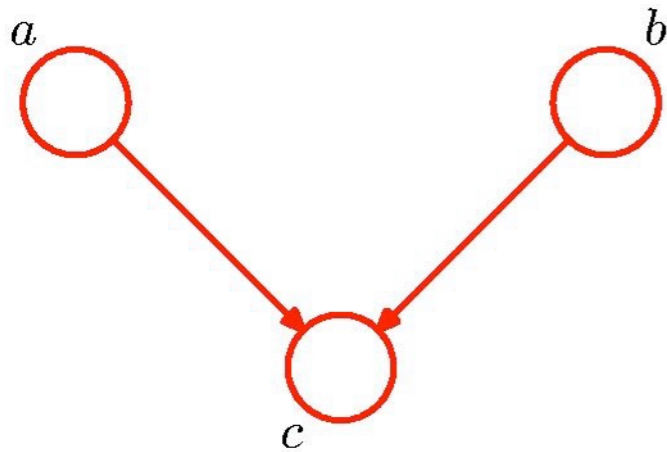


# The Head-to-Head Node



$$p(a) = 0.9 \quad p(b) = 0.9$$

$a$	$b$	$p(c a,b)$
1	1	0.8
1	0	0.2
0	1	0.2
0	0	0.1

Example:

a: Battery charged (0 or 1)

b: Fuel tank full (0 or 1)

c: Fuel gauge says full (0 or 1)

We can compute  $p(\neg c) = 0.315$

and  $p(\neg c | \neg b) = 0.81$

and obtain  $p(\neg b | \neg c) \approx 0.257$

similarly:  $p(\neg b | \neg c, \neg a) \approx 0.111$

**“ $a$  explains  $c$  away”**



# D-Separation

Say:  $A$ ,  $B$ , and  $C$  are non-intersecting subsets of nodes in a directed graph.

A path from  $A$  to  $B$  is **blocked** by  $C$  if it contains a node such that either

- a) the arrows on the path meet either **head-to-tail** or **tail-to-tail** at the node, and the node is **in** the set  $C$ , or
- b) the arrows meet **head-to-head** at the node, and neither the node, nor any of its descendants, are in the set  $C$ .

If all paths from  $A$  to  $B$  are blocked,  $A$  is said to be **d-separated** from  $B$  by  $C$ .

**Notation:**  $dsep(A, B|C)$



# D-Separation

Say:  $A$ ,  $B$ , and  $C$  are non-intersecting subsets of nodes in a directed graph.

• A path

a node

a) the ar

tail at th

b) the a

the nod

• If all p

be **d-separated** from  $B$  by  $C$ .

**Notation:**  $dsep(A, B|C)$

**D-Separation is a  
property of graphs  
and not of  
probability  
distributions**

contains

tail-to-

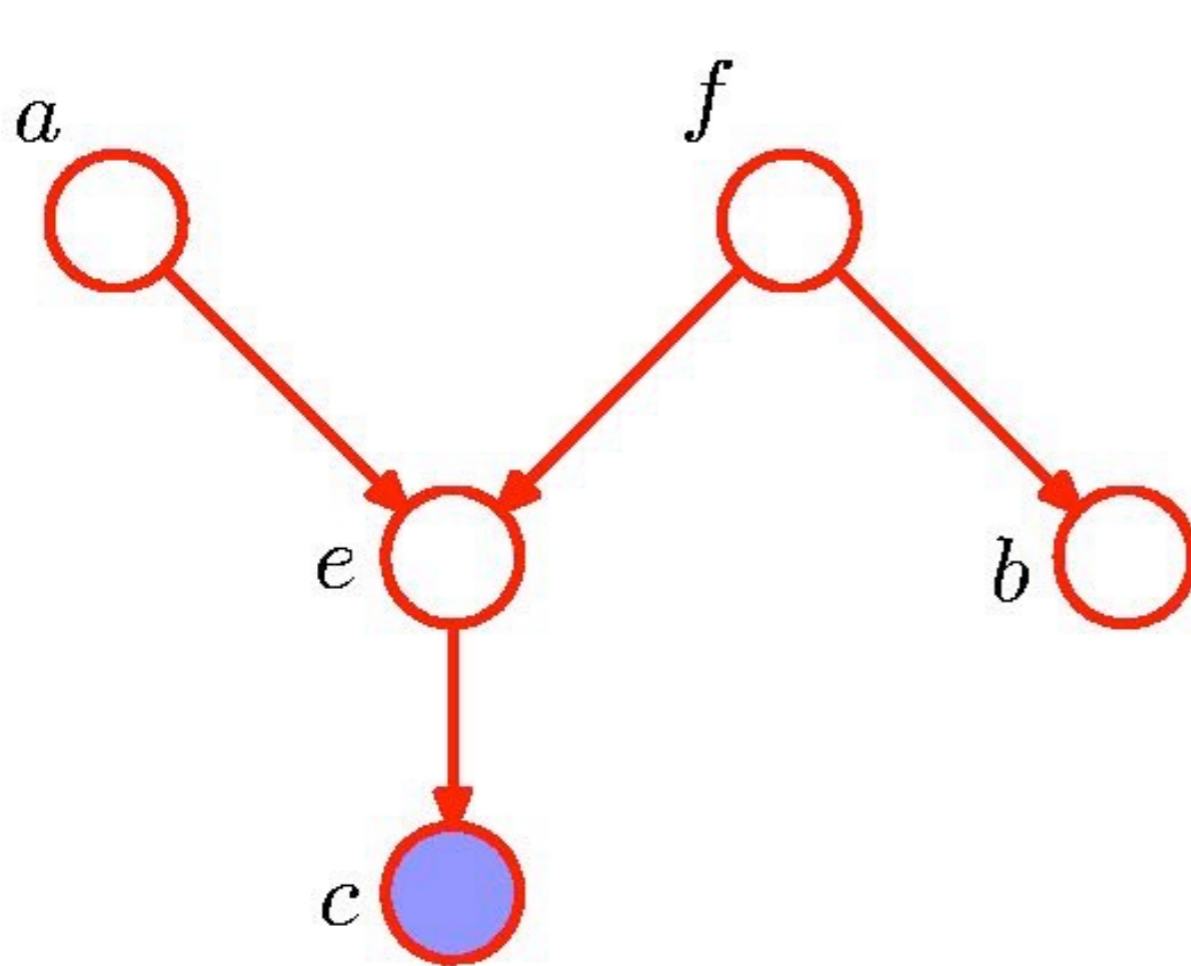
neither

C.

aid to

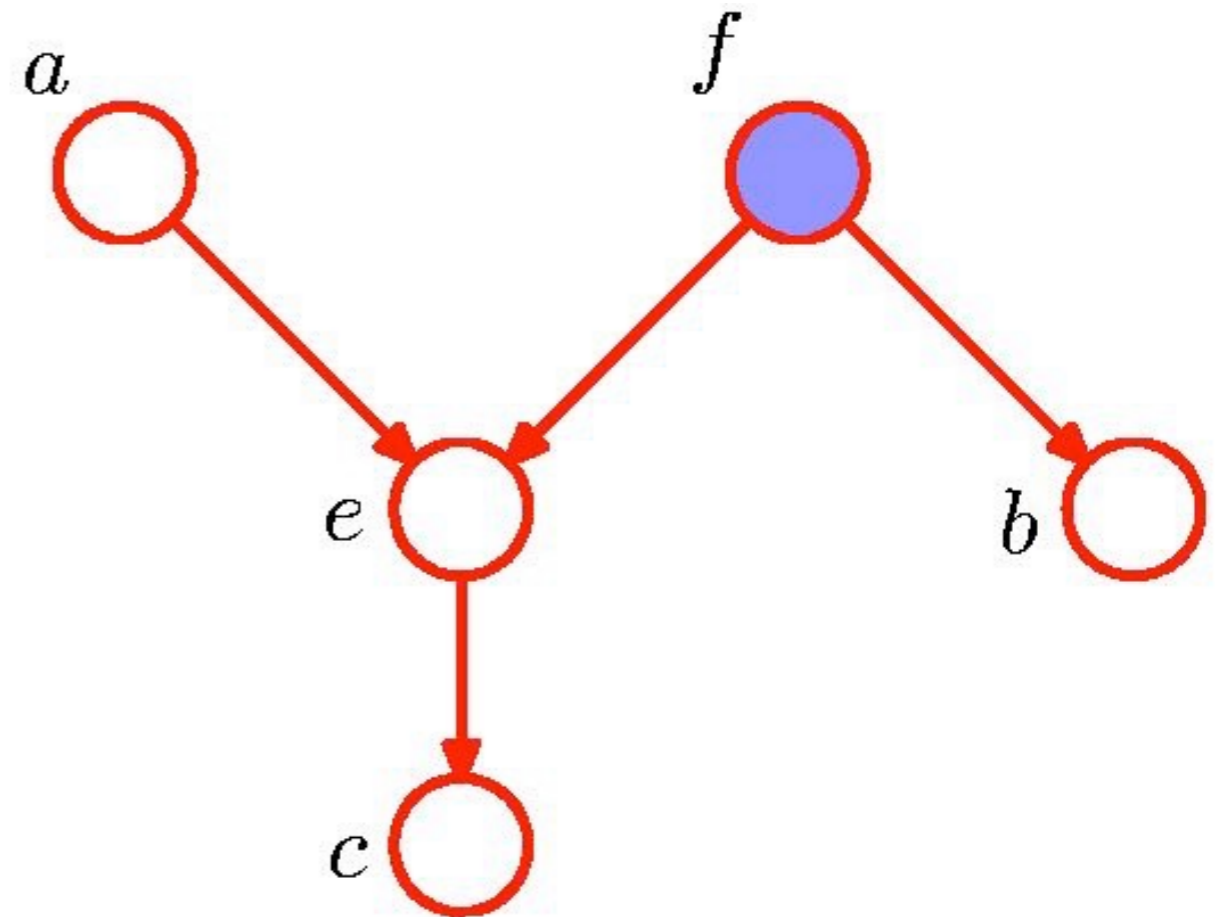


# D-Separation: Example



$$\neg \text{dsep}(a, b | c)$$

We condition on a descendant of  $e$ , i.e. it does not block the path from  $a$  to  $b$ .



$$\text{dsep}(a, b | f)$$

We condition on a tail-to-tail node on the only path from  $a$  to  $b$ , i.e.  $f$  blocks the path.



# I-Map

**Definition 4.1:** A graph  $G$  is called an **I-map** for a distribution  $p$  if every D-separation of  $G$  corresponds to a conditional independence relation satisfied by  $p$ :

$$\forall A, B, C : \text{dsep}(A, B, C) \Rightarrow A \perp\!\!\!\perp B \mid C$$

**Example:** The fully connected graph is an I-map for any distribution, as there are no D-separations in that graph.



# D-Map

**Definition 4.2:** A graph  $G$  is called an **D-map** for a distribution  $p$  if for every conditional independence relation satisfied by  $p$  there is a D-separation in  $G$  :

$$\forall A, B, C : A \perp\!\!\!\perp B \mid C \Rightarrow \text{dsep}(A, B, C)$$

**Example:** The graph without any edges is a D-map for any distribution, as all pairs of subsets of nodes are D-separated in that graph.



# Perfect Map

**Definition 4.3:** A graph  $G$  is called a **perfect map** for a distribution  $p$  if it is a D-map and an I-map of  $p$ .

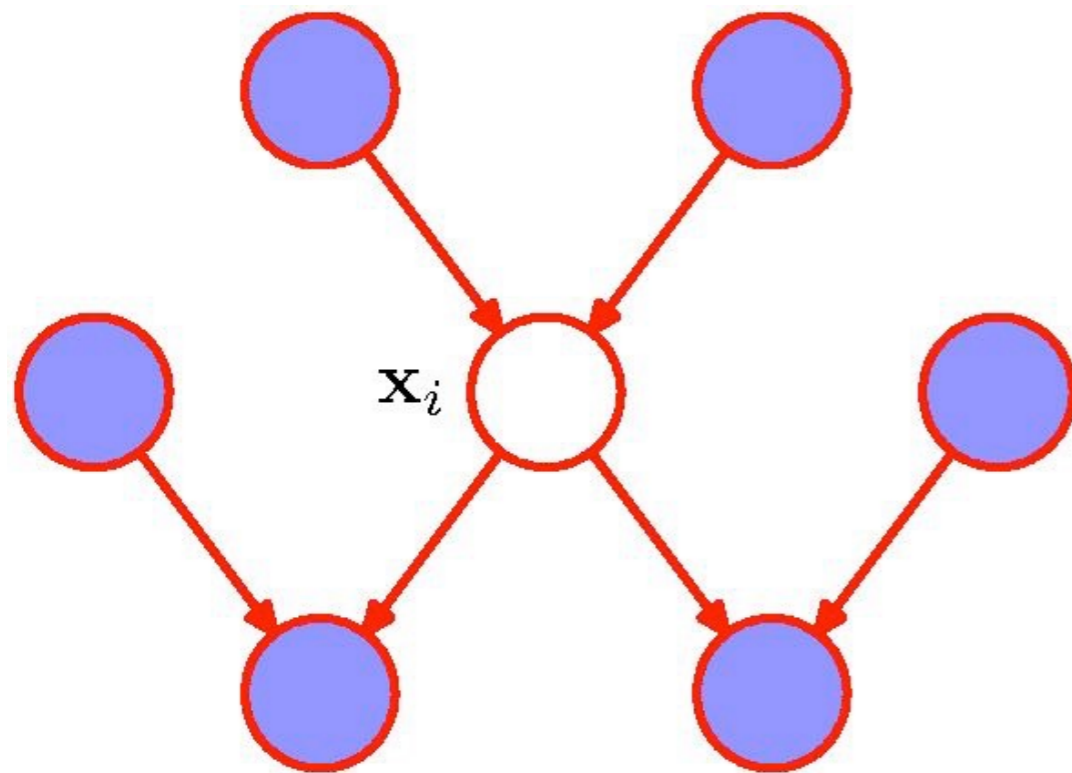
$$\forall A, B, C : A \perp\!\!\!\perp B \mid C \Leftrightarrow \text{dsep}(A, B, C)$$

A perfect map uniquely defines a probability distribution.



# The Markov Blanket

Consider a distribution of a node  $x_i$  conditioned on all other nodes:



**Markov blanket**  $\mathcal{M}_i$  at  $x_i$ : all parents, children and co-parents of  $x_i$ .

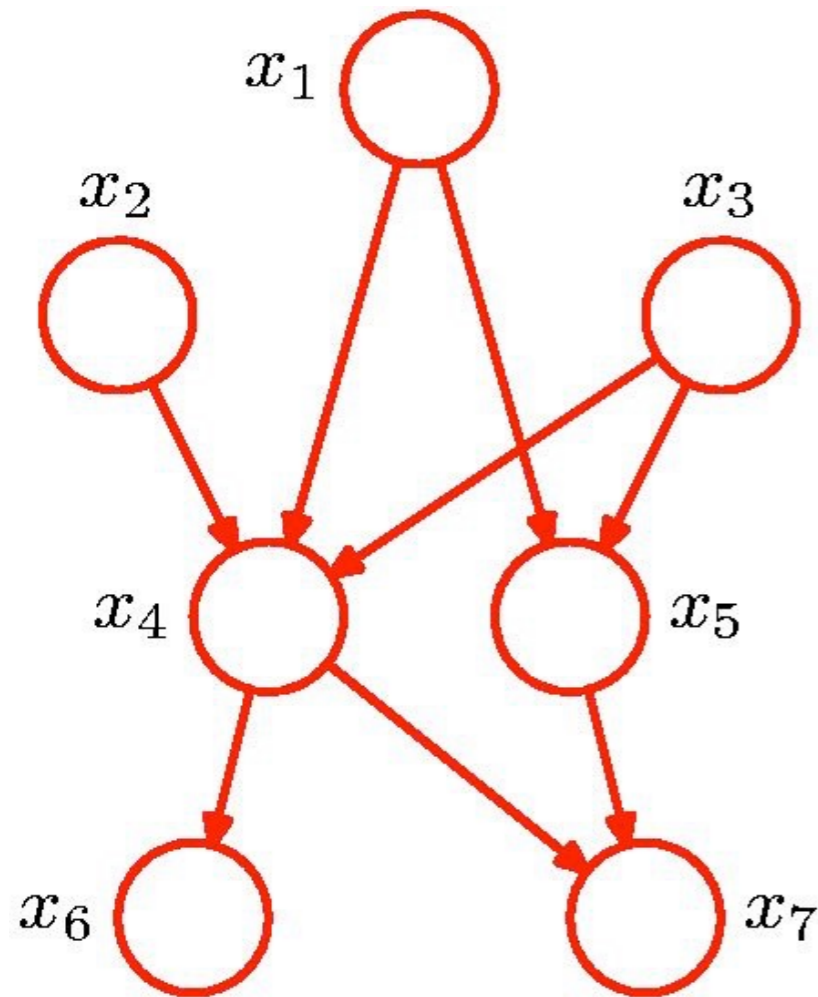
$$\begin{aligned} p(\mathbf{x}_i | \mathbf{x}_{\{j \neq i\}}) &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_M)}{\int p(\mathbf{x}_1, \dots, \mathbf{x}_M) d\mathbf{x}_i} \\ &= \frac{\prod_k p(\mathbf{x}_k | \text{pa}_k)}{\int \prod_k p(\mathbf{x}_k | \text{pa}_k) d\mathbf{x}_i} \end{aligned}$$

Factors independent of  $x_i$  cancel between numerator and denominator.





# Directed Graphical Models

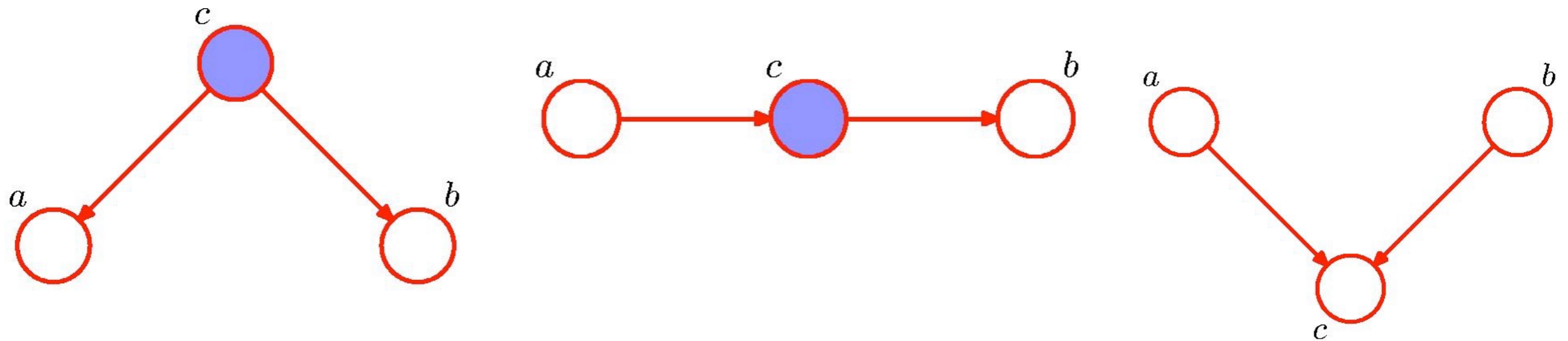


Directed graphical models can be used to represent **probability distributions**. This is useful to do **inference** and to **generate samples** from the distribution efficiently.

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



# Summary D-Separation



- D-separation is a property of graphs that can be easily determined
- An I-map assigns every d-separation a c.i. rel
- A D-map assigns every c.i. rel a d-separation
- Every Bayes net determines a unique prob. dist.



# Directed vs. Undirected Graphs

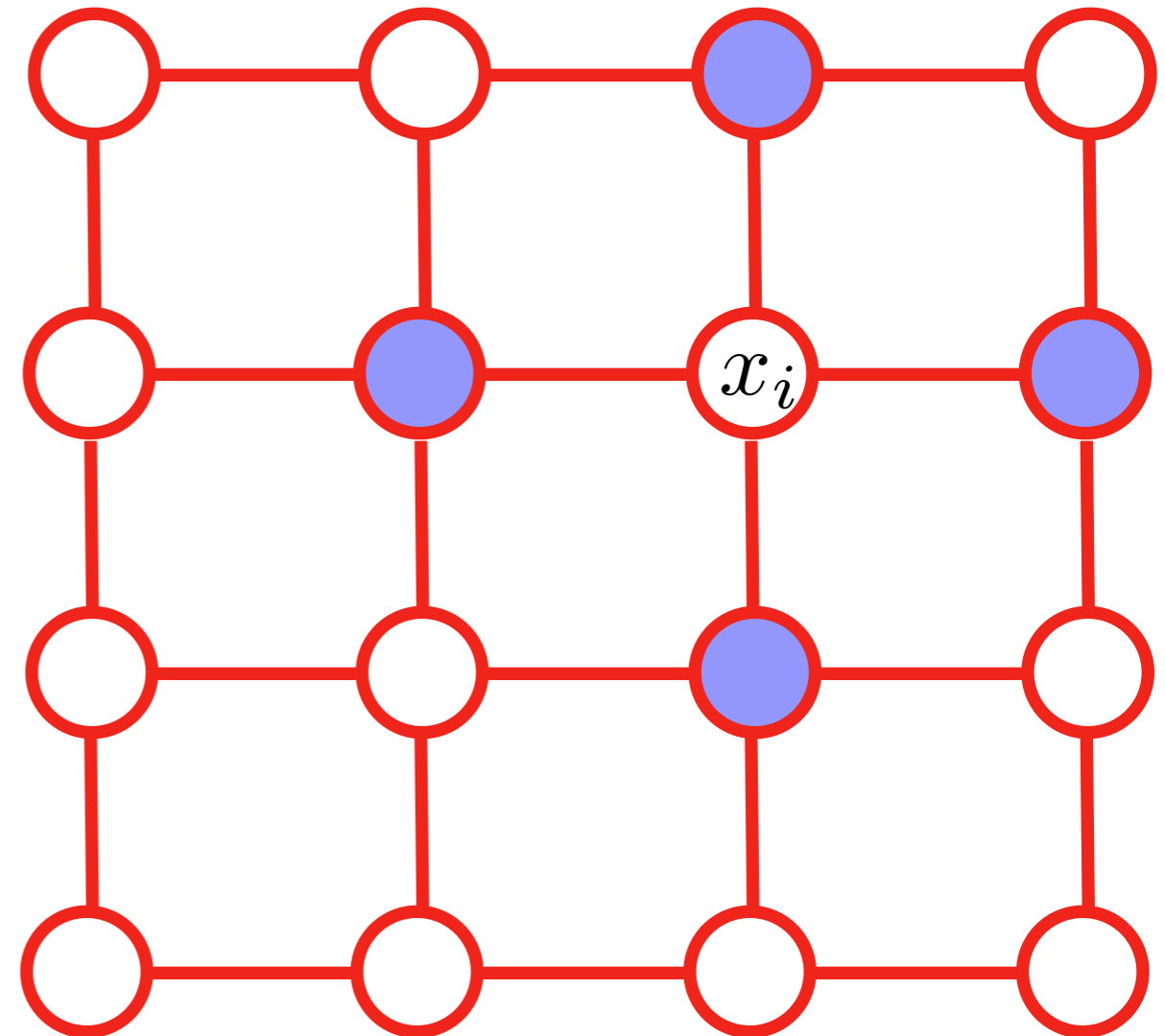
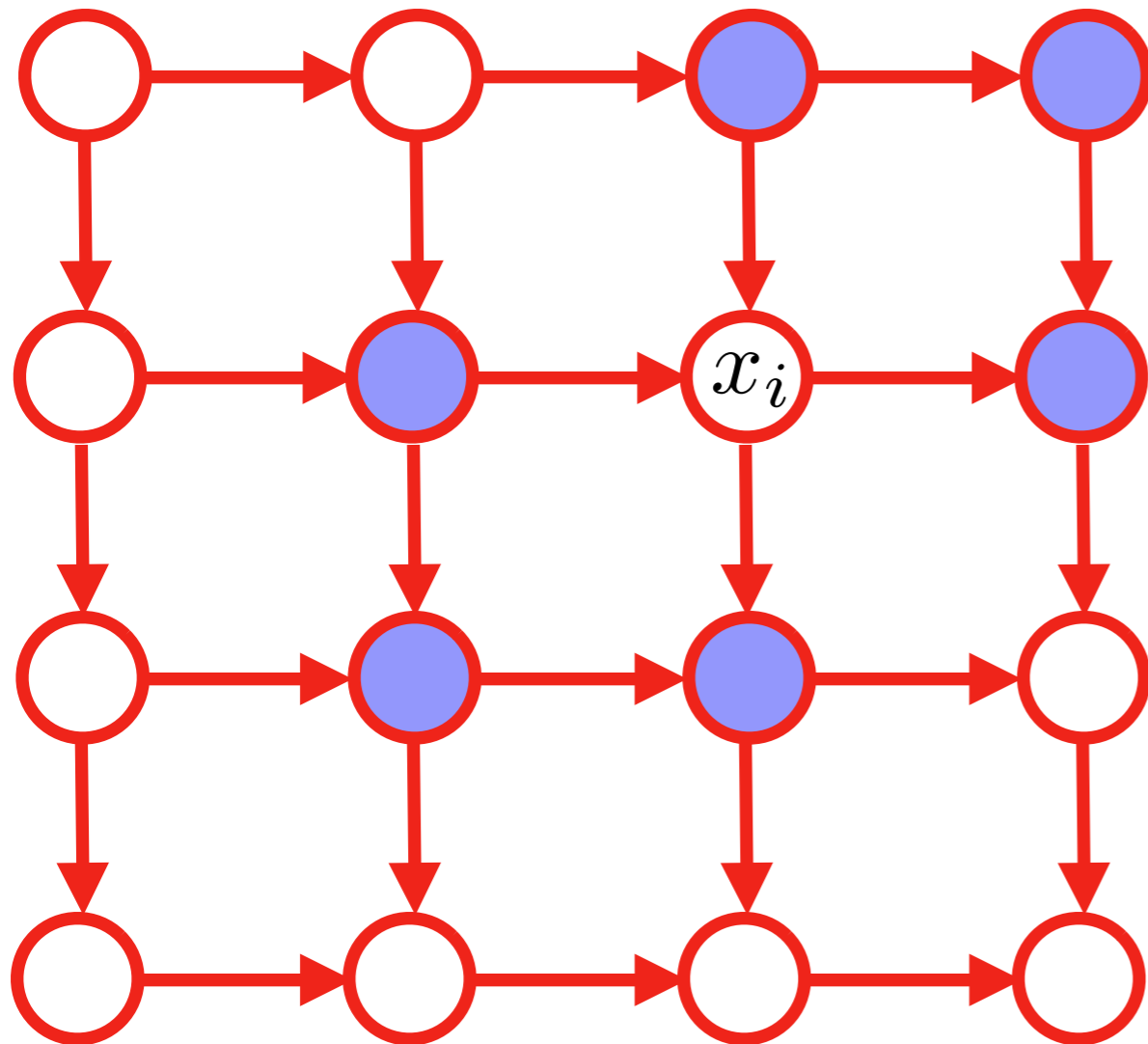
Using D-separation we can identify conditional independencies in directed graphical models, but:

- Is there a simpler, more intuitive way to express conditional independence in a graph?
- Can we find a representation for cases where an „ordering“ of the random variables is inappropriate (e.g. the pixels in a camera image)?

**Yes, we can:** by removing the directions of the edges we obtain an Undirected Graphical Model, also known as a **Markov Random Field**



# Example: Camera Image



- directions are counter-intuitive for images
- Markov blanket is not just the direct neighbors when using a directed model



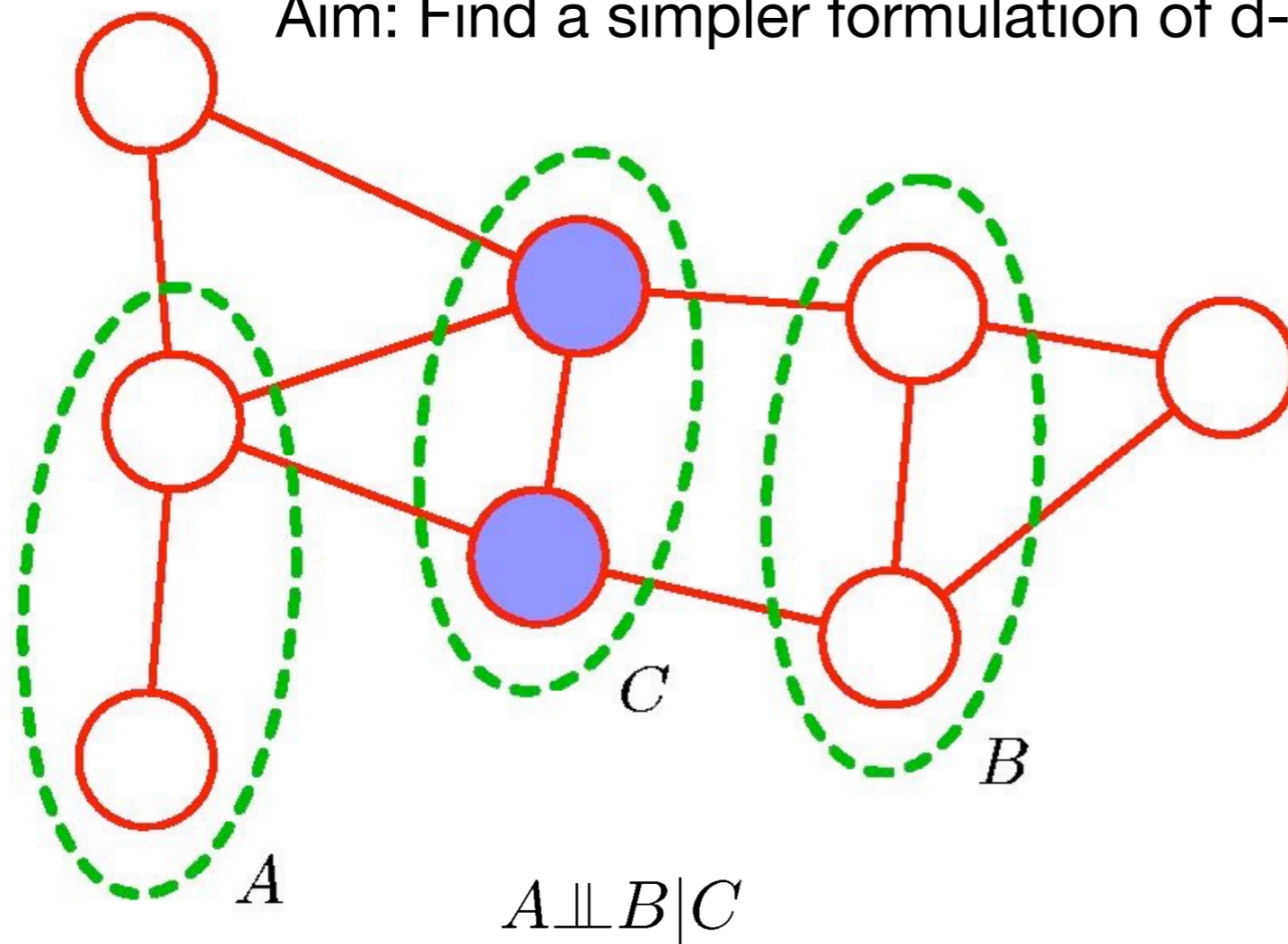
# Markov Random Fields

Aim: Find a simpler formulation of d-separation!



# Markov Random Fields

Aim: Find a simpler formulation of d-separation!

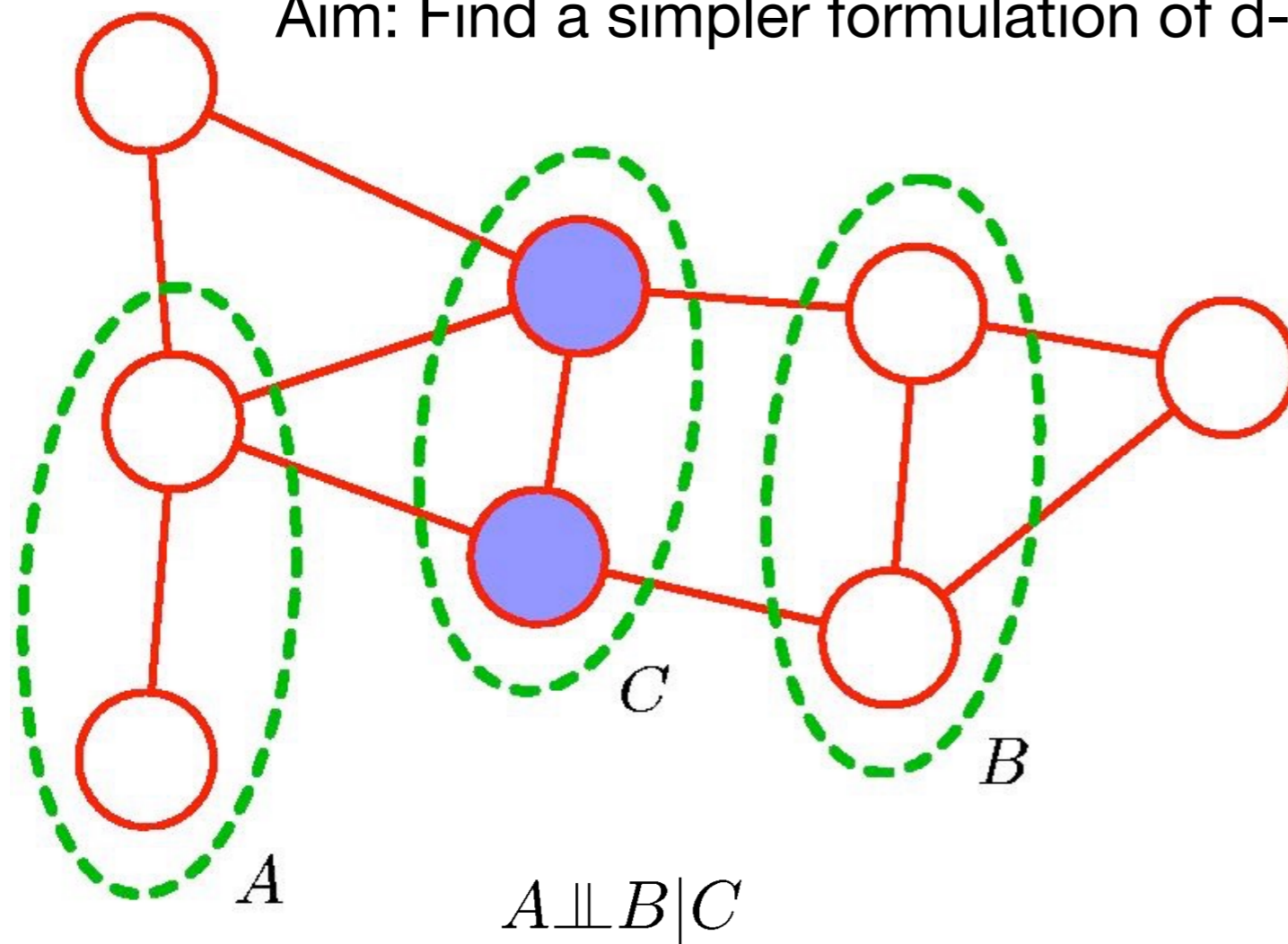


All paths from  $A$  to  $B$  go through  $C$ , i.e.  $C$  should block all paths.



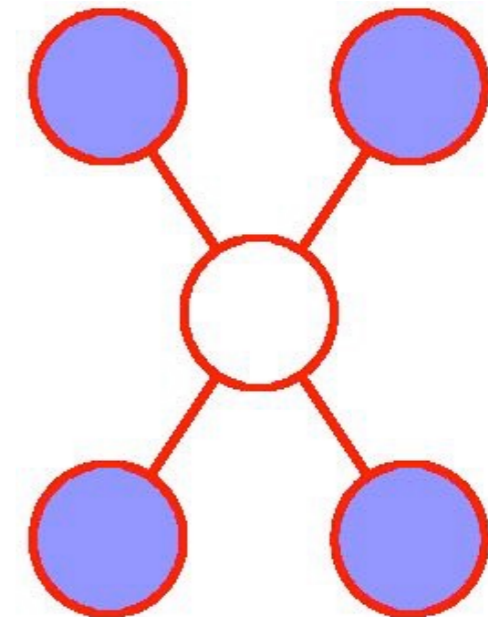
# Markov Random Fields

Aim: Find a simpler formulation of d-separation!



All paths from  $A$  to  $B$  go through  $C$ , i.e.  $C$  should block all paths.

Markov Blanket



We only need to condition on the **direct neighbors** of  $x$  to get c.i., because these already block every path from  $x$  to any other node.



# Factorization of MRFs

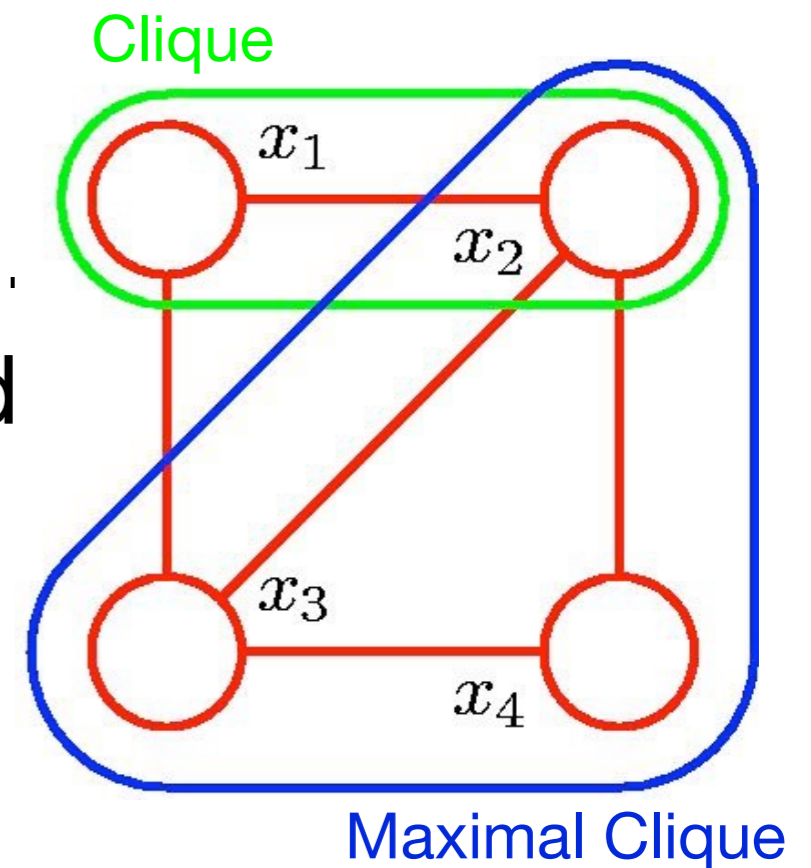
Any two nodes  $x_i$  and  $x_j$  that are not connected in an MRF are conditionally independent given all other nodes:

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

This means: **each factor must contain only nodes that are pairwise connected**

This motivates the consideration of cliques in the graph:

- A **clique** is a fully connected subgraph.
- A **maximal** clique can not be extended with another node without loosing the property of full connectivity.





# Factorization of MRFs

In general, a Markov Random Field is factorized as

$$p(\mathbf{x}) = \frac{\prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c)}{\sum_{\mathbf{x}'} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}'_c)} = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{x}_c) \quad (4.1)$$

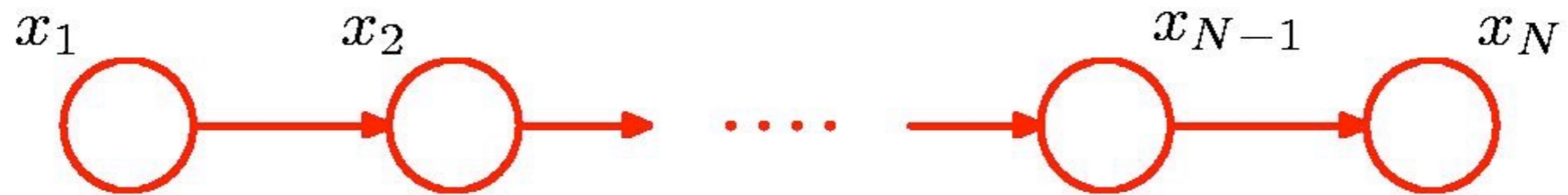
where  $\mathcal{C}$  is the set of all (maximal) cliques and  $\psi_c(\mathbf{x}_c)$  is a positive function of a given clique  $\mathbf{x}_c$  of nodes, called the **clique potential**.  $Z$  is called the **partition function**.

**Theorem (Hammersley/Clifford):** Any undirected model with associated clique potentials  $\psi_c$  is a perfect map for the probability distribution defined by Equation (4.1).

As a conclusion, all probability distributions that can be factorized as in (4.1), can be represented as an MRF.

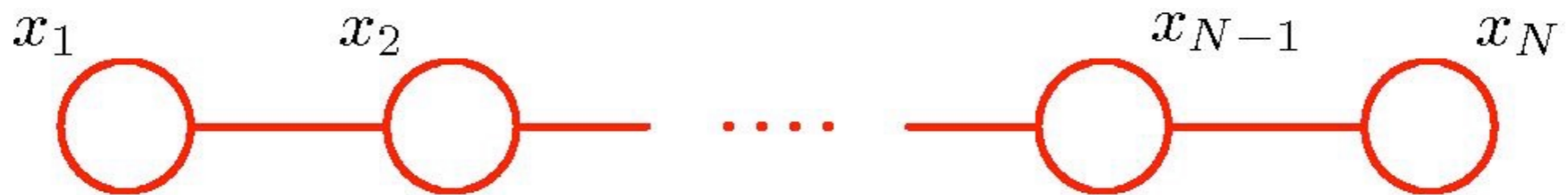


# Converting Directed to Undirected Graphs (1)



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

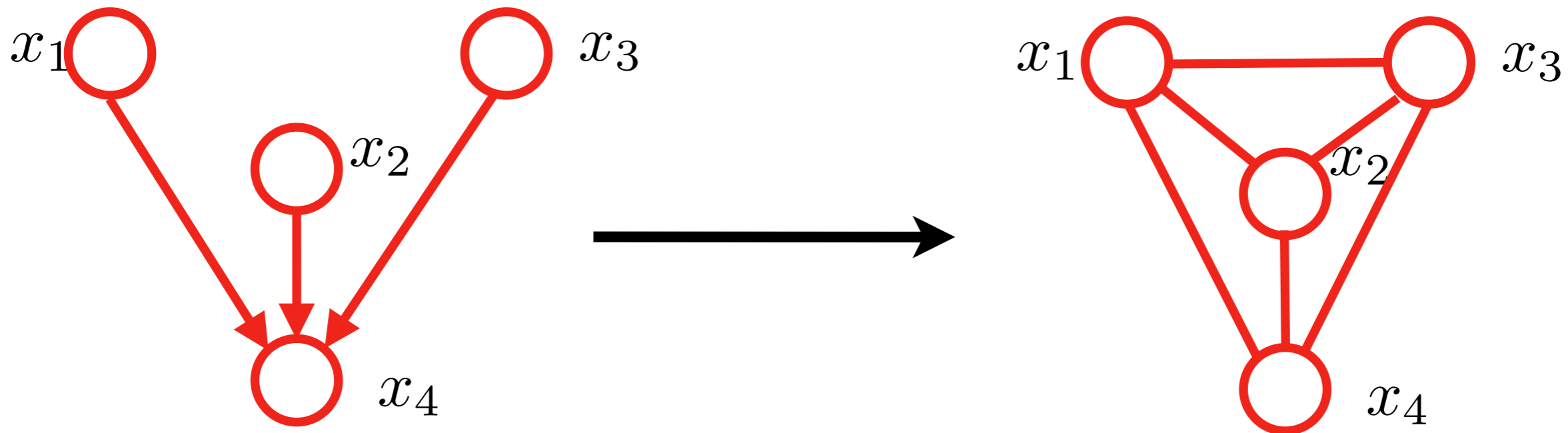
$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)$$



In this case:  $Z=1$



# Converting Directed to Undirected Graphs (2)



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

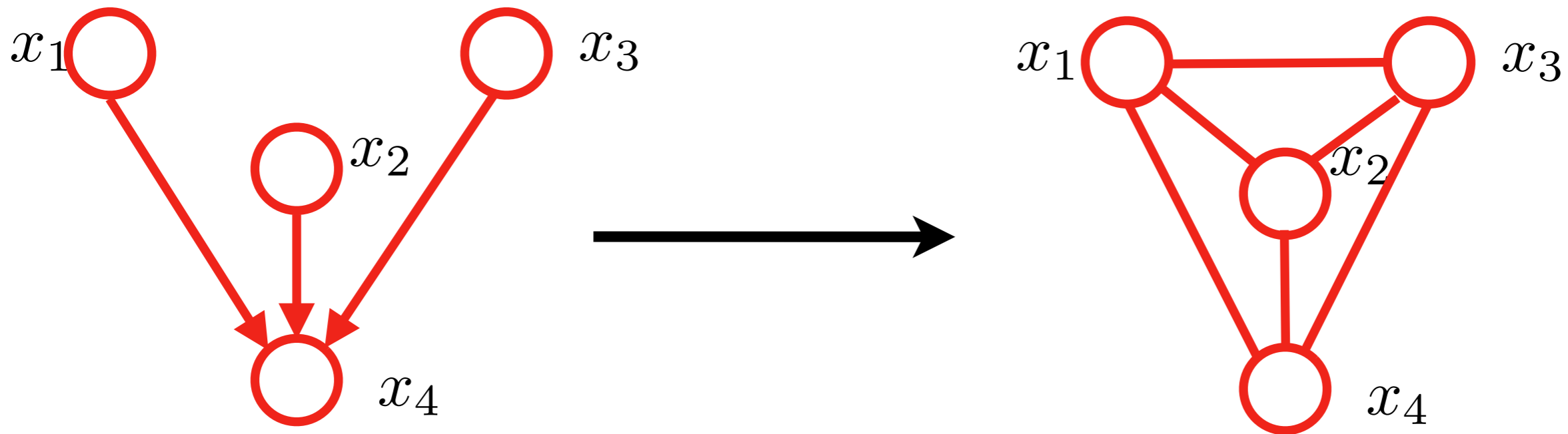
**In general:** conditional distributions in the directed graph are mapped to cliques in the undirected graph

**However:** the variables are **not** conditionally independent given the head-to-head node

Therefore: Connect all parents of head-to-head nodes with each other (**moralization**)



# Converting Directed to Undirected Graphs (2)



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

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

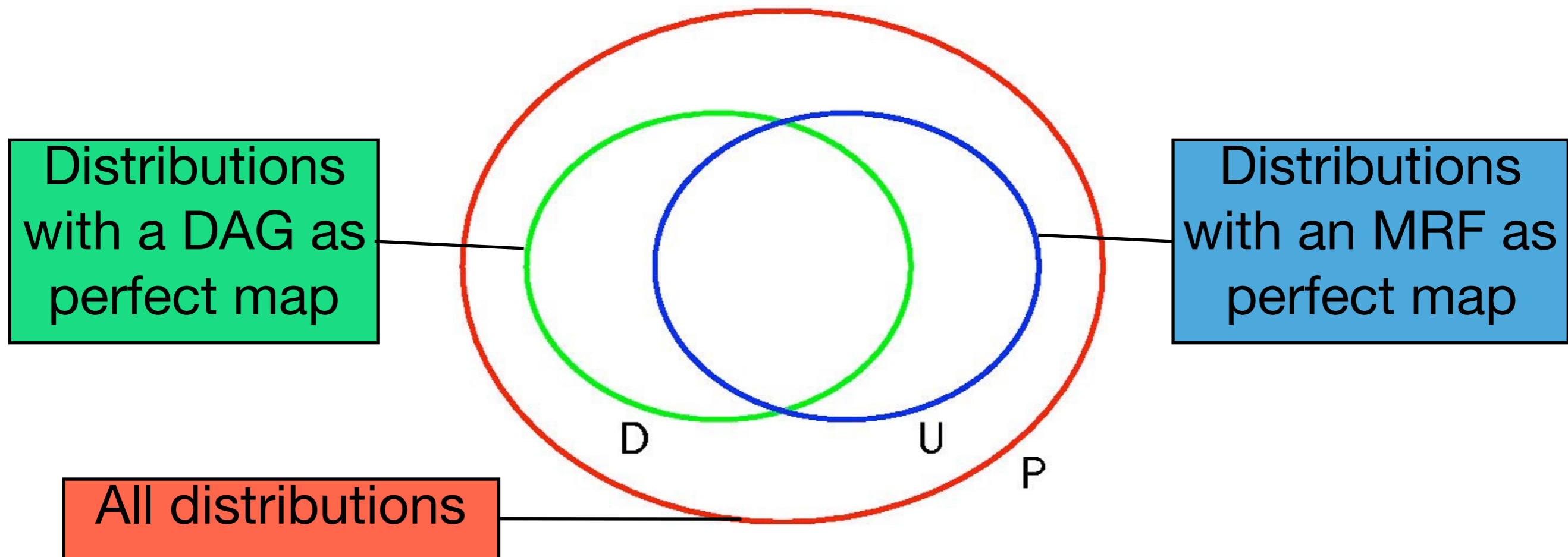
**Problem:** This process can remove conditional independence relations (inefficient)

**Generally:** There is no one-to-one mapping between the distributions represented by directed and by undirected graphs.

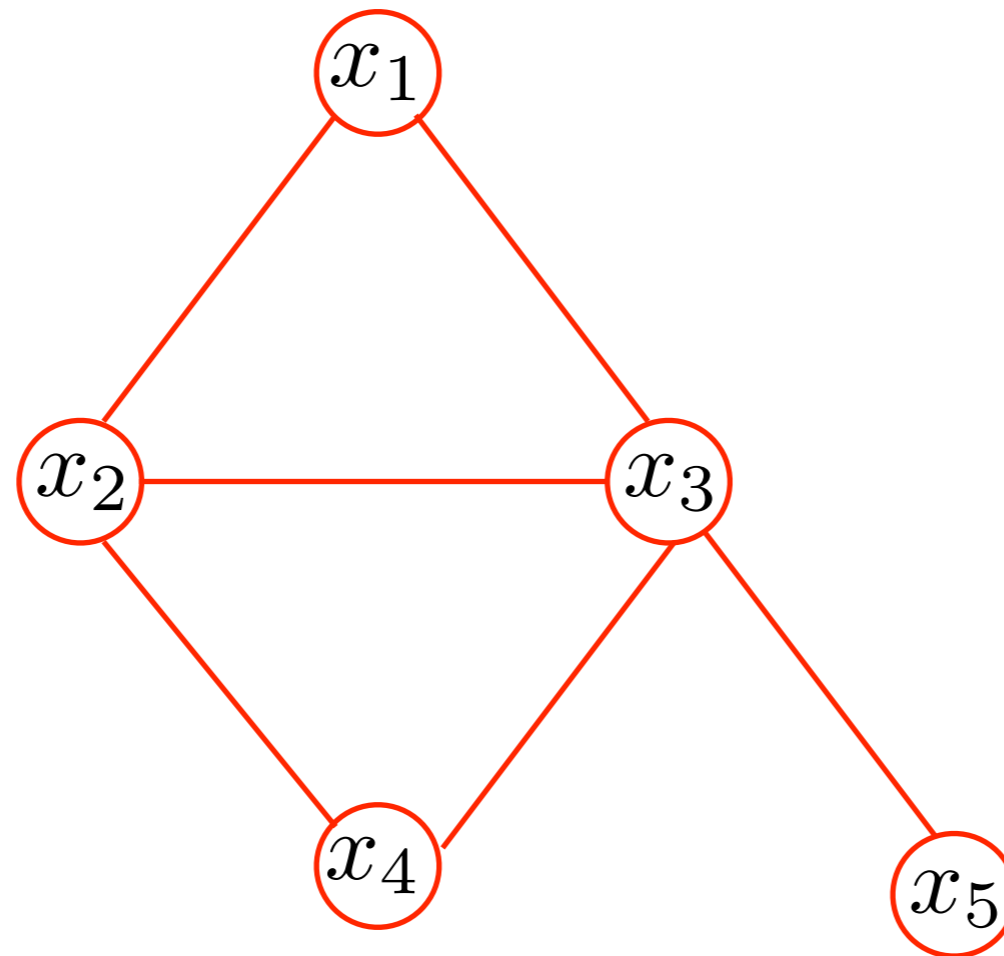


# Representability

- As for DAGs, we can define an I-map, a D-map and a perfect map for MRFs.
- The set of all distributions for which a DAG exists that is a perfect map is different from that for MRFs.



# A Simple Example

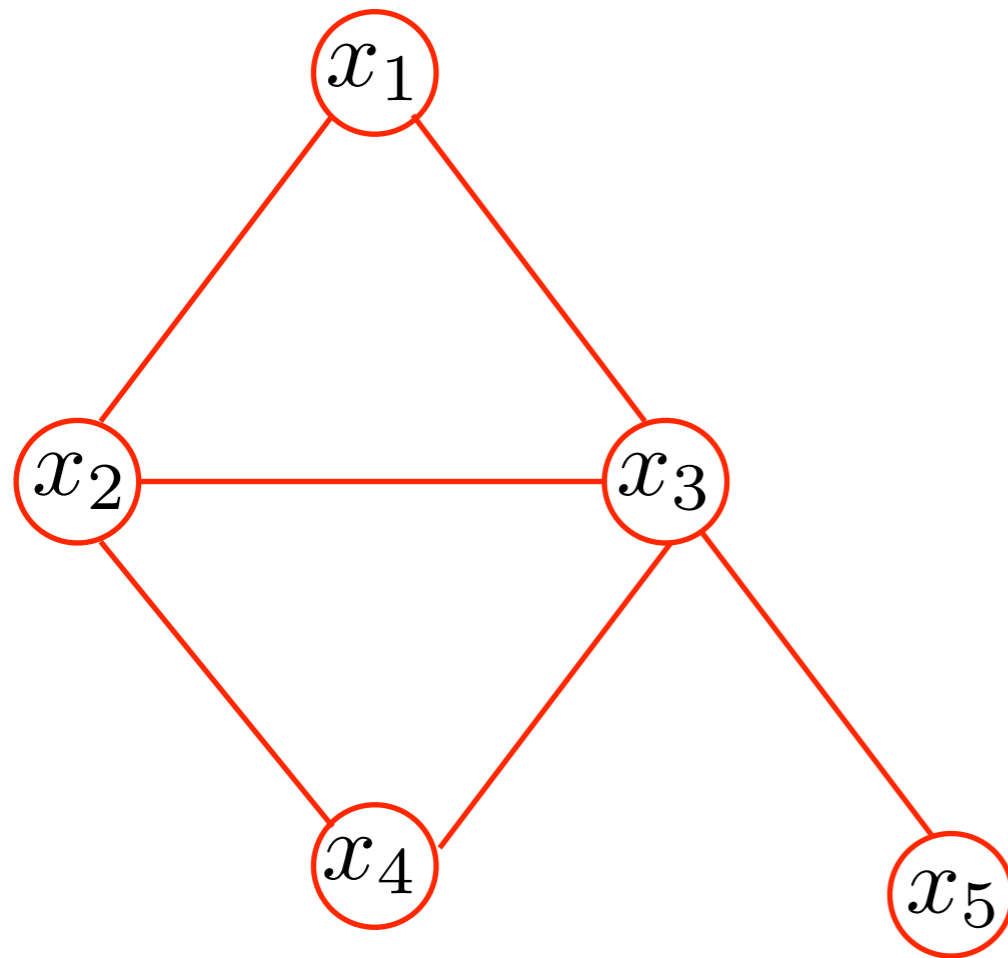


If a distribution  $p$  satisfies all conditional independence relationships of this graph, then we can write  $p$  as

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{123}(x_1, x_2, x_3) \psi_{234}(x_2, x_3, x_4) \psi_{35}(x_3, x_5)$$



# A Simple Example



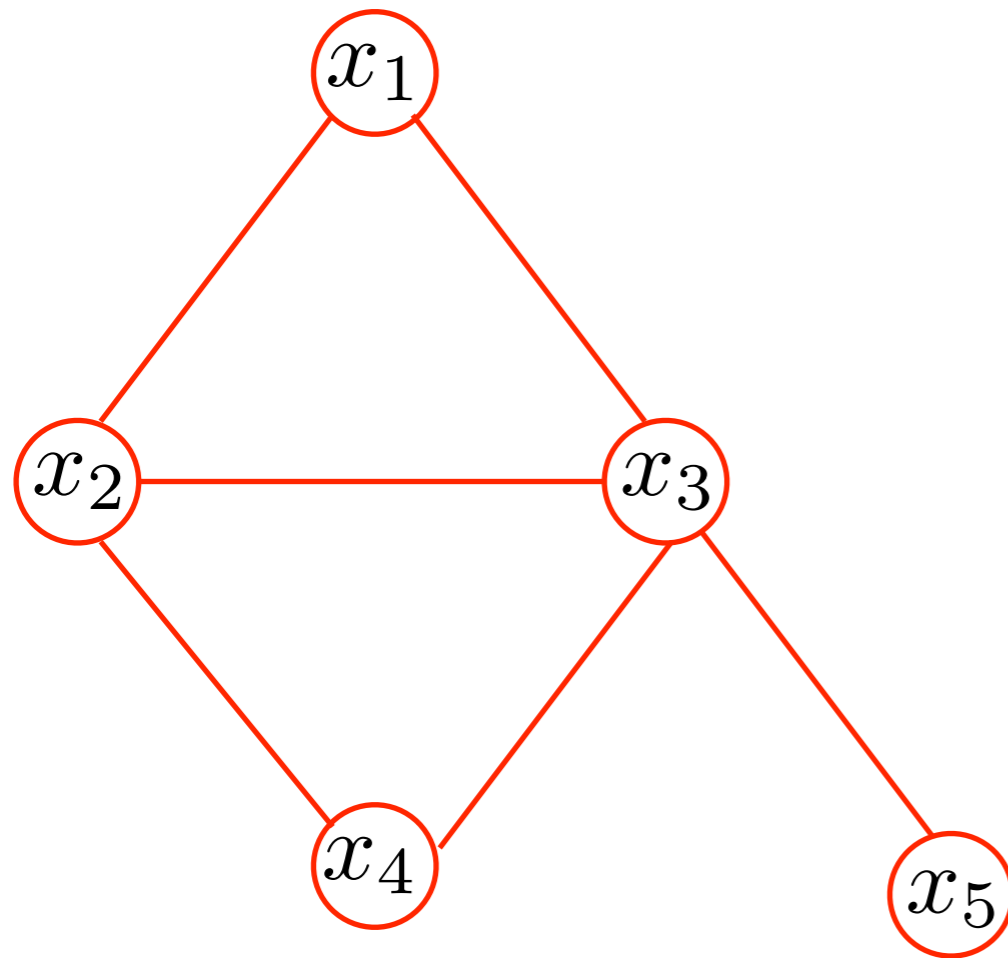
## How to define the potentials?

- Intuitively, the potential of a clique should be high, iff the joint probability of the corresponding random variables is high.

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{123}(x_1, x_2, x_3) \psi_{234}(x_2, x_3, x_4) \psi_{35}(x_3, x_5)$$



# A Simple Example



## How to define the potentials?

- Intuitively, the potential of a clique should be high, iff the joint probability of the corresponding random variables is high.
- In most cases the potential is defined using a **log-linear** model:

$$\log \psi_c(\mathbf{x}_c) = \phi_c(\mathbf{x}_c)^T \boldsymbol{\theta}$$

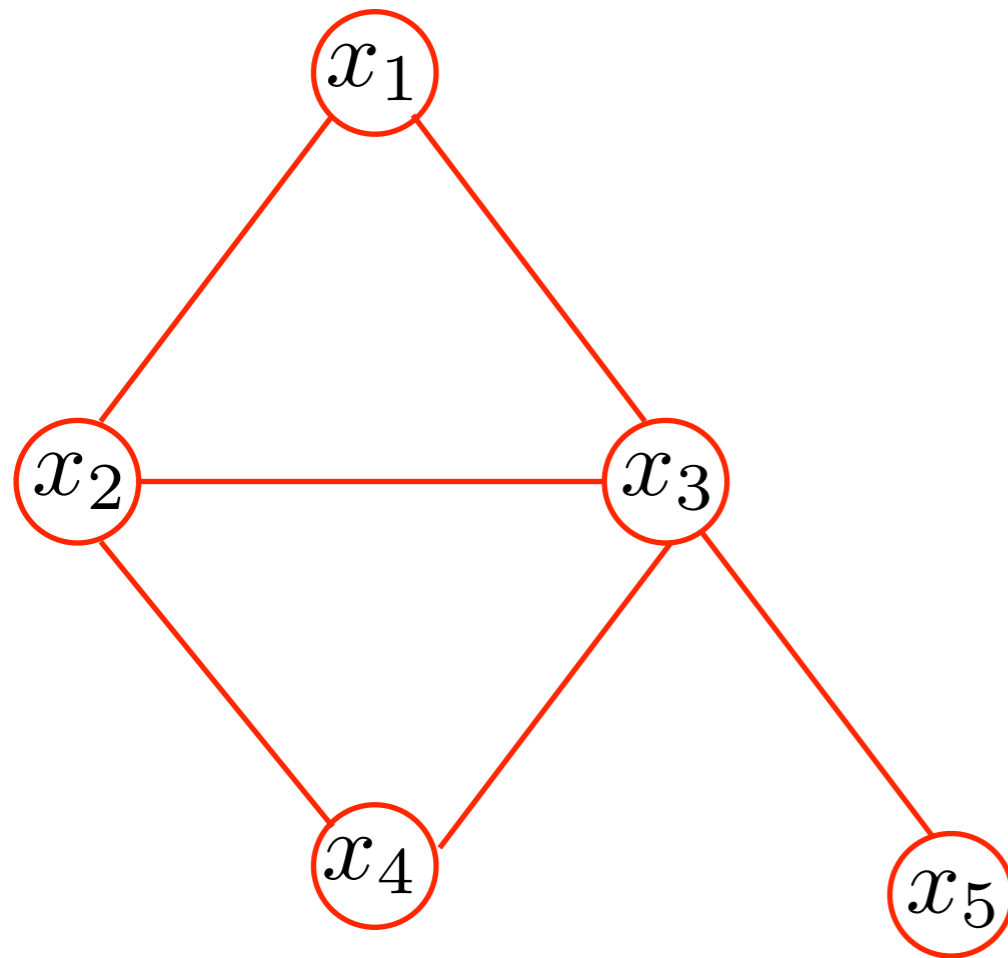
Feature function

Parameters





# A Simple Example



## How to define the potentials?

- Intuitively, the potential of a clique should be high, iff the joint probability of the corresponding random variables is high.
- In most cases the potential is defined using a **log-linear** model

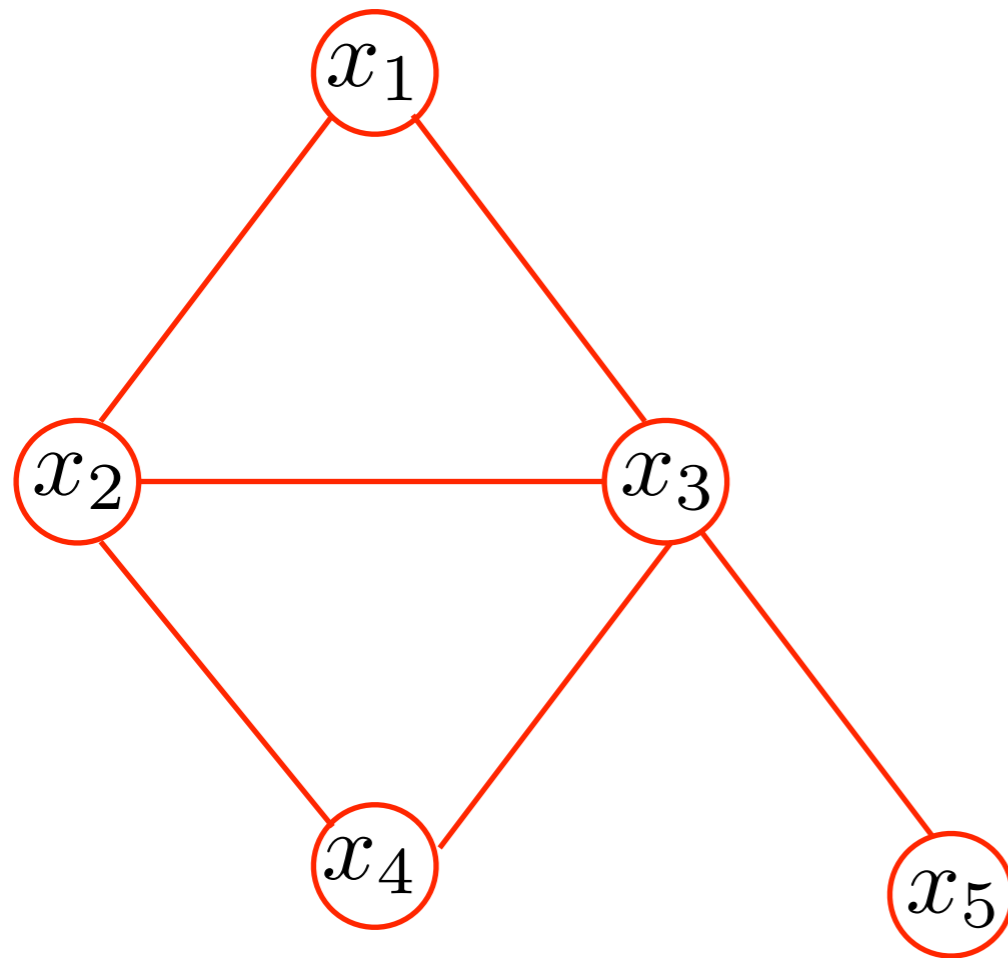
$$\log \psi_c(\mathbf{x}_c) = \phi_c(\mathbf{x}_c)^T \boldsymbol{\theta}$$

making the parameters explicit:

$$\Rightarrow \log p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)^T \boldsymbol{\theta} - \log Z(\boldsymbol{\theta})$$



# A Simple Example



**Using numbers, e.g.:**

- Let all variables be binary:

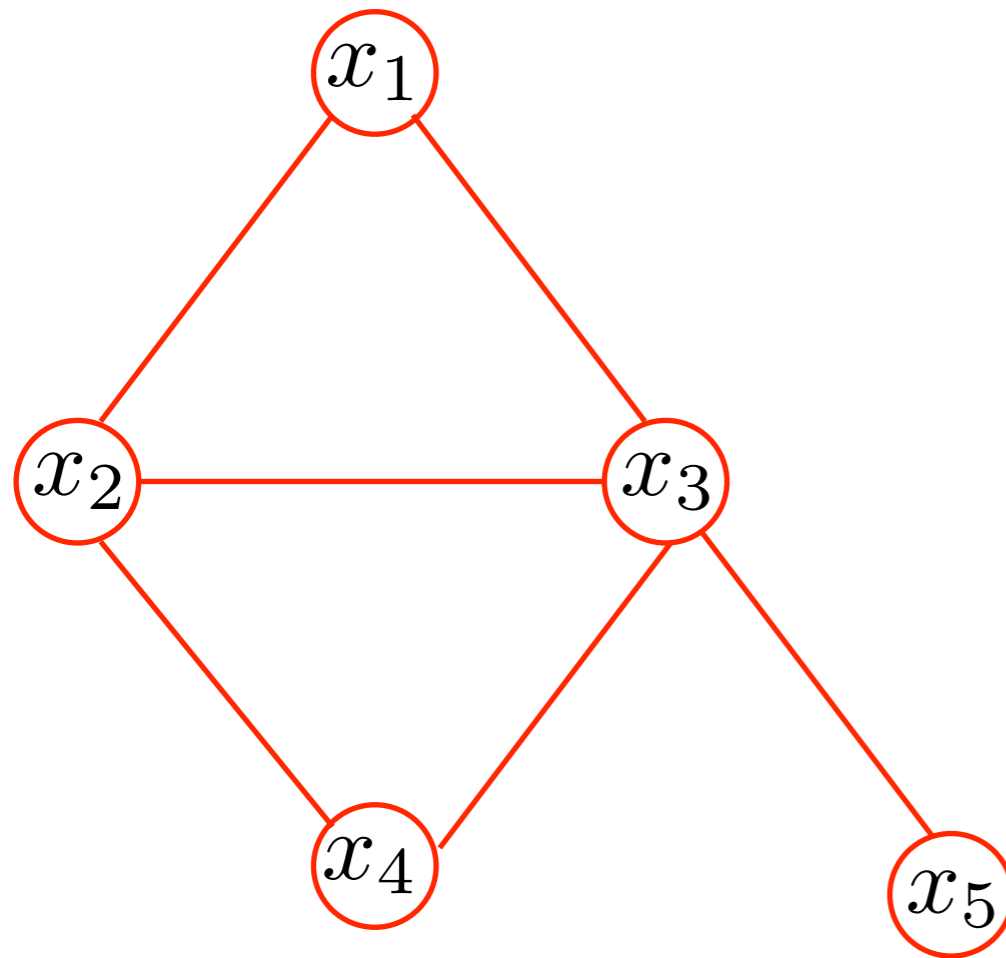
$$x_i \in \{0, 1\}$$

- We can define **features**  $\phi$

$$\phi_{ijk}(x_i, x_j, x_k) = \begin{cases} 1 & \text{if } x_n = 1 \ \forall n \in \{i, j, k\} \\ 0 & \text{otherwise} \end{cases}$$



# A Simple Example



Using numbers, e.g.:

- Let all variables be binary:

$$x_i \in \{0, 1\}$$

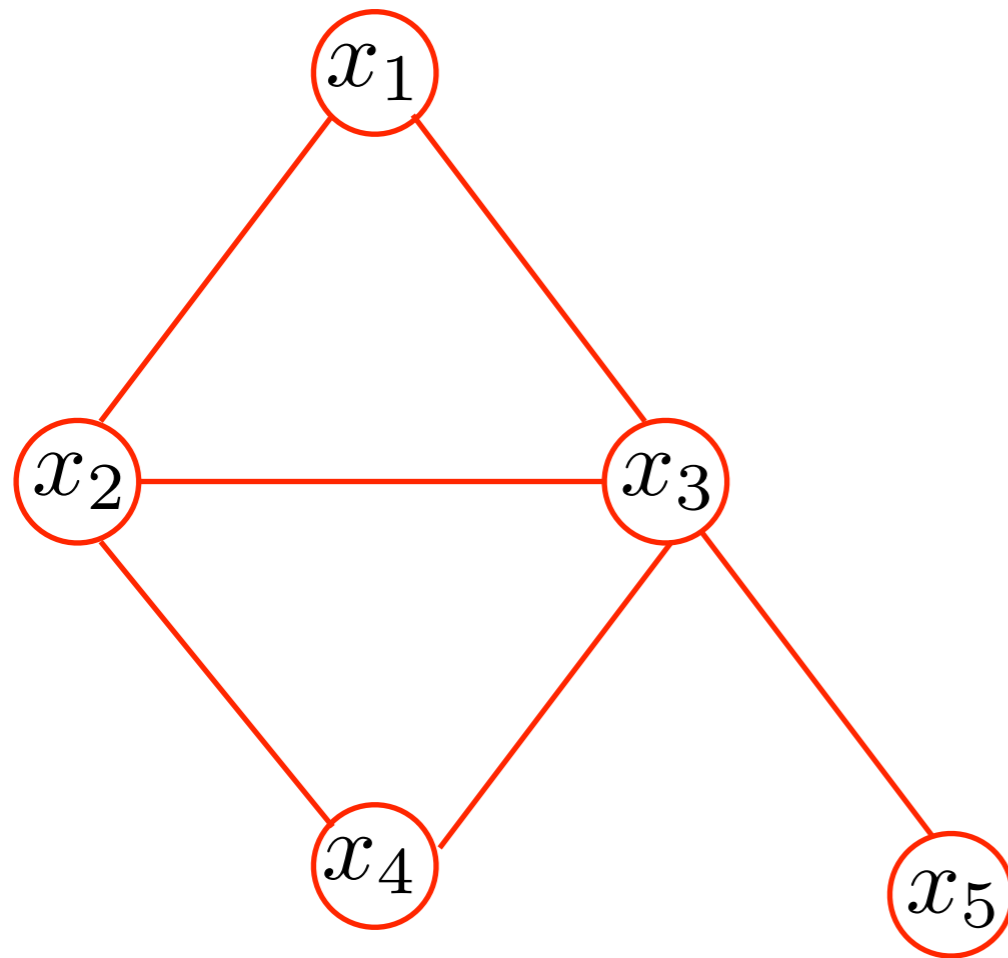
- We can define **features**  $\phi$
- and determine **weights**  $\theta$

$$\phi_{ijk}(x_i, x_j, x_k) = \begin{cases} 1 & \text{if } x_n = 1 \forall n \in \{i, j, k\} \\ 0 & \text{otherwise} \end{cases}$$

$$\theta = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1)^T$$



# A Simple Example



Using numbers, e.g.:

- Let all variables be binary:

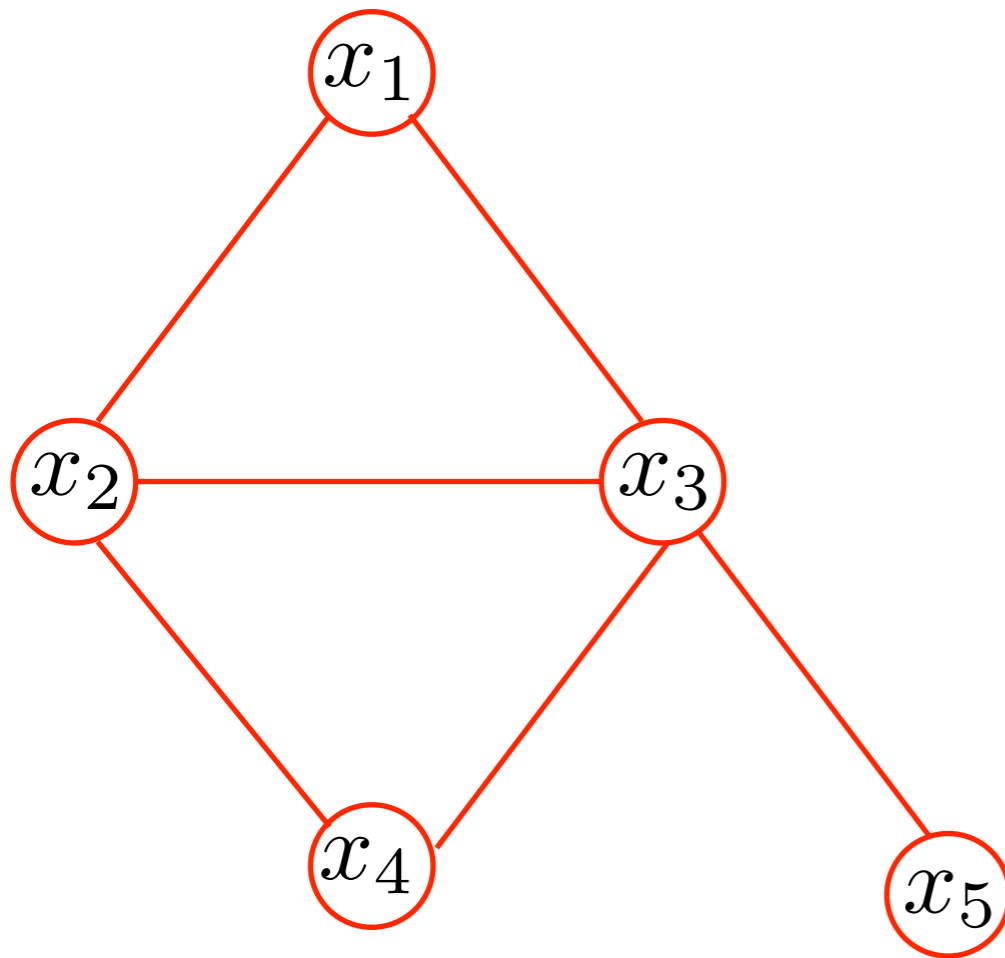
$$x_i \in \{0, 1\}$$

- We can define **features**  $\phi$
- and determine **weights**  $\theta$
- Then, we can compute the (log of the ) joint probability for each realisation of the  $x_i$

$$\log p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)^T \boldsymbol{\theta} - \log Z(\boldsymbol{\theta})$$



# A Simple Example



**Using numbers, e.g.:**

- The same graph can also be interpreted as a **binary** MRF
- This a more specific representation, but it is less complex (and therefore more efficient)
- In Computer Vision, we almost always use **binary** MRFs; they are a specific case of general MRFs:

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{12}(x_1, x_2) \psi_{13}(x_1, x_3) \psi_{23}(x_2, x_3) \psi_{24}(x_2, x_4) \psi_{34}(x_3, x_4) \psi_{35}(x_3, x_5)$$



# Using Graphical Models

We can use a graphical model to do **inference**:

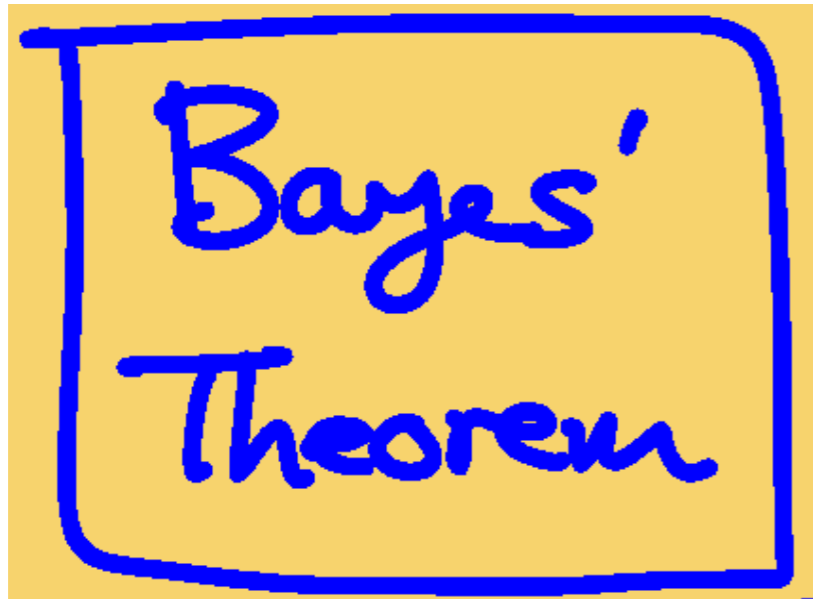
- We want to find  $\arg \max_{\mathbf{x}} p(\mathbf{x})$
- Some nodes in the graph are **observed**, for others we want to find the posterior distribution
- Also, computing the local **marginal distribution**  $p(x_n)$  at any node  $x_n$  can be done using inference.

Question: How can inference be done with a graphical model?

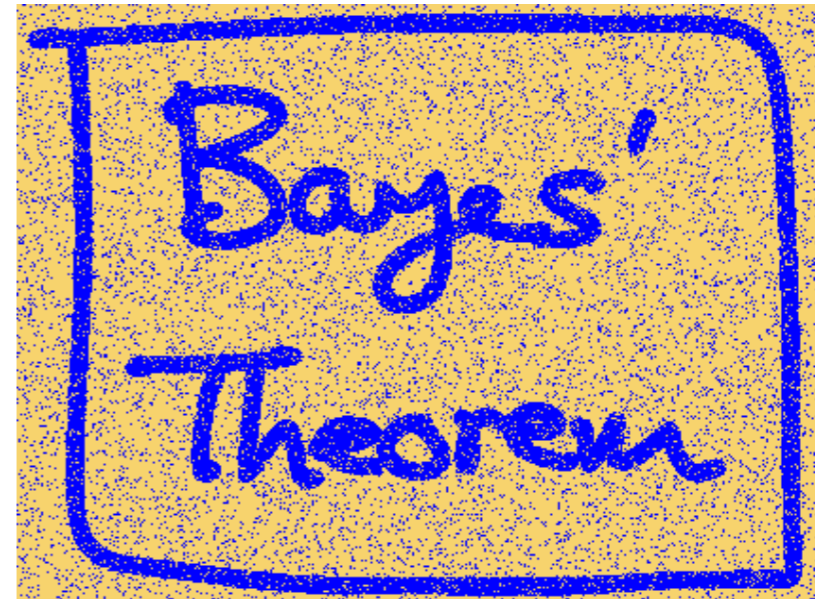
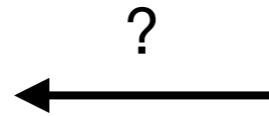
We will see that, when exploiting conditional independences, we can do efficient inference.



# Example Application: Denoising



Noise-free image



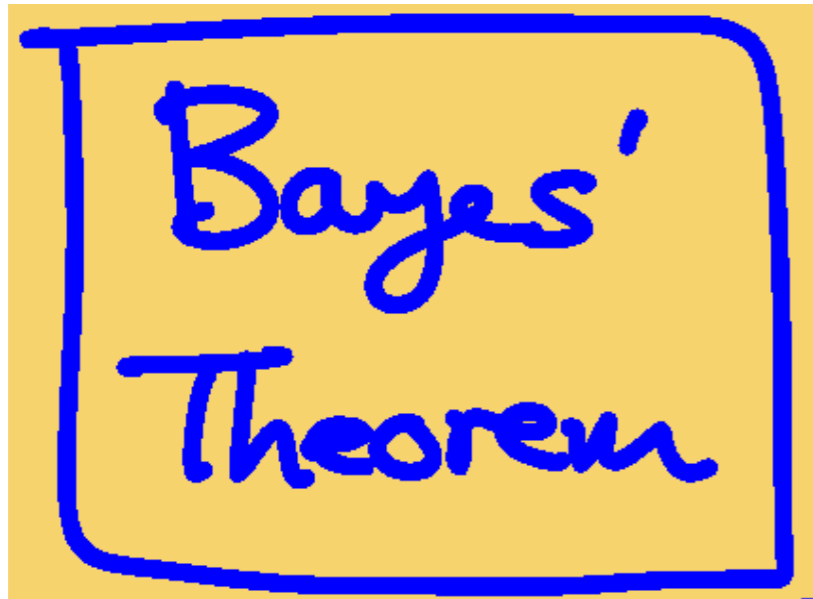
Noisy image (observation)

Aim: Recover the noise-free image from the noisy one

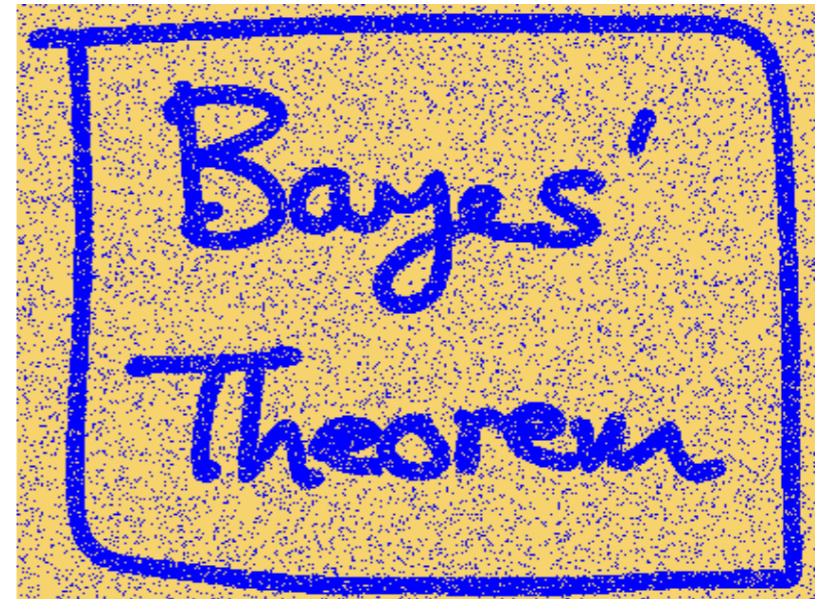
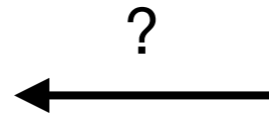
We model the original image with variables  $x_i \in \{-1, 1\}$  and the noisy image with pixel values  $y_i \in \{-1, 1\}$



# Example Application: Denoising



Noise-free image



Noisy image (observation)

Aim: Recover the noise-free image from the noisy one

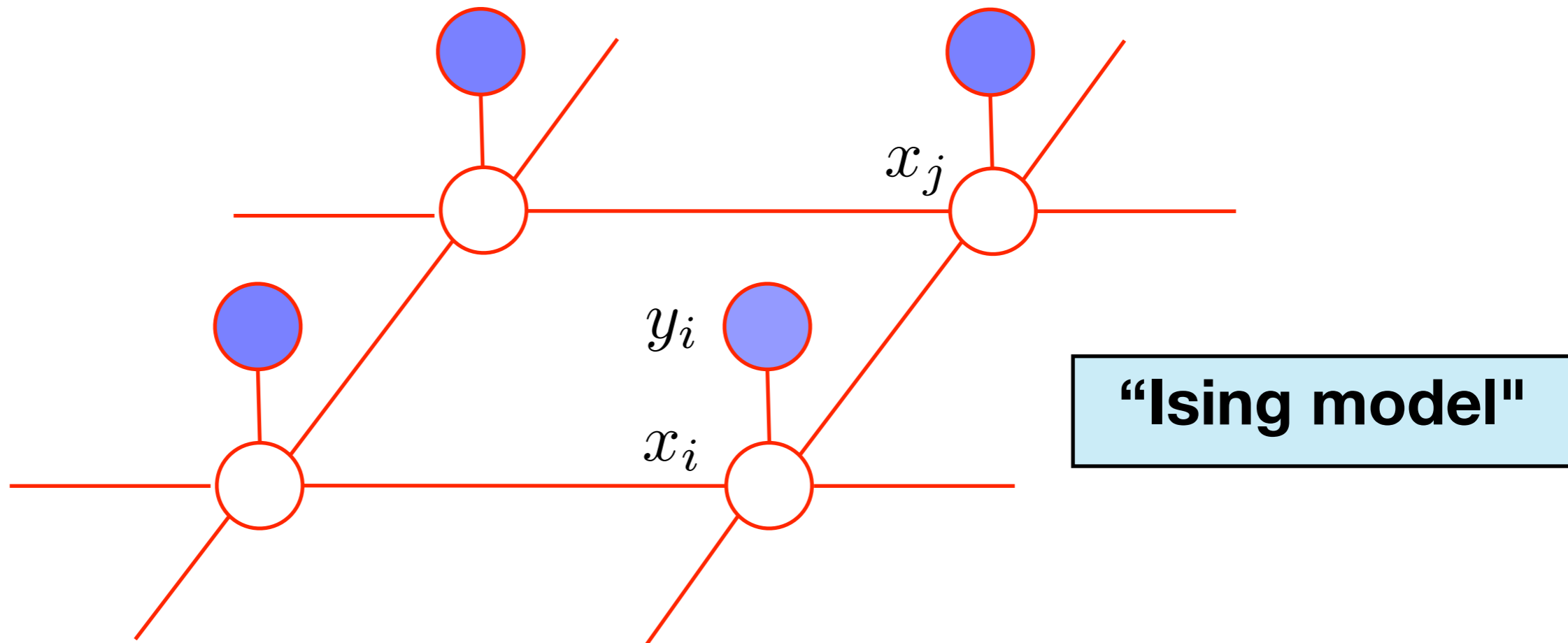
We model the original image with variables  $x_i \in \{-1, 1\}$  and the noisy image with pixel values  $y_i \in \{-1, 1\}$

We consider the true pixel values as **hidden** or **latent**





# Example Application: Denoising



We define two simple edge features:

$$\phi(x_i, y_i) = x_i y_i$$

$$\phi(x_i, x_j) = x_i x_j$$

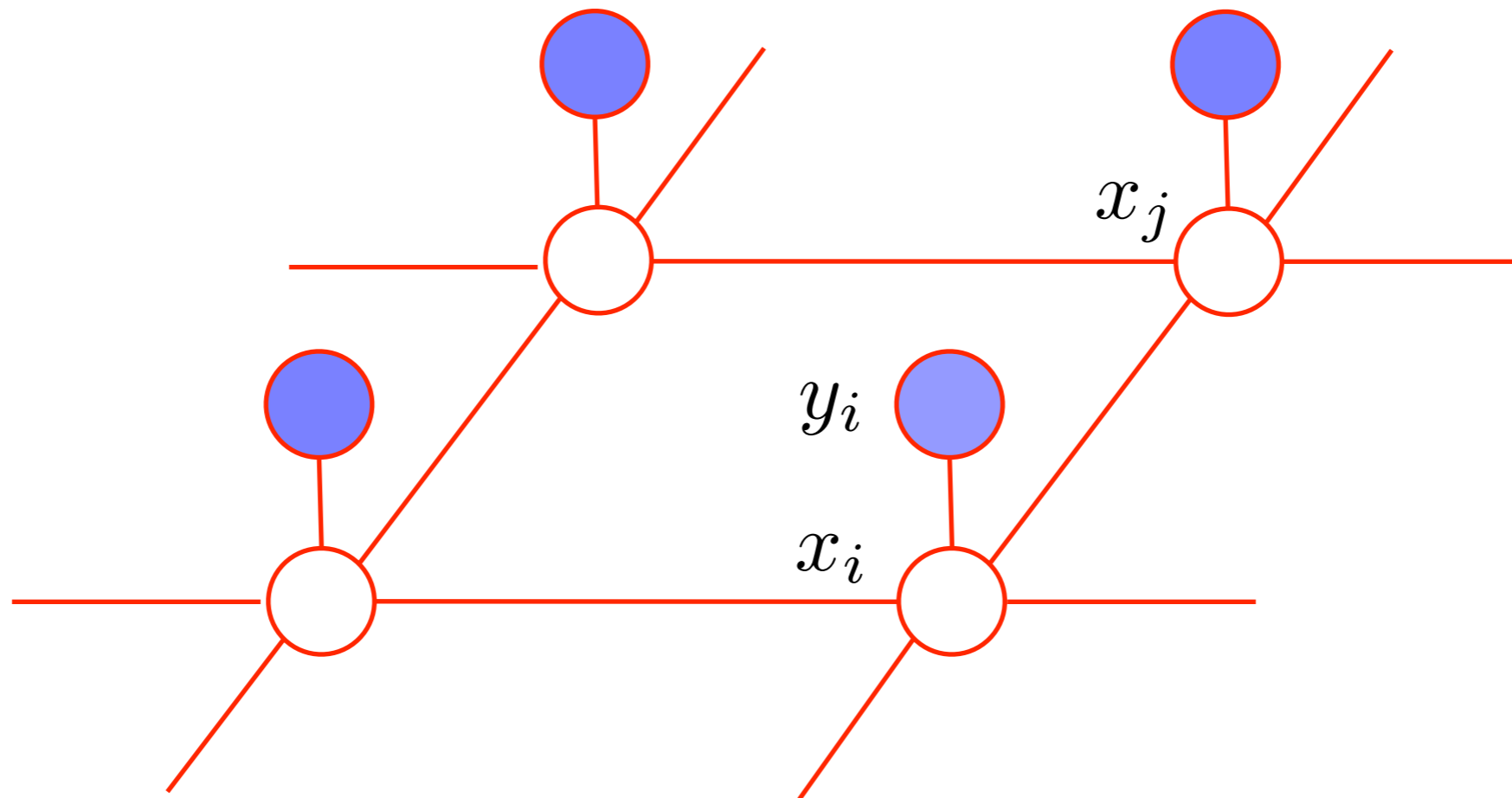
These are multiplied by parameters  $\beta$  and  $\eta$ :

$$\log \psi(x_i, y_i) = \eta x_i y_i$$

$$\log \psi(x_i, x_j) = \beta x_i x_j$$



# Example Application: Denoising



With this, we can compute the joint:

$$p(\mathbf{x}, \mathbf{y} \mid \eta, \beta) = \frac{1}{Z} \prod_i \exp(\eta x_i y_i) \prod_{i,j} \exp(\beta x_i x_j)$$

and its log:

$$\log p(\mathbf{x}, \mathbf{y} \mid \eta, \beta) = \eta \sum_i x_i y_i + \beta \sum_{i,j} x_i x_j - \log(Z)$$



# Example Application: Denoising

$$\log p(\mathbf{x}, \mathbf{y} \mid \eta, \beta) = \eta \sum_i x_i y_i + \beta \sum_{i,j} x_i x_j - \log(Z)$$

Our aim now is to find the hidden states  $x_i$  such that this log of the joint is maximal (or at least very high).

Simple approach is Iterated Conditional Modes (ICM):

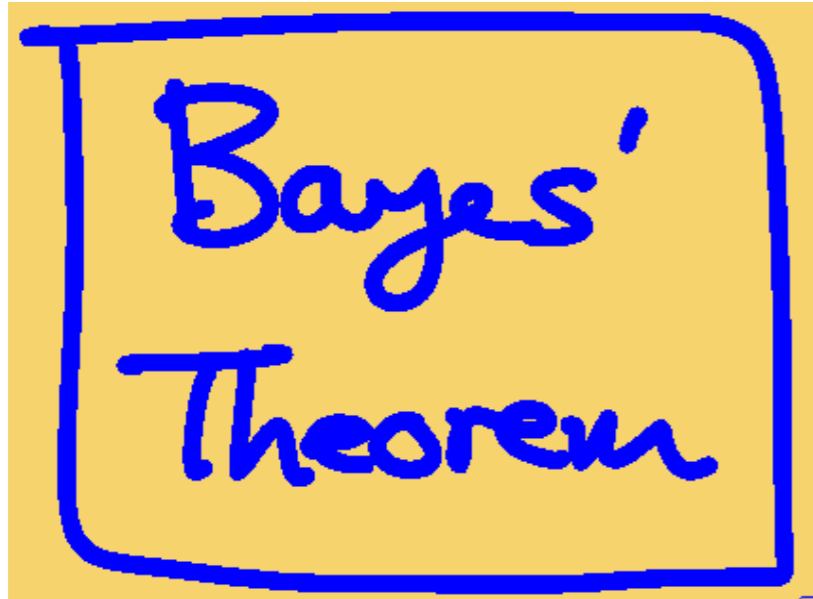
1. Initialize all  $x_i$  by corresponding  $y_i$
2. For all nodes  $x_i$  :
  1. set  $x_i$  to +1 and to -1 and evaluate  $\log p(\mathbf{x}, \mathbf{y} \mid \eta, \beta)$
  2. keep the value that gives higher log joint

This will keep or increase the joint in every step

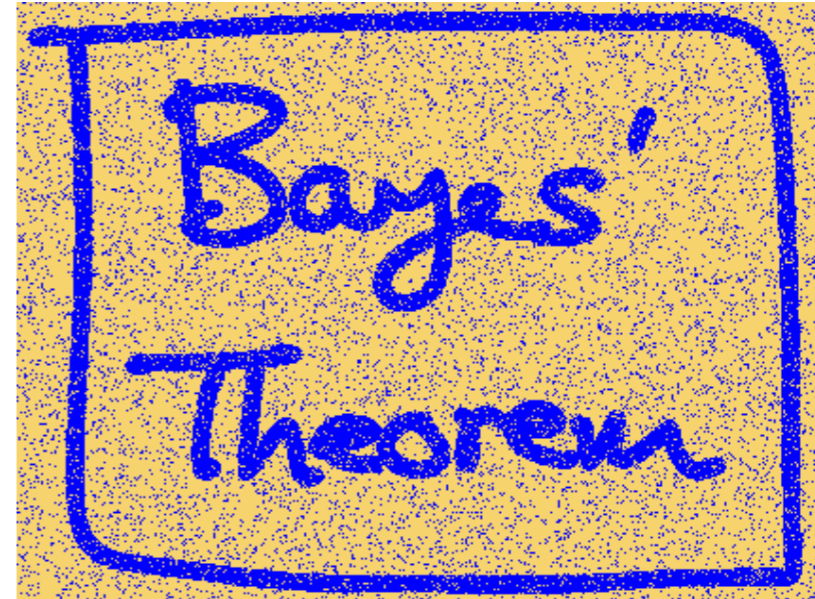
The nodes can be visited in order or randomly



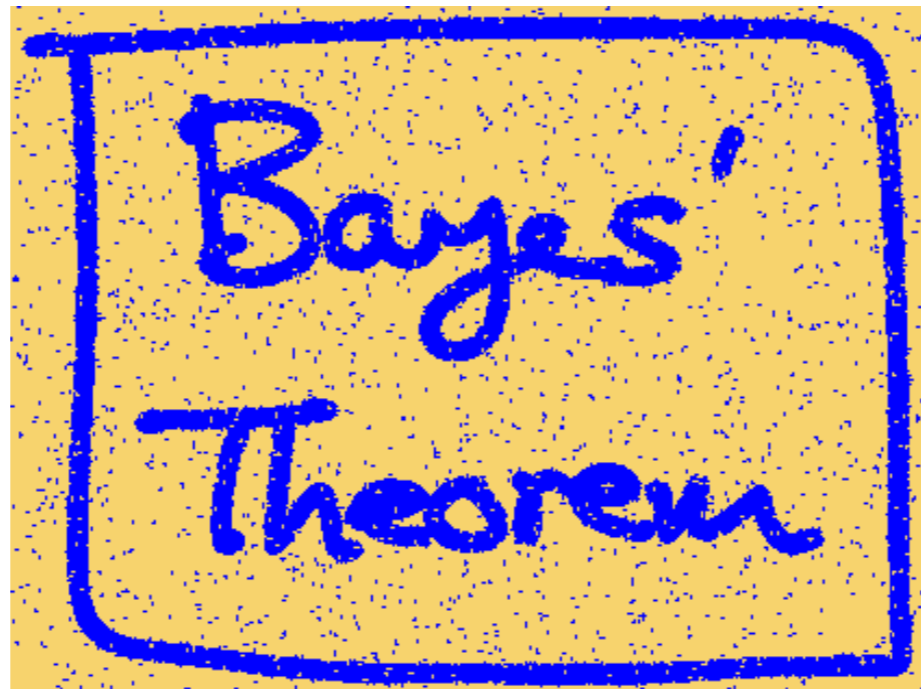
# Result of ICM



Noise-free image



Noisy image (observation)



Result of ICM

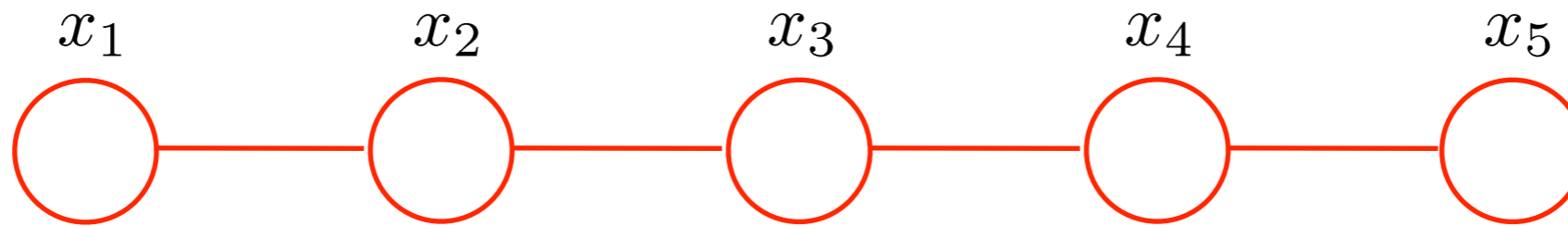


# General Inference in MRFs

- In general, we do not have such an easy model
- Therefore, we need more general inference methods for MRF
- The major aim is to exploit sparsity in the graphical model to make inference efficient



# Inference on a Chain



The joint probability is given by

$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \psi_{4,5}(x_4, x_5)$$

The marginal at  $x_3$  is  $p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$

In the general case with  $N$  nodes we have

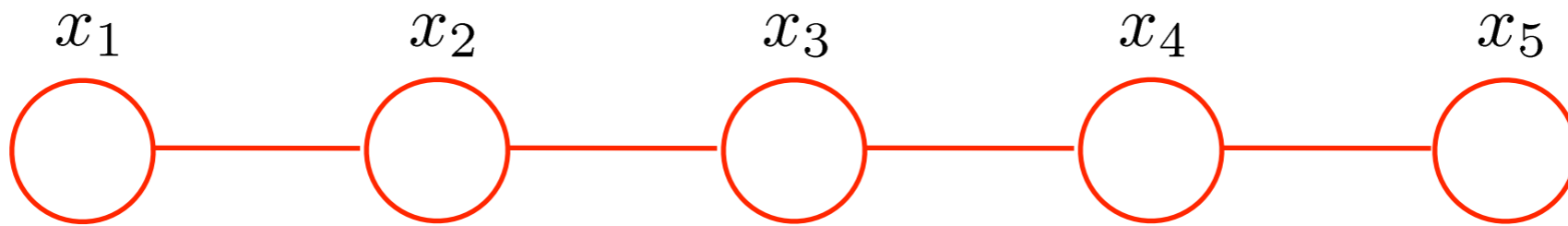
$$p(\mathbf{x}) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \cdots \psi_{N-1,N}(x_{N-1}, x_N)$$

and

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



# Inference on a Chain



$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} p(\mathbf{x})$$

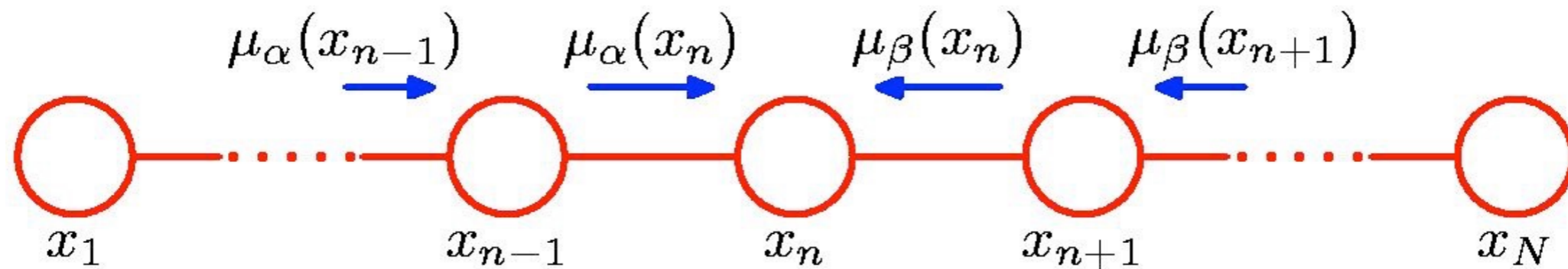
- This would mean  $K^N$  computations! A more efficient way is obtained by rearranging:

$$\begin{aligned}
 p(x_3) &= \frac{1}{Z} \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \psi_{4,5}(x_4, x_5) \\
 &= \frac{1}{Z} \sum_{x_2} \sum_{x_1} \sum_{x_4} \sum_{x_5} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \psi_{4,5}(x_4, x_5) \\
 &= \frac{1}{Z} \sum_{x_2} \psi_{2,3}(x_2, x_3) \underbrace{\sum_{x_1} \psi_{1,2}(x_1, x_2)}_{\mu_\alpha(x_3)} \underbrace{\sum_{x_4} \psi_{3,4}(x_3, x_4) \sum_{x_5} \psi_{4,5}(x_4, x_5)}_{\mu_\beta(x_3)}
 \end{aligned}$$

$\mu_\alpha(x_3) \leftarrow$  Vectors of size K  $\rightarrow \mu_\beta(x_3)$



# Inference on a Chain



In general, we have

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





# Inference on a Chain

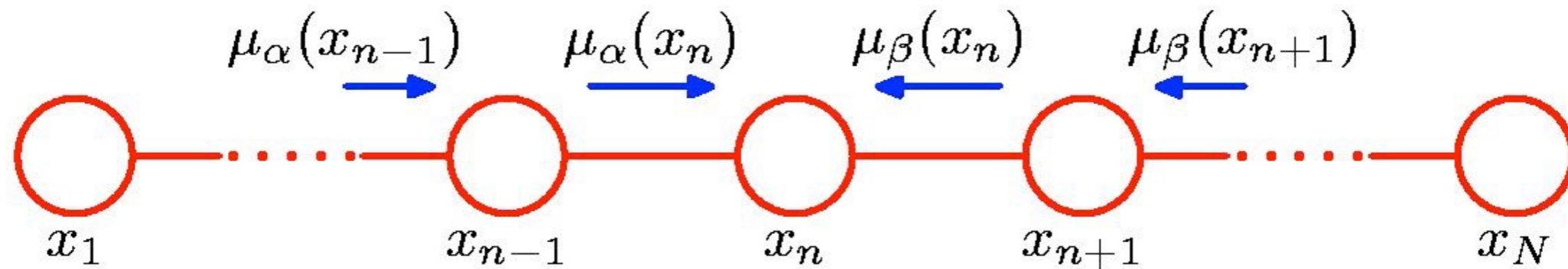
The **messages**  $\mu_\alpha$  and  $\mu_\beta$  can be computed recursively:

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

Computation of  $\mu_\alpha$  starts at the first node and computation of  $\mu_\beta$  starts at the last node.



# Inference on a Chain



- The first values of  $\mu_\alpha$  and  $\mu_\beta$  are:

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

- The partition function can be computed at any node:

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

- Overall, we have  $O(NK^2)$  operations to compute the marginal  $p(x_n)$



# 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  $Z$  **once** at a node  $x_m$ :  $Z = \sum_{x_m} \mu_\alpha(x_m) \mu_\beta(x_m)$
- Compute

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

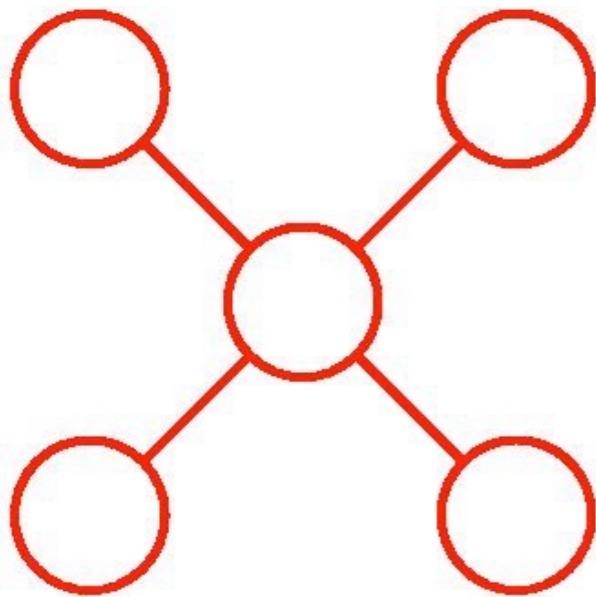
for all variables required.



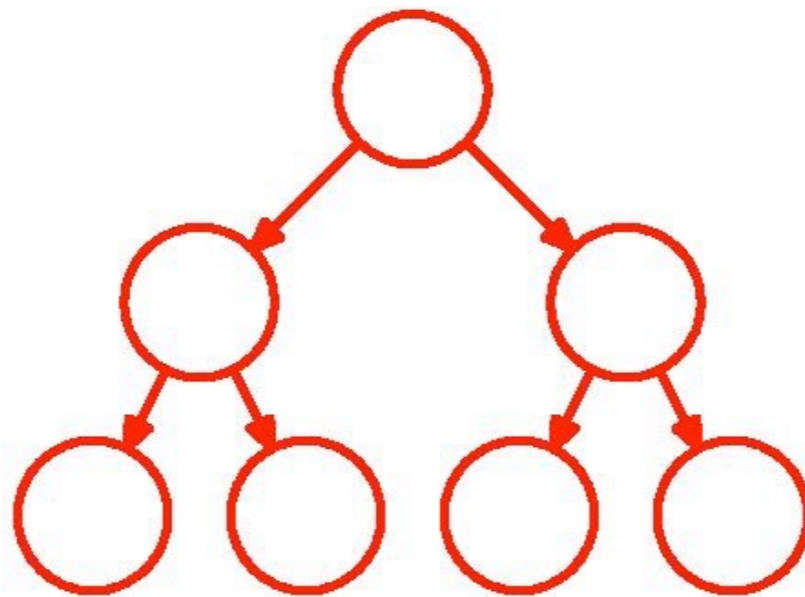
# More General Graphs

The message-passing algorithm can be extended to more general graphs:

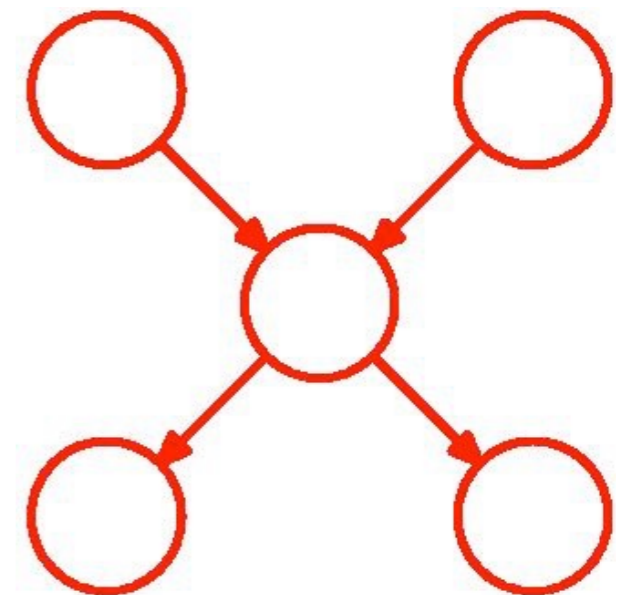
Undirected  
Tree



Directed  
Tree



Polytree



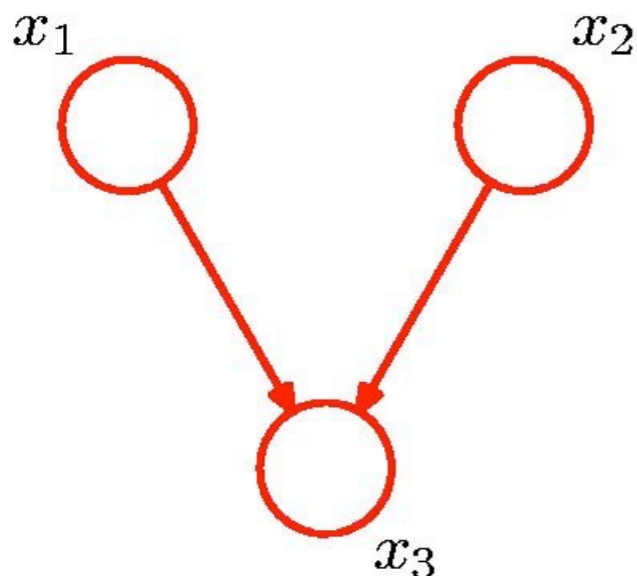
It is then known as the **sum-product algorithm**.

A special case of this is **belief propagation**.



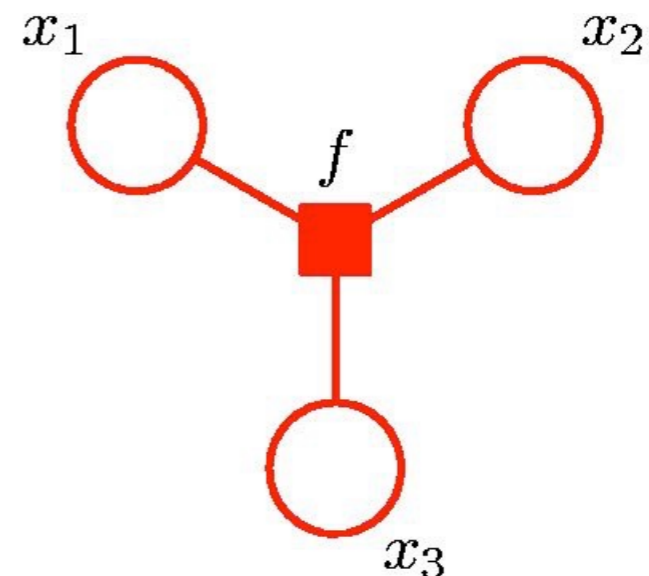
# Factor Graphs

- The Sum-product algorithm can be used to do inference on undirected and directed graphs.
- A representation that generalizes directed and undirected models is the **factor graph**.



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

Directed graph



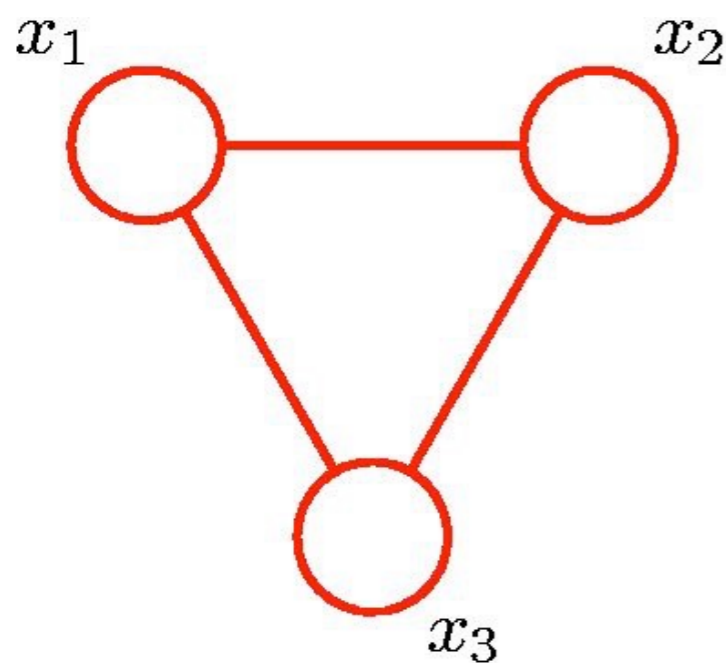
$$f(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3 | x_1, x_2)$$

Factor graph



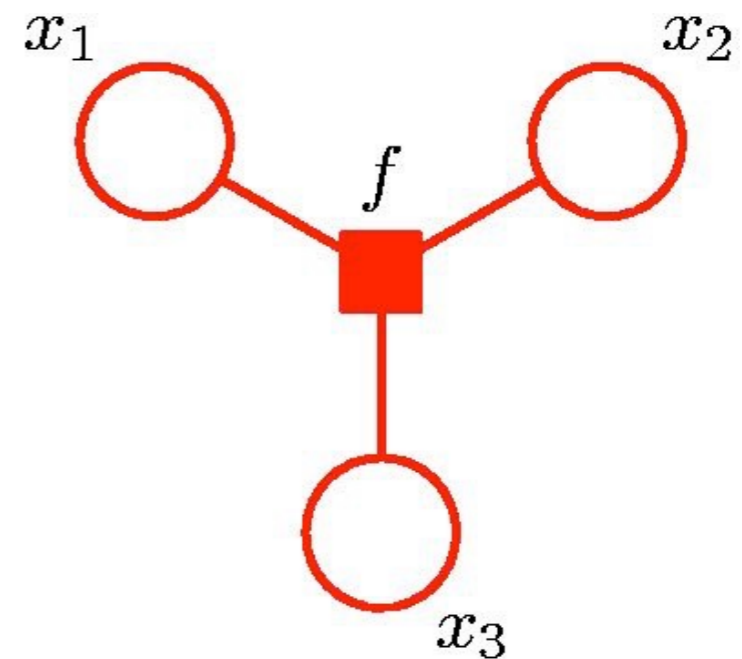
# Factor Graphs

- The Sum-product algorithm can be used to do inference on undirected and directed graphs.
- A representation that generalizes directed and undirected models is the **factor graph**.



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

Undirected graph



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

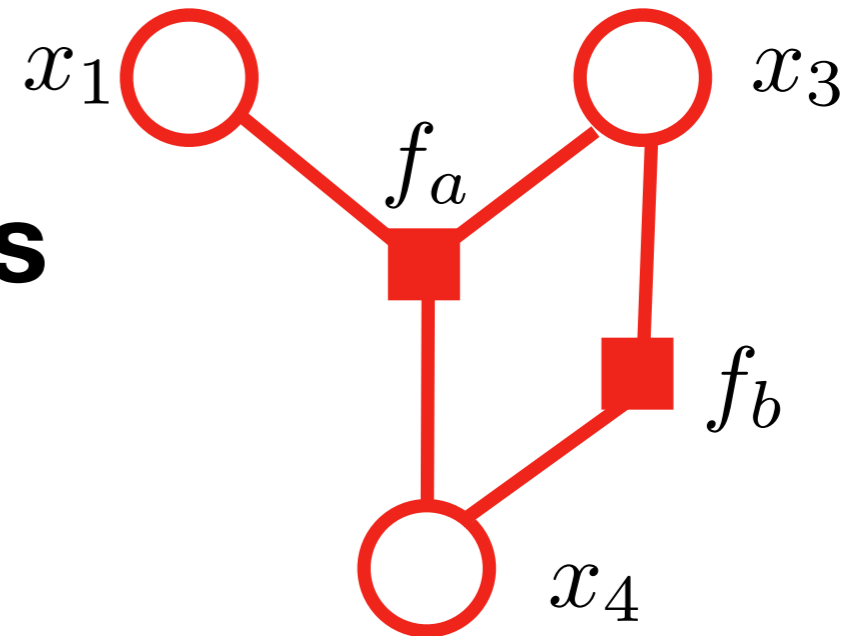
Factor graph



# Factor Graphs

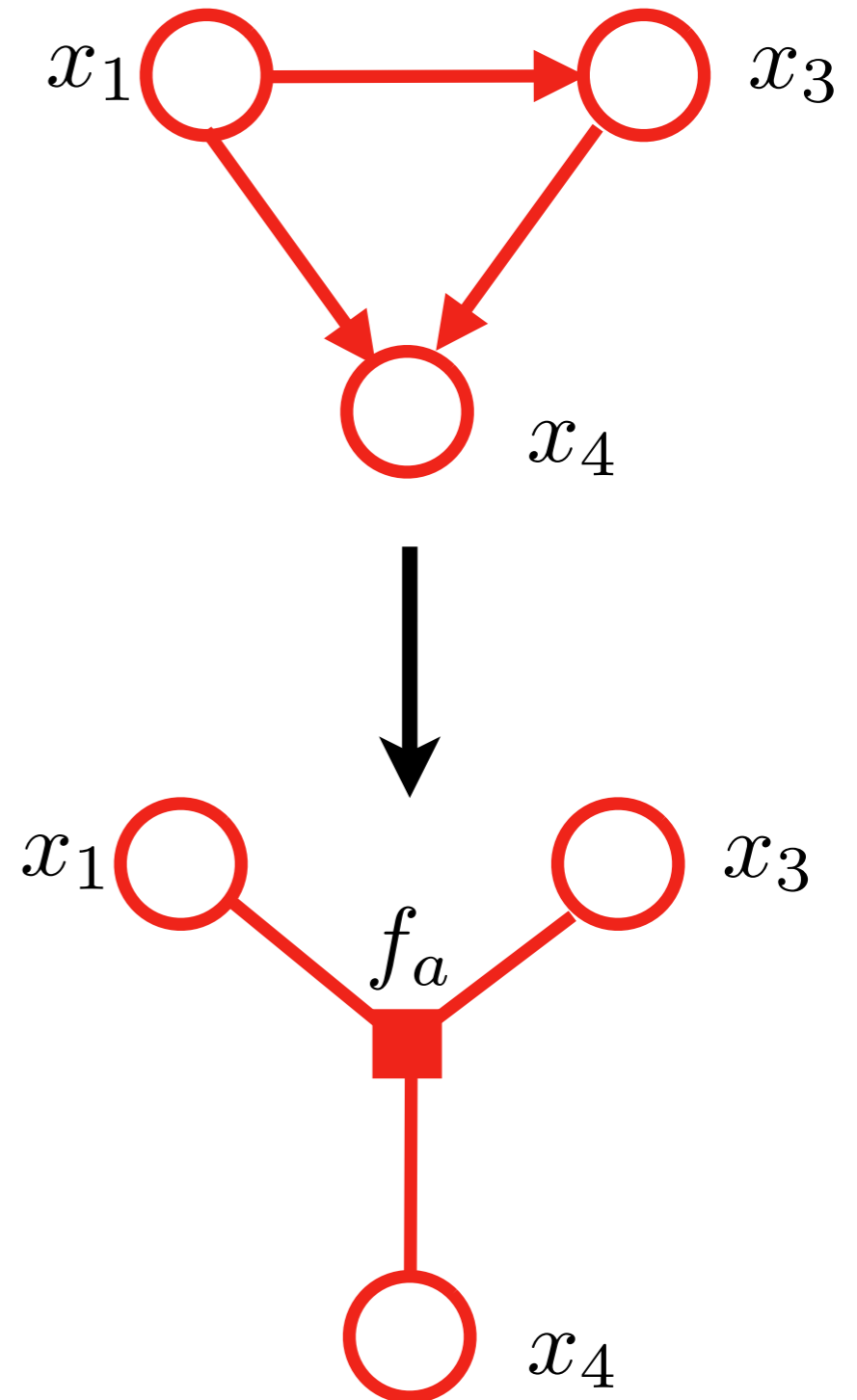
## Factor graphs

- can contain **multiple factors** for the same nodes
- are more general than undirected graphs
- are **bipartite**, i.e. they consist of two kinds of nodes and all edges connect nodes of different kind



# Factor Graphs

- Directed trees convert to tree-structured factor graphs
- The same holds for undirected trees
- Also: directed polytrees convert to tree-structured factor graphs
- And: Local cycles in a directed graph can be removed by converting to a factor graph





# Sum-Product Inference in General Graphical Models

1. Convert graph (directed or undirected) into a **factor graph** (there are no cycles)
2. If the goal is to **marginalize** at node  $x$ , then consider  $x$  as a root node
3. Initialize the recursion at the leaf nodes as:  
$$\mu_{f \rightarrow x}(x) = 1 \quad (\text{var}) \quad \text{or} \quad \mu_{x \rightarrow f}(x) = f(x) \quad (\text{fac})$$
4. Propagate messages from the leaves to  $x$
5. Propagate messages from  $x$  to the leaves
6. Obtain marginals at every node by multiplying all incoming messages



# Other Inference Algorithms

- Max-Sum algorithm: used to **maximize** the joint probability of all variables (no marginalization)
- Junction Tree algorithm: exact inference for general graphs (even with loops)
- Loopy belief propagation: approximate inference on general graphs (more efficient)

Special kind of undirected GM:

- Conditional Random fields (e.g.: classification)



# Conditional Random Fields

- Another kind of undirected graphical model is known as **Conditional Random Field (CRF)**.
- CRFs are used for classification where labels are represented as discrete random variables  $\mathbf{y}$  and features as continuous random variables  $\mathbf{x}$
- A CRF represents the conditional probability

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{\prod_C \phi_C(\mathbf{x}_C, \mathbf{y}_C; \mathbf{w})}{\sum_{\mathbf{y}'} \prod_C \phi_C(\mathbf{x}_C, \mathbf{y}'_C; \mathbf{w})}$$

where  $\mathbf{w}$  are parameters learned from training data.

- CRFs are **discriminative** and MRFs are **generative**



# Conditional Random Fields

Derivation of the formula for CRFs:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \frac{p(\mathbf{y}, \mathbf{x} \mid \mathbf{w})}{p(\mathbf{x} \mid \mathbf{w})} = \frac{p(\mathbf{y}, \mathbf{x} \mid \mathbf{w})}{\sum_{\mathbf{y}'} p(\mathbf{y}', \mathbf{x} \mid \mathbf{w})} = \frac{\prod_C \phi_C(\mathbf{x}_C, \mathbf{y}_C; \mathbf{w})}{Z \sum_{\mathbf{y}'} \prod_C \phi_C(\mathbf{x}_C, \mathbf{y}'_C; \mathbf{w})}$$

In the training phase, we compute parameters  $\mathbf{w}$  that maximize the posterior:

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} p(\mathbf{w} \mid \mathbf{x}, \mathbf{y}) = \arg \max_{\mathbf{w}} p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) p(\mathbf{w})$$

where  $(\mathbf{x}, \mathbf{y})$  is the training data and  $p(\mathbf{w})$  is a Gaussian prior. In the inference phase we maximize

$$\arg \max_{y^*} p(y^* \mid \mathbf{x}^*, \hat{\mathbf{w}})$$



# CRF Training

We minimize the negative log-posterior:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \{-\ln p(\mathbf{w} \mid \mathbf{x}^*, \mathbf{y}^*)\} = \arg \min_{\mathbf{w}} \{-\ln p(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{w}) - \ln p(\mathbf{w})\}$$

Computing the likelihood is intractable, as we have to compute the partition function for each  $\mathbf{w}$ . We can approximate the likelihood using **pseudo-likelihood**:

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathbf{w}) \approx \prod_i p(y_i^* \mid \mathcal{M}(y_i^*), \mathbf{x}^*, \mathbf{w})$$

where

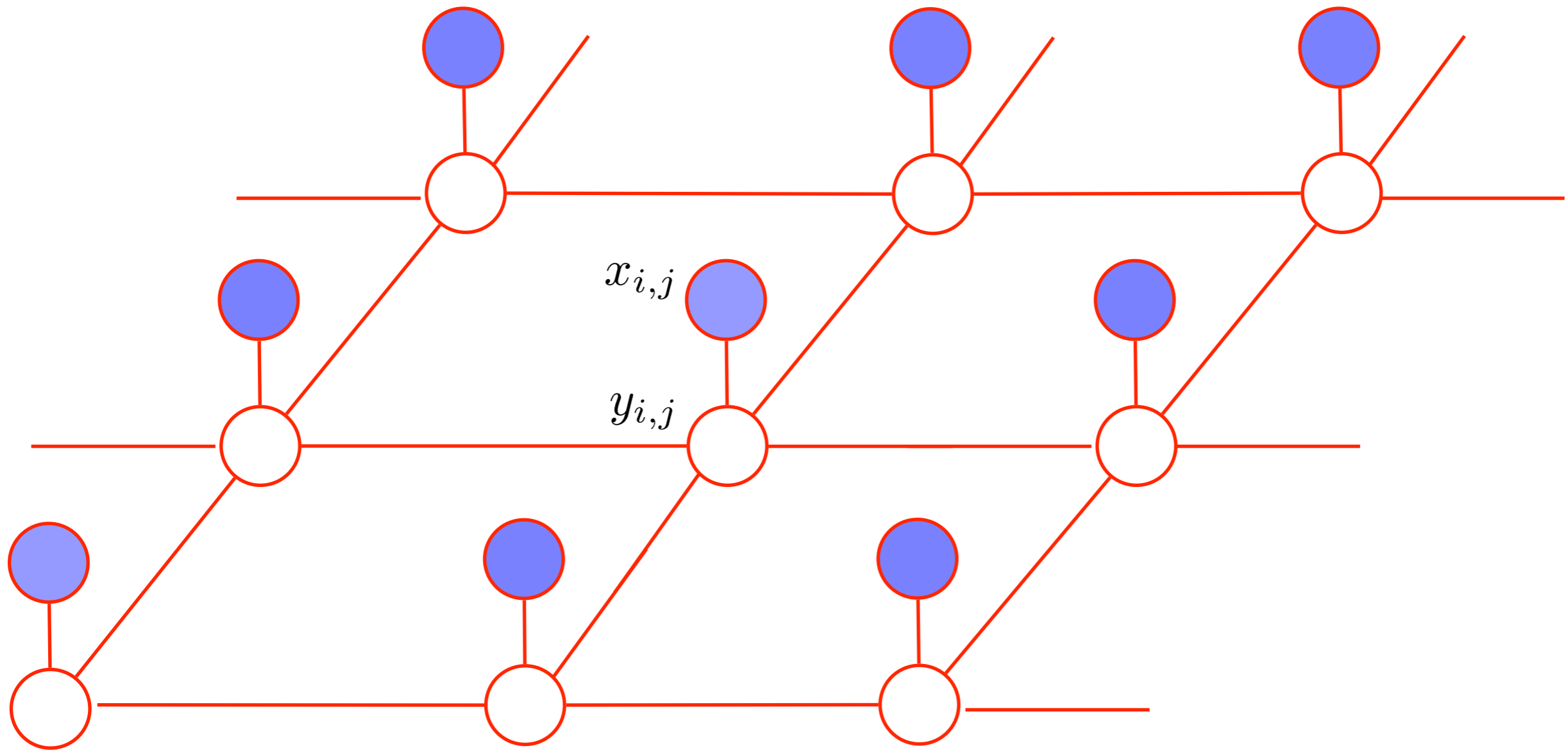
Markov blanket

$C_i$ : All cliques containing  $y_i$

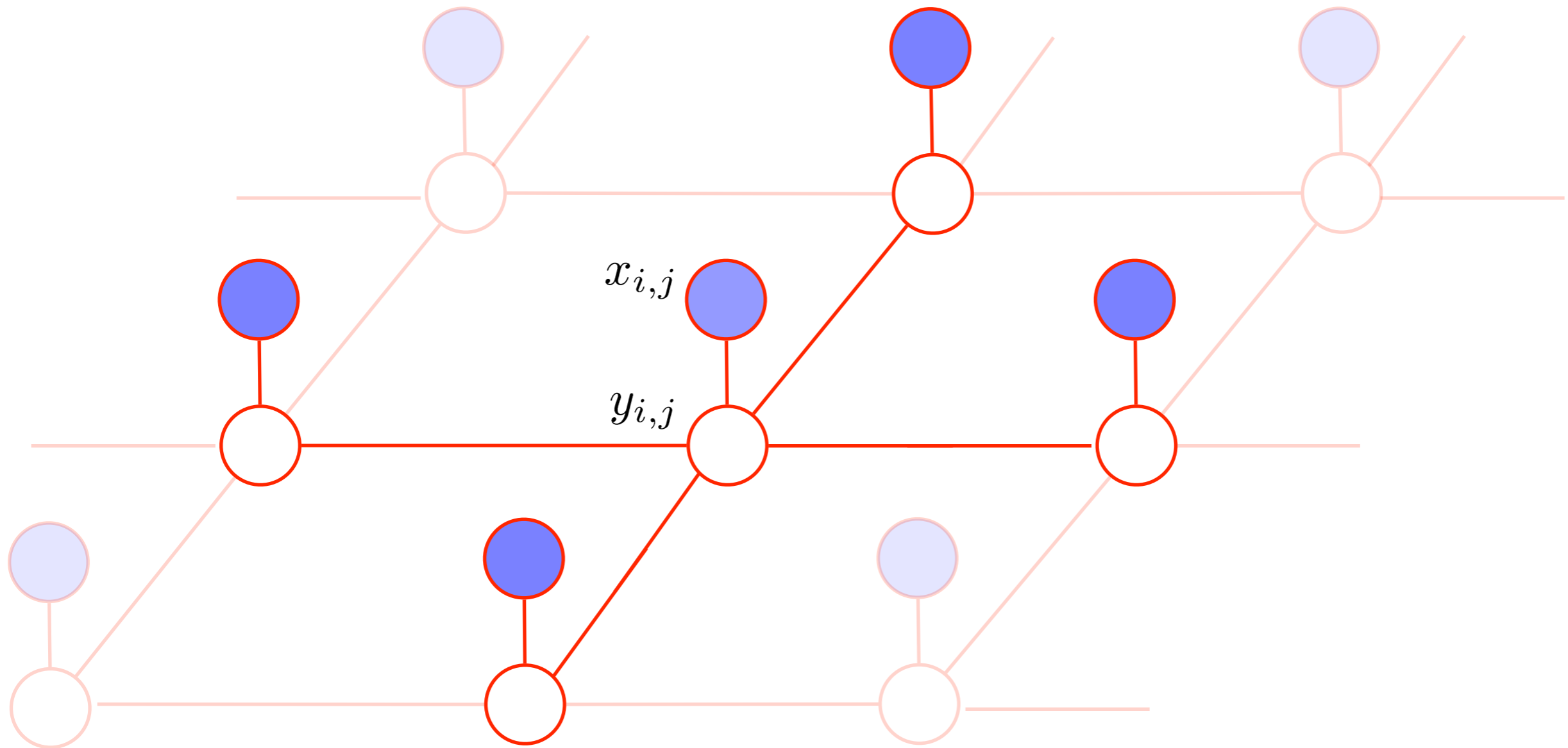
$$p(y_i^* \mid \mathcal{M}(y_i^*), \mathbf{x}^*, \mathbf{w}) = \frac{\prod_{C_i} \phi_{C_i}(\mathbf{x}_{C_i}^*, y_i^*, \mathbf{y}_{C_i}^*; \mathbf{w})}{\sum_{y_i'} \prod_{C_i} \phi_{C_i}(\mathbf{x}_{C_i}^*, y_i', \mathbf{y}_{C_i}^*; \mathbf{w})}$$



# Pseudo Likelihood



# Pseudo Likelihood



Pseudo-likelihood is computed only on the Markov blanket of  $y_i$  and its corresp. feature nodes.



# Potential Functions

- The only requirement for the potential functions is that they are positive. We achieve that with:

$$\phi_C(\mathbf{x}_C, \mathbf{y}_C, \mathbf{w}) := \exp(\mathbf{w}^T f(\mathbf{x}_C, \mathbf{y}_C))$$

where  $f$  is a compatibility function that is large if the labels  $\mathbf{y}_C$  fit well to the features  $\mathbf{x}_C$ .

- This is called the **log-linear model**.
- The function  $f$  can be, e.g. a local classifier





# Summary

- Undirected models (aka Markov random fields) provide an intuitive representation of conditional independence
- An MRF is defined as a **factorization** over clique potentials and normalized globally
- Directed and undirected models have different representative power (no simple “containment”)
- Inference on undirected Markov chains is efficient using message passing
- Factor graphs are more general; exact inference can be done efficiently using sum-product

