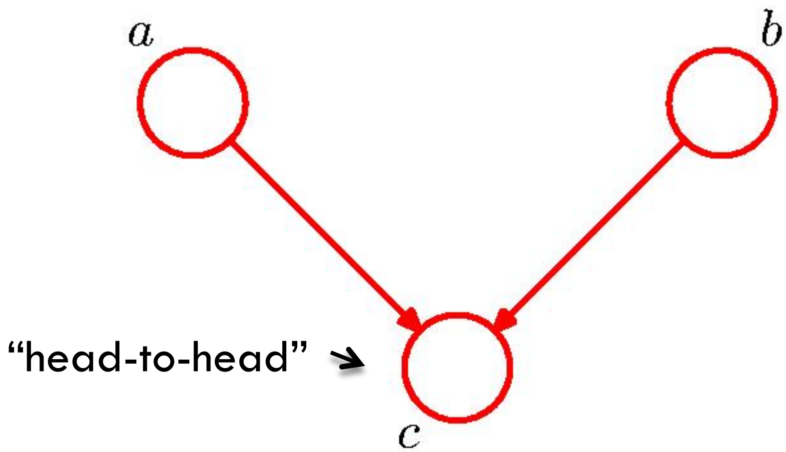


$$\begin{aligned} p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\ &= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\ &= p(a|c)p(b|c) \end{aligned}$$

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

- Thus observing (conditioning on) c renders *a* and *b* independent.

Conditional Independence: Example 3



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

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

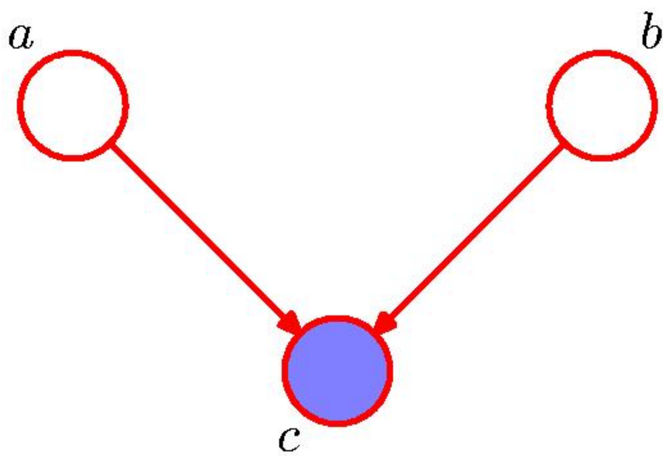
$$\longrightarrow \quad a \perp\!\!\!\perp b \mid \emptyset$$

- In this case, a and b are unconditionally independent (when c is not observed.)

END OF LECTURE
NOV 17, 2010

Conditional Independence: Example 3

- What if c is observed?



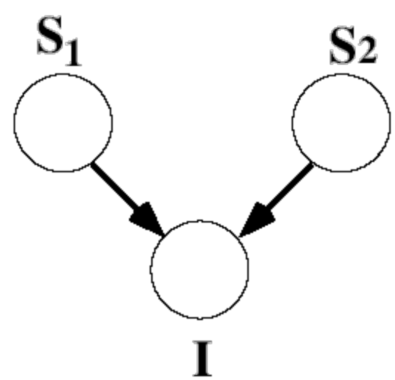
$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(b)p(c|a, b)}{p(c)}$$

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

- In this case, observation of c makes *a* and *b* statistically dependent!
- This is known as "explaining away"

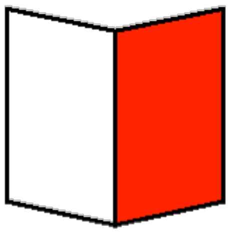
Causation

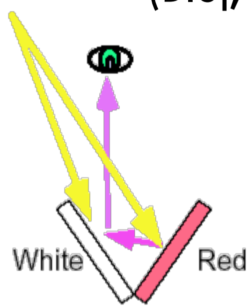
- Two events do not become relevant to each other merely by virtue of predicting a common consequence, but they do become relevant when the consequence is actually observed.

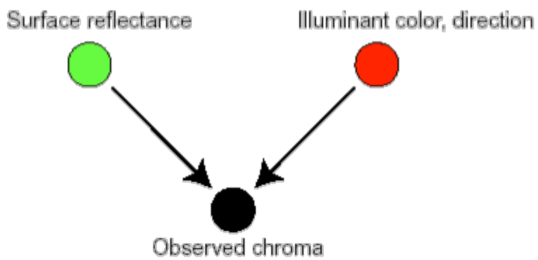


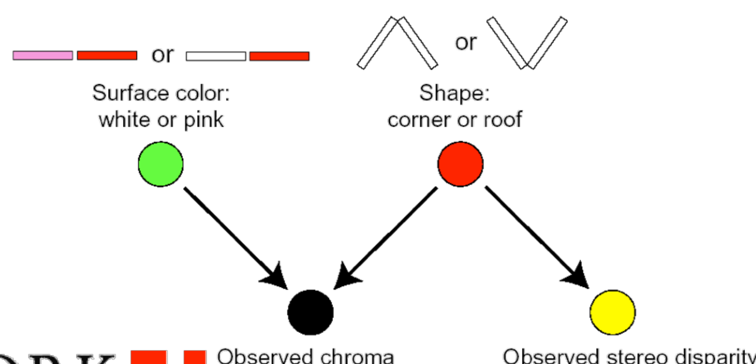
Example of Explaining Away: The Chromatic Mach Card

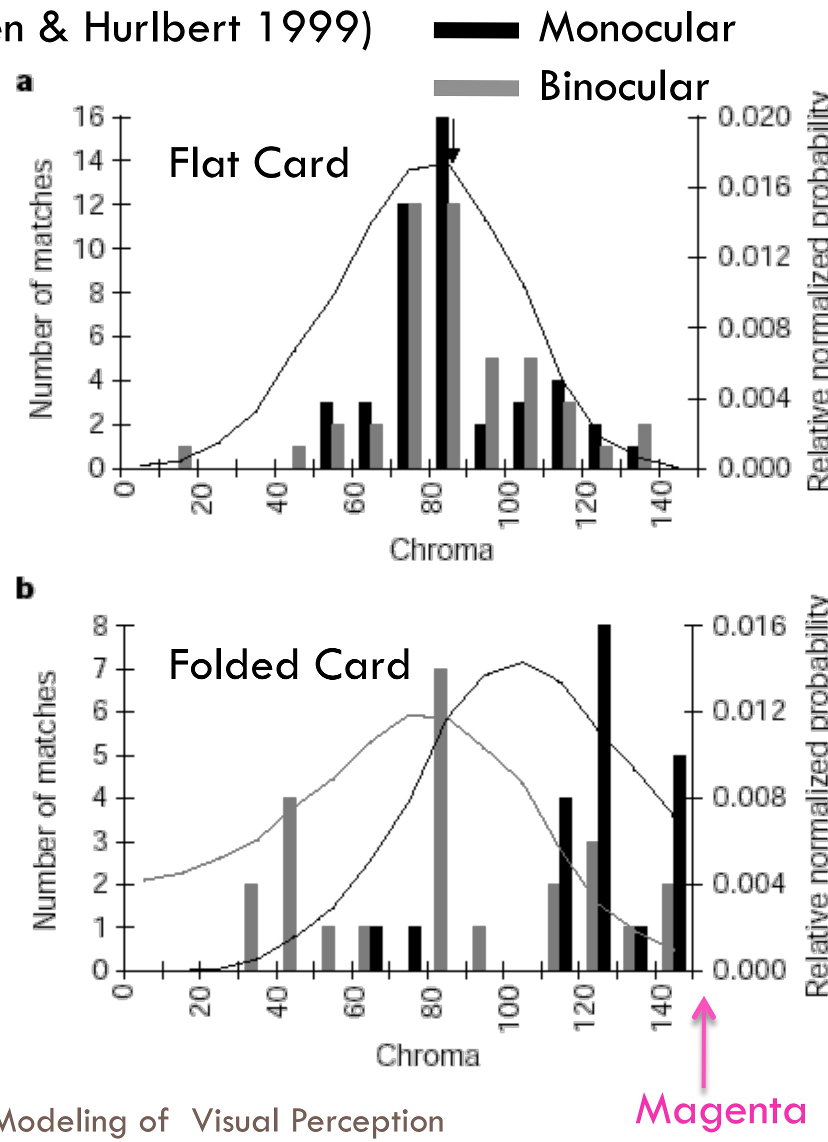
(Bloj, Kersten & Hurlbert 1999)





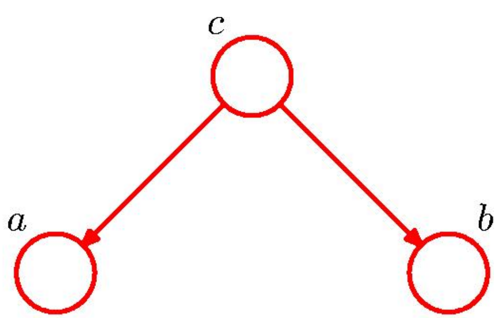




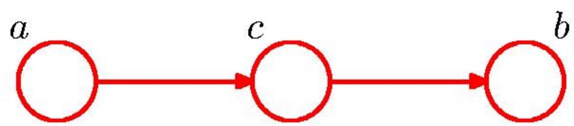


3 Basic Forms

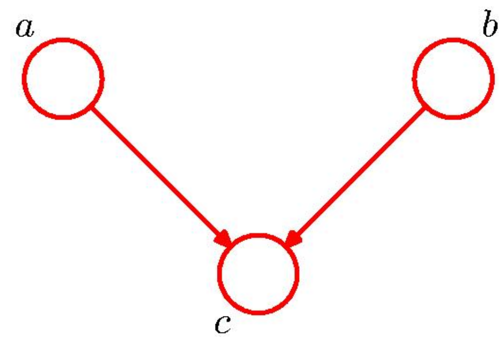
Tail-to-Tail



Head-to-Tail



Head-to-Head



D-separation

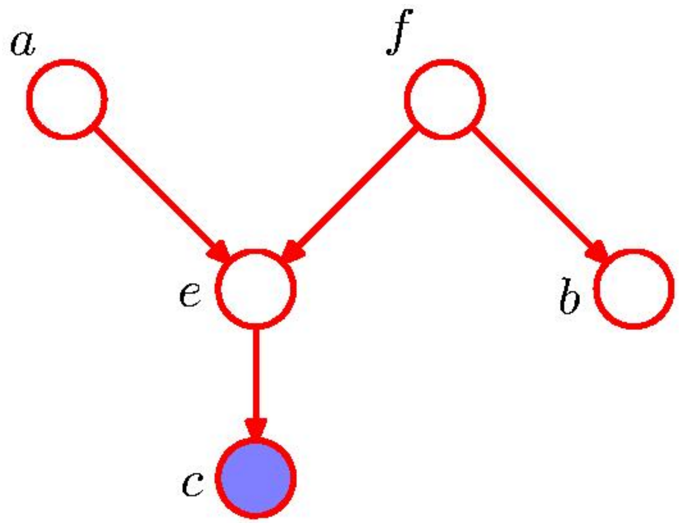
- A, B, and C are non-intersecting subsets of nodes in a directed graph.
- We wish to determine whether A and B are independent when conditioned on C.
- A path from A to B is said to be blocked if it contains a node such that either
    1. the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C, or
    2. 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

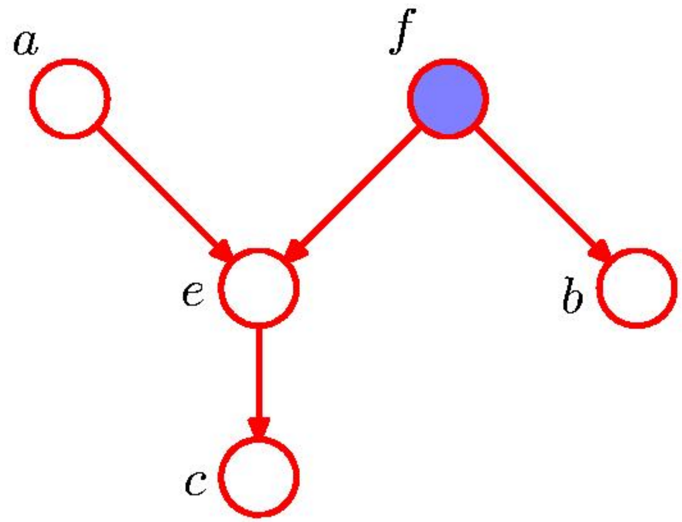
$$A \perp\!\!\!\perp B \mid C$$

END OF LECTURE
NOV 22, 2010

D-separation: Example

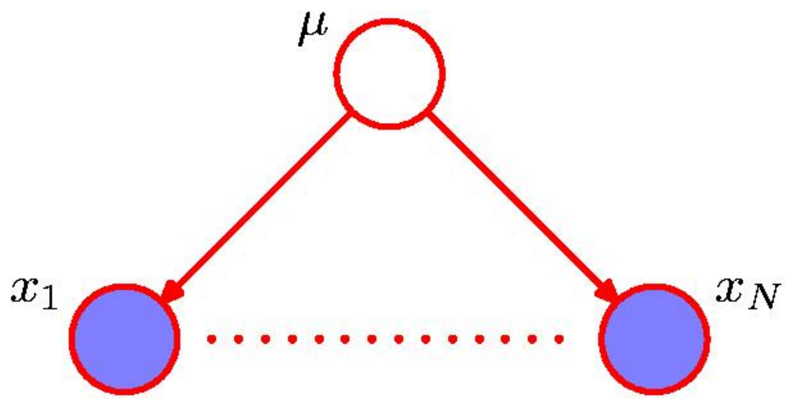
Are *a* and *b* independent when conditioned on c?





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

D-separation: I.I.D. Data

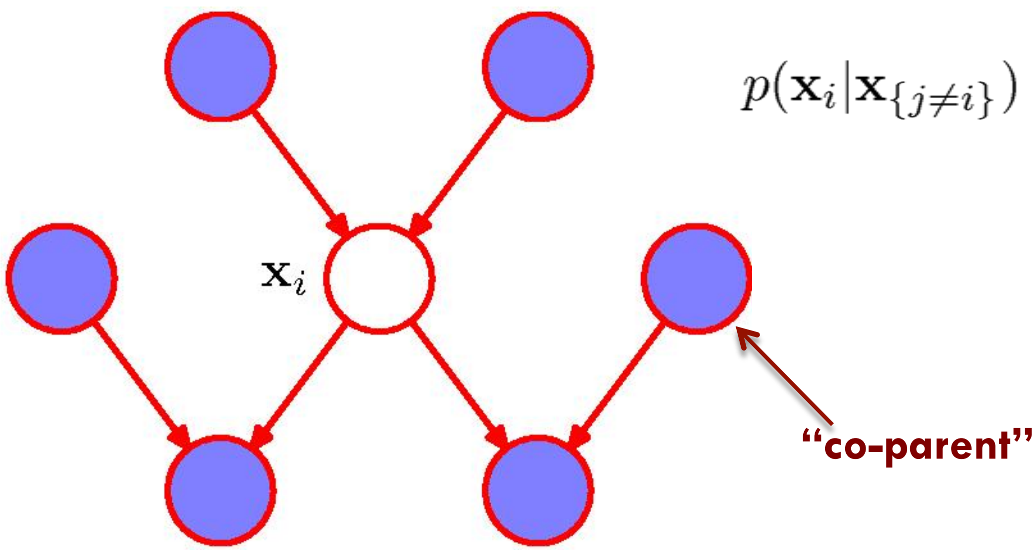


$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu)$$

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

The Markov Blanket

- The Markov blanket of a node x_i is the minimal set of nodes that separate x_i from the rest of the graph.



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

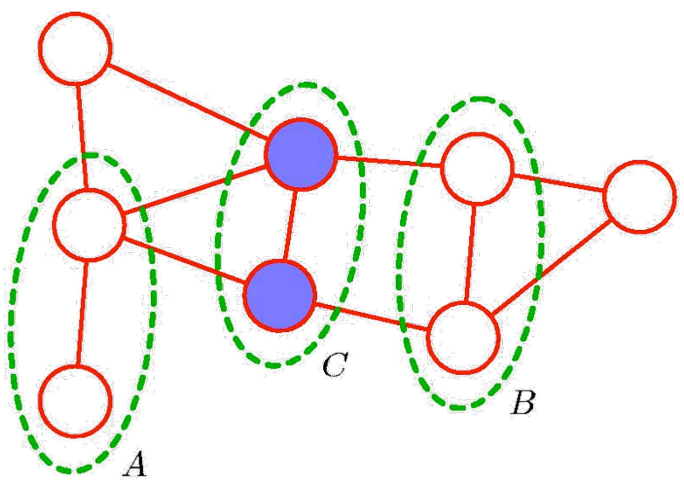
- Factors independent of x_i cancel.

$$= \frac{p(x_i \mid pa_i)\prod_{i\in pa_k} p(x_k \mid pa_k)}{\int p(x_i \mid pa_i)\prod_{i\in pa_k} p(x_k \mid pa_k)\,dx_i}$$

PART 2
UNDIRECTED GRAPHICAL MODELS (MARKOV RANDOM FIELDS)

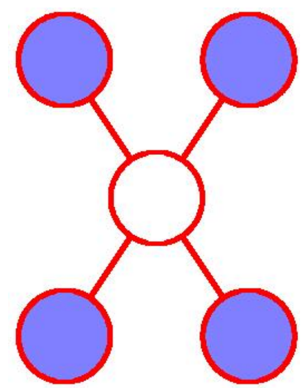
Markov Random Fields

- For MRFs, conditional independence is determined by graph separation: if all paths between A and B go through C, A and B are independent when conditioned on C.
- The Markov blanket of a node *x* is just the set of nodes directly connected to *x*. This is also known as the **neighbourhood** of x.



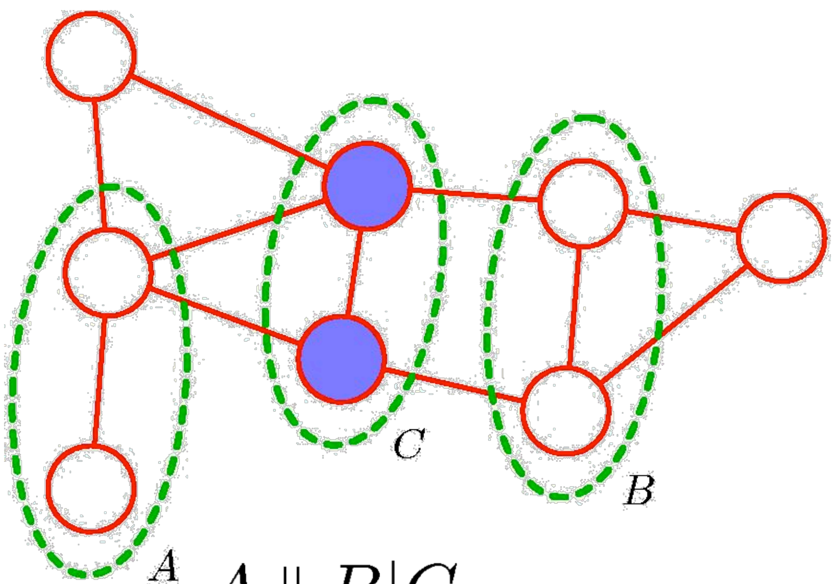
$$A \perp\!\!\!\perp B \mid C$$

Markov Blanket



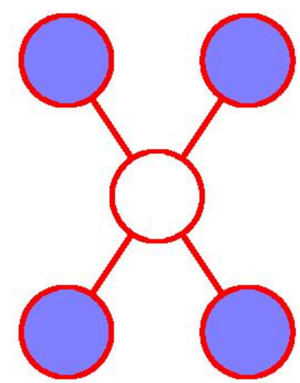
Markov Random Fields

- Thus, as for a directed graphical model, an MRF defines a set of conditional independence relationships between its variables.
- In fact, an MRF is defined by these conditional independence relationships (Markov properties).



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

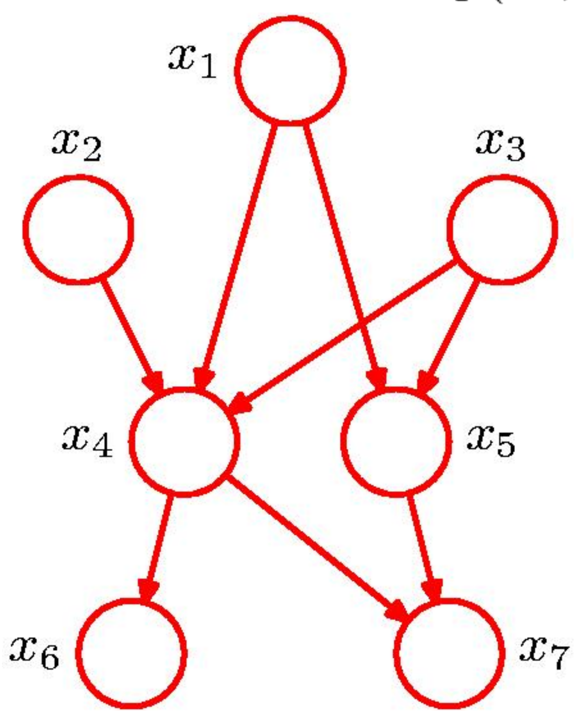
Markov Blanket



Factoring

- Recall how we factor **directed** graphs
- We seek a comparable method for **undirected** graphs.

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

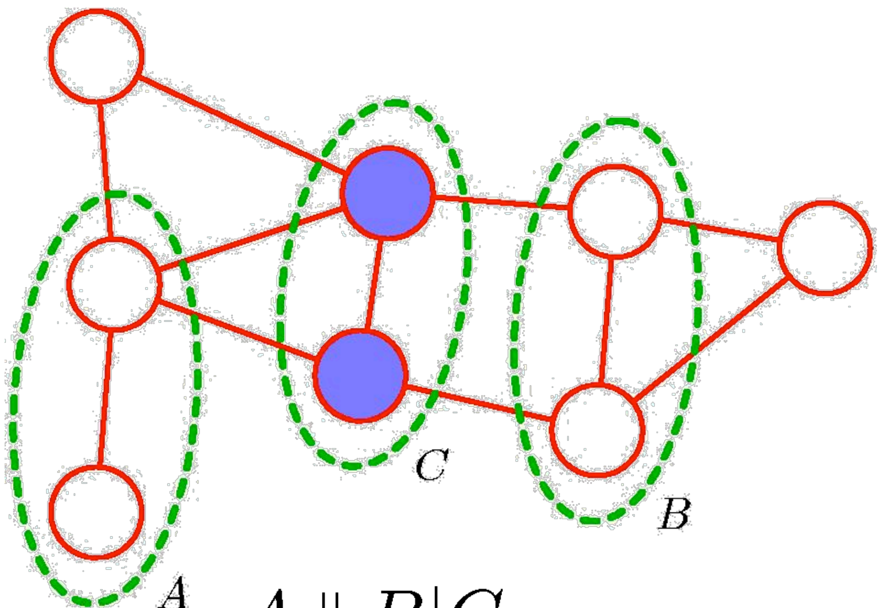


General Factorization

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

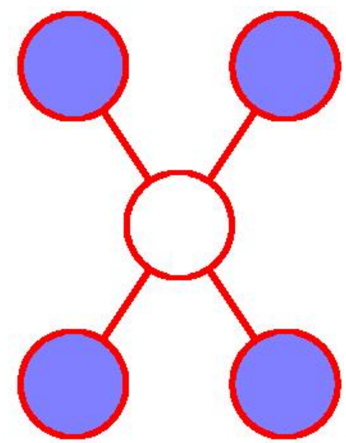
Factoring

- Nodes that are not directly connected are rendered independent by conditioning on the intervening nodes.
- Such nodes must therefore be in different factors in order for the conditional independence properties of the graph to be represented in the factorization.



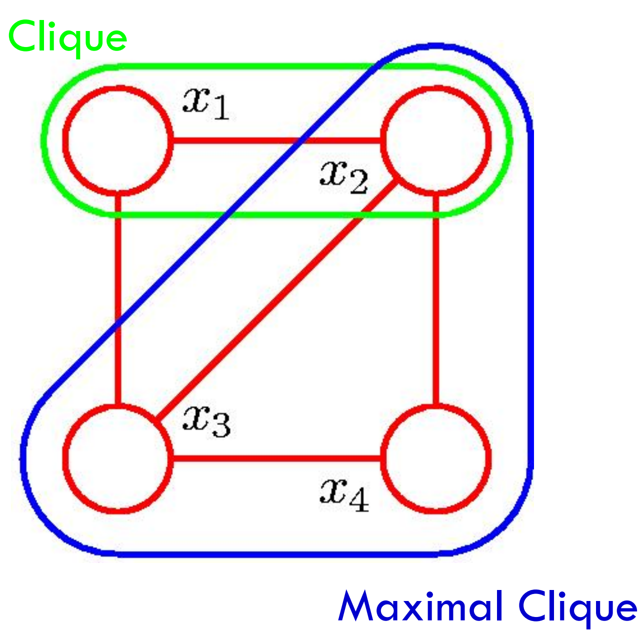
$$A \perp\!\!\!\perp B \mid C$$

Markov Blanket



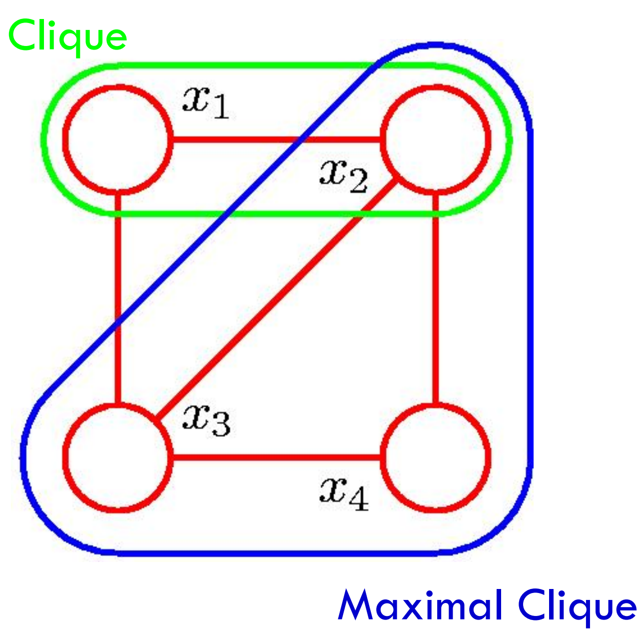
Cliques

- Thus two nodes should be in the same factor if and only if they are directly connected.
- This means that factors must consist of fully connected sets of nodes.
- Such fully-connected sets of nodes are called **cliques**.
- A clique that cannot be enlarged is called a **maximal clique**.



Cliques

- Thus each factor is a function of a clique.
- In fact, we can restrict factors to being functions of **maximal** cliques, since smaller cliques must be subsets of maximal cliques.



END OF LECTURE NOV 24, 2010

Potential Functions

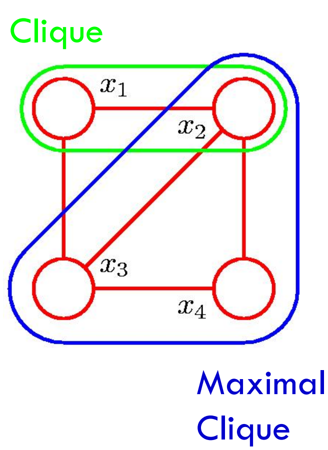
Let C denote a maximal clique, and $\mathbf{x}_C$ the variables in that clique.

Then the joint distribution is written as a product of **potential functions** $\psi_C(\mathbf{x}_C)$ over these maximal cliques:

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

where the normalizing constant Z (aka the **partition function**) is given by

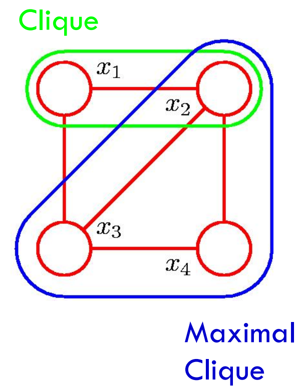
$$Z = \sum_{\mathbf{x}}\prod_C \psi_C(\mathbf{x}_C)$$



Potential Functions

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

- Since Z is a function of any parameters of ψ, it is needed in order to learn these parameters.
- Unfortunately calculation of Z is usually not feasible.
- For example, if $\mathbf{x}$ consists of M discrete variables x_i, each with K states, there are K^M possible configurations of $\mathbf{x}$, and hence K^M terms in Z.



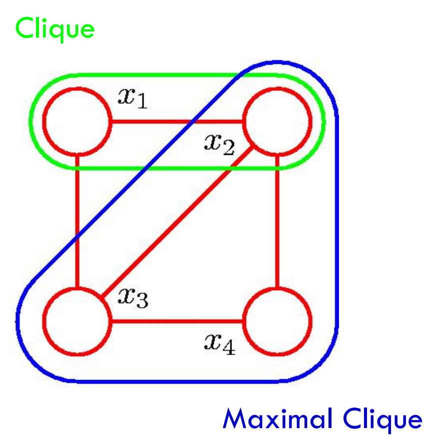
Potential Functions

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

- Evaluation of local conditional probabilities is feasible, since the partition function cancels out.
- To evaluate local marginals we can work with the unnormalized distributions, and then normalize the marginals at the end.

e.g., if $f(x_1) \propto p(x_1)$,

$$p(x_1 = a) = \frac{f(x_1 = a)}{\sum_{x_1} f(x_1)}$$



Boltzmann & Gibbs Distributions

If we restrict the potential functions $\psi_C(\mathbf{x}_C)$ to be strictly positive we can represent them as exponentials of energy functions $E(\mathbf{x}_C)$:

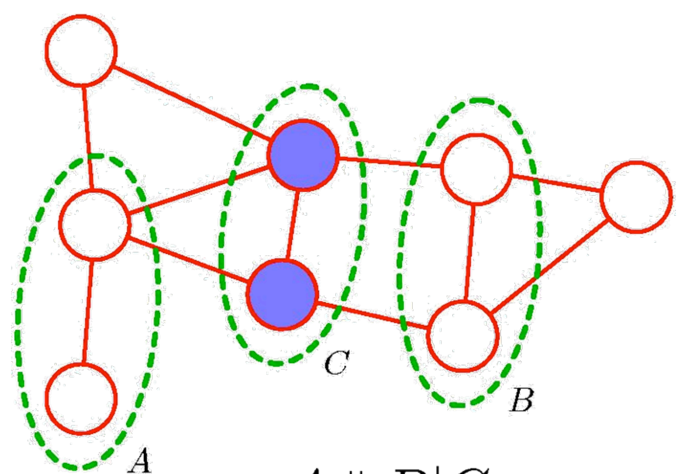
$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

Then $p(\mathbf{x}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{x}_C)$ is known as a **Boltzmann**, or **Gibbs** distribution.

A set of random variables $\mathbf{x}$ whose joint distribution is a Gibbs distribution is called a **Gibbs random field (GRF)**.

Hammersley-Clifford Theorem

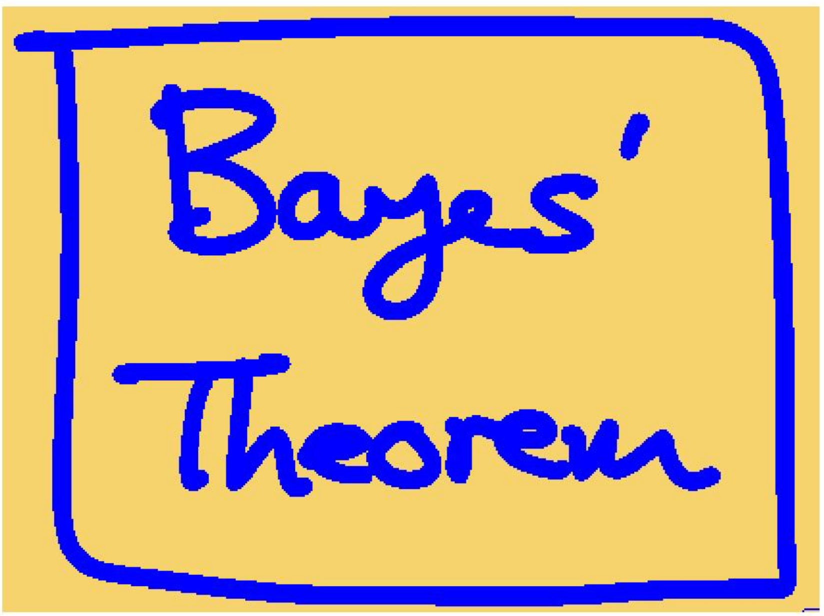
- An MRF is defined by a set of local conditional independence relationships.
- A GRF is defined by a joint distribution that factors into local exponential clique potentials.
- The **Hammersley-Clifford Theorem** establishes that any MRF defined over an undirected graph is also a GRF defined over the maximal cliques of that graph.
- This is of great importance, as it relates the **local** Markov properties of the system to the **global** probability of configurations.



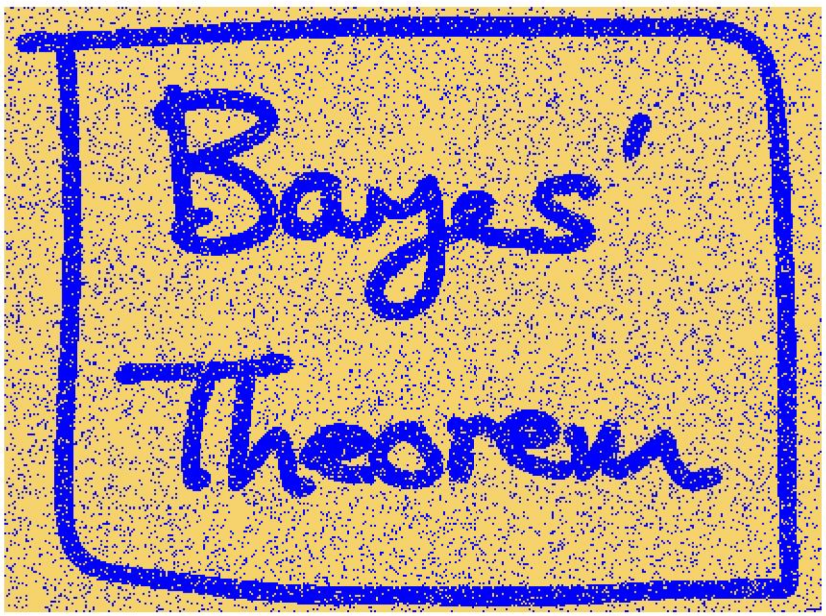
$$A \perp\!\!\!\perp B \mid C$$

MRF Example

Illustration: Image De-Noising

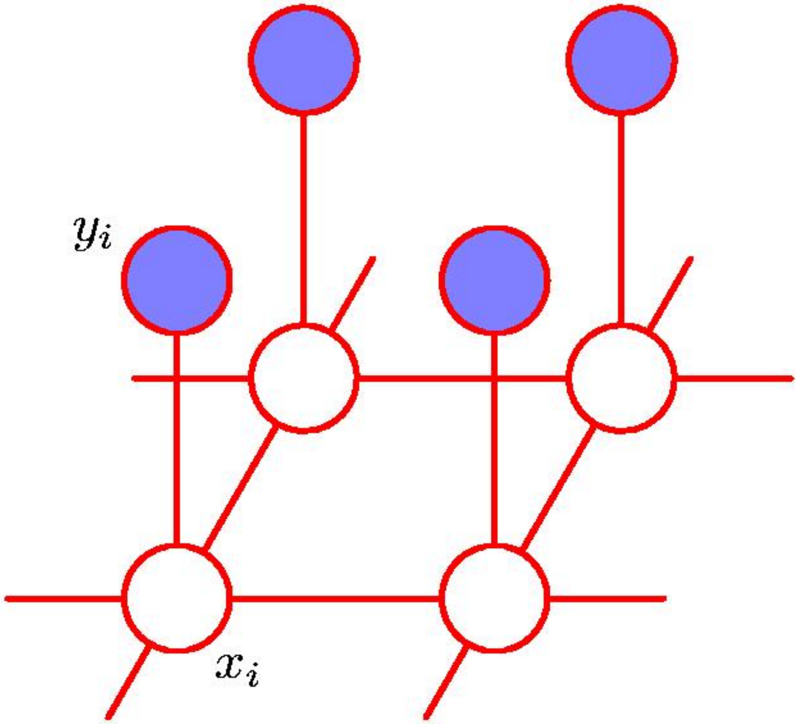


Original Image



Noisy Image

Illustration: Image De-Noising



Ising Model:

Binary image: $x_i, y_i \in \{-1, +1\}$

Bias Smoothness

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

Fidelity

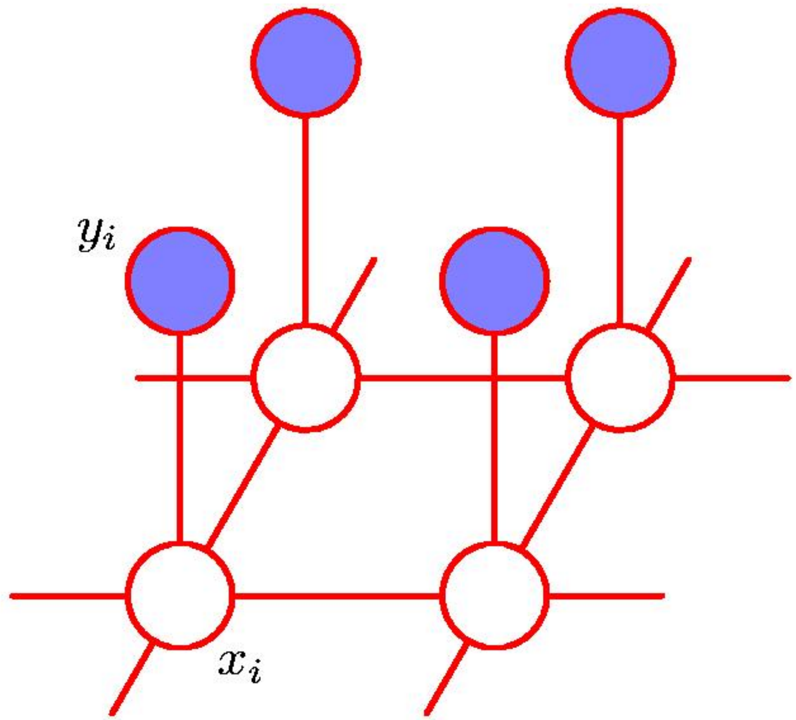
$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}$$

Inference

- Suppose we know the parameters h, β , η .
- How do we estimate the x that maximizes

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

$$E(\mathbf{x},\mathbf{y}) = \overbrace{h\sum_i x_i}^{\text{Bias}} - \overbrace{\beta\sum_{\{i,j\}} x_i x_j}^{\text{Smoothness}} \underbrace{-\eta\sum_i x_i y_i}_{\text{Fidelity}}$$



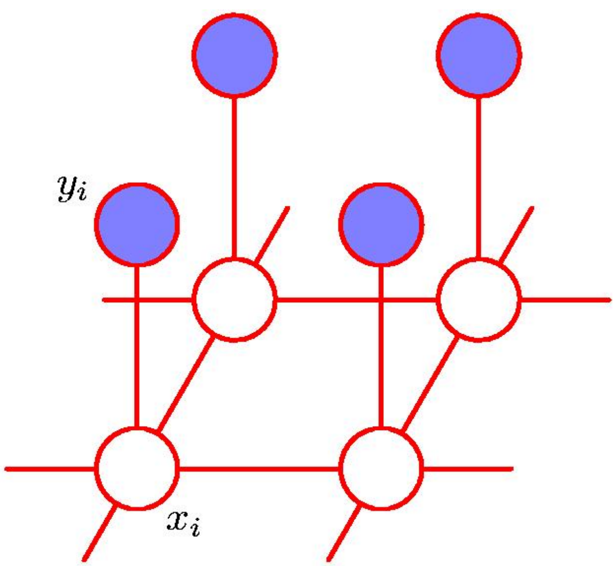
Inference Algorithm: ICM

- Iterated conditional modes (ICM) is a simple coordinate descent method for finding a local maximum of $p(\mathbf{x}|\mathbf{y})$.
- We simply select nodes xi in sequence (randomly or systematically), and flip their state if it lowers the energy.
- The algorithm halts when no local state change can lower the energy. This is a local maximum of p(x,y).

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

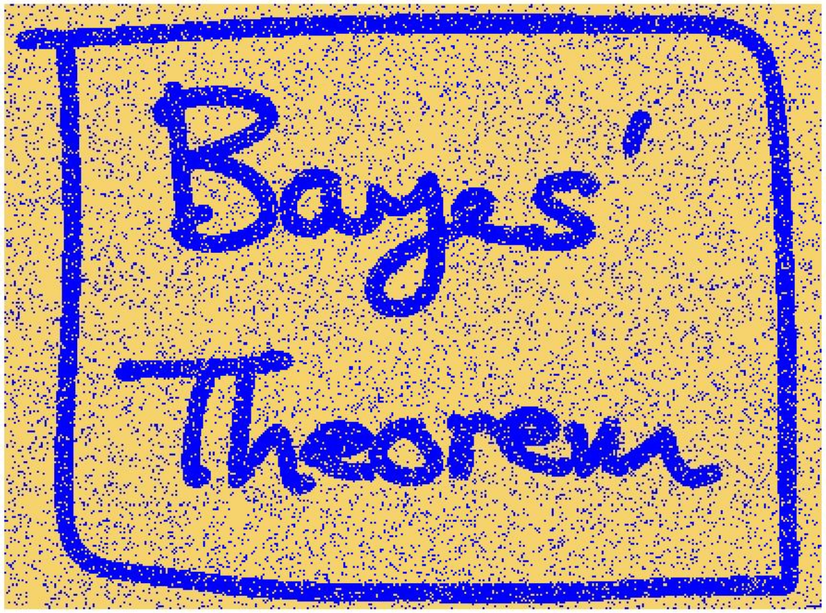
$$E(\mathbf{x},\mathbf{y}) = \underbrace{h\sum_i x_i}_{\text{Bias}} - \underbrace{\beta\sum_{\{i,j\}} x_i x_j}_{\text{Smoothness}} \underbrace{-\eta\sum_i x_i y_i}_{\text{Fidelity}}$$

Observed noisy pixels

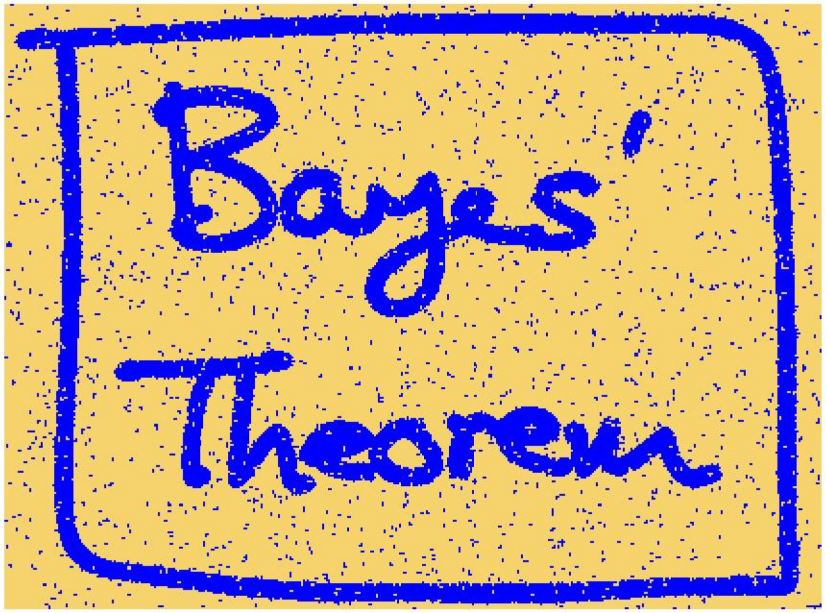


Unobserved original pixels

Illustration: Image De-Noising



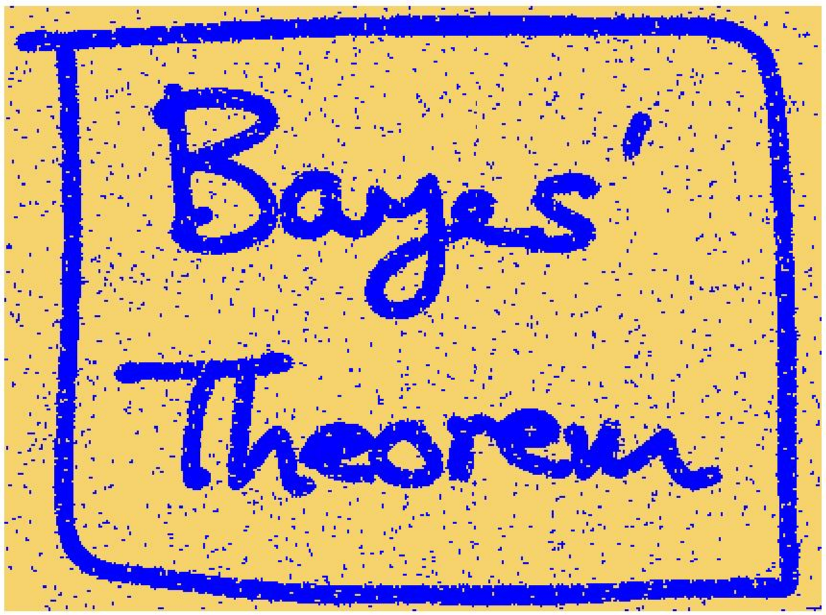
Noisy Image



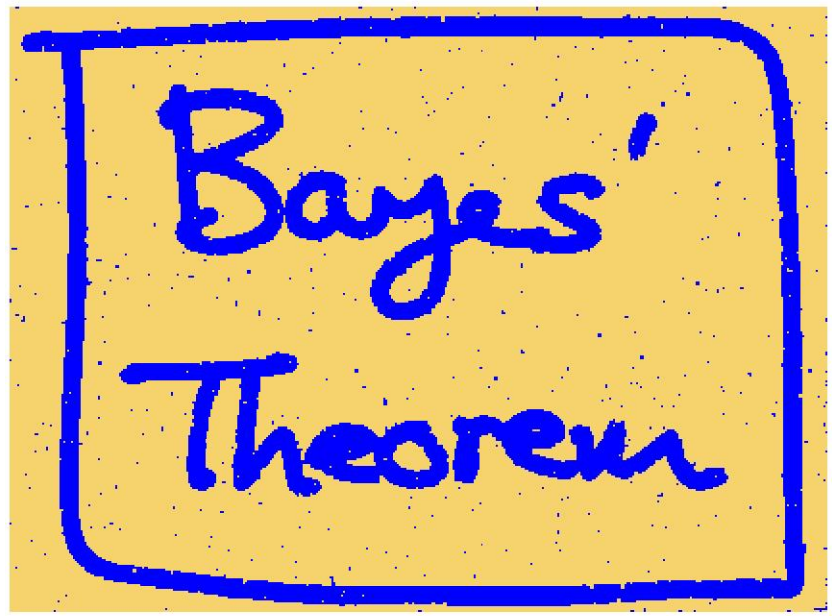
Restored Image (ICM)

Illustration: Image De-Noising

- ICM will only find a local maximum.
- In fact, for this problem, the global maximum can be found using graph cuts.



Restored Image (ICM)



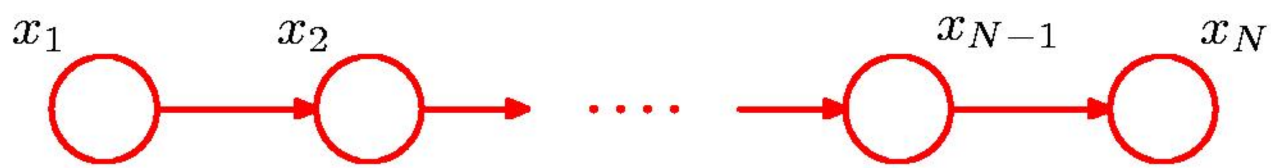
Restored Image (Graph cuts)

Relating Directed Graphs to MRFs

Converting Directed to Undirected Graphs

- Directed graphs can always be converted to undirected graphs.
- This is used for some inference techniques, e.g., the junction tree algorithm.
- However, some independence properties may no longer be represented after conversion.

Converting Directed to Undirected Graphs

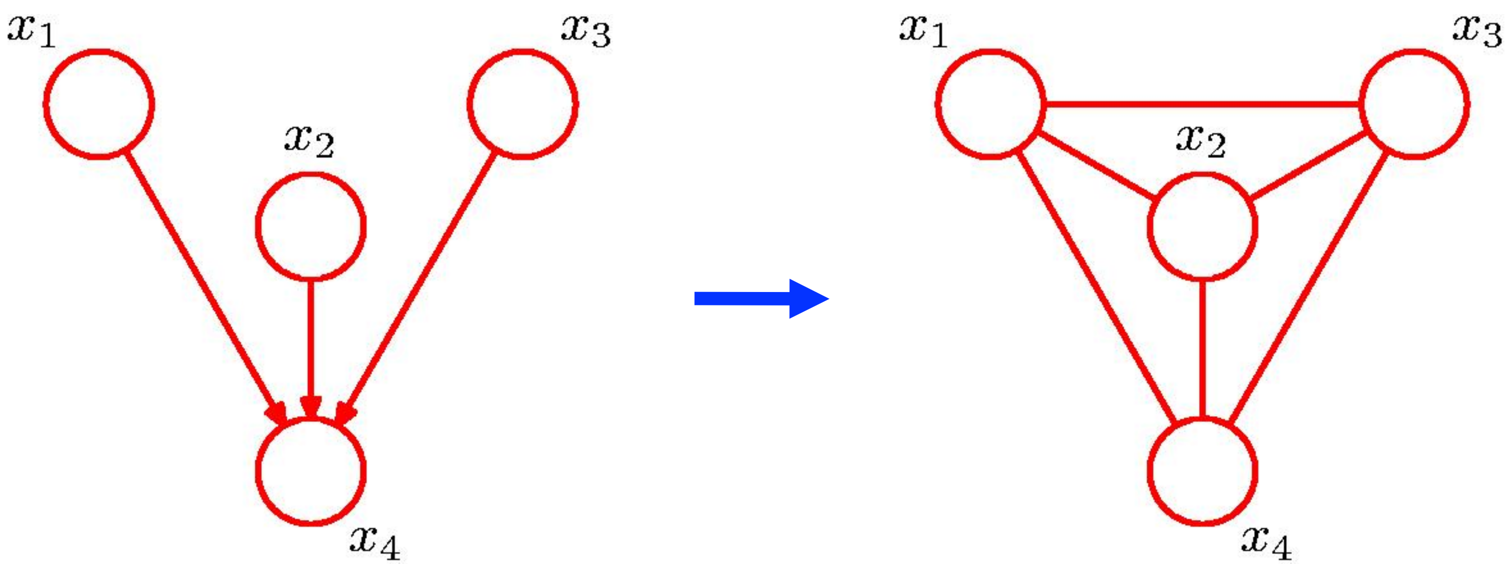


$$p(\mathbf{x}) = \underbrace{p(x_1)p(x_2|x_1)}\, p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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

Converting Directed to Undirected Graphs

- Additional links are required between co-parents



$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)$$

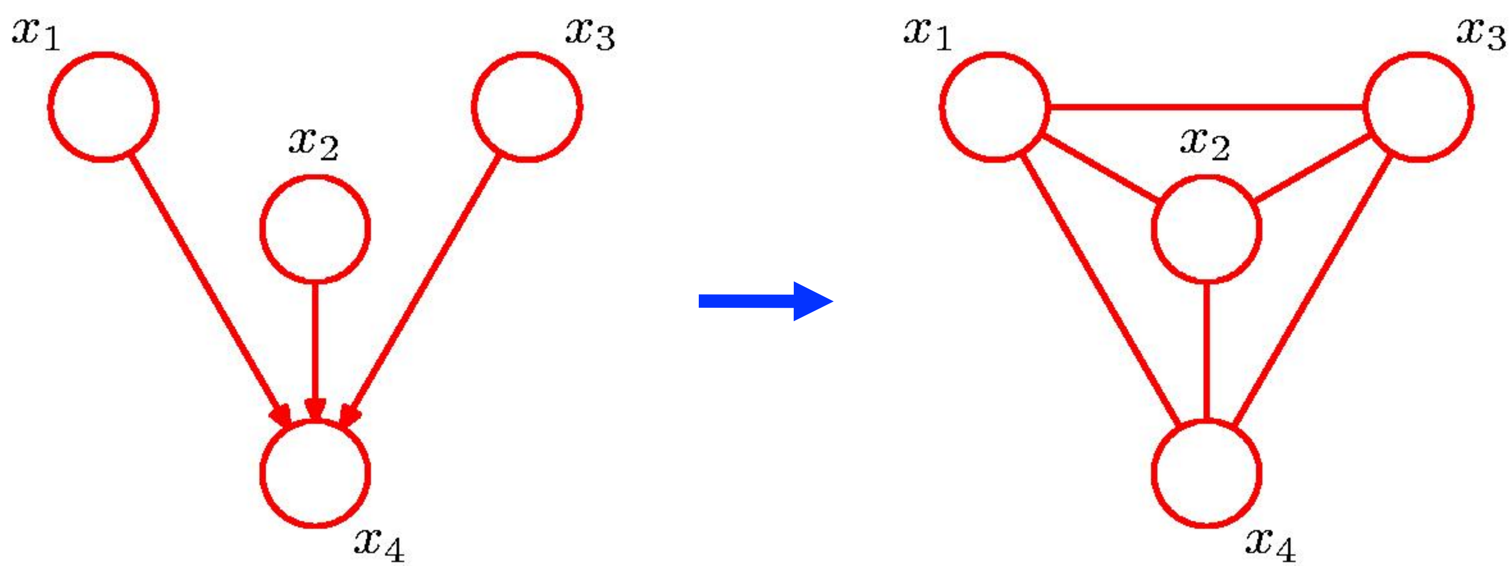
$$= \frac{1}{Z}\psi\left(x_1, x_2, x_3, x_4\right)$$

Converting Directed to Undirected Graphs

- Thus the general procedure is:
  - Add additional undirected links between all pairs of co-parents
  - Drop the arrows
  - Initialize the potentials to 1
  - Multiply the conditional factors into each corresponding potential
- Note that converting from undirected to directed is much less common, and more difficult.

Converting Directed to Undirected Graphs

- In this case, the independence properties represented in the original directed graph are lost after conversion.

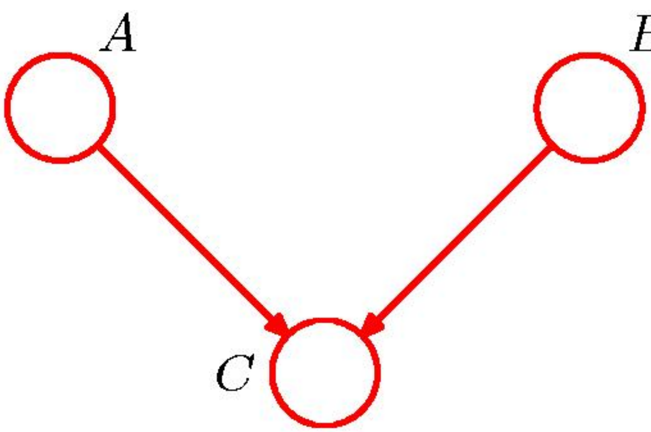


$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)$$

$$= \frac{1}{Z}\psi\left(x_1,x_2,x_3,x_4\right)$$

Directed vs. Undirected Graphs (2)

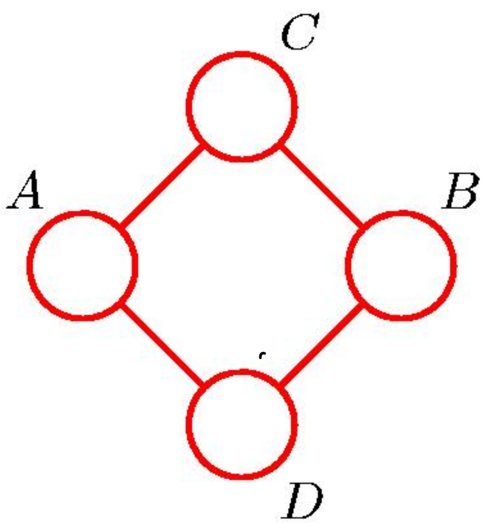
No **undirected** graph can represent these conditional independence properties.



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

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

No **directed** graph can represent these conditional independence properties.



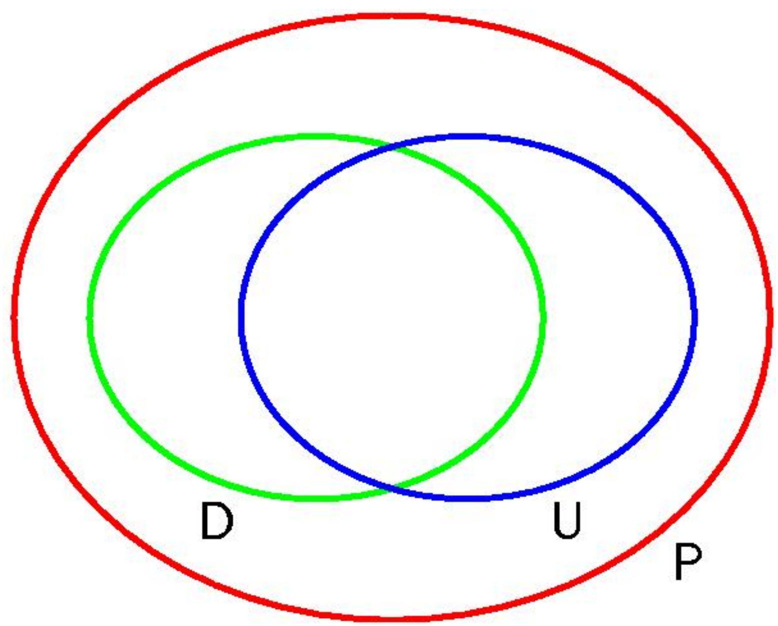
$$A \not\!\perp\!\!\!\perp B \mid \emptyset$$

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

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

Directed vs. Undirected Graphs (1)

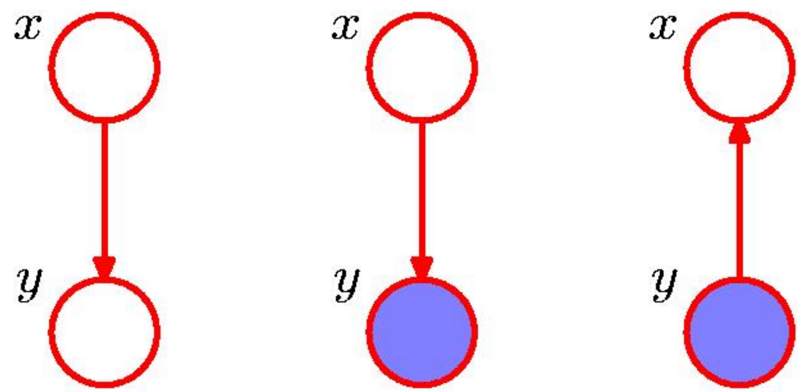
- P = set of all distributions over a set of variables **x**.
- D = set of all distributions whose conditional independence properties can be represented by a directed graph
- U = set of all distributions whose conditional independence properties can be represented by an undirected graph



PART 3
INFERENCE IN GRAPHICAL MODELS

Inference in Graphical Models

- In inference, we clamp some of the variables to observed values, and then compute the posterior over other, unobserved variables.
- Simple example:



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

Inference on a Chain

- Let's assume each variable is discrete, having *K* states.

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

- Computing marginal for one variable requires integrating out N-1 variables.

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

- If done naively, this summation will have K^{N-1} terms.

Inference on a Chain

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

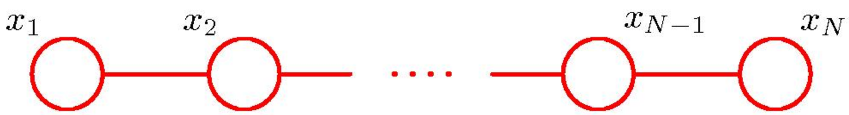
$$p(x_n) = \sum_{x_1}\cdots\sum_{x_{n-1}}\sum_{x_{n+1}}\cdots\sum_{x_N} p(\mathbf{x})$$

- This can be made much more efficient by exploiting the modularity of the joint probability.
- For example, note that:

$$\sum_{x_1,x_2,x_3} \psi(x_1,x_2)\psi(x_2,x_3) = \sum_{x_3,x_2}\left(\psi(x_2,x_3)\sum_{x_1}\psi(x_1,x_2)\right)$$

If all variables have K states, this reduces the number of arithmetical operations from K^3 additions and K^3 multiplications to $2K^2 + K$ additions and K^2 multiplications.

Inference on a Chain



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

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

- This principle can be applied recursively to the left and to the right of x_n:

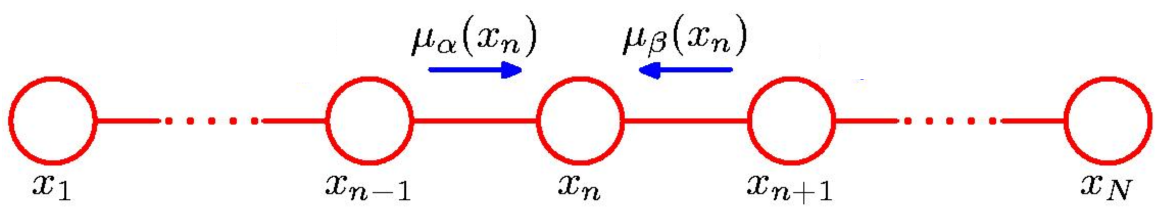
$$p(x_n) = \frac{1}{Z}\underbrace{\left[\sum_{x_{n-1}}\psi_{n-1,n}(x_{n-1}, x_n)\cdots\left[\sum_{x_1}\psi_{1,2}(x_1, x_2)\right]\cdots\right]}_{\mu_\alpha(x_n)}$$

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

This results in a reduction in the number of operations from $(N-1)K^N$ multiplications and K^{N-1} additions to $(N-3)K^2+K$ multiplications and $(N-1)K^2$ additions.

Inference on a Chain

- These two factors can be viewed as vector messages passed to x_n from the left and right portions of the network:

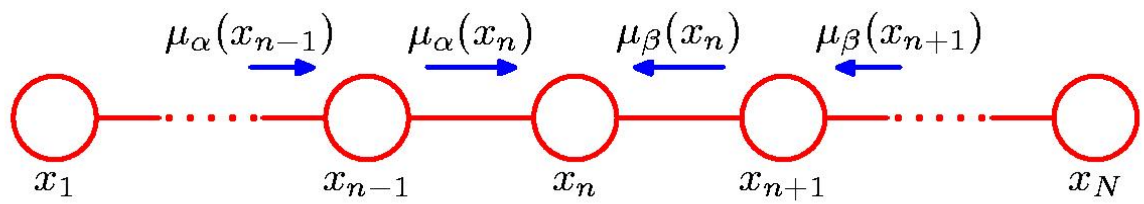


$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

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

Inference on a Chain

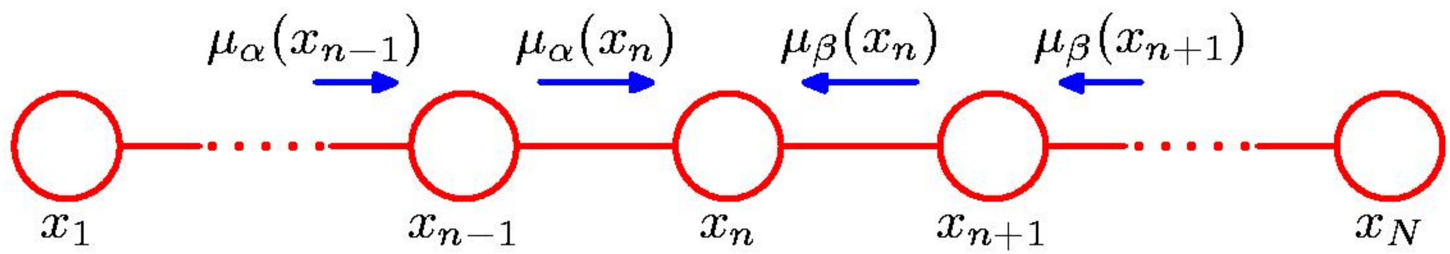
- These two messages can each in turn be broken down as the product of a matrix potential and a vector message:



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

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

Inference on a Chain



- Initial conditions:

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

- Normalization:

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

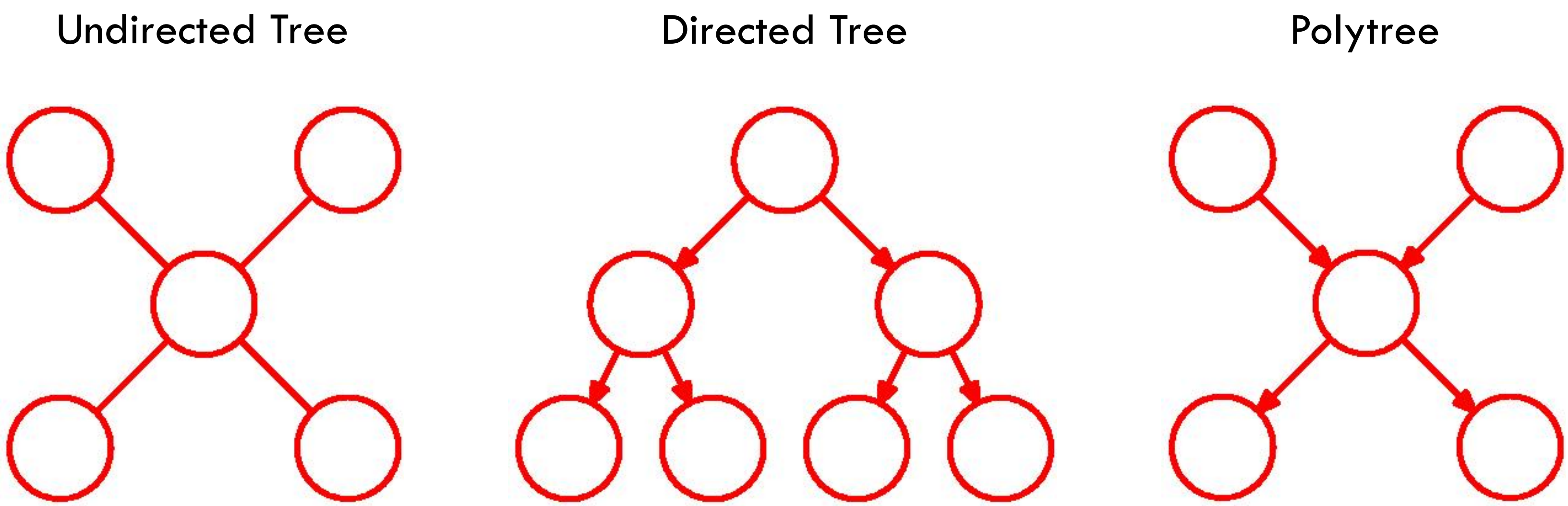
Inference on a Chain

- 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 for all variables required:

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

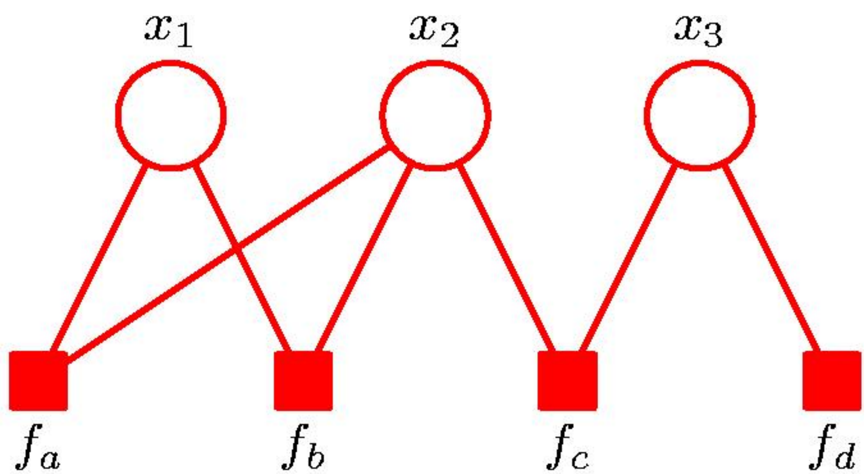
Trees

- Message passing can also be used to do efficient exact inference over trees.



Factor Graphs

- Factor graphs allow the conditional independence structure of both undirected and directed graphs to be represented explicitly in a common framework.

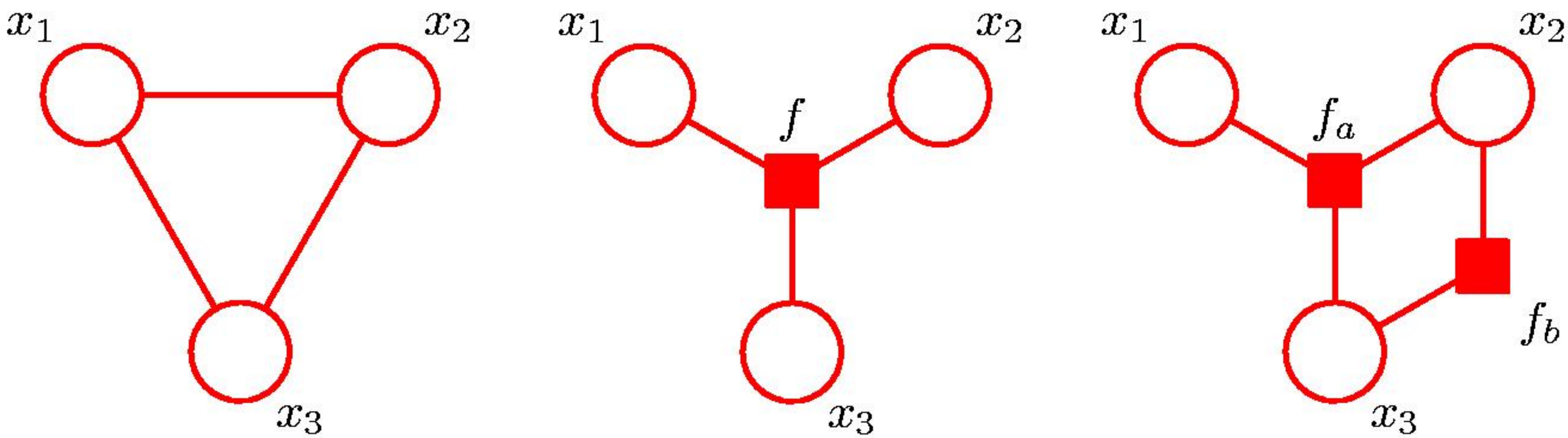


$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$ where f_s is a factor over a subset of variables $\mathbf{x}_s$.

Factor Graphs from Undirected Graphs

- Factor graphs can potentially communicate more detailed information about about the underlying factorization.



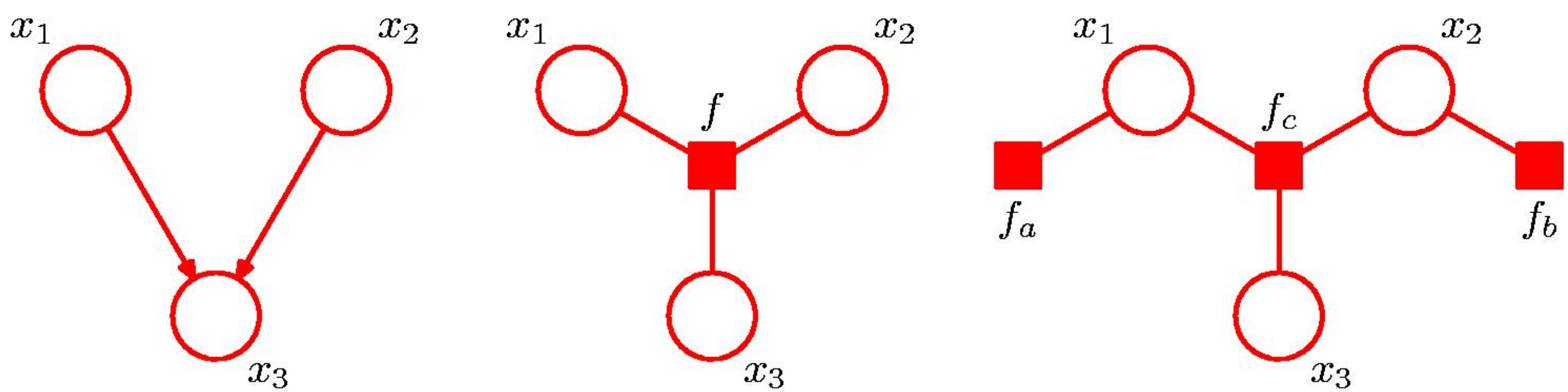
$$\psi(x_1, x_2, x_3)$$

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

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

END OF LECTURE NOV 29, 2010

Factor Graphs from Directed Graphs



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

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

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

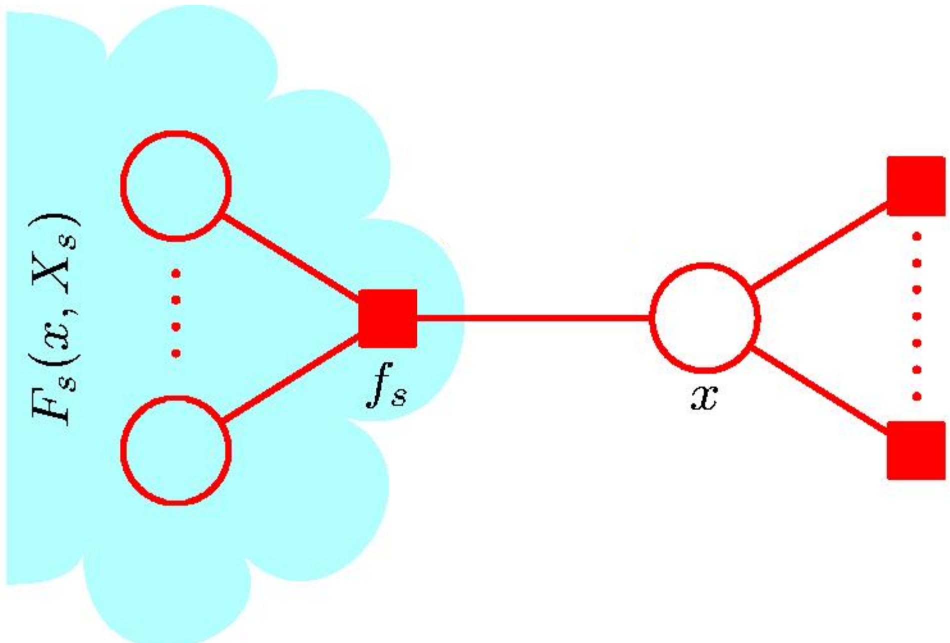
The Sum-Product Algorithm

The Sum-Product Algorithm (1)

- Objective:
  i. to obtain an efficient, exact inference algorithm for finding marginals in acyclic graphs;
  ii. in situations where several marginals are required, to allow computations to be shared efficiently.
- Key idea: Distributive Law of multiplication over addition

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

The Sum-Product Algorithm (2)

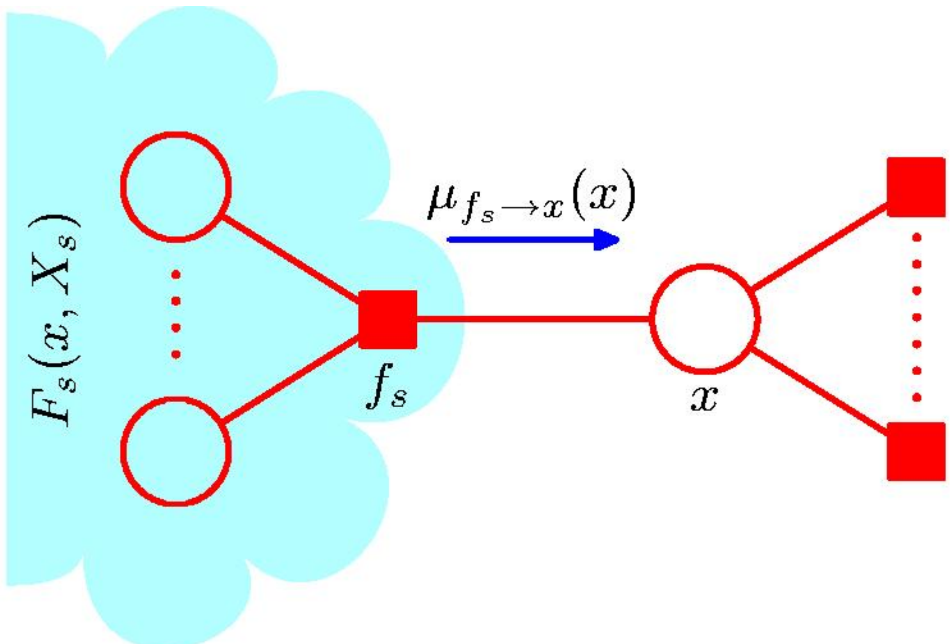


$$p(x) = \sum_{\mathbf{x}\backslash x} p(\mathbf{x})$$

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

where X_s is the set of all variables in the subtree connected to x via f_s.

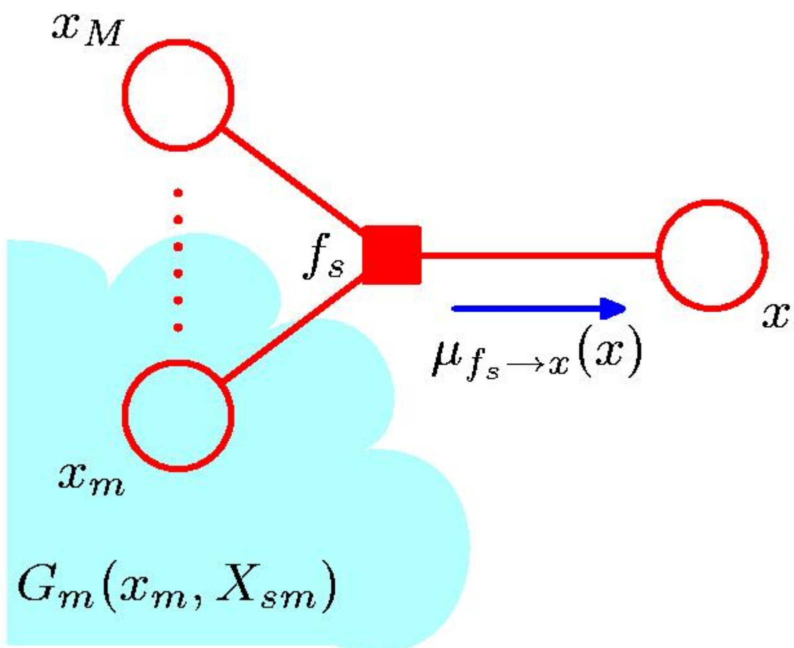
The Sum-Product Algorithm (3)



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

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

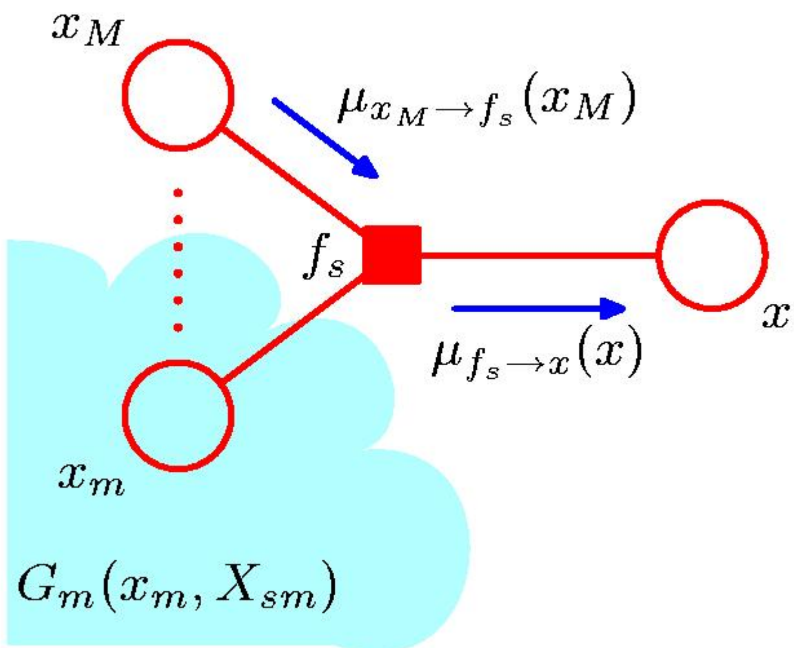
The Sum-Product Algorithm (4)



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

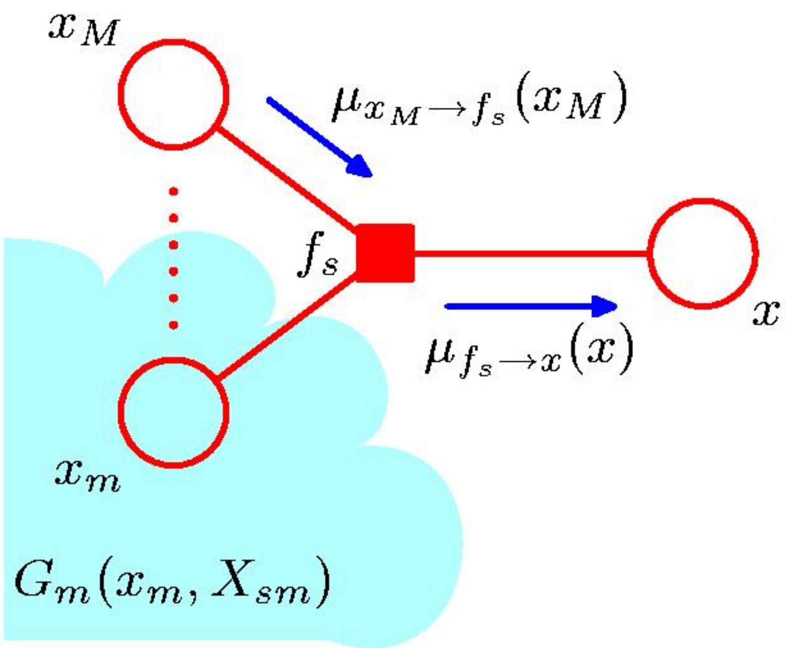
where X_{si} is the set of all variables in the subtree connected to f_s via x_i.

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)

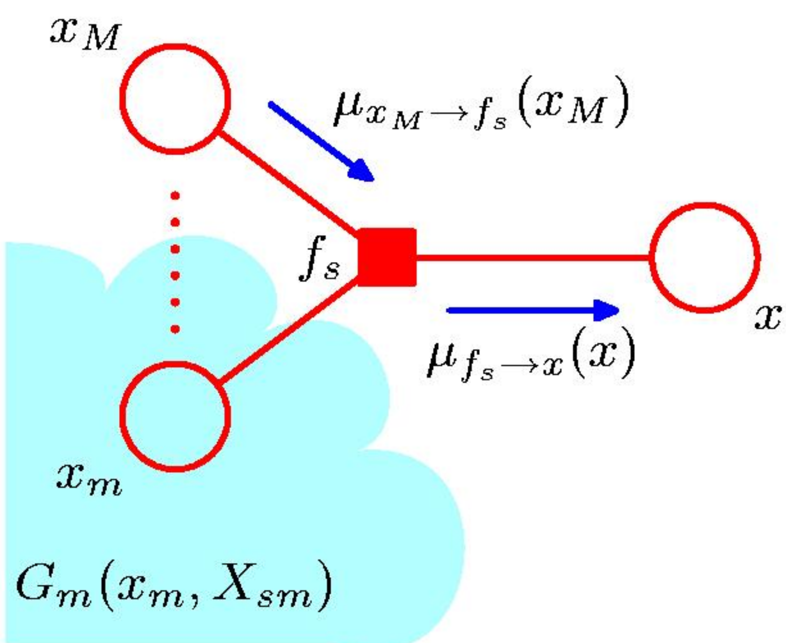


$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

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

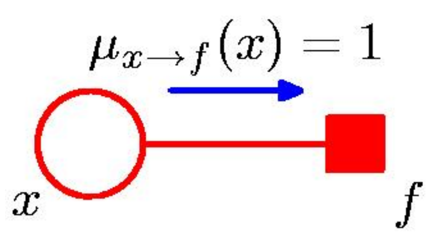
The Sum-Product Algorithm

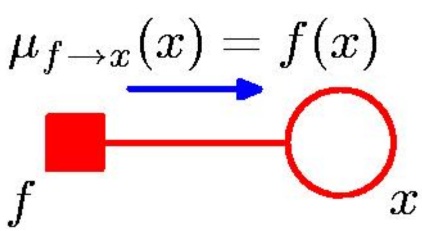
- Thus the marginal at x is given by the product of messages arriving at that node.
- Each message is computed recursively in terms of other messages.



The Sum-Product Algorithm (7)

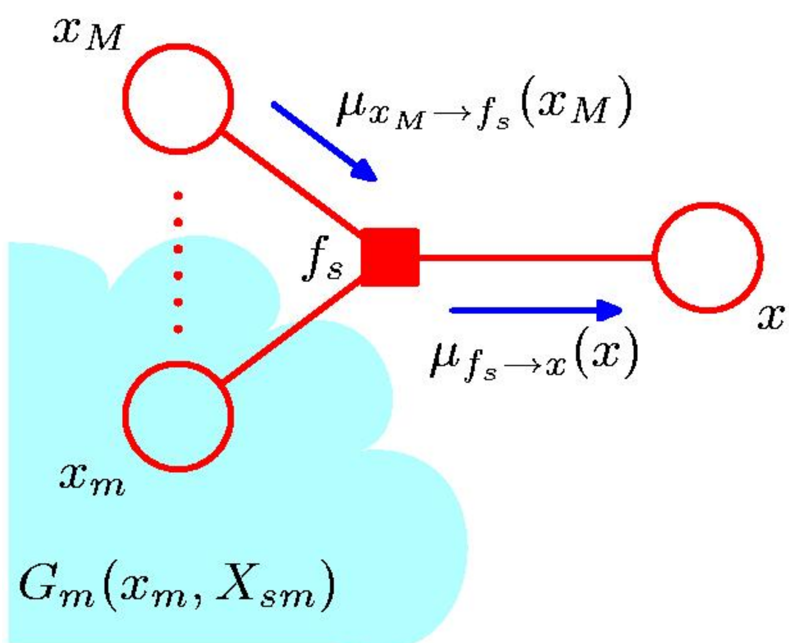
- Initialization
  - View x as the root of the tree
  - Begin at leaf nodes
    - Variable leaf nodes have a single factor node as parent
    - Factor leaf nodes have a single variable node as parent





The Sum-Product Algorithm

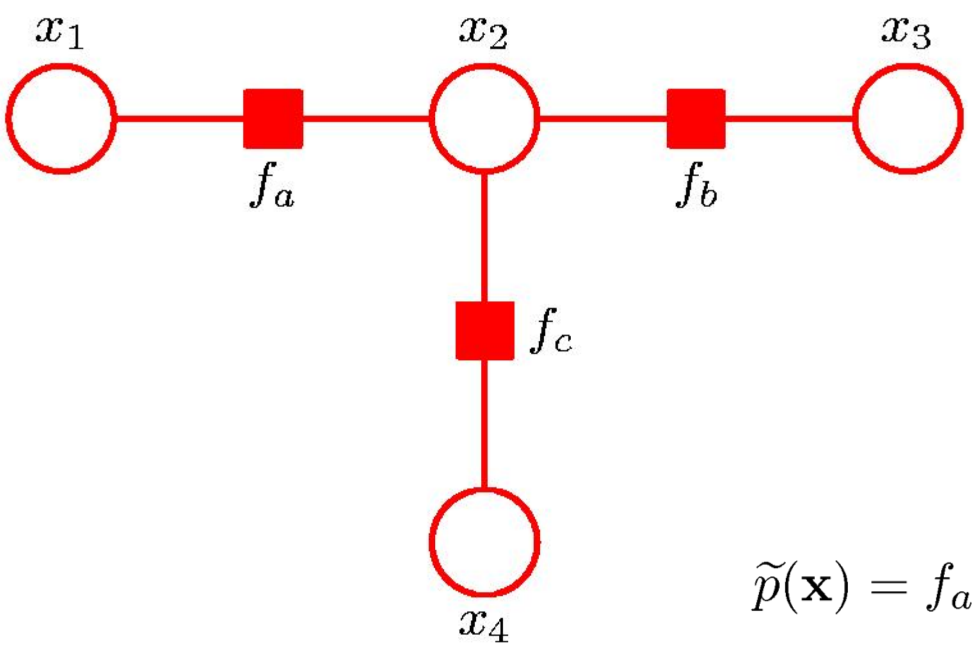
- Marginals for all variable nodes could be computed by simply repeating this process.
- But this is wasteful, as many of the required computations are shared.



The Sum-Product Algorithm (8)

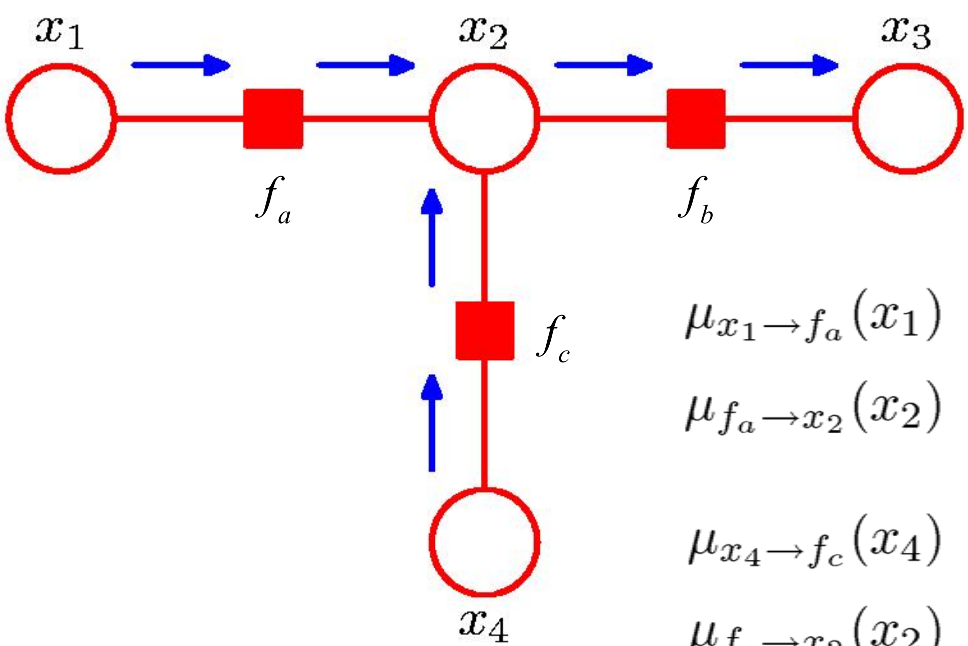
- To compute all local marginals at once:
  1. Pick an arbitrary node as root
  2. Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
  3. Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
  4. 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)



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

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

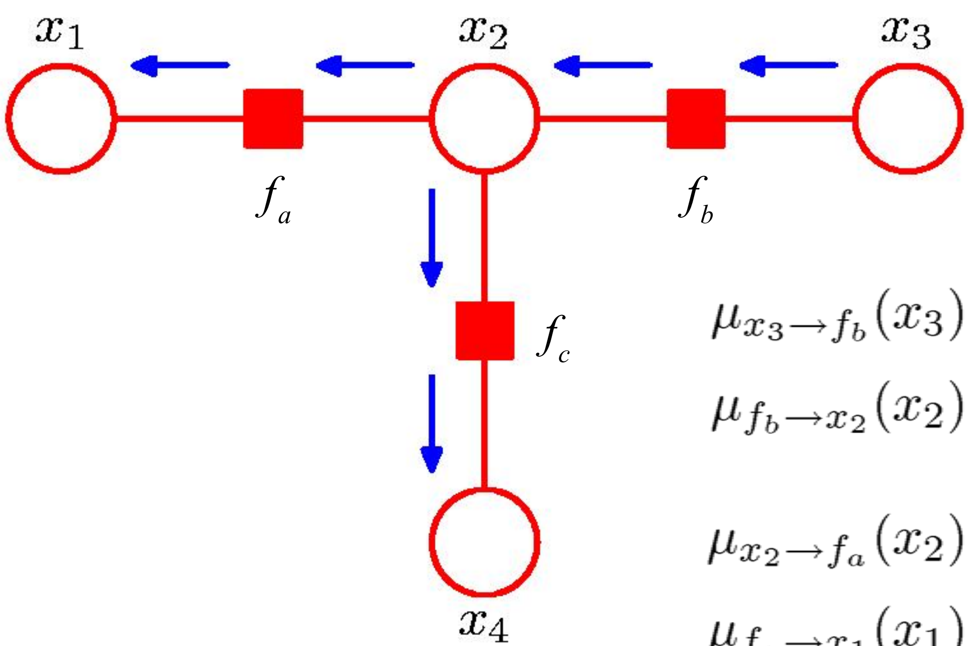
$$\mu_{x_4 \to f_c}(x_4) = 1$$

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

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

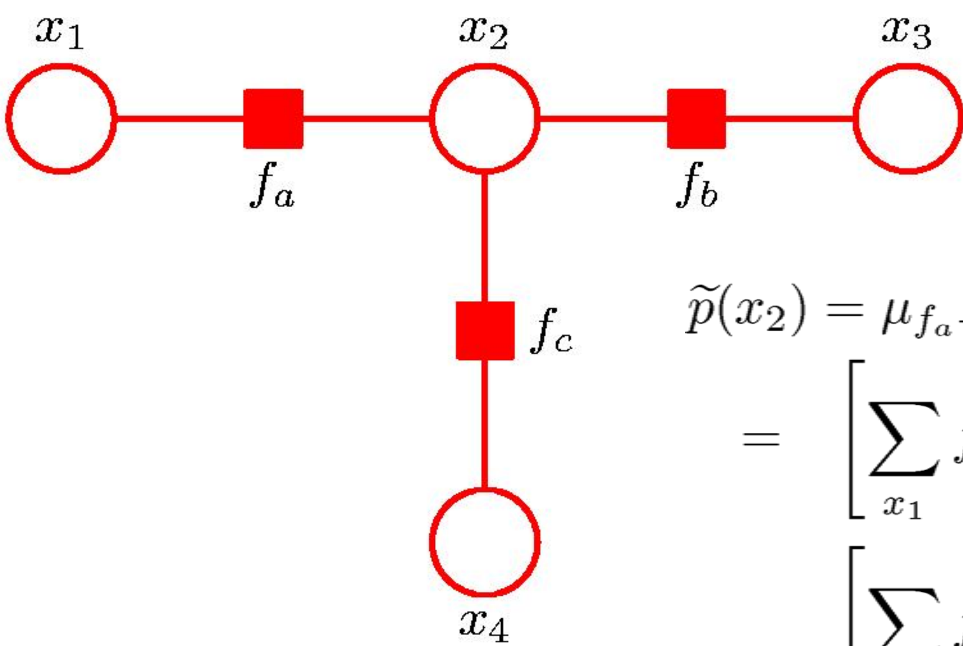
$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}(x_2)$$

Sum-Product: Example (3)



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

Sum-Product: Example (4)



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

The Max-Sum Algorithm

The Max-Sum Algorithm (1)

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

- In general, maximum marginals ≠ 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)
- To calculate max *p(x)*:

$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \dots \max_{x_N} p(\mathbf{x})$$

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

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

END OF LECTURE
DEC 1, 2010

The Max-Sum Algorithm (3)

- Generalizes to tree-structured factor graph
- Designate one node (x_n) as the root
- Starting at leaf nodes, propagate messages up to root.
- Final max probability is calculated by taking max over product of all incoming messages at root x_n:

$$\max_{x} p(x) = \max_{x_n} \prod_{f_s \in \mathrm{ne}(x_n)} \mu_{f_s \to x_n}(x_n)$$

The Max-Sum Algorithm (4)

- Max-Product → 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 (at root node *x*)

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

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

The Max-Sum Algorithm

- To determine the state of the other variables, we backtrack from the root node, using the state table ϕ:

Consider a factor node $f(x_s)$, $x_s = \{x, x_1, \ldots x_M\}$.

If node x_i is connected to the root through node x via $f(x_s)$, then the state table ϕ stores

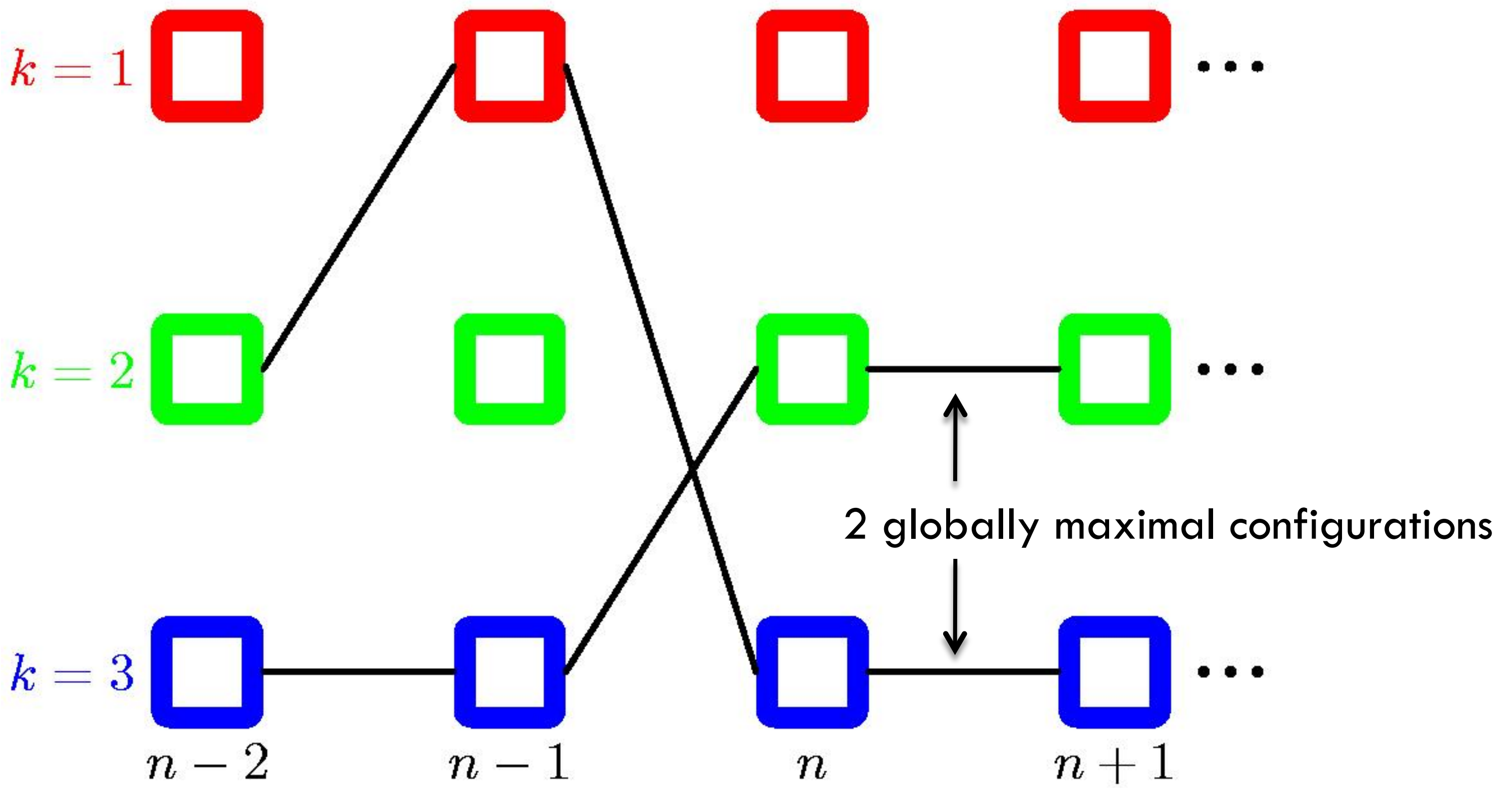
$$\phi(x) = \arg\max_{x_s \setminus x} \left(\log f(x_s) + \sum_{m \in x_s \setminus x} \mu_{x_m \to f}(x_m) \right)$$

So to recover the maximal configuration, we unwind from the root, using

$$x_i^{\max} = \phi_i(x^{\max})$$

The Max-Sum Algorithm (7)

- Example: Markov chain



Loopy Belief Propagation

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