Lecture 2

Expectation Maximisation and Dynamic Networks

G. Englebienne
M. Poel

University of Twente

Last week, we have seen how the factorisation of probability distributions could be represented as graphs.

- We have seen algorithms to efficiently compute marginal probabilities on such graphs.
- We have seen how this could be used to compute conditional probabilities.
- However in we needed to assume that the parameters of the distributions were known.

Today, we see how to learn these parameters efficiently.

Learning the parameters

We want to find the parameters that maximise the likelihood

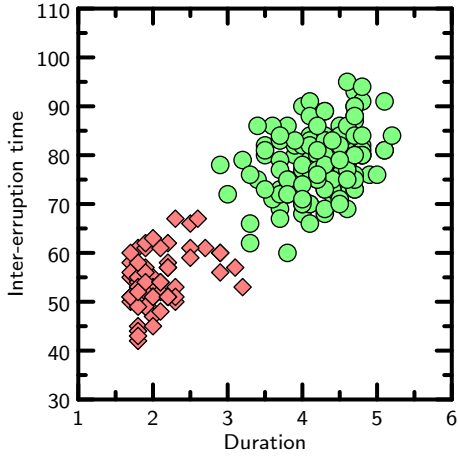
- MAP treatment is very similar
- (exact) full Bayesian treatment is often not tractable
- We'll see how to use sampling instead

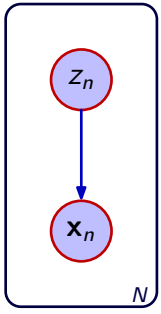
How can we find the maximum of the likelihood?

- If everything is observed, it's easy
- If we have latent variables, it's hard

Fully observed model

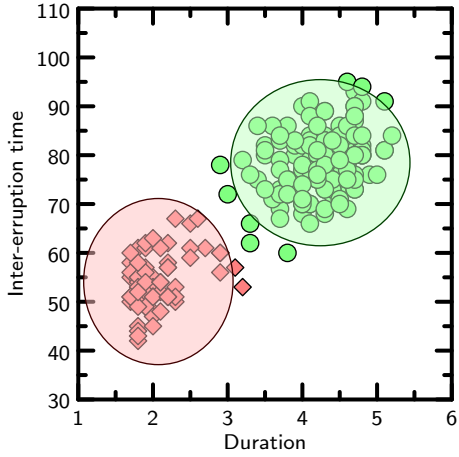
Example

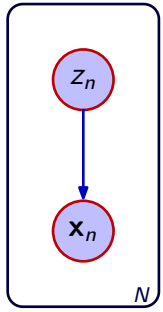




Fully observed model

Example





Fully observed model

$$p(\mathbf{x}_n, z_n) = \begin{cases} p(\mathcal{C}_1)\, p(\mathbf{x}_n|\mathcal{C}_1) & \text{if } z_n = 1 \\ p(\mathcal{C}_2)\, p(\mathbf{x}_n|\mathcal{C}_2) & \text{if } z_n = 0 \end{cases}$$

- Parametrisation:

$$\begin{aligned} p(\mathcal{C}_1) &= \pi \\ p(\mathcal{C}_2) &= 1 - \pi \\ p(\mathbf{x}_n|\mathcal{C}_1) &= \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ p(\mathbf{x}_n|\mathcal{C}_2) &= \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ \Longrightarrow \boldsymbol{\theta} &= \{\pi, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\} \end{aligned}$$

(Complete) likelihood:

$$p(\mathbf{x}_n, \mathbf{z}_n|\theta) = \left[p(\mathcal{C}_1)\, p(\mathbf{x}_n|\mathcal{C}_1)\right]^{z_n} \left[p(\mathcal{C}_2)\, p(\mathbf{x}_n|\mathcal{C}_2)\right]^{1-z_n}$$

Fully observed model

$$p(\mathbf{x}_n, z_n) = \begin{cases} p(\mathcal{C}_1)\, p(\mathbf{x}_n|\mathcal{C}_1) & \text{if } z_n = 1 \\ p(\mathcal{C}_2)\, p(\mathbf{x}_n|\mathcal{C}_2) & \text{if } z_n = 0 \end{cases}$$

- Parametrisation:

$$\begin{aligned} p(\mathcal{C}_1) &= \pi \\ p(\mathcal{C}_2) &= 1 - \pi \\ p(\mathbf{x}_n|\mathcal{C}_1) &= \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \\ p(\mathbf{x}_n|\mathcal{C}_2) &= \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \\ \Longrightarrow \boldsymbol{\theta} &= \{\pi, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\} \end{aligned}$$

(Complete) likelihood:

$$p(\mathbf{x}_n, \mathbf{z}_n|\boldsymbol{\theta}) = \left[p(\mathcal{C}_1)\, p(\mathbf{x}_n|\mathcal{C}_1)\right]^{z_n} \left[p(\mathcal{C}_2)\, p(\mathbf{x}_n|\mathcal{C}_2)\right]^{1-z_n}$$

Parameter optimisation

Complete likelihood:

$$p(\{\mathbf{x}_i, \mathbf{z}_i\}|\boldsymbol{\theta}) = \prod_{i=1}^{N} \left[\pi\, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\right]^{z_i} \left[(1-\pi)\, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right]^{1-z_i}$$

Maximise: take logarithm, set derivative $= 0$

$$\pi_1 = \frac{\sum_{i=1}^{N} z_i}{N} \qquad \pi_2 = \frac{\sum_{i=1}^{N} (1-z_i)}{N}$$

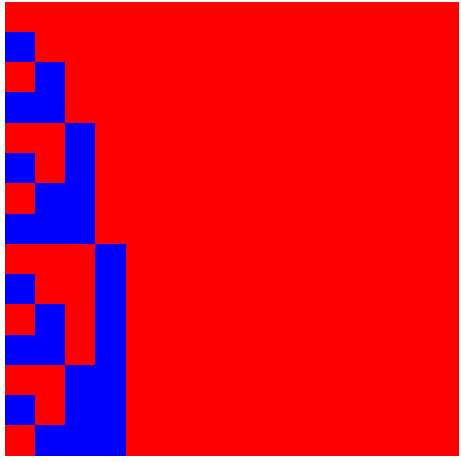
$$\boldsymbol{\mu}_1 = \frac{\sum_{i=1}^{N} z_i\, \mathbf{x}_i}{\sum_{i=1}^{N} z_i} \qquad \boldsymbol{\mu}_2 = \frac{\sum_{i=1}^{N} (1-z_i)\, \mathbf{x}_i}{\sum_{i=1}^{N} z_i}$$

$$\boldsymbol{\Sigma}_1 = \frac{\sum_{i=1}^{N} z_i\, \mathbf{x}_i \mathbf{x}_i^\top}{\sum_{i=1}^{N} z_i} - \boldsymbol{\mu}_1 \boldsymbol{\mu}_1^\top \qquad \boldsymbol{\Sigma}_2 = \frac{\sum_{i=1}^{N} (1-z_i)\, \mathbf{x}_i \mathbf{x}_i^\top}{\sum_{i=1}^{N} z_i} - \boldsymbol{\mu}_2 \boldsymbol{\mu}_2^\top$$

Hidden class label

So, what happens when the class label is not observed?

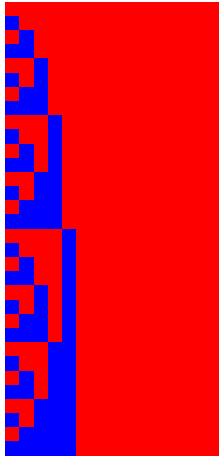
- List all possible assignments, pick best



Hidden class label

So, what happens when the class label is not observed?

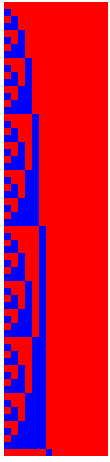
- List all possible assignments, pick best



Hidden class label

So, what happens when the class label is not observed?

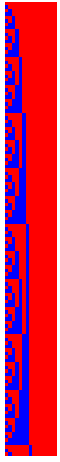
- List all possible assignments, pick best



Hidden class label

So, what happens when the class label is not observed?

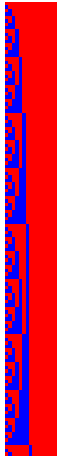
- List all possible assignments, pick best



Hidden class label

So, what happens when the class label is not observed?

- List all possible assignments, pick best



Hidden class label

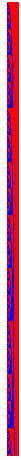
So, what happens when the class label is not observed?

- List all possible assignments, pick best

Hidden class label

So, what happens when the class label is not observed?

- List all possible assignments, pick best



Hidden class label

So, what happens when the class label is not observed?

- List all possible assignments, pick best

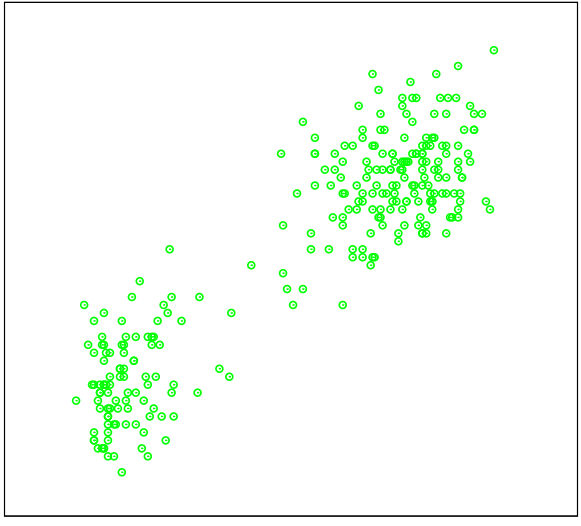
Hidden class label

So, what happens when the class label is not observed?

- List all possible assignments, pick best

Example: kMeans clustering

Cluster data in two groups in an unsupervised manner:



Properties of clustering

In order to cluster the data, we need:

- Some representation of what a cluster looks like
  - Let's assume for now that each cluster is fully defined by its centre.
- An assignment of each datapoint to one of the clusters
  - Let's assume that this is defined by the Euclidean distance to the clusters' centres.

The best configuration is the one where all datapoints are as close as possible to their cluster's centre

Properties of clustering

In order to cluster the data, we need:

- Some representation of what a cluster looks like
  - Let's assume for now that each cluster is fully defined by its centre.
- An assignment of each datapoint to one of the clusters
  - Let's assume that this is defined by the Euclidean distance to the clusters' centres.

The best configuration is the one where all datapoints are as close as possible to their cluster's centre

Properties of clustering

In order to cluster the data, we need:

- Some representation of what a cluster looks like
  - Let's assume for now that each cluster is fully defined by its centre.
- An assignment of each datapoint to one of the clusters
  - Let's assume that this is defined by the Euclidean distance to the clusters' centres.

The best configuration is the one where all datapoints are as close as possible to their cluster's centre

Optimising the clustering

An exhaustive search for the optimal clustering is intractable and requires C^N operations

- Where C is the number of clusters
- and N is the number of datapoints.

How do we find the optimal clustering without exhaustive search? Solve the clustering iteratively:

- Initialise the cluster means at random
- Repeat until convergence
  1. Assign each data point to the closest cluster mean
  2. Update each cluster's centre according to the associated data

Optimising the clustering

An exhaustive search for the optimal clustering is intractable and requires C^N operations

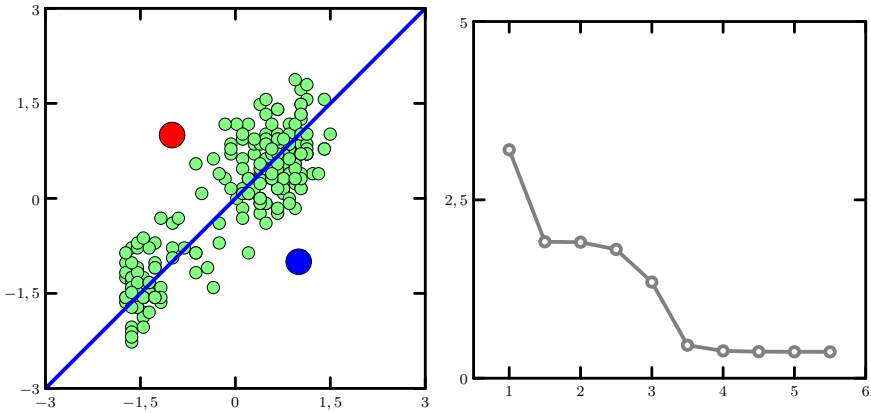
- Where C is the number of clusters
- and N is the number of datapoints.

How do we find the optimal clustering without exhaustive search? Solve the clustering iteratively:

- Initialise the cluster means at random
- Repeat until convergence
  1. Assign each data point to the closest cluster mean
  2. Update each cluster's centre according to the associated data

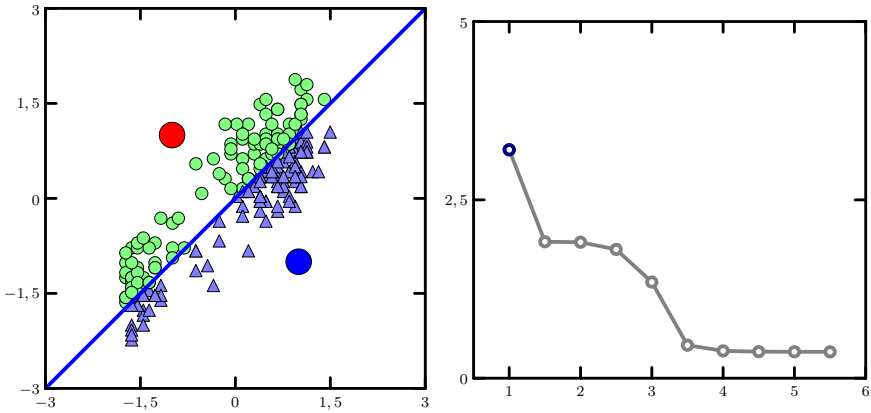
An iterative algorithm

Example: kMeans Clustering



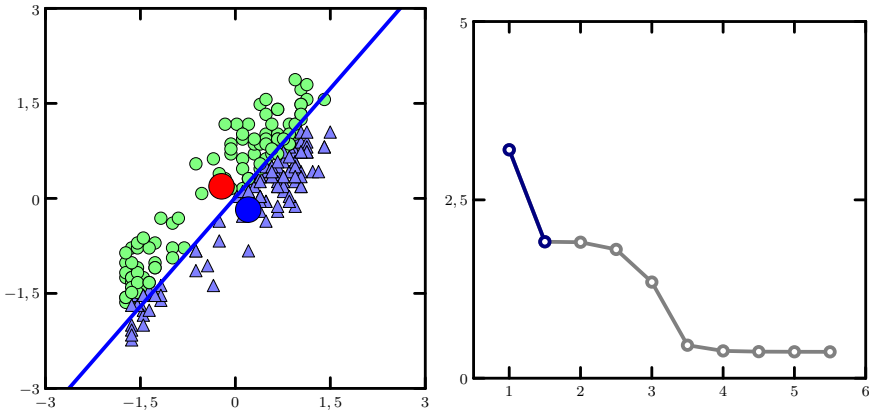
An iterative algorithm

Example: kMeans Clustering



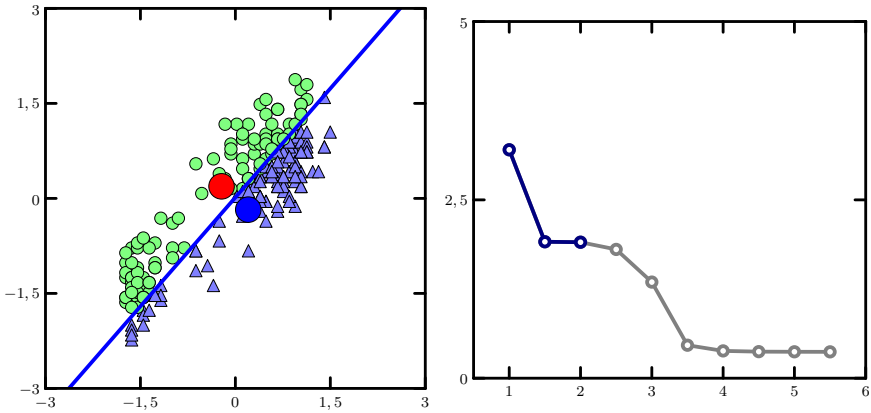
An iterative algorithm

Example: kMeans Clustering



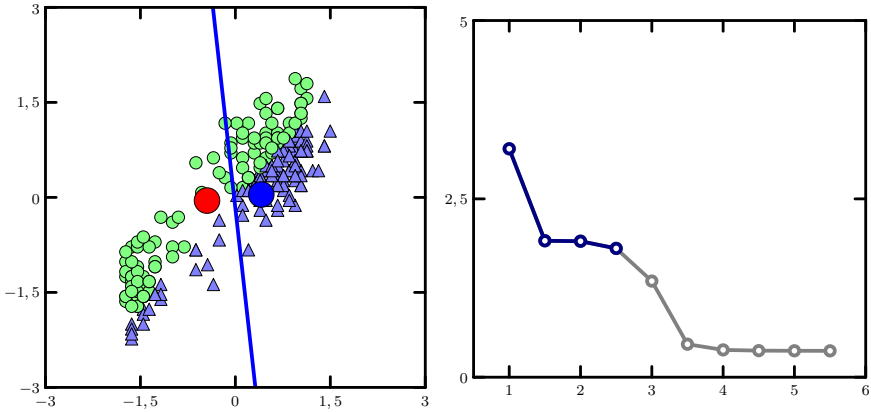
An iterative algorithm

Example: kMeans Clustering



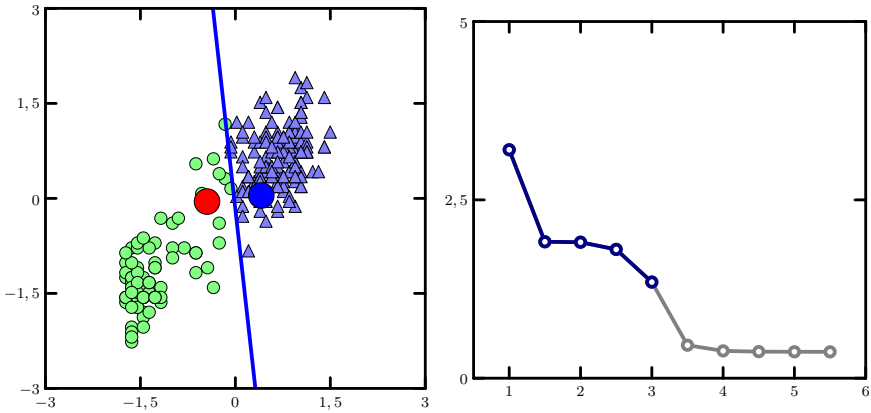
An iterative algorithm

Example: kMeans Clustering



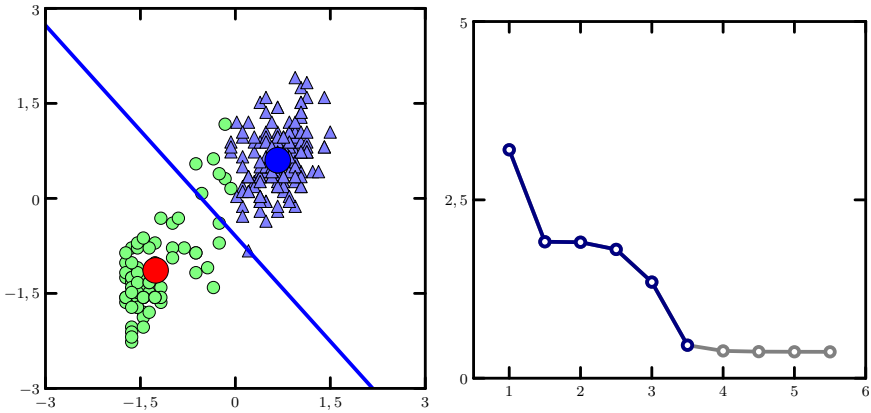
An iterative algorithm

Example: kMeans Clustering



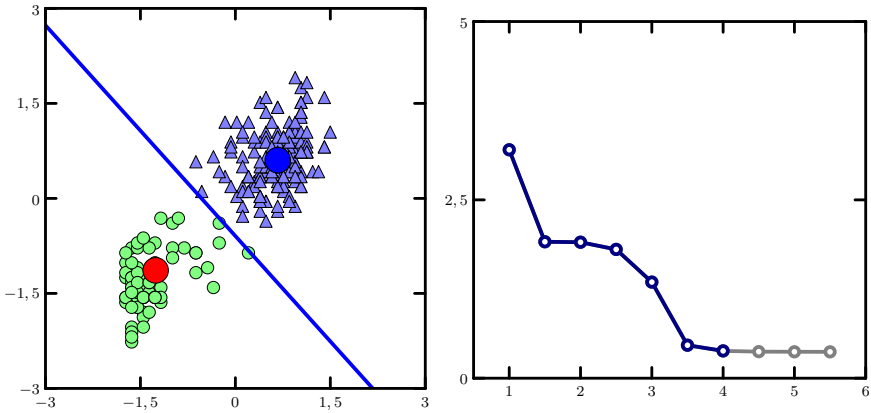
An iterative algorithm

Example: kMeans Clustering



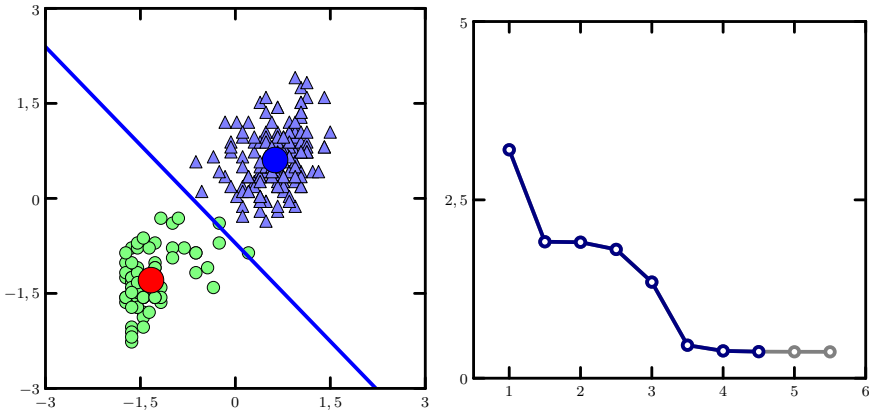
An iterative algorithm

Example: kMeans Clustering



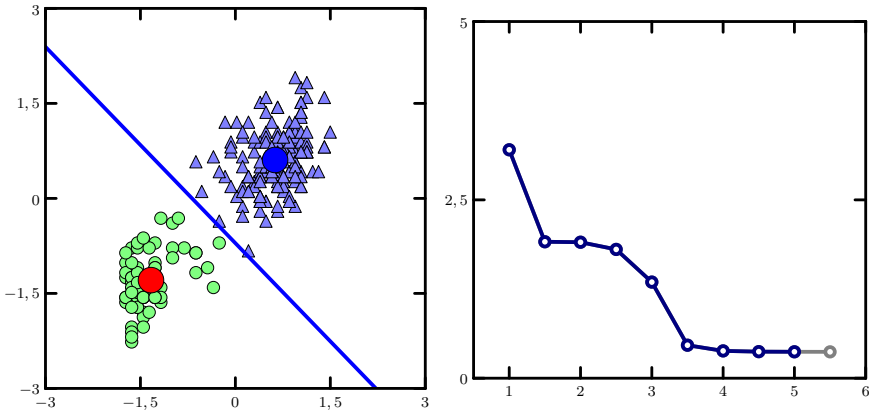
An iterative algorithm

Example: kMeans Clustering



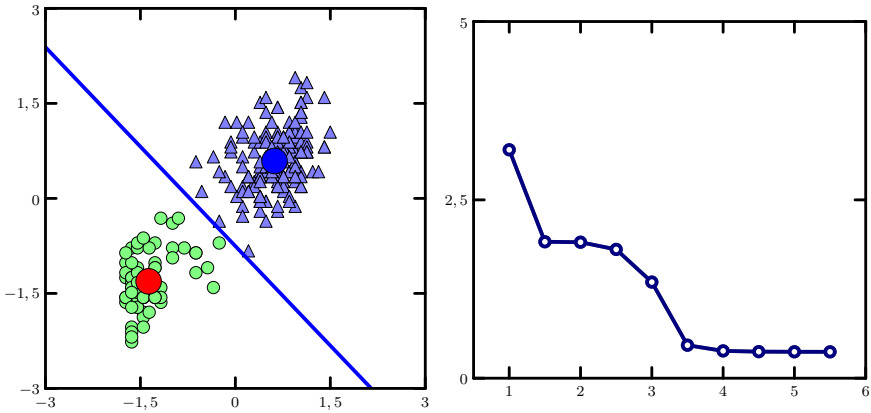
An iterative algorithm

Example: kMeans Clustering



An iterative algorithm

Example: kMeans Clustering



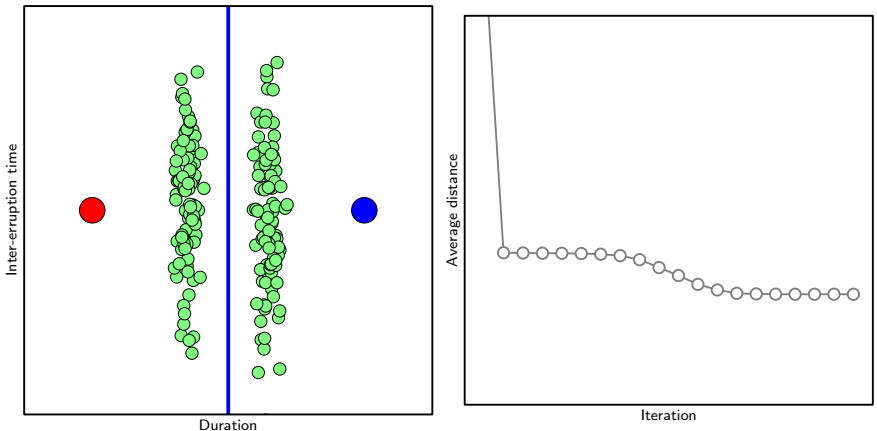
Problems with kMeans

There are some disadvantages to kMeans:

- **Euclidean distance**:
  - Only useful for some types of data
  - Not robust to outliers
  - Sensitive to scaling of data
  - Solution: Other distance measures
- **Hard assignments** At each iteration, each datapoint is assigned to exactly one cluster, even for doubtful cases.

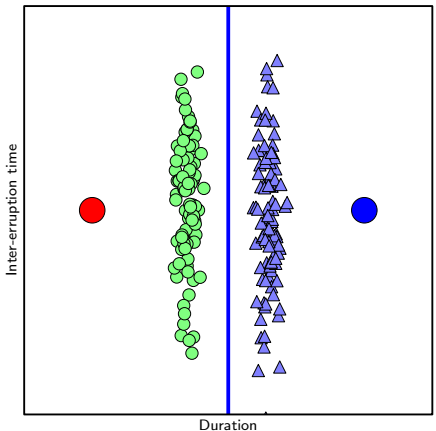
k-means sensitivity to scaling

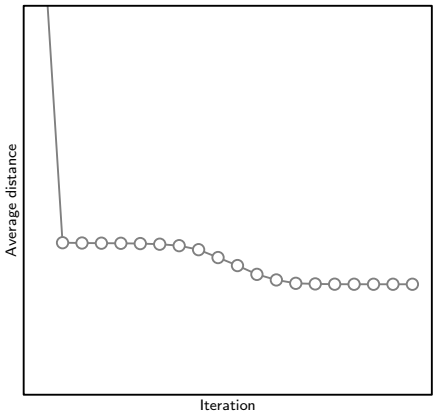
Example: kMeans Clustering



k-means sensitivity to scaling

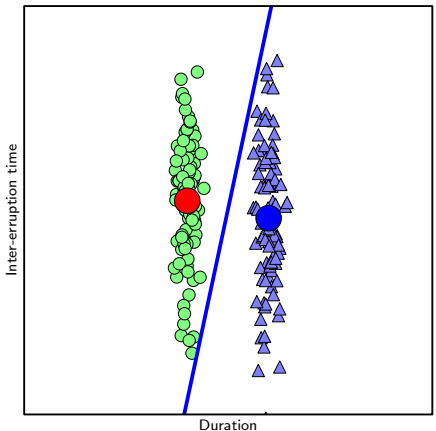
Example: kMeans Clustering

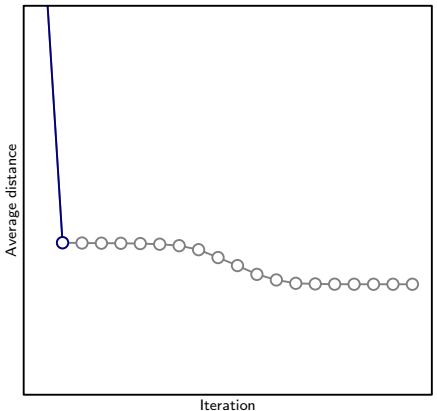




k-means sensitivity to scaling

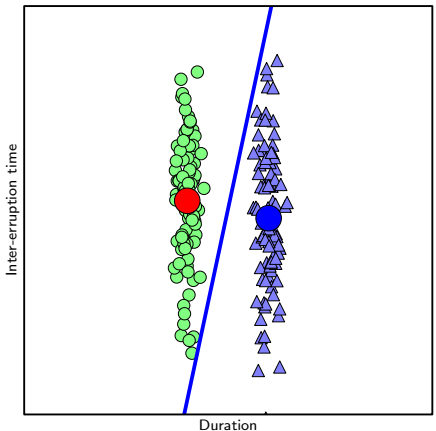
Example: kMeans Clustering

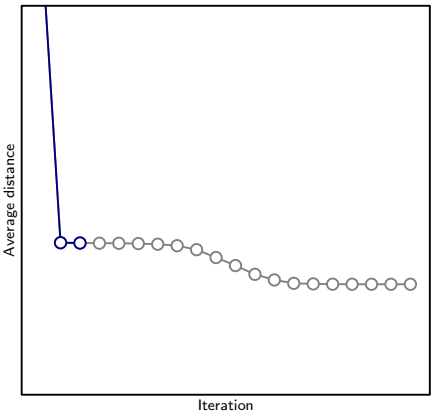




k-means sensitivity to scaling

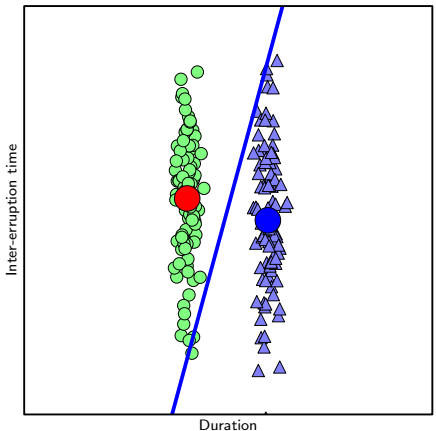
Example: kMeans Clustering

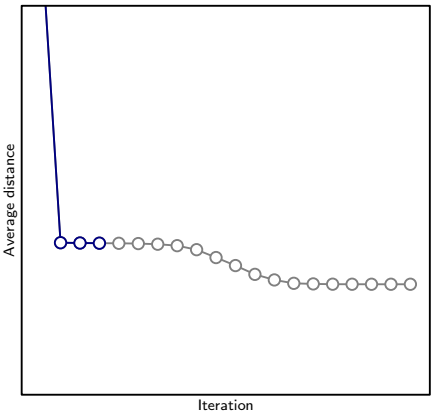




k-means sensitivity to scaling

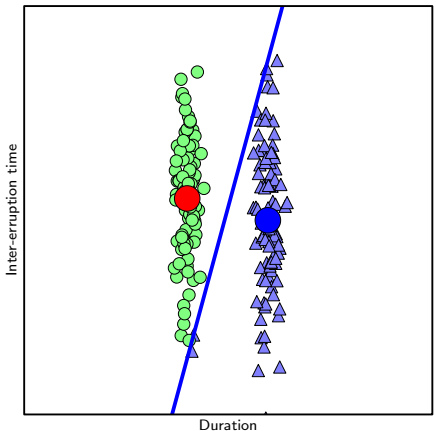
Example: kMeans Clustering

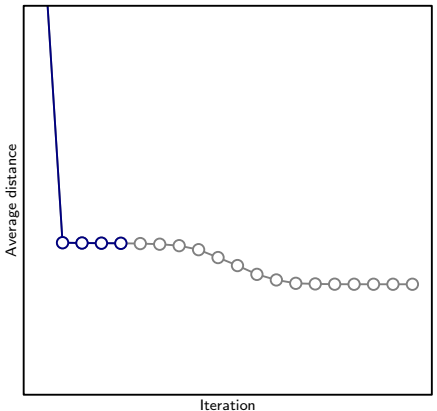




k-means sensitivity to scaling

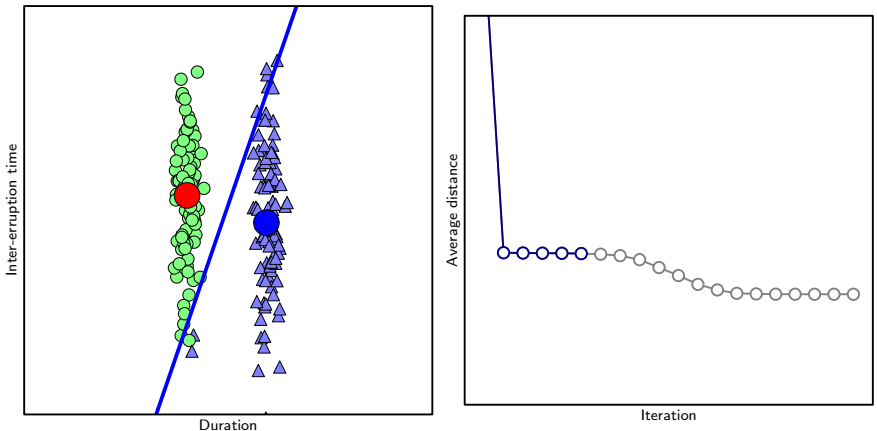
Example: kMeans Clustering





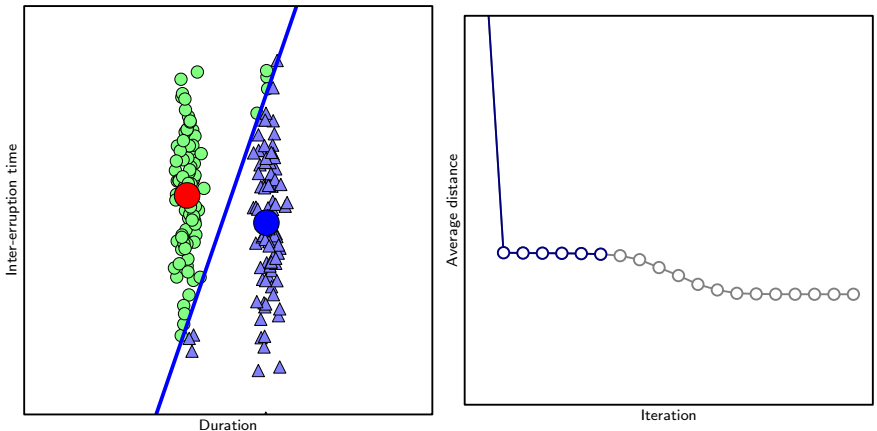
k-means sensitivity to scaling

Example: kMeans Clustering



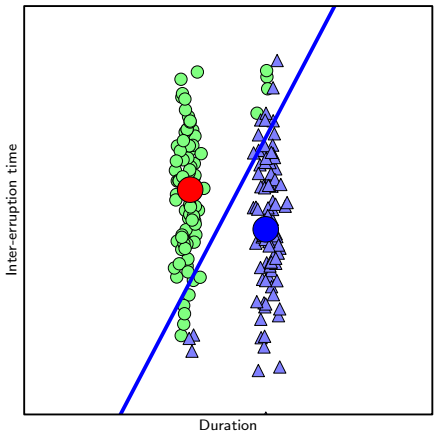
k-means sensitivity to scaling

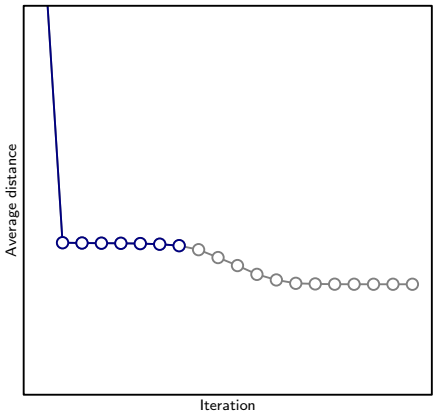
Example: kMeans Clustering



k-means sensitivity to scaling

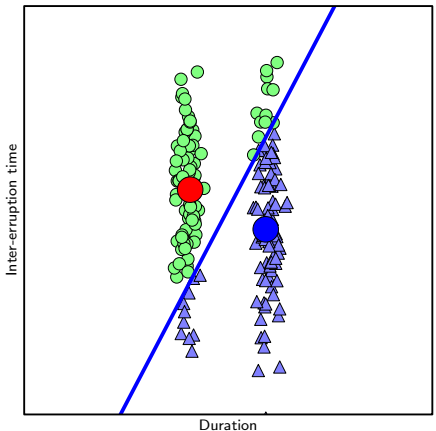
Example: kMeans Clustering

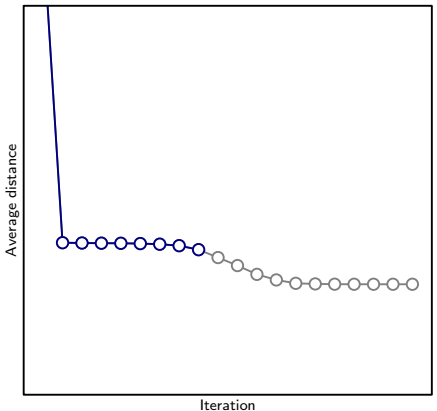




k-means sensitivity to scaling

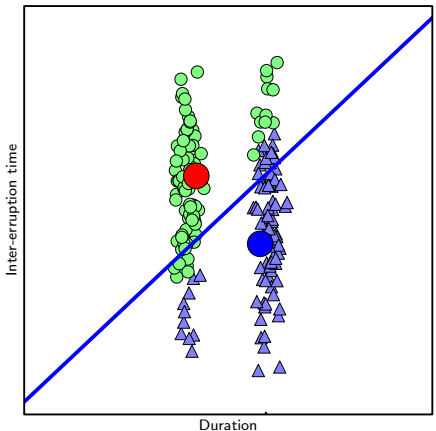
Example: kMeans Clustering

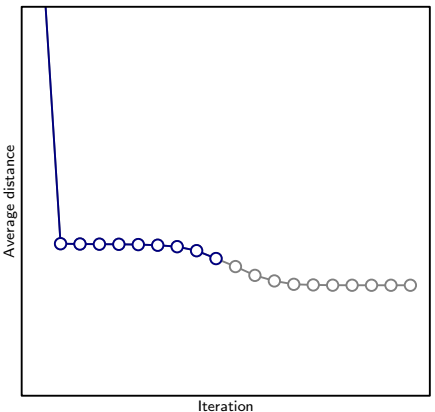




k-means sensitivity to scaling

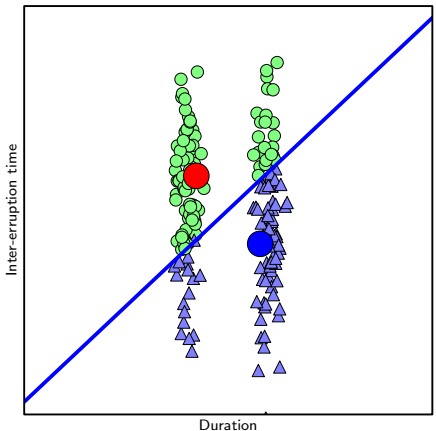
Example: kMeans Clustering

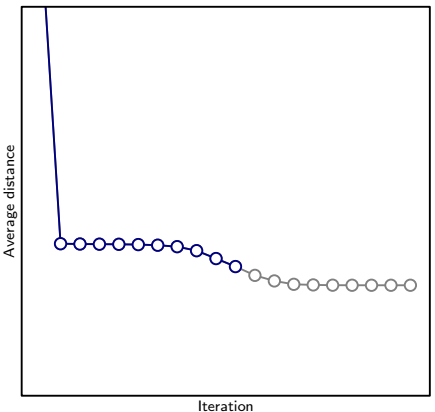




k-means sensitivity to scaling

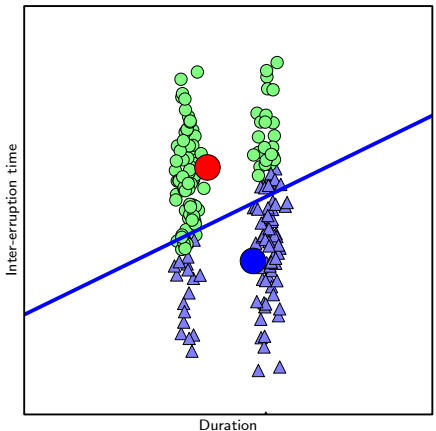
Example: kMeans Clustering

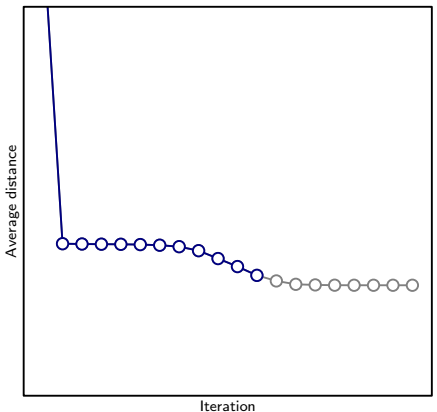




k-means sensitivity to scaling

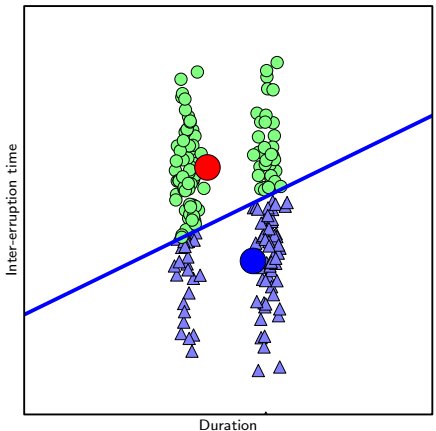
Example: kMeans Clustering

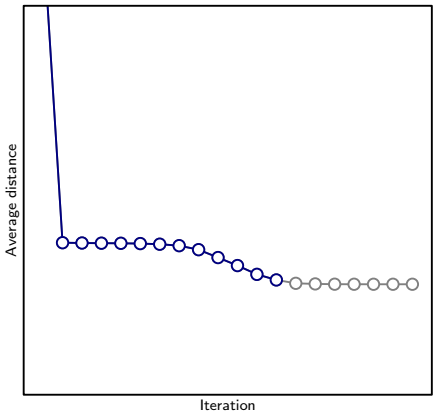




k-means sensitivity to scaling

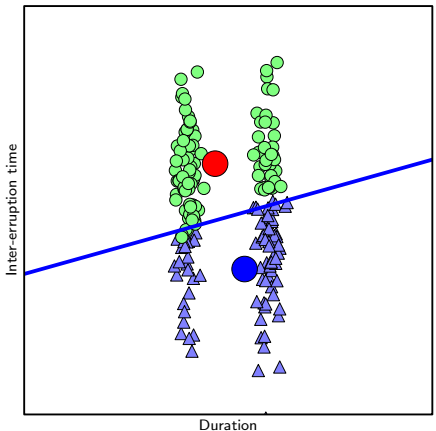
Example: kMeans Clustering

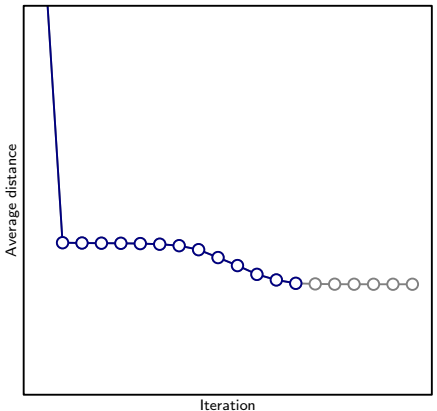




k-means sensitivity to scaling

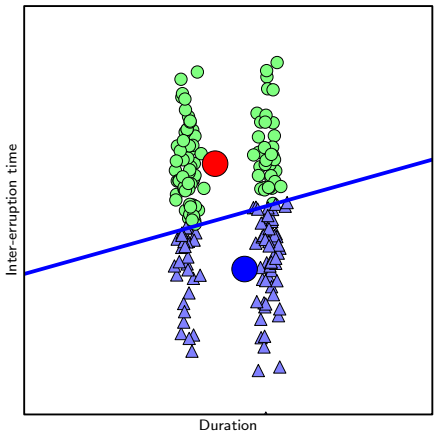
Example: kMeans Clustering

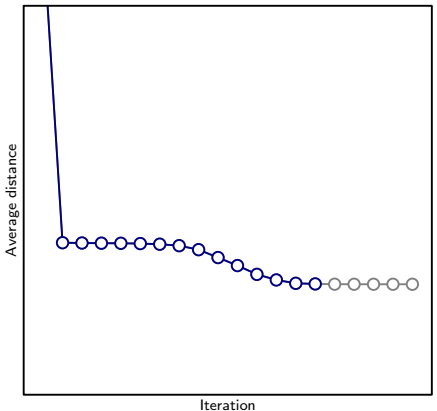




k-means sensitivity to scaling

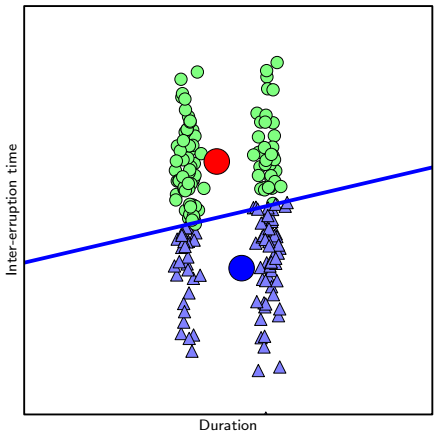
Example: kMeans Clustering

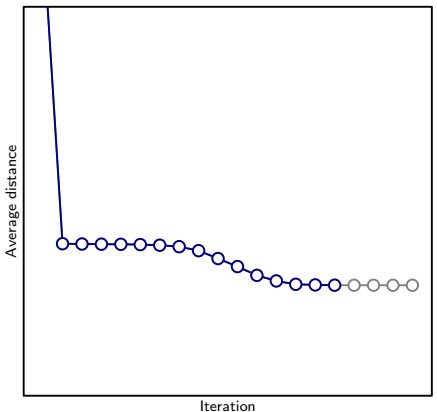




k-means sensitivity to scaling

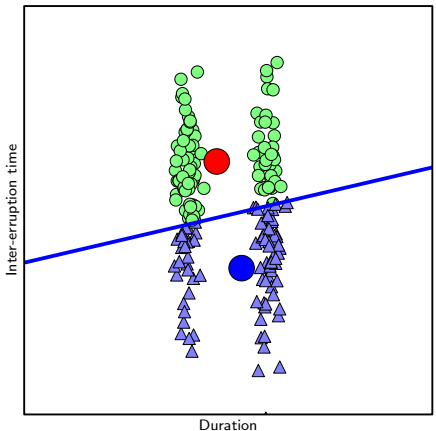
Example: kMeans Clustering

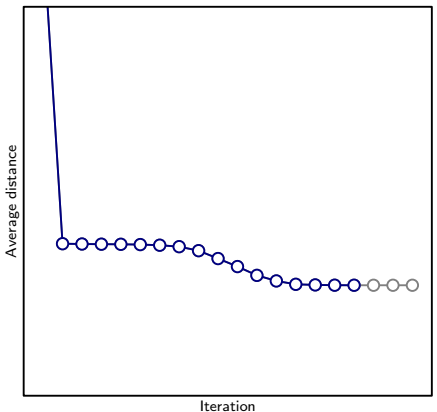




k-means sensitivity to scaling

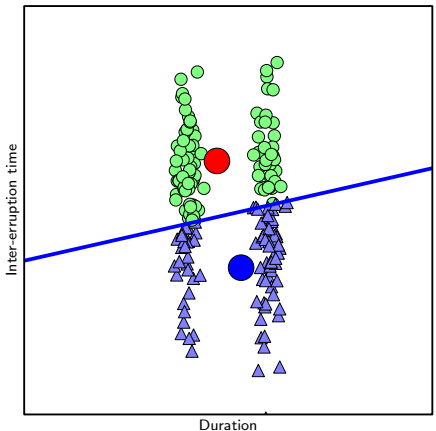
Example: kMeans Clustering

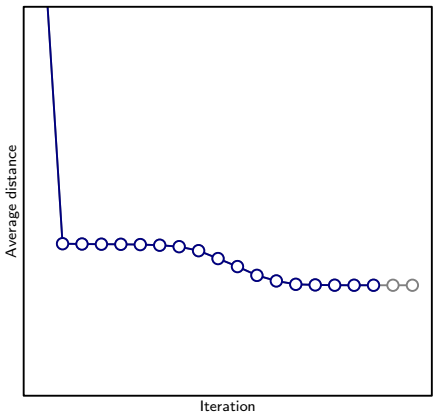




k-means sensitivity to scaling

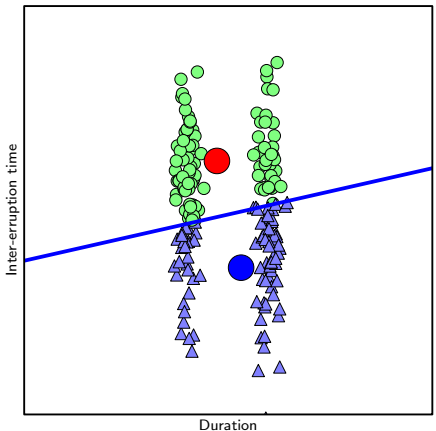
Example: kMeans Clustering

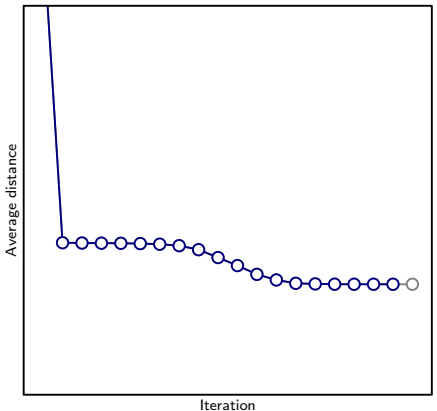




k-means sensitivity to scaling

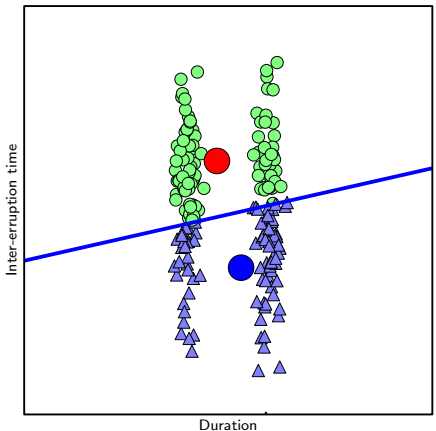
Example: kMeans Clustering

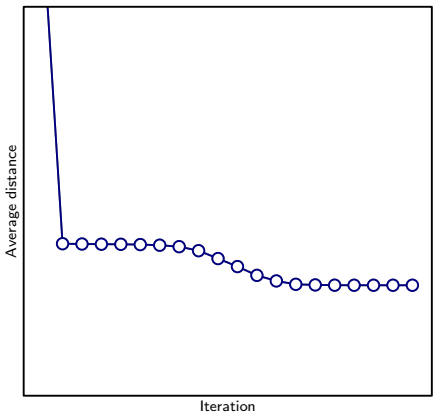




k-means sensitivity to scaling

Example: kMeans Clustering





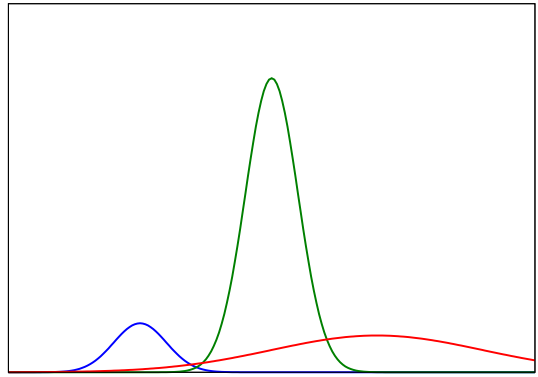
Mixtures of Gaussians

A mixture of Gaussians is a linear combination of Gaussians:

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

where $0 \leqslant \pi_k \leqslant 1$ and

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (2)$$



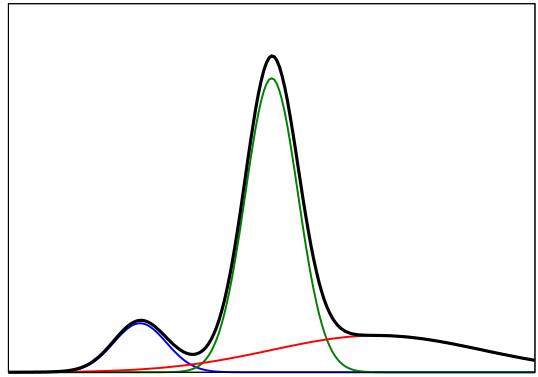
Mixtures of Gaussians

A mixture of Gaussians is a linear combination of Gaussians:

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

where $0 \leqslant \pi_k \leqslant 1$ and

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (2)$$



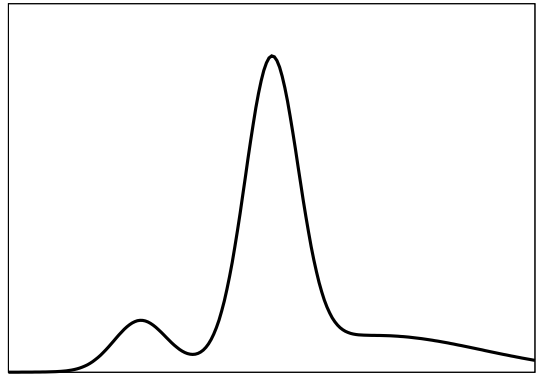
Mixtures of Gaussians

A mixture of Gaussians is a linear combination of Gaussians:

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

where $0 \leqslant \pi_k \leqslant 1$ and

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (2)$$



Alternative view of the mixture model

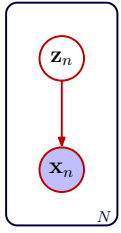
We introduce a binary random variable $\mathbf{z}$ in 1-of-K encoding, the probability of $\mathbf{z}$ can be writen as

$$p(z_k = 1) = \pi_k \text{ so that } p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

We choose the conditional distribution of $\mathbf{x}$ given a z_k as

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \text{ so that}$$

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



Then we have:

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

Optimising the likelihood

Similar to kMeans, finding the optimal parameters for $p(\mathbf{x})$ in a mixture of Gaussians is hard.
However, with our new representation, we can now work with $p(\mathbf{x}, \mathbf{z})$ rather than $p(\mathbf{x})$.
In particular, consider the log-likelihood of N datapoints $\mathbf{X}$:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right] \tag{3}$$

Setting the first derivative with respect to μ_k equal to zero, gives:

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_\ell \pi_\ell \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)} \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{4}$$

Optimising the likelihood

Similar to kMeans, finding the optimal parameters for $p(\mathbf{x})$ in a mixture of Gaussians is hard.
However, with our new representation, we can now work with $p(\mathbf{x}, \mathbf{z})$ rather than $p(\mathbf{x})$.
In particular, consider the log-likelihood of N datapoints $\mathbf{X}$:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right] \tag{3}$$

Setting the first derivative with respect to μ_k equal to zero, gives:

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_\ell \pi_\ell \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}}_{p(z_k|\mathbf{x}_n)} \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{4}$$

Likelihood maximisation

Using the notation $N_k = \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)$, we obtain that the values for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k that maximise the likelihood are:

$$\pi_k = \frac{N_k}{N}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)\, \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)\, (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

However $p(z_k|\mathbf{x}_n)$ is a function of $\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$

Likelihood maximisation

Using the notation $N_k = \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)$, we obtain that the values for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k that maximise the likelihood are:

$$\pi_k = \frac{N_k}{N}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)\, \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)\, (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

However $p(z_k|\mathbf{x}_n)$ is a function of $\pi_1, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$

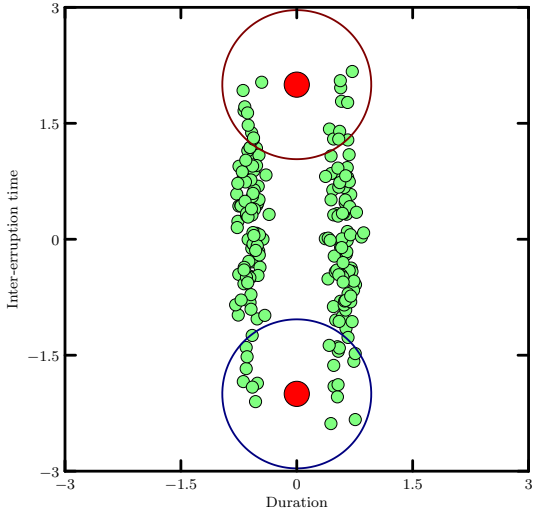
The Expectation-Maximisation algorithm is an iterative update where:

E-step Compute the posterior probabilities of the latent variables given the data and the current parameters (also called *responsibilities*), $p(z_k|\mathbf{x}_n, \boldsymbol{\theta})$

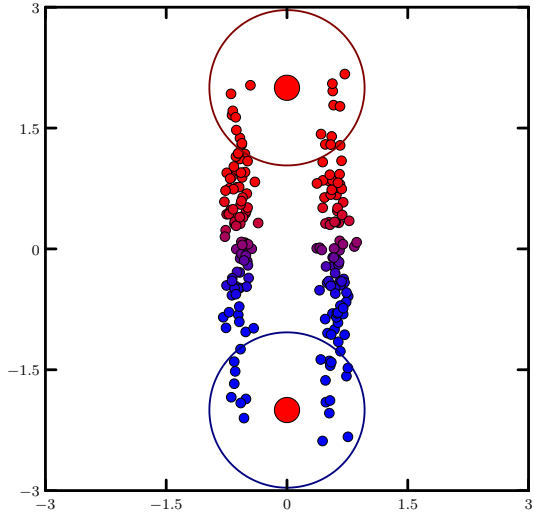
M-step Optimise the expectation of the complete log-likelihood with respect to the parameters

In practice, stop when the increase in likelihood falls below a certain threshold.

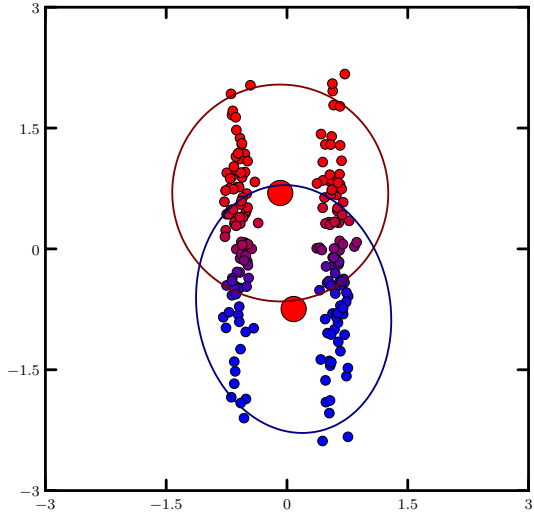
Example: Mixtures of Gaussians



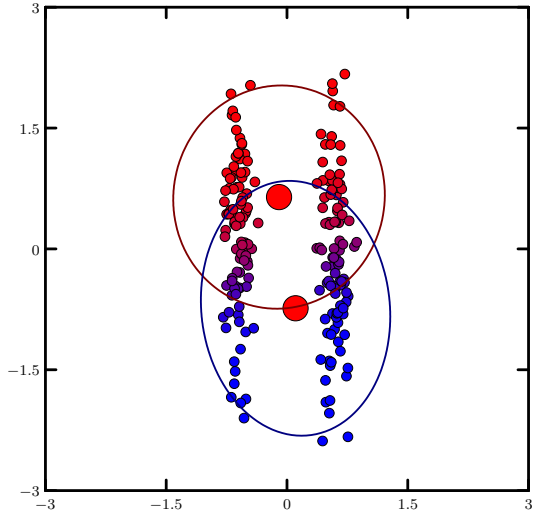
Example: Mixtures of Gaussians



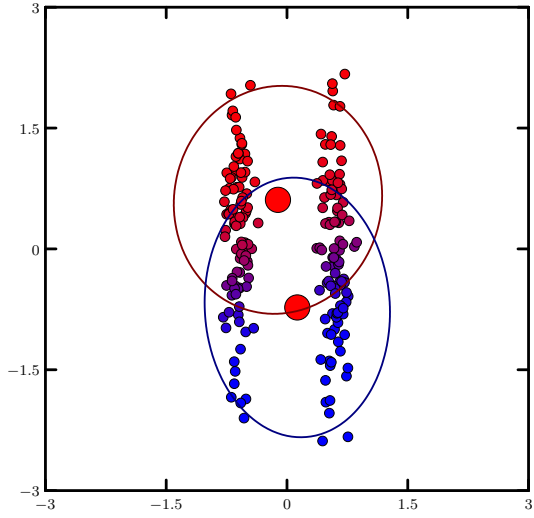
Example: Mixtures of Gaussians



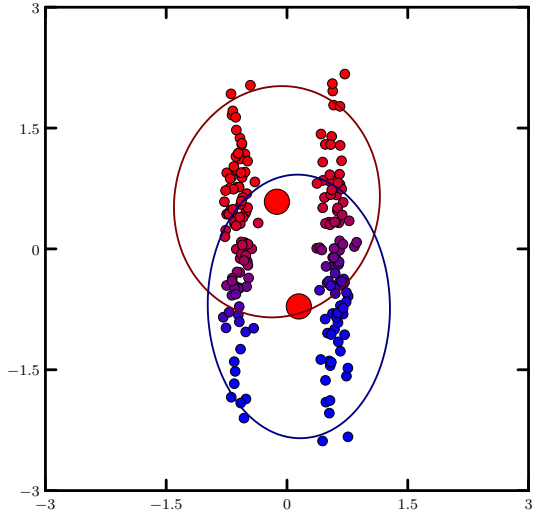
Example: Mixtures of Gaussians



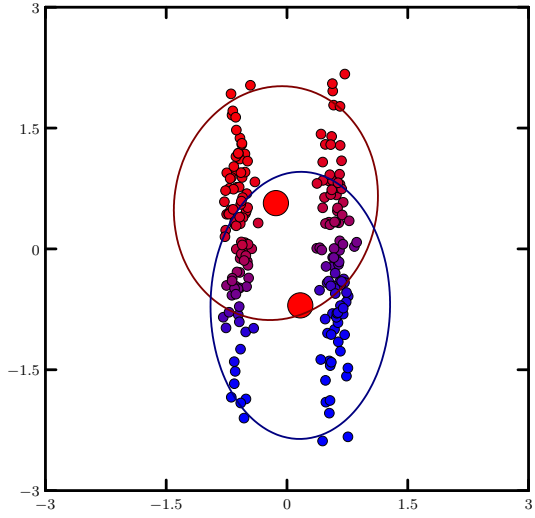
Example: Mixtures of Gaussians



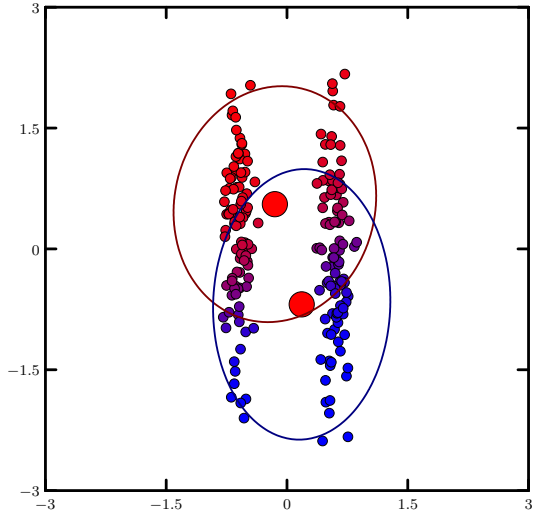
Example: Mixtures of Gaussians



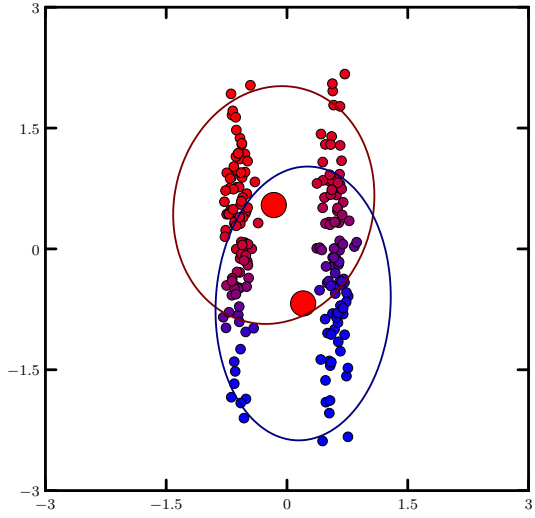
Example: Mixtures of Gaussians



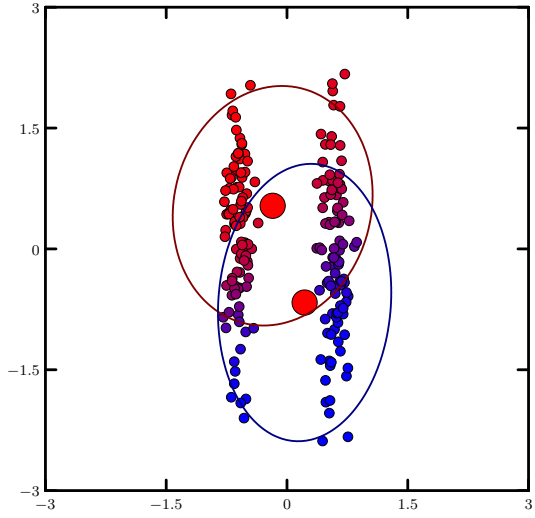
Example: Mixtures of Gaussians



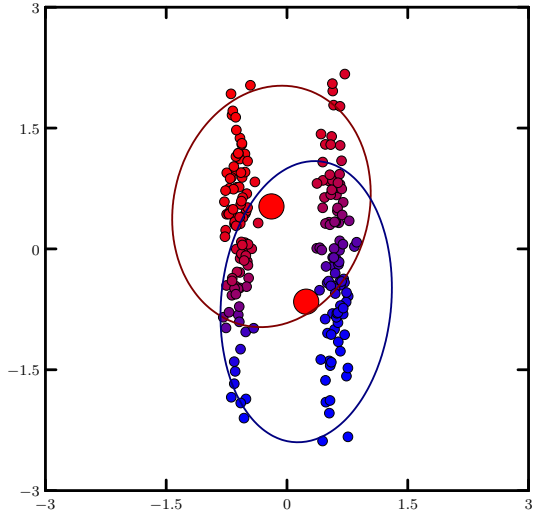
Example: Mixtures of Gaussians



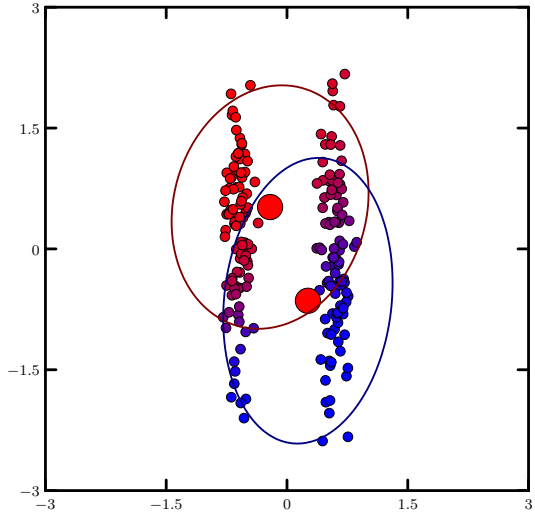
Example: Mixtures of Gaussians



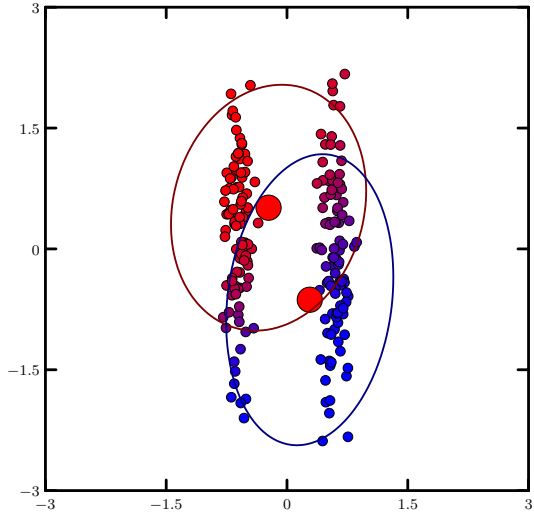
Example: Mixtures of Gaussians



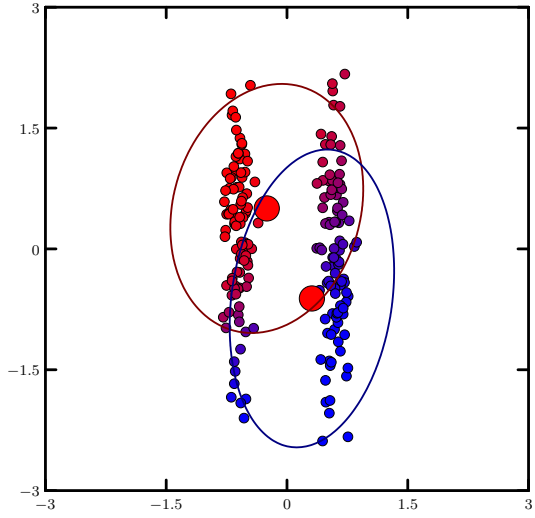
Example: Mixtures of Gaussians



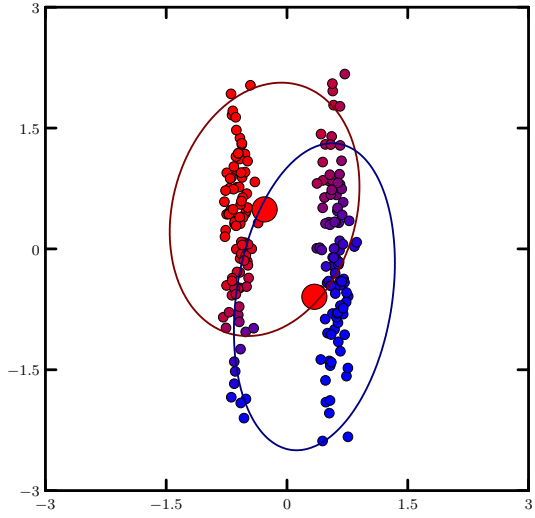
Example: Mixtures of Gaussians



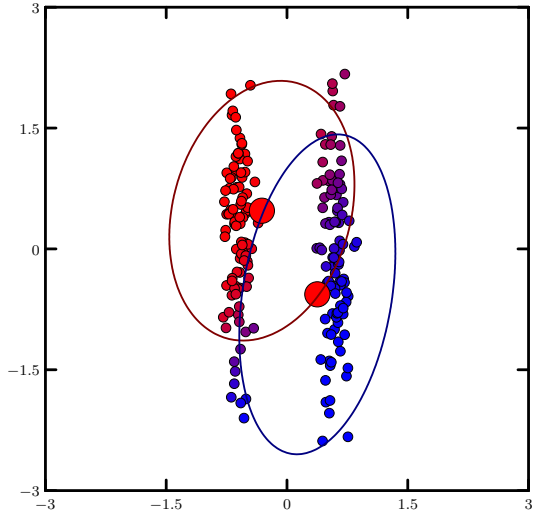
Example: Mixtures of Gaussians



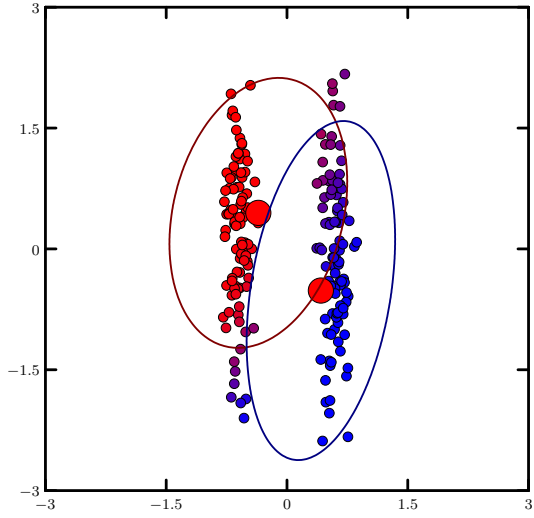
Example: Mixtures of Gaussians



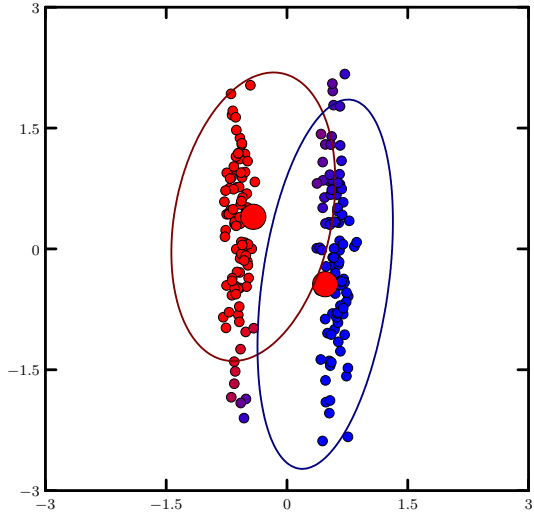
Example: Mixtures of Gaussians



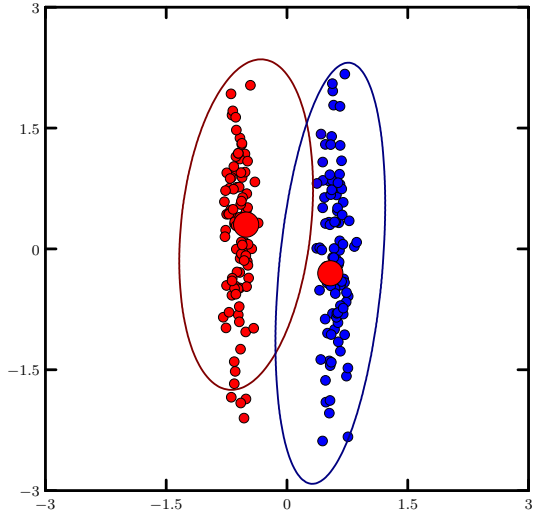
Example: Mixtures of Gaussians



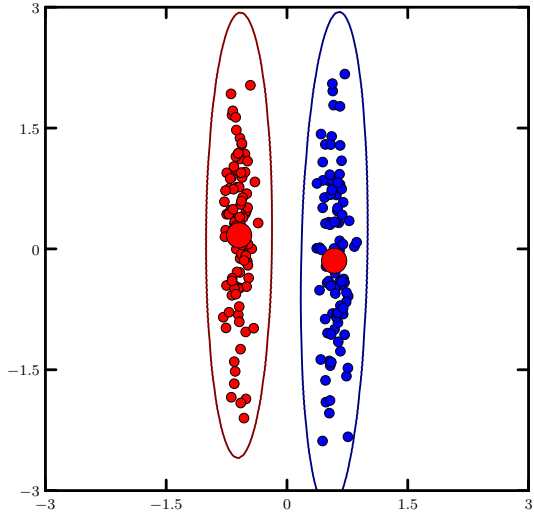
Example: Mixtures of Gaussians



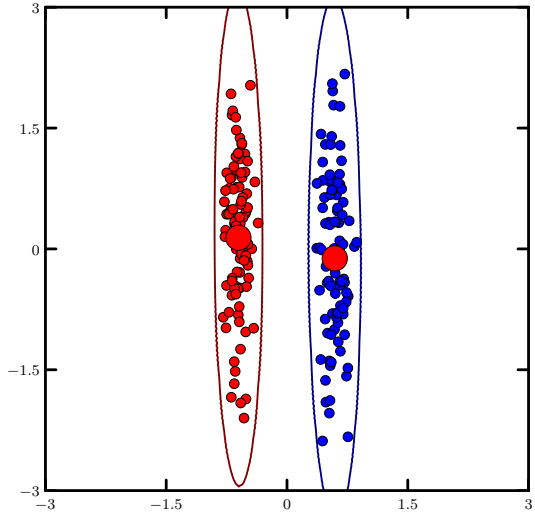
Example: Mixtures of Gaussians



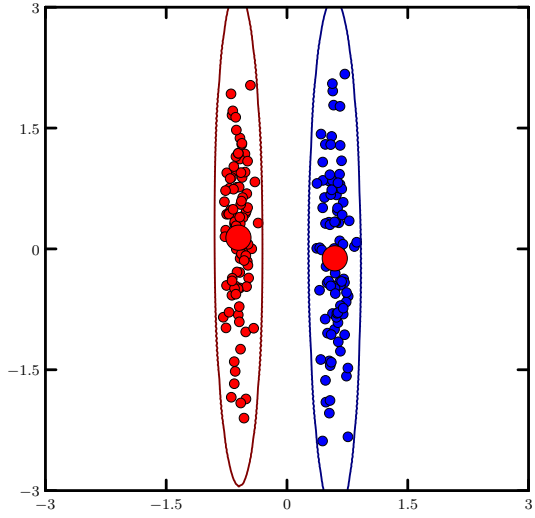
Example: Mixtures of Gaussians



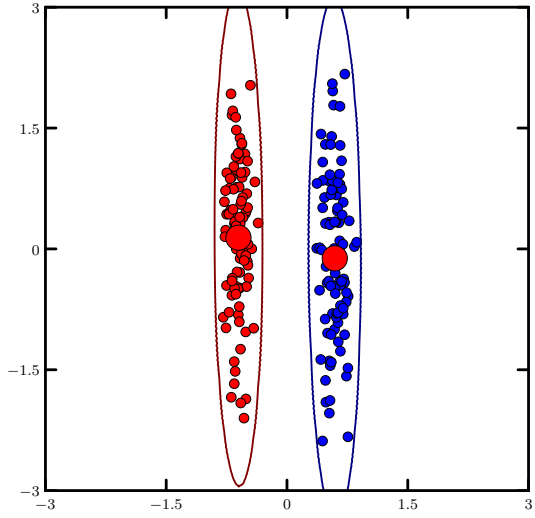
Example: Mixtures of Gaussians



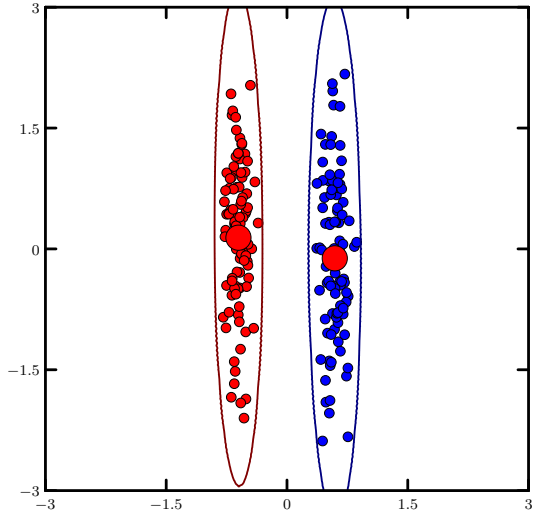
Example: Mixtures of Gaussians



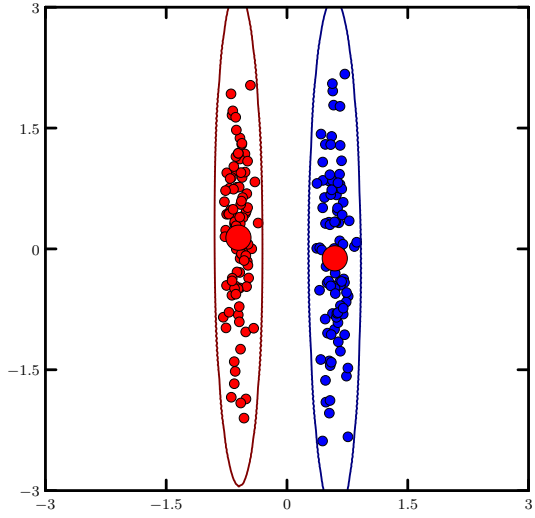
Example: Mixtures of Gaussians



Example: Mixtures of Gaussians



Example: Mixtures of Gaussians



About EM for mixtures of Gaussians

Some things to notice

- The problem is ill-posed: consider a component k with covariance $\sigma \mathbf{I}$. If the mean of the component falls exactly on a data point, its contribution to the likelihood is

$$\ln p(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{2\pi}\sigma} \tag{5}$$

  which, in the limit of $\sigma \to 0$ goes to infinity.
- A suitable prior on $\boldsymbol{\theta}$ avoids this problem
- The kMeans algorithm is equivalent with EM for a Gaussian mixture model, where the covariance is $\sigma \mathbf{I}$ for *all* mixture components in the limit $\sigma \to 0$.

In general, the EM algorithm is defined as follows. Optimising the *complete* log-likelihood

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{6}$$

would be easy, but we only observe $\mathbf{X}$. So let's optimise our best estimate of the complete log-likelihood: the expectation of the complete log-likelihood under our current parameter estimates $\boldsymbol{\theta}^{old}$:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \mathbb{E}_{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] \tag{7}$$

$$= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{8}$$

The E and M steps

- In the E-step we evaluate the distribution of the latent variables

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old}) \tag{9}$$

  so that we can compute the expectation of the complete log-likelihood
- In the M-step we maximise the complete log-likelihood with respect to $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{new} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) \tag{10}$$

If we incorporate a prior over $\boldsymbol{\theta}$, $p(\boldsymbol{\theta})$, the EM algorithm changes slightly:

E-Step : $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})$ does not depend on $p(\boldsymbol{\theta}^{old})$ since $\boldsymbol{\theta}^{old}$ are given, so the E-step remains identical.

M-Step : We now optimise

$$\mathbb{E}\left[\ln(p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})p(\boldsymbol{\theta}))\right] \tag{11}$$

where $p(\boldsymbol{\theta})$ is constant with respect to $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{old})$, so that we get:

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) + \ln p(\boldsymbol{\theta}) \tag{12}$$

Lower bound: why EM works

Consider the quantity

$$\begin{aligned}
&\sum_{\mathbf{z}} q(\mathbf{z}) \ln[\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}] \\
&\qquad = \sum_{\mathbf{z}} q(\mathbf{z}) \left[\ln p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) + \ln p(\mathbf{x}|\boldsymbol{\theta}) - \ln q(\mathbf{z})\right] \\
&\qquad = \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{q(\mathbf{z})} + \ln p(\mathbf{x}|\boldsymbol{\theta}) \underbrace{\sum_{\mathbf{z}} q(\mathbf{z})}_{=1}
\end{aligned}$$

so that

$$\ln p(\mathbf{x}|\boldsymbol{\theta}) = \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \ln[\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}]}_{\mathcal{L}(q,\boldsymbol{\theta})} + \underbrace{\left(-\sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{q(\mathbf{z})}\right)}_{KL(q||p)}$$

The KL-divergence is positive

Consider the log likelihood

$$1 = \sum_{\mathbf{z}} p(\mathbf{z}) = \sum_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})}$$

$$\ln 1 = \ln \sum_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})}$$

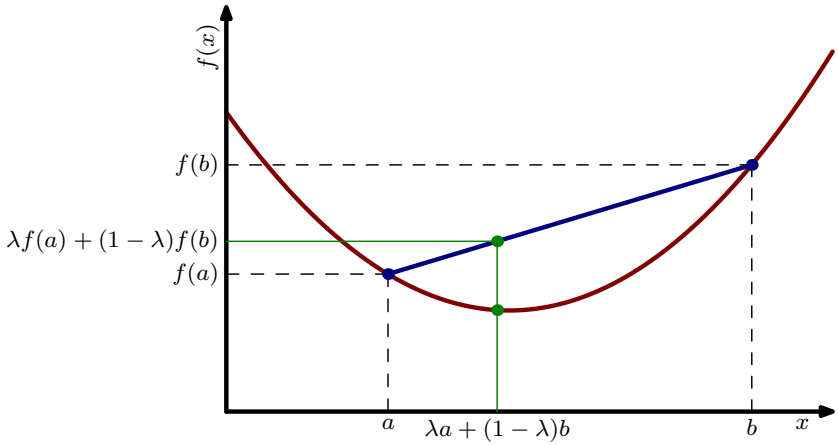
$$0 \geqslant \sum_{\mathbf{z}} q(\mathbf{z}) ln \frac{p(\mathbf{z})}{q(\mathbf{z})}$$

$$0 \leqslant - \sum_{\mathbf{z}} q(\mathbf{z}) ln \frac{p(\mathbf{z})}{q(\mathbf{z})}$$

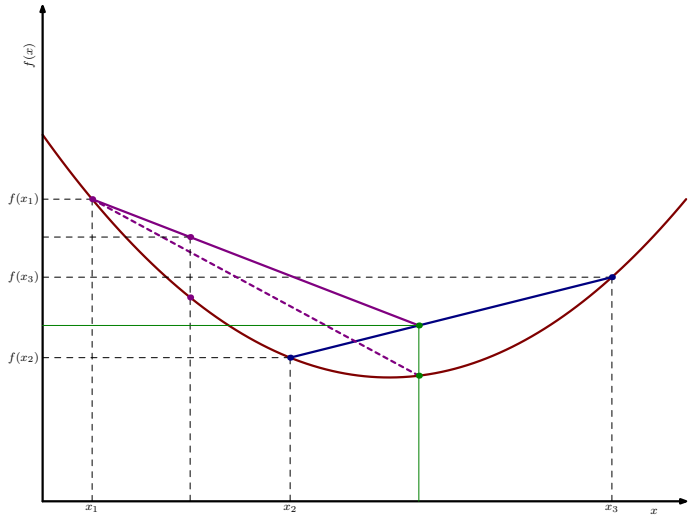
By *Jensen's inequality*, if $f(x)$ is concave:

$$\mathbb{E}[f(x)] \geqslant f(\mathbb{E}[x])$$

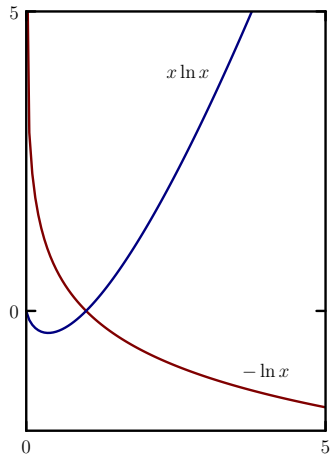
Jensen's inequality



Jensen's inequality



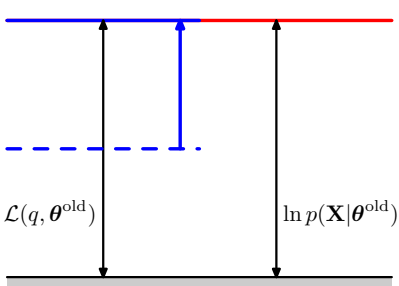
Jensen's inequality



E-Step

During the E-step, we maximise $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $q(\mathbf{z})$, leaving $\boldsymbol{\theta}^{old}$ untouched. Since $\ell(\boldsymbol{\theta}^{old})$ does not depend on $q(\mathbf{z})$, this can only be achieved by setting

$$KL(q||p) = 0$$



in other words

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$$

so that

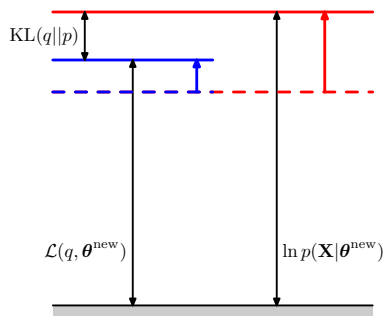
$$\mathcal{L}(q, \boldsymbol{\theta}^{old}) = \ln p(\mathbf{x}|\boldsymbol{\theta}^{old})$$

M-Step

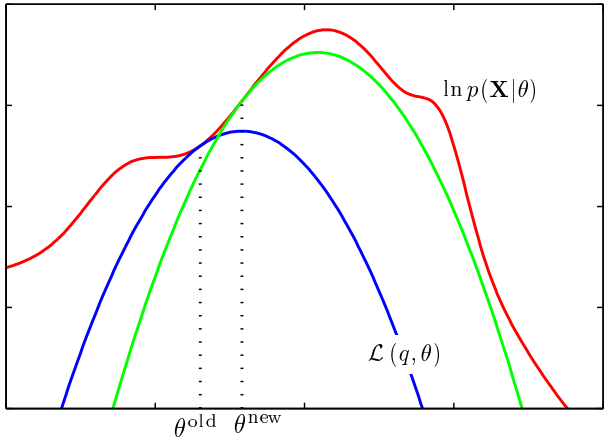
During the M-step, we maximise $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, leaving $q(\mathbf{z})$ untouched. Since $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{old})$,

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{z}, \mathbf{x}|\boldsymbol{\theta}) - \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{old}) \\ &= \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) + const\end{aligned}$$

- Maximising $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ changes $\ln p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$, so that $KL(q||p)$ increases.
- $p(\mathbf{x}|\boldsymbol{\theta})$ therefore increases at least as much as $\mathcal{L}(q, \boldsymbol{\theta})$



EM vs. Gradient Descent



Common extensions of EM

Sometimes, the following extensions of EM are used:

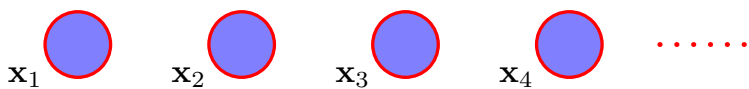
- When the datapoints are independent, the responsibilities $\mathbf{z}_n$ depend on $\mathbf{x}_n$ and $\boldsymbol{\theta}$ only, so that the E and M step can be computed online rather than in batch. This can converge faster than the batch version.
- Sometimes the E-step or M-step (or both) remain intractable. Increasing the likelihood (rather than maximising it) still guarantees increasing the likelihood. This is called Generalised EM (GEM)

Assume independence

Simplest solution:

- Assume that all observations are independent
- Problem: fails to exploit sequential patterns in the data

Example: Rainy days



The only information we can obtain from this model is the frequency of rainy days.

If we do not assume any independence, we can factorise the distribution as:

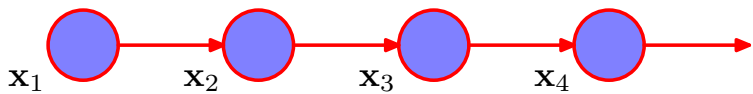
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) \tag{13}$$

However:

- In general the most recent observations are more likely to affect the current observation.
- If we assume that only the single most recent observation affects the current observation, we obtain the **first order Markov chain**

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_{n-1}) \tag{14}$$

First order Markov Chain

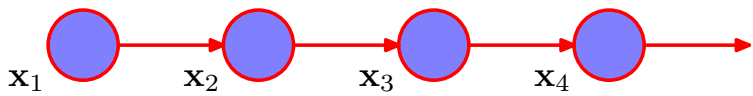


- In most applications, we constrain $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ to be equal over the length of the chain: **homogeneous markov chain**

Example

This was used to model character occurances in text

First order Markov Chain



- In most applications, we constrain $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ to be equal over the length of the chain: **homogeneous markov chain**

Example

This was used to model character occurances in text

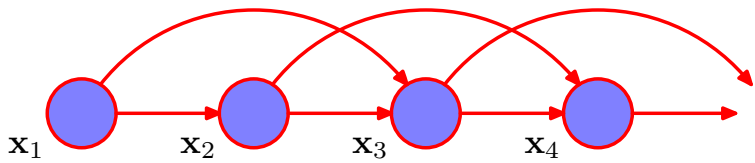
Higher order Markov chains

First order Markov chains are still very restrictive:

- Modelling trends requires more information from the past
- Second order Markov chain:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1) \prod_{n=3} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}) \tag{15}$$

Second order Markov chain



Higher order Markov chains

There is a high penalty for the extra flexibility:

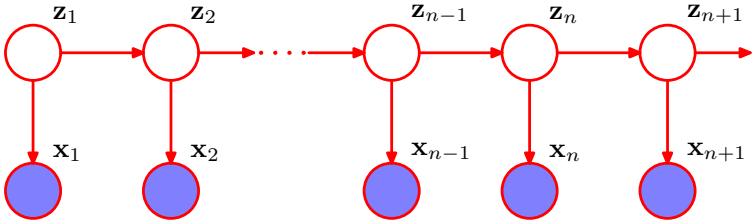
- Imagine K discrete states
  - First order chain: $K \times (K-1)$ independent parameters
  - Second order chain: $K^2 \times (K-1)$ parameters
  - In general, Mth order: $K^{M-1} \times (K-1)$ parameters
- The complexity grows exponentially with M

This becomes impractical very quickly.

How can we keep a longer memory and still restrict the number of parameters?

- Create a latent variable model: the probability of observations depends on the latent variable (see Mixtures of Gaussians)
- The latent variables are part of a latent Markov chain
- D-separation shows that all observations are now dependent

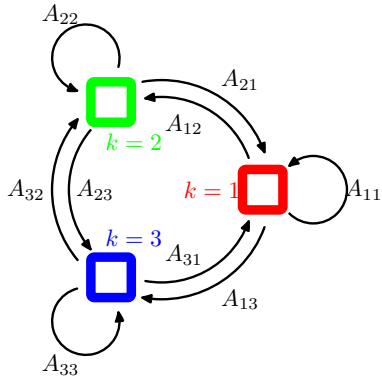
Hidden Markov Model



HMM as an FSA

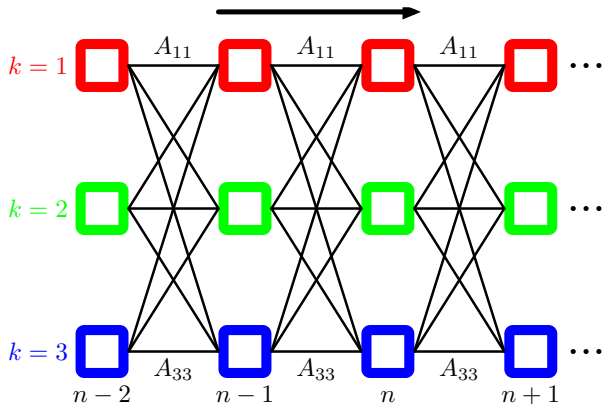
Hidden Markov Models are often viewed as Finite State Machines

- The arrows indicate the *transition probabilities*, typically denoted by the matrix **A**
- *Emission probabilities* are associated with each state
- Not depicted here, is the probability of starting a sequence in a given state k, typically denoted by π_k



Lattices

We can then "unroll" the FSA over time to form a lattice



Inference in the HMM

Inference allows us to answer the questions:

- What is the probability of each state, for each observation

Example

In speech recognition, what is the probability that a certain phoneme corresponds to a certain audio observation?

- What is the probability of seeing an observation sequence, according to the model?

Example

In elderly care, it is useful to monitor how well patients perform activities at home. It is very hard to have a model of the problems they may have, but it is possible to detect when their activities become very unlikely according to the model.

Inference in the HMM

Inference allows us to answer the questions:

- What is the probability of each state, for each observation

Example

In speech recognition, what is the probability that a certain phoneme corresponds to a certain audio observation?

- What is the probability of seeing an observation sequence, according to the model?

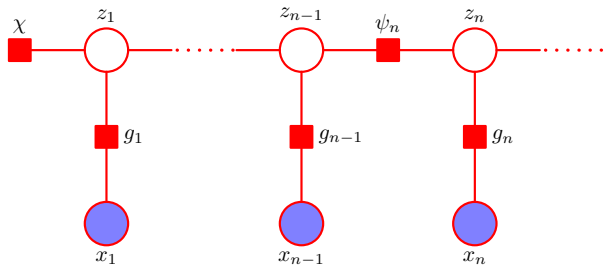
Example

In elderly care, it is useful to monitor how well patients perform activities at home. It is very hard to have a model of the problems they may have, but it is possible to detect when their activities become very unlikely according to the model.

The Forward-Backward Algorithm

HMM have been around for much longer than graphical models

- Specific versions of the sum-product and max-product were developed for the HMM
- The sum-product rule for HMM is known as the Forward-Backward or Baum-Welch algorithm
- It consists of a single forward and a single backward pass through the chain



The Forward pass

The message leaving the first node is

$$\begin{aligned}\mu_{z_1 \to \psi_1} &= p(x_1|z_1)\, p(z_1) \\ &= p(x_1, z_1)\end{aligned}$$

The message leaving the second node is

$$\begin{aligned}\mu_{z_2 \to \psi_3} &= p(x_2|z_2) \sum_{\mathbf{z}_1} p(x_1, z_1)\, p(z_2|z_1) \\ &= p(x_1, x_2, z_2)\end{aligned}$$

In general:

$$\begin{aligned}\mu_{\psi_{n-1} \to z_n} &= p(x_1, \ldots, x_n, z_n) \\ &= \alpha(z_n)\end{aligned}$$

The Backward pass

The message entering the penultimate node is

$$\begin{aligned}\mu_{\psi_{N-1}\to z_{N-1}} &= \sum_{\mathbf{z}_N} p(z_N|z_{N-1})\, p(x_N|z_N) \\ &= p(x_N|z_{N-1})\end{aligned}$$

The message entering the antepenultimate node is

$$\begin{aligned}\mu_{\psi_{N-2}\to z_{N-2}} &= \sum_{\mathbf{z}_{N-1}} p(z_{N-1}|z_{N-2})\, p(x_{N-1}|z_{N-1})\, p(x_N|z_{N-1}) \\ &= p(x_{N-1}, x_N|z_{N-2})\end{aligned}$$

In general, the "backward" message is:

$$\begin{aligned}\mu_{\psi_n\to z_n} &= p(x_{n+1}, \ldots, x_N|z_n) \\ &= \beta(z_n)\end{aligned}$$

The sequence of hidden states often has a real, physical explanation

Example

In speech recognition, the sequence of phonemes for a sequence of sound observations.

It is therefore useful to have the most likely single sequence of latent states, rather than the sequence of most likely latent states

- We obtain this using the max-sum algorithm
- For the HMM, this is called the Viterbi Algorithm

Training of the HMM

We can train the HMM using the EM algorithm, optimising

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{16}$$

We introduce

$$\gamma(z_{nk}) = p(z_{nk}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{17}$$

$$\xi(z_{n-1\,j}, z_{n,k}) = p(z_{n-1\,j}, z_{nk}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{18}$$

so that the expectation of the complete log-likelihood becomes

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{k} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n|z_k)$$

The *responsibilities* $\gamma(z_{nk})$ can be computed from the forward and backward probabilities:

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{19}$$

$$= \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \tag{20}$$

and similarly for the pairwise responsibilities:

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \tag{21}$$

$$= \frac{\alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \tag{22}$$

The M step

- Taking the first derivative of $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to the parameters and setting it to zero gives us the update rules:

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{23}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=2}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})} \tag{24}$$

- optimising the *emission* probabilities depends on the specific form of these, and is identical to a mixture model.

Example: Gaussian observations

If the observations have a Gaussian distribution, the maximum likelihood parameters are given by:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{25}$$

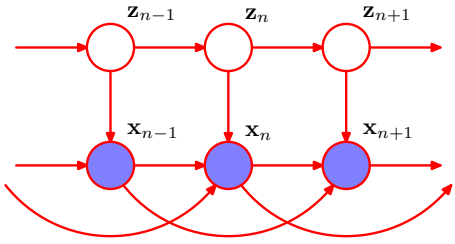
$$\boldsymbol{\Sigma}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{26}$$

(See Gaussian Mixture Models)

Extensions

The basic HMM framework can be extended in many ways

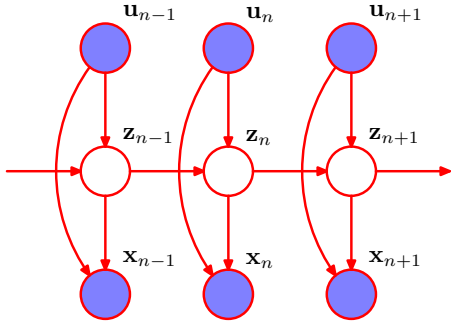
- Autoregressive HMM
- Input-output HMM
- Factorial HMM
- Explicit time model (Hidden semi-Markov model)



Extensions

The basic HMM framework can be extended in many ways

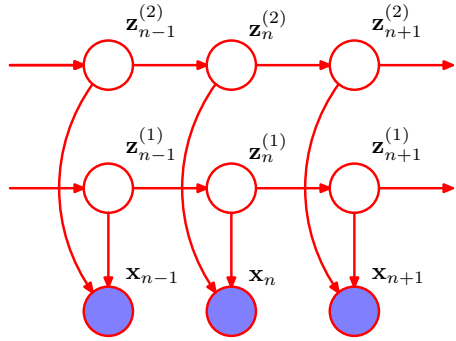
- Autoregressive HMM
- Input-output HMM
- Factorial HMM
- Explicit time model (Hidden semi-Markov model)



Extensions

The basic HMM framework can be extended in many ways

- Autoregressive HMM
- Input-output HMM
- Factorial HMM
- Explicit time model (Hidden semi-Markov model)

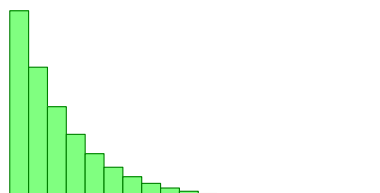


Extensions

The basic HMM framework can be extended in many ways

- Autoregressive HMM
- Input-output HMM
- Factorial HMM
- Explicit time model (Hidden semi-Markov model)

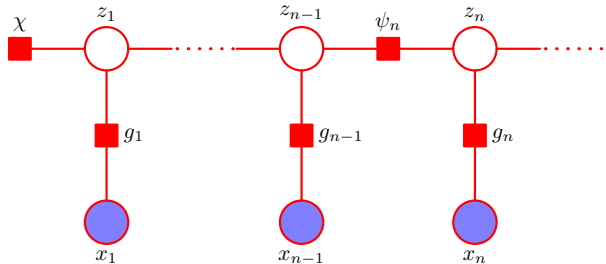
$$p(z_n, \ldots, z_{n+t} = z) = A_{zz}^t(1 - A_{zz})$$



Linear Dynamical System

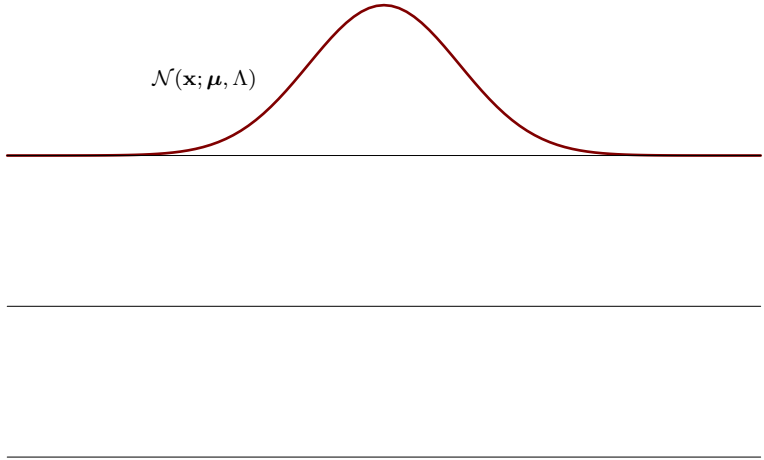
In the HMM, we assume that the latent variables are discrete.

- The observations (emission probabilities) can have any form (Bernoulli, Gaussian, Mixture of Gaussians, *etc.*)
- This is not a limitation of the graphical model per se.

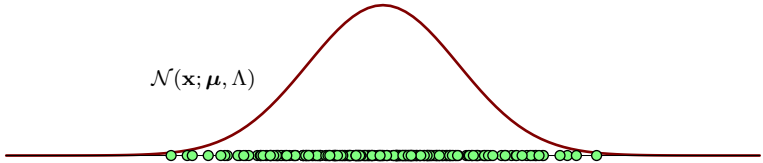


- One well-known model that allows for continuous latent variables is the Linear Dynamical System

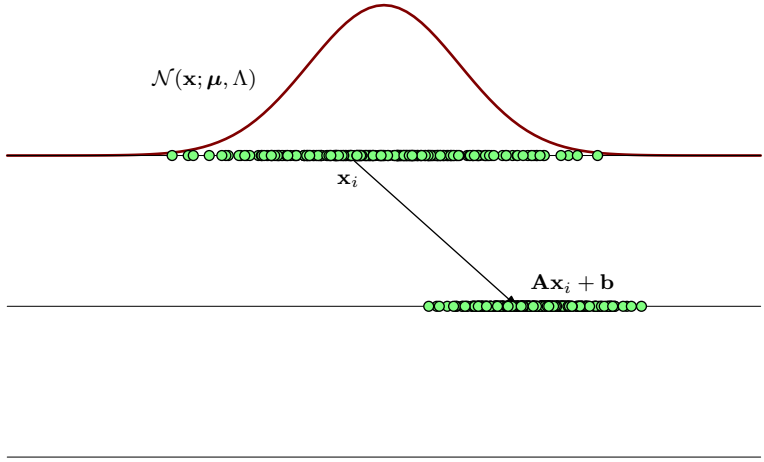
Linear-Gaussian system



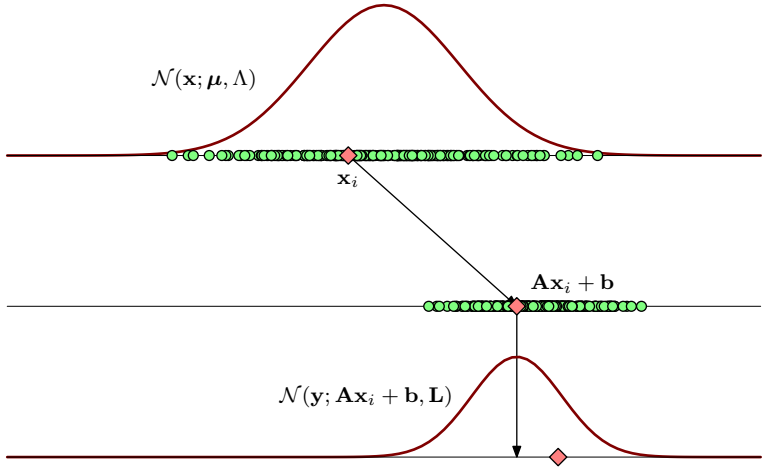
Linear-Gaussian system



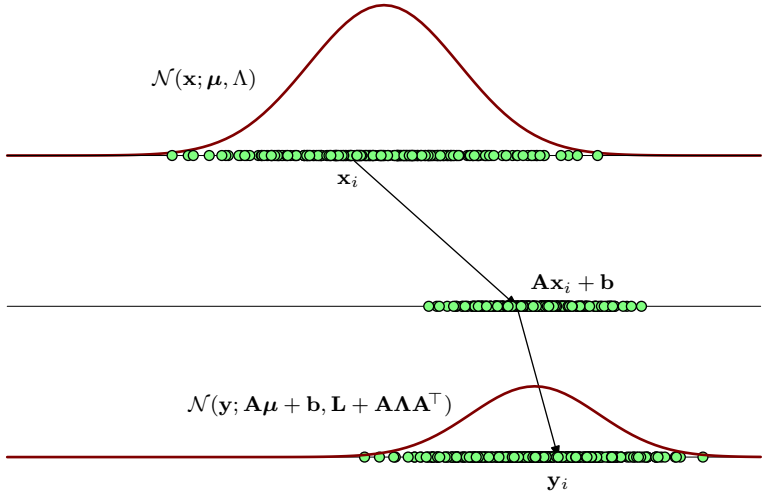
Linear-Gaussian system



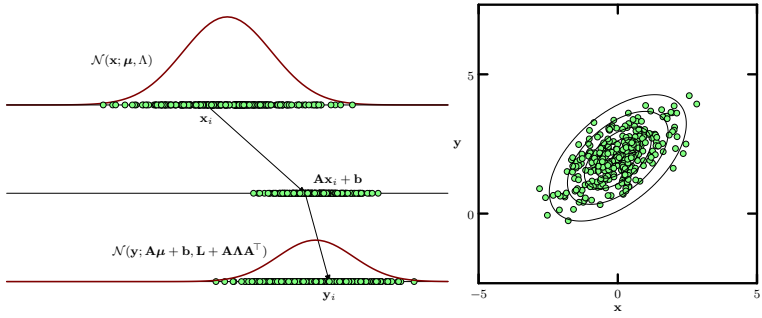
Linear-Gaussian system



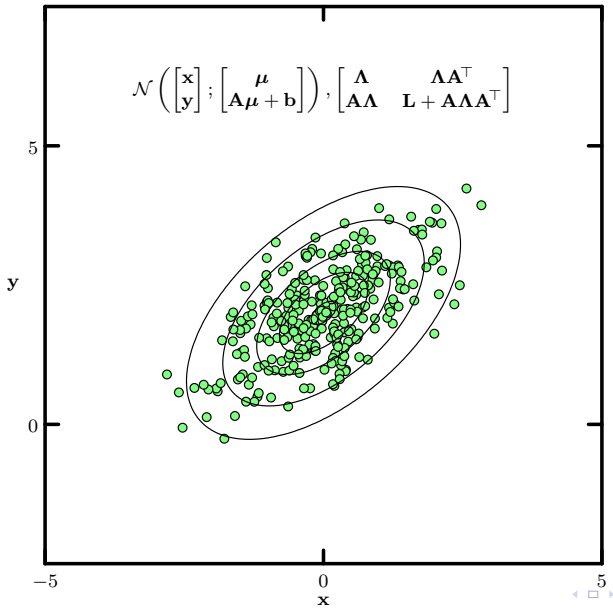
Linear-Gaussian system



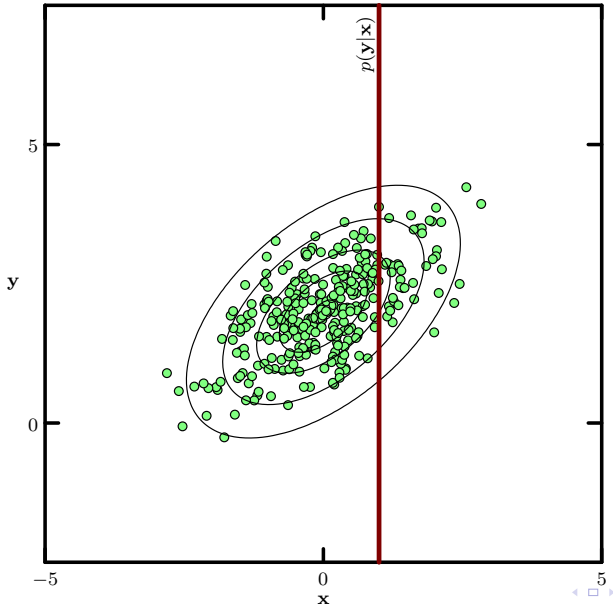
Linear-Gaussian system



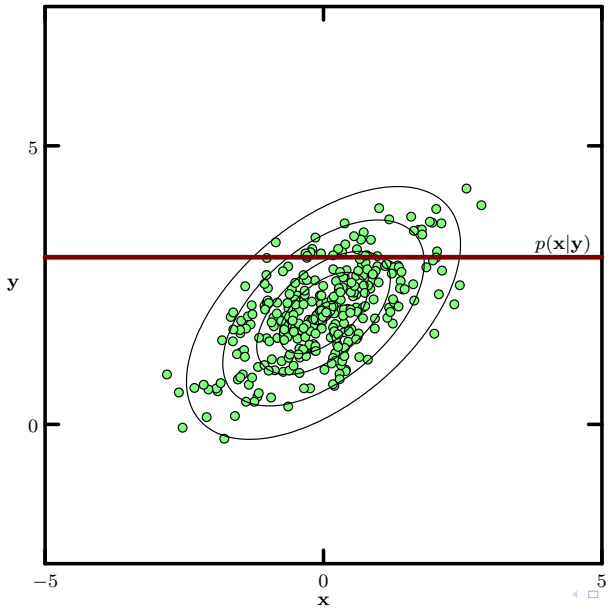
Linear-Gaussian system



Linear-Gaussian system



Linear-Gaussian system



The LDS assumes Gaussian distributions for all variables, where

- The mean of the next latent variable is a linear function of the previous latent variable:

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1}, \mathbf{\Gamma}) \tag{27}$$

- The current observation has a Gaussian distribution centred around the current latent variable

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}) \tag{28}$$

- Assuming Gaussian distributions ensures that the complexity of the messages does not increase along the chain
- The prior $p(\mathbf{z}_0)$ can be a mixture of K Gaussians, however

The LDS assumes Gaussian distributions for all variables, where

- The mean of the next latent variable is a linear function of the previous latent variable:

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1} + \mathbf{B}\mathbf{u}, \mathbf{\Gamma}) \tag{27}$$

- The current observation has a Gaussian distribution centred around the current latent variable

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}) \tag{28}$$

- Assuming Gaussian distributions ensures that the complexity of the messages does not increase along the chain
- The prior $p(\mathbf{z}_0)$ can be a mixture of K Gaussians, however

The LDS assumes Gaussian distributions for all variables, where

- The mean of the next latent variable is a linear function of the previous latent variable:

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1} + \mathbf{B}\mathbf{u}, \mathbf{\Gamma}) \tag{27}$$

- The current observation has a Gaussian distribution centred around the current latent variable

$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}) \tag{28}$$

- Assuming Gaussian distributions ensures that the complexity of the messages does not increase along the chain
- The prior $p(\mathbf{z}_0)$ can be a mixture of K Gaussians, however

The LDS assumes Gaussian distributions for all variables, where

- The mean of the next latent variable is a linear function of the previous latent variable:

$$p(\mathbf{z}_n|\mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n|\mathbf{A}\mathbf{z}_{n-1} + \mathbf{B}\mathbf{u}, \mathbf{\Gamma}) \tag{27}$$

- The current observation has a Gaussian distribution centred around the current latent variable

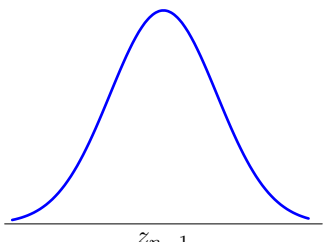
$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}) \tag{28}$$

- Assuming Gaussian distributions ensures that the complexity of the messages does not increase along the chain
- The prior $p(\mathbf{z}_0)$ can be a mixture of K Gaussians, however

Again, this is a model that was developed long before graphical models

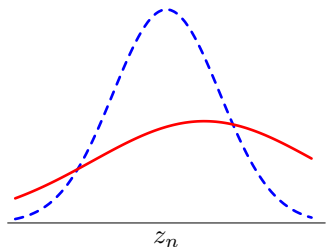
- If only the forward pass is done, it is called the *Kalman Filter*
- It was developed for optimal tracking of rockets and satelites
- If the backwards pass is performed as well, it is called the *Kalman Smoother*. The corresponding equations are called the *Rauch-Tung-Striebel* (RTS) equations.

Kalman Filter Updates



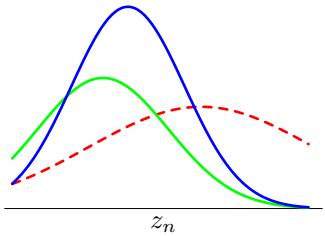
- Distribution over $\mathbf{z}_{n-1}$
- Prediction over $\mathbf{z}_n$
- Probability of the observation $\mathbf{x}_n$ given $\mathbf{z}_n$, and updated distribution over $\mathbf{z}_n$

Kalman Filter Updates



- Distribution over $\mathbf{z}_{n-1}$
- Prediction over $\mathbf{z}_n$
- Probability of the observation $\mathbf{x}_n$ given $\mathbf{z}_n$, and updated distribution over $\mathbf{z}_n$

Kalman Filter Updates



- Distribution over $\mathbf{z}_{n-1}$
- Prediction over $\mathbf{z}_n$
- Probability of the observation $\mathbf{x}_n$ given $\mathbf{z}_n$, and updated distribution over $\mathbf{z}_n$

Learning in the LDS

The Kalman filter is often used for tracking

- The matrices $\mathbf{A}$ and $\mathbf{C}$ are then known *a priori* — e.g.

$$\mathbf{z} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{29}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{30}$$

- Covariance matrices $\mathbf{\Gamma}$ and $\mathbf{\Sigma}$ must be estimated

. . . Sometimes process must be learnt as well

Learning in the LDS

As always, we can use EM to learn the model parameters

- The latent variables are continuous, posterior PDF is a Gaussian
- The expectation of the complete log-likelihood can be optimised in closed form
  - With respect to the distribution parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma})$
  - With respect to the deterministic model parameters $(\mathbf{A}, \mathbf{C})$

Wrap-up

Today, we have seen:

- Learning when we can do inference (Bishop, p. 439–441)
- Examples of the EM algorithm (Bishop, p. 423–439)
- A formal analysis of EM (Bishop, p. 450–453)
- Compared k-means with mixtures of Gaussians
- Markov Models (Bishop, p. 605-610)
- HMM (Bishop, p. 610-630)
- Introduction to the LDS (Bishop, p. 635-637)

Lab:

- The EM algorithm for GMM