Lecture 6

Expectation Maximisation

University of Amsterdam

Introduction

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

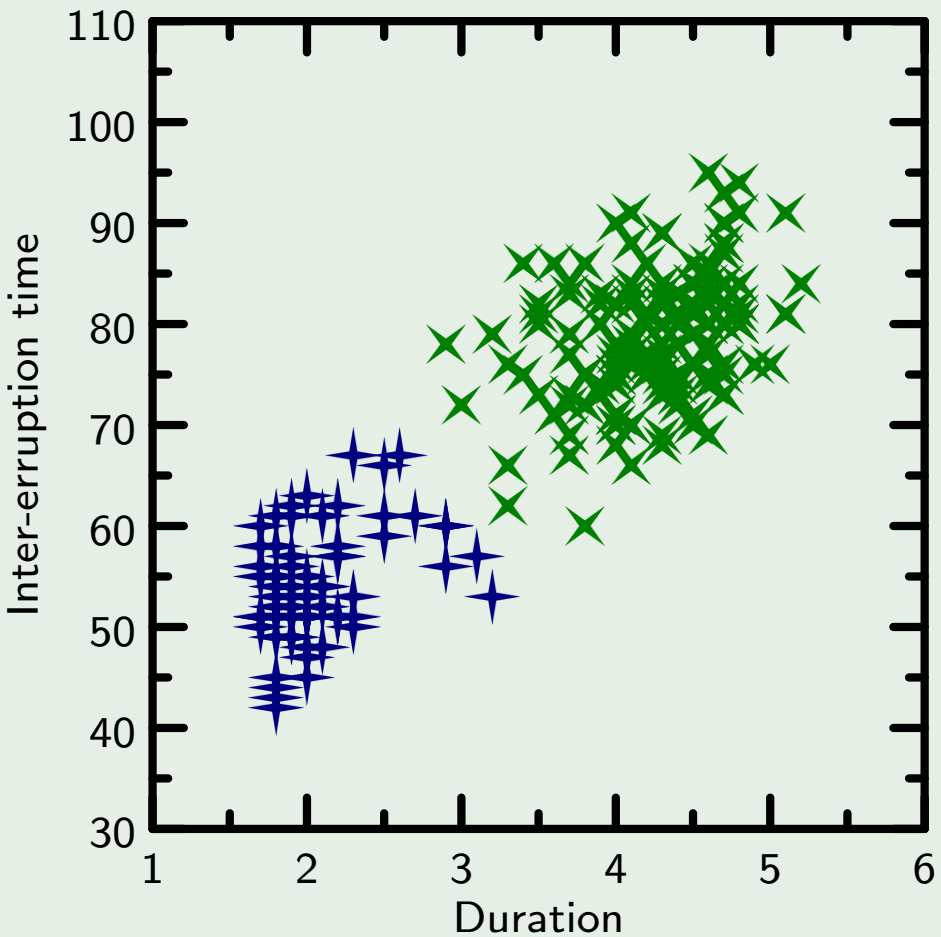
- MAP treatment is very similar
- (exact) full Bayesian treatment is often not tractable
- We'll see approximations in lecture 13

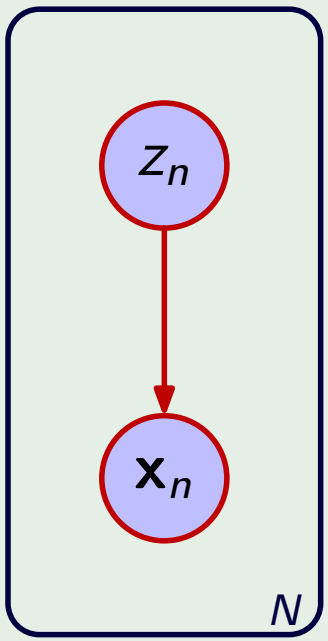
How can we find the maximum of the likelihood?

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

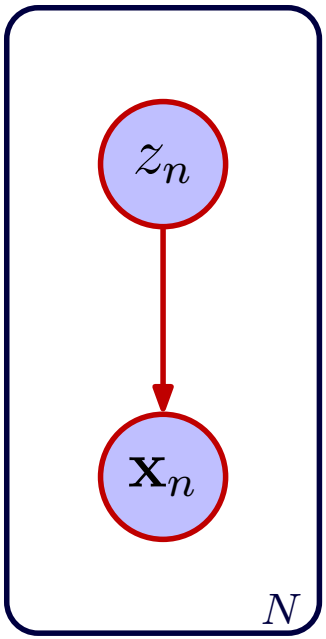
Fully observed model

Example





Fully observed model



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

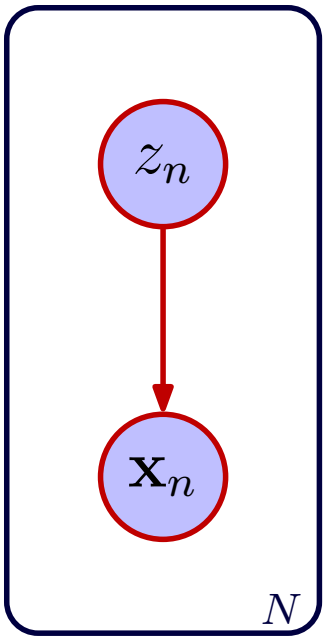
- Parametrisation:

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

(Complete) likelihood:

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

Fully observed model



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

- Parametrisation:

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

(Complete) likelihood:

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

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 $= 0$

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

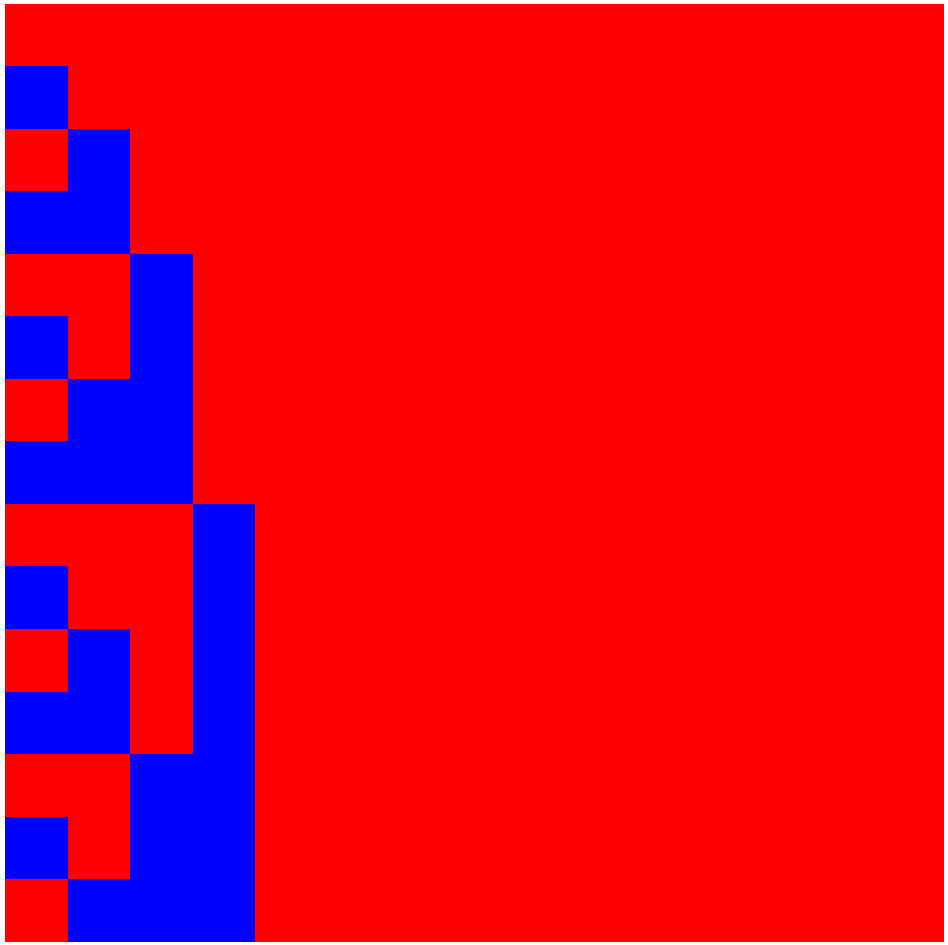
$$\boldsymbol{\mu}_1 = \frac{\sum_{i=1}^{N} z_i\, \mathbf{x}_i}{\sum_{i=1}^{N} z_i} \qquad \boldsymbol{\mu}_2 = \frac{\sum_{i=1}^{N} (1-z_i)\, \mathbf{x}_i}{\sum_{i=1}^{N} z_i}$$

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

Hidden class label

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

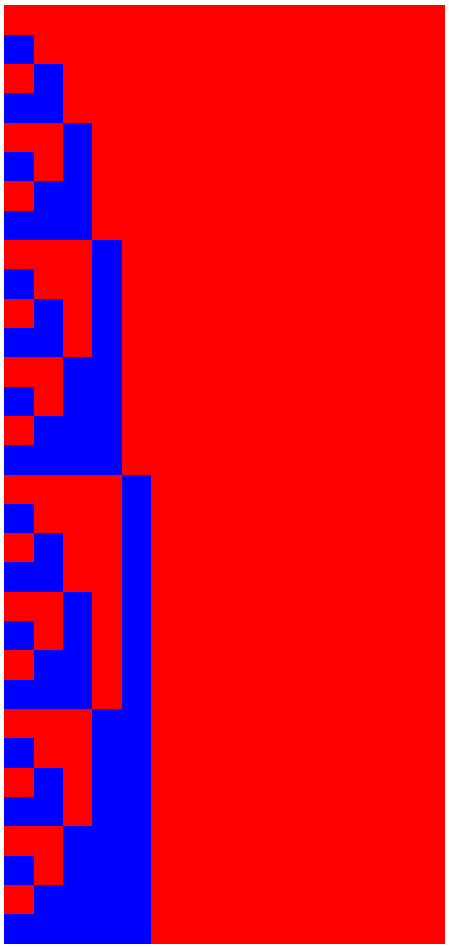
- List all possible assignments, pick best



Hidden class label

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

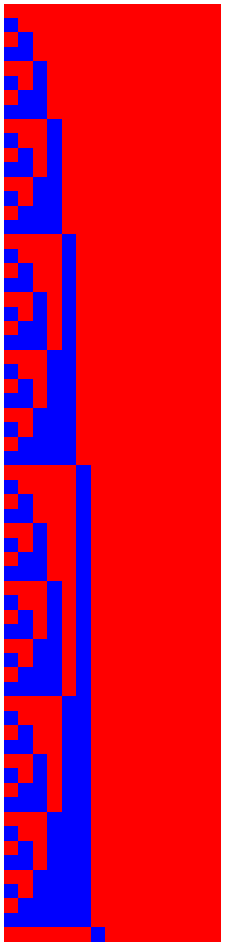
- List all possible assignments, pick best



Hidden class label

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

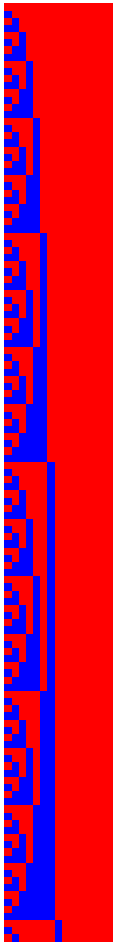
- List all possible assignments, pick best



Hidden class label

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

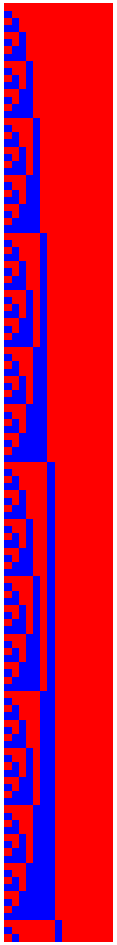
- List all possible assignments, pick best



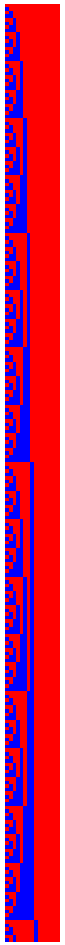
Hidden class label

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

- List all possible assignments, pick best



Hidden class label



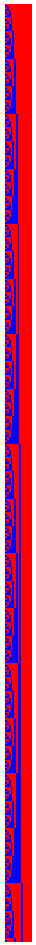
So, what happens when the class label is not observed?

- List all possible assignments, pick best

Hidden class label

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

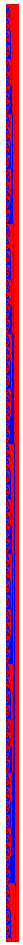
- List all possible assignments, pick best



Hidden class label

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

- List all possible assignments, pick best



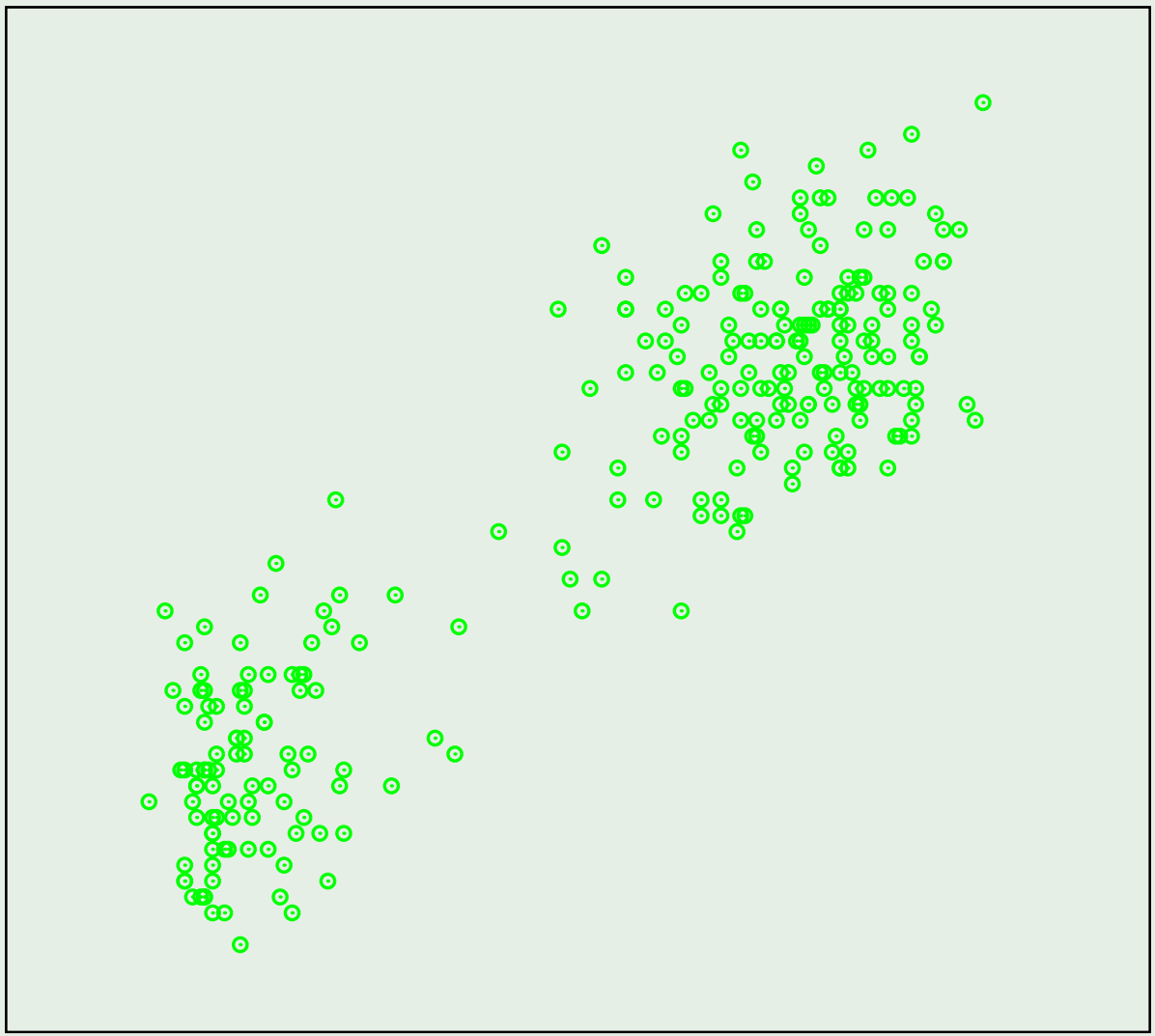
Hidden class label

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

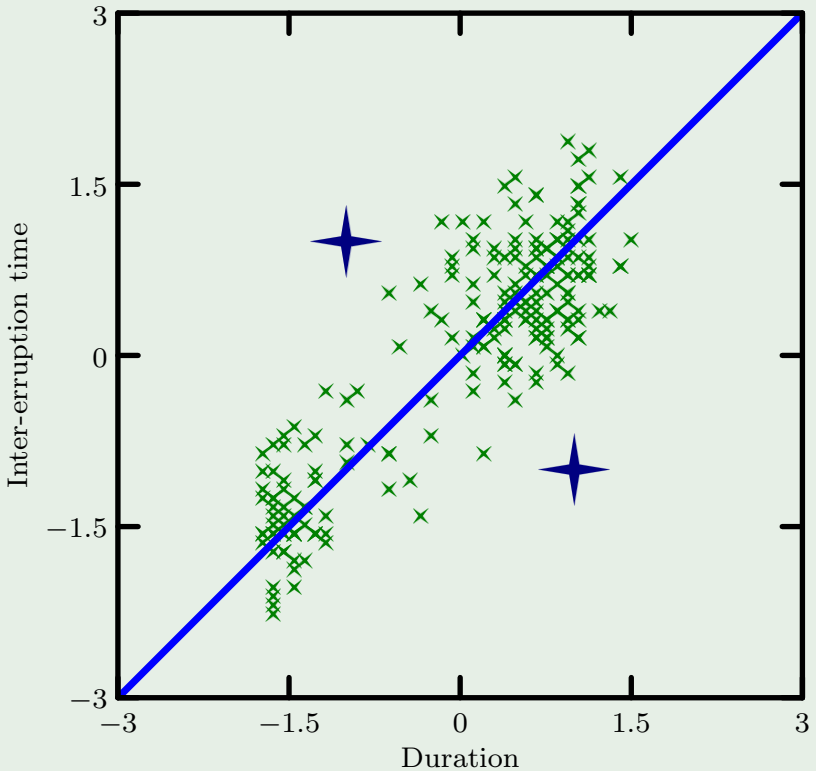
- Where C is the number of clusters
- and N is the number of datapoints.

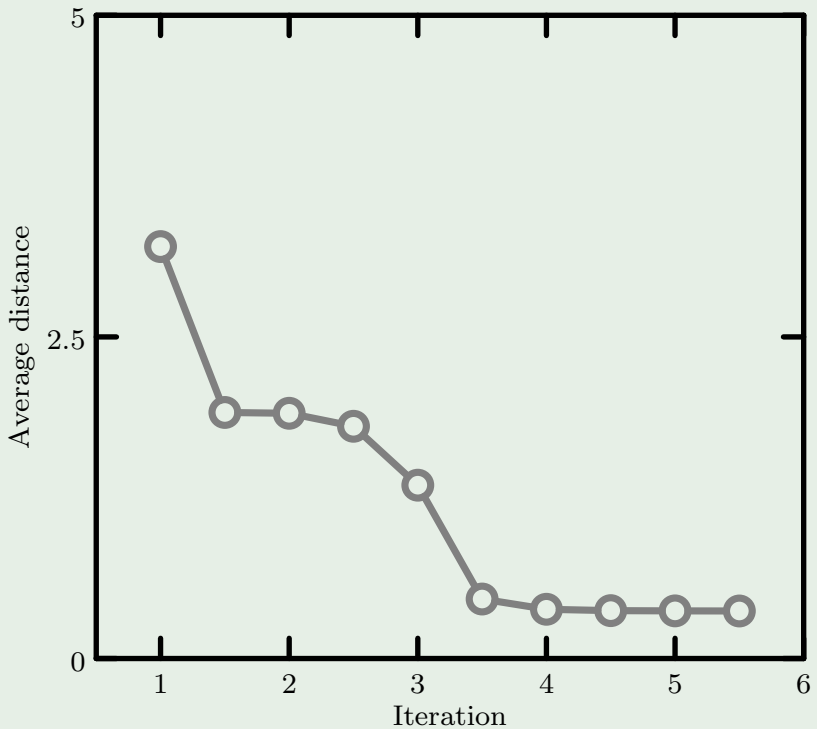
How do we find the optimal clustering without exhaustive search? Solve the clustering iteratively:

- 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

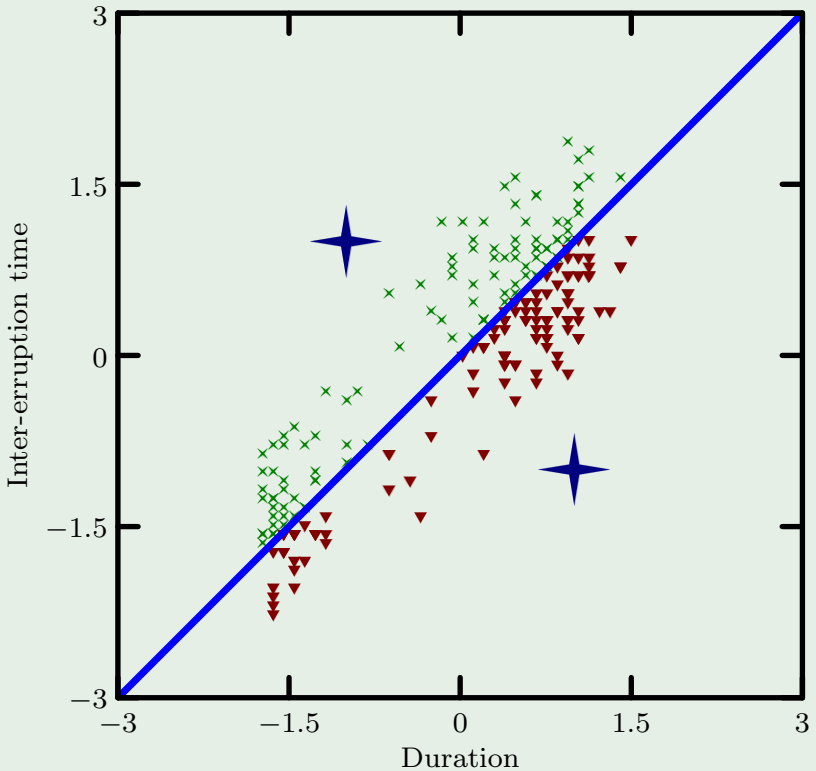
Example: kMeans Clustering

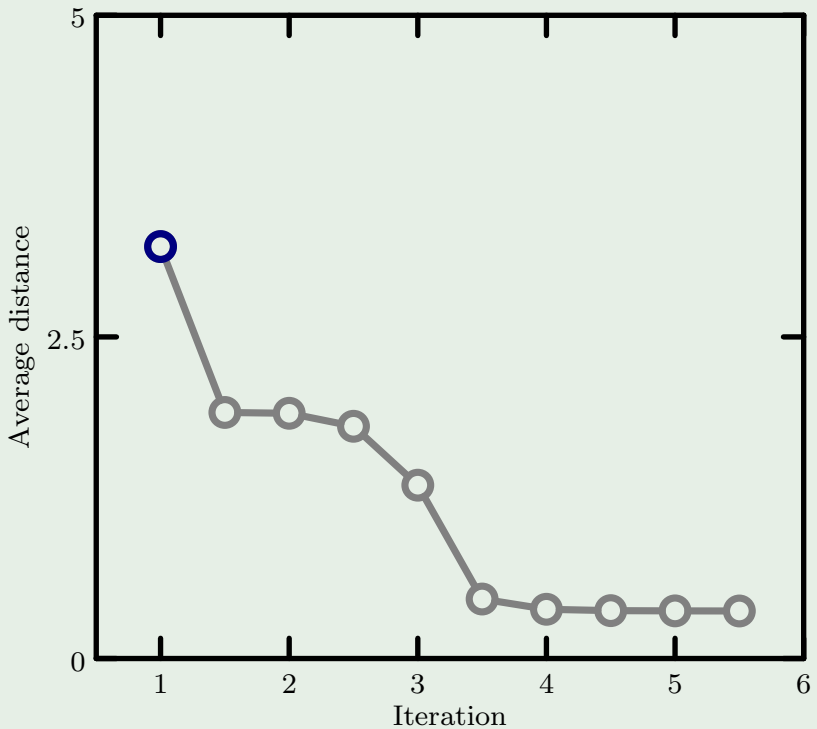




An iterative algorithm

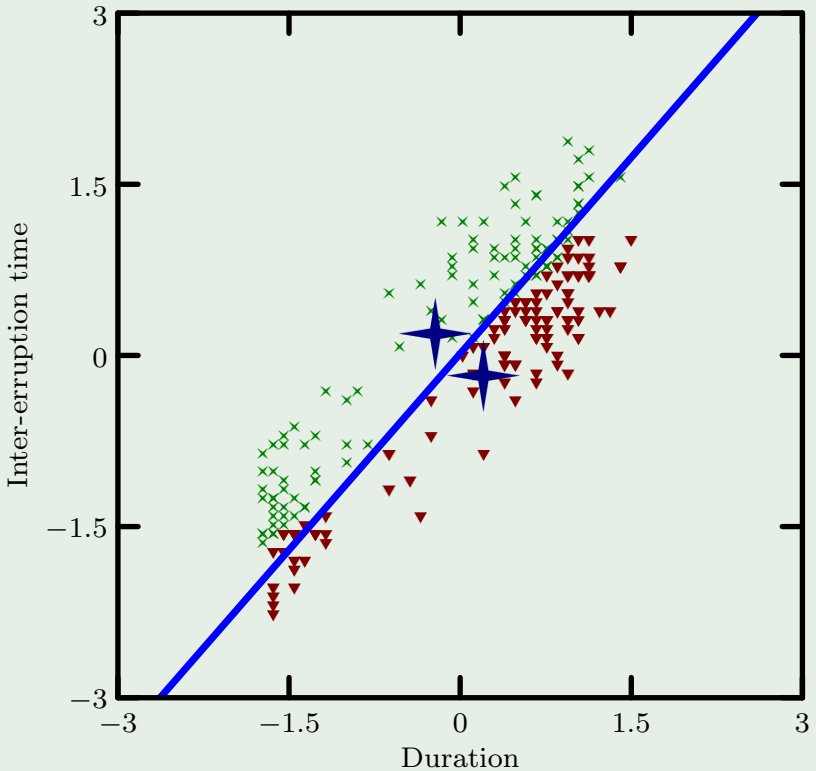
Example: kMeans Clustering

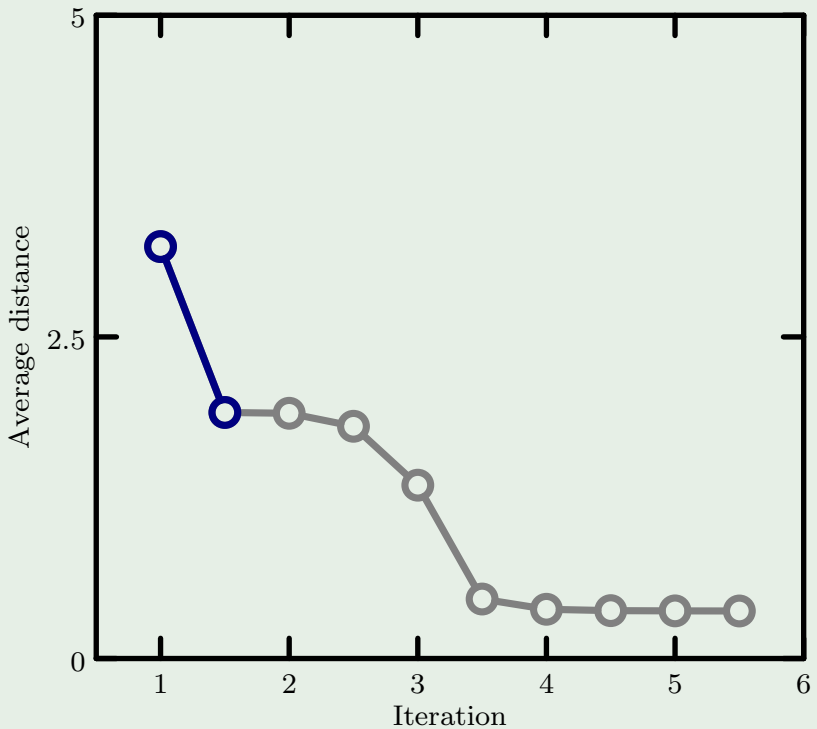




An iterative algorithm

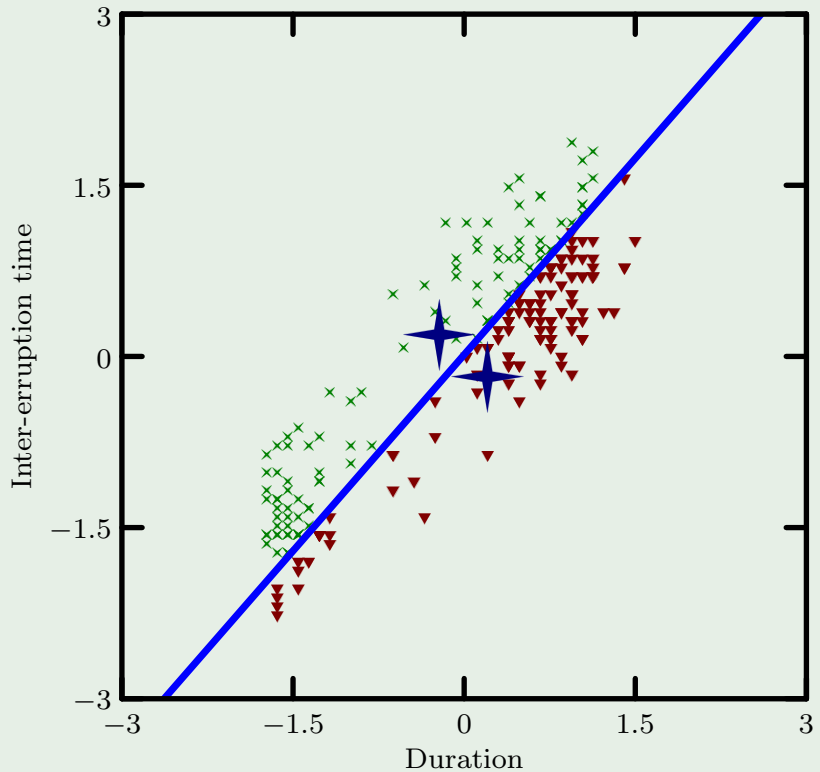
Example: kMeans Clustering

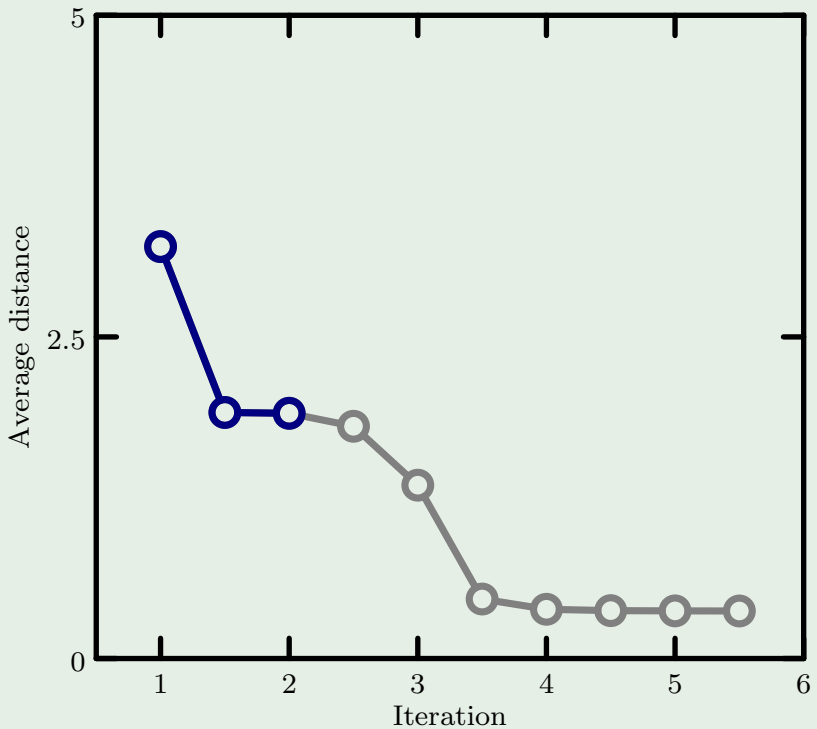




An iterative algorithm

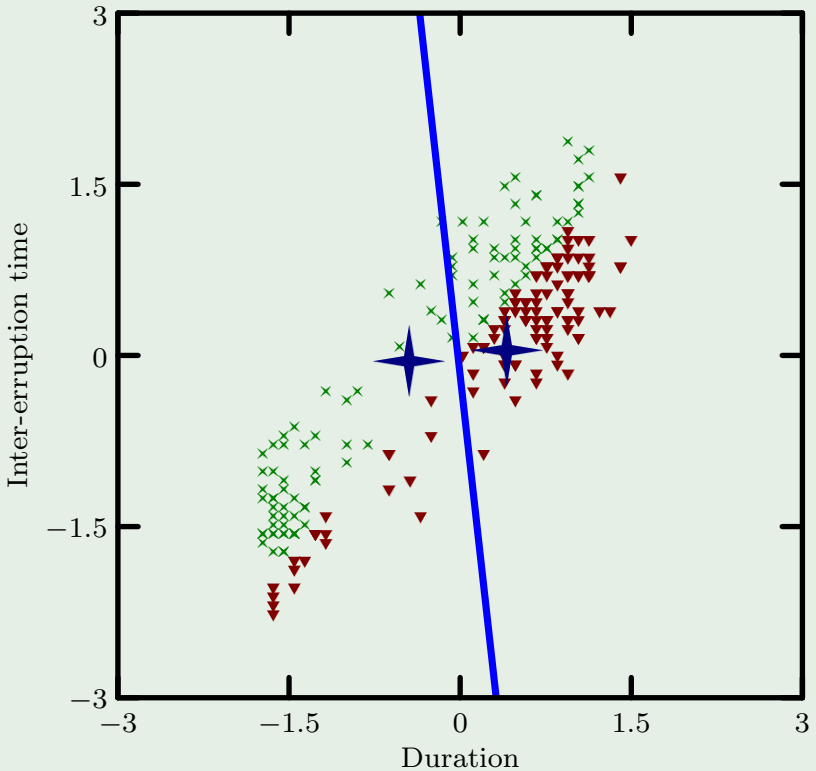
Example: kMeans Clustering

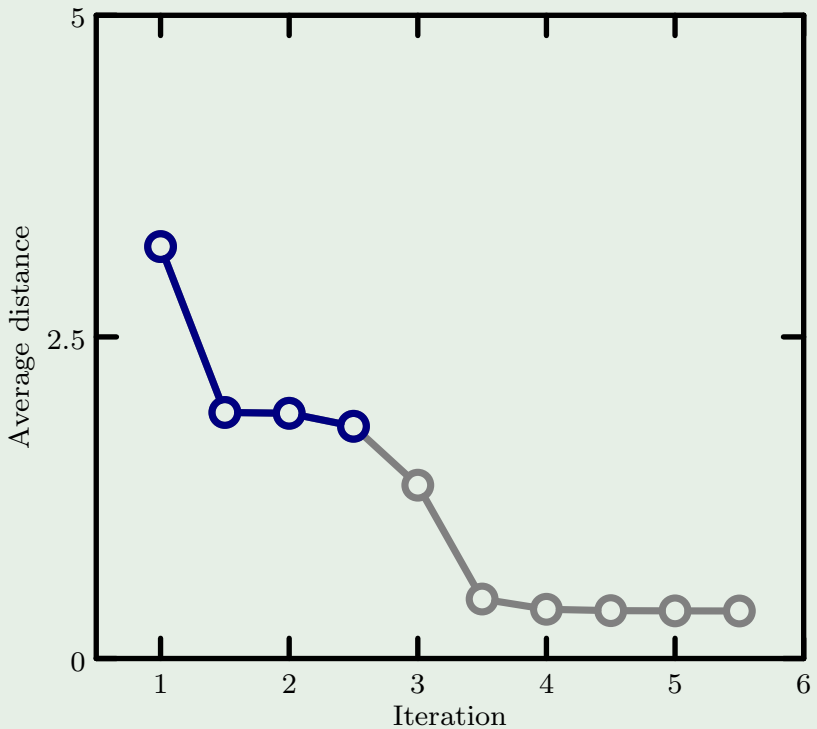




An iterative algorithm

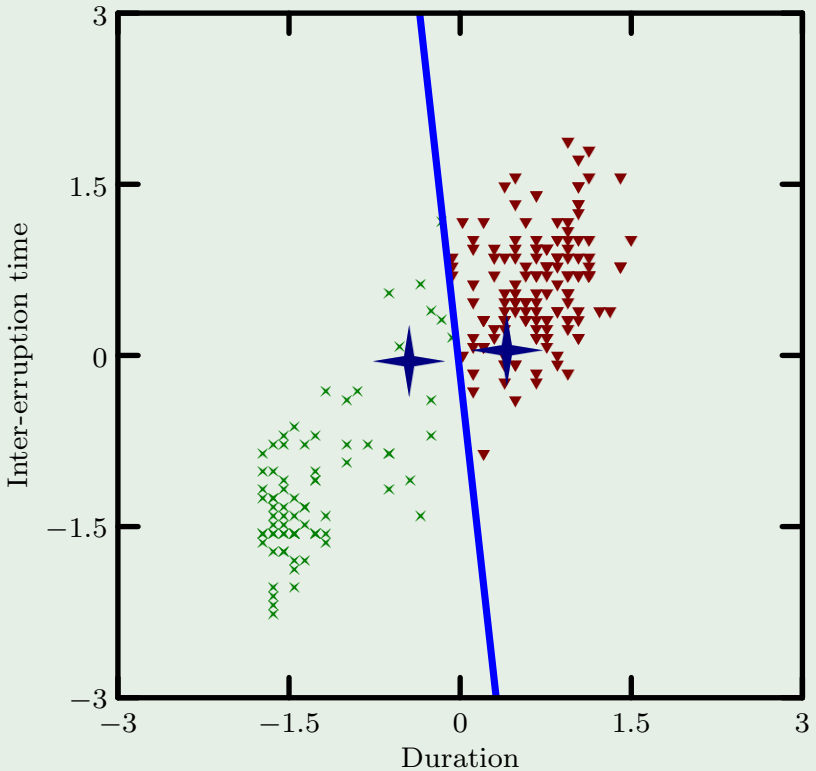
Example: kMeans Clustering

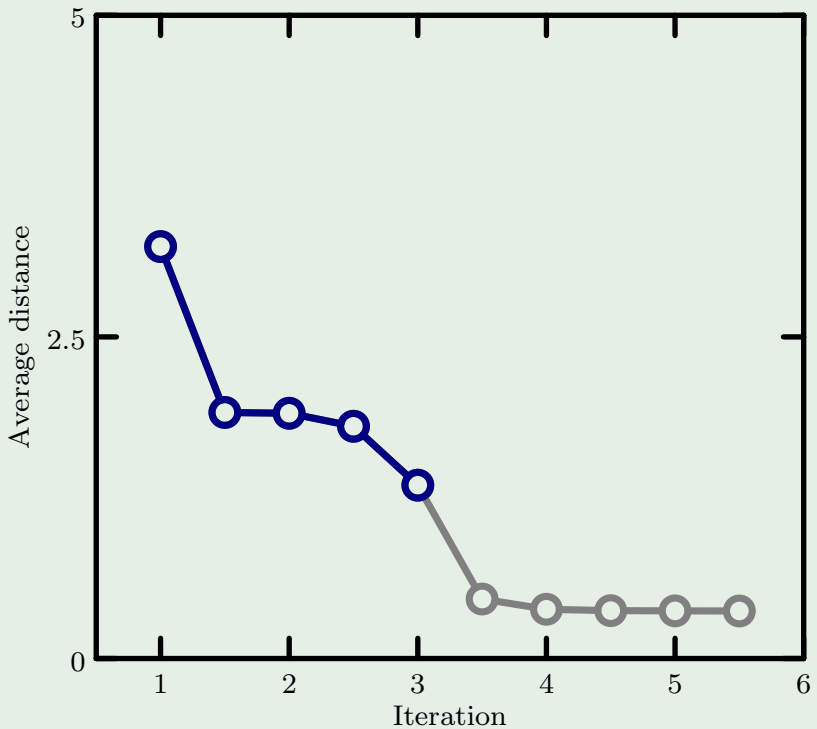




An iterative algorithm

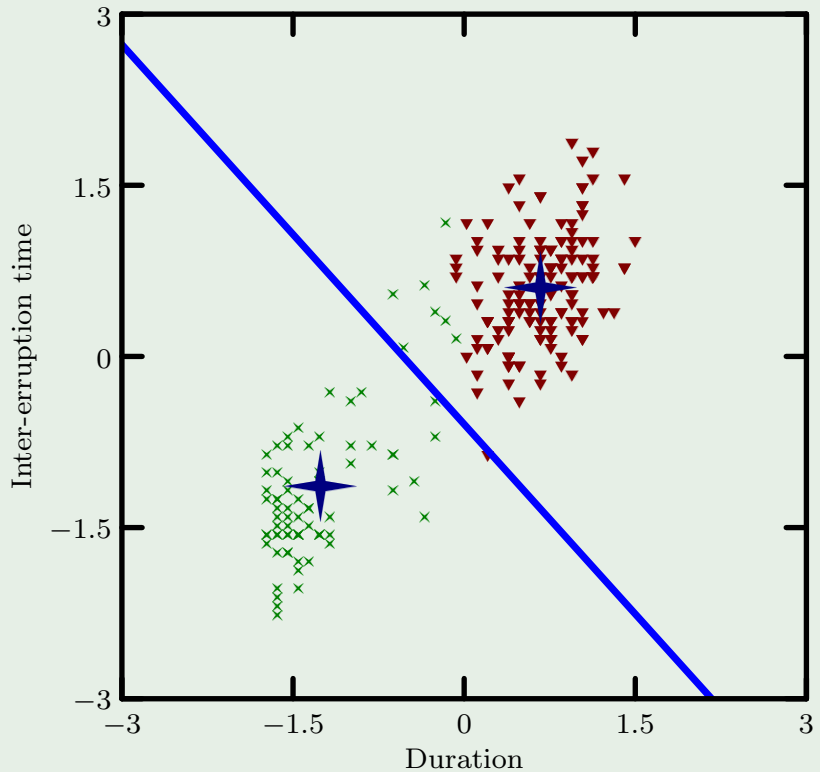
Example: kMeans Clustering

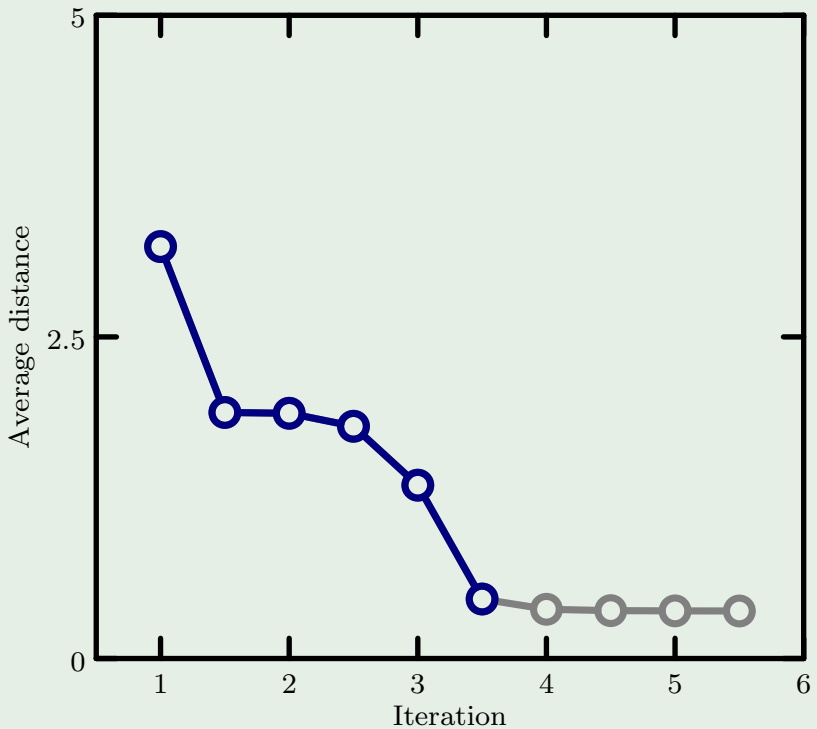




An iterative algorithm

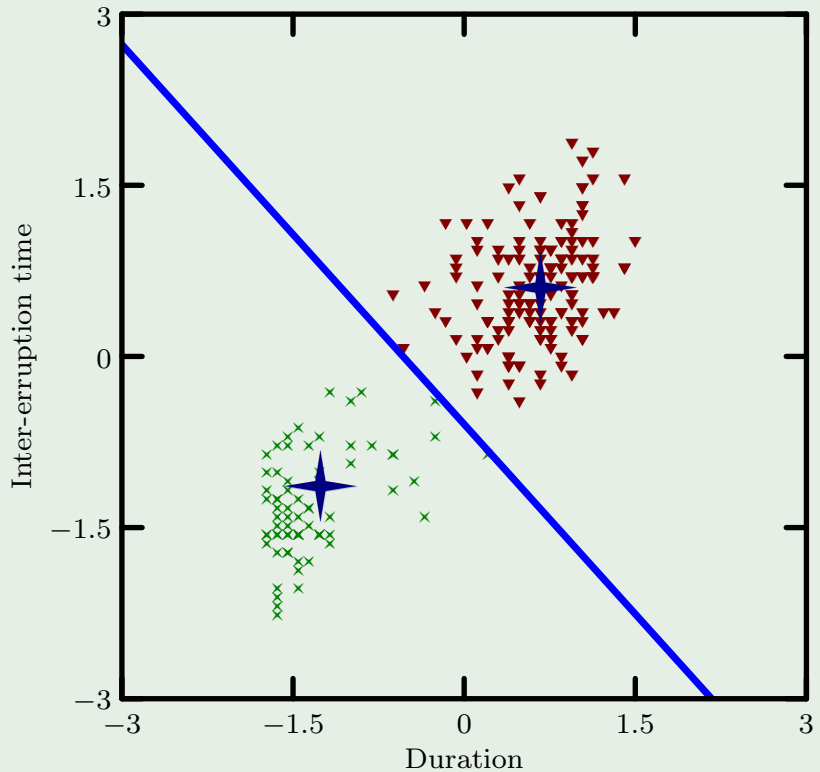
Example: kMeans Clustering

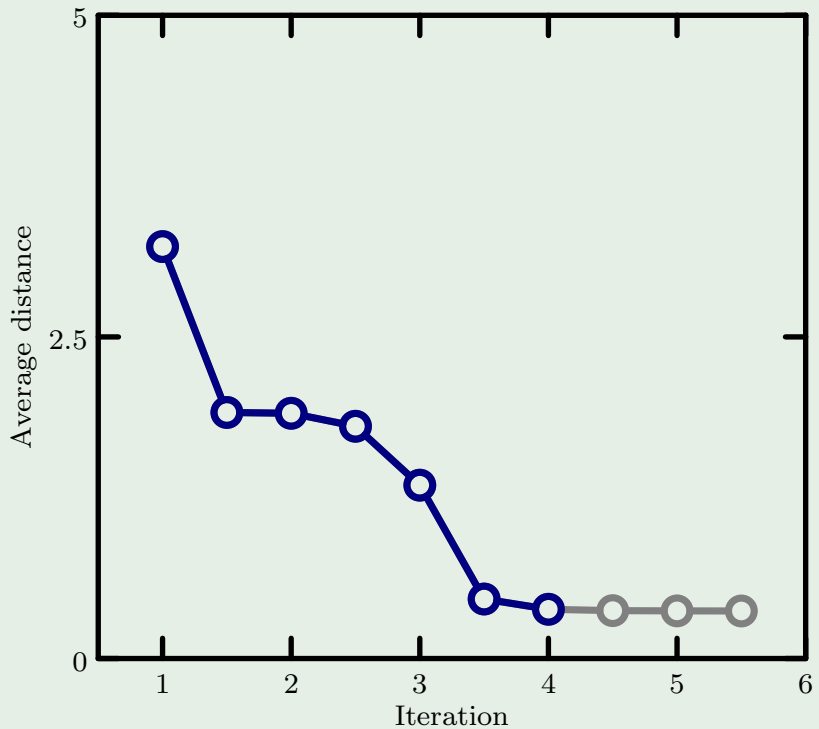




An iterative algorithm

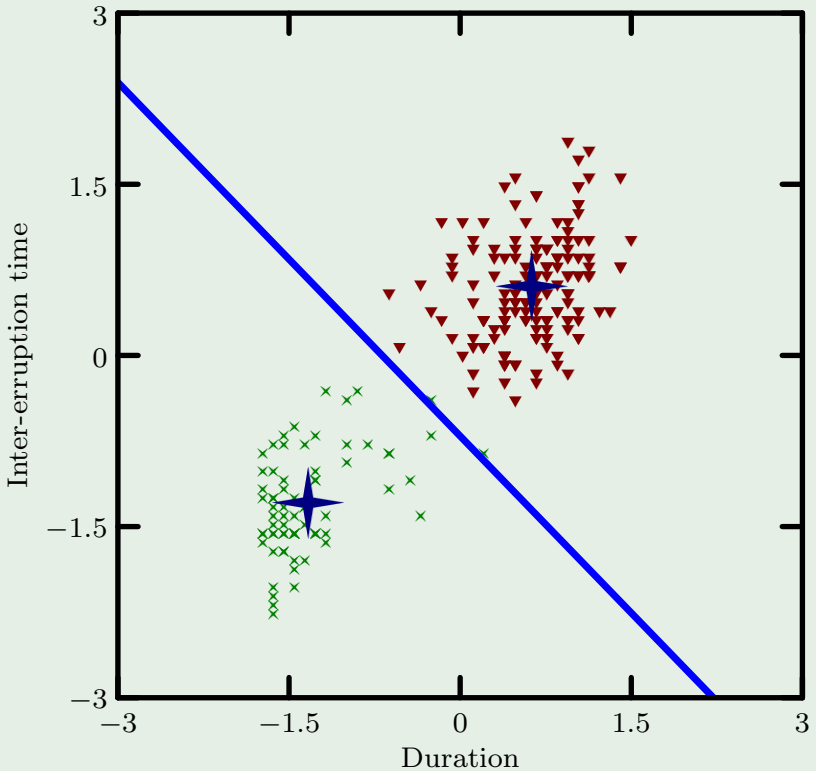
Example: kMeans Clustering

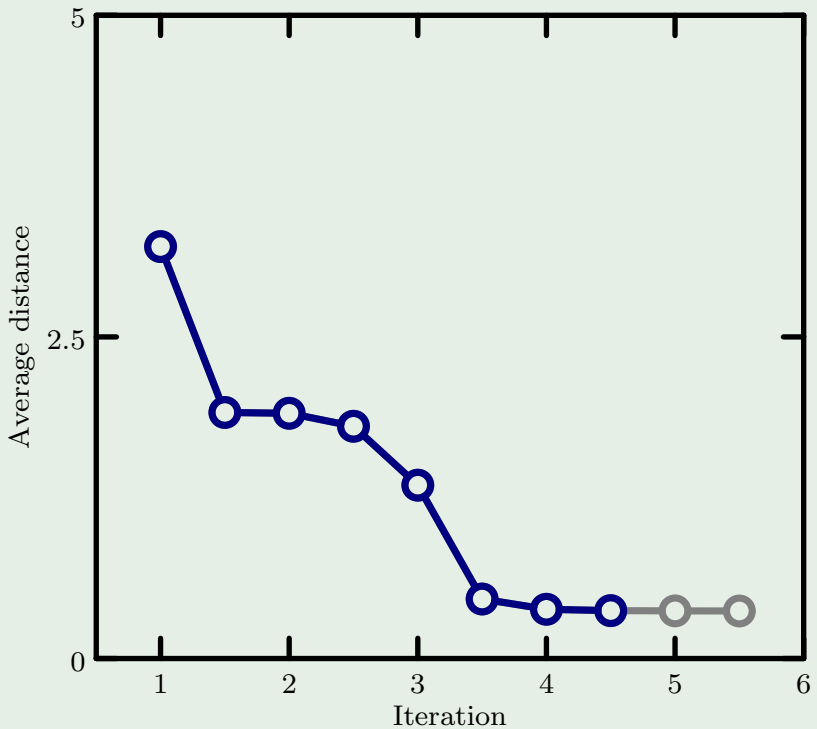




An iterative algorithm

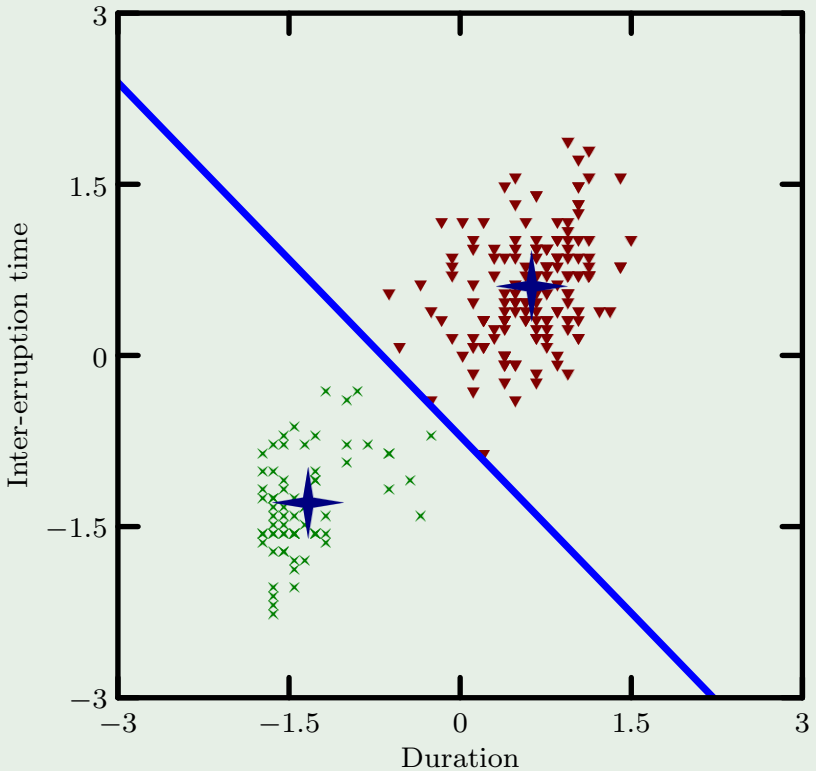
Example: kMeans Clustering

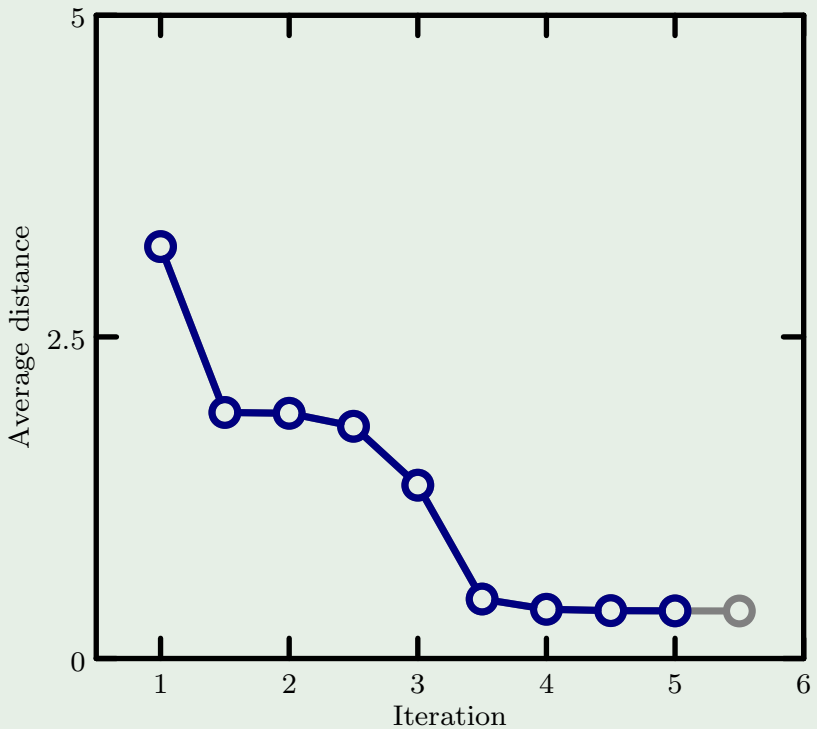




An iterative algorithm

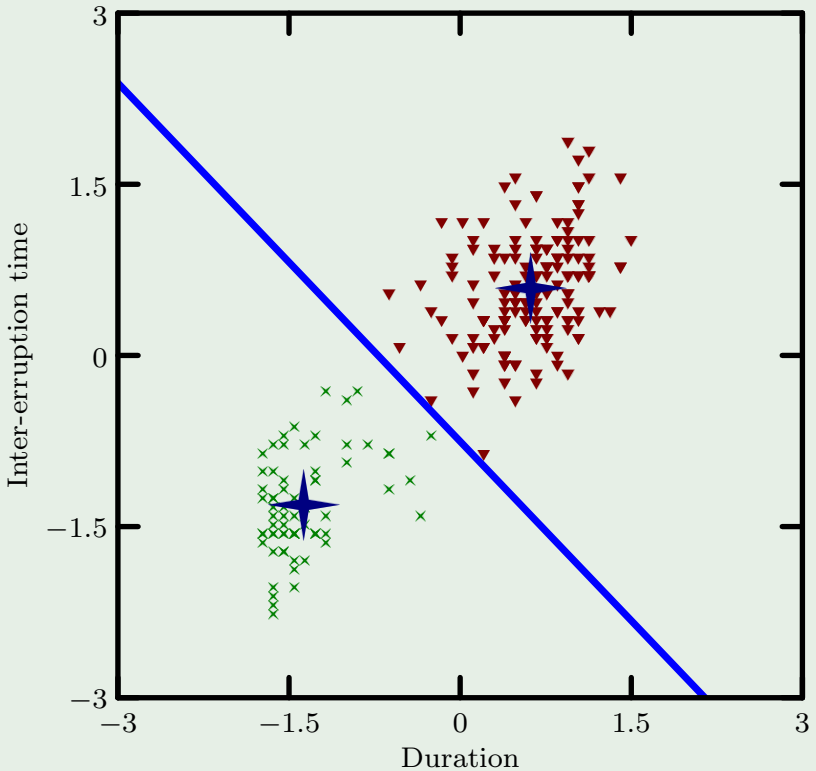
Example: kMeans Clustering

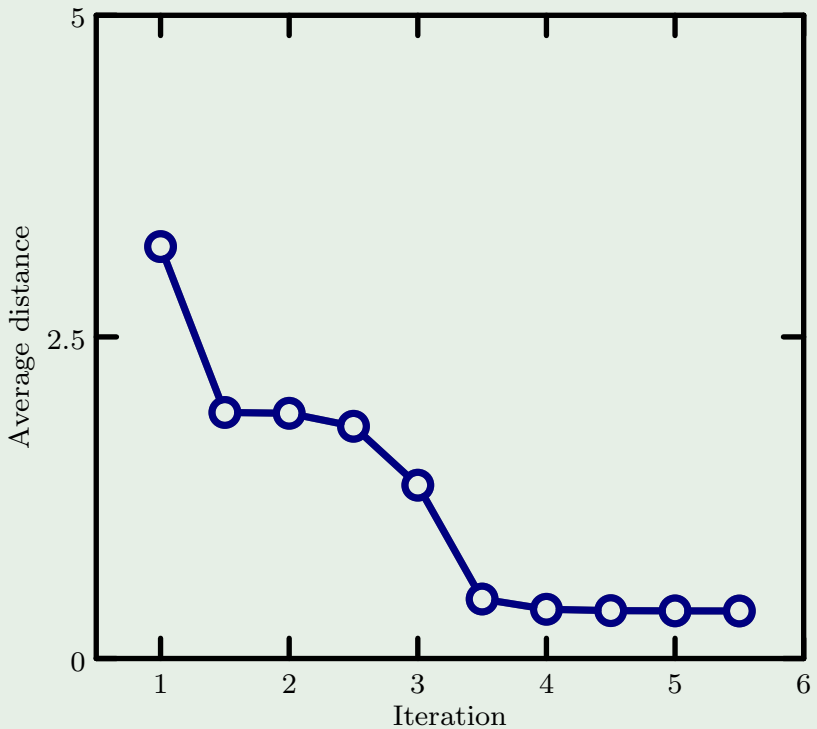




An iterative algorithm

Example: kMeans Clustering





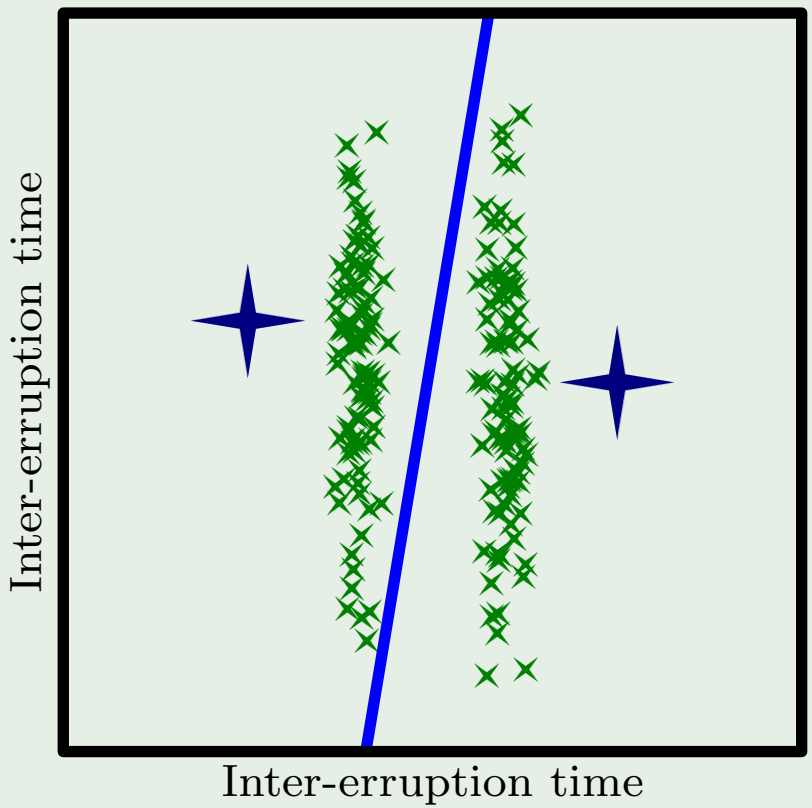
Problems with kMeans

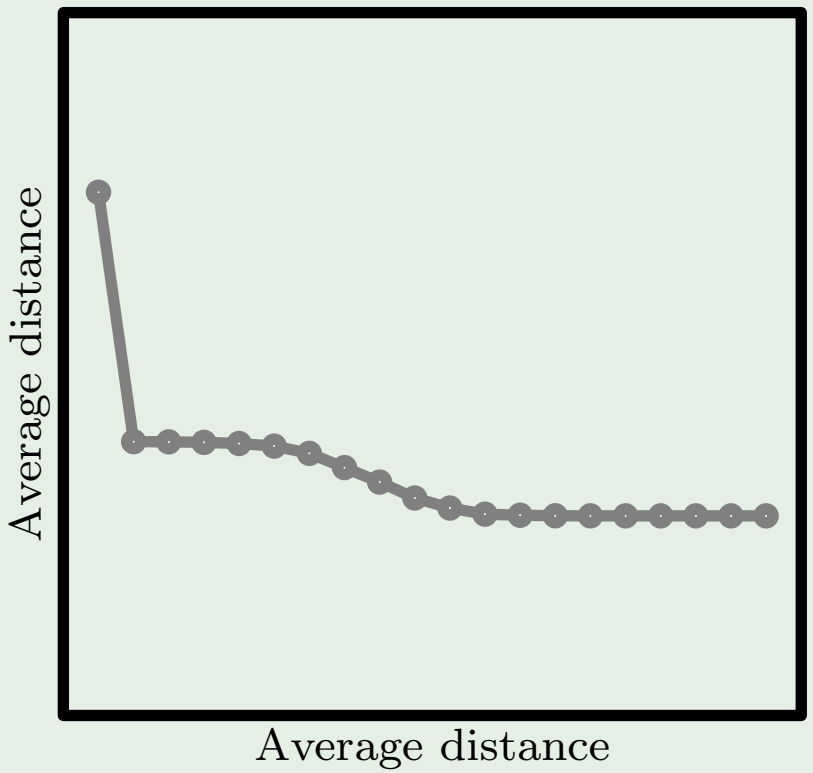
There are some disadvantages to kMeans:

- **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

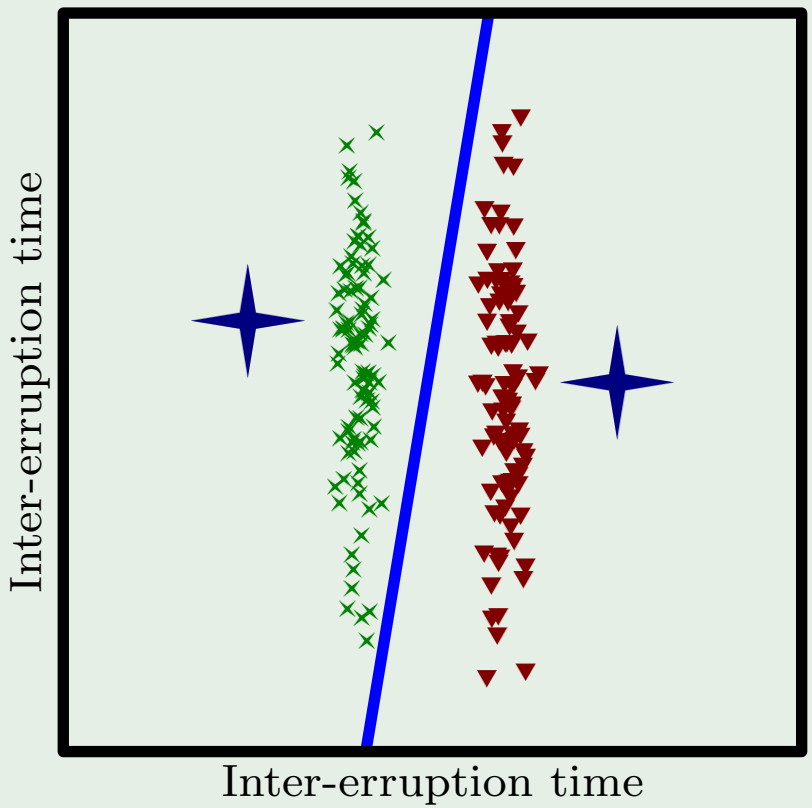
Example: kMeans Clustering

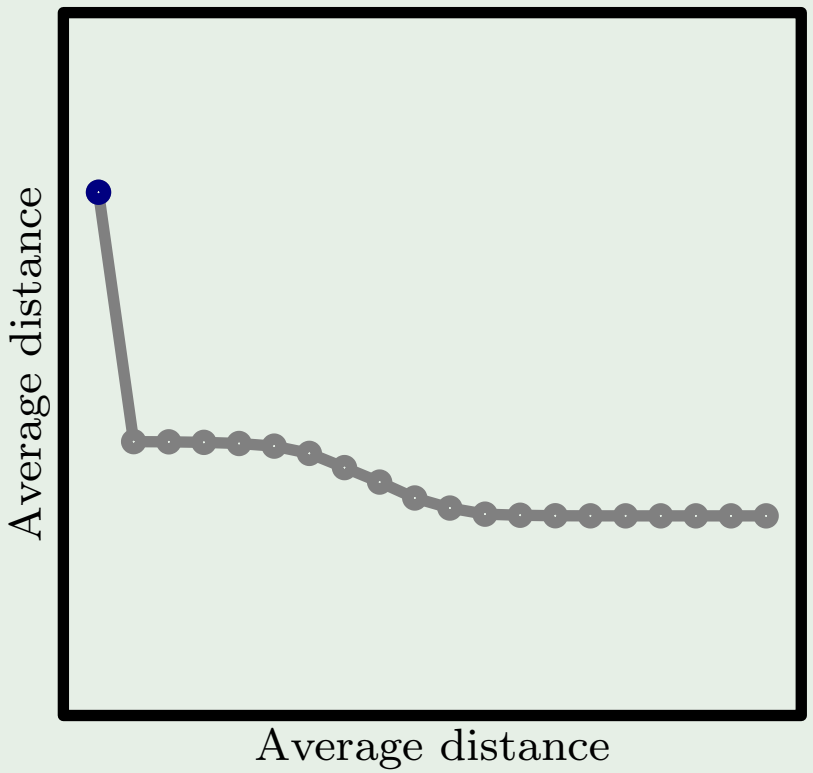




k-means sensitivity to scaling

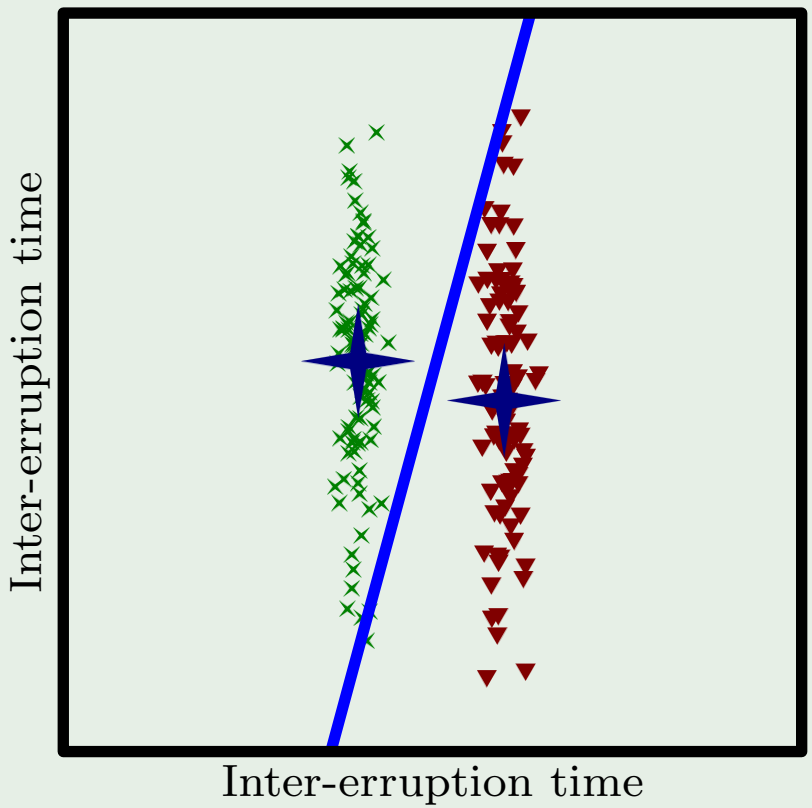
Example: kMeans Clustering

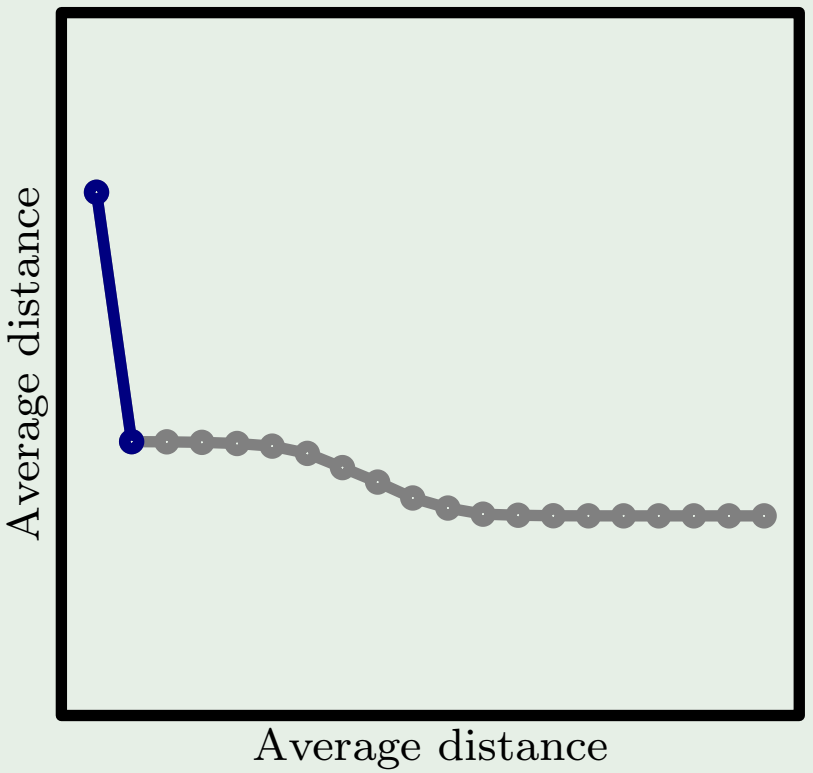




k-means sensitivity to scaling

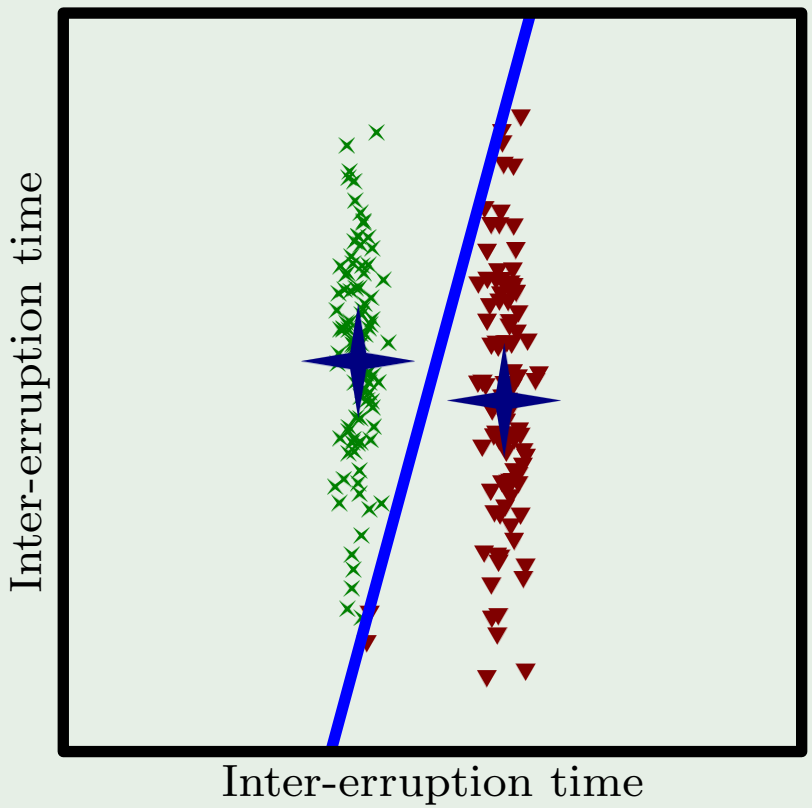
Example: kMeans Clustering

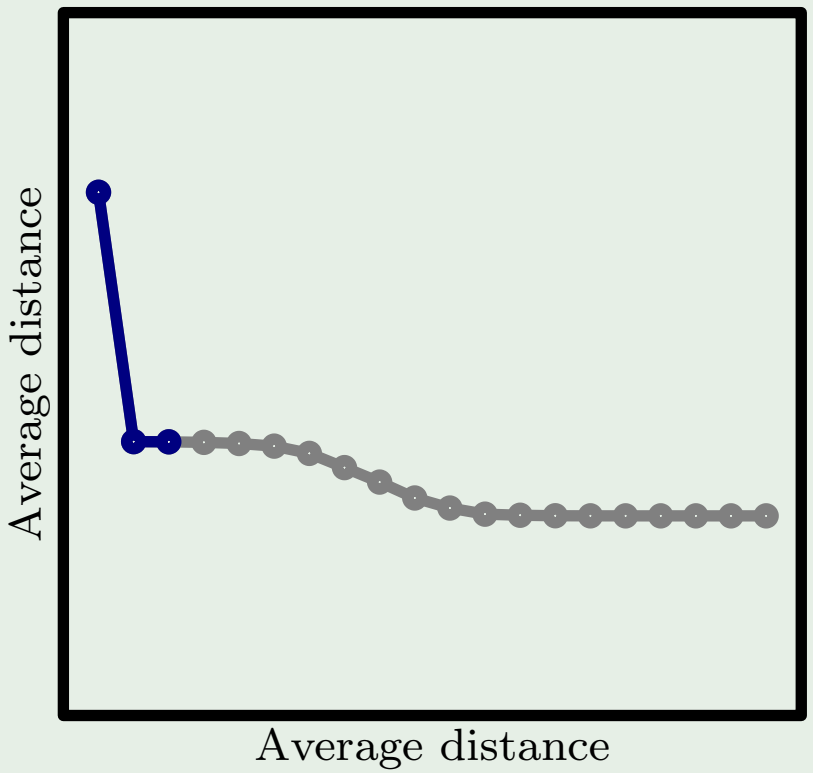




k-means sensitivity to scaling

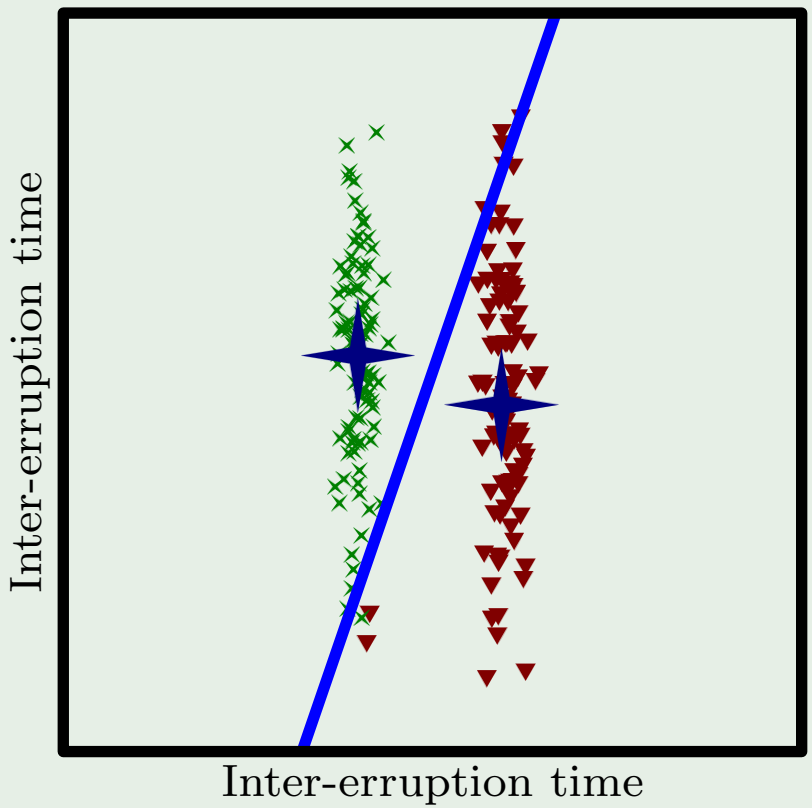
Example: kMeans Clustering

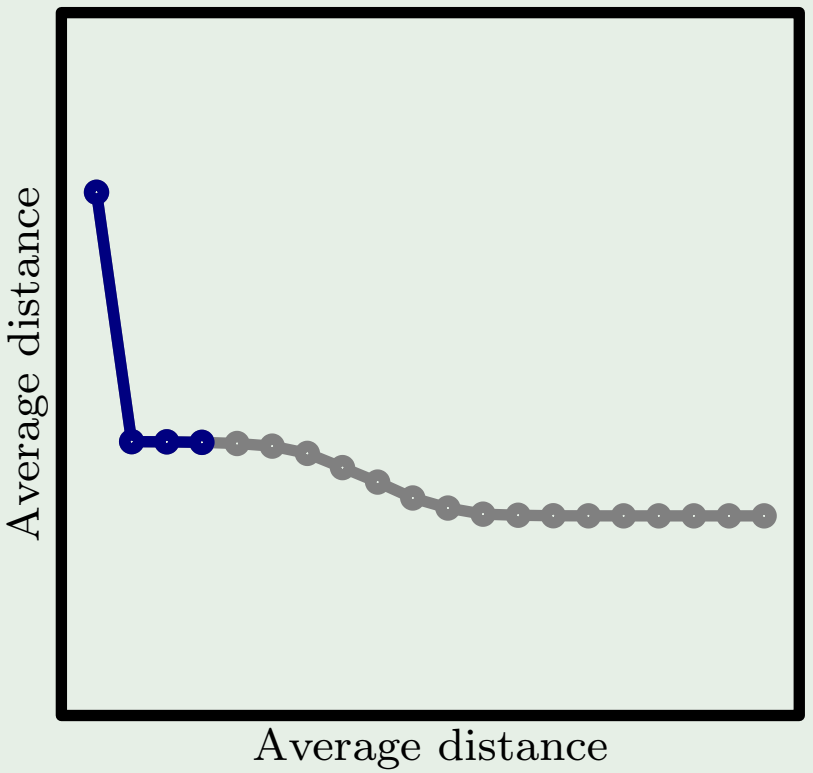




k-means sensitivity to scaling

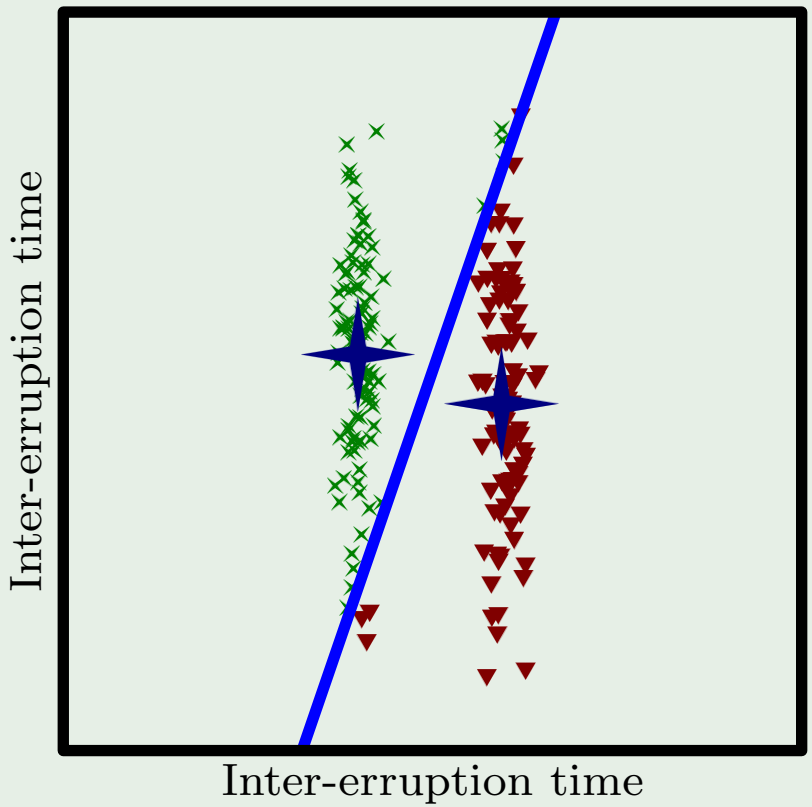
Example: kMeans Clustering

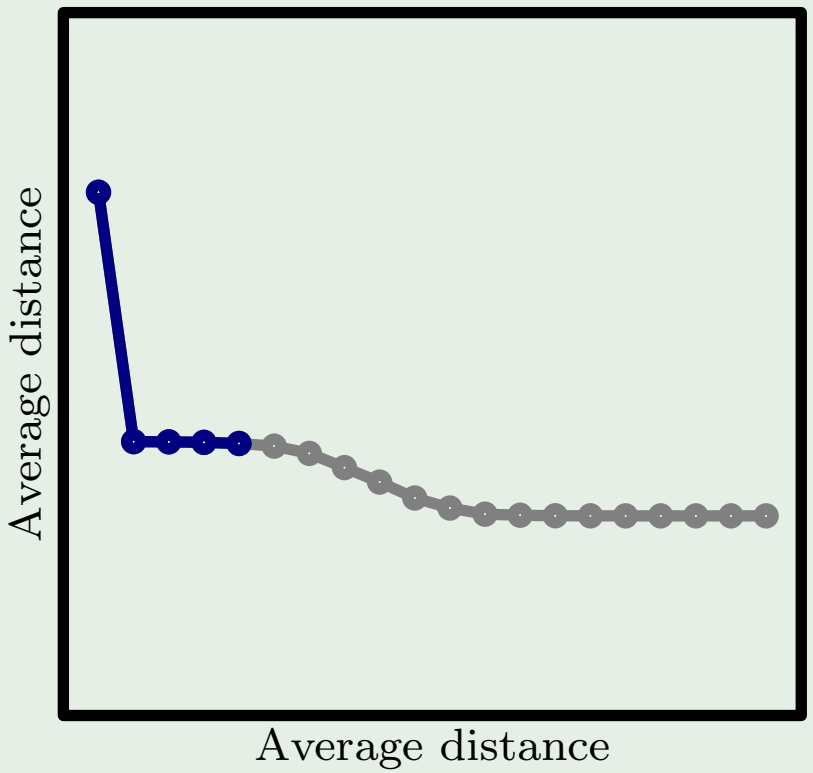




k-means sensitivity to scaling

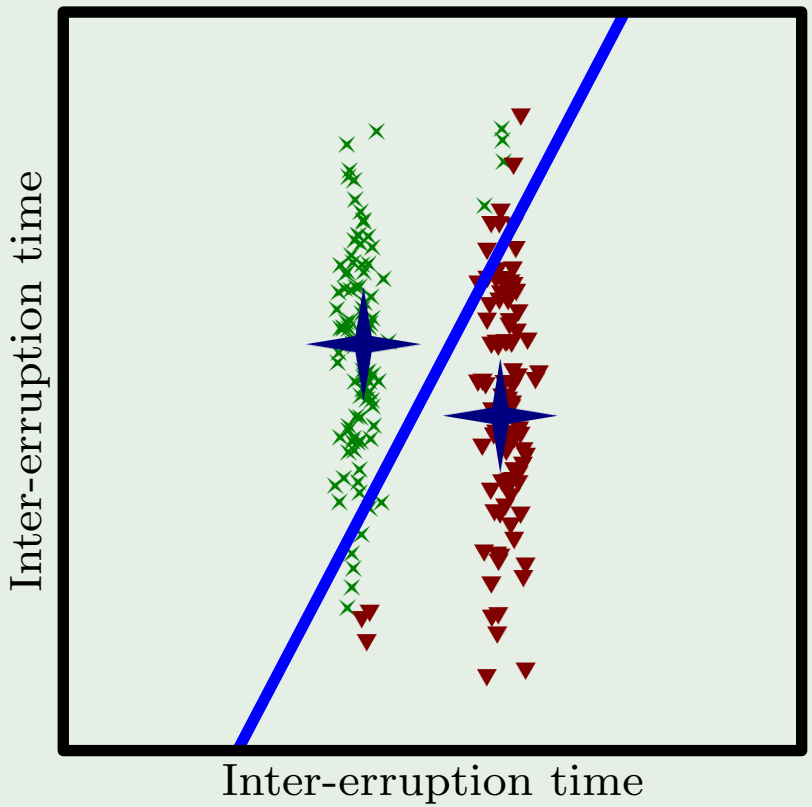
Example: kMeans Clustering

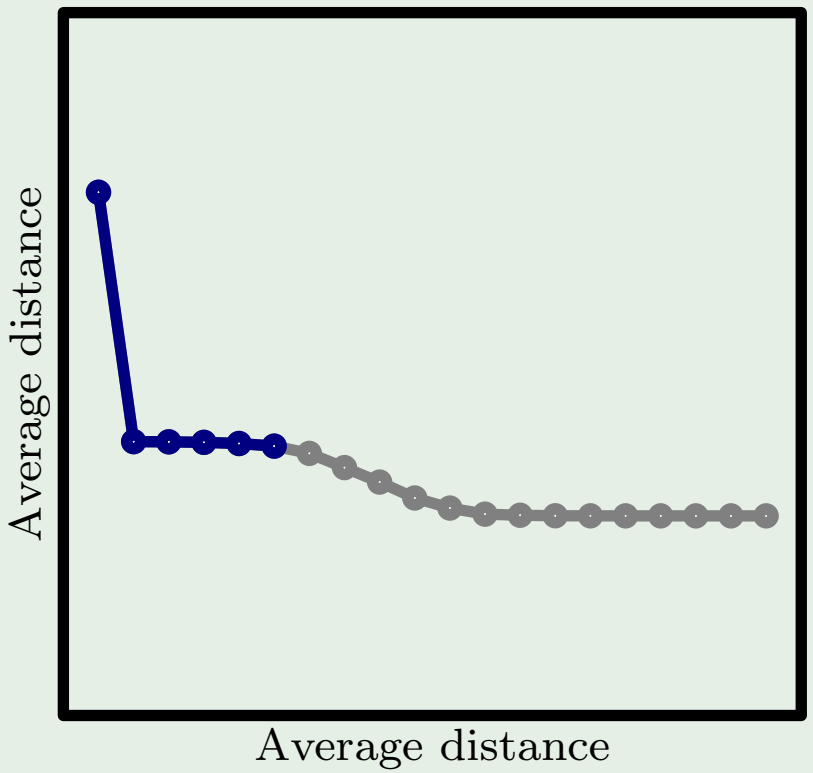




k-means sensitivity to scaling

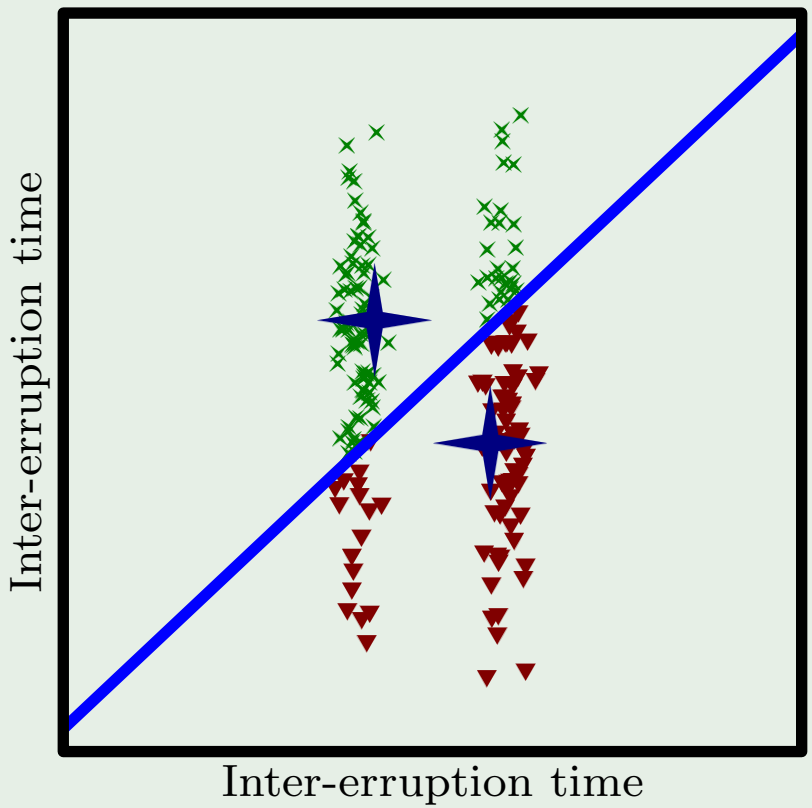
Example: kMeans Clustering

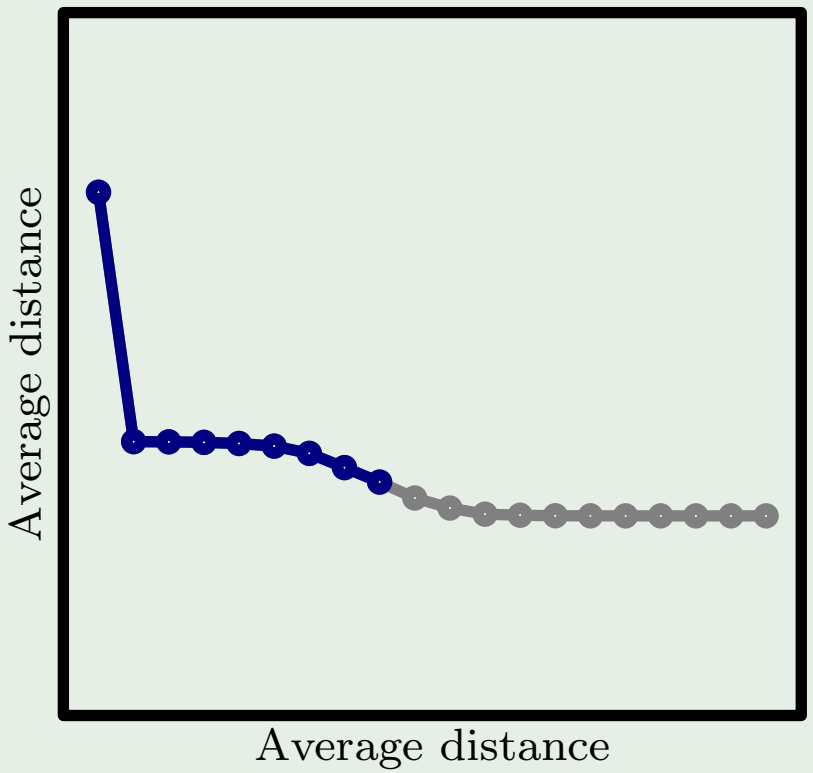




k-means sensitivity to scaling

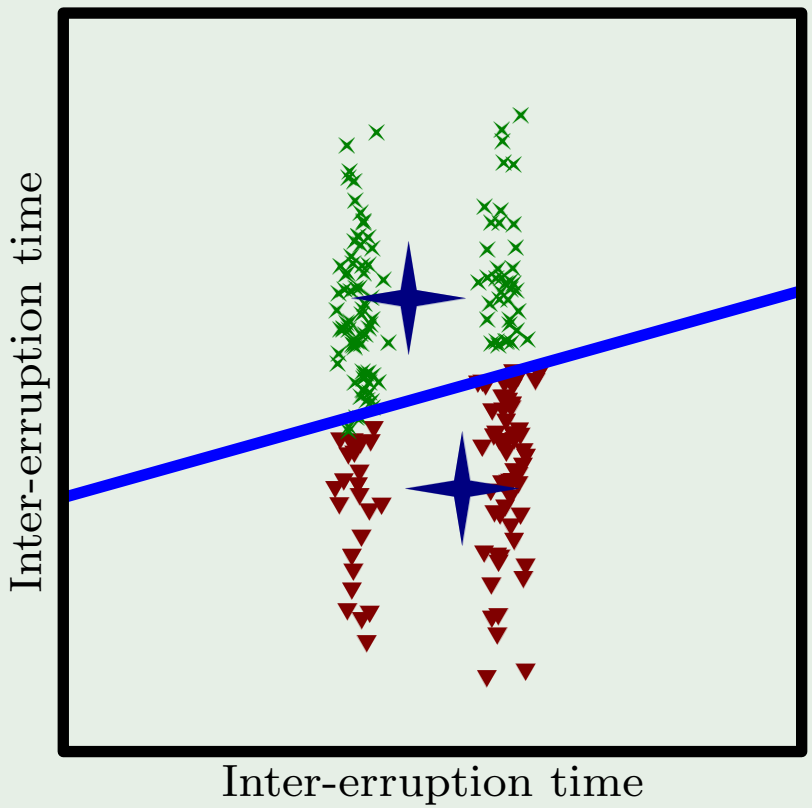
Example: kMeans Clustering

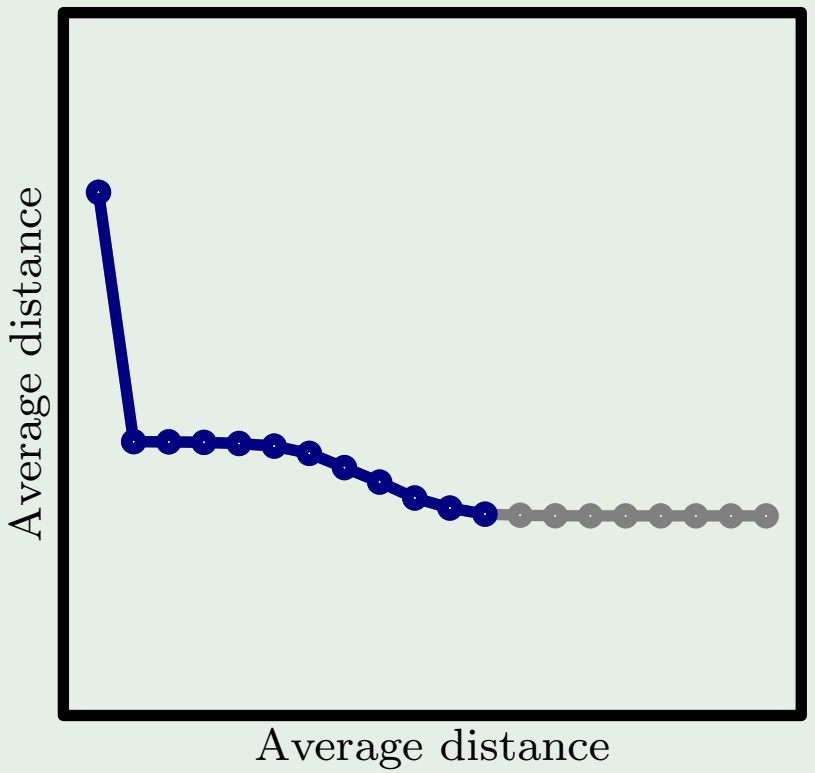




k-means sensitivity to scaling

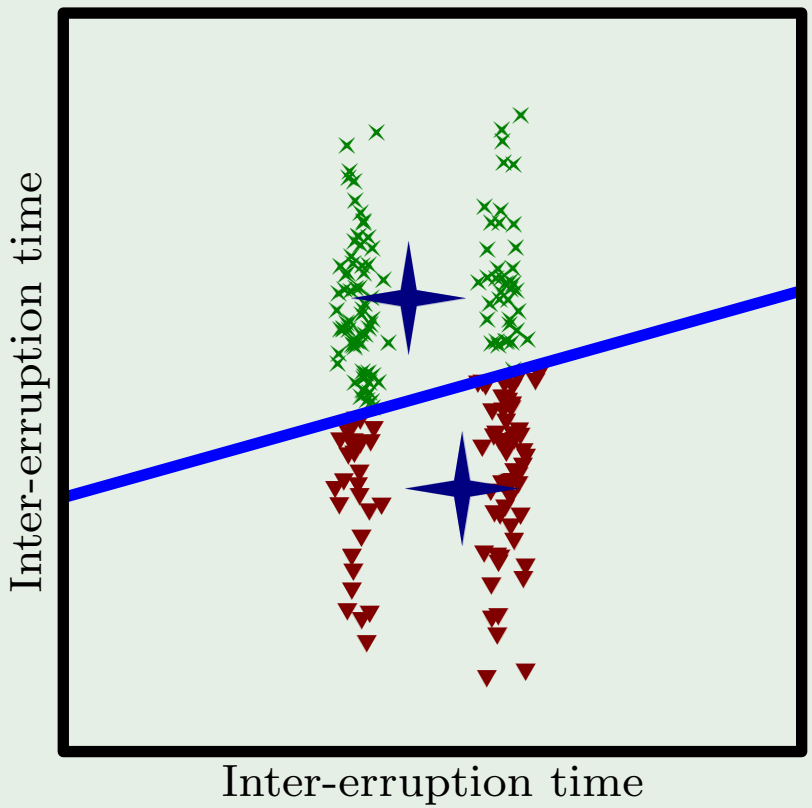
Example: kMeans Clustering

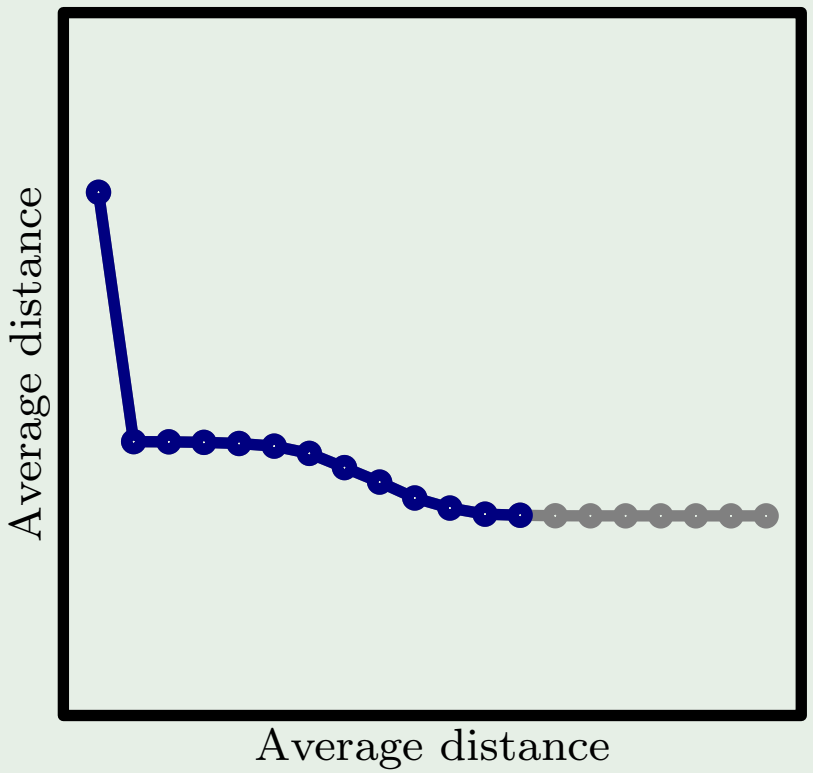




k-means sensitivity to scaling

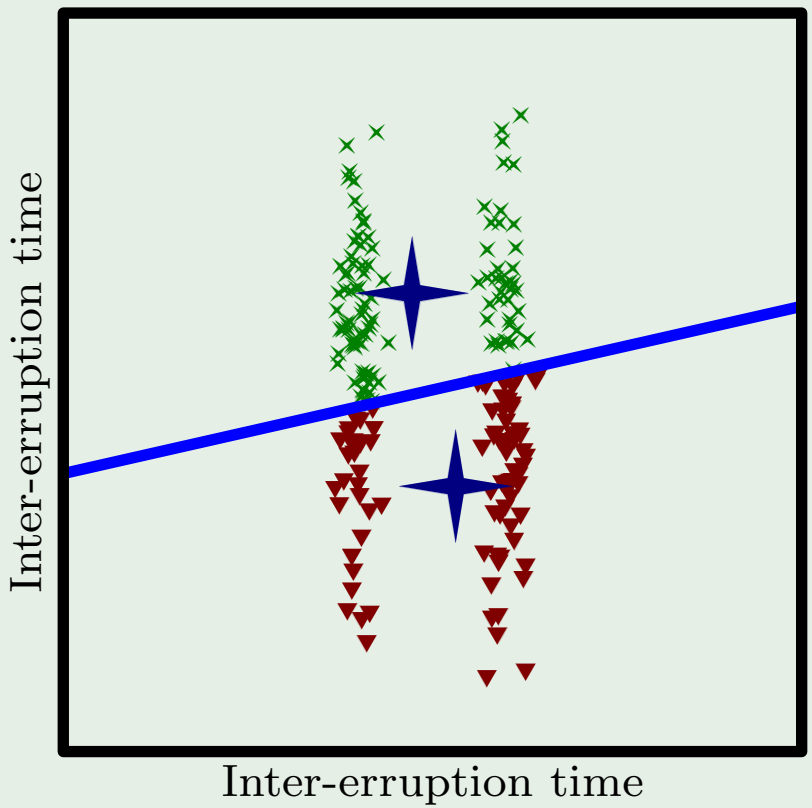
Example: kMeans Clustering

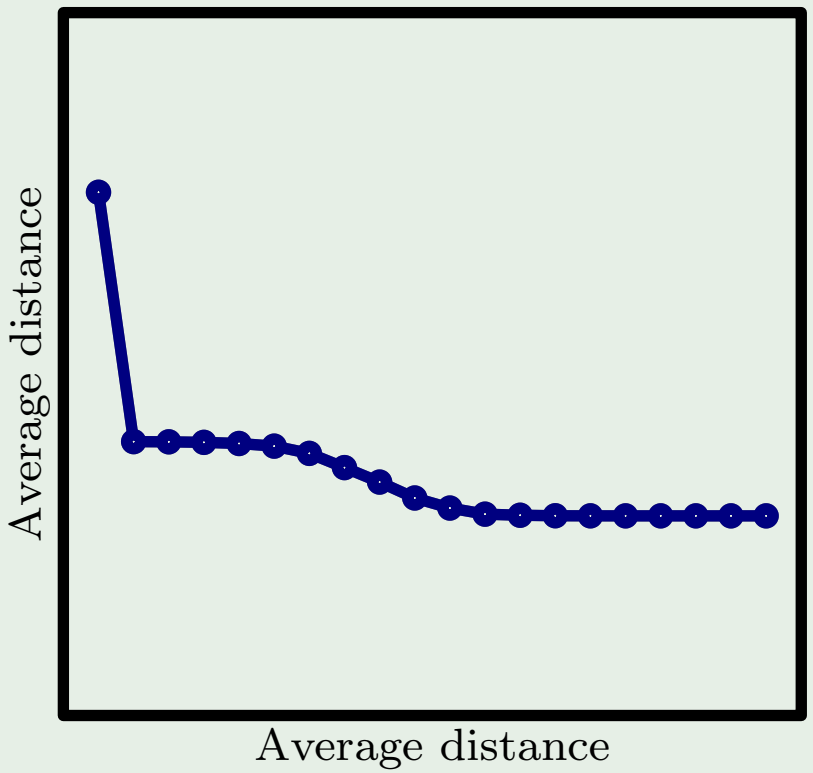




k-means sensitivity to scaling

Example: kMeans Clustering





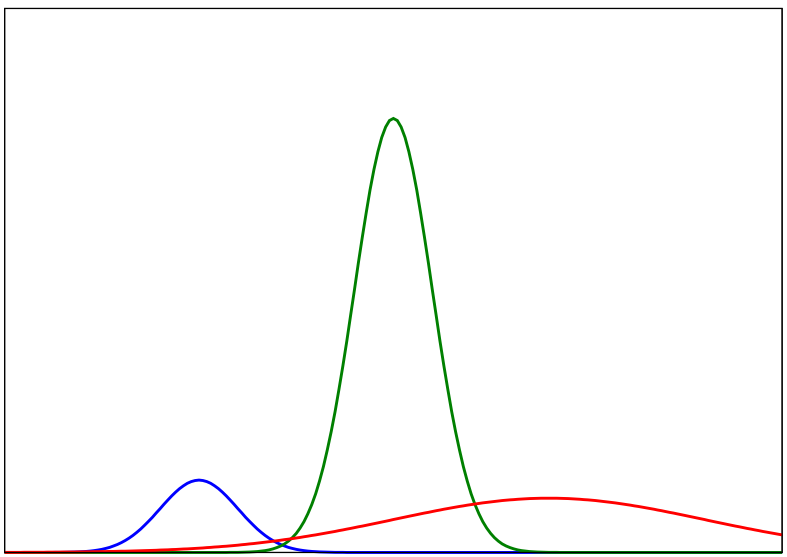
Mixtures of Gaussians

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

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



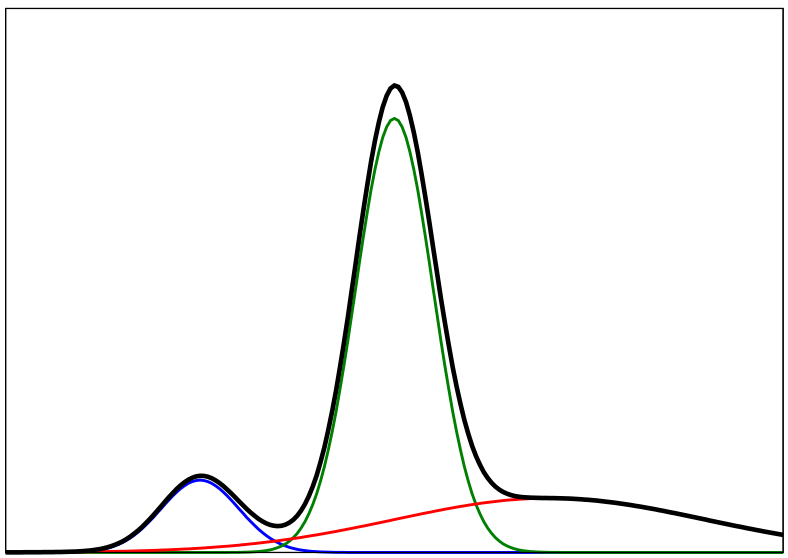
Mixtures of Gaussians

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

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



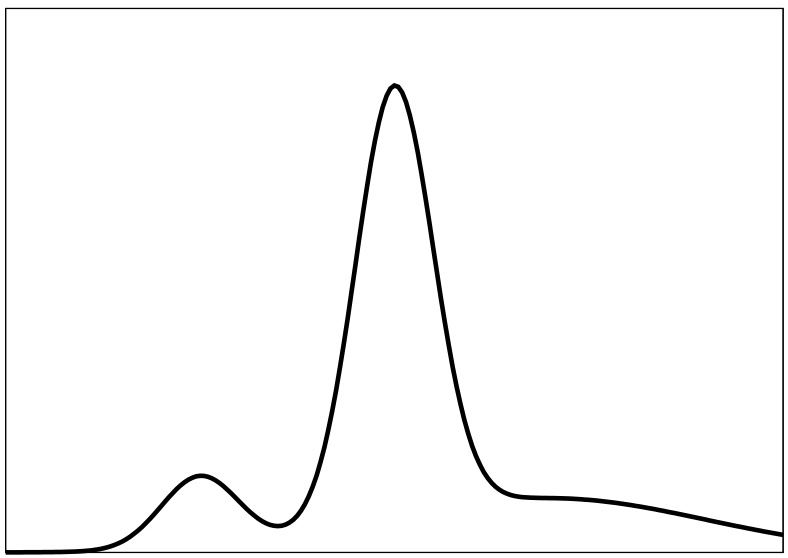
Mixtures of Gaussians

A mixture of Gaussians is a linear combination of Gaussians:

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

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

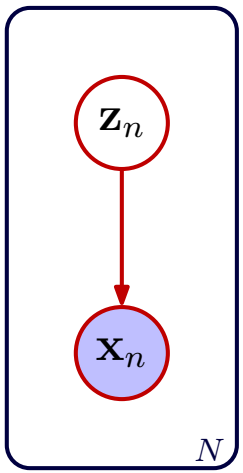
$$\sum_{k=1}^{K} \pi_k = 1 \quad (2)$$



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

The values for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k that maximise the likelihood are:

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

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

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

where we defined $N_k = \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)$

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

The values for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k that maximise the likelihood are:

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

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

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

where we defined $N_k = \sum_{n=1}^{N} p(z_k|\mathbf{x}_n)$

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 EM algorithm for Gaussian Mixtures

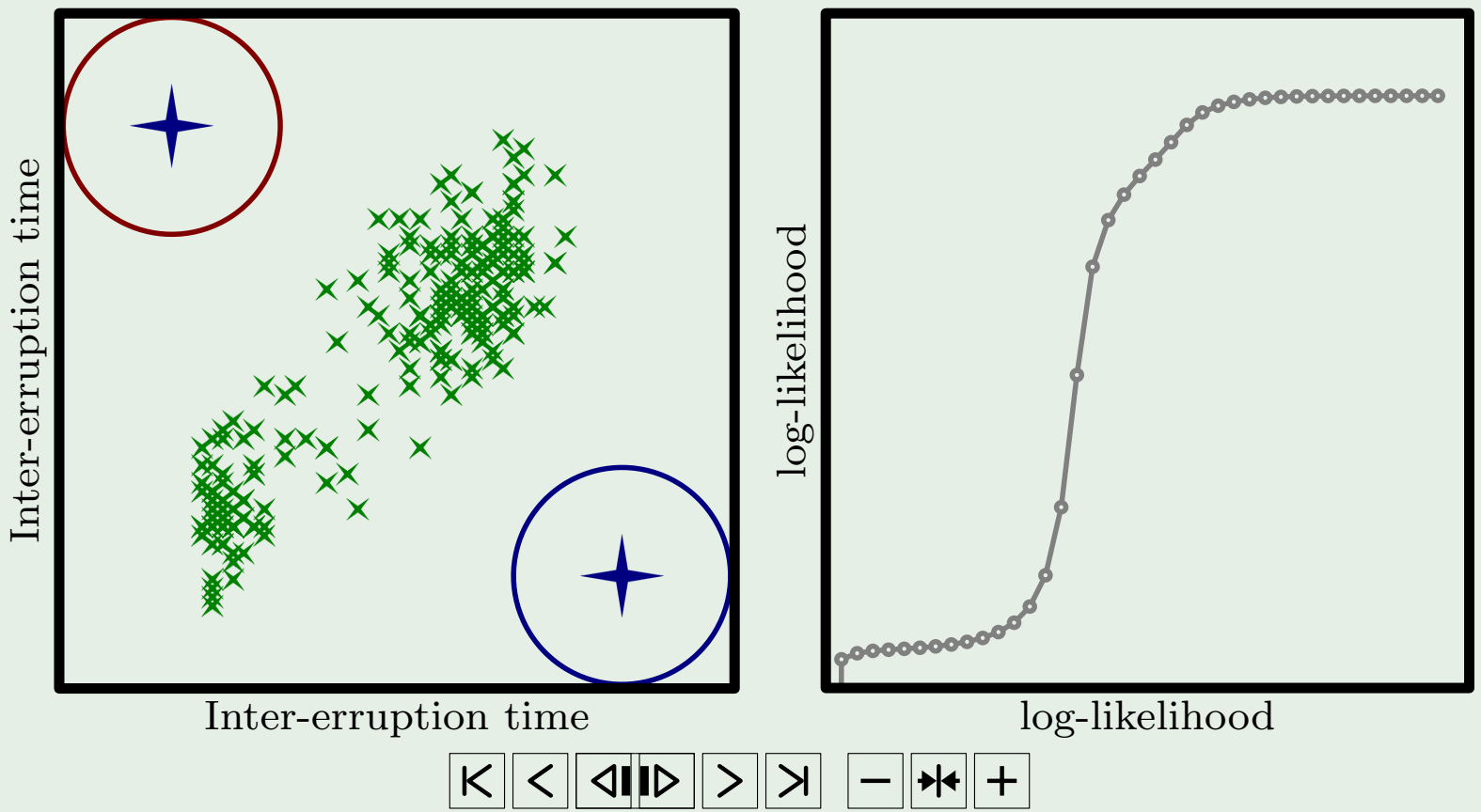
The Expectation-Maximisation algorithm is an iterative update where:

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



About EM for mixtures of Gaussians

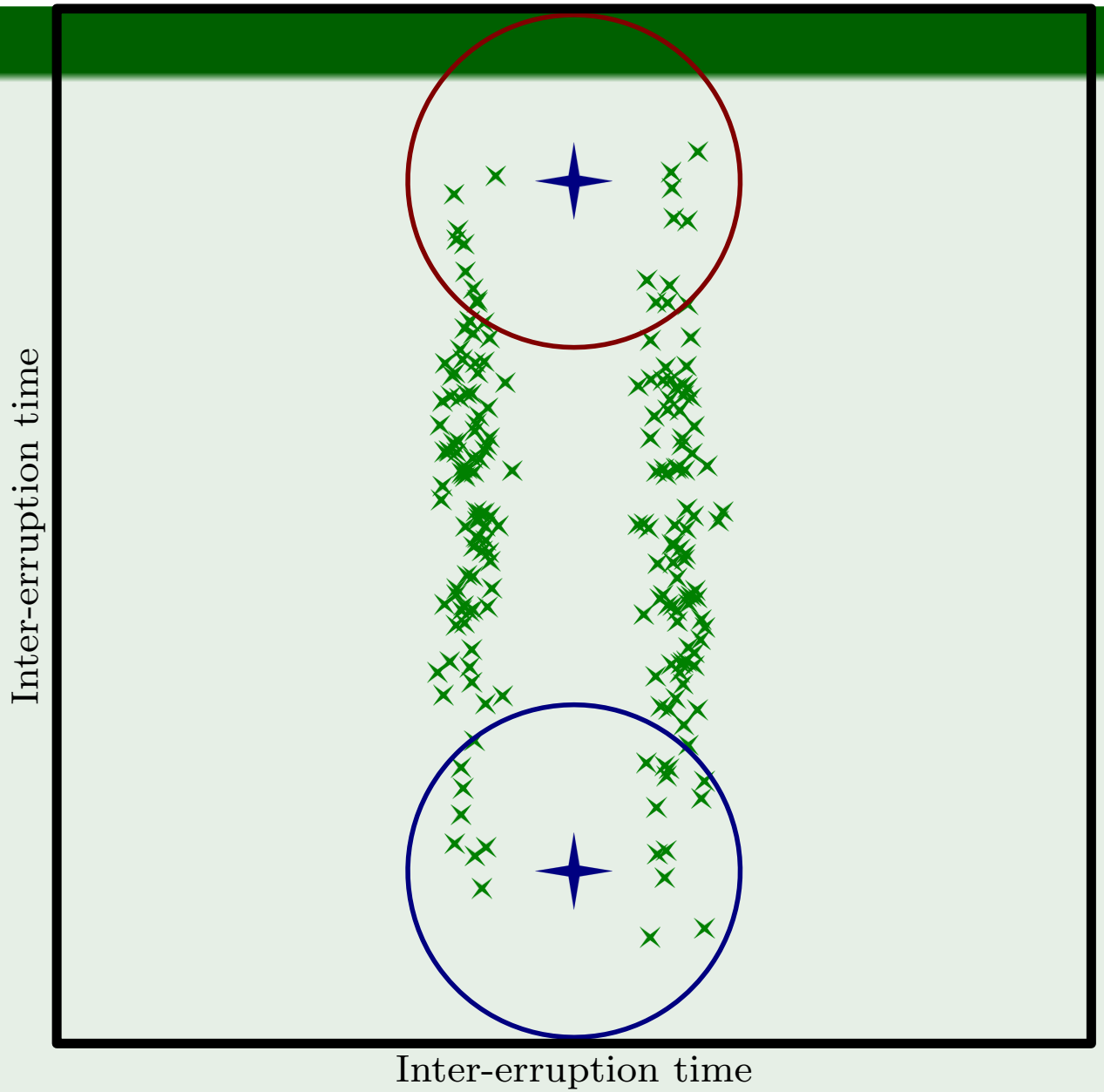
Some things to notice

- 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 $\sigma \to 0$.

Example



The General EM Algorithm

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

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

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 (although we do not need to compute that explicitly)
- In the M-step we maximise the complete log-likelihood with respect to $\boldsymbol{\theta}$

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

EM for MAP learning

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:

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

Why EM works

Consider the log likelihood

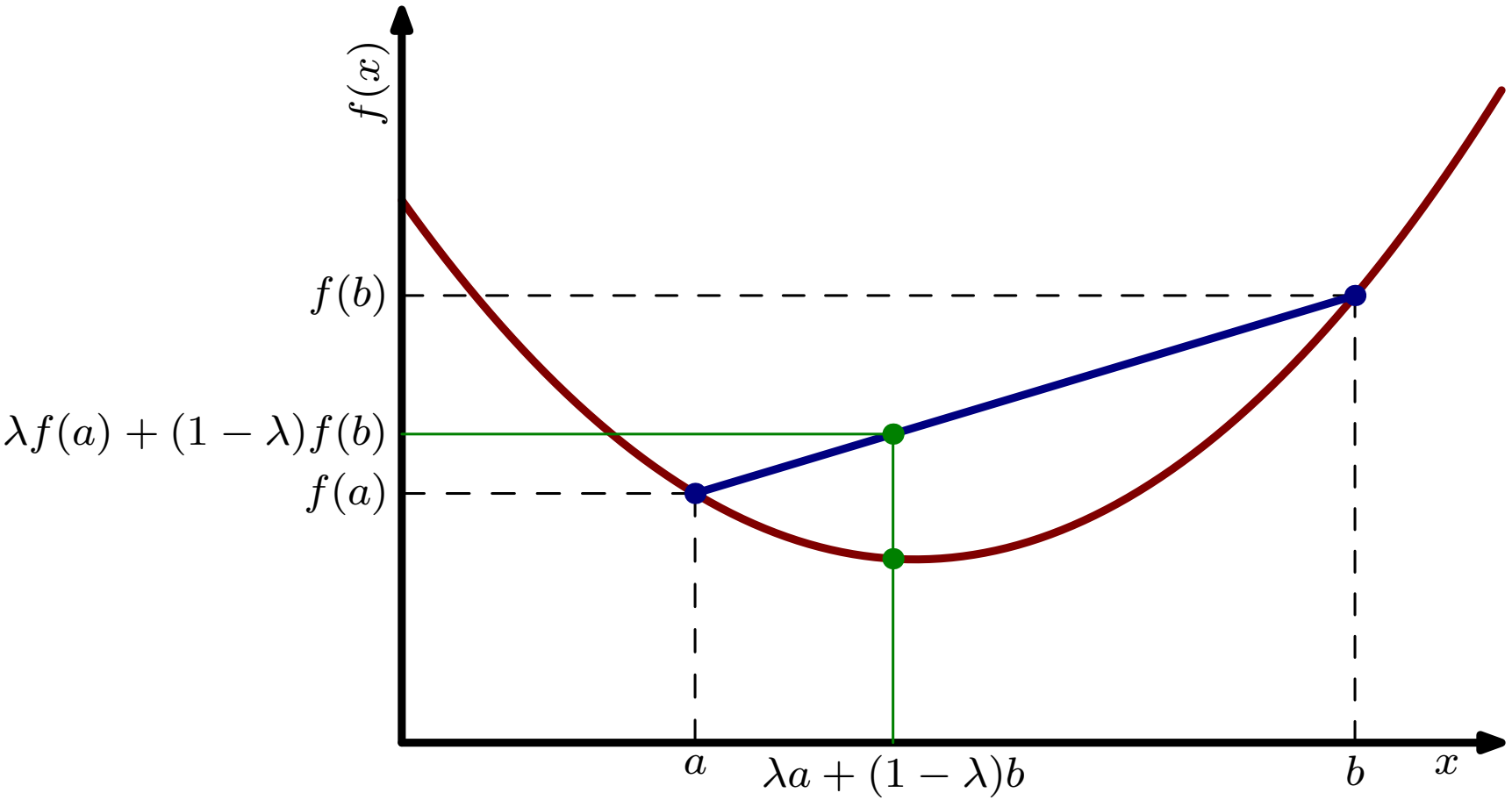
$$\ell(\boldsymbol{\theta}) = \ln p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} \ln p(\mathbf{Z}, \mathbf{X}|\boldsymbol{\theta}) \tag{13}$$

$$= \sum_{\mathbf{Z}} \ln[q(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})}] \tag{14}$$

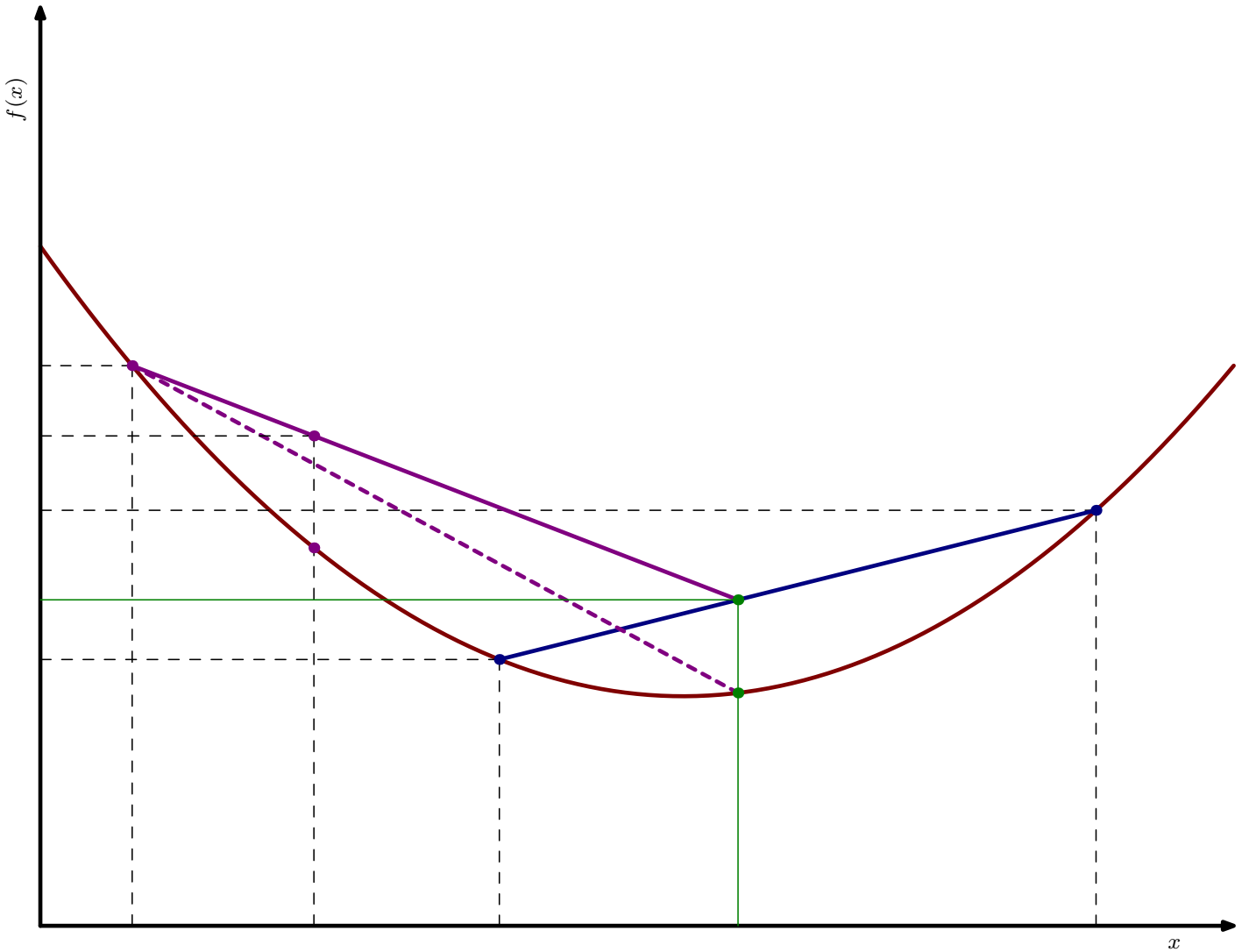
$$\geqslant \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln[\frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})}] \tag{15}$$

By *Jensen's inequality*

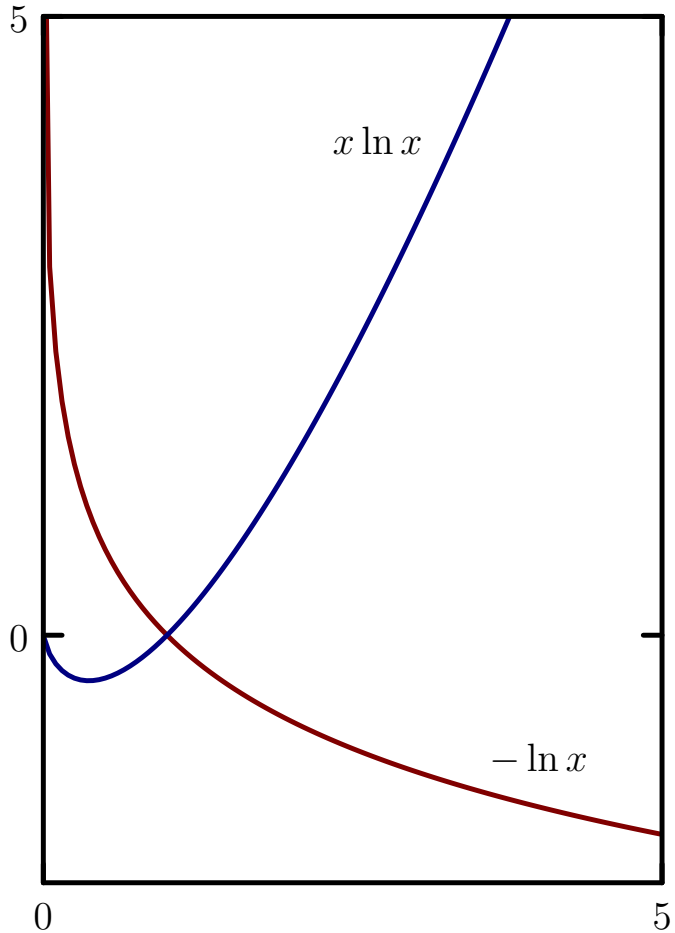
Jensen's inequality



Jensen's inequality



Jensen's inequality



Mutual entropy

Consider the quantity

$$\sum_{\mathbf{z}} q(\mathbf{z}) \ln[\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}]$$

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

$$= \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{q(\mathbf{z})} + \ln p(\mathbf{x}|\boldsymbol{\theta}) \underbrace{\sum_{\mathbf{z}} q(\mathbf{z})}_{=1}$$

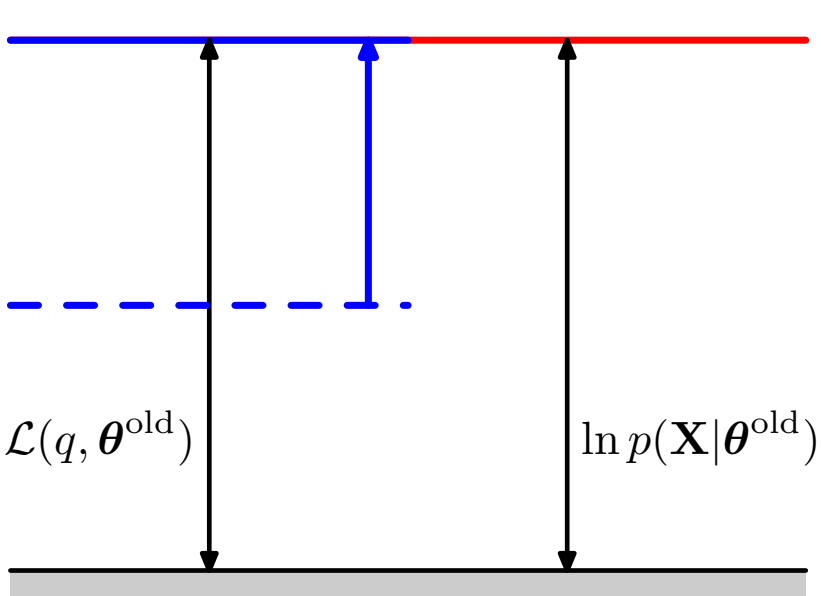
so that

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

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

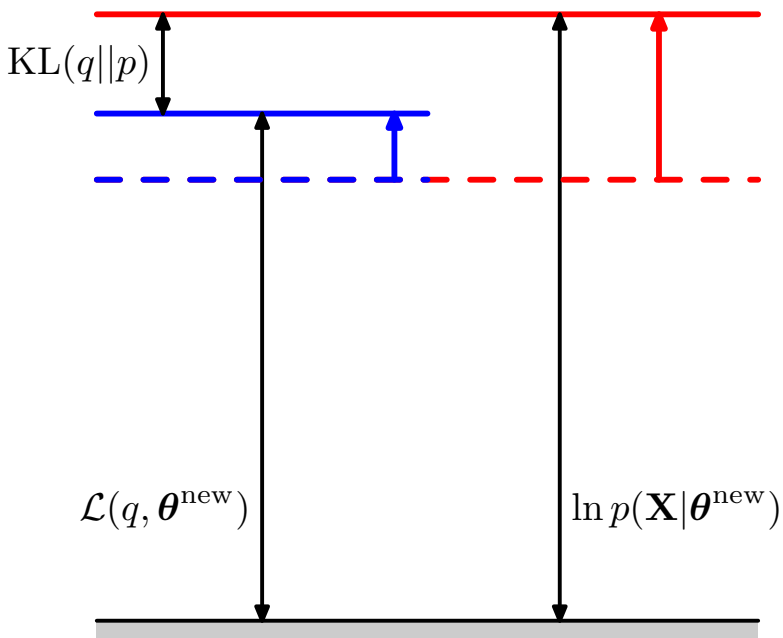
$$\mathcal{L}(q, \boldsymbol{\theta}^{old}) = \ln p(\mathbf{x}|\boldsymbol{\theta}^{old})$$

M-Step

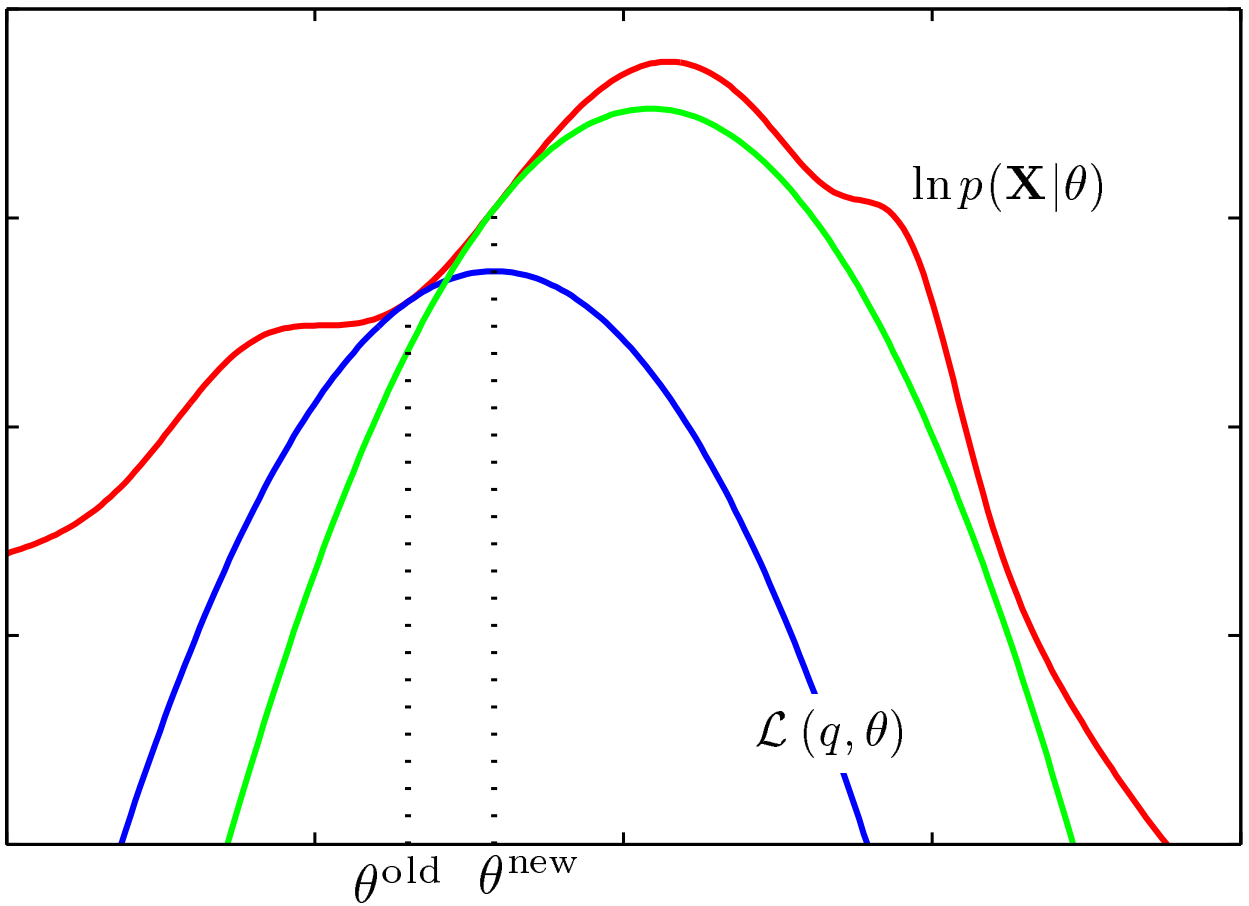
During the M-step, we maximise $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, leaving $q(\mathbf{z})$ untouched. Since $q(\mathbf{x}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{old})$,

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \ln p(\mathbf{z}, \mathbf{x}|\boldsymbol{\theta}) - \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \ln p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \\ &= \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \mathrm{H}[q(\mathbf{z})]\end{aligned}$$

- 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)

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

Lab:

- Implement the EM algorithm for mixtures of Gaussians