Probabilistic models

Eric Moulines and Guillaume Obozinski

Telecom ParisTech, Ecole des Ponts - ParisTech

Probabilistic Graphical Models

Outline

Outline

Statistical paradigms

Statistical Model

Parametric model – Definition:

Set of distributions parameterized by a parameter vector $\theta \in \Theta \subset \mathbb{R}^p$

$$\mathcal{P}_\Theta = \left\{ p(x|\theta) \mid \theta \in \Theta \right\}$$

Statistical Model

Parametric model – Definition:

Set of distributions parameterized by a parameter vector $\theta \in \Theta \subset \mathbb{R}^p$

$$\mathcal{P}_\Theta = \{p(x|\theta) \mid \theta \in \Theta\}$$

Bernoulli model: $X \sim \text{Ber}(\theta)$ $\Theta = [0, 1]$

$$p(x|\theta) = \theta^x (1-\theta)^{(1-x)}$$

Statistical Model

Parametric model – Definition:

Set of distributions parameterized by a parameter vector $\theta \in \Theta \subset \mathbb{R}^p$

$$\mathcal{P}_\Theta = \{p(x|\theta) \mid \theta \in \Theta\}$$

Bernoulli model: $X \sim \text{Ber}(\theta)$ $\quad \Theta = [0, 1]$

$$p(x|\theta) = \theta^x (1-\theta)^{(1-x)}$$

Binomial model: $X \sim \text{Bin}(n, \theta)$ $\quad \Theta = [0, 1]$

$$p(x|\theta) = \binom{n}{x} \theta^x (1-\theta)^{(1-x)}$$

Statistical Model

Parametric model – Definition:

Set of distributions parameterized by a parameter vector $\theta \in \Theta \subset \mathbb{R}^p$

$$\mathcal{P}_\Theta = \{p(x|\theta) \mid \theta \in \Theta\}$$

Bernoulli model: $X \sim \text{Ber}(\theta)$ $\Theta = [0, 1]$

$$p(x|\theta) = \theta^x (1-\theta)^{(1-x)}$$

Binomial model: $X \sim \text{Bin}(n, \theta)$ $\Theta = [0, 1]$

$$p(x|\theta) = \binom{n}{x} \theta^x (1-\theta)^{(1-x)}$$

Multinomial model: $X \sim \mathcal{M}(n, \pi_1, \pi_2, \ldots, \pi_K)$ $\Theta = [0, 1]^K$

$$p(x|\theta) = \binom{n}{x_1, \ldots, x_k} {\pi_1}^{x_1} \ \ldots \ {\pi_k}^{x_k}$$

Indicator variable coding for multinomial variables

Let C a r.v. taking values in $\{1, \ldots, K\}$, with

$$\mathbb{P}(C = k) = \pi_k.$$

Indicator variable coding for multinomial variables

Let C a r.v. taking values in $\{1, \ldots, K\}$, with

$$\mathbb{P}(C = k) = \pi_k.$$

We will code C with a r.v. $Y = (Y_1, \ldots, Y_K)^\top$ with

$$\boxed{Y_k = 1_{\{C=k\}}}$$

Indicator variable coding for multinomial variables

Let C a r.v. taking values in $\{1, \ldots, K\}$, with

$$\mathbb{P}(C = k) = \pi_k.$$

We will code C with a r.v. $Y = (Y_1, \ldots, Y_K)^\top$ with

$$\boxed{Y_k = 1_{\{C=k\}}}$$

For example if $K = 5$ and $c = 4$ then $\boldsymbol{y} = (0, 0, 0, 1, 0)^\top$.

Indicator variable coding for multinomial variables

Let C a r.v. taking values in $\{1, \ldots, K\}$, with

$$\mathbb{P}(C = k) = \pi_k.$$

We will code C with a r.v. $Y = (Y_1, \ldots, Y_K)^\top$ with

$$\boxed{Y_k = 1_{\{C=k\}}}$$

For example if $K = 5$ and $c = 4$ then $\boldsymbol{y} = (0, 0, 0, 1, 0)^\top$.
2mm So $\boldsymbol{y} \in \{0, 1\}^K$ with $\sum_{k=1}^K y_k = 1$.

Indicator variable coding for multinomial variables

Let C a r.v. taking values in $\{1, \ldots, K\}$, with

$$\mathbb{P}(C = k) = \pi_k.$$

We will code C with a r.v. $Y = (Y_1, \ldots, Y_K)^\top$ with

$$\boxed{Y_k = 1_{\{C=k\}}}$$

For example if $K = 5$ and $c = 4$ then $\boldsymbol{y} = (0, 0, 0, 1, 0)^\top$.
2mm So $\boldsymbol{y} \in \{0, 1\}^K$ with $\sum_{k=1}^K y_k = 1$.

$$\mathbb{P}(C = k) = \mathbb{P}(Y_k = 1) \quad \text{and} \quad \mathbb{P}(Y = y) = \prod_{k=1}^K \pi_k^{y_k}.$$

Bernoulli, Binomial, Multinomial

$Y \sim \text{Ber}(\pi)$	$(Y_1, \ldots, Y_K) \sim \mathcal{M}(1, \pi_1, \ldots, \pi_K)$
$p(y) = \pi^y (1-\pi)^{1-y}$	$p(\boldsymbol{y}) = \pi_1^{y_1} \ldots \pi_K^{y_K}$
$N_1 \sim \text{Bin}(n, \pi)$	$(N_1, \ldots, N_K) \sim \mathcal{M}(n, \pi_1, \ldots, \pi_K)$
$p(n_1) = \binom{n}{n_1} \pi^{n_1} (1-\pi)^{n-n_1}$	$p(\mathbf{n}) = \binom{n}{n_1 \quad \ldots \quad n_K} \pi_1^{n_1} \ldots \pi_K^{n_K}$

with

$$\binom{n}{i} = \frac{n!}{(n-i)!i!} \qquad \text{and} \qquad \binom{n}{n_1 \quad \ldots \quad n_K} = \frac{n!}{n_1! \ldots n_K!}$$

Gaussian model

Scalar Gaussian model : $X \sim \mathcal{N}(\mu, \sigma^2)$

X real valued r.v., and $\theta = (\mu, \sigma^2) \in \Theta = \mathbb{R} \times \mathbb{R}_+^*$.

$$p_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Gaussian model

Scalar Gaussian model : $X \sim \mathcal{N}(\mu, \sigma^2)$

X real valued r.v., and $\theta = (\mu, \sigma^2) \in \Theta = \mathbb{R} \times \mathbb{R}_+^*$.

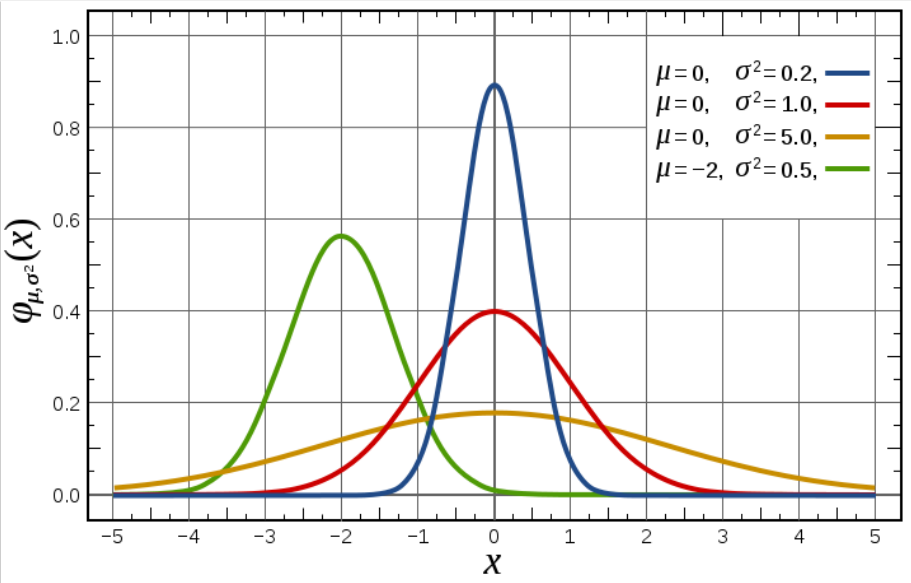
$$p_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Multivariate Gaussian model: $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

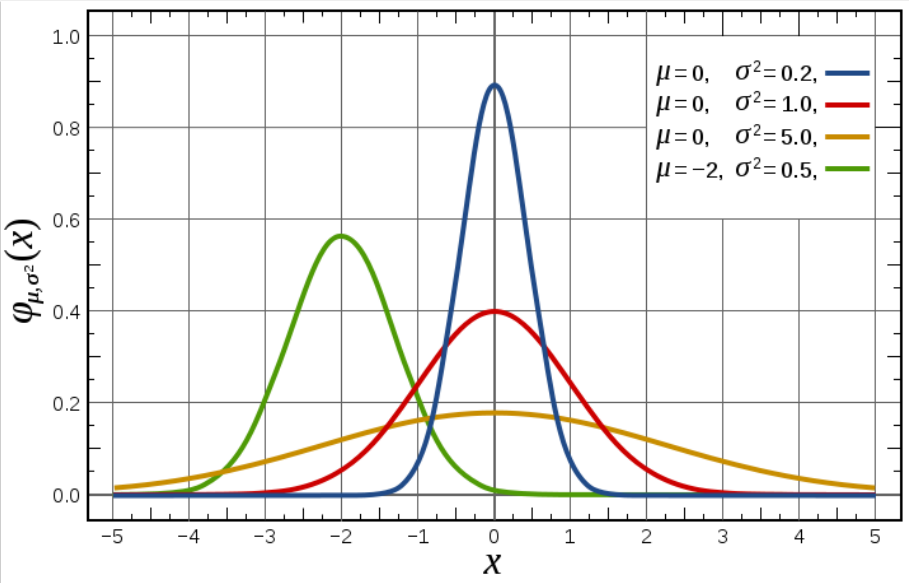
X r.v. taking values in $\mathbb{R}^d$. If $\mathcal{K}_n$ is the set of positive definite matrices of size $n \times n$, and $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma}) \in \Theta = \mathbb{R}^d \times \mathcal{K}_n$.

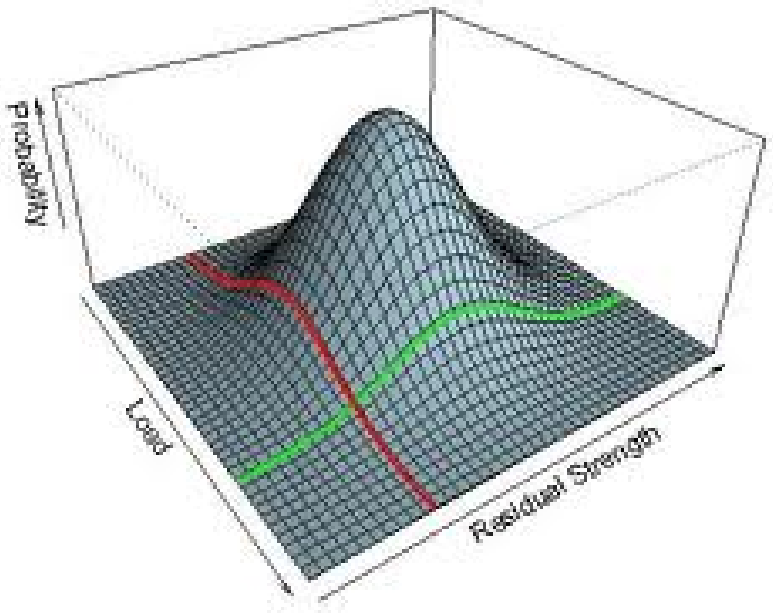
$$p_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Gaussian densities



Gaussian densities





Sample/Training set

The data used to learn or estimate a model typically consists of a collection of observations which are considered as realizations of random variables

$$X^{(1)}, \ldots, X^{(n)}$$

defined on a probability space equipped with the parametric family of distributions.

Sample/Training set

The data used to learn or estimate a model typically consists of a collection of observations which are considered as realizations of random variables

$$X^{(1)}, \ldots, X^{(n)}$$

defined on a probability space equipped with the parametric family of distributions. A common assumption is that the variables are **i.i.d.**

- **independent**
- **identically distributed**, i.e. have the same distribution P.

Sample/Training set

The data used to learn or estimate a model typically consists of a collection of observations which are considered as realizations of random variables

$$X^{(1)}, \ldots, X^{(n)}$$

defined on a probability space equipped with the parametric family of distributions. A common assumption is that the variables are **i.i.d.**

- **independent**
- **identically distributed**, i.e. have the same distribution P.

This collection of observations is called

- the *sample* or the *observations* in statistics
- the *sample***s** in engineering
- the *training set* in machine learning

Outline

Graphical Models

Graphical models offer several useful properties:

1. They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
2. Insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph.
3. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

Graphical Models

1. A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs).

Graphical Models

1. A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs).
2. In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relation- ships between these variables.

Graphical Models

1. A graph comprises nodes (also called vertices) connected by links (also known as edges or arcs).
2. In a probabilistic graphical model, each node represents a random variable (or group of random variables), and the links express probabilistic relation- ships between these variables.
3. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables.

Graphs: Nodes and Edges

- A graph is a data structure $\mathcal{K}$ consisting of a set of nodes and a set of edges.
- Denote by $\mathbf{x} = (x_1, \ldots, x_n)$ the set of nodes. A pair of nodes x_i, x_j can be connected by a directed edge $x_i \rightarrow x_j$ or an undirected edge $x_i - - x_j$.
- The set undirected edge of edges $\mathcal{E}$ is a set of pairs, where each pair is one of $x_i \rightarrow x_j$, $x_j \rightarrow x_i$, or or $x_i - - x_j$, for $x_i, x_j \in \mathcal{X}$, $i < j$.
- We restrict our attention to graphs that contain only edges of one kind. We say that a graph is directed if all edges are either $x_i \rightarrow x_j$ or $x_j \rightarrow x_i$. We say that a graph is undirected if all edges are $x_i - - x_j$.

Graphs: Parents, Childs, Neighbors

- Whenever we have that $x_i \to x_j \in \mathcal{E}$, we say that x_j is the child of x_i in $\mathcal{K}$, and that child x_i is the parent of x_j in $\mathcal{K}$.
- When we have $x_i - - x_j \in \mathcal{E}$, we say that x_i is a neighbor of $x_j \in \mathcal{K}$.
- We say that x and y are adjacent whenever $x \leftrightharpoons y \in \mathcal{E}$.
- We use pa_x to denote the parents of x, ch_x to denote its children, and nb_x to denote its neighbors.

Bayesian Networks / Markov Random Fields

1. We shall first discuss Bayesian networks, also known as directed graphical models, in which the links of the graphs have a particular directionality indicated by arrows.

Bayesian Networks / Markov Random Fields

1. We shall first discuss Bayesian networks, also known as directed graphical models, in which the links of the graphs have a particular directionality indicated by arrows.
2. The other major class of graphical models are Markov random fields, also known as undirected graphical models, in which the links do not carry arrows and have no directional significance.

Bayesian Networks / Markov Random Fields

1. We shall first discuss Bayesian networks, also known as directed graphical models, in which the links of the graphs have a particular directionality indicated by arrows.
2. The other major class of graphical models are Markov random fields, also known as undirected graphical models, in which the links do not carry arrows and have no directional significance.
3. Directed graphs are useful for expressing causal relationships between random variables, whereas undirected graphs are better suited to expressing soft constraints between random variables.

Bayesian Network

1. Consider first an arbitrary joint distribution $p(a, b, c)$ over three variables a, b, and c. Note that at this stage, we do not need to specify anything further about these variables, such as whether they are discrete or continuous. Indeed, one of the powerful aspects of graphical models is that a specific graph can make probabilistic statements for a broad class of distributions.

Bayesian Network

1. Consider first an arbitrary joint distribution $p(a, b, c)$ over three variables a, b, and c. Note that at this stage, we do not need to specify anything further about these variables, such as whether they are discrete or continuous. Indeed, one of the powerful aspects of graphical models is that a specific graph can make probabilistic statements for a broad class of distributions.
2. By application of the Bayes rule

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

Bayesian Network

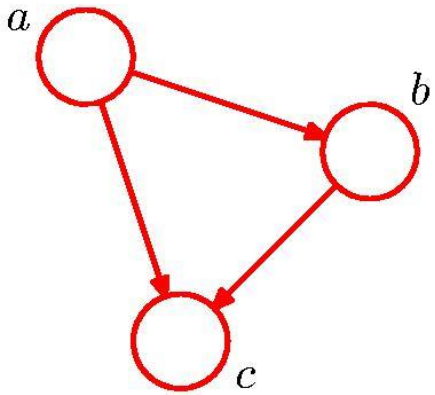
1. Consider first an arbitrary joint distribution $p(a, b, c)$ over three variables a, b, and c. Note that at this stage, we do not need to specify anything further about these variables, such as whether they are discrete or continuous. Indeed, one of the powerful aspects of graphical models is that a specific graph can make probabilistic statements for a broad class of distributions.
2. By application of the Bayes rule

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

3. Note that this decomposition holds for any choice of the joint distribution. We can now represent the right-hand side of in terms of a simple graphical model.

Bayesian Network

Directed Acyclic Graph (DAG)



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

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

Bayesian Network

1. An interesting point to note is that $p(a, b, c)$ is symmetrical with respect to the three variables a, b, and c, whereas the DAG representation $p(c|a, b)p(b|a)p(a)$ is not !.
2. Indeed, in making the DAG, we have implicitly chosen a particular ordering, namely a, b, c, and had we chosen a different ordering we would have obtained a different decomposition and hence a different graphical representation.
3. Take home message: different graphs might encode the same joint distribution.

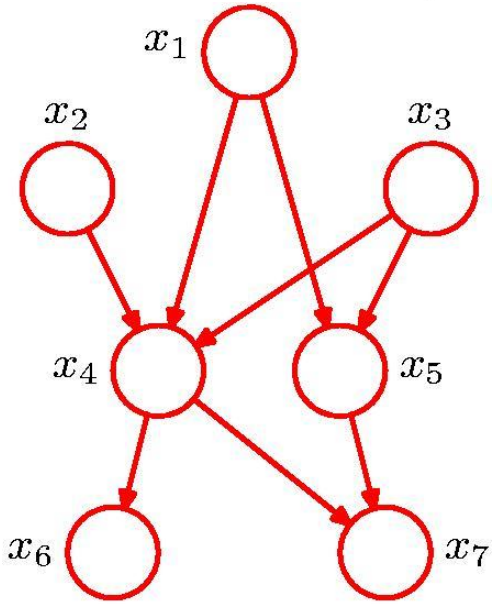
Fully connected DAG

1. Consider the joint distribution over K variables given by $p(x_1, \ldots, x_K)$. By repeated application of the Bayes rule, this joint distribution can be written as a product of conditional distributions, one for each of the variables

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

2. For a given choice of K, we can again represent this as a directed graph having K nodes, one for each conditional distribution, with each node having incoming links from all lower numbered nodes.
3. We say that this graph is fully connected because there is a link between every pair of nodes. It is the absence of links in the graph that conveys interesting information about the properties of the class of distributions that the graph represents.

Not fully connected DAG



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

General Factorization

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

1. The joint distribution defined by a graph is given by the product, over all of the nodes of the graph, of a conditional distribution for each node conditioned on the variables corresponding to the parents of that node in the graph.
2. Thus, for a graph with K nodes, the joint distribution is given by

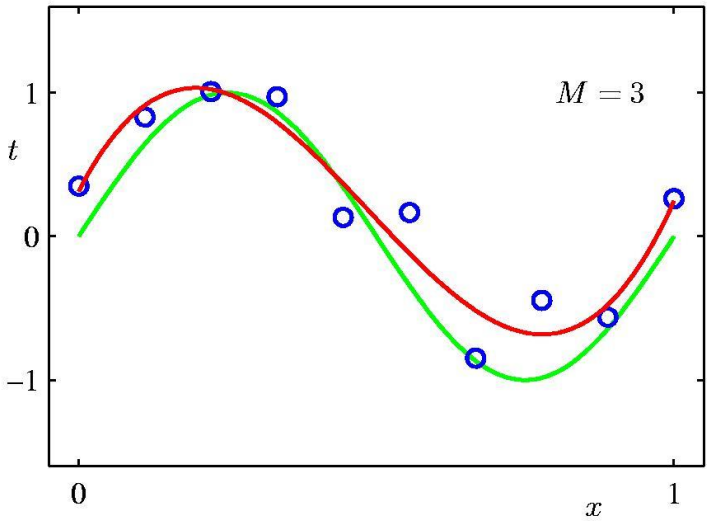
$$p(x_1, \ldots, x_K) = \prod_{k=1}^{K} p(x_k | \mathrm{pa}_k)$$

   where pa_k denotes the set of parents of x_k
3. This key equation expresses the factorization properties of the joint distribution for a directed graphical model.

Directed Acyclic Graphs (DAG)

1. It is easy to show that the representation $\prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$ is always correctly normalized provided the individual conditional distributions are normalized.
2. The directed graphs that we are considering are subject to an important restriction namely that there must be no directed cycles, in other words there are no closed paths within the graph such that we can move from node to node along links following the direction of the arrows and end up back at the starting node.
3. Such graphs are also called directed acyclic graphs, or DAGs.

Bayesian Curve Fitting



Polynomial

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

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

Bayesian Curve Fitting

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

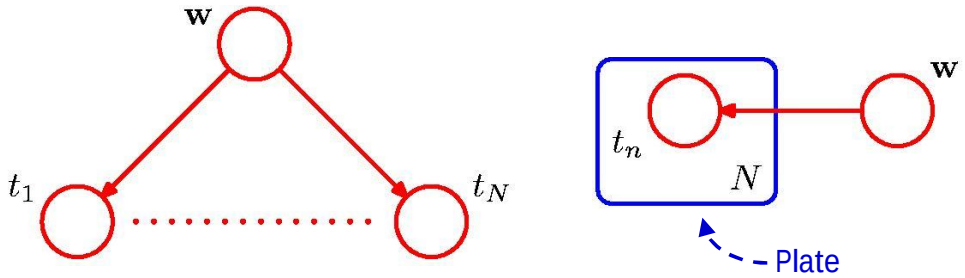


Figure: a draw a single representative node t_n and then surround this with a box, called a plate, labelled with N indicating that there are N nodes of this kind

Bayesian Curve Fitting

Input variables and explicit hyperparameters

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

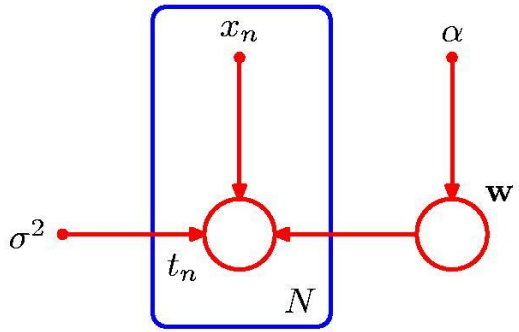


Figure: we shall adopt the convention that random variables will be denoted by open circles, and deterministic parameters will be denoted by smaller solid circles.

Bayesian Curve Fitting

Condition on data

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

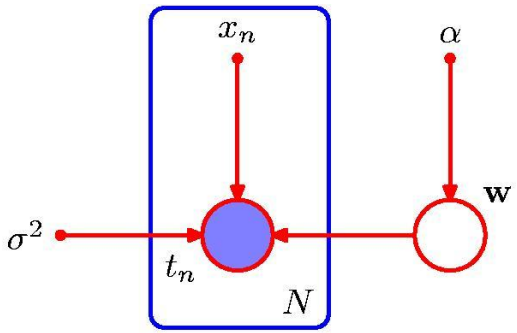
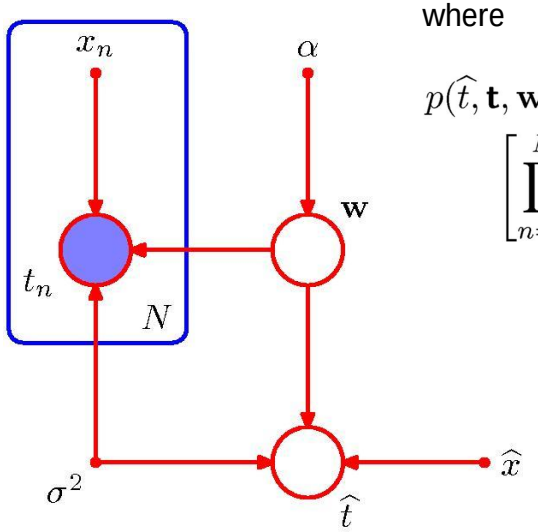


Figure: we will typically set some of the random variables to be observed. In a graphical model, we will denote such observed variables by shading the corresponding nodes.

Bayesian Curve Fitting - Learning

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



where

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

Figure: We denote observed variables by shading the corresponding nodes. Note that the value of w is not observed, and so w is an example of a latent variable or a hidden variable.

Bayesian Curve Fitting - Prediction

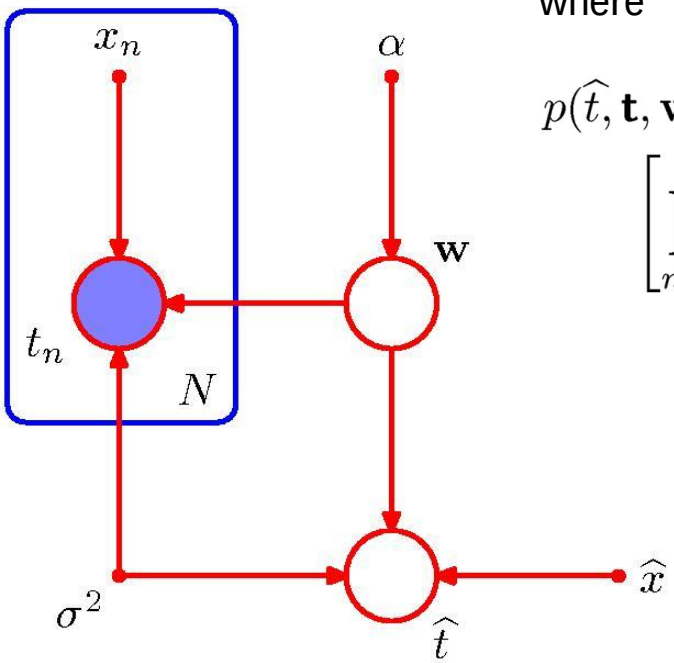
1. Having observed the values $\{t_n\}$ we can evaluate the posterior distribution of the polynomial coefficients w by determining the posterior distribution of this parameter.
2. For the moment, we note that this involves a straightforward application of the Bayes theorem

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

3. In general, model parameters such as w are of little direct interest in themselves, because our ultimate goal is to make predictions for new input values...

Bayesian Curve Fitting - Learning

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



where

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

Generative Models

1. There are many situations in which we wish to draw samples from a given prob- ability distribution.
2. ancestral sampling is particularly relevant to graphical models. Consider a joint distribution $p(x_1, \ldots, x_K)$ over K variables that factorizes according to a DAG
3. Assume that the variables have been ordered such that there are no links from any node to any lower numbered node, in other words each node has a higher number than any of its parents.

Ancestral Sampling

1. We start with the lowest-numbered node and draw a sample from the distribution $p(x_1)$, which we call $\hat{x}_1$.
2. We then work through each of the nodes in order, so that for node n we draw a sample from the conditional distribution $p(x_n|\mathrm{pa}_n)$ in which the parent variables have been set to their sampled values.
3. Once we have sampled from the final variable x_K, we will have achieved our objective of obtaining a sample from the joint distribution.
4. To obtain a sample from some marginal distribution corresponding to a subset of the variables, we simply take the sampled values for the required nodes and ignore the sampled values for the remaining nodes.

Generative Models

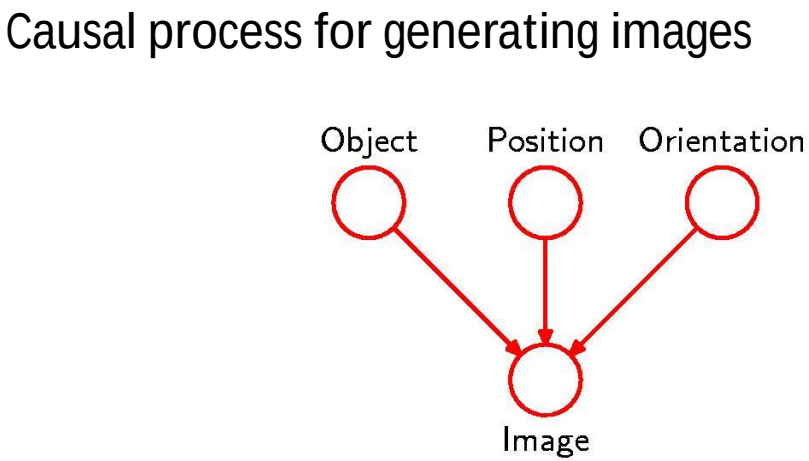


Figure: A graphical model representing the process by which images of objects are created, in which the identity of an object (a discrete variable) and the position and orientation of that object (continuous variables) have independent prior probabilities. The image (a vector of pixel intensities) has a probability distribution that is dependent on the identity of the object as well as on its position and orientation.

Discrete variables

1. The probability distribution $p(x|\mu)$ for a single discrete variable x having K possible states is given by

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

   and is governed by parameters $(\mu_1, \mu_2, \ldots, \mu_K)'$.
2. Due to the constraint $\sum_{k=1}^{K} \mu_k = 1$, only $K - 1$ values for μ_k need to be specified in order to define the distribution

Discrete variables

1. Suppose that we have two discrete variables, x_1 and x_2, each of which has K states, and we wish to model their joint distribution.
2. Denoting the probability of observing both $x_{1k} = 1$ and $x_{2l} = 1$ by the parameter μ_{kl}, the joint distribution is given by

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

3. This distribution is governed by $K^2 - 1$ parameters. The use of the product rule does not help (check !)
4. It is easily seen that the total number of parameters that must be specified for an arbitrary joint distribution over M variables is $K^M - 1$ and therefore grows exponentially with the number M of variables.

Discrete variables

1. Suppose that the variables x_1 and x_2 were independent, corresponding to the graphical model. Each variable is then described by a separate multinomial distribution, and the total number of parameters would be $2(K-1)$.
2. For a distribution over M independent discrete variables, each having K states, the total number of parameters would be $M(K-1)$, which therefore grows linearly with the number of variables.
3. From a graphical perspective, we have reduced the number of parameters by dropping links in the graph, at the expense of having a restricted class of distributions.

Discrete variables

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

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

Independent joint distribution: $2K - 1$ parameters

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

Discrete variables

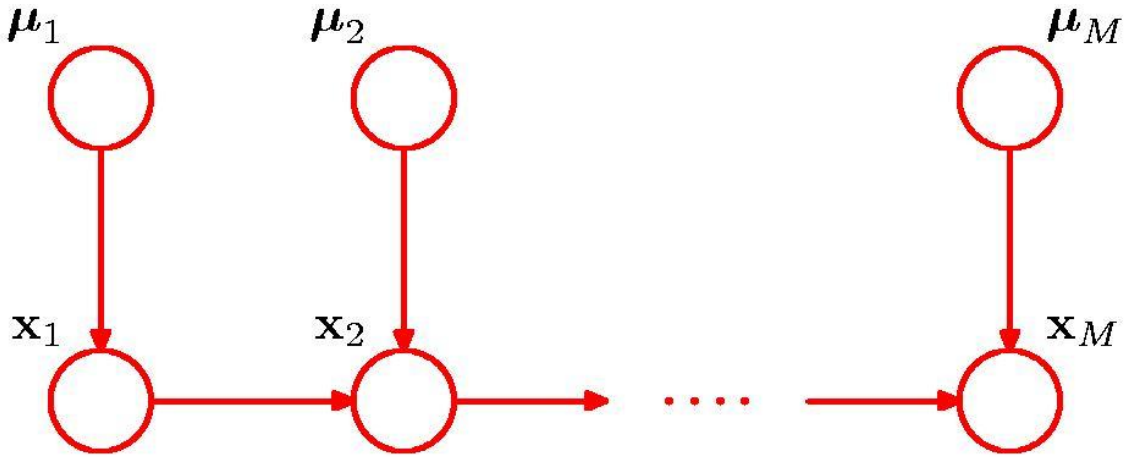
1. If we have M discrete variables $x_1, \ldots, x_M$, we can model the joint distribution using a directed graph with one variable corresponding to each node.
2. The conditional distribution at each node is given by a set of nonnegative parameters subject to the usual normalization constraint.
3. If the graph is fully connected then we have a completely general distribution having $K^M - 1$ parameters
4. if there are no links in the graph the joint distribution factorizes into the product of the marginals, and the total number of parameters is $M(K - 1)$.
5. Graphs having intermediate levels of connectivity allow for more general distributions than the fully factorized one while requiring fewer parameters than the general joint distribution.

Discrete variables: Markov chain

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

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

Discrete variables: tying



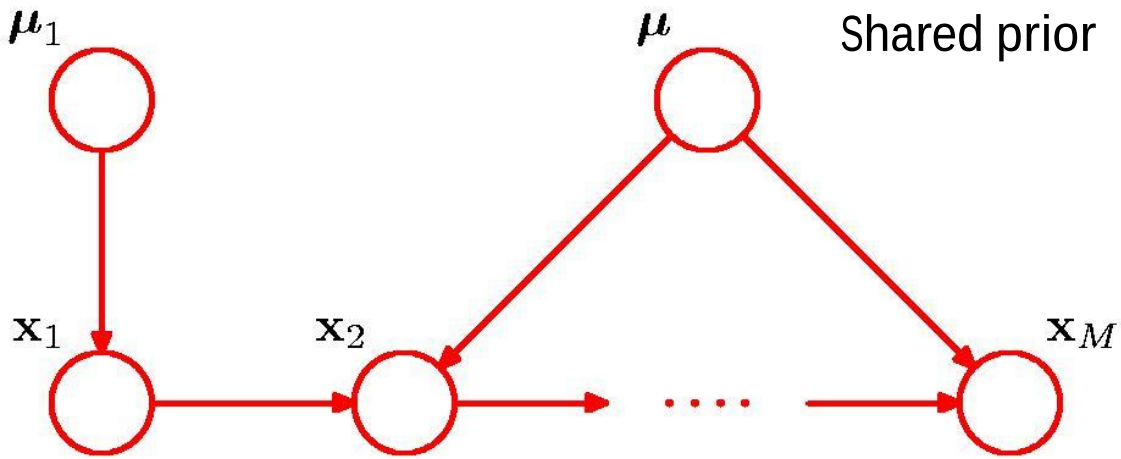
$$p\left(\{\mathbf{x}_m, \boldsymbol{\mu}_m\}\right) = p\left(\mathbf{x}_1 \,|\boldsymbol{\mu}_1\right) p\left(\boldsymbol{\mu}_1\right) \prod_{m=2}^{M} p\left(\mathbf{x}_m|\mathbf{x}_{m-1}, \boldsymbol{\mu}_m\right) p\left(\boldsymbol{\mu}_m\right)$$

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

Tying variables

1. An alternative way to reduce the number of independent parameters in a model is by sharing parameters (also known as tying of parameters).
2. For instance, in the chain example, we can arrange that all of the conditional distributions $p(x_i|x_{i-1})$, for $i = 2, \ldots, M$, are governed by the same set of $K(K-1)$ parameters.
3. Together with the $K-1$ parameters governing the distribution of x_1, this gives a total of $K^2 - 1$ parameters that must be specified in order to define the joint distribution.

Discrete variables: tying



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

Discrete variables: parametric models

1. Another way of controlling the exponential growth in the number of parameters in models of discrete variables is to use parameterized models for the conditional distributions instead of complete tables of conditional probability values.
2. assume that we are willing to model $p(y, x_1, \ldots, x_M)$ where x_i represent binary variables. Each of the parent variables x_i is governed by a single parameter μ_i, giving M parameters in total for the parents mode.
3. The conditional distribution $p(y|x_1, \ldots, x_M)$ would require 2^M parameters..

Discrete variables: parametric models

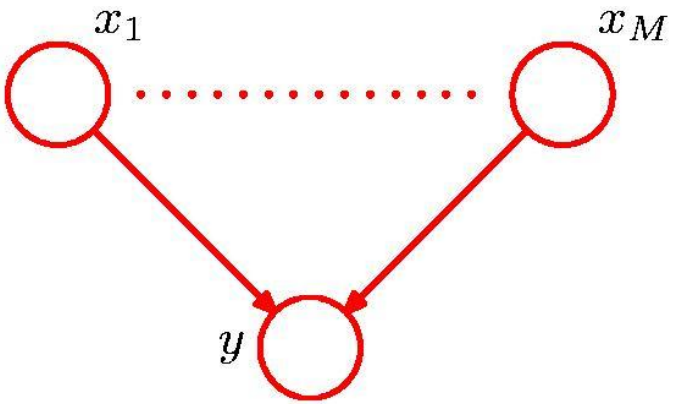
1. A much more parsimonious representation can be obtained by using a parametric form:

$$p(y|x_1, \ldots, x_M) = \sigma(w_0 + \sum_{i=1}^{M} w_i x_i) = \sigma(\mathbf{w}'\mathbf{x})$$

   where σ is the sigmoid function.

2. This is a more restricted form of conditional distribution than the general case but is now governed by a number of parameters that grows linearly with M.

Discrete variables



If $x_1, \ldots, x_M$ are discrete, K-state variables, $p(y = 1|x_1, \ldots, x_M)$ in general has $O(K^M)$ parameters.

The parameterized form

$$p(y = 1|x_1, \ldots, x_M) = \sigma\left(w_0 + \sum_{i=1}^{M} w_i x_i\right) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

requires only M + 1 parameters

Linear Gaussian Process

1. Consider an arbitrary directed acyclic graph over D variables in which node i represents a single continuous random variable x_i having a Gaussian distribution.
2. The mean of this distribution is taken to be a linear combination of the states of its parent nodes pa_i of node i

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

where $w_{i,j}$ and b_i are parameters governing the mean, and v_i is the variance of the conditional distribution for x_i.

Linear Gaussian Process

1. The log of the joint distribution is then the log of the product of these conditionals over all nodes in the graph and hence takes the form

$$\log p(\mathbf{x}) = \sum_{i=1}^{D} \log p(x_i|\mathrm{pa}_i)$$

$$= -\sum_{i=1}^{D} \frac{1}{2v_i} \left(x_i - \sum_{j \in \mathrm{pa}_i} w_{i,j} x_j - b_j \right)^2 + C$$

   which is a quadratic function of the components of $\mathbf{x}$.

2. Take-home message the joint distribution $p(\mathbf{x})$ is a multivariate Gaussian.

Linear Gaussian model

1. We can find the components of $\mathbb{E}[x] = (\mathbb{E}[x_1], \ldots, \mathbb{E}[x_D])'$ by starting at the lowest numbered node and working recursively through the graph (here we again assume that the nodes are numbered such that each node has a higher number than its parents).
2. Similarly, we can compute the covariance matrix for $\mathbf{x}$ in the form of a recursion relation

$$\mathrm{Cov}(x_i, x_j) = \sum_{k \in \mathrm{pa}_j} w_{j,k} \mathrm{Cov}(x_i, x_k) + \delta_{i,j} v_j$$

and so the covariance can similarly be evaluated recursively starting from the lowest numbered node.

Outline

Definition

1. Consider three variables a, b, and c, and suppose that the

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

   We say that a is conditionally independent of b given c.

2. This can be expressed in a slightly different way if we consider the joint distribution of a and b conditioned on c,

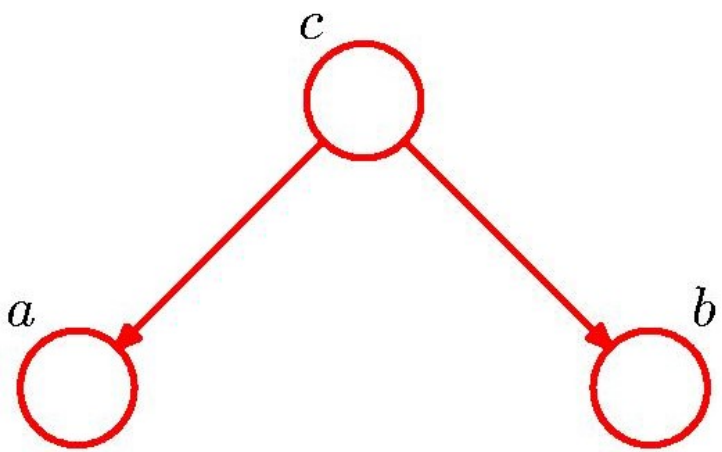
$$p(a,b|c) = p(a|b,c)p(b|c) = p(a|c)p(b|c)$$

3. Notation $a \perp b|c$.

d-separation

1. An important and elegant feature of graphical models is that conditional independence properties of the joint distribution can be read directly from the graph without having to perform any analytical manipulations.
2. The general framework for achieving this is called *d*-separation, where the "*d*" stands for "directed"
3. In the sequel, we shall motivate the concept of d-separation and give a general statement of the d-separation criterion.

Conditional Independence

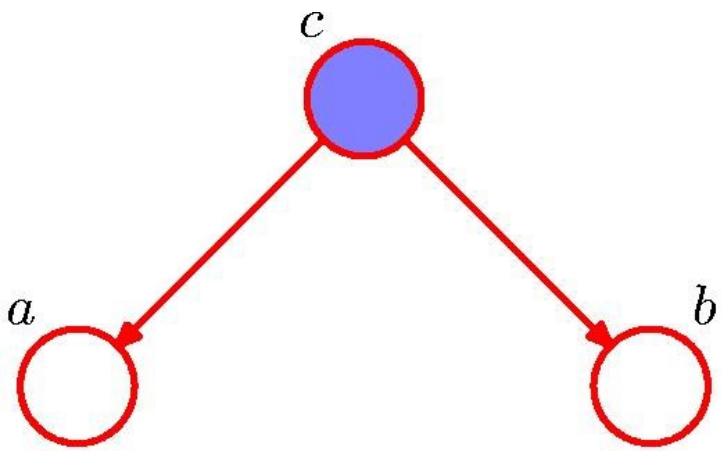


$$p(a, b, c) = p(a|c)p(b|c)p(c)$$

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

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

Conditional Independence



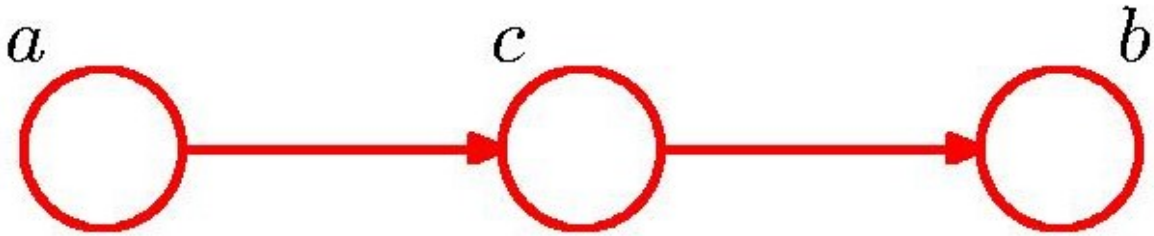
$$
\begin{aligned}
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= p(a|c)p(b|c)
\end{aligned}
$$

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

Tail-to-tail

1. The node c is said to be tail-to-tail with respect to this path because the node is connected to the tails of the two arrows, and the presence of such a path connecting nodes a and b causes these nodes to be dependent.
2. However, when we condition on node c, the conditioned node blocks the path from a to b and causes a and b to become (conditionally) independent.

Conditional Independence

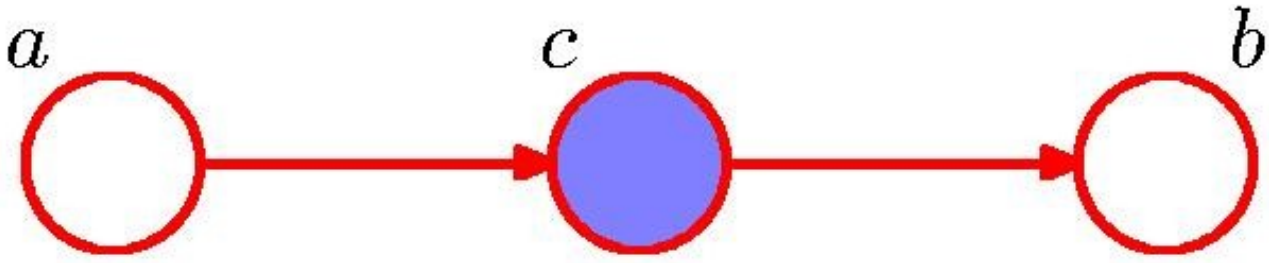


$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

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

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

Conditional Independence



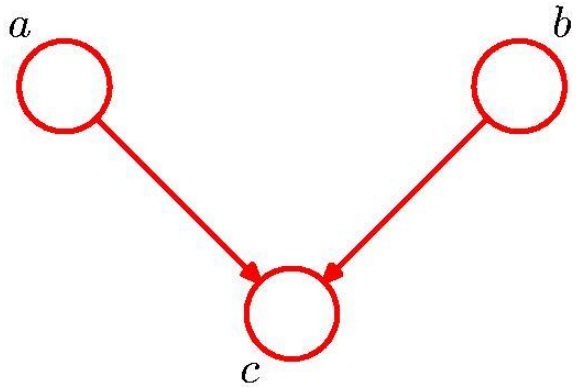
$$
\begin{aligned}
p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
&= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\
&= p(a|c)p(b|c)
\end{aligned}
$$

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

Head-to-tail

1. The node c is said to be head-to-tail with respect to the path from node a to node b. Such a path connects nodes a and b and renders them dependent.
2. If we now observe c, then this observation blocks the path from a to b and so we obtain the conditional independence property $a \perp b|c$.

Conditional Independence

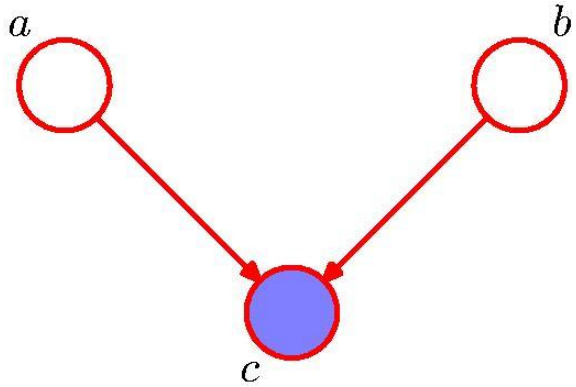


$$p(a, b, c) = p(a)p(b)p(c|a, b)$$

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

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

Conditional Independence



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

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

Conditional independence: head-to-head

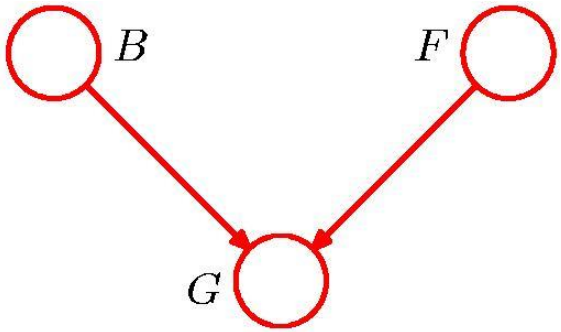
1. Graphically, we say that node c is head-to-head with respect to the path from a to b because it connects to the heads of the two arrows.
2. When node c is unobserved, it blocks the path, and the variables a and b are independent.
3. However, conditioning on c unblocks the path and renders a and b dependent.

Summary

1. a tail-to-tail node or a head-to-tail node leaves a path unblocked unless it is observed in which case it blocks the path.
2. By contrast, a head-to-head node blocks a path if it is unobserved, but once the node, and/or at least one of its descendants, is observed the path becomes unblocked.

Conditional Independence

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



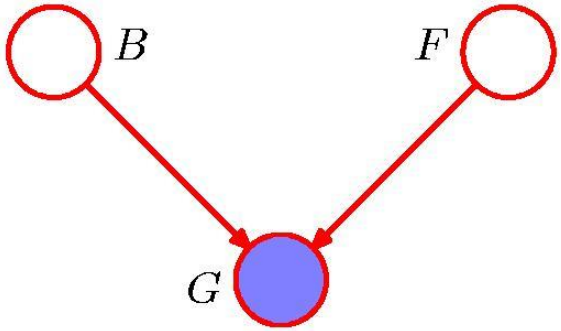
$$
\begin{aligned}
p(B=1) &= 0.9 \\
p(F=1) &= 0.9
\end{aligned}
$$

and hence

$$
p(F=0) = 0.1
$$

B = Battery (0=flat, 1=fully charged)
F = Fuel Tank (0=empty, 1=full)
G = Fuel Gauge Reading (0=empty, 1=full)

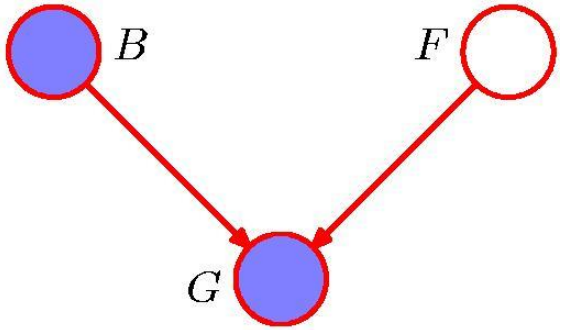
Conditional Independence



$$
\begin{aligned}
p(F=0|G=0) &= \frac{p(G=0|F=0)p(F=0)}{p(G=0)} \\
&\simeq 0.257
\end{aligned}
$$

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

Conditional Independence



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

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

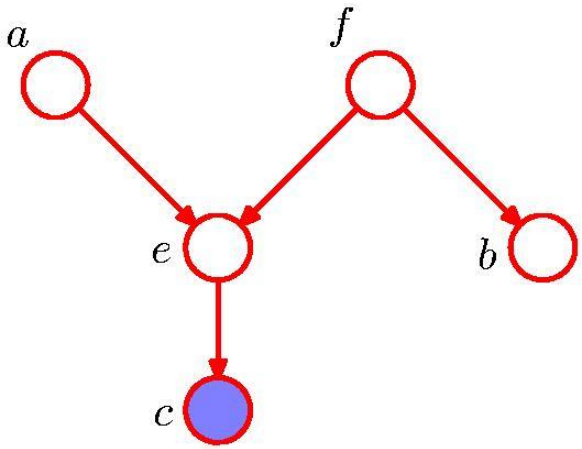
d-separation

- A, B, and C are non-intersecting subsets of nodes in a directed graph.
- A path from A to B is blocked if it contains a node such that either

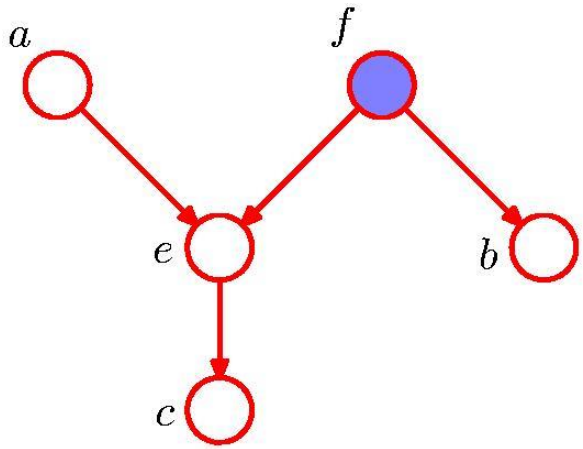
  (a) the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C, or
  (b) the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in the set C.

- If all paths from A to B are blocked, A is said to be d-separated from B by C.
- If A is d-separated from B by C, the joint distribution over all variables in the graph satisfies $A \perp B | C$.

d-separation: example

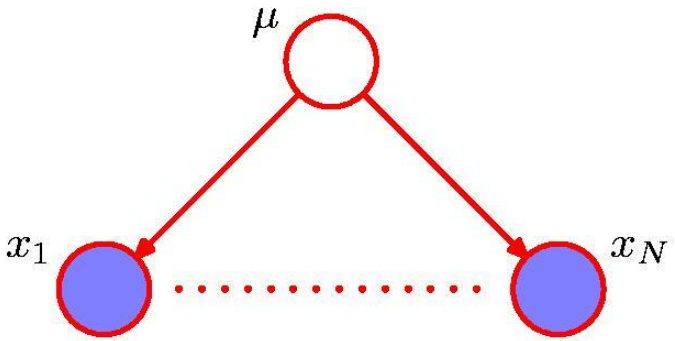


$a \not\perp\!\!\!\perp b \mid c$



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

d-separation: I.I.D.



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

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

Outline

Markov Random Field / Network

- A directed graphical models specify a factorization of the joint distribution over a set of variables into a product of local conditional distributions.
- They also define a set of conditional independence properties that must be satisfied by any distribution that factorizes according to the graph.
- We turn now to the second major class of graphical models that are described by undirected graphs and that again specify both a factorization and a set of conditional independence relations.

Markov Random Field / Network

- A Markov random field, also known as a Markov Network or an undirected graphical model 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.
- In the case of undirected graphs, it is convenient to begin with a discussion of conditional independence properties.

Markov random Field / Network

- In the case of directed graphs, we saw that it was possible to test whether a particular conditional independence property holds by applying a graphical test called d-separation.
- This involved testing whether or not the paths connecting two sets of nodes were blocked. The definition of blocked, however, was somewhat subtle due to the presence of paths having head-to-head nodes.
- We might ask whether it is possible to define an alternative graphical semantics for probability distributions such that conditional independence is determined by simple graph separation.
- By removing the directionality from the links of the graph, the asymmetry between parent and child nodes is removed,and so the subtleties associated with head-to-head nodes no longer arise.

Markov Random Field / Network

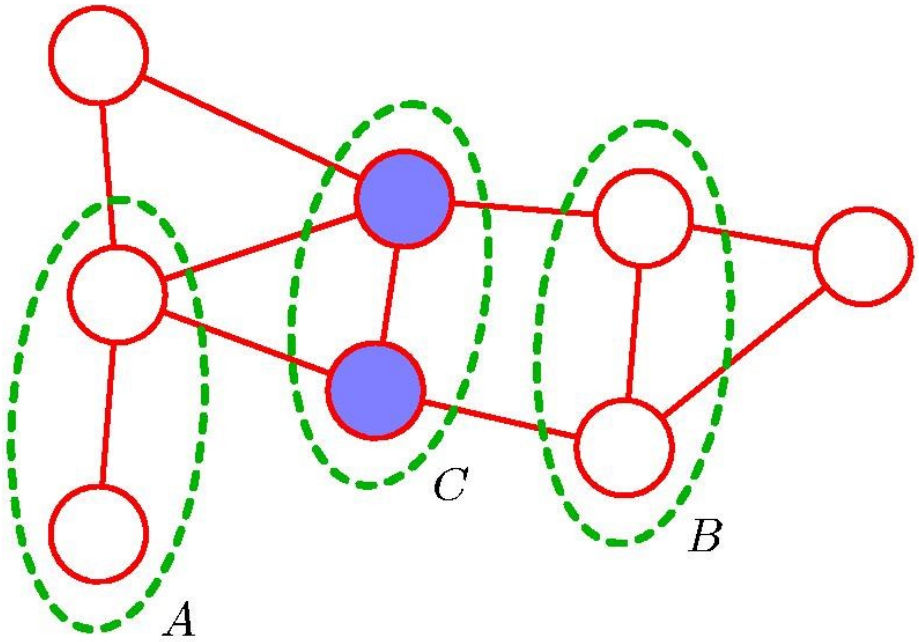


Figure: Every path from any node in set A to any node in set B passes through at least one node in set C. Consequently the conditional independence property $A \perp B|C$ holds for any probability distribution described by this graph.

Factorization properties

1. We now seek a factorization rule for undirected graphs that will correspond to the above conditional independence test.
2. If we consider two nodes x_i and x_j that are not connected by a link, then these variables must be conditionally independent given all other nodes in the graph.
3. This conditional independence property can be expressed as

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

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

Factorization properties

1. The factorization of the joint distribution must therefore be such that x_i and x_j do not appear in the same factor in order for the conditional independence property to hold for all possible distributions belonging to the graph.
2. This leads us to consider a graphical concept called a clique, which is defined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset.
3. A maximal clique is 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.

Maximal Clique

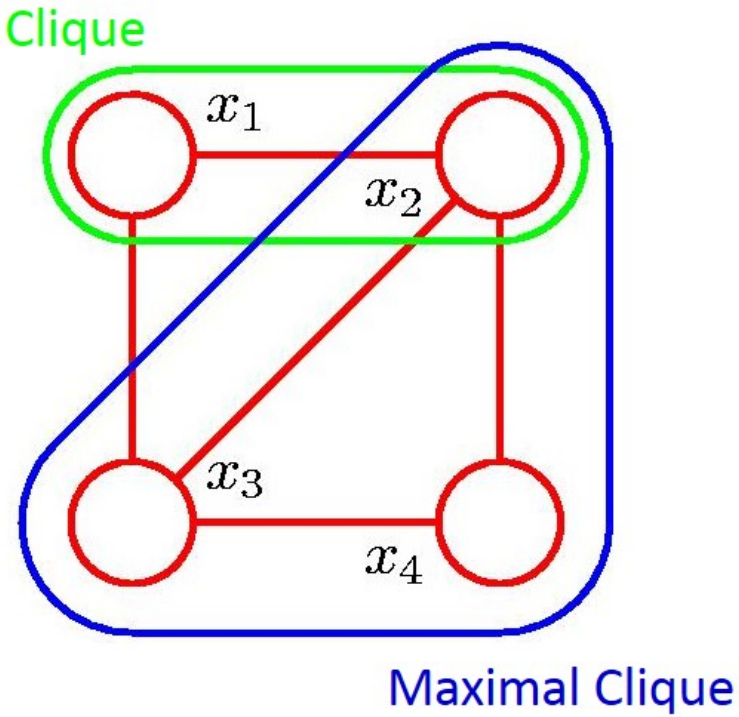


Figure: This graph has five cliques of two nodes given by $\{x_1, x_2\}$, $\{x_2, x_3\}$, $\{x_3, x_4\}$, $\{x_4, x_2\}$, and $\{x_1, x_3\}$, as well as two maximal cliques given by $\{x_1, x_2, x_3\}$ and $\{x_2, x_3, x_4\}$. The set $\{x_1, x_2, x_3, x_4\}$ is not a clique because of the missing link from x_1 to x_4

Joint Distribution

- Let us denote a clique by C and the set of variables in that clique by $\mathbf{x}_C$. Then the joint distribution is written as a product of potential functions $\psi_C(\mathbf{x}_C)$ over the maximal cliques of the graph

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

- The factors $\{\psi_C(\mathbf{x}_C)\}$ that parameterize the Markov Network are called the clique potential.
- Here, the quantity Z is the normalizing constant or partition function.

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

which ensures that $p(\mathbf{x})$ is properly normalized.

Factorization

- We can more generally define the factors in the decomposition of the joint distribution to be functions of the variables in cliques (not necessarily maximal cliques).

$$p(\mathbf{x}) = Z^{-1} \prod_D \psi_D(\mathbf{x}_D) \,,$$

  where the product is over a set of cliques $\{D_i\}_{i=1}^K$.
- Note that, because every clique is a subset of some maximal clique, we can reduce the number of factors in our parameterization by allowing factors only for maximal cliques.
- More precisely, let $C_1, \ldots, C_k$ be the cliques in $\mathcal{H}$. We can parameterize p using a set of factors $\{\psi_{C_i}(\mathbf{x}_{C_i})\}_{i=1}^k$: Any factorization in terms of clique can be converted into this form simply by assigning each factor to a maximal clique and multiplying all of the factors assigned to each maximal clique.

Maximal clique / clique potentials

- Although it can be used without loss of generality, the parametrization using maximal clique potentials generally obscures structure that is present in the original set of factors.
- For example, consider a fully connected graph over $\mathcal{X}$; in this case, the graph specifies no conditional independence assumptions.
- If all of the variables are binary, each factor over an edge would have 4 parameters, and the total number of parameters in the graph would be $4\binom{n}{2}$.
- However, the number of parameters required to specify a joint distribution over n binary variables is $2^n - 1$.
- Thus, pairwise factors simply do not have enough parameters to encompass the space of joint distributions: only the pairwise interactions, and not interactions that involve combinations of values of larger subsets of variables.

Maximal clique / clique potentials

- Here, we have a potential for every pair of variables, so the Markov network associated with this distribution is a single large clique containing all variables.
- If we associate a factor with this single clique, it would be exponentially large in the number of variables, whereas the original parametrization in terms of edges requires only a quadratic number of parameters.
- Hence, even if using the decomposition of the joint distribution in terms of maximal clique potential is minimal (in terms of the number of factors), the parametrization of the joint distribution in terms of pairs is more convenient.

Joint Distribution

- Note that we do not restrict the choice of potential functions to those that have a specific probabilistic interpretation as marginal or conditional distributions.
- This is in contrast to directed graphs in which each factor represents the conditional distribution of the corresponding variable, conditioned on the state of its parents.
- However, in special cases, for instance where the undirected graph is constructed by starting with a directed graph, the potential functions may indeed have such an interpretation.

Joint Distribution

- One consequence of the generality of the potential functions $\psi_C(\mathbf{x}_C)$ is that their product will in general not be correctly normalized.
- We therefore have to introduce an explicit normalization factor. Recall that for directed graphs, the joint distribution is automatically normalized as a consequence of the normalization of each of the conditional distributions in the factorization.
- The presence of this normalization constant is one of the major limitations of undirected graphs. If we have a model with M discrete nodes each having K states, then the evaluation of the normalization term involves summing over K^M states and so (in the worst case) is exponential in the size of the model.

Hammersley-Clifford Theorem

- Consider the set of all possible distributions defined over a fixed set of variables corresponding to the nodes of a particular undirected graph.
- We can define U I to be 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.
- Similarly, we can define UF to be the set of such distributions that can be expressed as a factorization of the form $p(\mathbf{x}) = \prod_C \psi_C(\mathbf{x}_C)$ with respect to the maximal cliques of the graph.
- The Hammersley-Clifford theorem (Clifford, 1990) states that the sets UI and UF are identical.

Energy functions

- Because we are restricted to potential functions which are strictly positive it is convenient to express them as exponentials, so that

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

  where $E_C(\mathbf{x}_C)$ is called an energy function, and the exponential representation is called the Boltzmann distribution.
- The joint distribution is defined as the product of potentials, and so the total energy is obtained by adding the energies of each of the maximal cliques.

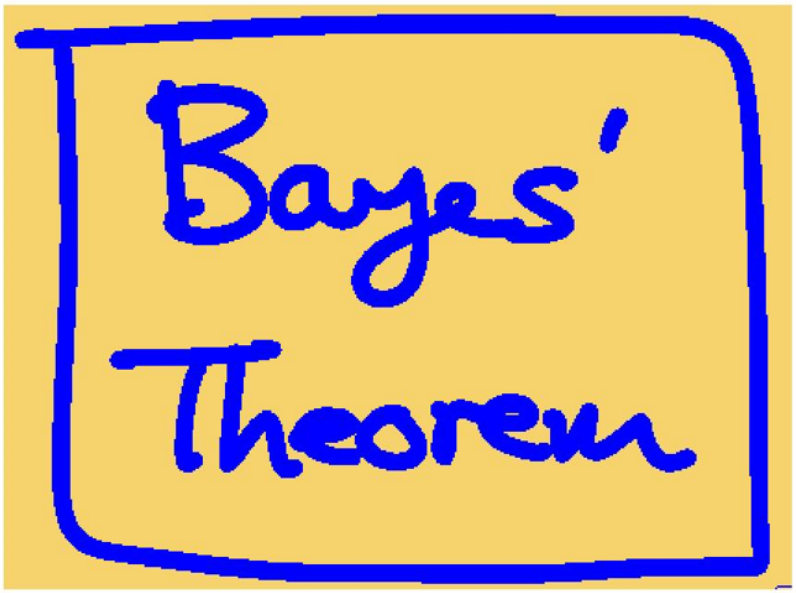
Energy functions

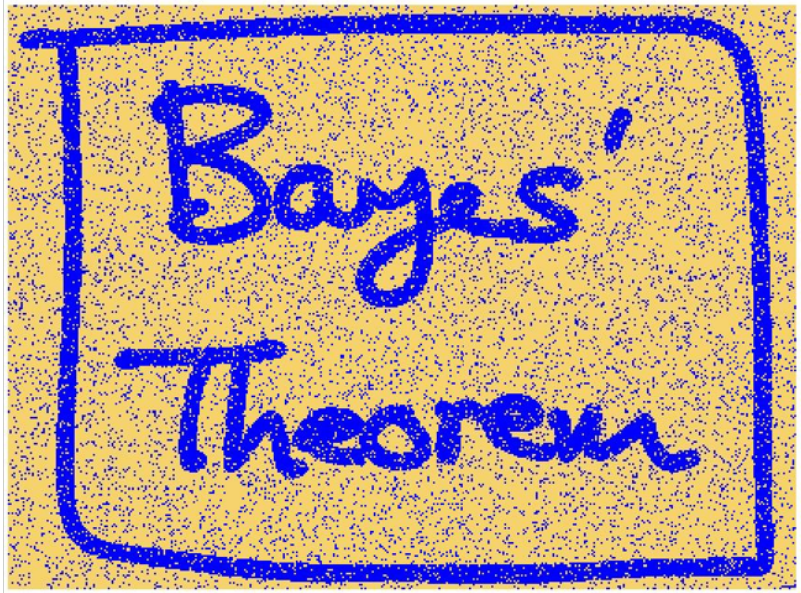
- The use of the word energy derives from statistical physics, where the probability of a physical state (for example, a configuration of a set of electrons), depends inversely on its energy.
- In contrast to the factors in the joint distribution for a directed graph, the potentials in an undirected graph do not have a specific probabilistic interpretation.
- Although this gives greater flexibility in choosing the potential functions it does raise the question of how to motivate a choice of potential function for a particular application.

Image Denoising

- Let the observed noisy image be described by an array of binary pixel values $y_i \in \{-1, +1\}$, where the index $i = 1, \ldots, D$ runs over all pixels.
- We shall suppose that the image is obtained by taking an unknown noise-free image, described by binary pixel values $x_i \in \{-1, +1\}$ and randomly flipping the sign of pixels with some small probability.

Image Denoising





- Because the noise level is small, we know that there will be a strong dependence between x_i and y_i.
- We also know that neighbouring pixels x_i and x_j in an image are strongly dependent.
- This prior knowledge can be captured using the Markov random field model ...

Image Denoising

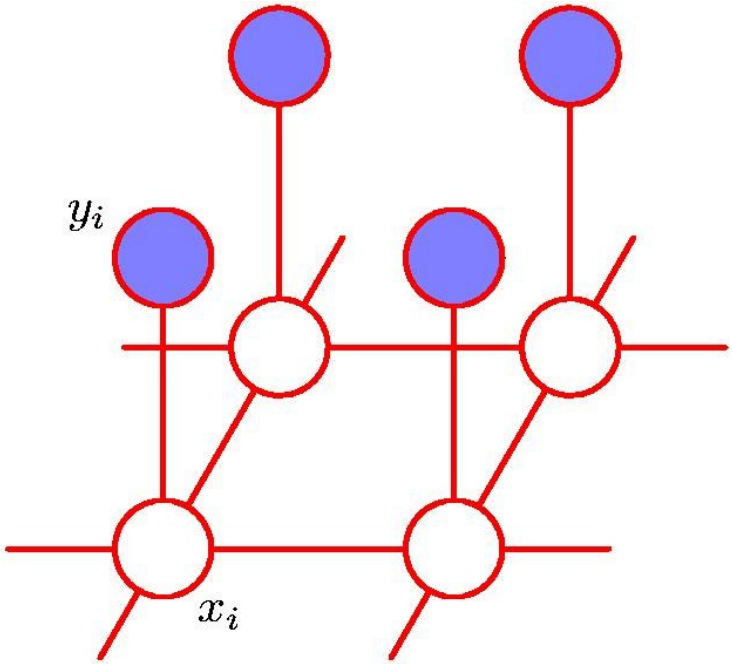


Figure: An undirected graphical model representing a Markov random field for image denoising, in which x_i is a binary variable denoting the state of pixel i in the unknown noise-free image, and y_i denotes the corresponding value of pixel i in the observed noisy image. The graph has two types of cliques, each of which contains two variables.

Energy functions

- The cliques of the form $\{x_i, y_i\}$ have an associated energy function that expresses the dependence between these variables.
- We choose a very simple energy function for these cliques of the form $-\eta x_i y_i$ where η is a positive constant.
- This has the desired effect of giving a lower energy (thus encouraging a higher probability) when x_i and y_i have the same sign and a higher energy when they have the opposite sign.

Energy functions

- The remaining cliques comprise pairs of variables $\{x_i, x_j\}$ where i and j are indices of neighbouring pixels.
- Again, we want the energy to be lower when the pixels have the same sign than when they have the opposite sign, and so we choose an energy given by $-\beta x_i x_j$ where β is a positive constant.

Energy functions

- Because a potential function is an arbitrary, nonnegative function over a maximal clique, we can multiply it by any nonnegative functions of subsets of the clique, or equivalently we can add the corresponding energies.
- In this example, this allows us to add an extra term hx_i for each pixel i in the noise-free image. Such a term has the effect of biasing the model towards pixel values that have one particular sign in preference to the other.
- The complete energy function for the model then takes the form

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

Posterior mode

- We now fix the elements of $\boldsymbol{y}$ to the observed values given by the pixels of the noisy image, which implicitly defines a conditional distribution $p(\mathbf{x}|\boldsymbol{y})$ over noise- free images.
- This is an example of the Ising model, which has been widely studied in statistical physics.
- For the purposes of image restoration, we wish to find an image $\mathbf{x}$ which maximizes the posterior distribution $p(\mathbf{x}|\boldsymbol{y})$.
- To do this we shall use a simple iterative technique called iterated conditional modes, or ICM which is simply an application of coordinate-wise gradient ascent.

Ising Models and Boltzmann Machines

- One of the earliest types of Markov network models is the Ising model, which first arose as a model for the energy of a physical system involving a system of interacting atoms.
- In these systems, each atom is associated with a binary random variable $x_i \in \{+1, -1\}$, whose value defines the direction of the atom's spin with

$$p(\mathbf{x}) = Z^{-1} \exp\left(-\sum_{i<j} w_{i,j} x_i x_j - \sum_i h_i x_i\right)$$

- The energy function is symmetric in x_i, x_j ; it makes a contribution of $w_{i,j}$ to the energy function when $x_i = x_j$ (so both atoms have the same spin) and a contribution of $-w_{i,j}$ otherwise.
- The Ising model also contains a set of parameters h_i to bias individual variables to have one spin or another.

Ising Models

- When $w_{i,j} > 0$ the model prefers to align the spins of the two atoms; in this case, the interaction is called ferromagnetic. When $w_{i,j} < 0$ the interaction is called antiferromagnetic. When $w_{i,j} = 0$ the atoms are non-interacting.
- Much work has gone into studying particular types of Ising models, attempting to answer a variety of questions, usually as the number of atoms goes to infinity. For example, we might ask the probability of a configuration in which a majority of the spins are $+1$ or -1, versus the probability of more mixed configurations.
- The answer to this question depends heavily on the strength of the interaction between the variables; These questions, and many others, have been investigated extensively by physicists, and the answers are known (in some cases even analytically) for several cases.

Iterated Conditional Modes (ICM)

- The idea is first to initialize the variables $\{x_i\}$, which we do by simply setting $x_i = y_i$ for every i.
- Then we take one node x_j at a time and we evaluate the total energy for the two possible states $x_j = +1$ and $x_j = -1$, keeping all other node variables fixed, and set x_j to whichever state has the lower energy.
- This will either leave the probability unchanged, if x_j is unchanged, or will increase it. Because only one variable is changed, this is a simple local computation that can be performed efficiently.
- We then repeat the update for another site, and so on, until some suitable stopping criterion is satisfied.

Iterated Conditional Modes

- The nodes may be updated in a systematic way, for instance by repeatedly raster scanning through the image, or by choosing nodes at random.
- If we have a sequence of updates in which every site is visited at least once, and in which no changes to the variables are made, then by definition the algorithm will have converged to a local maximum of the probability.
- Beware! This need not, however, correspond to the global maximum !

Image Denoising

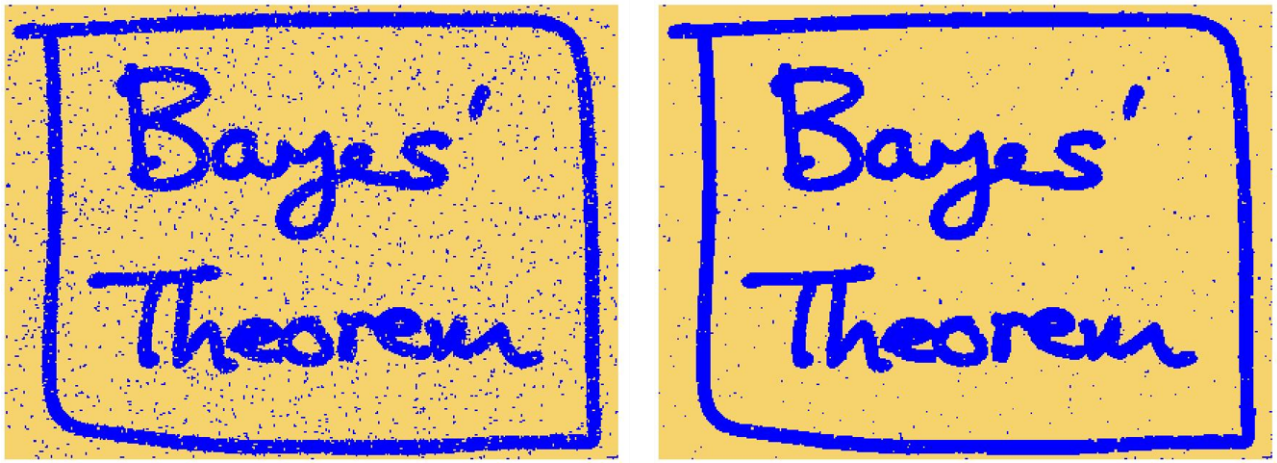


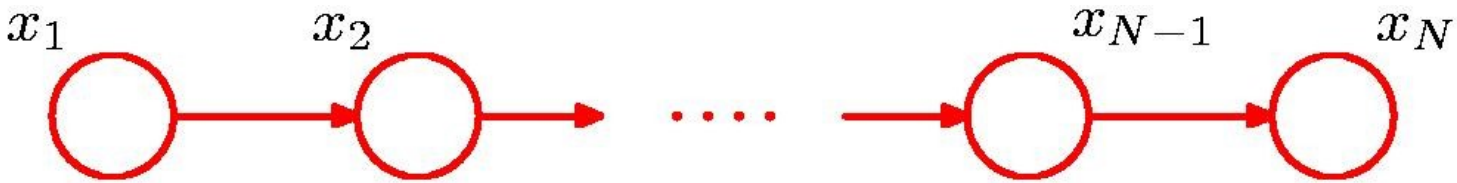
Figure: For the purposes of this simple illustration, we have fixed the parameters to be $\beta = 1.0$, $\eta = 2.1$ and $h = 0$. Note that leaving $h = 0$ simply means that the prior probabilities of the two states of x_i are equal.

Relations between Directed and Undirected graphs

Consider the problem of taking a model that is specified using a directed graph and trying to convert it to an undirected graph. In some cases this is straightforward... For example, consider the following joint distribution

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

whose DAG representation is

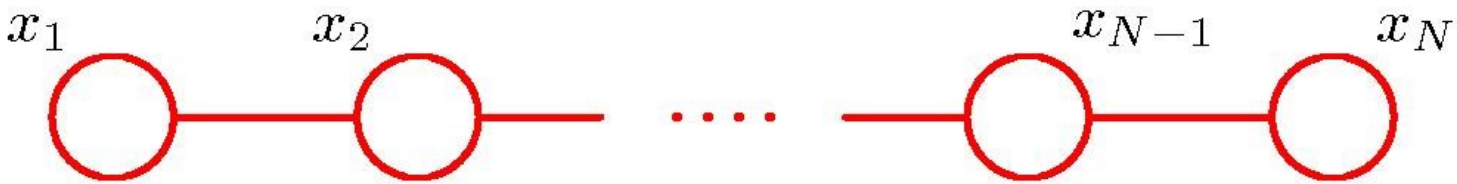


Relations between Directed and Undirected graphs

In the undirected graph, the maximal cliques are simply the pairs of neighbouring nodes, and the joint distribution may be written as

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

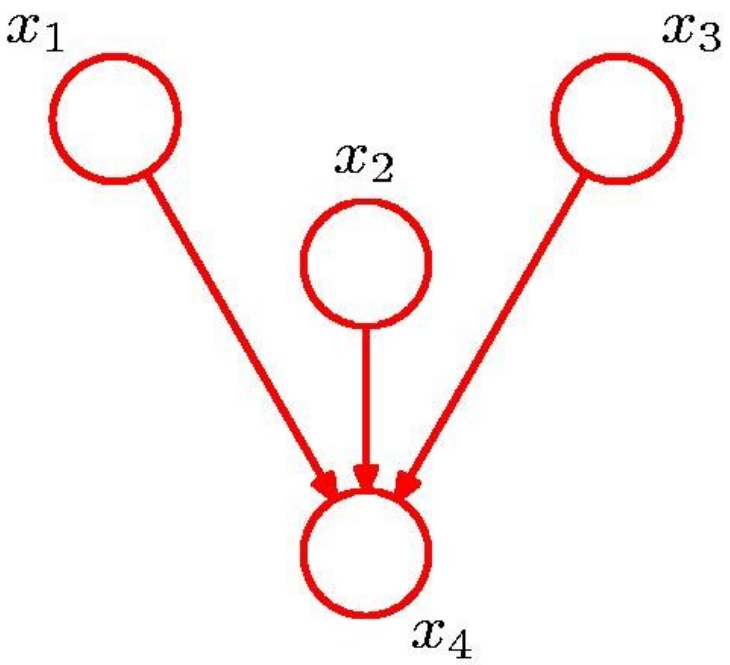
This is easily done by identifying $\psi_{1,2}(x_1, x_2) = p(x_1)p(x_2|x_1)$ and for $j > 1$, $\psi_{j,j+1}(x_j, x_{j+1}) = p(x_{j+1}|x_j)$. In this case, the partition function $Z = 1$.



Generalizations

- Problem convert any distribution specified by a factorization over a directed graph into one specified by a factorization over an undirected graph.
- This can be achieved if the clique potentials of the undirected graph are given by the conditional distributions of the directed graph. In order for this to be valid, we must ensure that the set of variables that appears in each of the conditional distributions is a member of at least one clique of the undirected graph.
- For nodes on the directed graph having just one parent, this is achieved simply by replacing the directed link with an undirected link (previous example).
- However, for nodes in the directed graph having more than one parent, this is not sufficient. These nodes are the nodes that have head-to-head dependence !

An example

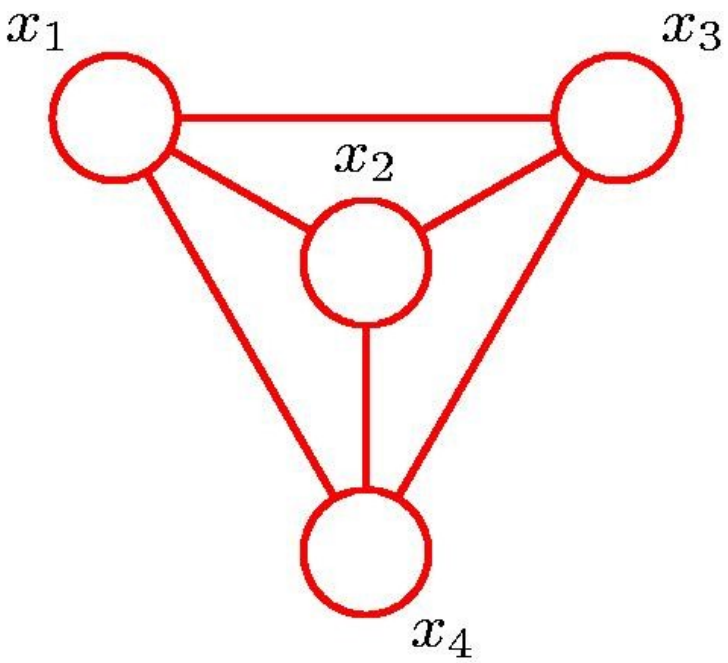


The joint distribution for the directed graph takes the form

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

We see that the factor $p(x_4|x_1, x_2, x_3)$ involves the four variables $\{x_i\}_{i=1}^4$ and so these must all belong to a single clique if this conditional distribution is to be absorbed into a clique potential.

An example



To ensure this, we add extra links between all pairs of parents of the node x_4. Anachronistically, this process of marrying the parents has become known as moralization, and the resulting undirected graph, after dropping the arrows, is called the moral graph. It is important to observe that the moral graph in this example is fully connected and so exhibits no conditional independence properties, in contrast to the original directed graph.

Moral Graph

- To convert a directed graph into an undirected graph, we first add additional undirected links between all pairs of parents for each node in the graph and then drop the arrows on the original links to give the moral graph.
- More formally: The moral graph $\mathcal{M}[\mathcal{G}]$ of a Bayesian network structure $\mathcal{G}$ over $\mathcal{X}$ is the undirected graph over $\mathcal{X}$ that contains an undirected edge between x and x' if:
  (a) there is a directed edge between x and x' (in either direction), or
  (b) the nodes x and x' are both parents of the same node.

Moral Graph

- Let us consider how "close" the moralized graph is to the original graph $\mathcal{G}$. Intuitively, the addition of the moralizing edges to the Markov network $\mathcal{H}$ leads to the loss of independence information implied by the graph structure.
- For example, if our Bayesian network has head-to-head connections $x \rightarrow z \leftarrow y$, with no edge between x and y, the Markov network $\mathcal{M}[\mathcal{G}]$ loses the information that the nodes x and y are marginally independent (not given Z).
- However, information is not always lost. Intuitively, moralization causes loss of information about independencies only when it introduces new edges into the graph.

Moral Graph

- We then take each conditional distribution factor in the original directed graph and multiply it into one of the clique potentials. There will always exist at least one maximal clique that contains all of the variables in the factor as a result of the moralization step.
- Note that in all cases the partition function is given by $Z = 1$.

Outline

Linear Graph

The joint distribution for this graph takes the form

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

- We shall consider the specific case in which the N nodes represent discrete variables each having K states, in which case each potential function $\psi_{n-1,n}(x_{n-1},\ x_n)$ comprises an $K \times K$ table, and so joint distribution has $(N-1)K^2$ parameters.
- Let us consider the inference problem of finding the marginal distribution $p(x_n)$ for a specific node x_n that is part way along the chain (no observed nodes).

Linear Graph

- The marginal is obtained by summing joint distribution over all variables except x_n, so that

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

- In a naive implementation, we would first evaluate the joint distribution and then perform the summations explicitly. The joint distribution can be represented as a set of numbers, one for each possible value for $\mathbf{x}$.
- Because there are N variables each with K states, there are K^N values for $\mathbf{x}$ and so evaluation and storage of the joint distribution, as well as marginalization to obtain $p(x_n)$, all involve storage and computation that scale exponentially with the length N of the chain.

Exploiting conditional independence

- A much more efficient algorithm by exploiting the conditional independence properties of the graphical model.
- If we substitute the factorized expression for joint distribution, then we can rearrange the order of the summations and the multiplications to allow the required marginal to be evaluated much more efficiently.
- Consider for instance the summation over x_N. The potential $\psi_{N-1,N}(x_{N-1}, x_N)$ is the only one that depends on x_N, and so we can perform the summation

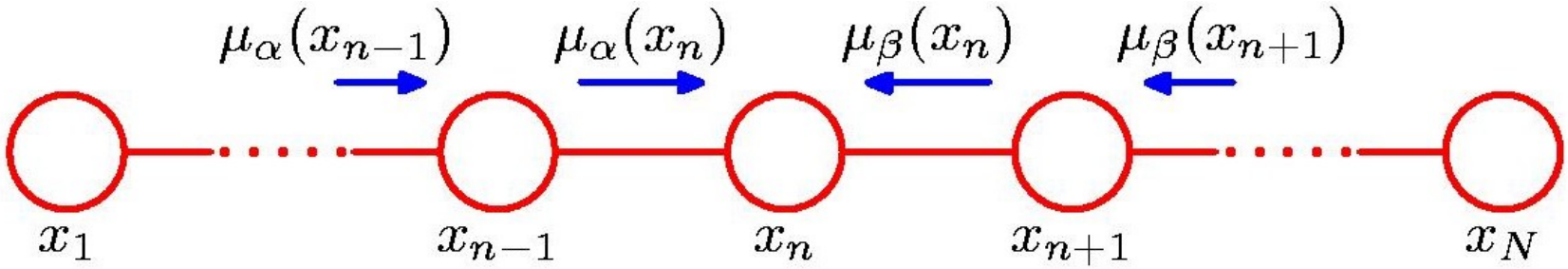
$$\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

first to give a function of x_{N-1}.

Exploiting Conditional Independence

- We can then use this to perform the summation over x_{N-1}, which will involve only this new function together with the potential $\psi_{N-2,N-1}(x_{N-2},\ x_{N-1})$, because this is the only other place that x_{N-1} appears.
- Similarly, the summation over x_1 involves only the potential $\psi_{1,2}(x_1,\ x_2)$ and so can be performed separately to give a function of x_2, and so on.
- Because each summation removes a variable, this can be viewed as the removal of a node from the graph.
- If we group the potentials and summations in this way, we can rearrange the computations much more efficiently !

Linear graph



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

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

Computational cost

- We have to perform $N-1$ summations each of which is over K states and each of which involves a function of two variables. For instance, the summation over x_1 involves only the function $\psi_{1,2}(x_1,\ x_2)$, which is a table of $K \times K$ numbers.
- We have to sum this table over x_1 for each value of x_2 and so this has $O(K^2)$ cost. The resulting vector of K numbers is multiplied by the matrix of numbers $\psi_{2,3}(x_2,\ x_3)$ and so is again $O(K^2)$.
- Because there are $N-1$ summations and multiplications of this kind, the total cost of evaluating the marginal $p(x_n)$ is $O(NK^2)$.

Computational cost

- This is linear in the length of the chain, in contrast to exponential in the naive approach.
- Exploiting the conditional independence properties allows to obtain efficient calculation.
- If the graph had been fully connected, there would have been no conditional independence properties, and we would have been forced to work directly with the full joint distribution.
- Take-home: Many conditional independence properties $\Rightarrow$ Efficient inference algorithm.

Message Passing

- We now give a powerful interpretation of this calculation in terms of the passing local messages around on the graph.
- The expression for the marginal $p(x_n)$ decomposes into the product of two factors times the normalization constant

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

- We shall interpret $\mu_\alpha(x_n)$ as a message passed forwards along the chain from node x_{n-1} to node x_n.
- Similarly, $\mu_\beta(x_n)$ can be viewed as a message passed backwards along the chain to node x_n from node x_{n+1}.
- Note that each of the messages comprises a set of K values, one for each choice of x_n, and so the product of two messages should be interpreted as the point-wise multiplication of the elements of the two messages to give another set of K values.

Forward Messages

- The message $\mu_\alpha(x_n)$ can be evaluated recursively because

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

- We therefore first evaluate $\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1,\ x_2)$ and then apply $\mu_\alpha(x_k) = \sum_{x_{k-1}} \mu_\alpha(x_{k-1})\psi_{k-1,k}(x_{k-1}, x_k)$ repeatedly until we reach the desired node.
- The outgoing message $\mu_\alpha(x_k)$ is obtained by multiplying the incoming message $\mu_\alpha(x_{k-1})$ by the local potential involving the node variable and the outgoing variable and then summing over the node variable.

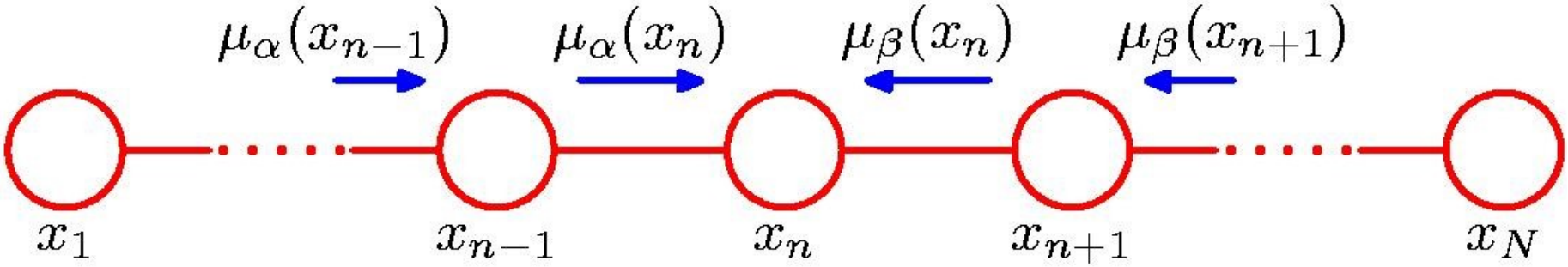
Backward Messages

- Similarly, the message $\mu_\beta(x_n)$ can be evaluated recursively by starting with node x_N and using

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

- We therefore evaluate $\mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1},\ x_N)$ and then compute backward-in-time $\mu_\beta(x_{k-1}) = \sum_{x_k} \psi_{k-1,k}(x_{k-1}, x_k)$ repeatedly until we reach the desired node.

Forward-Backward messages



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

$$= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}).$$

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

$$= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_\beta(x_{n+1}).$$

Computation of all Marginal distributions

- Now suppose we wish to evaluate the marginals $p(x_n)$ for every node $n \in \{1, \ldots N\}$ in the chain.
- Simply applying the above procedure separately for each node will have computational cost that is $O(N^2M^2)$. However, such an approach would be very wasteful of computation !
- For instance, to find $p(x_1)$ we need to propagate a message $\mu_\beta(\cdot)$ from node x_N back to node x_2.
- Similarly, to evaluate $p(x_2)$ we need to propagate a messages $\mu_\beta(\cdot)$ from node x_N back to node x_3. This will involve much duplicated computation because most of the messages will be identical in the two cases.

Computation of all marginal distributions

- Suppose instead we first launch a message $\mu_\beta(x_{N-1})$ starting from node x_N and propagate corresponding messages all the way back to node x_1, and suppose we similarly launch a message $\mu_\alpha(x_2)$ starting from node x_1 and propagate the corresponding messages all the way forward to node x_N.
- Provided we store all of the intermediate messages along the way, then any node can evaluate its marginal simply by applying

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

- The computational cost is only twice that for finding the marginal of a single node, rather than N times as much. Observe that a message has passed once in each direction across each link in the graph. Note also that the normalization constant Z need be evaluated only once, using any convenient node.

Including Observations

- If some of the nodes in the graph are observed, then the corresponding variables are simply clamped to their observed values and there is no summation.
- To see this, note that the effect of clamping a variable x_n to an observed value $\hat{x}_n$ can be expressed by multiplying the joint distribution by (one or more copies of) an additional function $\delta(x_n, \hat{x}_n)$, which takes the value 1 when $x_n = \hat{x}_n$ and the value 0 otherwise.
- One such function can then be absorbed into each of the potentials that contain x_n. Summations over x_n then contain only one term in which $x_n = \hat{x}_n$.

From marginals to pairs

- Now suppose we wish to calculate the joint distribution $p(x_{n-1}, x_n)$ for two neighbouring nodes on the chain. This is similar to the evaluation of the marginal for a single node, except that there are now two variables that are not summed out.
- The joint distribution can be written in the form

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

- Thus we can obtain the joint distributions over all of the sets of variables in each of the potentials directly once we have completed the message passing required to obtain the marginals.

From marginals to pairs

- This is a useful result because in practice we may wish to use parametric forms for the clique potentials, or equivalently for the conditional distributions if we started from a directed graph.
- In order to learn the parameters of these potentials in situations where not all of the variables are observed, we can employ the *EM algorithm* and it turns out that the local joint distributions of the cliques, conditioned on any observed data, is precisely what is needed in the E step.
- We shall consider an illustration of this for HMM later on !.

Trees

- We have seen that exact inference on a graph comprising a chain of nodes can be performed efficiently in time that is linear in the number of nodes using that can be interpreted in terms messages passed along the chain.
- More generally, inference can be performed efficiently using local message passing on a broader class of graphs called trees.
- In particular, we shall shortly generalize the message passing formalism derived above for chains to give the sum-product algorithm, which provides an efficient framework for exact inference in tree-structured graphs.

Trees

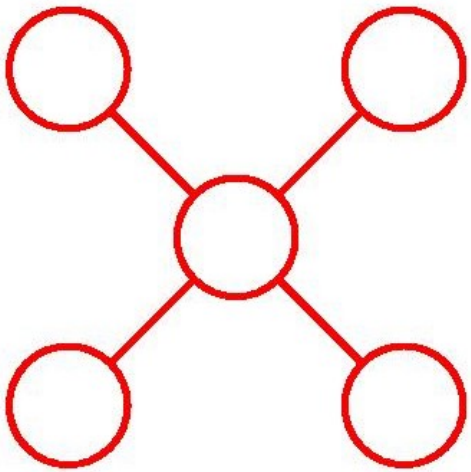
- In the case of an undirected graph, a tree is defined as a graph in which there is one, and only one, path between any pair of nodes. Such graphs therefore do not have loops.
- In the case of a directed graph, a tree is defined such that there is a single node, called the root, which has no parent, and all other nodes have one parent.
- If we convert a directed tree into an undirected graph, we see that the moralization step will not add any links as all nodes have at most one parent: the moralized graph will be an undirected tree.

Poly-Trees

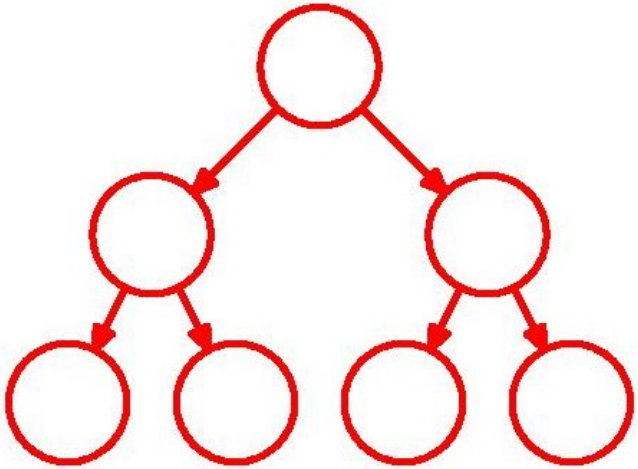
- If there are nodes in a directed graph that have more than one parent, but there is still only one path (ignoring the direction of the arrows) between any two nodes, then the graph is a called a polytree.
- Such a graph will have more than one node with the property of having no parents, and furthermore, the corresponding moralized undirected graph will have loops.

Tree

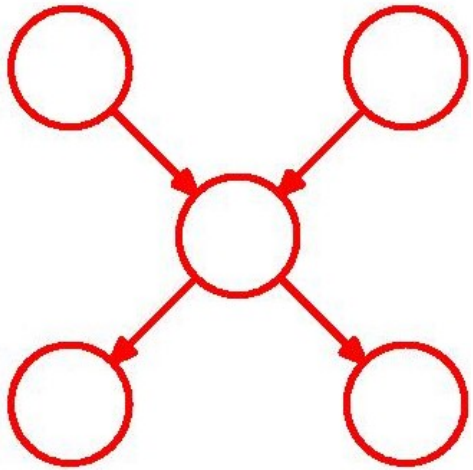
Undirected Tree



Directed Tree



Polytree



Factor Graph

- The sum-product algorithm can be cast in a simple form if we first introduce 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.

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

  where $\mathbf{x}_s$ denotes a subset of the variables.
- For directed graphs, the factors $f_s(\mathbf{x}_s)$ are local conditional distributions. For undirected graphs, the factors are potential functions over the maximal cliques.

Factor Graph

- In a factor graph, there is a node (circle) for every variable in the distribution, as was the case for directed and undirected graphs.
- There are also additional nodes (small squares) for each factor $f_s(\mathbf{x}_s)$ in the joint distribution.
- Finally, there are undirected links connecting each factor node to all of the variables nodes on which that factor depends.

Factor Graphs

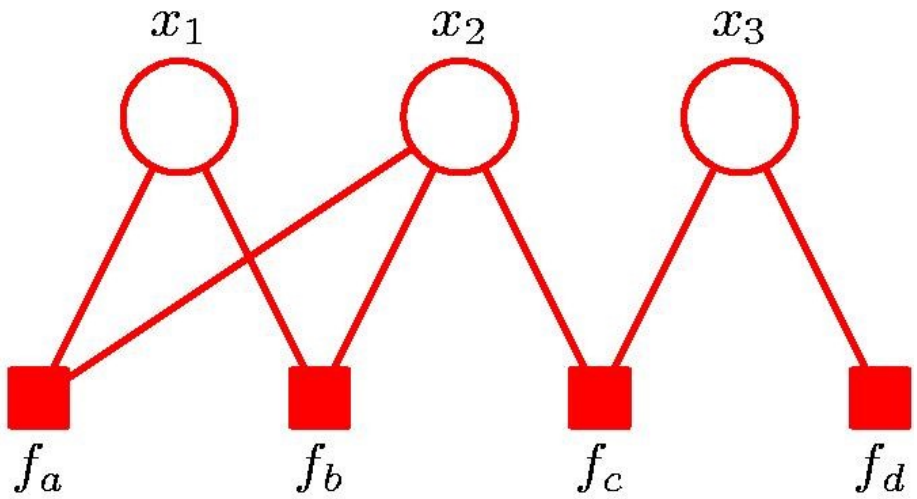
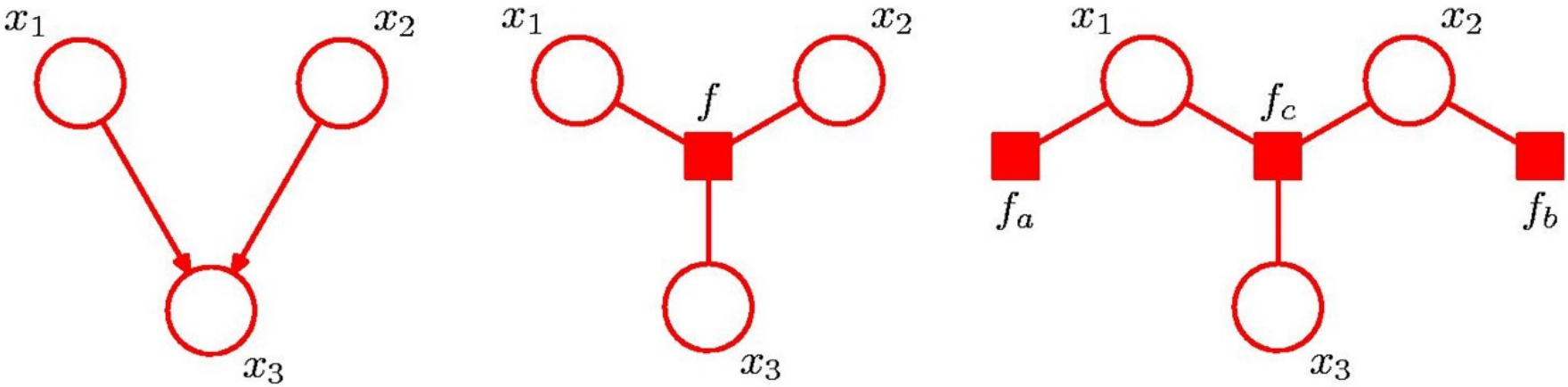


Figure: Factor graph associated to the factorization $p(\mathbf{x}) = f_a(x_1,\ x_2)f_b(x_1,\ x_2)f_c(x_2,\ x_3)f_d(x_3)$ Note that there are two factors $f_a(x_1,\ x_2)$ and $f_b(x_1,\ x_2)$ that are defined over the same set of variables. In an undirected graph, the product of two such factors would simply be lumped together into the same clique potential. The factor graph, however, keeps such factors explicit to convey information about the underlying factorization.

Bipartite

- Factor graphs are said to be bipartite because they consist of two distinct kinds nodes, and all links go between nodes of opposite type.
- In general, factor graphs can therefore always be drawn as two rows of nodes (variable nodes at the top and factor nodes at the bottom) with links between the rows.
- In some situations, however, other ways of laying out the graph may be more intuitive, for example when the factor graph is derived from a directed or undirected graph.

Factor graphs from directed tree



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

$$f(x_1, x_2, x_3) = p(x_1)p(x_2)p(_3|x_1, x_2)$$

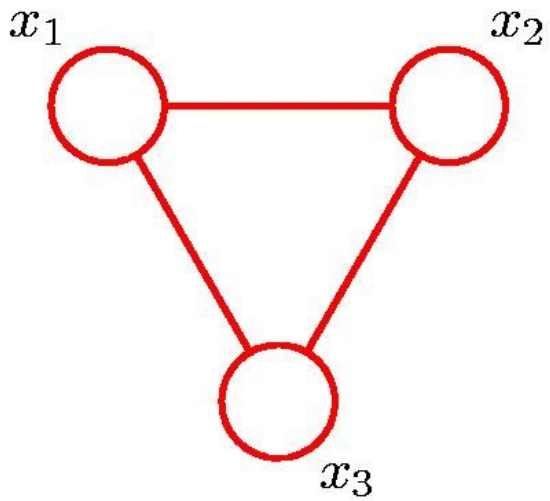
$$f_a(x_1) = p(x_1)$$

$$f_b(x_2) = p(x_2)$$

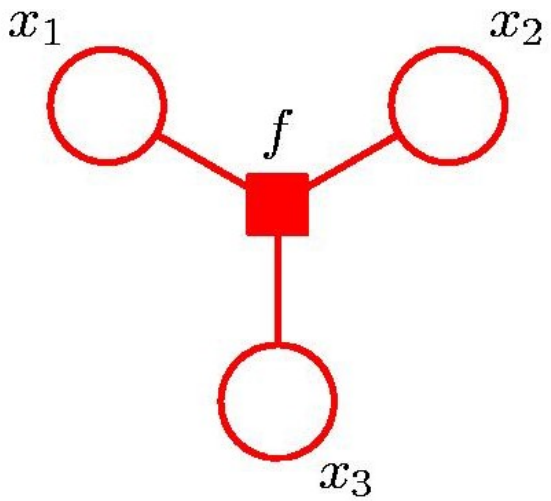
$$f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2)$$

Figure: Create variable nodes in the factor graph corresponding to the nodes of the directed graph, and then create factor nodes corresponding to the conditional distributions, and add the appropriate links. Again, there can be multiple factor graphs all of which correspond to the same directed graph.

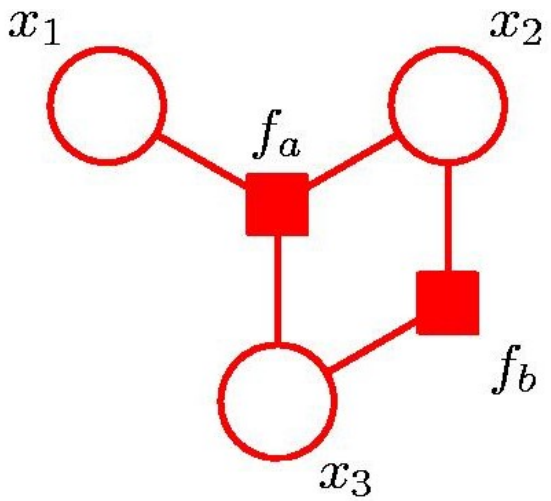
Factor graphs from undirected graphs



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



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



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

Figure: Create variable nodes corresponding to the nodes in the original undirected graph, and then create additional factor nodes corresponding to the maximal cliques. The factors are then set equal to the clique potentials. There may be several different factor graphs that correspond to the same undirected graph.

Factors graphs from Polytree

- If we take a directed or undirected tree and convert it into a factor graph, then the result will again be a tree (in other words, the factor graph will have no loops, and there will be one and only one path connecting any two nodes).
- For a directed polytree, conversion to an undirected graph results in loops (moralization)...Conversion to a factor graph again results in a tree.
- 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 factor function.

Sum-product Algorithm

- We shall focus on the problem of evaluating local marginals over nodes or subsets of nodes, which will lead us to the sum-product algorithm.
- These techniques can be modified to allow the most probable state to be found, giving rise to the max-sum algorithm.
- Also we shall suppose that all of the variables in the model are discrete, and so marginalization corresponds to performing sums. The framework, however, is equally applicable to linear-Gaussian models in which case marginalization involves (explicit) integration.

Marginal computation

- We begin by considering the problem of finding the marginal $p(x)$ for particular variable node x. For the moment, we shall suppose that all of the variables are hidden.
- By definition, the marginal is obtained by summing joint distribution over all variables except x so that

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

- The idea is to substitute for $p(\mathbf{x})$ using the factor graph expression

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

and then interchange summations and products in order to obtain an efficient algorithm.

Sum-Product

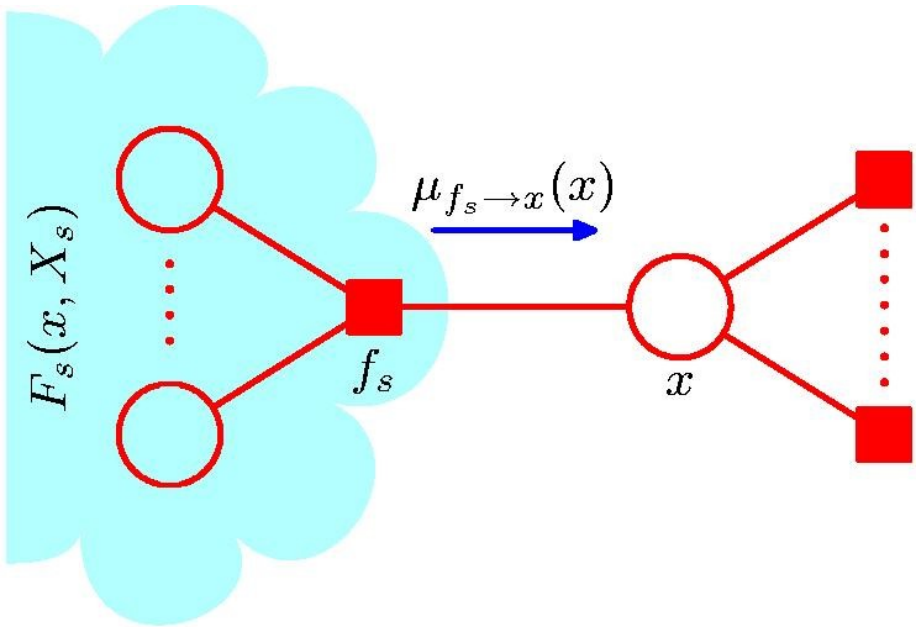


Figure: The tree structure of the graph allows us to partition the factors in the joint distribution into groups, with one group associated with each of the factor nodes that is a neighbour of the variable node x.

Sum-Product

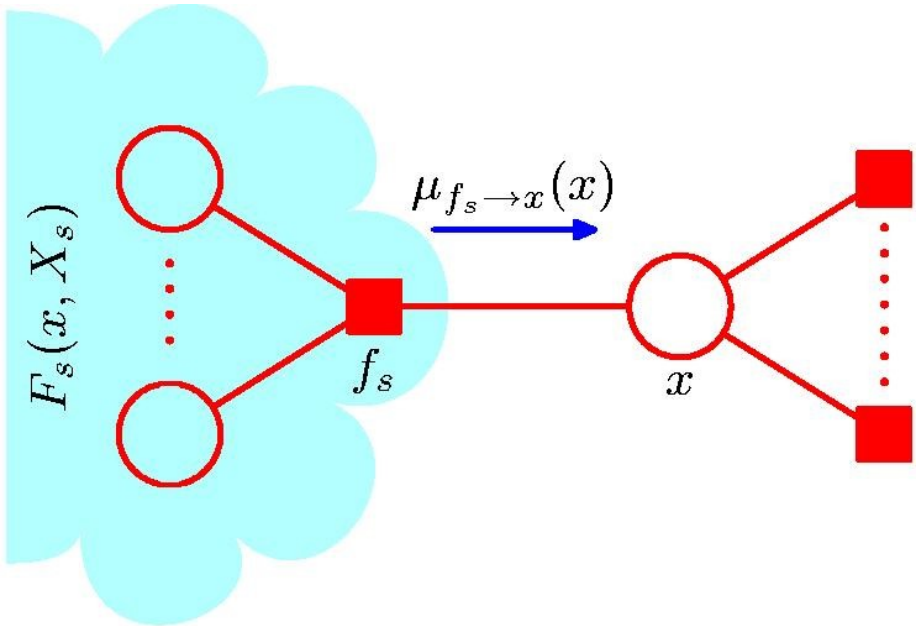


Figure: The joint distribution can be written as a product of the form $p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x,\ X_s)$ where $\mathrm{ne}(x)$ denotes the set of factor nodes that are neighbours of x, and X_s denotes the set of all variables in the subtree connected to the variable node x via the factor node f_s, and $F_s(x,\ X_s)$ represents the product of all the factors in the group associated with factor f_s.

Sum-Product

Interchanging the sums and products, the marginal distribution $p(x)$ is obtained by summing over $\{X_s\}_{s\in \mathrm{ne}(x)}$,

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

where the functions $\mu_{f_s\to x}(x)$ can be viewed as messages passing from the factor nodes f_s to the variable node x.

The required marginal $p(x)$ is given by the product of all the incoming messages arriving at node x.

Sum-product

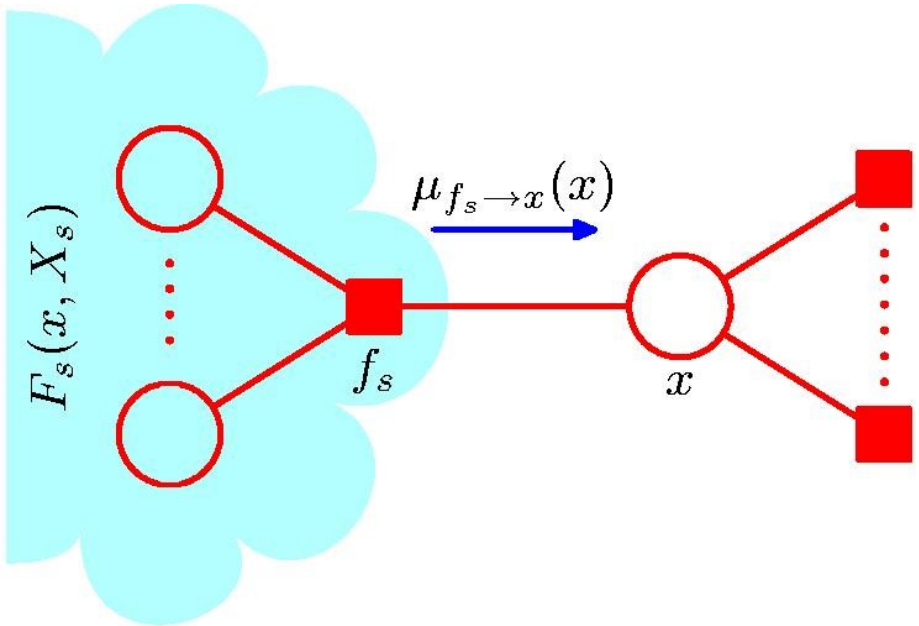


Figure: $p(x) = \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$ with $\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x,\ X_s)$

Sum-Product

Each factor $F_s(x,\ X_s)$ is described by a factor (sub-)graph and so can itself be factorized.

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

where $x_1, \ldots, x_M$ are the variables associated with factor f_s in addition to x and $\{X_{si}\}_{i=1}^{M}$ are all the variables in the subtree connected to $\{x_i\}_{i=1}^{M}$.

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

where $\mathrm{ne}(f_s)$ denotes the set variable nodes that are neighbours of the factor node f_s.

Sum-product

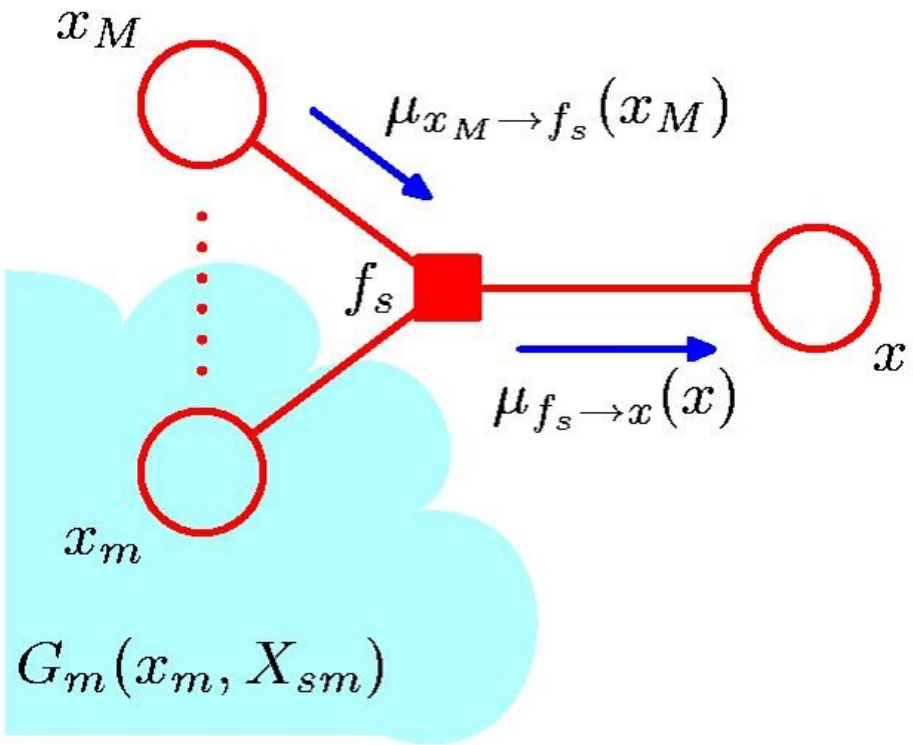


Figure: $\mu_{f_s \to x}(x) = \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$ with $\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm})$

Two kinds of messages

- We have introduced two distinct kinds of message
  (i) messages that go from factor nodes to variable nodes denoted $\mu_{f \to x}(x)$
  (ii) messages that go from variable nodes to factor nodes denoted $\mu_{x \to f}(x)$.
- In each case, the messages passed along a link are always a function of the variable associated with the variable node that link connects to.

From factor node to variable node

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

- Compute the product of the incoming messages along all links $m \in \mathrm{ne}(f_s) \setminus x$ coming into the factor node.
- Multiply by the factor $f_s(x, x_1, \ldots, x_M)$ associated with that node,
- Marginalize over all of the variables associated with the incoming messages.

A factor node can send a message to a variable node once it has received incoming messages from all other neighbouring variable nodes.

From variable node to factor node

The term $G_m(x_m,\ X_{sm})$ associated with node x_m is given by a product of terms $F_l(x_m,\ X_{ml})$ each associated with one of the factor nodes f_l that is linked to node x_m excluding node f_s (the graph is bipartite !)

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

where the product is taken over all (factor nodes) neighbours of node x_m except for node f_s.

Each of the factors $F_l(x_m,\ X_{ml})$ again represents a subtree of the original graph.

From variable node to factor node

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \left[\sum_{X_{ml}} F_l(x_m,\ X_{ml}) \right]$$
$$= \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

To evaluate the message sent by a variable node to an adjacent factor node along the connecting link, we simply take the product of the incoming messages along all of the other links.

- Any variable node that has only two neighbours performs no computation but simply passes messages through unchanged.
- A variable node can send a message to a factor node once it has received incoming messages from all other neighbouring factor nodes.

Sum-Product

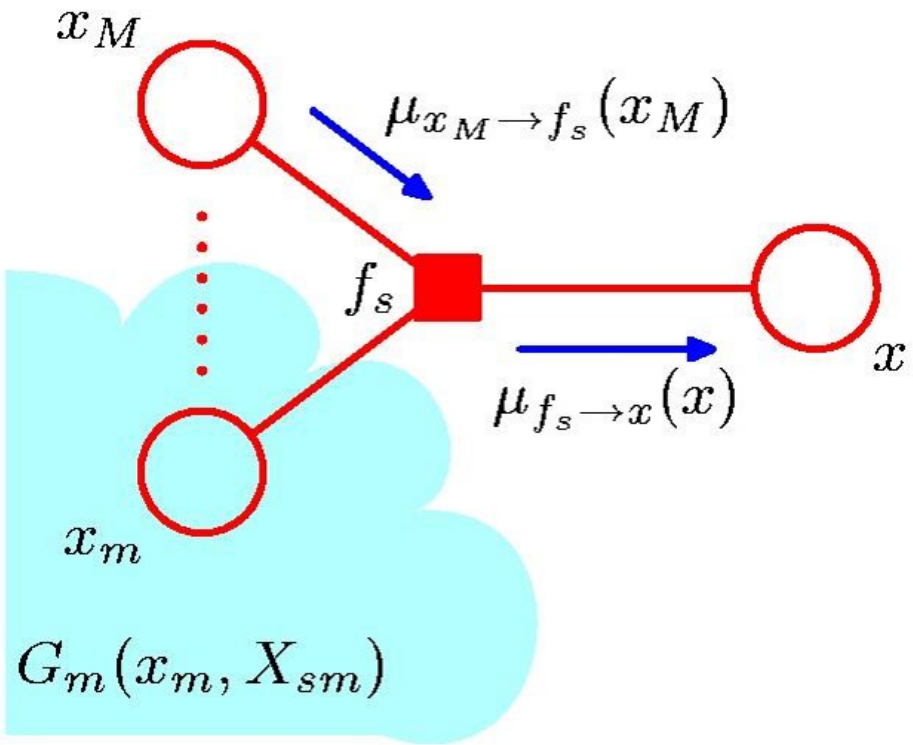


Figure: $\mu_{x_m \to f_s}(x_m) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$

Algorithm

- We view the variable node x as the root of the factor graph.
- Initialize messages at the leaves of the graph. The message that it sends along its one and only link is given by

  (i) if a leaf node is a variable node,

$$\mu_{y \to g}(y) = 1$$

  (ii) if the leaf node is a factor node,

$$\mu_{g \to y}(y) = f(y)$$

- The message passing steps are then applied recursively until messages have been propagated along every link, and the root node has received messages from all of its neighbours.
- Each node can send a message towards the root once it has received messages from all of its other neighbours. Once the root node has received messages from all of its neighbours, the required marginal can be evaluated by computing the products over all messages !

Sum-Product Algorithm

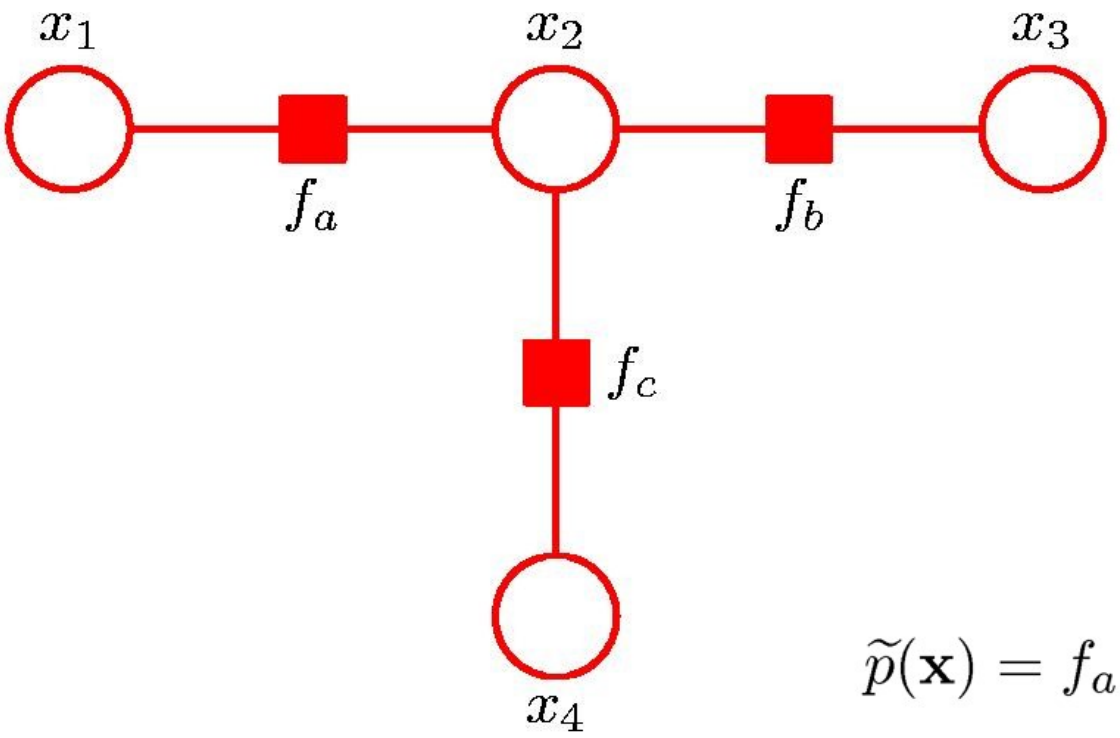
- Arbitrarily pick any (variable or factor) node and designate it as the root.
- Propagate messages from the leaves to the root as before. At this point, the root node will have received messages from all of its neighbours.
- Send out messages from the root to all of its neighbours. These in turn will then have received messages from all their neighbours and so can send out messages along the links going away from the root, and so on. In this way, messages are passed outwards from the root all the way to the leaves.
- A message will have passed in both directions across every link in the graph, and every node will have received a message from all of its neighbours.

Sum-Product Algorithm

To compute local marginals:

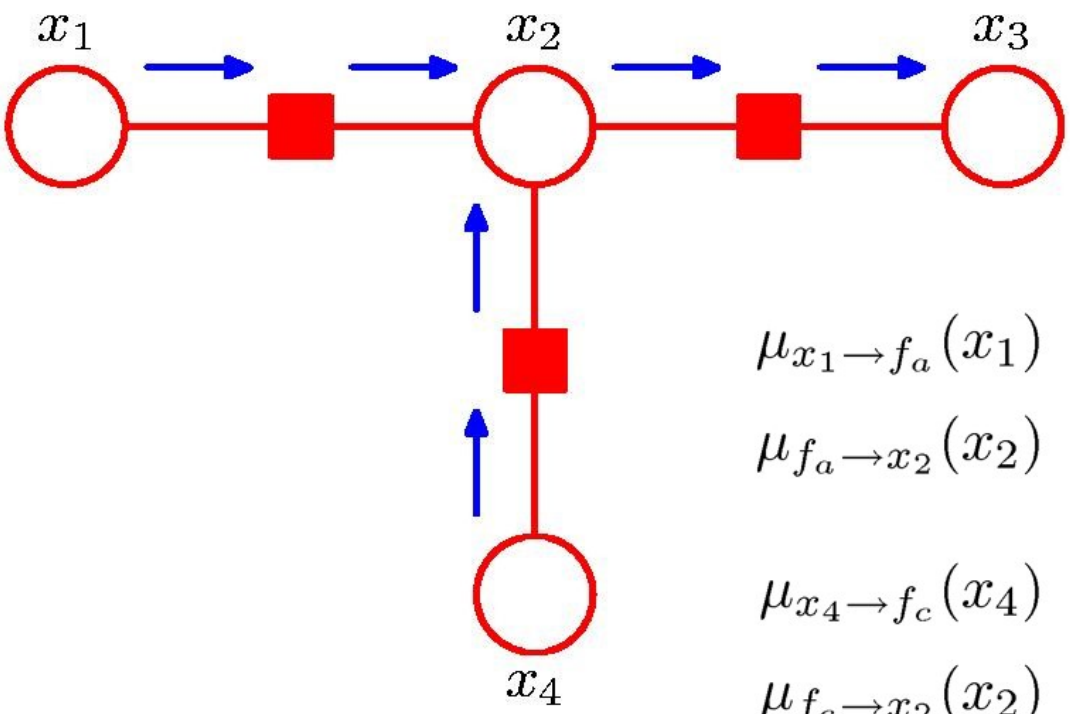
- Pick an arbitrary node as root
- Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
- Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
- Compute the product of received messages at each node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

Sum-Product: Example



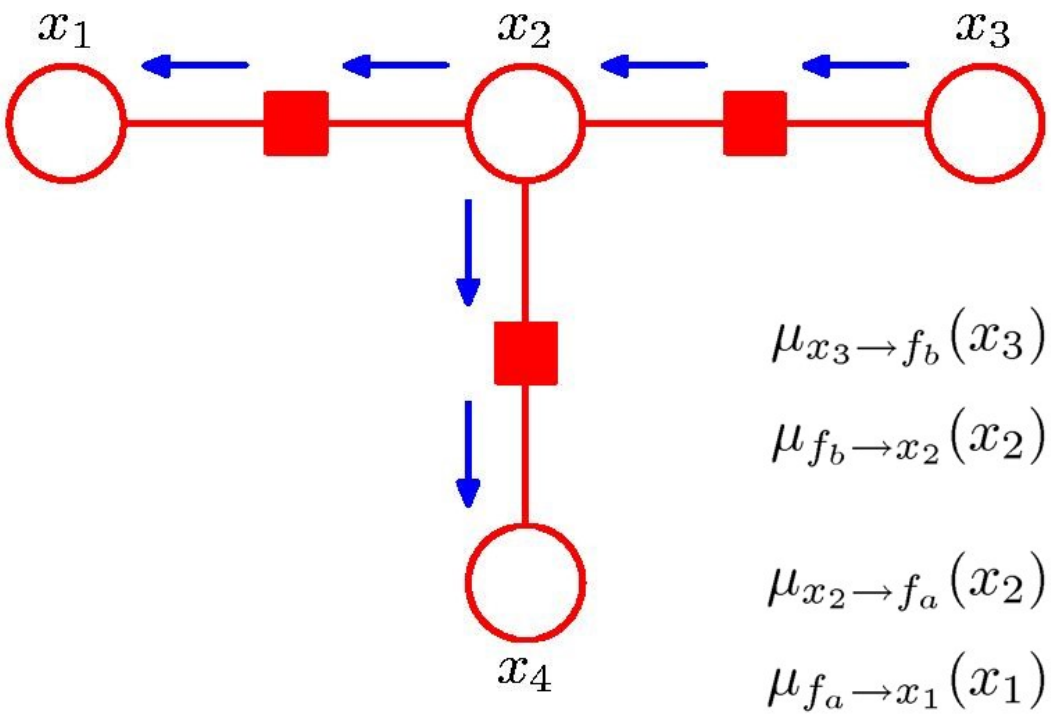
$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

Sum-Product: Example



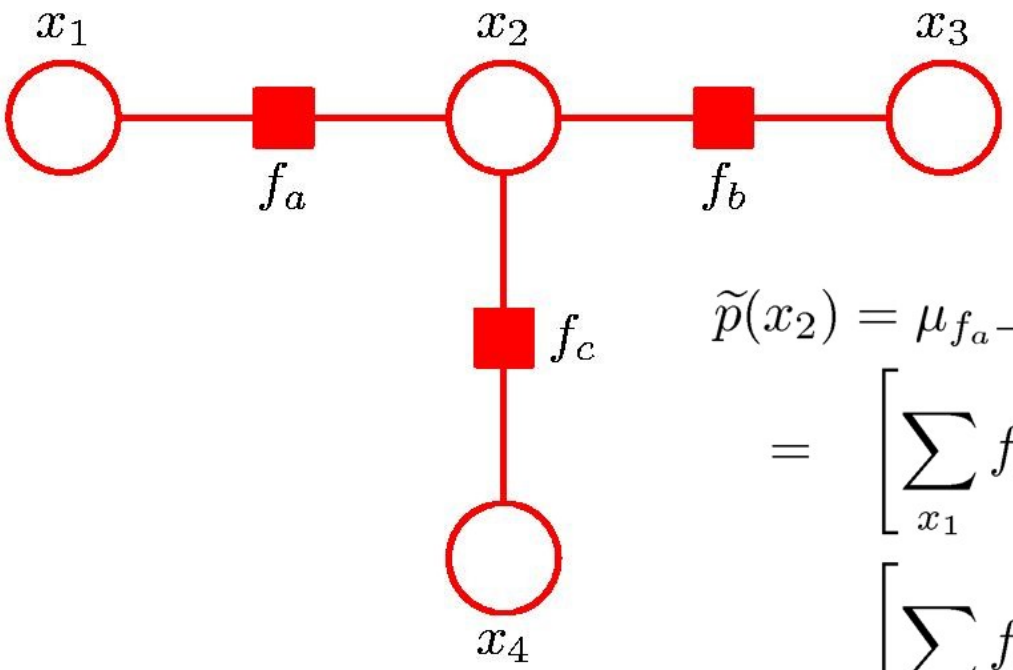
$$
\begin{aligned}
\mu_{x_1 \to f_a}(x_1) &= 1 \\
\mu_{f_a \to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\
\mu_{x_4 \to f_c}(x_4) &= 1 \\
\mu_{f_c \to x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\
\mu_{x_2 \to f_b}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_b \to x_3}(x_3) &= \sum_{x_2} f_b(x_2,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

Sum-Product: Example



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

Sum-Product: Example



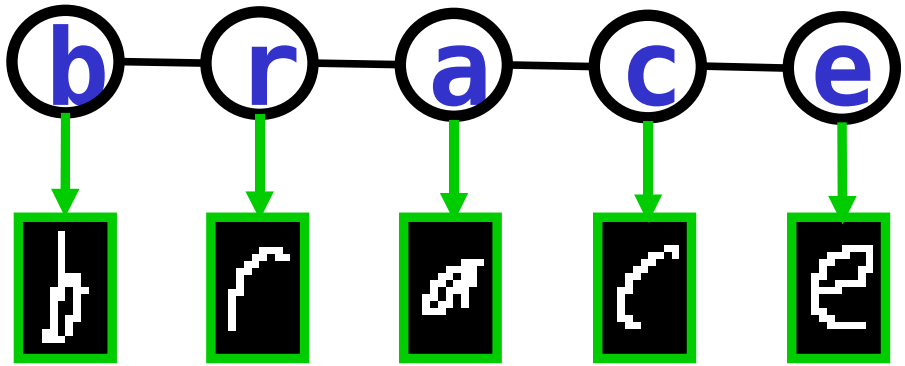
$$\begin{aligned}
\widetilde{p}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
&= \left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right] \\
&\quad \left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \sum_{x_1}\sum_{x_3}\sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3)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

Outline

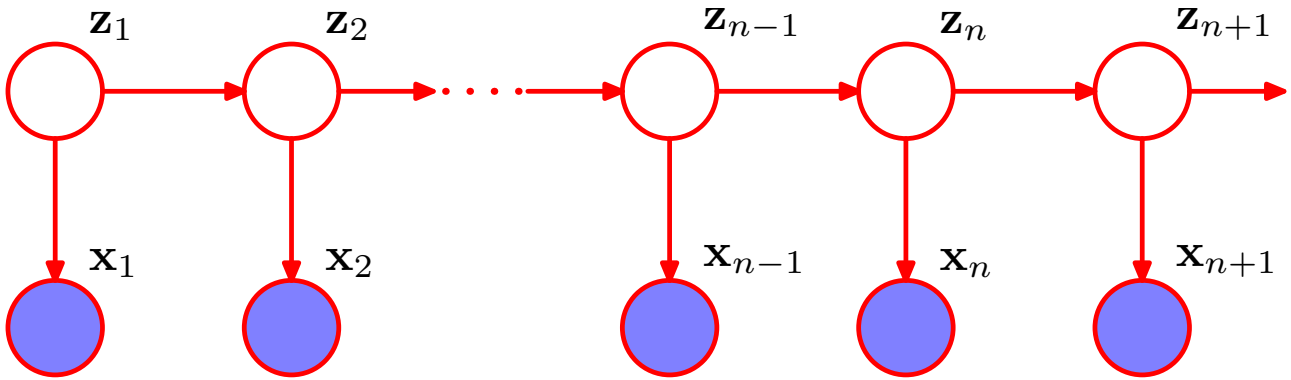
Hidden Markov models

Hidden Markov models (HMM)

- speech recognition
- natural language processing
- OCR
- biological sequences (proteins, DNA)



Hidden Markov Model(HMM)



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) \prod_{n=1} p(\mathbf{x}_n|\mathbf{z}_n)$$

Homogeneous Markov chain

- $\mathbf{z}_n \in \{0,1\}^K$ indicator variable for the state $(1, \ldots, K)$
- Homogeneous Markov chain: $\forall n,\ p(\mathbf{z}_n|\mathbf{z}_{n-1}) = p(\mathbf{z}_2|\mathbf{z}_1)$
- $\mathbf{x}_n$ emitted symbol $(\{0,1\}^K)$ / observation $(\mathbb{R}^d)$

Hidden Markov Model (HMM)

Parametrization

distribution of initial state $p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

Hidden Markov Model (HMM)

Parametrization

distribution of initial state $p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

transition matrix $p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}; A) = \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{z_{n-1,j}\, z_{nk}}$

Hidden Markov Model (HMM)

Parametrization

distribution of initial state $p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

transition matrix $p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}; A) = \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{z_{n-1,j}\, z_{nk}}$

emission probabilities $p(\mathbf{x}_n | \boldsymbol{z}_n; \phi)$ e.g. Gaussian Mixture

Hidden Markov Model (HMM)

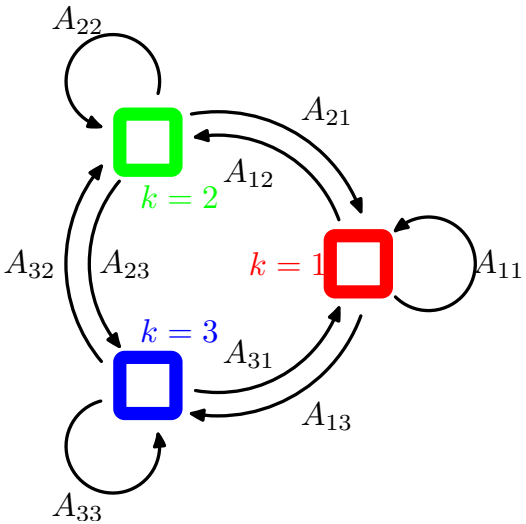
Parametrization

distribution of initial state $p(\boldsymbol{z}_1; \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$

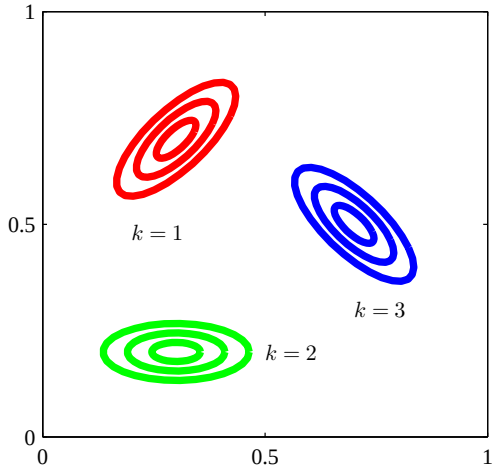
transition matrix $p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}; A) = \prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{z_{n-1,j}\, z_{nk}}$

emission probabilities $p(\mathbf{x}_n | \boldsymbol{z}_n; \phi)$ e.g. Gaussian Mixture

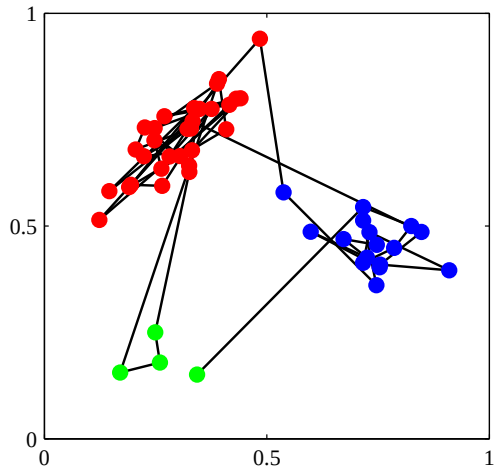
Interpretation



Transistions of $\boldsymbol{z}_n$



$p(\mathbf{x}_n | \boldsymbol{z}_n)$



Trajectory of $\mathbf{x}_n$

Maximum likelihood for HMMs

Applying the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Maximum likelihood for HMMs

Applying the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Espectation of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n|\phi_k)$$

Maximum likelihood for HMMs

Applying the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Espectation of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n|\phi_k)$$

When maximizing w.r.t. $\{\pi, A\}$ one obtains

$$\boxed{\pi_k^{t+1} = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}} \qquad \boxed{A_{jk}^{t+1} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}}$$

Maximum likelihood for HMMs

Applying the EM algorithm

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) \qquad \xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = p(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t)$$

Espectation of the log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \log p(x_n|\phi_k)$$

When maximizing w.r.t. $\{\pi, A\}$ one obtains

$$\boxed{\pi_k^{t+1} = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}} \qquad \boxed{A_{jk}^{t+1} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}}$$

If the emissions are Gaussians we have as well:

$$\boldsymbol{\mu}_k^{t+1} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \qquad \boldsymbol{\Sigma}_k^{t+1} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

Maximum likelihood for HMMs

Application of the sum-product algorithm

In the context of HMM, the algorithm is known as *forward-backward*.

The following messages are propagated

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$

Maximum likelihood for HMMs

Application of the sum-product algorithm

In the context of HMM, the algorithm is known as *forward-backward*.

The following messages are propagated

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$
- backward $\beta(\boldsymbol{z}_n) = \sum_{\boldsymbol{z}_{n+1}} \beta(\boldsymbol{z}_{n+1}) p(\mathbf{x}_{n+1}|\boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n)$

Maximum likelihood for HMMs

Application of the sum-product algorithm

In the context of HMM, the algorithm is known as *forward-backward*.

The following messages are propagated

- forward $\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$
- backward $\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n)$

they satisfy the properties:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) \qquad \beta(\mathbf{z}_n) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\mathbf{z}_n)$$

Maximum likelihood for HMMs

Application of the sum-product algorithm

In the context of HMM, the algorithm is known as *forward-backward*.

The following messages are propagated

- forward $\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_n|\boldsymbol{z}_n) \sum_{\boldsymbol{z}_{n-1}} \alpha(\boldsymbol{z}_{n-1}) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$
- backward $\beta(\boldsymbol{z}_n) = \sum_{\boldsymbol{z}_{n+1}} \beta(\boldsymbol{z}_{n+1}) p(\mathbf{x}_{n+1}|\boldsymbol{z}_{n+1}) p(\boldsymbol{z}_{n+1}|\boldsymbol{z}_n)$

they satisfy the properties:

$$\alpha(\boldsymbol{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{z}_n) \qquad \beta(\boldsymbol{z}_n) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N|\boldsymbol{z}_n)$$

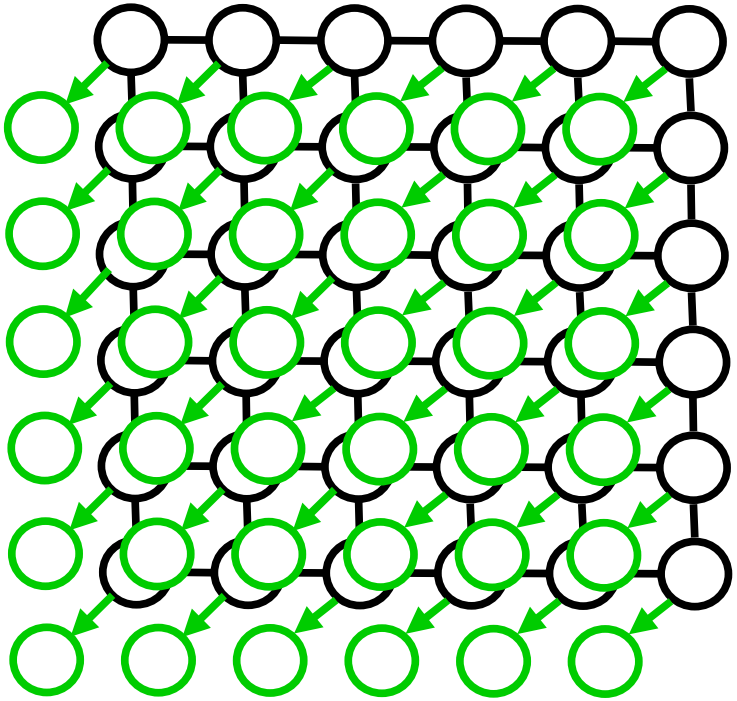
Finally we obtain the marginal probabilities:

$$\gamma(\boldsymbol{z}_n) = p(\boldsymbol{z}_n|\boldsymbol{X}, \boldsymbol{\theta}^t) = \frac{\alpha(\boldsymbol{z}_n)\beta(\boldsymbol{z}_n)}{p(\boldsymbol{X}|\boldsymbol{\theta}^t)}$$

et

$$\xi(\boldsymbol{z}_{n-1}, \boldsymbol{z}_n) = \frac{\alpha(\mathbf{x}_{n-1}) p(\mathbf{x}_n|\boldsymbol{z}_n) p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \beta(\mathbf{x}_n)}{p(\boldsymbol{X}|\boldsymbol{\theta}^t)}$$

Hidden Markov Field



Original image

Segmentation

Outline

The EM algorithm for the Gaussian mixture model

Gaussian mixture model

- K components
- $\boldsymbol{z}$ component indicator
- $\boldsymbol{z} = (z_1, \ldots, z_K)^\top \in \{0, 1\}^K$
- $\boldsymbol{z} \sim \mathcal{M}(1, (\pi_1, \ldots, \pi_K))$
- $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$

Gaussian mixture model

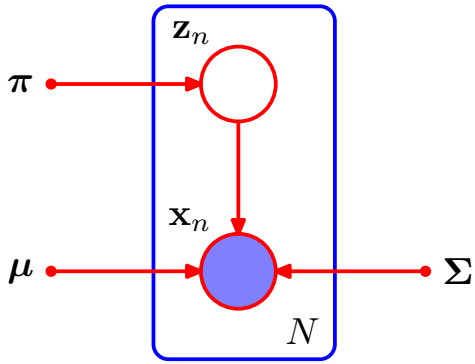
- K components
- $\boldsymbol{z}$ component indicator
- $\boldsymbol{z} = (z_1, \ldots, z_K)^\top \in \{0, 1\}^K$
- $\boldsymbol{z} \sim \mathcal{M}(1, (\pi_1, \ldots, \pi_K))$
- $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- $p(\mathbf{x}|\boldsymbol{z}; (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_k) = \sum_{k=1}^{K} z_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Gaussian mixture model

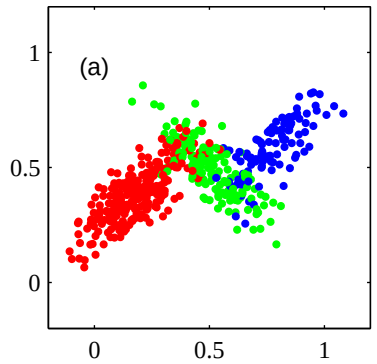
- K components
- $\boldsymbol{z}$ component indicator
- $\boldsymbol{z} = (z_1, \ldots, z_K)^\top \in \{0, 1\}^K$
- $\boldsymbol{z} \sim \mathcal{M}(1, (\pi_1, \ldots, \pi_K))$
- $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$
- $p(\mathbf{x}|\boldsymbol{z}; (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_k) = \sum_{k=1}^{K} z_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- $p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Gaussian mixture model

- K components
- $\boldsymbol{z}$ component indicator
- $\boldsymbol{z} = (z_1, \ldots, z_K)^\top \in \{0,1\}^K$
- $\boldsymbol{z} \sim \mathcal{M}(1, (\pi_1, \ldots, \pi_K))$
- $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$



- $p(\mathbf{x}|\boldsymbol{z}; (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_k) = \sum_{k=1}^{K} z_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- $p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$



- Estimation: $\underset{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}{\text{argmax}} \log \left[\sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$p(\mathbf{x}) =$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z})$$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} =$$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Issue

- The marginal log-likelihood $\tilde{\ell}(\theta) = \sum_i \log(p(\mathbf{x}^{(i)}))$ with $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$ is now complicated

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Issue

- The marginal log-likelihood $\tilde{\ell}(\theta) = \sum_i \log(p(\mathbf{x}^{(i)}))$ with $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$ is now complicated
- No hope to find a simple solution to the maximum likelihood problem

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Issue

- The marginal log-likelihood $\tilde{\ell}(\theta) = \sum_i \log(p(\mathbf{x}^{(i)}))$ with $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$ is now complicated
- No hope to find a simple solution to the maximum likelihood problem
- By contrast the complete log-likelihood has a rather simple form:

$$\tilde{\ell}(\theta) =$$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Issue

- The marginal log-likelihood $\tilde{\ell}(\theta) = \sum_i \log(p(\mathbf{x}^{(i)}))$ with $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$ is now complicated
- No hope to find a simple solution to the maximum likelihood problem
- By contrast the complete log-likelihood has a rather simple form:

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)})$$

Applying maximum likelihood to the Gaussian mixture

Let $\mathcal{Z} = \{z \in \{0,1\}^K \mid \sum_{k=1}^K z_k = 1\}$

$$p(\mathbf{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\mathbf{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} \prod_{k=1}^{K} \left[\pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]^{z_k} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Issue

- The marginal log-likelihood $\tilde{\ell}(\theta) = \sum_i \log(p(\mathbf{x}^{(i)}))$ with $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$ is now complicated
- No hope to find a simple solution to the maximum likelihood problem
- By contrast the complete log-likelihood has a rather simple form:

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,\ k} z_k^{(i)} \log \mathcal{N}(x^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) =$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)})$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

$$p(z_k^{(i)} = 1 \mid \mathbf{x}; \theta) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

$$p(z_k^{(i)} = 1 \mid \mathbf{x}; \theta) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$\rightarrow$ Seems a chicken and egg problem...

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

$$p(z_k^{(i)} = 1 \mid \mathbf{x}; \theta) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$\rightarrow$ Seems a chicken and egg problem...

- In addition, we want to solve

$$\max_{\theta} \sum_{i} \log \left(\sum_{\boldsymbol{z}^{(i)}} p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) \right)$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

$$p(z_k^{(i)} = 1 \mid \mathbf{x}; \theta) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$\rightarrow$ Seems a chicken and egg problem...

- In addition, we want to solve

$$\max_{\theta} \sum_i \log \left(\sum_{\boldsymbol{z}^{(i)}} p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) \right) \quad \text{and not} \quad \max_{\substack{\theta, \\ \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(M)}}} \sum_i \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)})$$

Applying maximum likelihood to the multinomial mixture

$$\tilde{\ell}(\theta) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) = \sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k),$$

- If we knew $\boldsymbol{z}^{(i)}$ we could maximize $\tilde{\ell}(\theta)$.
- If we knew $\theta = \left(\boldsymbol{\pi}, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{1 \leq k \leq K}\right)$, we could find the best $\boldsymbol{z}^{(i)}$ since we could compute the true a posteriori on $\boldsymbol{z}^{(i)}$ given $\mathbf{x}^{(i)}$:

$$p(z_k^{(i)} = 1 \mid \mathbf{x}; \theta) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$\rightarrow$ Seems a chicken and egg problem...

- In addition, we want to solve

$$\max_{\theta} \sum_i \log \left(\sum_{\boldsymbol{z}^{(i)}} p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}) \right) \quad \text{and not} \quad \max_{\substack{\theta, \\ \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(M)}}} \sum_i \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)})$$

- Can we still use the intuitions above to construct an algorithm maximizing the marginal likelihood?

Principle of the Expectation-Maximization Algorithm

$$\log p(\mathbf{x};\boldsymbol{\theta}) \quad =$$

Principle of the Expectation-Maximization Algorithm

$$\log p(\mathbf{x};\boldsymbol{\theta}) \quad = \quad \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})$$

Principle of the Expectation-Maximization Algorithm

$$\log p(\mathbf{x};\boldsymbol{\theta}) \quad = \quad \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}$$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\end{aligned}$$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}
\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q)
\end{aligned}$$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}
\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})
\end{aligned}$$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}
\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})
\end{aligned}$$

- This shows that $\mathcal{L}(q,\boldsymbol{\theta}) \leq \log p(\mathbf{x};\boldsymbol{\theta})$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}
\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\
&= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})
\end{aligned}$$

- This shows that $\mathcal{L}(q,\boldsymbol{\theta}) \leq \log p(\mathbf{x};\boldsymbol{\theta})$
- Moreover: $\boldsymbol{\theta} \mapsto \mathcal{L}(q,\boldsymbol{\theta})$ is a **concave** function.

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})\end{aligned}$$

- This shows that $\mathcal{L}(q,\boldsymbol{\theta}) \leq \log p(\mathbf{x};\boldsymbol{\theta})$
- Moreover: $\boldsymbol{\theta} \mapsto \mathcal{L}(q,\boldsymbol{\theta})$ is a **concave** function.
- Finally it is possible to show that

$$\mathcal{L}(q,\boldsymbol{\theta}) = \log p(\mathbf{x};\boldsymbol{\theta}) - KL(q||p(\cdot|\mathbf{x};\boldsymbol{\theta}))$$

Principle of the Expectation-Maximization Algorithm

$$\begin{aligned}\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})\end{aligned}$$

- This shows that $\mathcal{L}(q,\boldsymbol{\theta}) \leq \log p(\mathbf{x};\boldsymbol{\theta})$
- Moreover: $\boldsymbol{\theta} \mapsto \mathcal{L}(q,\boldsymbol{\theta})$ is a **concave** function.
- Finally it is possible to show that

$$\mathcal{L}(q,\boldsymbol{\theta}) = \log p(\mathbf{x};\boldsymbol{\theta}) - KL(q||p(\cdot|\mathbf{x};\boldsymbol{\theta}))$$

So that if we set $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x};\boldsymbol{\theta}^{(t)})$ then

$$L(q,\boldsymbol{\theta}^{(t)}) = p(\mathbf{x};\theta^{(t)}).$$

Principle of the Expectation-Maximization Algorithm

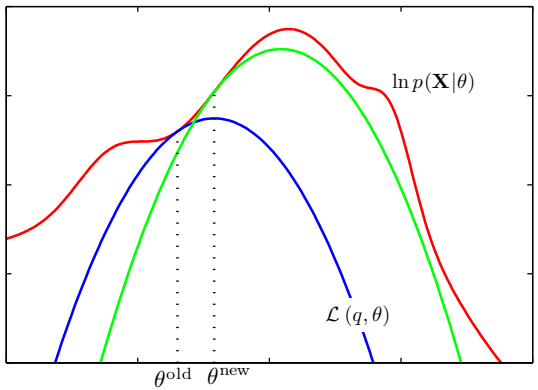
$$\begin{aligned}\log p(\mathbf{x};\boldsymbol{\theta}) &= \log \sum_{\boldsymbol{z}} p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta}) = \log \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})} \\ &= \mathbb{E}_q[\log p(\mathbf{x},\boldsymbol{z};\boldsymbol{\theta})] + H(q) =: \mathcal{L}(q,\boldsymbol{\theta})\end{aligned}$$

- This shows that $\mathcal{L}(q,\boldsymbol{\theta}) \leq \log p(\mathbf{x};\boldsymbol{\theta})$
- Moreover: $\boldsymbol{\theta} \mapsto \mathcal{L}(q,\boldsymbol{\theta})$ is a **concave** function.
- Finally it is possible to show that

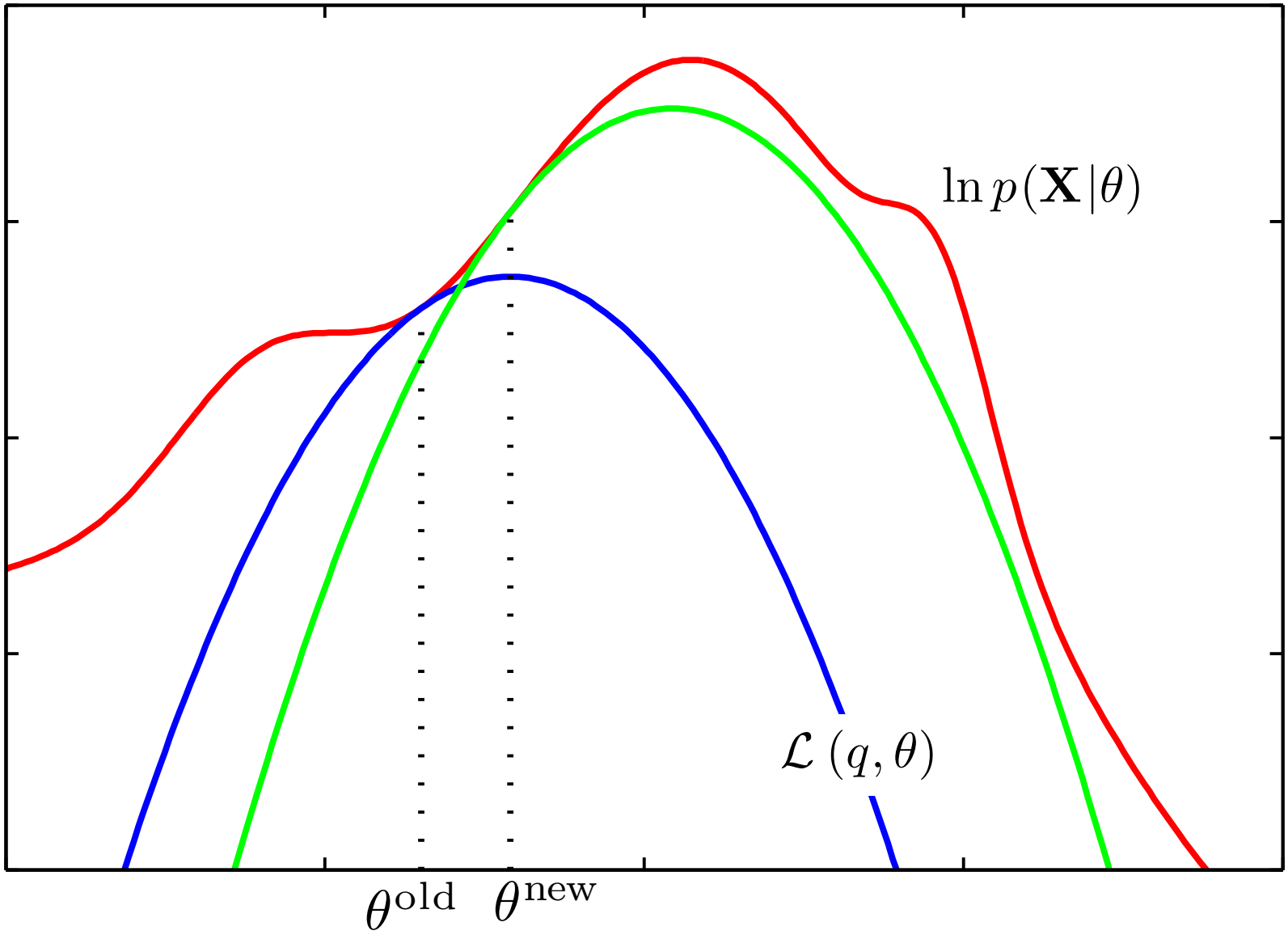
$$\mathcal{L}(q,\boldsymbol{\theta}) = \log p(\mathbf{x};\boldsymbol{\theta}) - KL(q||p(\cdot|\mathbf{x};\boldsymbol{\theta}))$$

So that if we set $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x};\boldsymbol{\theta}^{(t)})$ then

$$L(q,\boldsymbol{\theta}^{(t)}) = p(\mathbf{x};\theta^{(t)}).$$

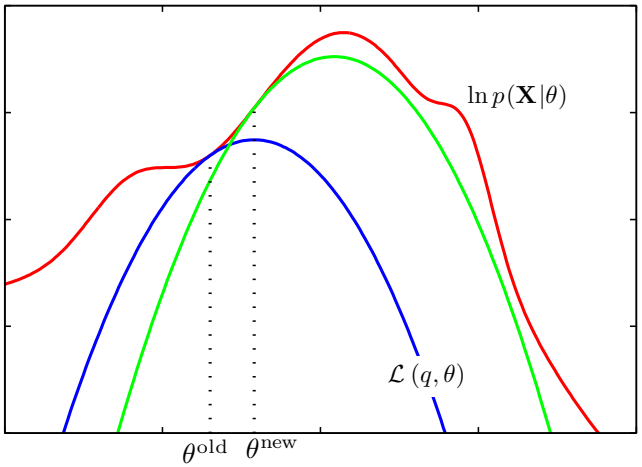


A graphical idea of the EM algorithm



Expectation Maximization algorithm

Expectation step



Maximization step

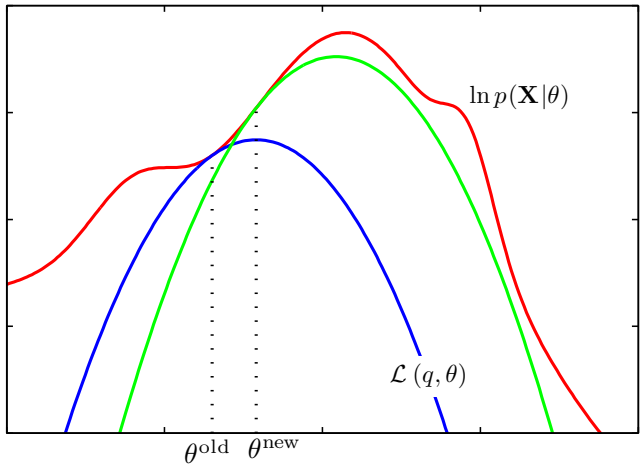
$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{(t-1)}$$

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{(t)}$$

Expectation Maximization algorithm

Expectation step

1. $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x}; \boldsymbol{\theta}^{(t-1)})$



Maximization step

$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{(t-1)}$$

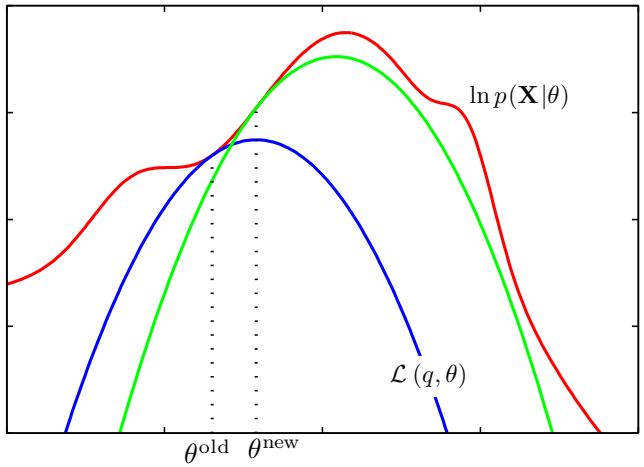
$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{(t)}$$

Expectation Maximization algorithm

Expectation step

1. $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x}; \boldsymbol{\theta}^{(t-1)})$
2. $\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_q\big[\log p(\mathbf{x}, \boldsymbol{z}; \boldsymbol{\theta})\big] + H(q)$

Maximization step



$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{(t-1)}$$

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{(t)}$$

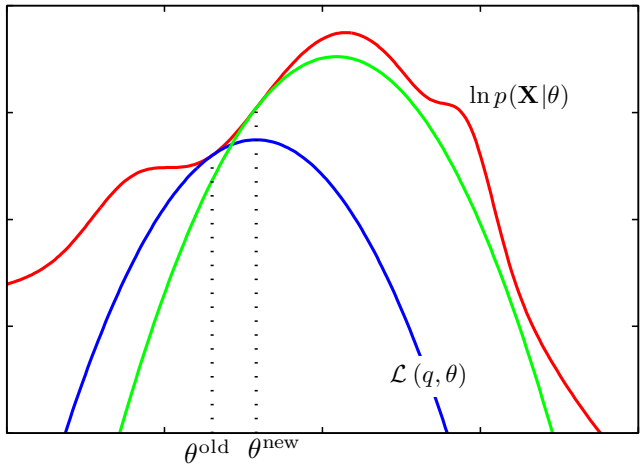
Expectation Maximization algorithm

Expectation step

1. $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x}; \boldsymbol{\theta}^{(t-1)})$
2. $\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_q\big[\log p(\mathbf{x}, \boldsymbol{z}; \boldsymbol{\theta})\big] + H(q)$

Maximization step

1. $\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_q\big[\log p(\mathbf{x}, \boldsymbol{z}; \boldsymbol{\theta})\big]$



$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{(t-1)}$$

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{(t)}$$

Expectation Maximization algorithm

Initialize $\boldsymbol{\theta} = \boldsymbol{\theta}_0$

WHILE (Not converged)

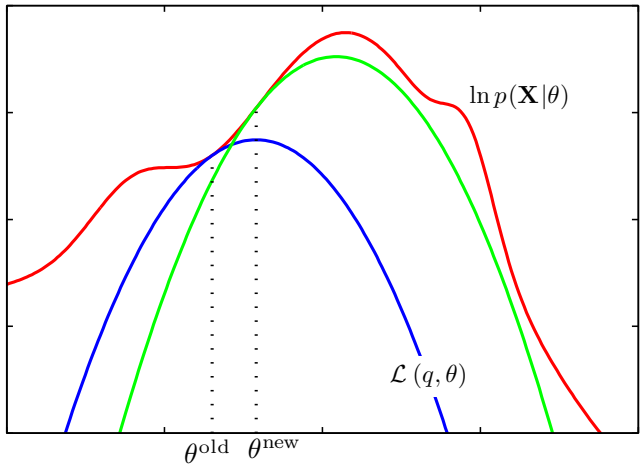
Expectation step

1. $q(\boldsymbol{z}) = p(\boldsymbol{z} \mid \mathbf{x}; \boldsymbol{\theta}^{(t-1)})$
2. $\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_q\big[\log p(\mathbf{x}, \boldsymbol{z}; \boldsymbol{\theta})\big] + H(q)$

Maximization step

1. $\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_q\big[\log p(\mathbf{x}, \boldsymbol{z}; \boldsymbol{\theta})\big]$

ENDWHILE



$$\boldsymbol{\theta}^{\text{old}} = \boldsymbol{\theta}^{(t-1)}$$

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{(t)}$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] \quad =$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] \quad = \quad \mathbb{E}_{q^{(t)}}\left[\log p(\boldsymbol{X}, \boldsymbol{Z}; \theta)\right]$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$\begin{aligned}
\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] &= \mathbb{E}_{q^{(t)}}\left[\log p(\boldsymbol{X}, \boldsymbol{Z}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}; \theta)\right]
\end{aligned}$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$
\begin{aligned}
\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] &= \mathbb{E}_{q^{(t)}}\left[\log p(\boldsymbol{X}, \boldsymbol{Z}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k)\right]
\end{aligned}
$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$
\begin{aligned}
\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] &= \mathbb{E}_{q^{(t)}}\left[\log p(\boldsymbol{X}, \boldsymbol{Z}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k)\right] \\
&= \sum_{i,\,k} \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right] \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right] \log(\pi_k)
\end{aligned}
$$

Expected complete log-likelihood

With the notation: $q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$, we have

$$
\begin{aligned}
\mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right] &= \mathbb{E}_{q^{(t)}}\left[\log p(\boldsymbol{X}, \boldsymbol{Z}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}, \boldsymbol{z}^{(i)}; \theta)\right] \\
&= \mathbb{E}_{q^{(t)}}\left[\sum_{i,k} z_k^{(i)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} z_k^{(i)} \log(\pi_k)\right] \\
&= \sum_{i,\,k} \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right] \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right] \log(\pi_k) \\
&= \sum_{i,\,k} q_{ik}^{(t)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{i,k} q_{ik}^{(t)} \log(\pi_k)
\end{aligned}
$$

Expectation step for the Gaussian mixture

We computed previously $q_i^{(t)}(\boldsymbol{z}^{(i)})$, which is a multinomial distribution defined by

$$q_i^{(t)}(\boldsymbol{z}^{(i)}) = p(\boldsymbol{z}^{(i)}|\mathbf{x}^{(i)}; \theta^{(t-1)})$$

Expectation step for the Gaussian mixture

We computed previously $q_i^{(t)}(\boldsymbol{z}^{(i)})$, which is a multinomial distribution defined by

$$q_i^{(t)}(\boldsymbol{z}^{(i)}) = p(\boldsymbol{z}^{(i)}|\mathbf{x}^{(i)}; \theta^{(t-1)})$$

Abusing notation we will denote $(q_{i1}^{(t)}, \ldots, q_{iK}^{(t)})$ the corresponding vector of probabilities defined by

$$q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$$

Expectation step for the Gaussian mixture

We computed previously $q_i^{(t)}(\boldsymbol{z}^{(i)})$, which is a multinomial distribution defined by

$$q_i^{(t)}(\boldsymbol{z}^{(i)}) = p(\boldsymbol{z}^{(i)}|\mathbf{x}^{(i)}; \theta^{(t-1)})$$

Abusing notation we will denote $(q_{i1}^{(t)}, \ldots, q_{iK}^{(t)})$ the corresponding vector of probabilities defined by

$$q_{ik}^{(t)} = \mathbb{P}_{q_i^{(t)}}(z_k^{(i)} = 1) = \mathbb{E}_{q_i^{(t)}}\left[z_k^{(i)}\right]$$

$$q_{ik}^{(t)} = p(z_k^{(i)} = 1 \mid \mathbf{x}^{(i)}; \theta^{(t-1)}) = \frac{\pi_k^{(t-1)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k^{(t-1)}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_{j=1}^{K} \pi_j^{(t-1)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_j^{(t-1)})}$$

Maximization step for the Gaussian mixture

$$\left(\boldsymbol{\pi}^t, (\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})_{1 \leq k \leq K}\right) = \operatorname*{argmax}_{\theta} \mathbb{E}_{q^{(t)}} \left[\tilde{\ell}(\theta)\right]$$

Maximization step for the Gaussian mixture

$$\left(\boldsymbol{\pi}^t, (\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})_{1 \leq k \leq K}\right) = \underset{\theta}{\text{argmax}}\, \mathbb{E}_{q^{(t)}}\left[\tilde{\ell}(\theta)\right]$$

This yields the updates:

$$\boxed{\boldsymbol{\mu}_k^{(t)} = \frac{\sum_i \mathbf{x}^{(i)}\, q_{ik}^{(t)}}{\sum_i q_{ik}^{(t)}}}, \qquad \boxed{\boldsymbol{\Sigma}_k^{(t)} = \frac{\sum_i \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t)}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t)}\right)^\top q_{ik}^{(t)}}{\sum_i q_{ik}^{(t)}}}$$

and $$\boxed{\pi_k^{(t)} = \frac{\sum_i q_{ik}^{(t)}}{\sum_{i,k'} q_{ik'}^{(t)}}}$$

Final EM algorithm for the Multinomial mixture model

Initialize $\boldsymbol{\theta} = \boldsymbol{\theta}_0$

WHILE (Not converged)

Expectation step

$$q_{ik}^{(t)} \leftarrow \frac{\pi_k^{(t-1)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k^{(t-1)}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_{j=1}^{K} \pi_j^{(t-1)} \log \mathcal{N}(\mathbf{x}^{(i)}, \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_j^{(t-1)})}$$

Maximization step

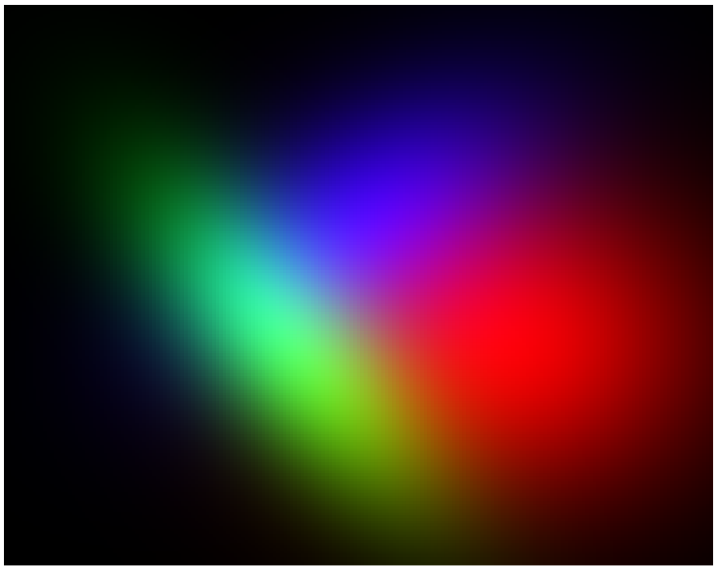
$$\boldsymbol{\mu}_k^{(t)} = \frac{\sum_i \mathbf{x}^{(i)}\, q_{ik}^{(t)}}{\sum_i q_{ik}^{(t)}}, \quad \boldsymbol{\Sigma}_k^{(t)} = \frac{\sum_i \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t)}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t)}\right)^\top q_{ik}^{(t)}}{\sum_i q_{ik}^{(t)}}$$

$$\text{and} \qquad \pi_k^{(t)} = \frac{\sum_i q_{ik}^{(t)}}{\sum_{i,k'} q_{ik'}^{(t)}}$$

ENDWHILE

EM Algorithm for the Gaussian mixture model III

$p(\mathbf{x}|z)$



$p(z|\mathbf{x})$