# An Introduction to Probabilistic Graphical Models

Cédric Archambeau

Xerox Research Centre Europe
cedric.archambeau@xrce.xerox.com

Pascal Bootcamp
Marseille, France, July 2010

# Reference material

- D. Koller and N. Friedman (2009): Probabilistic graphical models: principles and techniques.
- C. Bishop (2006): Pattern recognition and machine learning. (Graphical models chapter available online, as well as the figures — many are used in these slides after post-processing by Iain Murray and Frank Wood.)
- K. Murphy (2001): An introduction to graphical models.
- Michael Jordan (1999): Learning in graphical models.
- S. Lauritzen (1996): Graphical models.
- J. Pearl (1988): Probabilistic reasoning in intelligent systems.
- Tutorials (e.g Tiberio Caetano at ECML 2009) and talks on videolectures!

# Statistical machine learning:
# A mariage between statistics and computer science

- Data is omnipresent (web, images, sound, sensors, ...), but inherently noisy and unreliable
- Modelling strategy is to assume the data was generated according to some (hierarchy of) probability distributions
- Amount of data grows exponentially over time, so computational complexity is major issue!
- Questions we would like to answer:
  - Estimate parameters of the model in light of the data
  - Compute probabilities of particular outcomes given these parameters
  - Find particularly interesting realizations (e.g. most probable assignment)
  - Can we do these tasks efficiently?

# Graphical models:
# A marriage between probability theory and graph theory

- Designed to deal with uncertainty and complexity, increasingly important in machine learning and computational statistics
- Multivariate probabilistic models, structured in terms of conditional independence assumptions
- Graph theoretic aspect provides intuitive representation and is helpful to analyse, reason on and devise new models
- Complex sytems are built by combining simpler parts and the possible relations among them in a probabilistic way
- Probability theory is the glue, ensuring whole system is consistent and can take data into account

# Graphical models are applied in ...

- Bioinformatics
- Natural language processing
- Document processing
- Speech processing
- Image processing
- Computer vision
- Time series analysis
- Economics
- Physics
- Social Sciences
- ...

# Organising document collections (Blei et al., JMLR 2003)

| "Arts" | "Budgets" | "Children" | "Education" |
|---|---|---|---|
| NEW | MILLION | CHILDREN | SCHOOL |
| FILM | TAX | WOMEN | STUDENTS |
| SHOW | PROGRAM | PEOPLE | SCHOOLS |
| MUSIC | BUDGET | CHILD | EDUCATION |
| MOVIE | BILLION | YEARS | TEACHERS |
| PLAY | FEDERAL | FAMILIES | HIGH |
| MUSICAL | YEAR | WORK | PUBLIC |
| BEST | SPENDING | PARENTS | TEACHER |
| ACTOR | NEW | SAYS | BENNETT |
| FIRST | STATE | FAMILY | MANIGAT |
| YORK | PLAN | WELFARE | NAMPHY |
| OPERA | MONEY | MEN | STATE |
| THEATER | PROGRAMS | PERCENT | PRESIDENT |
| ACTRESS | GOVERNMENT | CARE | ELEMENTARY |
| LOVE | CONGRESS | LIFE | HAITI |

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

- Discovering themes/topics in large text corpora
- Simple generative model for text (bag-of-words assumption)
- Monitor trends, discover social network, etc.

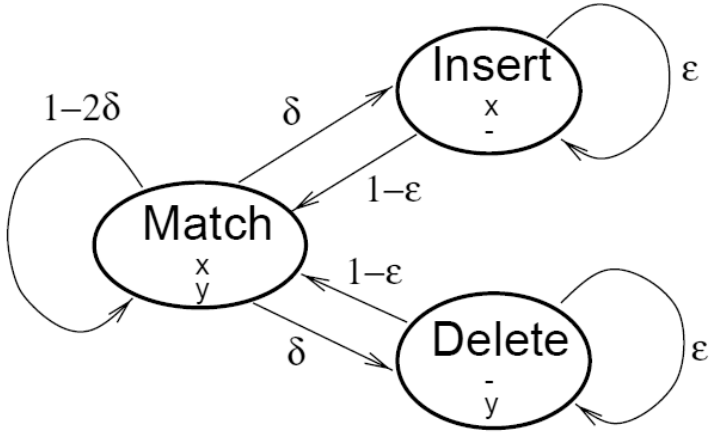
# Bioinformatics (Husmeier et al.)

Raw sequences

A T C G T C A G C T C
A C C C A G G T C

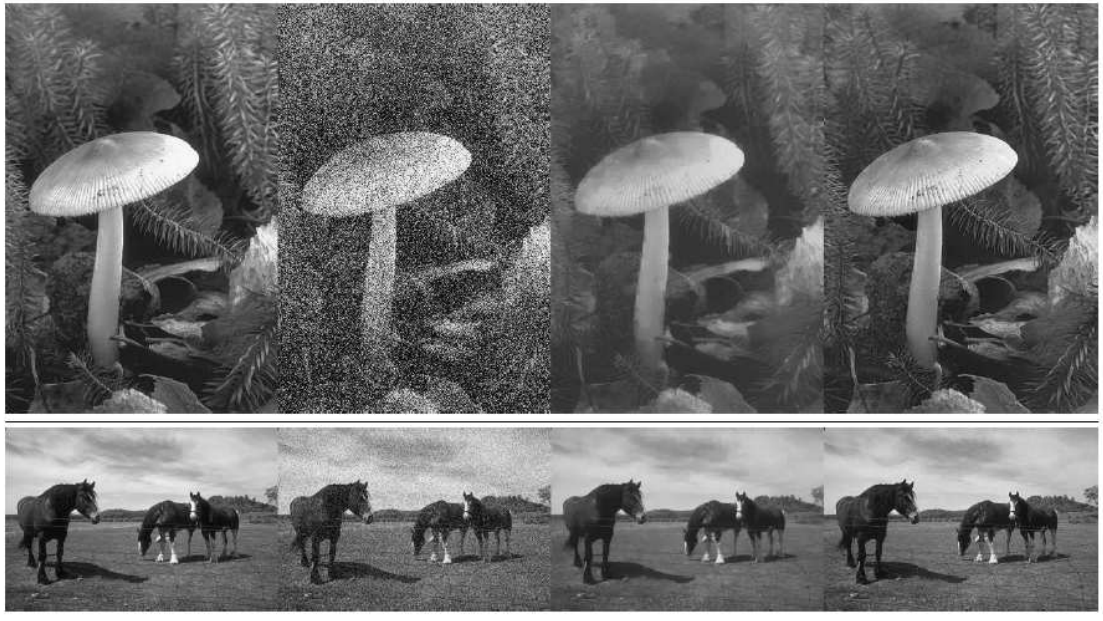
After alignment

A T C G T C A G C T C
A C C C – – A G G T C



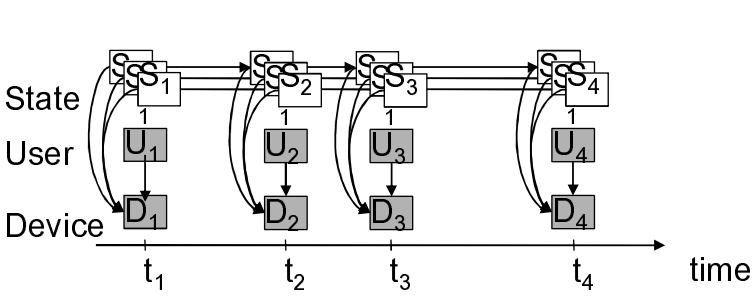

- Pairwise sequence alignment with gaps
- Biological sequences come in families
- Understand evolution, different species, etc.

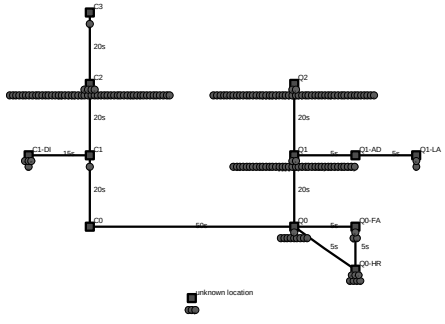
# Image denoising (McAuley et al., ICML 2006)



(Markov random fields, use neighborhood information)

# Printer infrastructure management

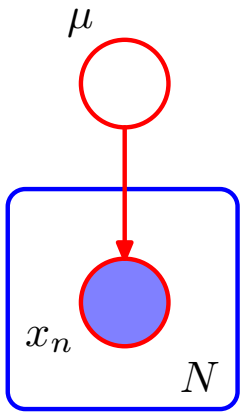



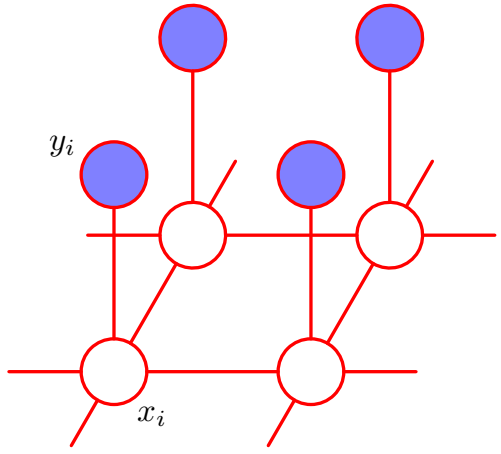
(infrastructure map, user and device locations, printing profiles, device characteristics, soft failure detection, infractructure optimisation, ...)

# Overview

- Introduction
- Bayesian networks
- Markov networks (Markov Random fields)
- Elements of exact inference
- Elements of learning

# Basics




- Nodes denote random variables, shaded nodes are observed, unshaded are unobserved (latent, hidden) random variables
- (Lack of) edges represent conditional independencies, plates indicate replications
- Directed graphs: Bayesian networks or nets, belief networks, generative models, etc.
- Undirected graphs: Markov networks, Markov Random Fields, etc.
- Combinations are called chain graphs

# Conditional independence (CI)

- Statistical independence: $X \perp\!\!\!\perp Y$

$$p(x, y) = p(x)p(y)$$

- Conditional independence: $X \perp\!\!\!\perp Y|Z$

$$\begin{aligned} p(x, y|z) &= p(x|y, z)p(y|z) = p(x|z)p(y|z), \\ p(x|y, z) &= p(x|z), \\ p(y|x, z) &= p(y|z). \end{aligned}$$

- Graphical models are useful when the compactness of the model arises from conditional independence statements:

$$\underbrace{p(x, y|z)}_{f_1(3 \text{ variables})} = \underbrace{p(x|z)}_{f_2(2 \text{ variables})} \times \underbrace{p(y|z)}_{f_3(2 \text{ variables})}$$

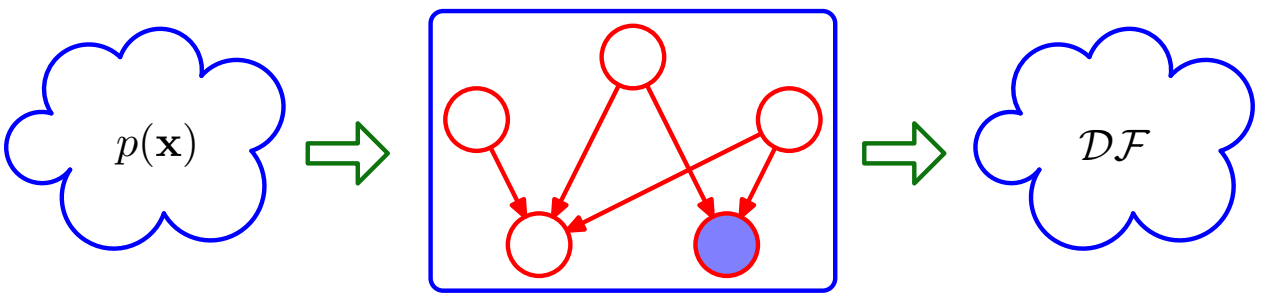
- CI imposes constraints on the model (some random variables cannot take arbitrary values while conditioning)

# CI examples

- My wife's mood $\perp\!\!\!\perp$ my boss' mood | my mood
- My genome $\perp\!\!\!\perp$ my grandmother's genome | my mother's genome
- My position on a board game $\perp\!\!\!\perp$ my first position | my last position
- The color of a pixel $\perp\!\!\!\perp$ the color of faraway pixels | the color of neighboring pixels
- This year's harvest $\perp\!\!\!\perp$ last year's harvest | this year's cummulative sunshine and rainfall
- ...

# Probabilistic graphical model

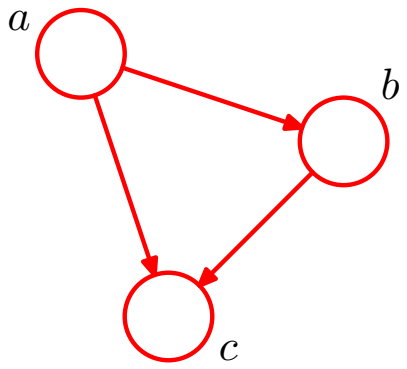
- Let $\{X_n\}_{n=1}^N$ be a set of random variables.
- A probabilistic graphical model is a family of joint probability distributions $p(x_1, \ldots, x_N)$ for which some CI assumptions hold.
- The set of CI assumptions $\{X_i \perp\!\!\!\perp X_j | X_k\}$ induces a structure in $p(x_1, \ldots, x_N)$, which can be read off from the graph.
- This structure allows us to make computations more tractable and storage more efficient.
- In the case of Bayesian networks this is sometimes called a directed factorisation (DF) filtering of the joint:



# Overview

- Introduction
- Bayesian networks
- Markov networks (Markov Random fields)
- Elements of exact inference
- Elements of learning

# Bayesian networks (directed graphical models)




- A Bayesian network is a set of probability distributions associated to a directed acyclic graph (DAG).
- Node $a$ is a parent of node $b$ if there is a link from $a$ to $b$ (conversely we say that $b$ is a child of $a$).
- A node is independent of its ancestors given its parents.
- Choosing a topological ordering $(a, b, c)$ will lead to a particular decomposition of the joint:

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

- Random variables can be discrete or continuous.

# Factorisation in directed graphical models

- CIs and a fixed ordering of the nodes of the DAG lead to a particular factorisation of the joint.
- For a graph with $N$ nodes, we decompose the joint in terms of conditionals on the parents:

$$p(x_1, \ldots, x_N) = \prod_{n=1}^{N} p(x_n | \mathrm{pa}_n), \qquad \mathrm{pa}_n : \text{ parents of } x_n.$$

- The factorisation is in terms of local conditional distributions.
- The decomposition is not unique, but the joint is correctly normalised.
- Is this type of factorisation useful?

# Factorisation in directed graphical models


- Consider the special case where $M = 3$:

$$p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2), \quad \forall m : x_m \in \{1, \ldots, K\}.$$

- Factorisation allows us to exploit the distributive law to make computations more tractable:

  without: $p(x_2) = \sum_{x_1, x_3} p(x_1, x_2, x_3)$ is $\mathcal{O}(K^3)$

  with: $p(x_2) = \sum_{x_2} p(x_1, x_2) \sum_{x_3} p(x_3|x_2)$ is $\mathcal{O}(2K^2)$

- Factorisation leads to a more efficient representation:

  without: requires $K^M - 1$ parameters

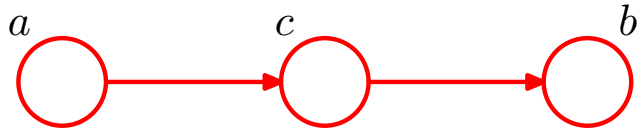
  with: requires $K - 1 + (M - 1)K(K - 1)$ parameters

# D-separation

- CIs are usually known by the (human) expert.
- CIs are imposed by removing links.
- Is the CI property equivalent to factorisation?
- Do we induce other (hidden) CIs?

- D-separation are a set of rules to read off CIs directly from a DAG.
- D-separation does not require manipulations of the joint, but it does imply a factorised form ($\sim$ simplification).

# Head-to-tail nodes: independence

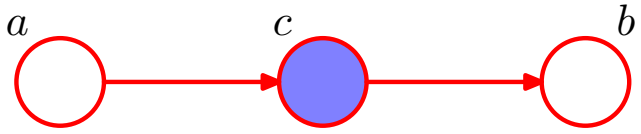



- Consider the head-to-tail node $c$. Are $a$ and $b$ independent?
- Let's check $p(a, b) \stackrel{?}{=} p(a)p(b)$:

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

- In general we do not obtain the statistical independence property.

# Head-to-tail nodes: conditional independence

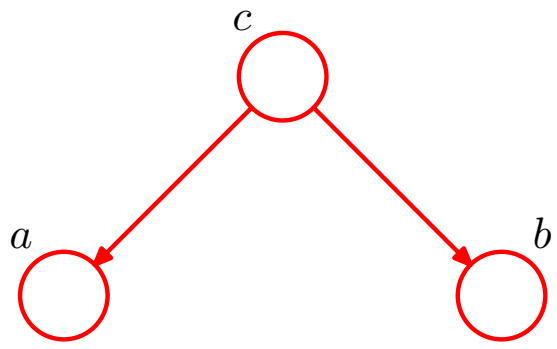



- Assume $c$ is observed. Are $a$ and $b$ conditionally independent?
- Let's check $p(a, b|c) \stackrel{?}{=} p(a|c)p(b|c)$:

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

- We obtain the conditional independence property: $a \perp\!\!\!\perp b|c$.
- Applying Bayes rule reverts the link!

# Tail-to-tail nodes: independence

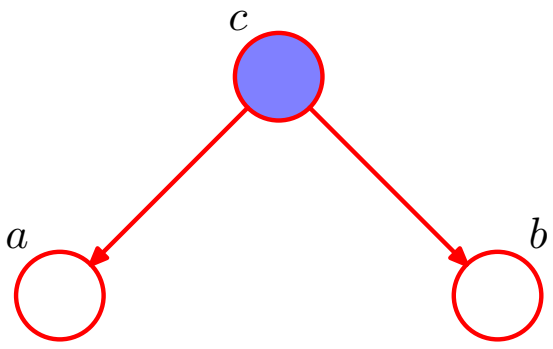



- Consider the tail-to-tail node $c$. Are $a$ and $b$ independent?
- Let's check $p(a, b) \stackrel{?}{=} p(a)p(b)$:

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

- In general we do not obtain the statistical independence property.

# Tail-to-tail nodes: conditional independence

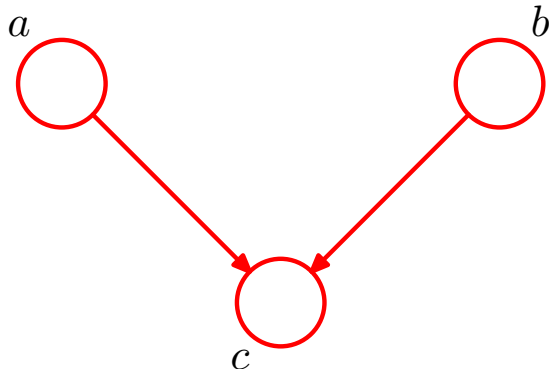



- Assume $c$ is observed. Are $a$ and $b$ conditionally independent?
- Let's check $p(a, b|c) \stackrel{?}{=} p(a|c)p(b|c)$:

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

- We obtain the conditional independence property: $a \perp\!\!\!\perp b|c$.

# Head-to-head nodes: independence

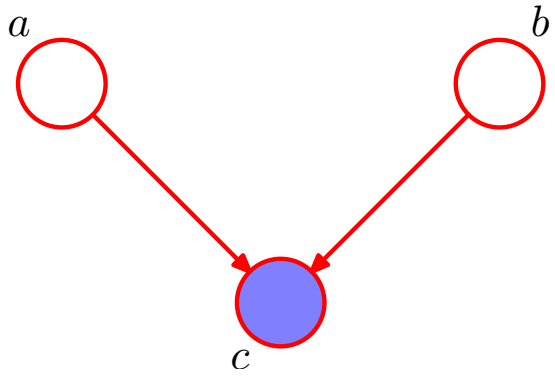



- Consider the head-to-head node $c$. Are $a$ and $b$ independent?
- Let's check $p(a, b) \stackrel{?}{=} p(a)p(b)$:

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

- We obtain the statistical independence property: $a \perp\!\!\!\perp b$.

# Head-to-head nodes: conditional independence




- Assume $c$ is observed. Are $a$ and $b$ conditionally independent?
- Let's check $p(a, b|c) \stackrel{?}{=} p(a|c)p(b|c)$:

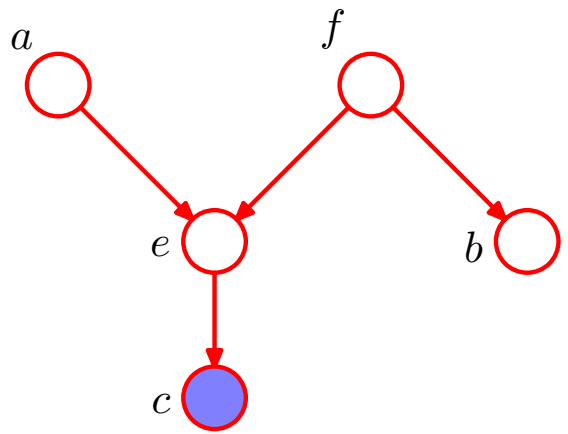
$$p(a, b|c) = \frac{p(a, b, c)}{p(c)} = \frac{p(a)p(b)p(c|a, b)}{p(c)}$$

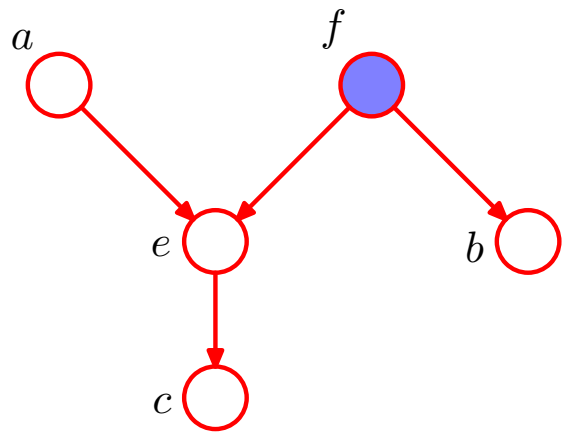
- In general we do not obtain the conditional independence property.

# Blocked paths

A blocked path is one containing at least one of the following types of nodes:

- An observed head-to-tail or tail-to-tail node.
- An unobserved head-to-head node, of which none of the descendants are observed.



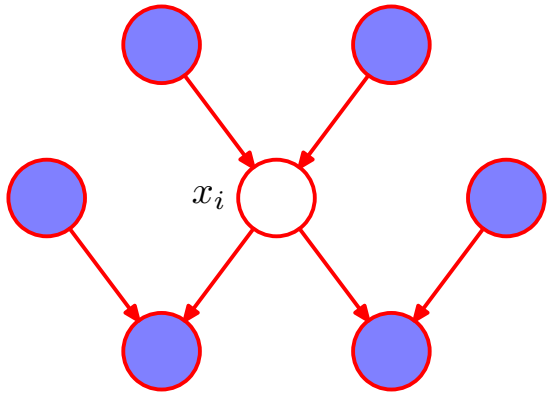
# D-separation, CI and factorisation

- Let $A$, $B$ and $C$ be nonintersecting sets of nodes. $A \perp\!\!\!\perp B | C$ if all possible paths from any node in $A$ to any node in $B$ are blocked:
  - If the arrows on the path meet head-to-tail or tail-to-tail at some nodes, then these nodes are in $C$;
  - If the arrows on the path meet head-to-head at some nodes, then none of these nodes or its descendants are in $C$.
- Parameters do not play a role in d-separation (are always tail-to-tail), but integrating them out usually destroys CIs
- CI properties and factorisation are equivalent:

  fact $\Rightarrow$ CI: Let $p$ be a probability distribution that factorises according to a DAG. If $A$, $B$ and $C$ are disjoint subsets of nodes such that $A$ is d-separated from $B$ by $C$, then $p(A, B, C)$ satisfies $A \perp\!\!\!\perp B | C$.

  CI $\Rightarrow$ fact: If $p$ satisfies the CI properties implied by d-separation over a particular DAG, then it factorises according to this DAG.

# Markov blanket in Bayesian networks




- Consider a Bayesian network associated to a DAG:

$$p(x_1, \ldots, x_N) = \prod_n p(x_n|\mathrm{pa}_n).$$

- Using CI we can express any conditional $p(x_i|\{x_j\}_{j\neq i})$:

$$p(x_i|\{x_n\}_{n\neq i}) = \frac{\prod_n p(x_n|\mathrm{pa}_n)}{\sum_{x_i}\prod_n p(x_n|\mathrm{pa}_n)} \propto p(x_i|\mathrm{pa}_i)\prod_{n_i} p(x_{n_i}|\mathrm{pa}_{n_i}),$$

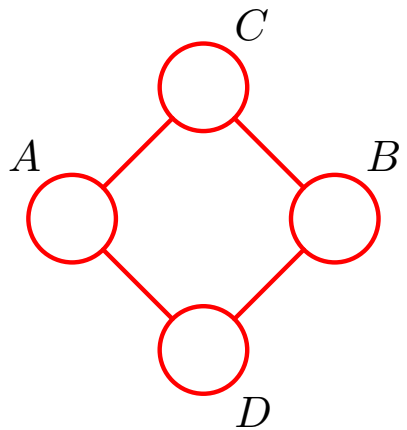
  where $\mathrm{pa}_{n_i}$ includes node $x_i$.
- The Markov blanket of $x_i$ is the minimal set of nodes that isolates $x_i$ from the rest of the graph, that is parents and children of $x_i$, as well as co-parents of the children of $x_i$.

# Overview

- Introduction
- Bayesian networks
- Markov networks (Markov Random fields)
- Elements of exact inference
- Elements of learning

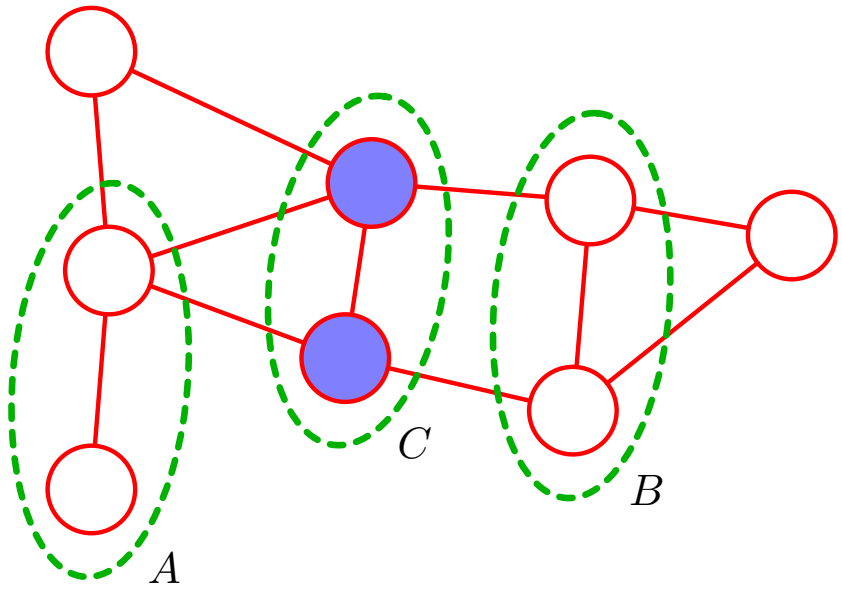
# Markov random fields (undirected graphical models)

- Some CI assumptions cannot be satisfied by any Bayesian network.
- Markov random fields (MRFs) allow for the specification of a different class of CIs.
- Based on different graphical semantics, we would like CI to be directly determined by simple graph separation.



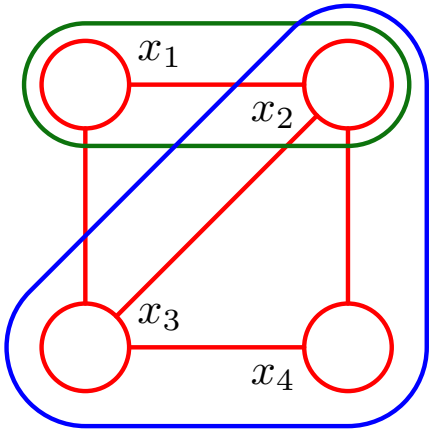

- A MRF is a set of probability distributions associated to an undirected graph.
- The absence of edges imply CIs, random variables can be discrete or continuous.

# Graph separation




- $A \perp\!\!\!\perp B | C$ if all paths connecting nodes in $A$ to nodes in $B$ pass through nodes in $C$.
- We say that the path is blocked and the CI property holds.
- The Markov blanket in MRFs takes a very simple form: it consists of all the neighboring nodes.

# Cliques




- A clique of a graph is a complete subgraph (every pair of nodes is connected by an edge).
- A maximal clique of a graph is clique which is not a subset of another clique.

# Factorisation in undirected graphical models

- For a graph with $K$ maximal cliques, we can decompose the joint as

$$p(x_1, \ldots, x_N) = \frac{1}{Z} \prod_{k=1}^{K} \psi_{C_k}(\mathbf{x}_{C_k}),$$

  where $\mathbf{x}_{C_k}$ are the nodes belonging to the maximal clique $C_k$.
- The factorisation is in terms of local potential functions $\{\psi_{C_k}(\cdot)\}_k$, with $\psi_{C_k}(\cdot) \geq 0$ for all $k$.
- The potential functions do not necessarily have a probabilistic interpretation (do not have to be conditionals or marginals).
- Computing the normalising constnat $Z$ (also known as partition function) is the main difficulty:

$$Z = \sum_{x_1} \ldots \sum_{x_N} \prod_{k=1}^{K} \psi_{C_k}(\mathbf{x}_{C_k}).$$

- For continuous random variables the sum is replaced by an integral.

# Separation, CI and factorisation

- CI properties and factorisation are equivalent in MRFs with positive potentials:

  fact $\Rightarrow$ CI: Let $p$ be probability distribution that factorises according to an undirected graph. If $A$, $B$ and $C$ are disjoint subsets of nodes such that $C$ separates $A$ from $B$, then $p(A, B, C)$ satisfies $A \perp\!\!\!\perp B | C$.

  CI $\Rightarrow$ fact: If $p$ is strictly positive ($p(x) > 0, \forall x$) and satisfies the CI properties implied by graph separation over the undirected graph, then it factorises according to this graph.

- The latter is known as the Hammersley-Clifford theorem.
- $\psi_C(\mathbf{x}_C)$ being restricted to be positive, we interpret it as a Boltzmann distribution:

$$\psi_C(\mathbf{x}_C) \propto e^{-E(\mathbf{x}_C)},$$

where $E(\mathbf{x}_C)$ is the energy function.

# MRFs versus Bayesian networks

Similarities:

- CI properties of the joint are encoded into the graph structure and define families of structured probability distributions.
- CI properties are related to concepts of separation of (groups of) nodes in the graph.
- Local entities in the graph imply the simplified algebraic structure (factorization) of the joint.
- Reasoning in graphical models does not require to specifiy the local functions.

Differences:

- The set of probability distributions represented by MRFs is different from the set represented by Bayesian networks.
- MRFs have a normalization constant that couples all factors, whereas Bayesian networks have not.
- Factors in Bayesian networks are probability distributions, while factors in MRFs are nonnegative potentials.

# Mapping a Bayesian networks into a MRF

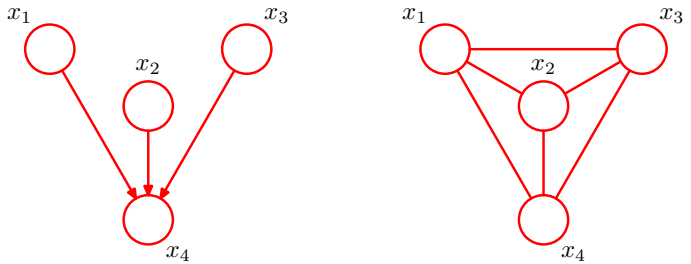

- Bayesian network:

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_2|x_1)\ldots p(x_N|x_{N-1}).$$

- MRF:

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

- The mapping is here straightforward.
- When there are head-to-head nodes one has to add edges to convert the Bayesian network into the undirected graph ($\sim$ moralisation):



# Overview

- Introduction
- Bayesian networks
- Markov networks (Markov Random fields)
- Elements of exact inference
- Elements of learning

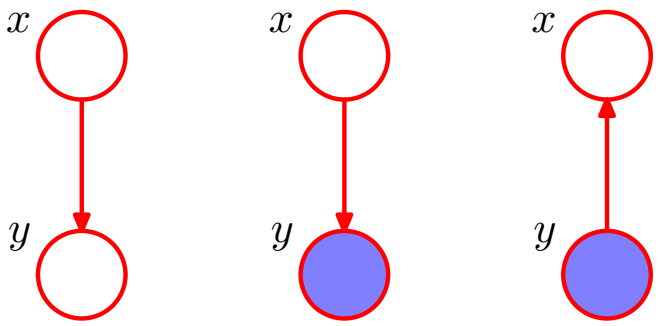
# Exact inference in graphical models

- Assume some nodes are clamped, i.e. some random variables are observed (data).
- The goal is to compute the posterior distribution of one or more subsets of nodes given the observed ones.
- Exact probabilistic inference algorithms:
  - Belief propagation (and sum-product)
  - Max-product (and max-sum)
  - Junction tree algorithm
- The distributive law is the key to efficient inference in graphical models:

$$\underbrace{ab + ac}_{3 \text{ operations}} = \underbrace{a(b + c)}_{2 \text{ operations}} .$$

- Most inference algorithms can be viewed as propagating messages around the graph.
- The focus will be on discrete random variables, but results equally hold for continuous random variables (replacing sums by integrals).

# Graphical interpretation of Bayes' rule

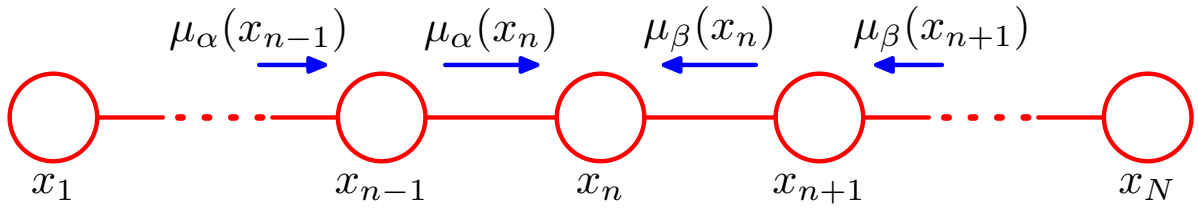



(a) (b) (c)

$$\underbrace{p(y|x)}_{\text{posterior}} = \frac{\overbrace{p(x|y)}^{\text{likelihood}}\overbrace{p(y)}^{\text{prior}}}{\underbrace{p(x)}_{\text{evidence}}}$$

- Allows us to update a prior belief on some random variable $y$ into a posterior belief, based on the observation $x$.
- DAG (b) leads to $p(x, y) = p(x)p(y|x)$.
- Applying Bayes' rule results in DAG (c): $p(x, y) = p(y)p(x|y)$.

# Belief propagation in a (Markov) chain




- Can we compute the marginal $p(x_n)$ efficiently?
- Naive computation is $\mathcal{O}(K^N)$ if $x_n$ can take $K$ values for all $n$:

$$p(x_n) = \frac{1}{Z} \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_N} \psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N).$$

- Using the distributive law is $\mathcal{O}(NK^2)$: $p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n)$, where $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$ are messages passed forward and backward along the chain:

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \Big[ \sum_{x_{n-2}} \cdots \Big[ \sum_{x_1} \psi_{1,2}(x_1, x_2) \Big] \cdots \Big],$$

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

# Belief propagation in a (Markov) chain

- To compute $p(x_n)$, we only need the *incoming messages* to $x_n$.
- To compute a message to the right (left), we need all previous messages coming from the left (right).
- The messages can be computed recursively:

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1}), \qquad \mu_\alpha(x_1) = 1,$$
$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1})\mu_\beta(x_{n+1}), \qquad \mu_\beta(x_N) = 1.$$

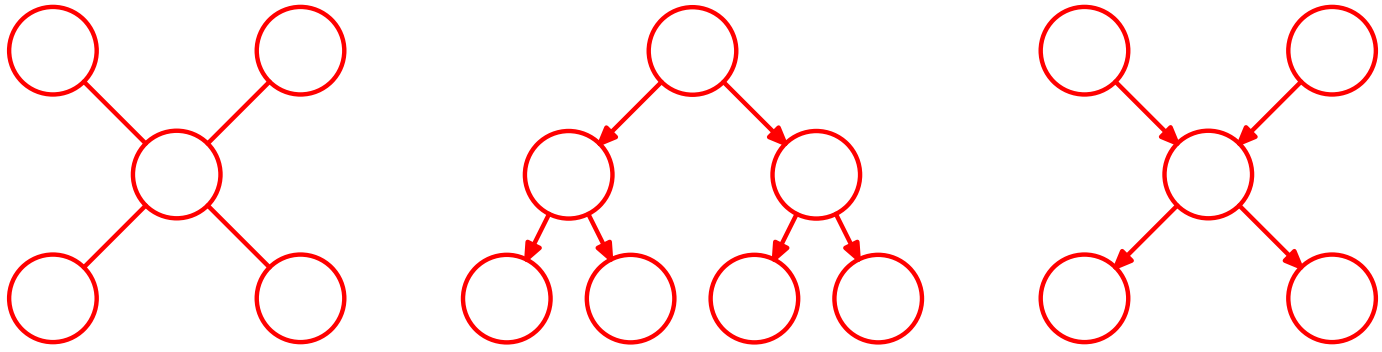
- A chain with $N$ nodes require $2(N-1)$ messages to be computed, observed nodes are clamped to their observed values.
- Computing the normalising constant can be done at any node and is $\mathcal{O}(K)$:

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

- Joint distributions of neighboring nodes are expressed in terms of the same messages:

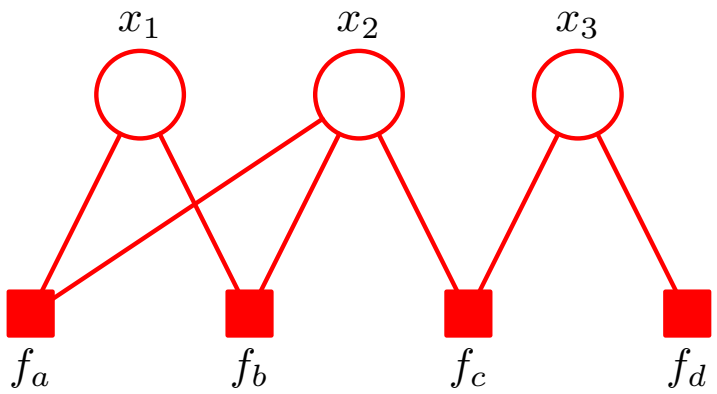
$$p(x_n, x_{n-1}) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n)$$

# Belief propagation in a tree



- In chains the forward (backward) message arriving in a node $x_n$ captures all the information from the nodes to the left (right) of $x_n$.
- Every node can be seen as a leaf node after it has received the forward or backward message.
- True leaves give us the right place to start the message passing along the chain.
- This property also holds in (poly)trees (graph where only a single path connects any pair of nodes, no loops).

# Factor graph

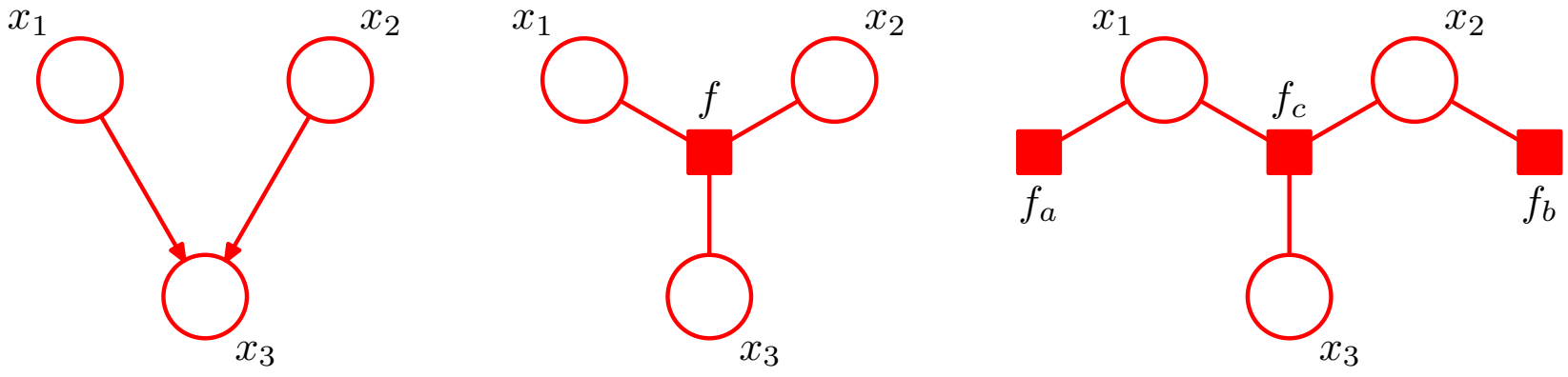



- The nodes are (groups of) random variables and squares are factors.
- Family of joint probability distributions associated to a bipartite graph:

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

- Factor $f_s$ is a function of the neighboring nodes $\mathbf{x}_s$.
- Factor graphs incorporate explicit details about the factorisation, but factor nodes do no correspond to CIs.
- They preserve the tree structure of DAGs and undirected graphs!

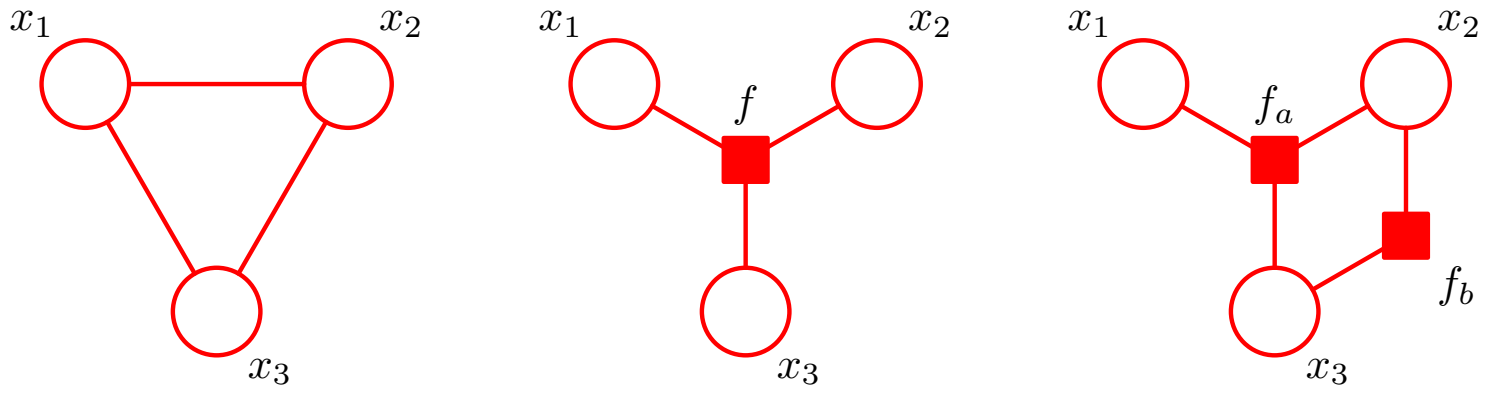
# Converting a DAG into a factor graph




- Create nodes corresponding to the orginal DAG.
- Create factor nodes corresponding the conditional distributions.
- Conversion is not unique:

$$f(x_1, x_2, x_3) = p(x1)p(x_2)p(x_3|x_1, x_2),$$
$$f_a(x_1) = p(x1), \quad f_b(x_2) = p(x_2), \quad f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2).$$

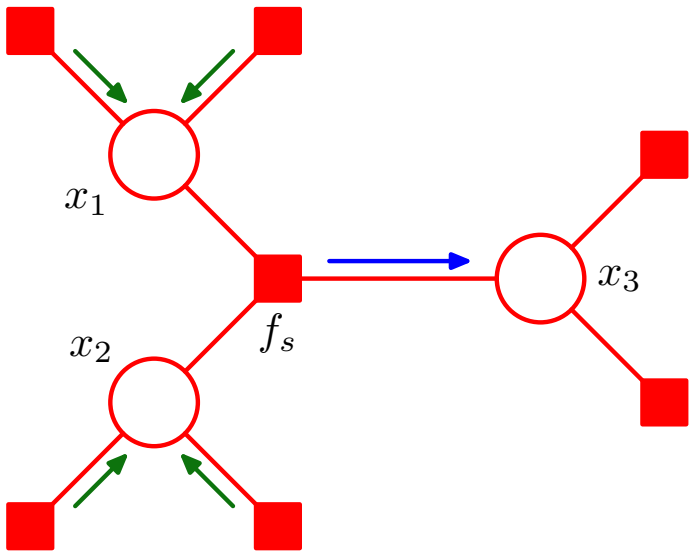
# Converting an undirected graph into a factor graph




- Create nodes corresponding to the orginal undirected graph.
- Create factor nodes corresponding the potential functions.
- Conversion is not unique:

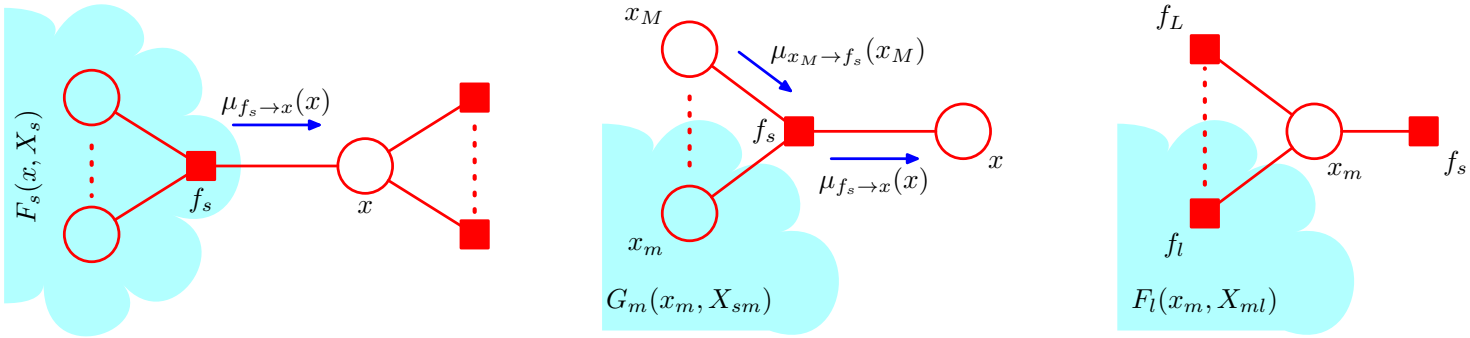
$$f(x_1, x_2, x_3) = \psi_{1,2,3}(x_1, x_2, x_3),$$
$$f_a(x_1, x_2, x_3) f_b(x_2, x_3) = \psi_{1,2,3}(x_1, x_2, x_3).$$

# Sum-product algorithm




- General class of exact inference algorithm in trees
- Operates on factor graphs
- Designed to evaluate local marginals efficiently
- Shares computations whenever possible
- The idea is again to exploit the distributive law
- Messages sent out of factors nodes to other factor nodes

# Sum-product algorithm




- We are interested in the marginal $p(x_n)$:

$$p(x_n) = \sum_{\mathbf{x} \setminus x_n} \prod_s f_s(\mathbf{x}_s) = \prod_s \mu_{f_s \to x_n}(x_n).$$

- Let $f_s$ be a factor neighboring $x_n$ and let $x_1, \ldots, x_m$ be the other nodes neighboring $f_s$. The message from $f_s$ to $x_n$ is defined as

$$\mu_{f_s \to x_n}(x_n) = \sum_{x_1} \ldots \sum_{x_M} f_s(\mathbf{x}_s) \prod_m \mu_{x_m \to f_s}(x_m).$$

- Let $f_l$ be the other factors neighboring $x_m$. The message from $x_m$ to node $f_s$ is defined as

$$\mu_{x_m \to f_s}(x_m) = \prod_l \mu_{f_l \to x_m}(x_m).$$

# Sum-product algorithm

- Messages are computed recursively in the tree.
- Some node $x_n$ is viewed as the root (choice has no impact on result).
- The recursion states at the leave nodes or leave factors:


- The algorithm proceeds in two steps to compute all marginals:
  1. Messages are propagated from the leaves to some root until this root has received the messages from all its neighbors to compute $p(x_n)$.
  2. Messages are then passed outwards from this root to the leaves.
- The total number of messages to compute is two times the number of edges (only twice the number computations for a single marginal).
- For factor graphs derived from undirected graphs $Z$ can be computed at any node.
- The marginal associated to a factor is given by

$$p(\mathbf{x}_s) = f_s(\mathbf{x}_s) \prod_{n\in \mathrm{ne}_s} \mu_{x_n\to f_s}(x_n), \qquad \mathrm{ne}_s : \text{neigbors of } f_s.$$

# Incorporating data

- So far we focussed on computing marginals in absence of data.
- The goal in inference is to compute posterior distributions over hidden variables $\mathbf{z}$ conditionned on observations $\mathbf{x}$:

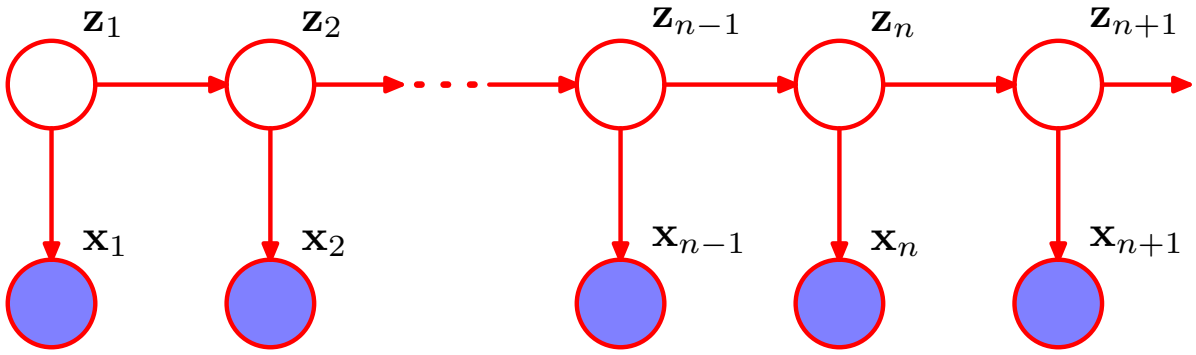
$$p(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{x}, \mathbf{z}).$$

- Observations correspond to clamped nodes (their value is fixed):

$$p(\mathbf{z}|\mathbf{x} = \hat{\mathbf{x}}) \propto p(\mathbf{x}, \mathbf{z}) \prod_i \delta_{\hat{x}_i}(x_i),$$

  where $\delta(\cdot)$ is the Kronecker delta (or Dirac for continuous r.v.)
- The sum-product algorithm leads to the posteriors $p(z_n|\mathbf{x})$ for all $n$ up to a normalisation constant.
- The normalising constant is obtained as before.

# Hidden Markov model (HMM)




- Flexible model for sequential data (e.g. time series, DNA, text, ...)
- Markov property (future is independent of past given present):

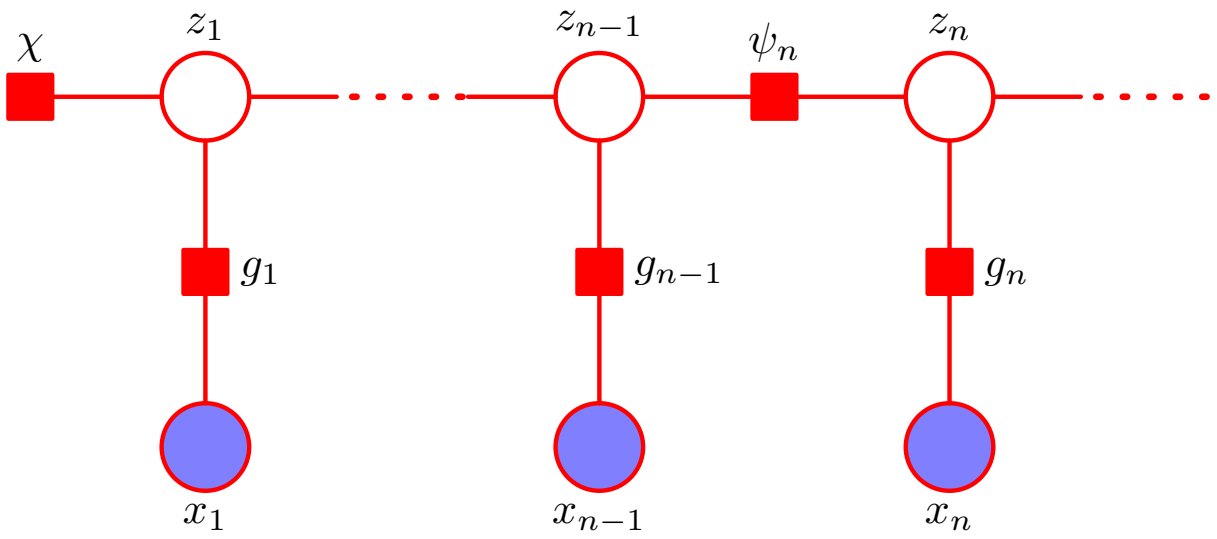
$$z_{n-1} \perp\!\!\!\perp z_{n+1} | z_n.$$

- Hidden Markov chain of discrete variables, observed variables $\{x_n\}$ are discrete or continuous:

$$p(x_1, \ldots, x_N, z_1, \ldots, z_N) = p(z_1) \left( \prod_{n=2}^{N} p(z_n|z_{n-1}) \right) \left( \prod_{n=1}^{N} p(x_n|z_n) \right).$$

- D-separation shows that $x_n$ given $\{x_i\}_{i<n}$ does not exhibit any CI!

# Sum-product in action (HMM factor graphs)

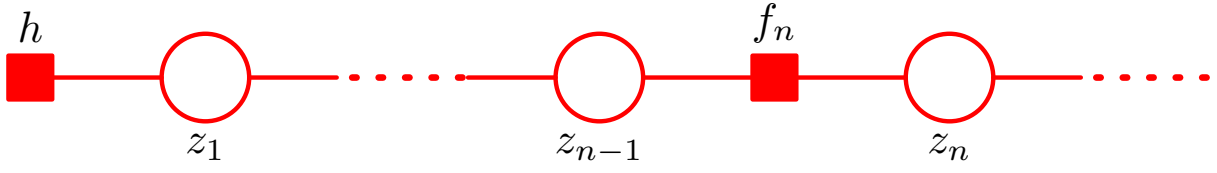



- Factor graph:

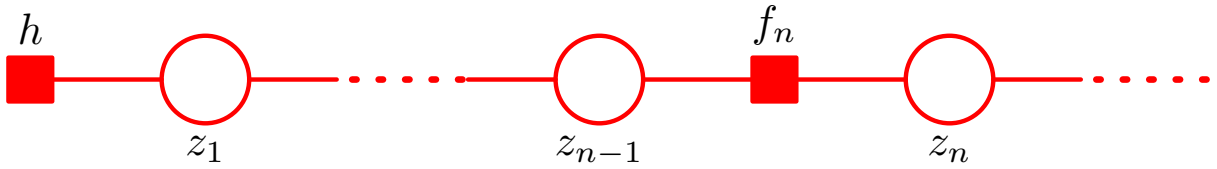
$$g_n = p(x_n|z_n), \quad \psi_n(z_{n-1}, z_n) = p(z_n|z_{n-1}), \quad \chi(z_1) = p(z_1).$$

- Simplified factor graph:

$$f_n(z_{n-1}, z_n) = p(z_n|z_{n-1})p(x_n|z_n), \quad h(z_1) = p(z_1)p(x_1|z_1).$$



# Sum-product in action (HMM forward recursion)




- Let $h$ be the leaf node and $\mathbf{z}_N$ the root node.
- Left messages from/to factors are given by

$$\mu_{f_n \to z_n}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n) \mu_{z_{n-1} \to f_n}(z_{n-1}),$$

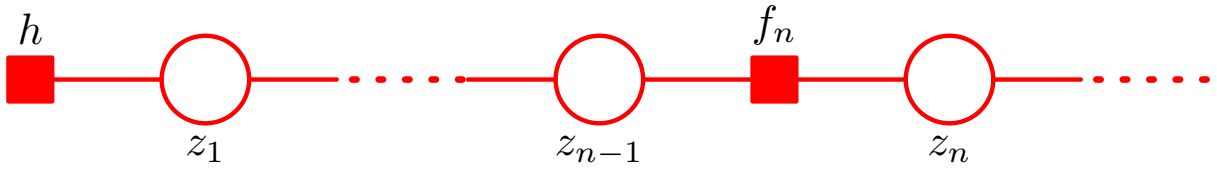
$$\mu_{z_{n-1} \to f_n}(z_{n-1}) = \mu_{f_{n-1} \to z_{n-1}}(z_{n-1}).$$

- We obtain the forward recursion (from leaf to root):

$$\mu_{f_n \to z_n}(z_n) = p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \mu_{f_{n-1} \to z_{n-1}}(z_{n-1}),$$

$$\mu_{h \to z_1}(z_1) = p(x_1|z_1) p(z_1).$$

# Sum-product in action (HMM backward recursion)




- Right messages from/to factors are given by

$$\mu_{f_n \to z_{n-1}}(z_{n-1}) = \sum_{z_n} f_n(z_{n-1}, z_n)\mu_{z_n \to f_n}(z_n),$$

$$\mu_{z_n \to f_n}(z_n) = \mu_{f_{n+1} \to z_n}(z_n).$$

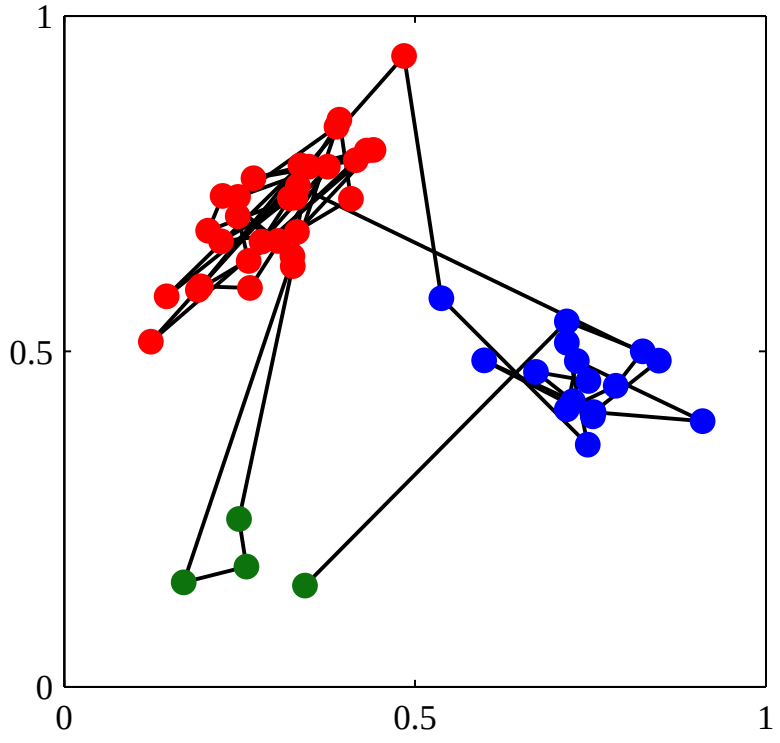
- We obtain the backward recursion (from root to leaf):

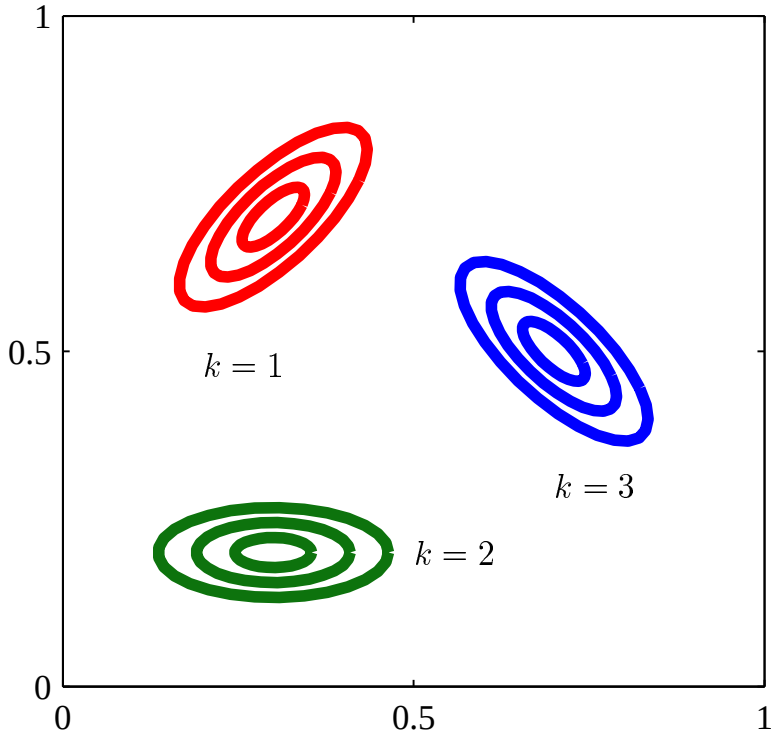
$$\mu_{f_n \to z_{n-1}}(z_{n-1}) = \sum_{z_n} p(x_n|z_n)p(z_n|z_{n-1})\mu_{f_{n+1} \to z_n}(z_n),$$

$$\mu_{z_N \to f_N}(z_1) = 1 = \mu_{f_N \to z_{N-1}}(z_{N-1}).$$

- The posterior marginals are proportional to the product of the incoming messages.

# HMM with 3 states and Gaussian noise




(5% probability of state transition.)

# Applications of HMMs

- Speech processing
- Handwritten recognition
- Matching of DNA sequences
- Financial time series
- Prediction of energy consumption
- ...

# Max-product algorithm

- Max-product is a variant of sum-product algorithm
- The goal is to find the most probable joint configuration:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} p(\mathbf{x}), \qquad \mathbf{x} = (x_1, \ldots, x_N).$$

- Different from computing all marginals by sum-product and then maximise them individually.
- The distributive law still applies:

$$\max(ab, ac) = a \max(b, c), \qquad a > 0.$$

- Max-product works exactly the same way as sum-product for messages from leaves to root, just replace sums by max operator!

# Max-sum algorithm

- To avoid numerical undeflow we use the max-sum algorithm, which operates on the logarithm of the joint:

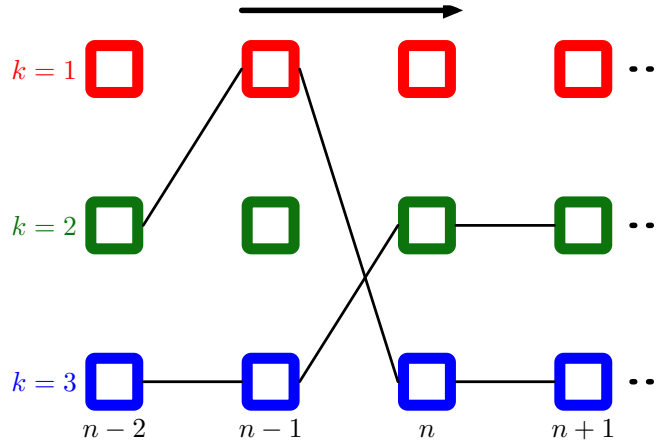
$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} \ln p(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{x}} \sum_s \ln f_s(\mathbf{x}_s).$$

- This is ok as the logarithm is a monotonic ($a > b \Rightarrow \ln a > \ln b$), such that

$$\max(\ln ab, \ln ac) = \ln a + \max(\ln b, \ln c).$$

- Max-sum works exactly the same way as sum-product for messages from leaves to root: replace sums by max and products by sums of logarithms.
- For messages from root to leaves the story is different...

# Max-sum algorithm with backtracking




- When propagating messages from leaves to root, multiple configurations can lead to a maximum value of the joint.
- Store the maximising configurations of previous variables wrt the next variables and backtrack to restore the maximising path.
- Max-sum with backtracking on a chain is known as the Viterbi algorithm.

# Junction-tree algorithm

- Generalization of sum-product for arbitrary graphs (not just trees)
- Can be applied to any graph (DAG or undirected), but efficient only for certain classes of graphs.
- DAGs are first transformed into undirected graphs by moralisation.
- JT operates on chordal graphs (triangulated graph), which are eventually transformed into junction trees.
- Exact marginalization is performed by means of sum-product type of algorithm.
- Complexity grows exponentially with the treewidth (number of variables in largest clique).

# Overview

- Introduction
- Bayesian networks
- Markov networks (Markov Random fields)
- Elements of exact inference
- Elements of learning

# Learning in graphical models

- Probabilistic inference aims at computing posterior distributions of hidden variables (or their most likely configuration) given data.
- The quality of statistical models depend on the parameter setting:

$$p(x_1, \ldots, x_N; \theta_1, \ldots, \theta_K) = \frac{1}{Z(\boldsymbol{\theta})} \prod_s f_s(\mathbf{x}_s; \boldsymbol{\theta}_s).$$

- The goal of learning or statistical inference is to estimate these parameters:
  - Maximum likelihood
  - Maximum a posteriori
  - Expectation-maximisation algorithm
- We assume the data $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{M}$ are drawn i.i.d.:
  - All $\mathbf{x}^{(i)}$ are drawn from $p(\mathbf{x}; \boldsymbol{\theta})$ (identical assumption)
  - $\mathbf{x}^{(i)} \perp\!\!\!\perp \mathbf{x}^{(j)}$ for $i \neq j$ (independence assumption)

# Maximum likelihood (ML)

- Assume there are no latent (hidden) variables.
- The joint probability of observing i.i.d. data is a function of the parameters:

$$\ell(\boldsymbol{\theta}; \mathbf{X}) = \ln \prod_i p(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sum_i \sum_s \ln f_s(\mathbf{x}_s^{(i)}; \boldsymbol{\theta}_s) - M \ln Z(\boldsymbol{\theta}).$$

- The goal in maximum likelihood learning is to find the parameters that maximise the log-likelihood function:

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{X}).$$

- Altenatively one can minimise the negative log-likelihood.
- A local optimum must satisfy $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{X}) = 0$, or equivalently $\nabla_{\boldsymbol{\theta}_s} \ell(\boldsymbol{\theta}; \mathbf{X}) = 0$ for all $s$.

# ML in Bayesian networks

- ML estimation reduces to the estimation of $s$ local ML estimation problems.
- We have $\ln Z(\boldsymbol{\theta}) = 0$ and the constraints $\{\sum_{\mathbf{x}_s} f_s(\mathbf{x}_s; \boldsymbol{\theta}_s) = 1\}$.
- We obtain a set $2S$ equations by taking the derivatives wrt the parameters $\{\boldsymbol{\theta}_s\}$ and the Lagrange multipliers $\{\lambda_s\}$.
- Each pair of equations in $\boldsymbol{\theta}_s$ and $\lambda_s$ can be solved independently.
- Your favourite numerical optimisation tool can be used when there is no closed form solution.

# Maximum likelihood in MRFs

- In MRFs we have $\ln Z(\boldsymbol{\theta}) \neq 0$, such that the resulting equations are coupled!
- The optimisation problem is often difficult and nonlinear, with multiple local optima.
- The problem becomes convex for distributions in the exponential family, which includes the Gaussian, Binomial, Multinomial, Poisson, Gamma, Beta, etc. as special cases.

# Maximum a posteriori (MAP)

- ML estimation can lead to overfitting (for small data set).
- MAP estimation is a first step towards Bayesian statistics.
- The idea is to impose a prior distribution on the parameters, which has the effect of penalising unreasonable values.
- MAP estimation maximises $p(\boldsymbol{\theta}|\mathbf{X}) \propto p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})$:

$$\ell_{\mathrm{MAP}}(\boldsymbol{\theta}; \mathbf{X}) = \sum_i \sum_s \ln f_s(\mathbf{x}_s^{(i)}; \boldsymbol{\theta}_s) - M \ln Z(\boldsymbol{\theta}) + \sum_s \ln p(\boldsymbol{\theta}_s).$$

- MAP is sometimes called penalised maximum likelihood, it is a way of introducing regularisation
- MAP estimation leads to point estimates of $\boldsymbol{\theta}$ (while Bayesian statistics is interested in the full posterior).

# Expectation-maximisation (EM)

- Assume there are observed as well as latent variables:

$$p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_s f_s(\mathbf{x}_s, \mathbf{z}_s; \boldsymbol{\theta}_s).$$

- If we knew the latent variables $\{\mathbf{z}^{(i)}\}$, the problem would reduce to ML (or MAP) estimation.
- Since $\{\mathbf{z}^{(i)}\}$ are unobserved, ML requires to maximise the *incomplete* log-likelihood:

$$\begin{aligned}\ell(\boldsymbol{\theta}; \mathbf{X}) &= \ln \prod_i \sum_{\mathbf{z}^{(i)}} p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}; \boldsymbol{\theta}) \\ &= \sum_i \ln \sum_{\mathbf{z}^{(i)}} \prod_s f_s(\mathbf{x}_s^{(i)}; \boldsymbol{\theta}_s) - M \ln Z(\boldsymbol{\theta}).\end{aligned}$$

- The product is "blocked" inside the logarithm because of the sum, making the marginalisation often analytically intractable.

# EM (lower bound)

- The key idea is to maximise the expected value of the log-complete likelihood since $\mathbf{Z} = \{\mathbf{z}^{(i)}\}$ are unobserved:

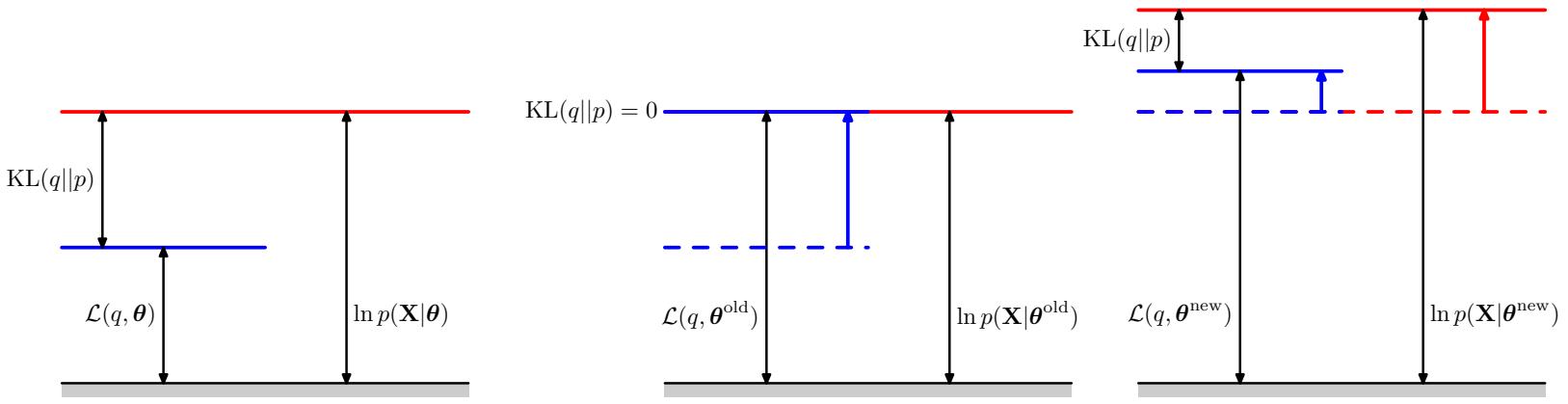
$$\begin{aligned}
\ell(\boldsymbol{\theta}; \mathbf{X}) &= \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta}) \\
&= \ln \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta})}{q(\mathbf{Z})} \\
&\geq \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta})}{q(\mathbf{Z})} \equiv \mathcal{L}(q, \boldsymbol{\theta})
\end{aligned}$$

  where $q(\mathbf{Z})$ is called the variational distribution.
- The lower bound follows from Jensen's inequality:

$$f(x) \text{ is convex} \Rightarrow \mathbb{E}(f(x)) \geq f(\mathbb{E}(x)).$$

- The quantity $-\mathcal{L}(q, \boldsymbol{\theta})$ can be interpretted as the (variational) free energy from statistical physics.

# EM (principle)

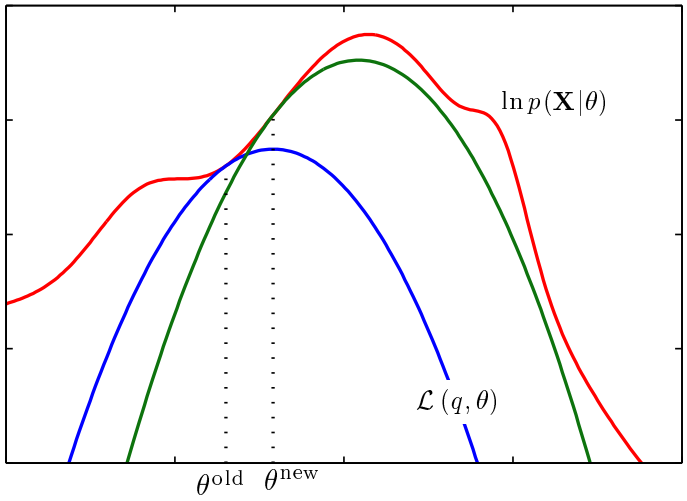



EM is based on two decompositions of the bound $\mathcal{L}(q, \boldsymbol{\theta})$:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \ln p(\mathbf{X}|\boldsymbol{\theta}) - \mathrm{KL}[q(\mathbf{Z}) \| p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})],$$
$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_q\{\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})\} + \mathrm{H}[q(\mathbf{Z})].$$

where $\mathrm{KL}[q\|p] = \mathbb{E}_q\{\ln \frac{q}{p}\}$ is the Kullback-Leibler divergence (or relative entropy) and $\mathrm{H}[q] = -\mathbb{E}\{\ln q)\}$ the entropy.

# EM (algorithm)



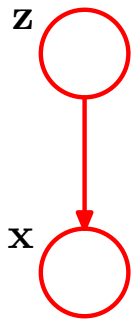

- Maximise lower bound by alternating between 2 steps:

  E step: Minimise KL for fixed $\boldsymbol{\theta}$ by setting $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})$.

  M step: Maximise $\mathbb{E}_q\{\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})\}$ for given $q(\mathbf{Z})$.

- Gradient ascent to local maxima of $\ell(\boldsymbol{\theta}; \mathbf{X})$, by construction it ensures monotonic increase of the bound.
- ML estimates of the parameters, still ok if $q$ is a good approximation of the posterior (approximate E step).
- Sum-product algorithm can be used in E step.

# Mixture of Bernoulli distributions




$$p(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\pi}) = \sum_k \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k) = \sum_k \pi_k \text{Bernoulli}(\boldsymbol{\mu}_k),$$

$$p(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_n \mu_{kn}^{x_n}(1-\mu_{kn})^{1-x_n}, \quad \pi_k \in [0,1], \quad \sum_k \pi_k = 1.$$

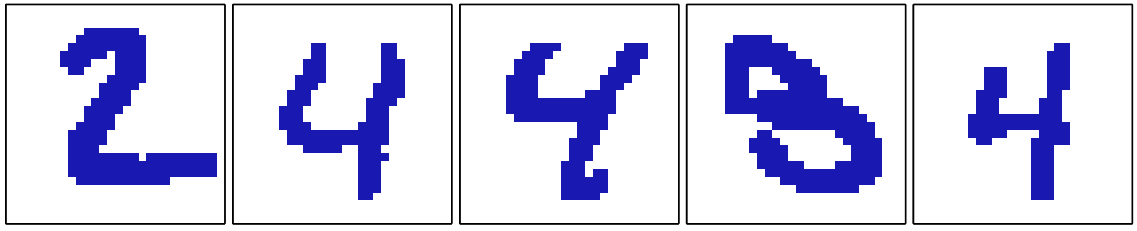
- No closed form solution for ML estimates of $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \pi_k\}$.
- For each set of binary variales $\mathbf{x}$ we introduce a discrete latent variable $z$ which indicates the mixture component:

$$p(z|\boldsymbol{\pi}) = \text{Discrete}(\boldsymbol{\pi}) = \prod_k \pi_k^{\delta_k(z)}.$$

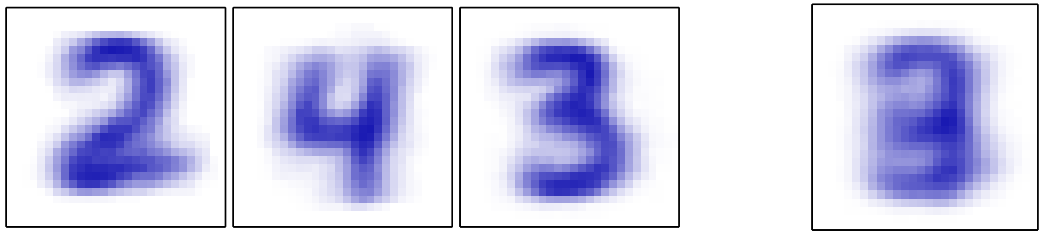
- The new graphical model is completed by

$$p(\mathbf{x}|z,\boldsymbol{\mu}) = \prod_k \left[\prod_n \text{Bernoulli}(\mu_{kn})\right]^{\delta_k(z)}.$$

# Mixture of Bernoullis (application)



- Pixelised handwritten digits, converted from grey scale to binary images by thresholding
- Goal is to cluster the images (recognise digit automatically), learning is done with EM algorithm
- The bottom figure shows the mean images for each of the 3 clusters, as well as the mean image when considering a single Bernoulli.



# Mixture of Bernoullis (EM updates)

E step : responsibilities:

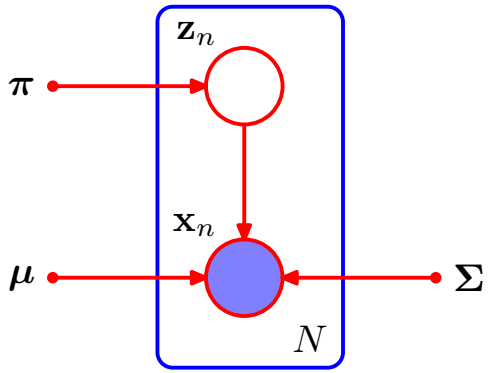
$$\rho_{ik} \equiv \mathbb{E}\{z^{(i)} = k\} = \frac{\pi_k p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k)}{\sum_{k'} \pi_{k'} p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_{k'})}$$

M step : mean and mixture proportions:

$$\boldsymbol{\mu}_k = \frac{1}{M_k} \sum_{i=1}^{M} \rho_{ik} \mathbf{x}^{(i)}, \qquad M_k = \sum_{i=1}^{M} \rho_{ik},$$

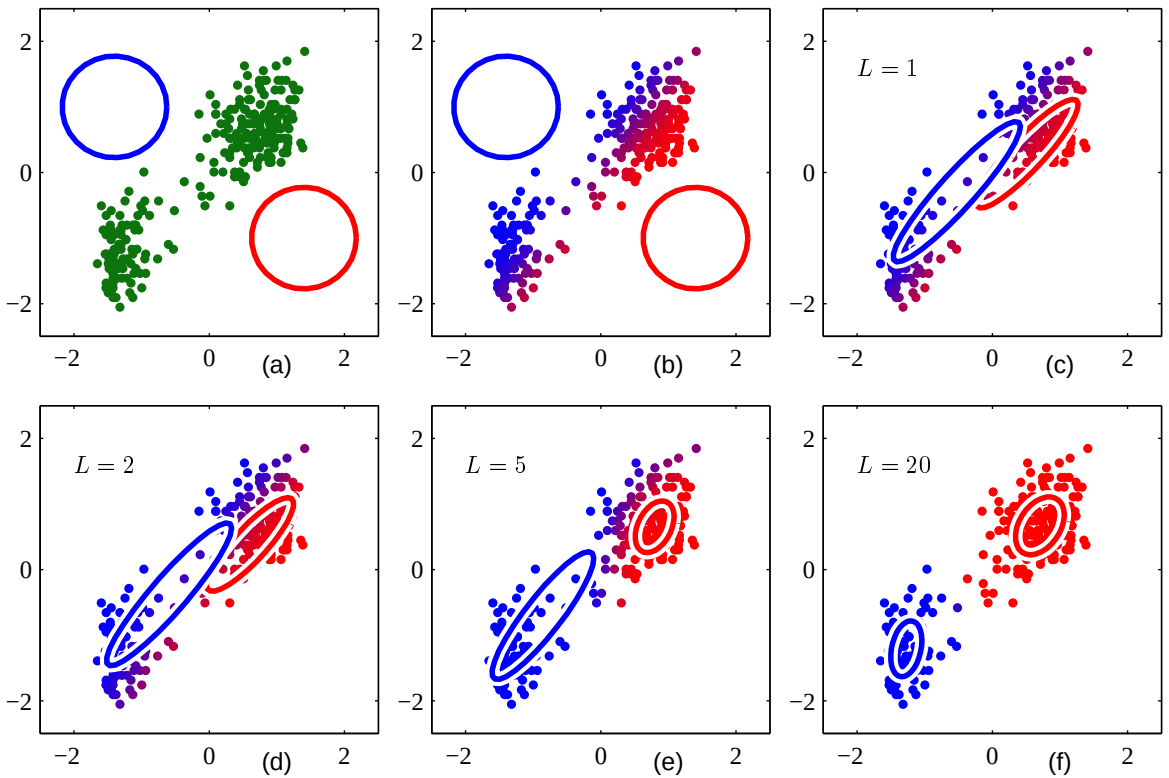
$$\pi_k = \frac{M_k}{M}.$$

# Mixture of Gaussians (Old Faithful geyser data)




$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_k \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

$$\pi_k \in [0, 1], \quad \sum_k \pi_k = 1.$$



# Want to know more about our research?

- Xerox Research Centre Europe (Grenoble, France): www.xrce.xerox.com
- Research positions in machine learning, natural language processing, machine translation, textual and visual pattern analysis, mechansim design, social network analysis...
- Internship and PhD opportunities all year round, so don't hesitate to get in touch!