Graphical Models

Jun Kuang, Renyu Zhu

April 2, 2019

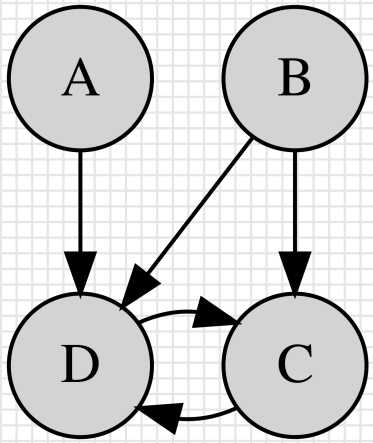
Outline

- Introduction
- Bayesian Networks
- Conditional Independence
- Markov Random Fields
- Inference in Graphic Models

1. Introduction

Concept

A graphical model or probabilistic graphical model or structured probabilistic model is a probabilistic model that use graphs to describe which random variables (*nodes* or vertices) in the probability distribution interact (links or edges or arcs) with each other directly.



Types

- Bayesian network or directed graphical model.
- Markov random fields or undirected graphical model.
- Factor graph.
- Other types: chain graph, ancestral graph, clique tree, etc.

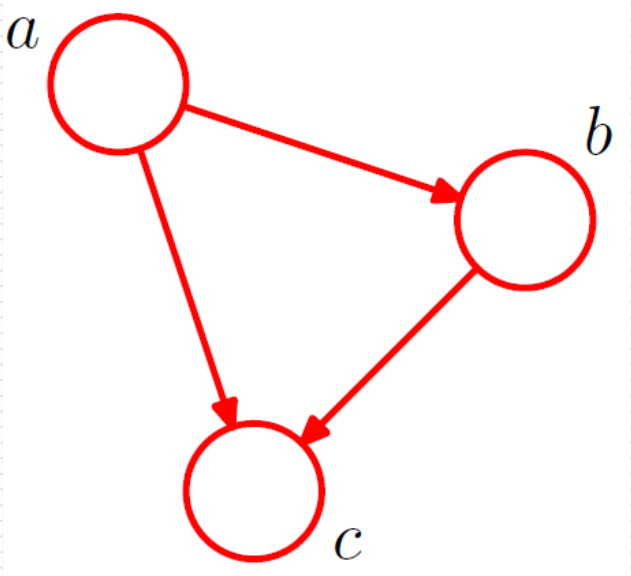
Useful Properties

- Visualizing the structure of a probabilistic model and design new models.
- Insights into the properties of the model, including conditional independence properties.
- Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations.

2. Bayesian Networks

A Simple BN

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



Extension

K variables:

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

General terms:

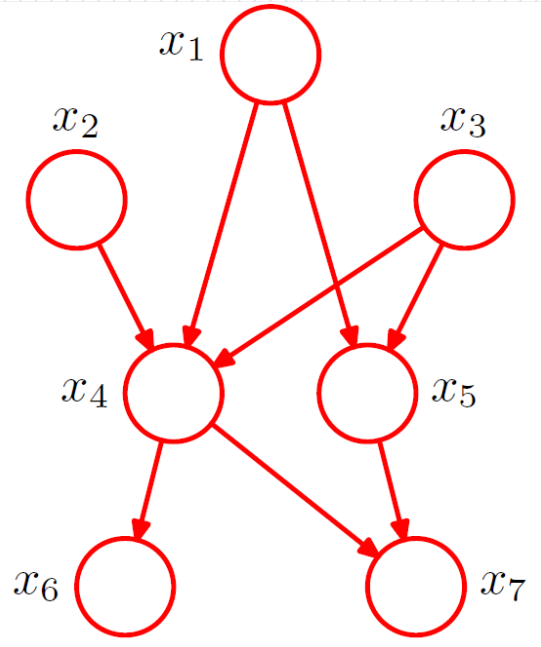
$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|pa_k),$$

where pa_k denotes the set of parents of x_k, and $\mathbf{x} = \{x_1, \cdots, x_K\}$. (Factorization)

"Absence" Matters

A joint distribution for 7 variables BN:

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2)p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5).$$

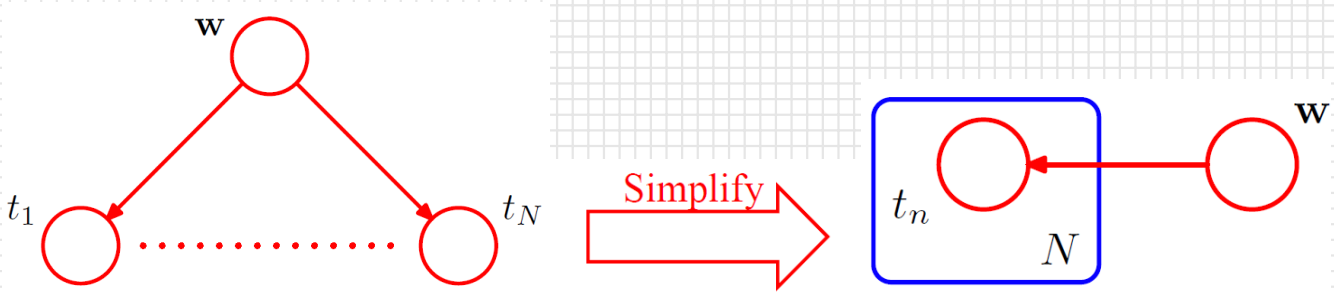


Example: Polynomial Regression (1)

Joint distribution:

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

where $\mathbf{w}$ are polynomial coefficients and $\mathbf{t} = (t_1, \cdots, t_N)^T$ is observed data.

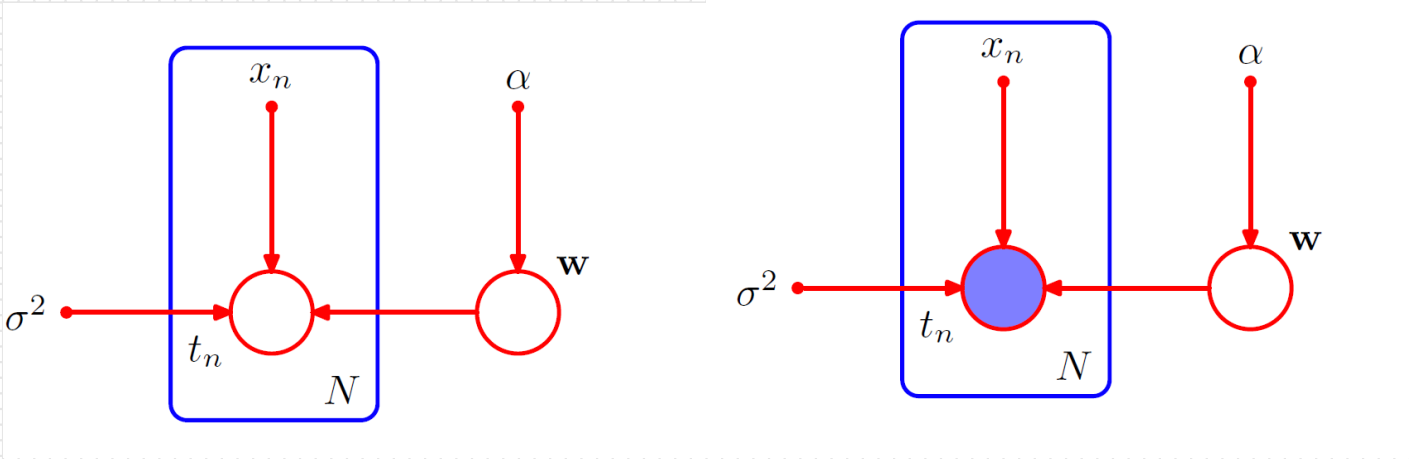


Example: Polynomial Regression (2)

Add parameters and variables:

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

where $\mathbf{x} = (x_1, x_2, \cdots, x_N)^T$ is input data, σ^2 is noise variance and α is hyperparameter.

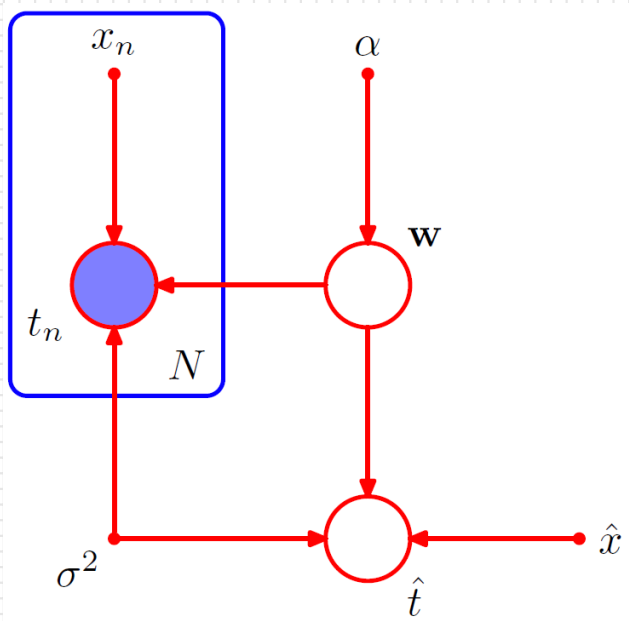


Example: Polynomial Regression (3)

Given a new input value $\hat{x}$:

$$p(\hat{t}, \mathbf{t}, \mathbf{w}|\hat{x}, x, \alpha, \sigma^2) = p(\mathbf{t}, \mathbf{w}|\mathbf{x}, \alpha, \sigma^2) p(\hat{t}|\hat{x}, \mathbf{w}, \sigma^2)$$

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



Example: Polynomial Regression (4)

Predictive distribution for $\hat{t}$:

$$
\begin{aligned}
p(\hat{t}|\hat{x}, \mathbf{x}, \mathbf{t}, \alpha, \sigma^2) &= \frac{p(\hat{t}, \mathbf{t}|\hat{x}, \mathbf{x}, \alpha, \sigma^2)}{p(\mathbf{t}|\hat{x}, \mathbf{x}, \alpha, \sigma^2)} \\
&\propto p(\hat{t}, \mathbf{t}|\hat{x}, \mathbf{x}, \alpha, \sigma^2) \\
&\propto \int p(\hat{t}, \mathbf{t}, \mathbf{w}|\hat{x}, \mathbf{x}, \alpha, \sigma^2) d\mathbf{w}
\end{aligned}
$$

Generative Models

Ancestral sampling:

1. Draw a sample from the distribution $p(x_1)$;
2. Draw a sample from the conditional distribution $p(x_n|pa_n)$.

Graphical models captures the causal process by which the observed data was generated.

Discrete Variables (1)

A single discrete variable $\mathbf{x}$ having K possible states:

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k},$$

where $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_K)^T$ are parameters, and $\sum_k \mu_k = 1$. Only $K - 1$ values for μ_k is needed.

Discrete Variables (2)

Two discrete variables:

$$p(\mathbf{x_1}, \mathbf{x_2}|\boldsymbol{\mu}) = \prod_{k=1}^{K}\prod_{l=1}^{K} \mu_{kl}^{x_{1k}x_{2l}},$$

governed by $K^2 - 1$ parameters.

M discrete variables:

$$p(\mathbf{x_1}, \cdots, \mathbf{x_M}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \cdots \prod_{l=1}^{K} \mu_{k\ldots l}^{x_{1k}\cdots x_{Ml}},$$

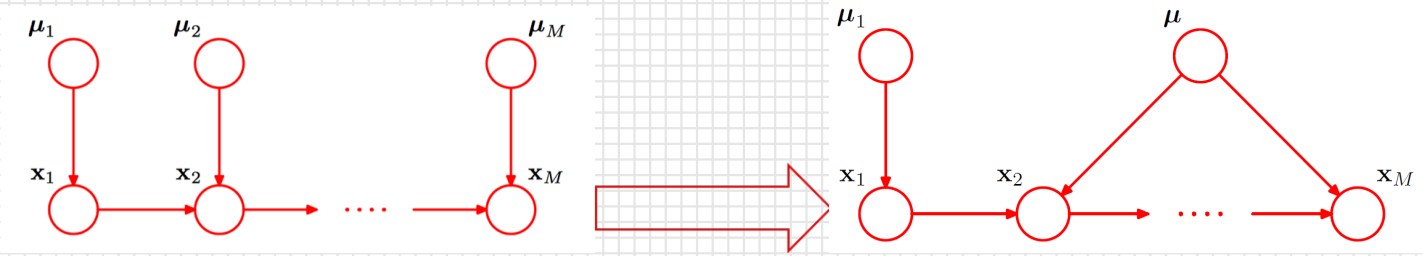
governed by $K^M - 1$ parameters.

Discrete Variables (3)

We can use graph to:

- drop links: $K-1+(M-1)K(K-1)$

- share parameters: K^2-1



Linear-Gaussian Models (1)

A DAG for D Gaussian variables x_i:

$$p(x_i|pa_i) = N(x_i| \sum_{j \in pa_i} w_{ij} x_j + b_i, v_i),$$

where w_{ij} and b_i are parameters, v_i is variance, pa_i are parent nodes of x_i.

The joint probability distribution is:

$$\log p(x) = \sum_{i=1}^{D} \log p(x_i|pa_i)$$

$$= -\sum_{i=1}^{D} \frac{1}{2v_i}(x_i - \sum_{j \in pa_i} w_{ij} x_j - b_i)^2 + Const$$

Linear-Gaussian Models(2)

Determining the mean and covariance of the joint distribution recursively:

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

Taking the expectation:

$$E[x_i] = \sum_{j \in pa_i} w_{ij} E[x_j] + b_i. \quad (E[\epsilon_i] = 0)$$

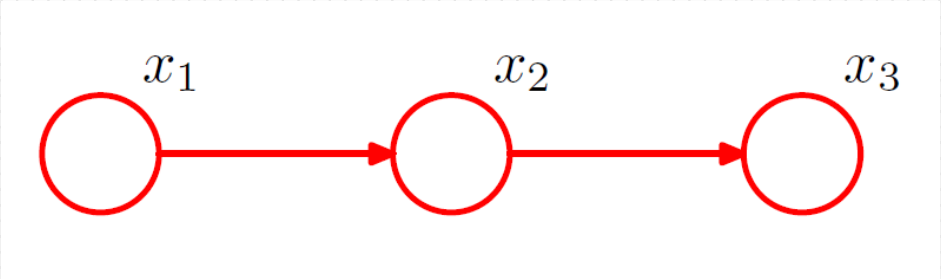
Linear-Gaussian Models (3)

The covariance for node x_i, x_j (i<j):

$$
\begin{aligned}
cov[x_i, x_j] &= E[(x_i - E[x_i])(x_j - E[x_j])] \\
&= E[(x_i - E[x_i])\{\sum_{k \in pa_j} w_{jk}(x_k - E[x_k]) + \sqrt{v_j}\epsilon_j\}] \\
&= \sum_{k \in pa_j} w_{jk} cov[x_i, x_k] + E[(x_i - E[x_i])\sqrt{v_j}\epsilon_j] \\
E[(x_i - E[x_i])\sqrt{v_j}\epsilon_j] &= E[\{\sum_{m \in pa_i} w_{im}(x_m - E[x_m]) + \sqrt{v_i}\epsilon_i\}\sqrt{v_j}\epsilon_j] \\
&= E[\sqrt{v_i}\epsilon_i\sqrt{v_j}\epsilon_j] \quad (m < i < j) \\
&= E[\epsilon_i\epsilon_j]v_j \quad (v_i = v_j = v) \\
&= I_{ij}v_j \quad (E[\epsilon_i\epsilon_j] = I_{ij})
\end{aligned}
$$

Linear-Gaussian Models (4)

Example:



$$\boldsymbol{\mu} = (b_1, b_2 + w_{21} b_1, b_3 + w_{32} b_2 + w_{32} w_{32} b_1)^T,$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} v_1 & w_{21} v_1 & w_{32} w_{21} v_1 \\ w_{21} v_1 & v_2 + w_{21}^2 v_1 & w_{32}(v_2 + w_{21}^2 v_1) \\ w_{32} w_{21} v_1 & w_{32}(v_2 + w_{21}^2 v_1) & v_3 + w_{32}^2 (v_2 + w_{21}^2 v_1) \end{bmatrix}$$

3. Conditional Independence

Definition

Conditional independence:

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

Another expression:

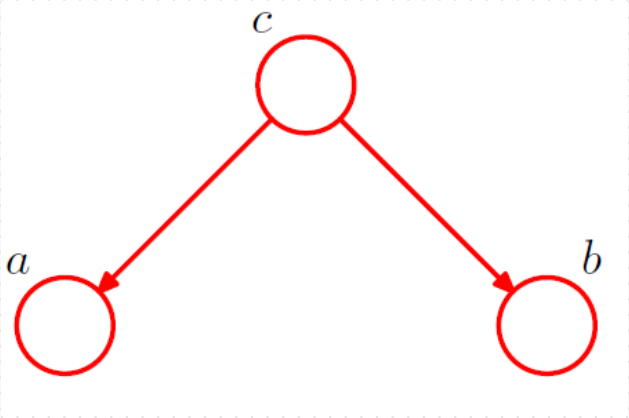
$$\begin{aligned} p(a,b|c) &= p(a|b,c)p(b|c) \\ &= p(a|c)p(b|c) \end{aligned}$$

Shorthand notation:

$$a \perp\!\!\!\perp b|c$$

Three Typical Examples (1)

Example A: (Hidden c)

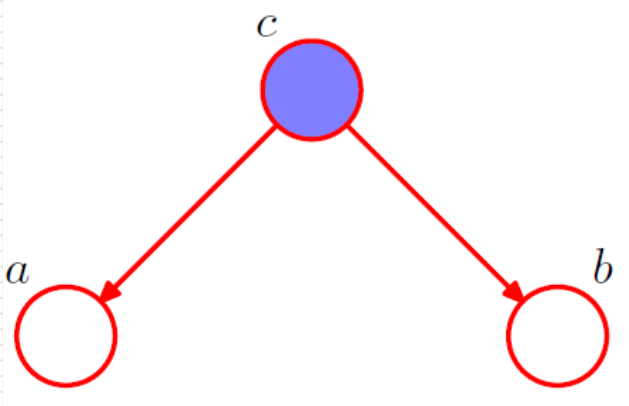


$$p(a,b) = \sum_c p(a,b,c)$$
$$= \sum_c p(a|c)p(b|c)p(c)$$

Therefore: $a \not\perp\!\!\!\perp b|\emptyset$.

Three Typical Examples (2)

Example A: (Observed c)

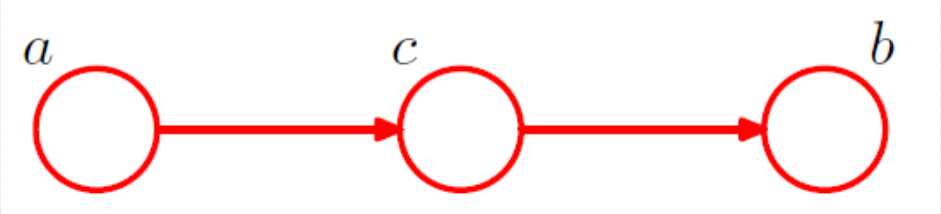


$$p(a,b|c) = \frac{p(a,b,c)}{p(c)}$$
$$= p(a|c)p(b|c)$$

Therefore: $a \perp\!\!\!\perp b|c$. (tail-to-tail)

Three Typical Examples (3)

Example B: (Hidden c)



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

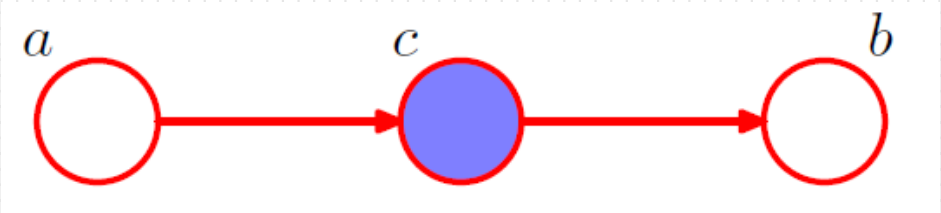
$$= \sum_c p(a)p(c|a)p(b|c)$$

$$= p(a)p(b|a) \text{(Total Probability Theorem)}$$

Therefore: $a \not\perp\!\!\!\perp b|\emptyset$.

Three Typical Examples (4)

Example B: (Observed c)

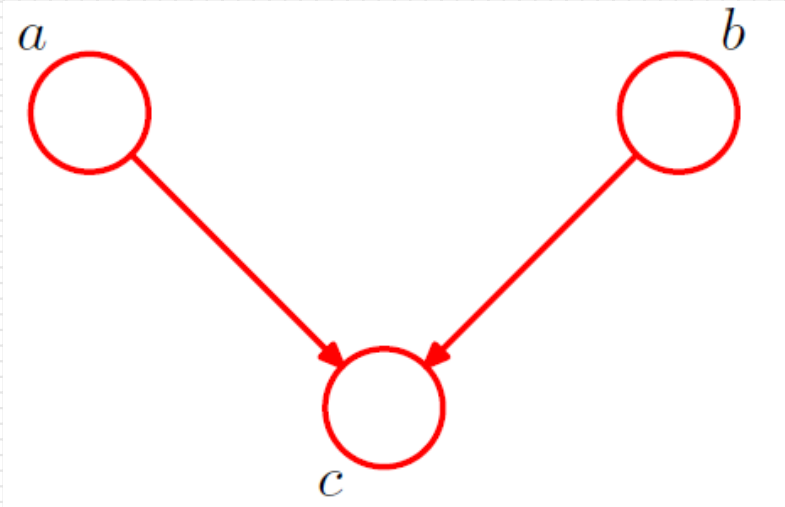


$$p(a,b|c) = \frac{p(a,b,c)}{p(c)}$$
$$= \frac{p(a)p(c|a)p(b|c)}{p(c)}$$
$$= p(a|c)p(b|c) \text{(Bayes' Rule)}$$

Therefore: $a \perp\!\!\!\perp b|c$. (head-to-tail)

Three Typical Examples (5)

Example C: (Hidden c)

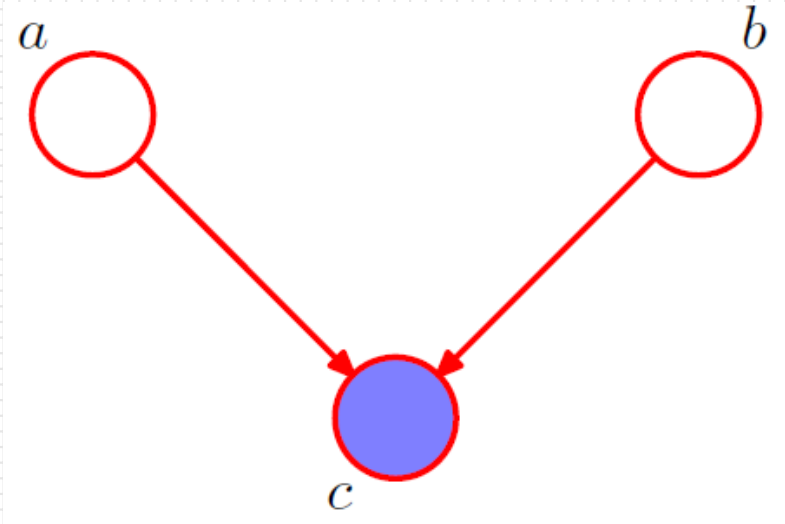


$$p(a,b) = \sum_c p(a,b,c)$$
$$= \sum_c p(a)p(b)p(c|a,b)$$
$$= p(a)p(b)$$

Therefore: $a \perp\!\!\!\perp b | \emptyset$.

Three Typical Examples (6)

Example C: (Observed c)

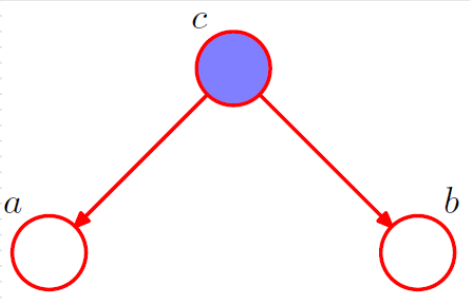


$$p(a,b|c) = \frac{p(a,b,c)}{p(c)}$$
$$= \frac{p(a)p(b)p(c|a,b)}{p(c)}$$

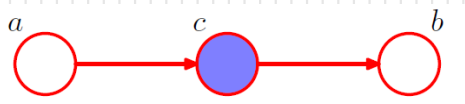
Therefore: $a \not\perp\!\!\!\perp b|c$. (head-to-head)

Takeaways

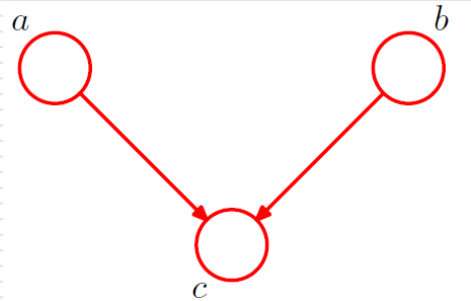
- A tail-to-tail node or a head-to-tail node leaves a path blocked if it is observed.
- A head-to-head node leaves a path unblocked if it and/or one of its descendants are observed.



$a \perp\!\!\!\perp b | c$



$a \perp\!\!\!\perp b | c$



$a \perp\!\!\!\perp b | \emptyset$

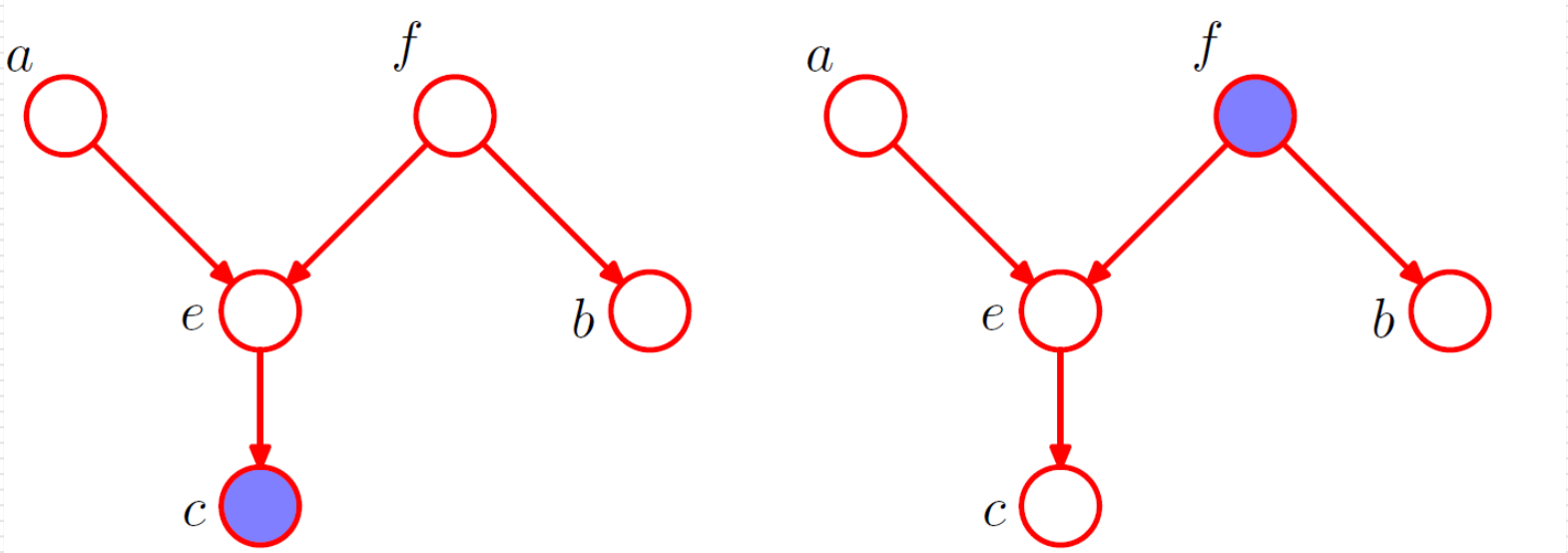
D-separation (1)

Considering a general directed graph in which A, B, and C are arbitrary nonintersecting sets of nodes. Any path, from any node in A to any node in B, is said to be blocked if it includes 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 in the set C, or
2. the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, is in the set C.

D-separation (2)

If all paths are blocked, the A is said to be d-seperated from B by C, and joint distribution over all of the variables in the graph will satisfy $A \perp\!\!\!\perp B | C$.

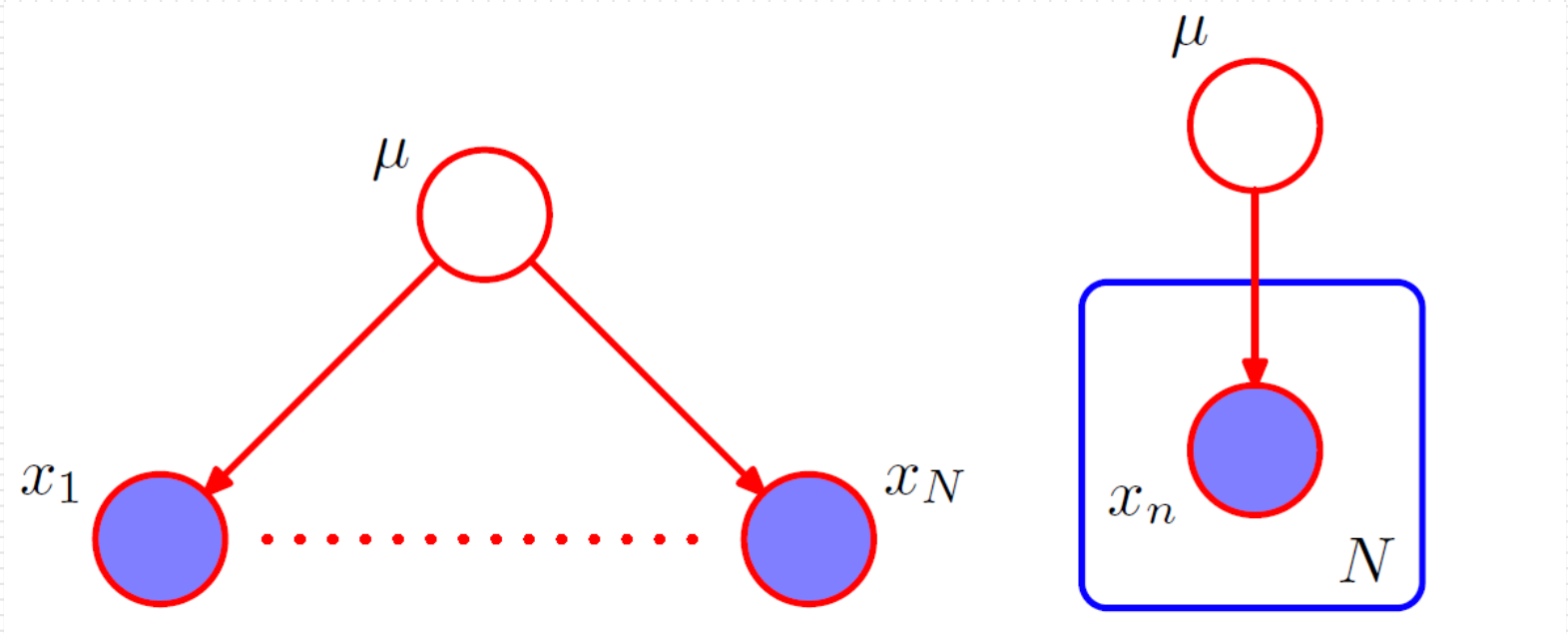


D-separation in I.I.D.

μ observed: $p(D|\mu) = \prod_{n=1}^{N} p(x_n|\mu)$.

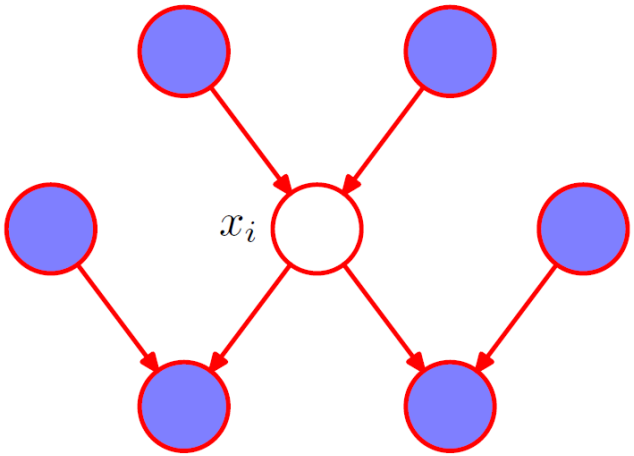
μ hidden:

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



Markov Blanket (or Boundary)

The Markov blanket of a node x_i comprises the set of parents, children and co-parents of the node.



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

$$= \frac{\prod_k p(\mathbf{x}_k|\mathrm{pa}_k)}{\int \prod_k p(\mathbf{x}_k|\mathrm{pa}_k)\,\mathrm{d}\mathbf{x}_i}$$

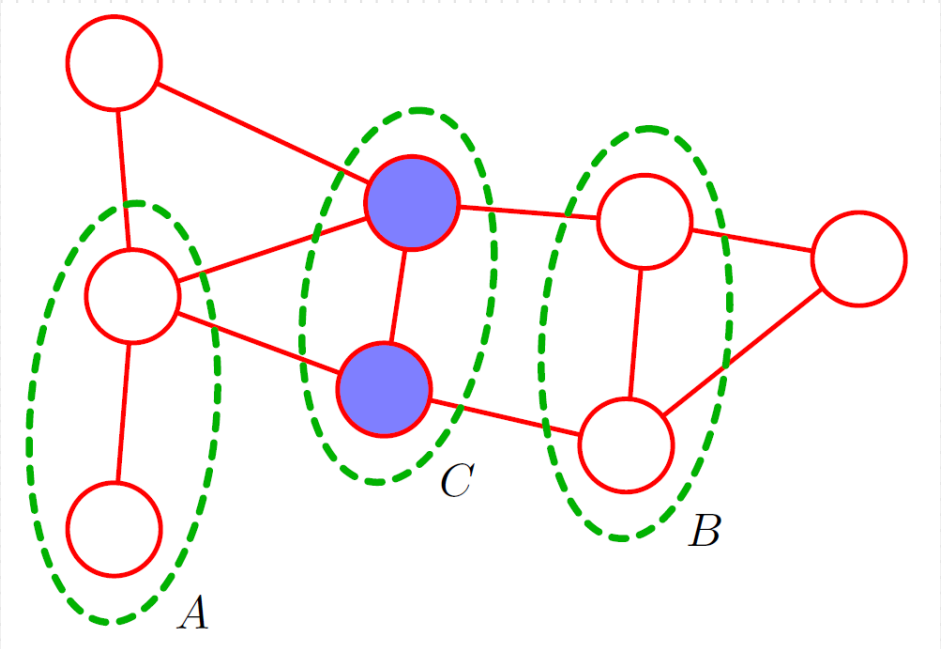
4. Markov Random Fields

Definition

A Markov random field, also known as a Markov network or an undirected graphical model (Kindermann and Snell, 1980), has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links each of which connects a pair of nodes. The links are undirected, that is they do not carry arrows.

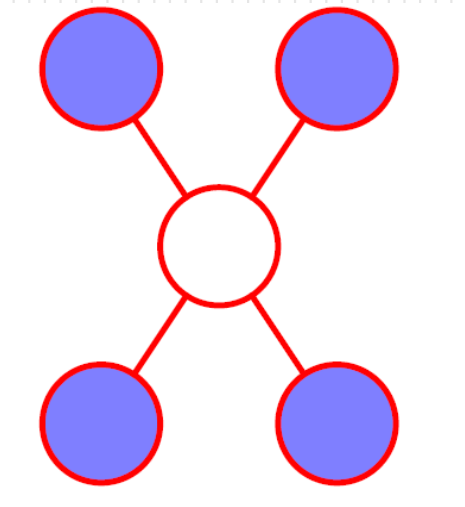
Conditional Independence

Considering all possible paths that connect nodes in set A to nodes in set B. If all such paths pass through one or more nodes in set C, then all such paths are blocked ($A \perp\!\!\!\perp B | C$):



Markov Blanket

Conditionally independent of all other nodes conditioned only on the neighbouring nodes.



Factorization

Considering two nodes x_i and and x_j that are not connected by a link, then these variables must be conditionally independent given all other nodes in the graph:

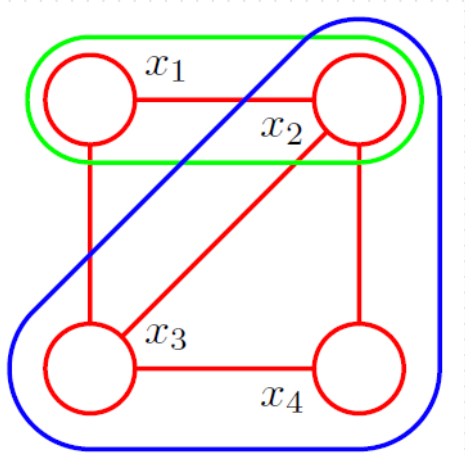
$$p(x_i, x_j | x_{\setminus\{i,j\}}) = p(x_i | x_{\setminus\{i,j\}}) p(x_j | x_{\setminus\{i,j\}})$$

where $x_{\setminus\{i,j\}}$ denotes the set x of all variables with x_i and x_j removed.

Clique

Definition: a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset.

Maximal clique: a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique.



Potential Function

The joint distribution can be written by potential functions $\psi_C(\mathbf{x}_c)$ and maximal cliques:

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

where Z, sometimes called partition functions, is a normalization constant and is given by: (Normalization in DAG?)

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

Normalization in DAG

In the directed graph, each factor represents the conditional distribution of the corresponding variable, conditioned the state of its parents. In other words, the conditional probability distributions of all factors are normalized by the probability of their parents. Therefore, the joint distribution is automatically normalized.

Hammersley-Clifford Theorem

$\mathcal{UI}$:the set of such distributions that are consistent with the set of conditional independence statements that can be read from the graph using graph separation.
$\mathcal{UF}$:the set of such distributions that can be expressed as a factorization of the form with respect to the maximal cliques of the graph.
Hammersley-Clifford Theorem states: the sets $\mathcal{UI}$ and $\mathcal{UF}$ are identical.

Energy Function

Restricting potential functions positively and expressing them as exponentials:

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

where $E(\mathbf{x}_c)$ is called an energy function, and the exponential representation is called the Boltzmann distribution. The bigger energy function is, the smaller potential function is.

Illustration: Image de-noising (1)

Taking an unknown noise-free image, described by binary pixel values $x_i \in \{-1, +1\}$ with index $i = 1, \cdots, D$ over all pixels.

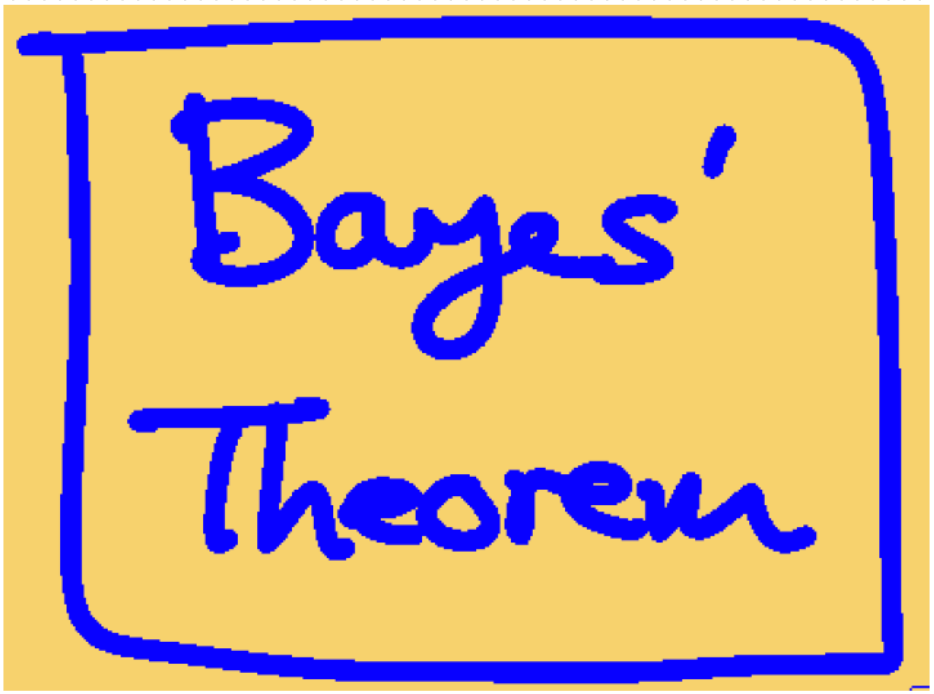


Illustration: Image de-noising (2)

Randomly flipping the sign of pixels with some small probability (10% for this). The noisy image is described by binary pixel values $y_i \in \{-1, +1\}$.

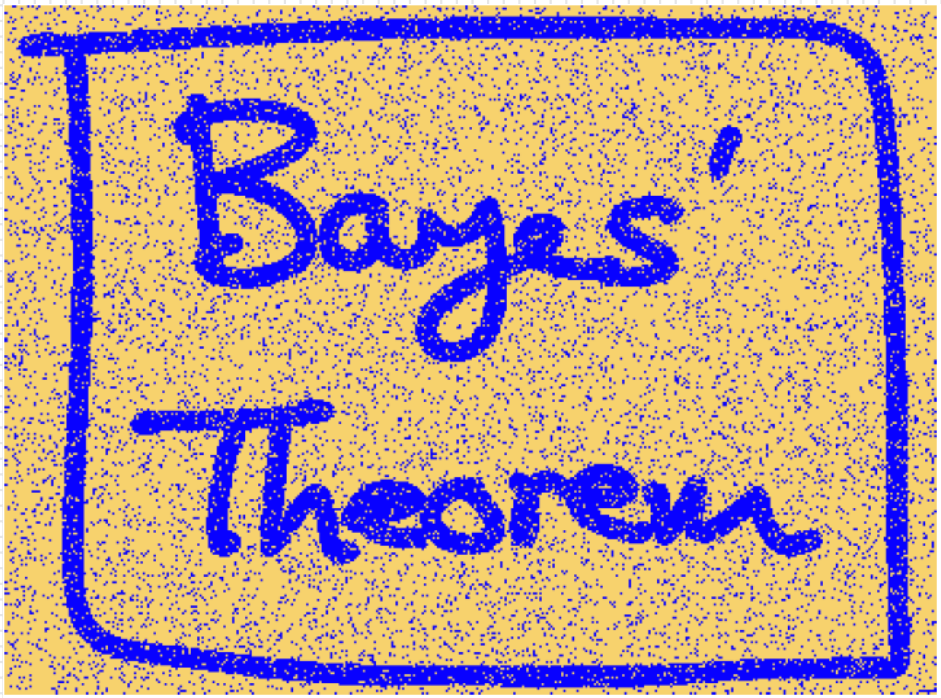


Illustration: Image de-noising (3)

Using MRF to capture the prior knowledge:

- the strong correlation between x_i and y_i (the small noise).
- the strong correlation between neighbouring pixels x_i and x_j.

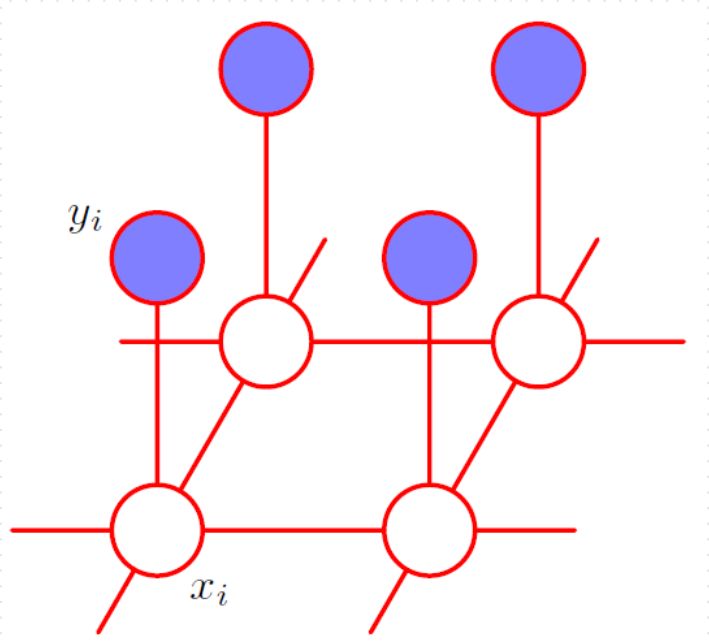


Illustration: Image de-noising (4)

Constructing energy function:

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

- h is a bias term that makes model prefer one particular sign.
- β and η are positive constants that make x_i and x_j, x_i and y_i have same sign respectively.

Illustration: Image de-noising (5)

Using ICM (iterated conditional modes) to solve the model:

1. Initializing the variables $\{x_i\} : x_i = y_i$;
2. Choosing one node x_j's state (± 1) for the lower energy at a time while keeping all other node variables fixed.
3. Repeating step 2 until stopping criterion is satisfied.

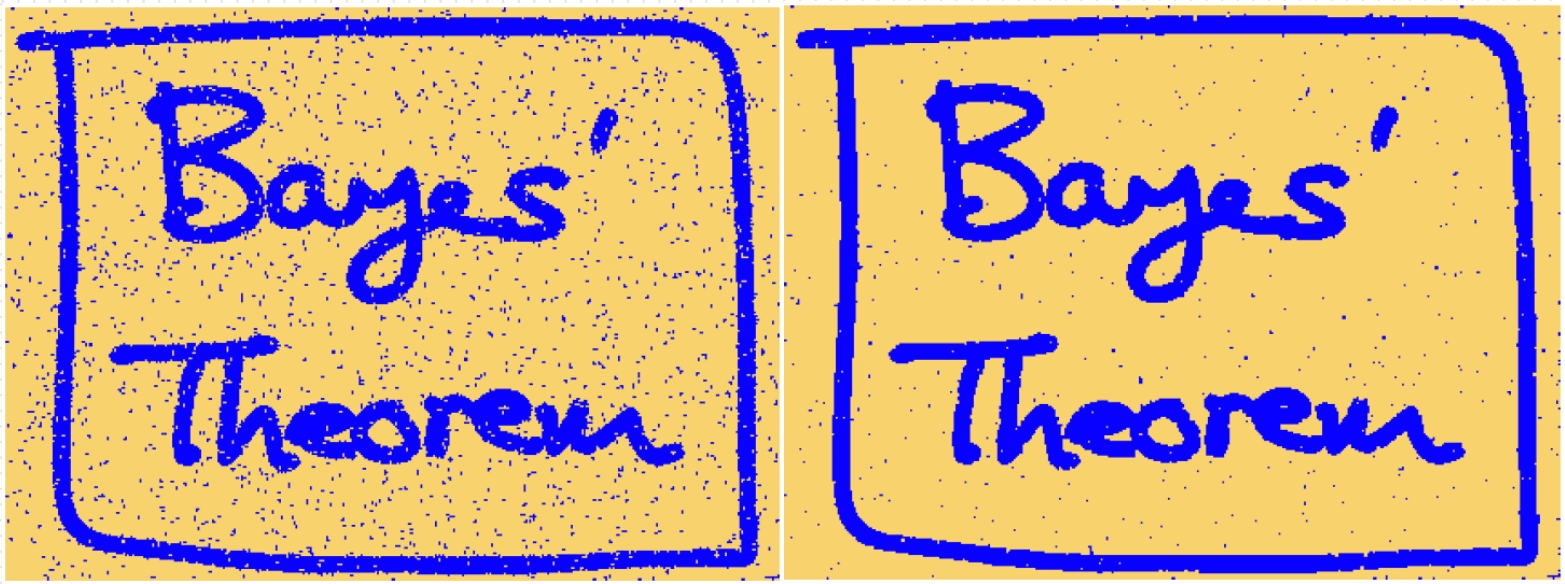
Illustration: Image de-noising (6)

Another way to minimize the energy function is the graph cut algorithm. The graph cut algorithm can global minimize a class of functions:

$$E(x_1, \cdots, x_n) = \sum_i E^i(x_i) + \sum_{i<j} E_{i,j}(x_i, x_j) + \sum_{i<j<k} E_{i,j,k}(x_i, x_j, x_k)$$

Illustration: Image de-noising (7)

Left figure: ICM with $\beta = 1.0, \eta = 2.1, h = 0$. (Local minimum)
Right figure: Algorithms based on graph cut. (Global minimum)



DAG and UAG

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

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

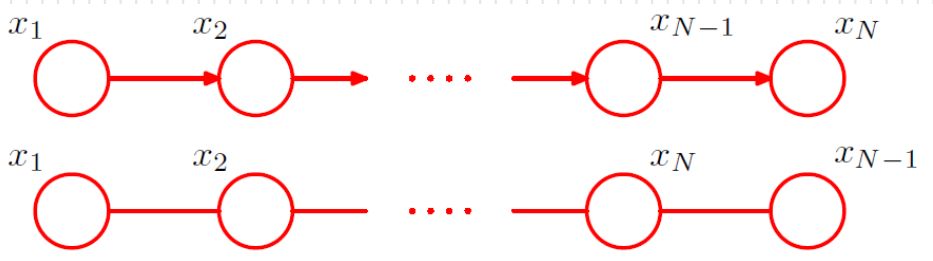
Transformation:

$$\psi_{1,2}(x_1, x_2) = p(x_1)p(x_2|x_1);$$

$$\psi_{2,3}(x_2, x_3) = p(x_3|x_2);$$

$$\vdots$$

$$\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N|x_{N-1}).$$

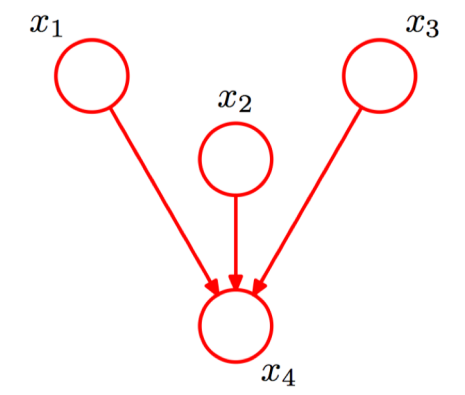


Transfer DAG to UAG

The joint distribution of the following DAG is:

$$p(x) = p(x_1)p(x_2)p(x_3)p(x_4|x_1x_2x_3)$$

The $x_1, x_2, x_3.x_4$ must all belong to a single clique if this conditional distribution is to be absorbed into a clique potential.

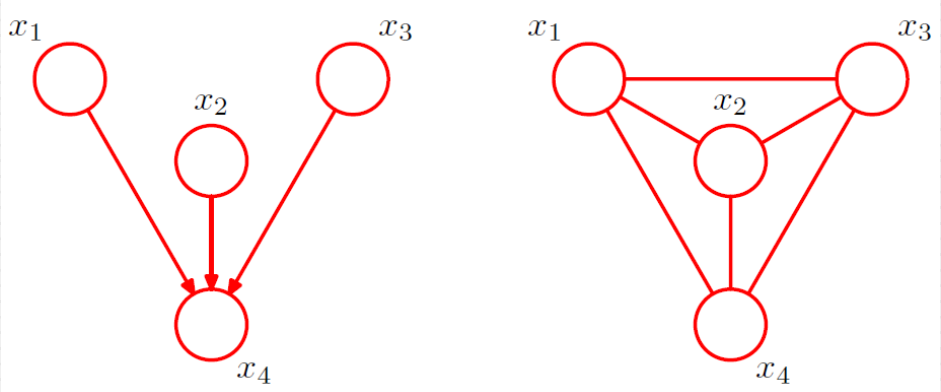


Moral Graph

To make the x_1, x_2, x_3, x_4 form a clique, we should add links between all pairs of parents of x_4

Moralization: process of "marring the parents" (adding links).

Moral graph: moralizing and dropping the arrows.



General to convert DAG to UAG

1. Generating corresponding moral graph;
2. Initializing all of the clique potentials of the moral graph to 1;
3. Taking each conditional distribution factor in the original directed graph and multiplying it into one of the clique potentials.

Notice: discarding some conditional independence properties.

Other Graphs of Conditional Independence

Dependence map: every conditional independence statement satisfied by the distribution is reflected in the graph.
Independence map: every conditional independence statement implied by a graph is satisfied by a specific distribution.
Perfect map: dependence map + independence map.

5. Inference in Graphical Models

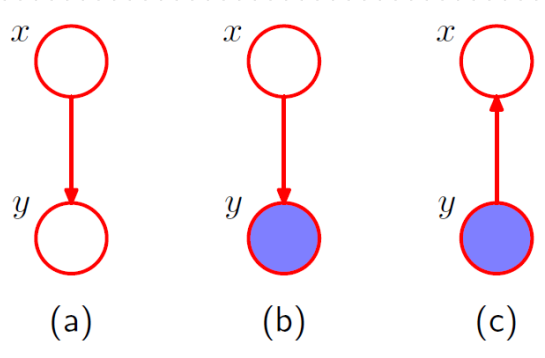
Introduction

Inference:

compute the posterior distribution of nodes with other observed nodes.

A simple example: Bayes' theorem.

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



Inference on A Chain

Joint distribution:

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

Marginal distribution for node x_n: $O(K^{N-1})$

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

Efficient Computation with Conditional Independence

Rearranging the order of the summations and the multiplications: $O(NK^2)$

$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)} . \qquad (8.52)$$

Local Messages (1)

The expression of $p(x_n)$ can be written as:

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

where $\mu_\alpha(x_n)$ can be viewed as message passed from x_{n-1} and evaluated recursively:

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

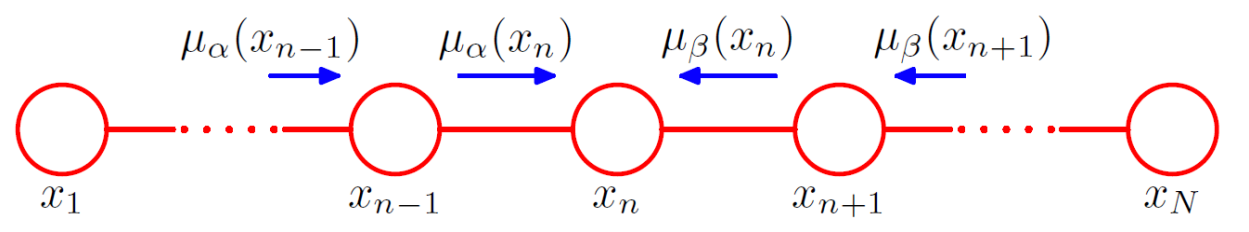
Local Messages (2)

and $\mu_\beta(x_n)$ can be viewed as message passed from x_{n+1} and evaluated recursively:

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

and Z can be easily evaluated by summing $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$ over all states of x_n. $O(K)$

Efficient Computation with Recurrence Relation



Computation Cost: $O(N^2K^2)$. Wasteful!
Solution: store intermediate messages.
$O(2NK^2)$
Notice: Z can be evaluated once by any convenient node.

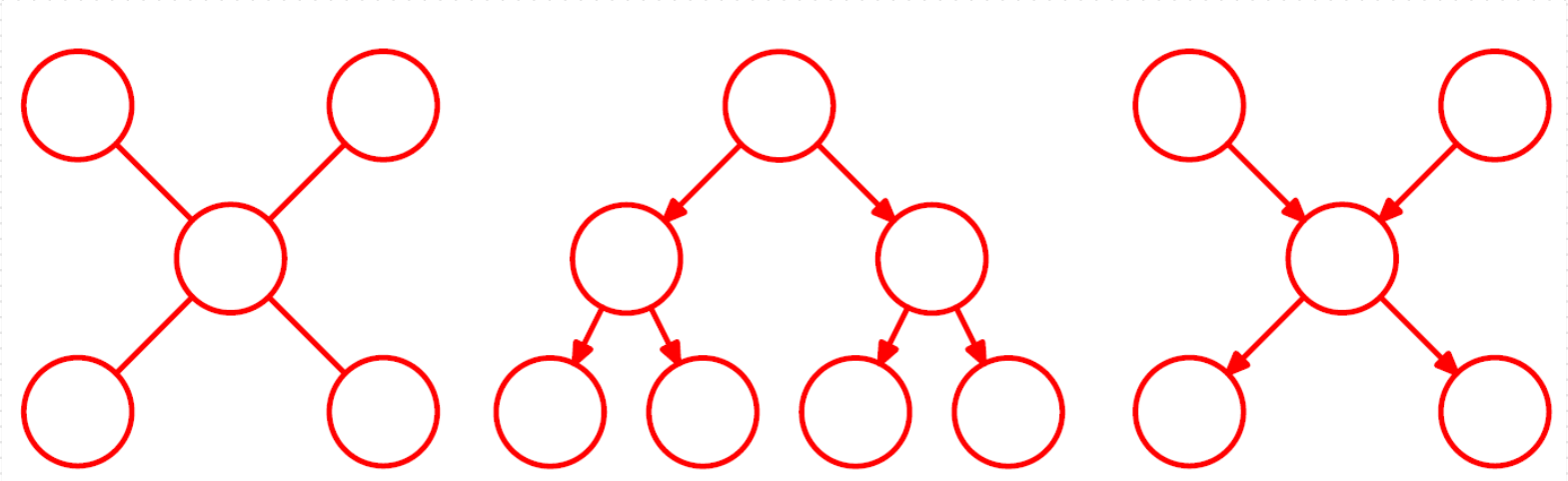
Joint Distribution

Calculating the joint distribution $p(x_{n-1}, x_n)$ for two neighbouring nodes on the chain:

$$p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n).$$

Trees

Definition: an undirected graph in which there is one, and only one, path between any pair of nodes.



From left to right: an undirected tree, a directed tree and a polytree.

Factor Graph

Both directed and undirected graphs allow a global function of several variables 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.

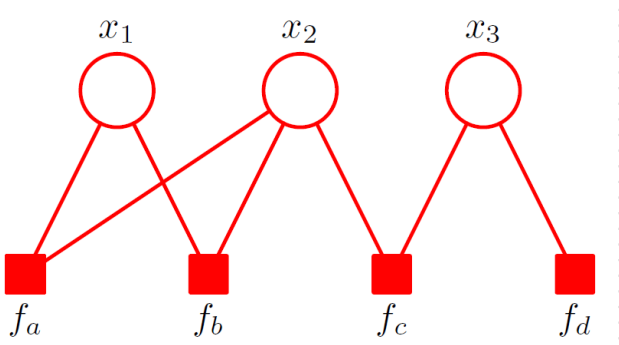
Representation

Writing the joint distribution over a set of variables in the form of a product of factors:

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

where factor f_s is a function of a corresponding set of variables x_s.

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



Bipartite

Illustration (1)

- An undirected Graph along with two different factor graphs.

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

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

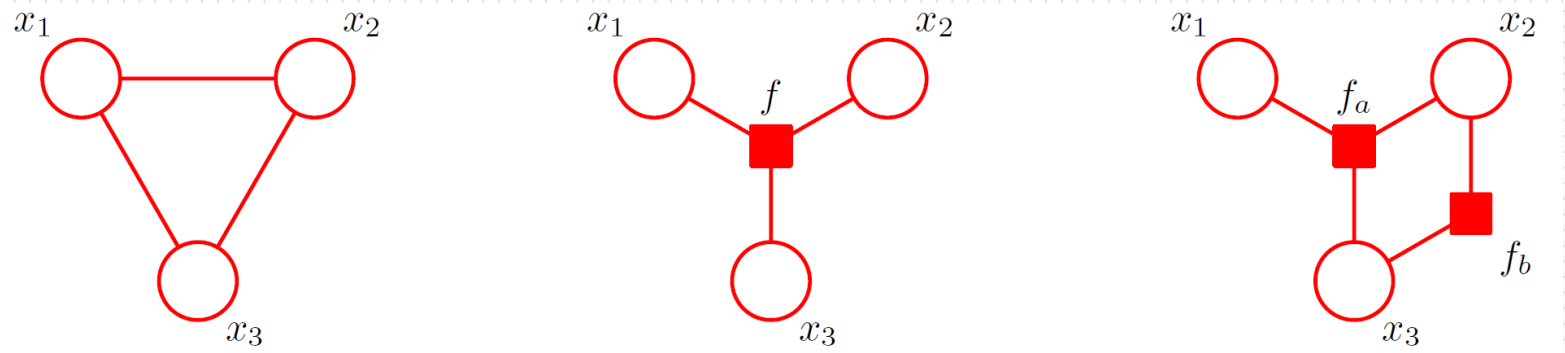


Illustration (2)

- A directed Graph along with two different factor graphs.

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

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

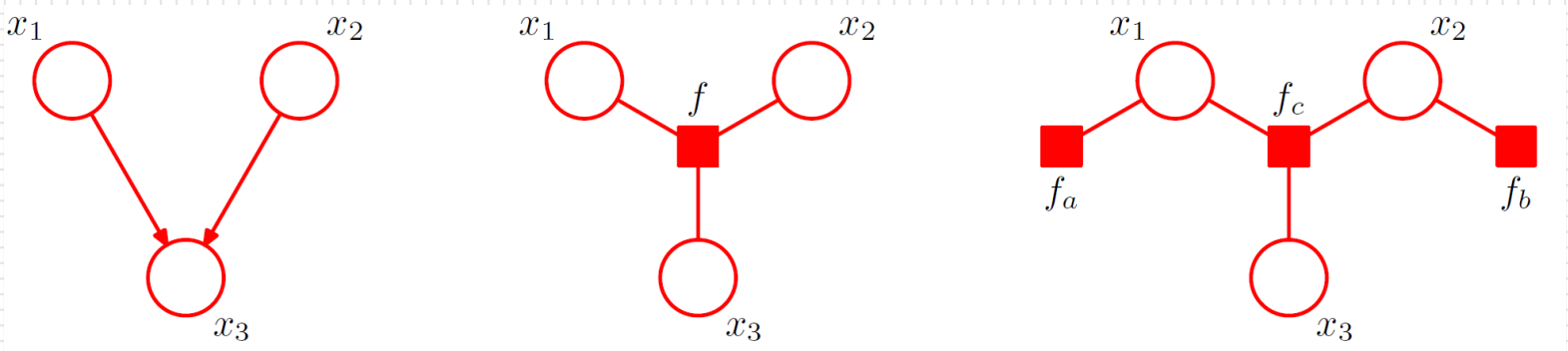


Illustration (3)

- A directed polytree, conversion to an undirected graph results in loops due to the moralization step, whereas conversion to a factor graph again results in a tree.

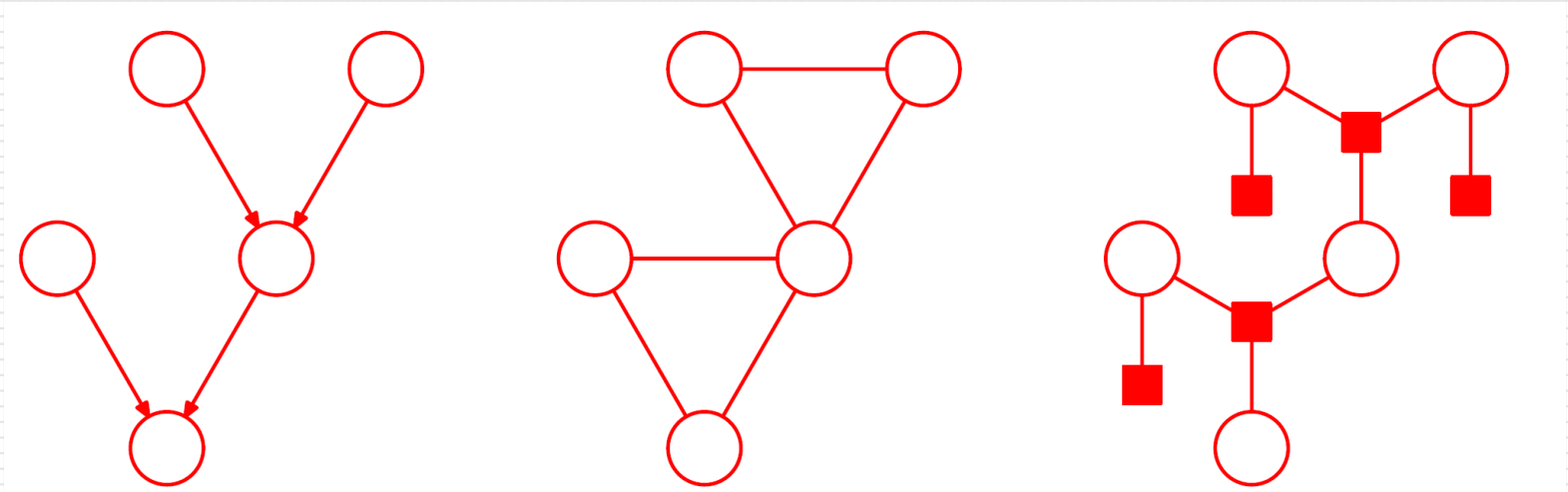


Illustration (4)

- Local cycles in a directed graph due to links connecting parents of a node can be removed on conversion to a factor graph by defining the appropriate factorfunction.

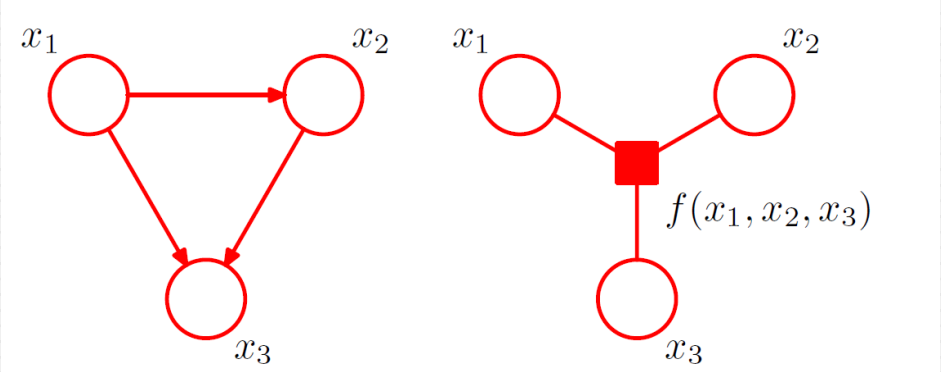
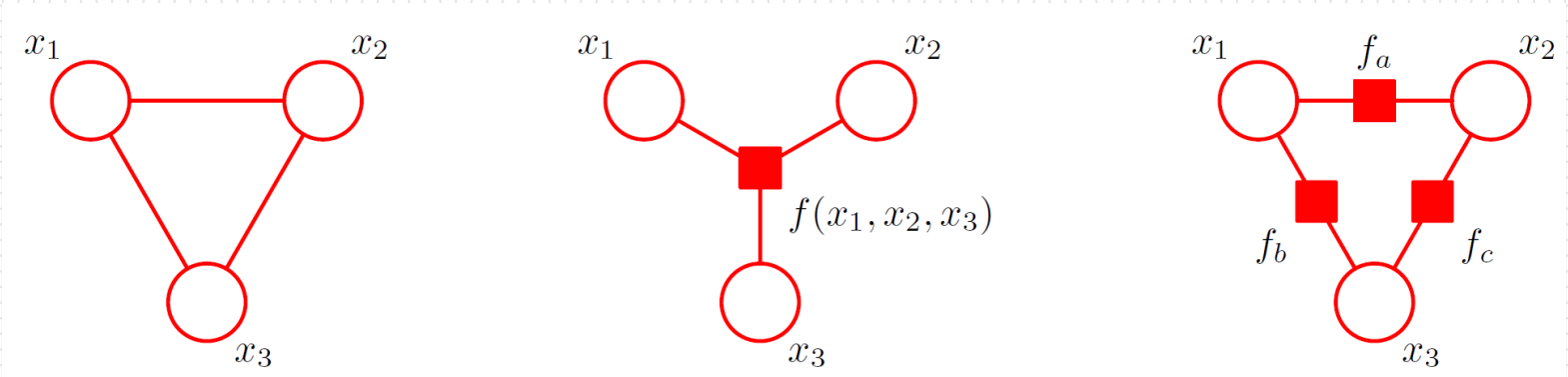


Illustration (5)

- A fully connected undirected graph along with two different factor graphs.

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

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



The Sum-product Algorithm

Objective: a exact inference algorithm that focuses on the problem of evaluating local marginals efficiently over nodes or subset of nodes in tree-structured graphs.
Notice: there are other exact inference algorithms on directed graphs without loops, such as belief propagation, but sum-product algorithm is simpler to derive and to apply, as well as being more general.

Derivation (1)

Considering the problem of finding the marginal $p(x)$ for particular variable node x:

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

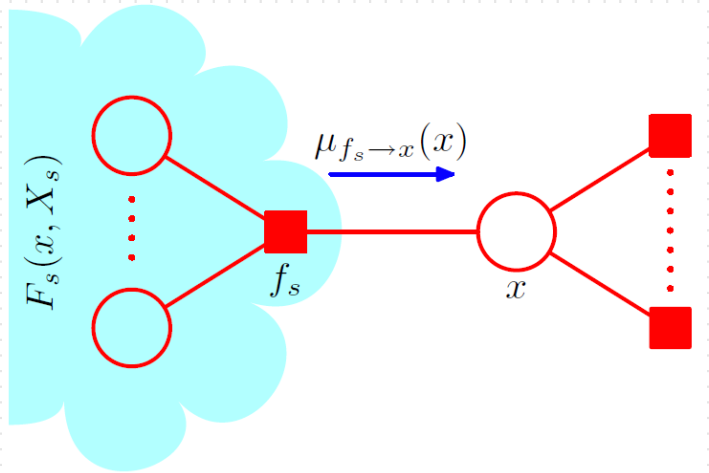
where $\mathbf{x} \setminus x$ denotes the set of variables in $\mathbf{x}$ with variable x omitted.

Derivation (2)

The joint distribution can be written as a product of the form:

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

where $ne(x)$ denotes the set of factor nodes that are neighbours of x.



Derivation (3)

Thus,

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

Interchanging the sums and products:

$$p(x) = \prod_{s \in ne(x)} \sum_{X_s} F_s(x, X_s) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x),$$

where $\mu_{f_s \to x}(x)$ is a set of functions and defined by (Factor to Variable)

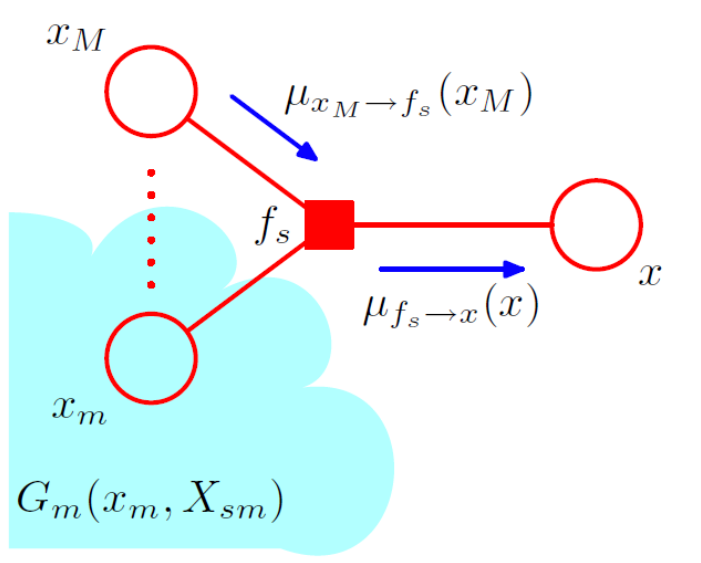
$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s).$$

Derivation (4)

Since each factor $F_s(x, X_s)$ is described by a factor (sub-)graph and factorized, thus

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

where $x_1, \cdots, x_M$ denotes the variables associated with factor f_s, in addition to x.



Derivation (5)

Thus,

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

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

where $\mu_{x_m \to f_s}(x_m)$ is defined as

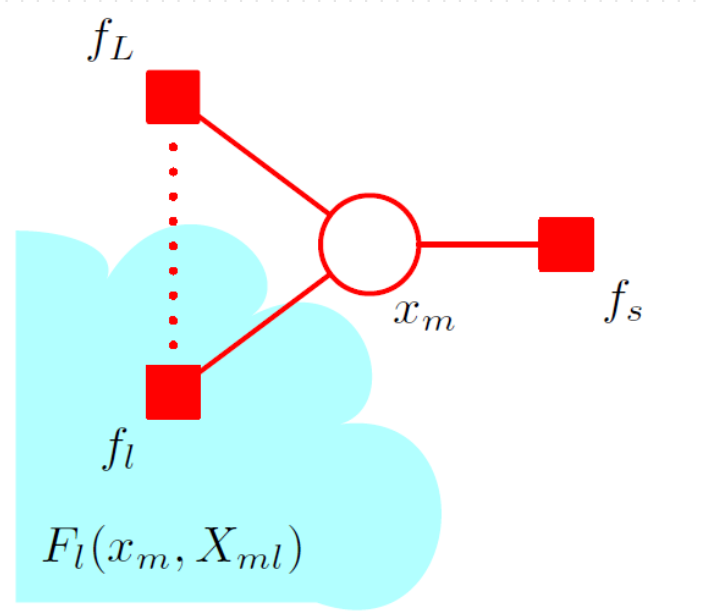
$$\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm}).$$

(Variable to Factor)

Derivation (6)

Making use of the (sub-)graph factorization again:

$$G_m(x_m, X_{sm}) = \prod_{l \in ne(x_m) \setminus f_s} F_l(x_m, X_{lm}).$$



Derivation (7)

Thus,

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

$$= \prod_{l \in ne(x_m) \setminus f_s} \left[\sum_{X_{sm}} F_l(x_m, X_{lm}) \right]$$

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

Derivation (8)

Recursive computing until reaching a leaf node.

If a leaf node is a variable node:

$$\mu_{x \to f}(x) = 1;$$

If a leaf is a factor node:

$$\mu_{x \to f}(x) = f(x).$$

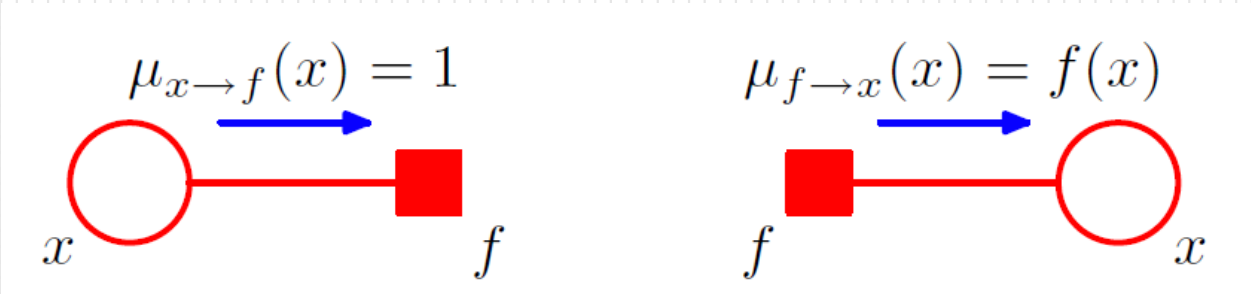


Illustration (1)

Unnormalized joint distribution:

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

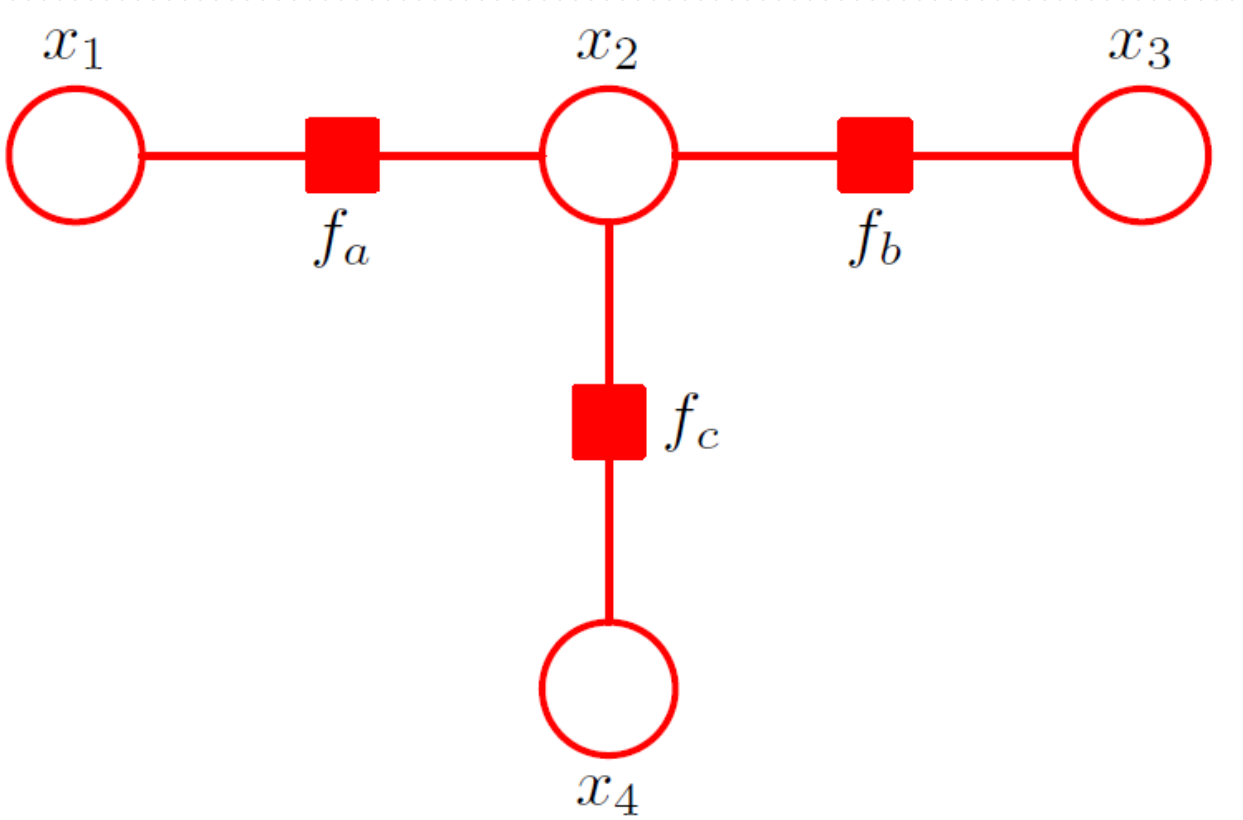


Illustration (2)

Designating node x_3 as the root, in which case there are two leaf nodes x_1 and x_4. Starting with the leafnodes:

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

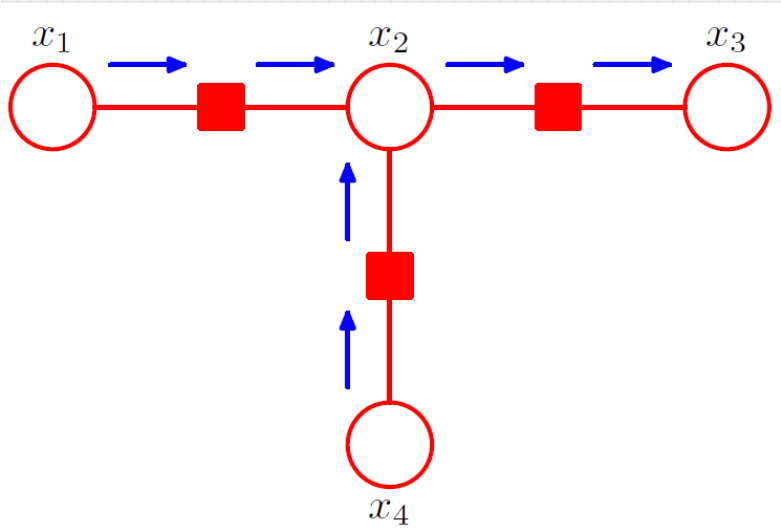


Illustration (3)

Propagating messages from the root node out to the leaf nodes After leafnodes' propagation is complete:

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

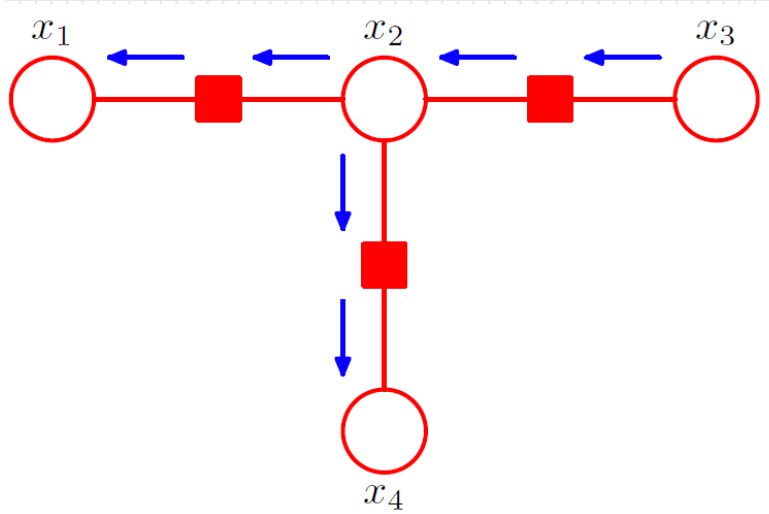


Illustration (4)

Evaluating the marginals:

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

Other Considerations

- Effectiveness: inductive arguments.
- Efficiency: store intermediate result.
- Marginals for factor node: $p(\mathbf{x}_s) = f_s(\mathbf{x}_s) \prod_{i \in ne(f_s)} \mu_{x_i \to f_s}(x_i)$.
- Normalization: automatic for DAG and one computation for UAG.
- Observed variables: mutiply $\prod_i I(v_i, \hat{v}_i)$.
- Continuous variables: replace summations by integration.

The Max-sum Algorithm

Objective:

1. to find a setting of the variables that has the largest probability;
2. to find the value of that probability.

An application of dynamic programming in the context of graphical models.

A Simple Approach

Running the sum-product algorithm to obtain the marginals $p(x_i)$ for every variable, and then, for each marginal in turn, to find the value x_i^* that maximizes that marginal. But in practice, we maximize joint distribution:

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x});$$

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

Joint Distribution vs. Marginal Distribution

Example of a joint distribution over two binary variables for which the maximum of the joint distribution occurs for different variable values compared to the maximum of the two marginals.

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

Maximize Joint Distribution

Objective:

$$\begin{aligned}\max_{\mathbf{x}} p(\mathbf{x}) &= \max_{x_1} \cdots \max_{x_N} p(\mathbf{x}) \\ &= \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} [\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)] \\ &= \frac{1}{Z} \max_{x_1} \left[\max_{x_2} \left[\psi_{1,2}(x_1, x_2) \left[\cdots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right] \right],\end{aligned}$$

which can be applied to arbitrary tree-structured factor graphs. And final maximization is called the max-product algorithm.

Max-sum (1)

Taking logarithm and replacing the products in the max-product algorithm with sums:

$$\max \mu_{f_s \to x}(x) = \max \log \sum_{x_1} \cdots \sum_{x_M} f(x, x_1, \cdots, x_M) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

$$= \max_{x_1, \cdots, x_M} \left[\log \sum_{x_1} \cdots \sum_{x_M} f(x, x_1, \cdots, x_M) + \sum_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \right]$$

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

And for leaf nodes,

$$\mu_{x \to f}(x) = 0, \mu_{f \to x}(x) = \log f(x).$$

Max-sum (2)

Thus,

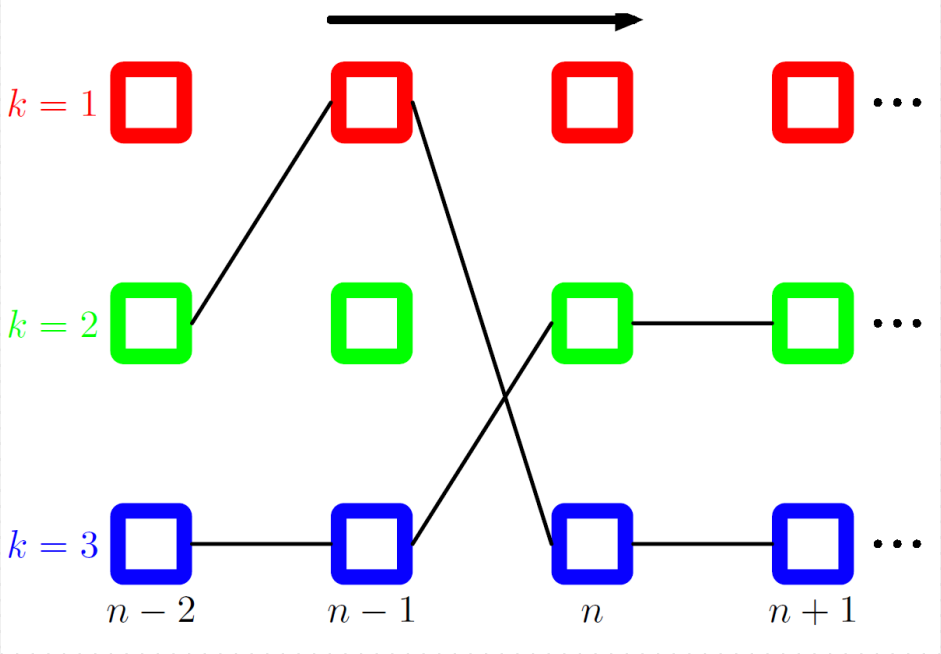
$$p^{\max} = \max_x \left[\sum_{s \in ne(x)} \mu_{f_s \to x}(x) \right],$$

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

But, Local vs. global again!

Lattice Diagram

A lattice, or trellis, diagram showing explicitly the K possible states (one per row of the diagram) for each of the variables x_n in the chain model.



Solution

1. Maximize $x_1, \cdots, x_M$;
2. Recording values of the variables $x_1, \cdots, x_M$ gave rise to the maximum;
3. Using these stored values to assign consistent maximizing states $x_1^{\max}, x_2^{\max}, \cdots, x_M^{\max}$.

Similar to Viterbi algorithm in hidden Markov models.

Exact Inference in General Graphs

- The sum-product and max-sum algorithms provide efficient and exact solutions to inference problems in tree-structured graphs.
- The message passing framework can be generalized to arbitrary graph topologies, giving an exact inference procedure known as the junction tree algorithm.

Loopy Belief Propagation

- For many problems of practical interest, exact inference will not be feasible, so we need effective approximation methods, such as variational method and Monte Carlo methods.
- Loopy belief propagation: a simple approach to approximate inference in graphs with loops, which uses message passing schedule.

Learning Graph Structure

- There is also interest in going beyond the inference problem and learning the graph structure itself from data.
- Computation for marginalization is challenging.
- Exploring the space of structures is problematic.

Application

1. Topic model, like LDA (Latent dirichlet allocation). Blei, 2003.
2. RBM (Restricted Boltzmann Machine). Smolensky, 1986.
3. Graph Neural Networks. Tsinghua University. 2018.