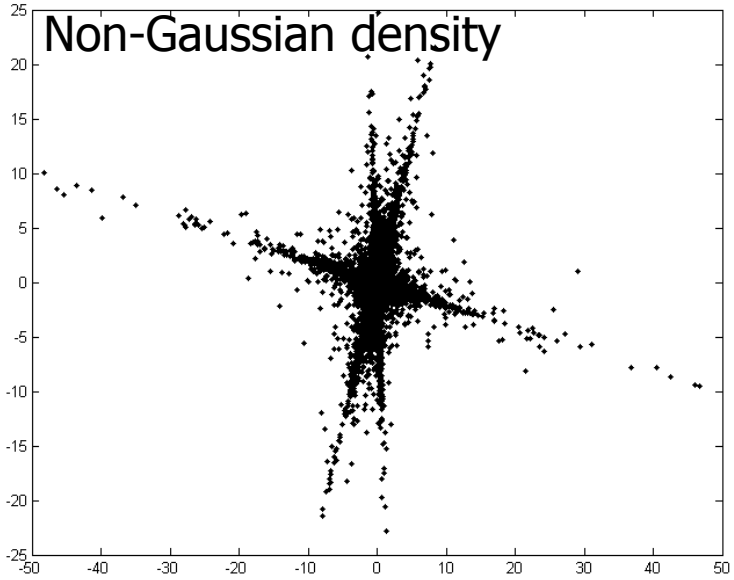
Clustering

Limitations of Gaussian Models

Not all data is well represented by Gaussian models…

Therefore need a possible way of representing more general densities

This is a task for unsupervised learning



Various possible 'density' models:

- Clustering models
- Principal component Analysis
- Independent Component Analysis
- Hidden Markov Models (these introduce temporal structure)

Clustering models

Simple argument – if data is poorly represented by a single blob, try multiple blobs. This is called clustering.

Motivation:

- Provides convenient organisation of data: identify 'interesting' features
- *May* identify *real* structural properties within the data (but not necessarily!)
- Provides an efficient description of the data (e.g. vector quantization)
- Build probability models for arbitrary (non-Gaussian) densities

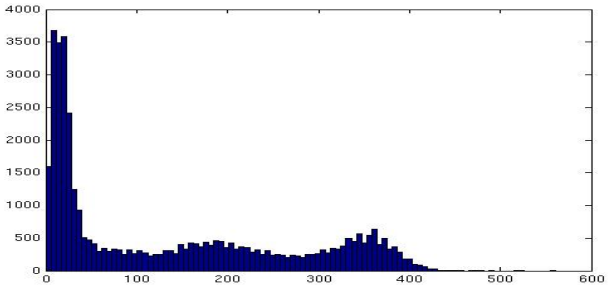
Example: Speech labelling is very costly. Hence we could cluster data and then learn cluster labels

We will concentrate on two forms of clustering:

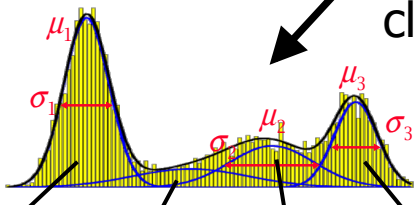
1. Mixtures of Gaussians (with Max. Lik.)
2. K-means clustering

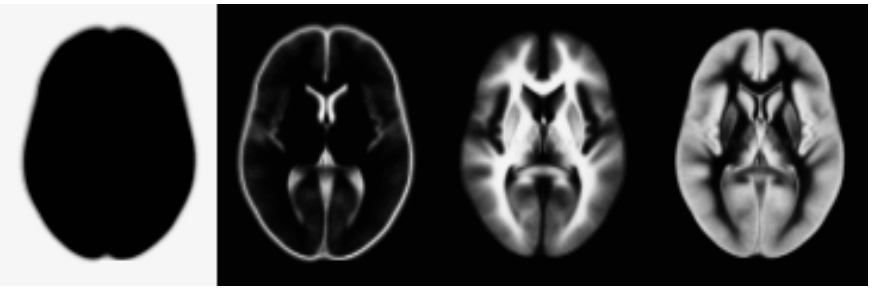
Image segmentation example

Intensity histogram



clustering model





segmentation

Mixtures of Gaussians (MoGs)

One obvious mixture model is the mixture of Gaussians (MoGs) which has the general form:

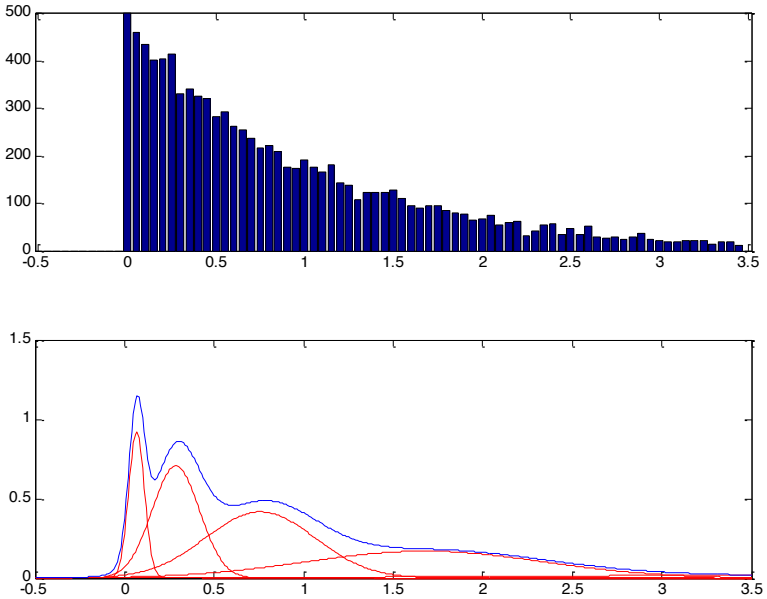
$$P_{mog}(x) \propto \sum_{k=1}^{K} \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k))$$

MoGs are very flexible and can theoretically be used to approximate any density to arbitrary accuracy (with enough Gaussians)...

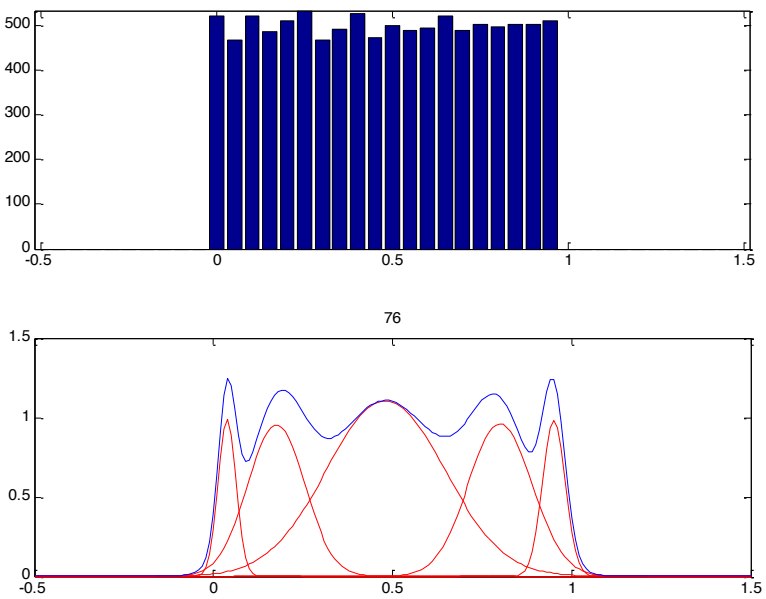
$$\int_x \left| \mathrm{P}(x) - \mathrm{P}_{\mathrm{mog}}(x) \right| dx < \varepsilon$$

Examples of MoGs

Here are a couple of MoG approximations using 5 Gaussians:



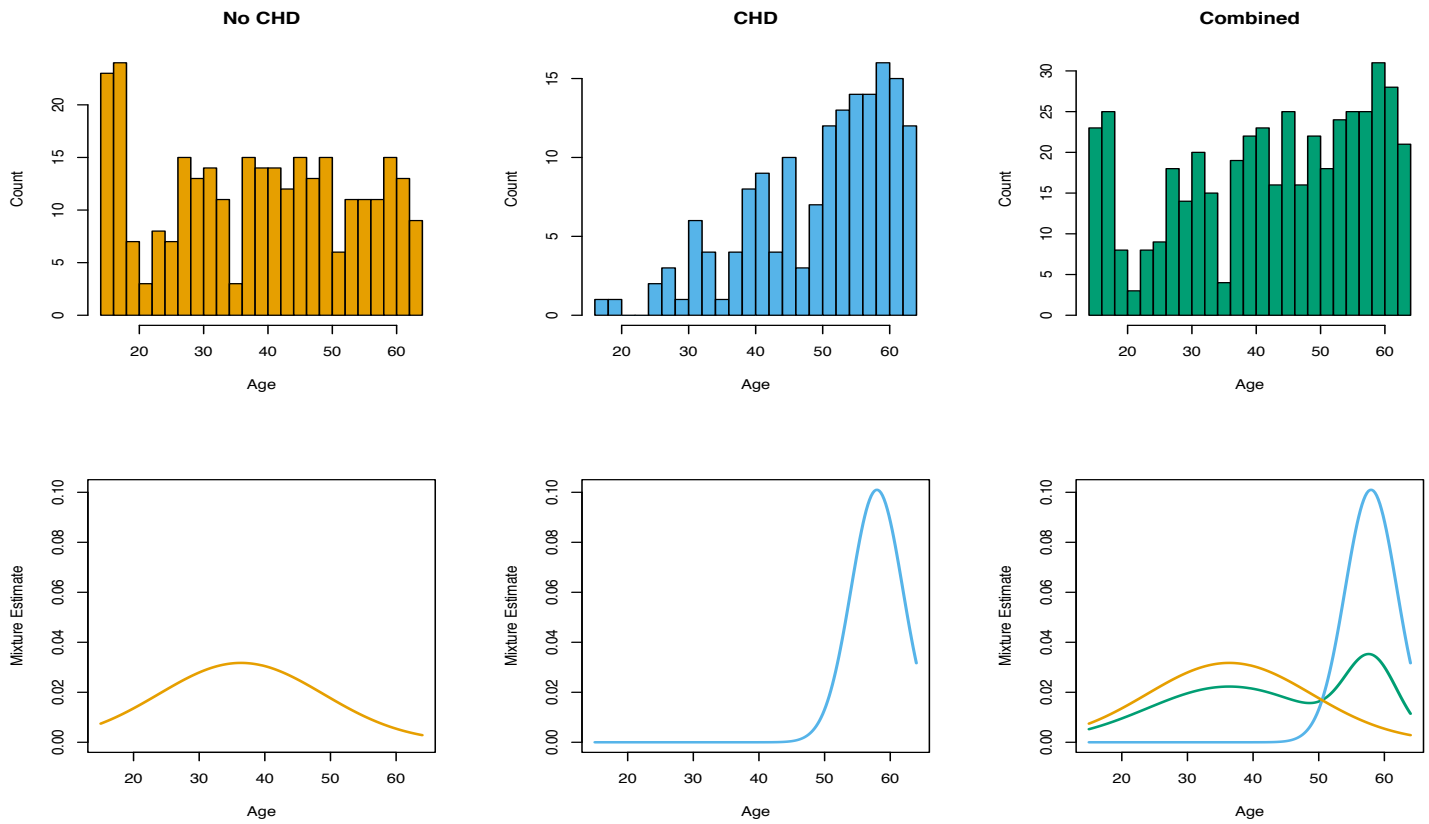
Exponential distribution



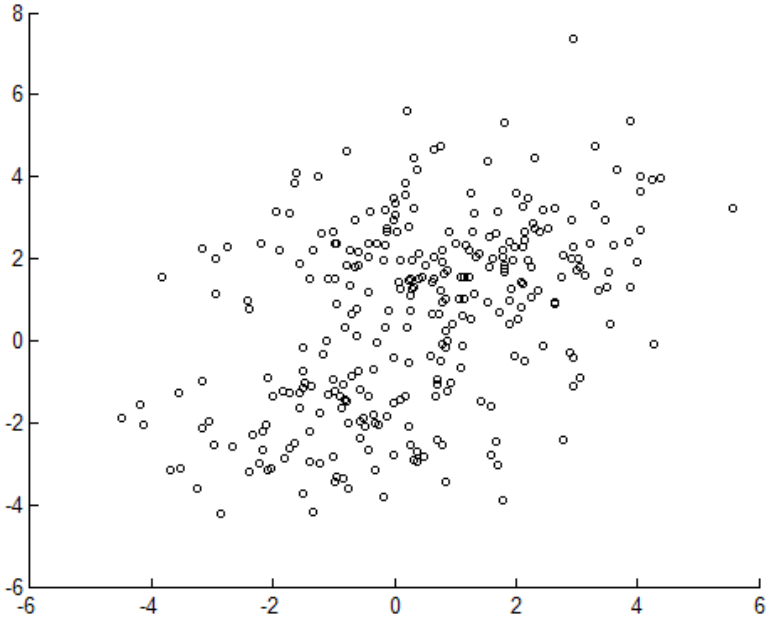
Uniform distribution

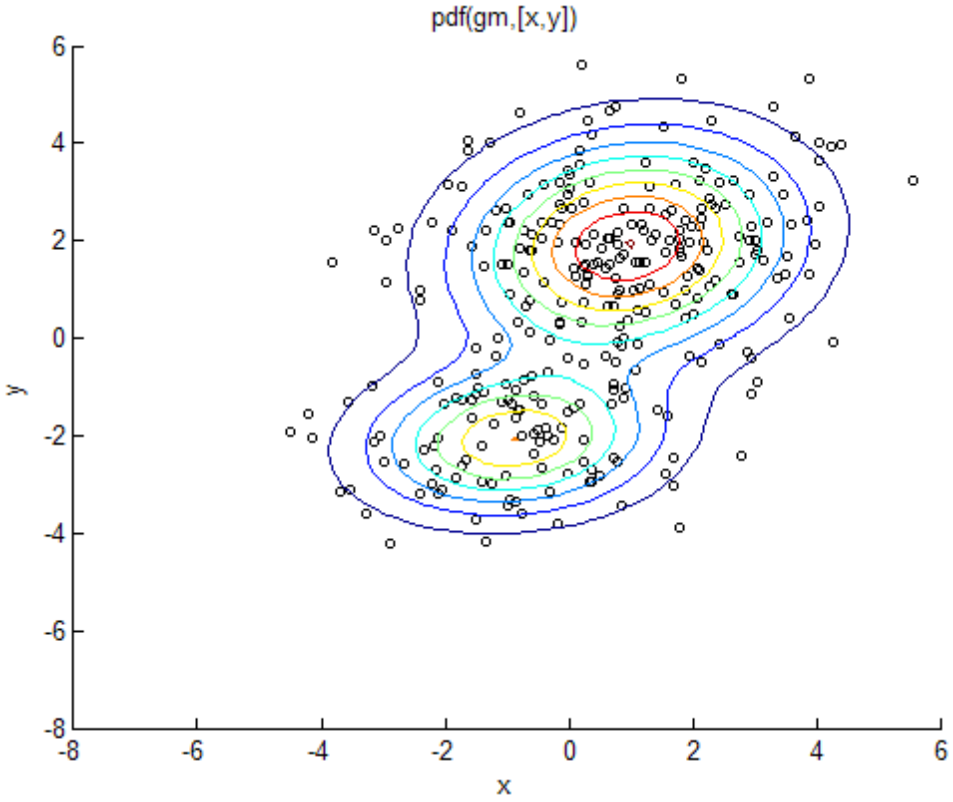
Notice in these two cases the individual Gaussians have no special meaning.

An another example in 1D



An example in 2D





Learning mixture models I

How do we estimate mixture models? Here we will look at Maximum Likelihood. Let θ represent the set of parameters of interest.

Let $p(\mathrm{x} \mid \boldsymbol{\theta}) = \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{P}(\mathrm{c_k})\, \mathrm{P}(\mathrm{x} \mid \mathrm{c_k}, \theta_k)$ so that:

$$E = -\ln p(\chi \mid \boldsymbol{\theta}) \; = -\sum_{n=1}^{N} \ln p(x_n \mid \boldsymbol{\theta})$$

ML estimates imply $\frac{\partial E}{\partial \theta} = 0$

$$\rightarrow \quad -\sum_{n=1}^{N} \frac{\partial p(x_n \mid \boldsymbol{\theta}) / \partial \theta}{p(x_n \mid \boldsymbol{\theta})} = 0 \qquad \text{(since } \partial(\ln\, p(\theta)) / \partial \theta \; = \frac{\partial p(\theta) / \partial \theta}{p(\theta)}\text{)}$$

differentiating w.r.t parameters in the *k*th mixture:

$$\frac{\partial E}{\partial \theta_k} = -\sum_{n=1}^{N} \frac{p(c_k)}{p(x_n \mid \boldsymbol{\theta})} \times \frac{\partial}{\partial \theta_k}\, p(x_n \mid c_k, \theta_k) = 0$$

Learning mixture models II

Applying $\partial(\ln f(\theta))/\partial\theta = \dfrac{\partial f(\theta)/\partial\theta}{f(\theta)}$ again gives:

$$\frac{\partial E}{\partial \theta_k} = -\sum_{n=1}^{N} \frac{p(c_k)p(x_n \mid c_k,\theta)}{p(x_n \mid \theta)} \frac{\partial}{\partial \theta_k} \ln p(x_n \mid c_k,\theta)$$

Note this is a *weighted* version of what we would get if we knew the classes

What are the weights? (applying Bayes rule)

$$p(c_k \mid x_n,\theta) = \frac{p(c_k)p(x_n \mid c_k,\theta)}{p(x_n \mid \theta)}$$

That is: we weigh the contribution of x_n with the probability that it belongs to class c_k:

$$\frac{\partial E}{\partial \theta_k} = -\sum_{n=1}^{N} p(c_k \mid x_n,\theta) \frac{\partial}{\partial \theta_k} \ln p(x_n \mid c_k,\theta) = 0$$

Parameter estimation (cont.)

$$\sum_{n=1}^{N} p(c_k \mid x_n,\theta)\ \frac{\partial}{\partial \theta_k} \ln p(x_n \mid c_k,\theta_k) = 0$$

Ignoring the $p(c_k \mid x_n,\theta)$ term we have the standard ML solution *given* component c_k.

The $p(x_n \mid c_k,\theta_k)$ term is the probability that the point x_n belongs to the c_k component model.

Can we use this to help compute the ML solution for the mixture?

EM iterative update

Although $p(c_k \mid x_n, \theta)$ is not independent of θ_k we can treat it as such and generate the following iterative algorithm:

Expectation - Maximization (beginning with initial values $\hat{\theta}^{(0)}$)

E – step

calculate: $\hat{P}_{n,k}^{(i)} = p(c_k \mid x_n, \hat{\theta}^{(i)})$

for all k and n

M-step

maximise the expected log likelihood:

$$\hat{\theta}_k^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \sum_n \hat{P}_{n,k}^{(i)} \ln p(x_n \mid c_k, \theta_k)$$

A converged solution is a stationary point (e.g. local maximum) of the likelihood

EM algorithm for MoGs I

We can now easily apply the EM algorithm to estimate MoG parameters.

Let $w_k^{(i)}$ represent the ith estimate for $p(c_k)$:

- E-step

$$\text{let } \alpha_{\text{n,k}} = \frac{w_k^{(i)}}{\det(\Sigma_k)^{1/2}} \exp\left(-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)\right)$$

$$\text{then } \hat{P}_{n,k}^{(i)} = \frac{\alpha_{\text{n,k}}}{\sum_{\text{j}} \alpha_{\text{n,j}}}$$

using $p(c_k \mid x_n, \theta^{(i)}) = p(x_n \mid c_k, \theta^{(i)}) p(c_k) / p(x_n \mid \theta^{(i)})$

EM algorithm for MoGs II

- M - step

$$1.\ \mu_{\mathrm{k}}^{(i+1)} = \sum_n \hat{P}_{n,k}^{(i)} x_n \Big/ \sum_n \hat{P}_{n,k}^{(i)} \qquad \text{(weighted mean)}$$

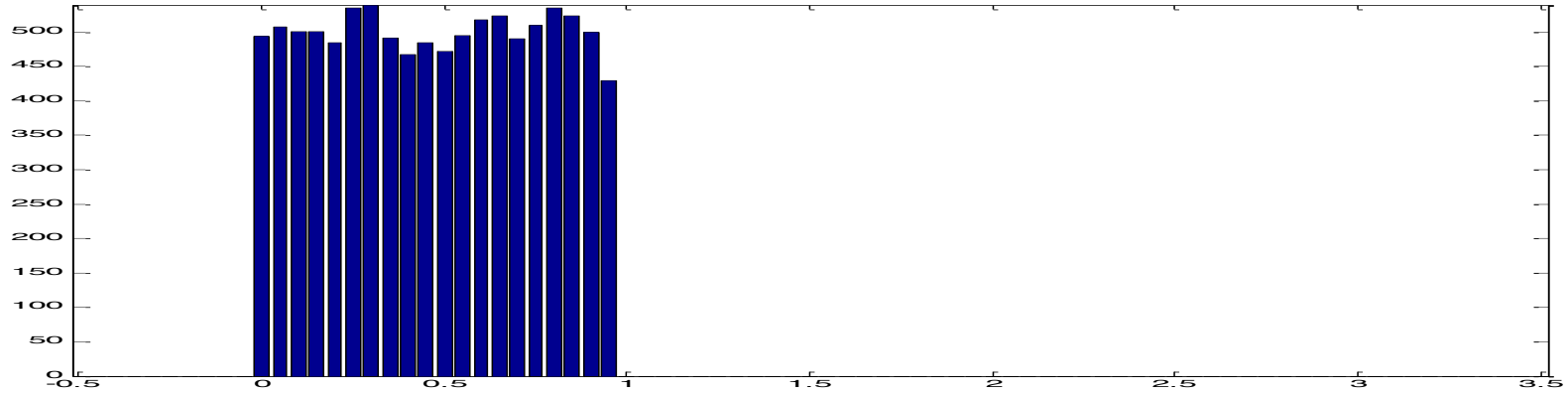
$$2.\ \Sigma_{\mathrm{k}}^{(i+1)} = \frac{\sum_n \hat{P}_{n,k}^{(i)} (x_n - \mu_{\mathrm{k}}^{(i+1)})(x_n - \mu_{\mathrm{k}}^{(i+1)})^T)}{\sum_n \hat{P}_{n,k}^{(i)}} \qquad \text{(weighted covariance)}$$

It is also possible to show that we can estimate the priors for each class c_k using the following update :

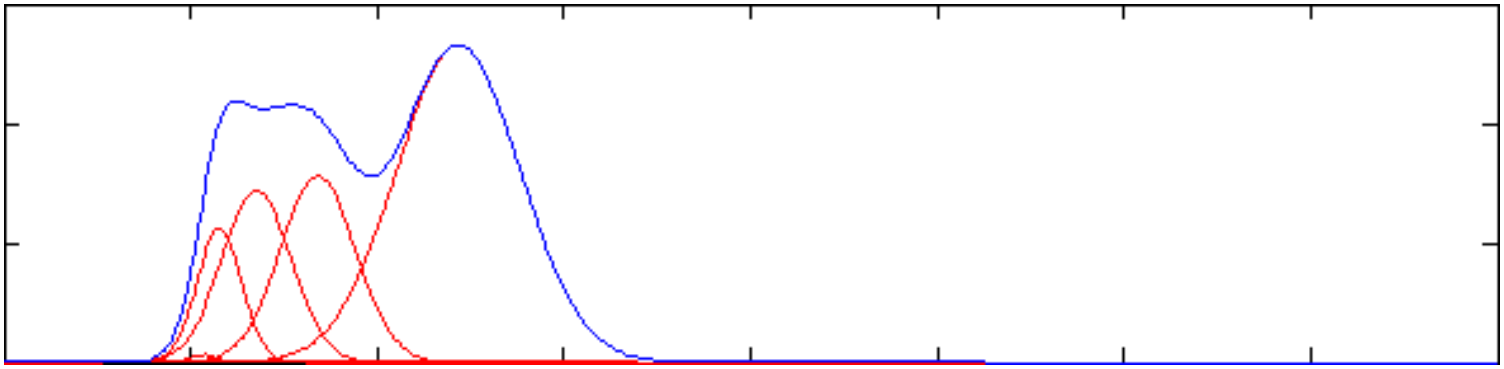
$$3.\ w_k^{(i)} = \frac{1}{N} \sum_n \hat{P}_{n,k}^{(i)}$$

All the variables can be updated as weighted averages!

EM algorithm for MoG example



This demonstrates 100 EM iterations of a 5 Gaussian MoG



Convergence issues

Because we are using ML learning we need to worry about overfitting. MoGs can potentially exhibit a very interesting form of overfitting – **Singular points**

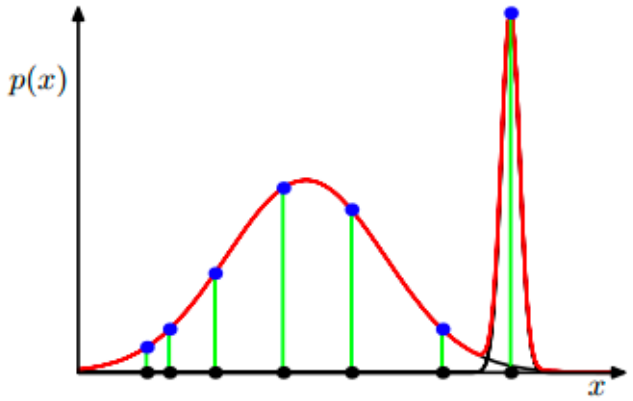
Recall the likelihood for x_n is:

$$P(x_n \mid \mu_1,\ldots,\mu_K,\Sigma_1,\ldots\Sigma_K) \propto \sum_{k=1}^{K} \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp(-\frac{1}{2}(x_n-\mu_k)^T \Sigma_k^{-1}(x_n-\mu_k))$$

Now suppose $x_n = \mu_k$ for some k and $\Sigma_k \rightarrow 0$. Then

$$P(x_n \mid \mu_1,\ldots,\mu_K,\Sigma_1,\ldots\Sigma_K) \approx \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp(0) \rightarrow \infty$$

So the likelihood → ∞.
These singular points ALWAYS exist.
In practice the optimization algorithm usually misses them
and finds a local maximum!



K - means clustering

A simpler clustering algorithm that combines mixture modelling and classification is as follows. First initialize ***K*** centres μ_k then iteratively repeat the following steps.

Step - 1 : classify x_n according to the nearest μ_k

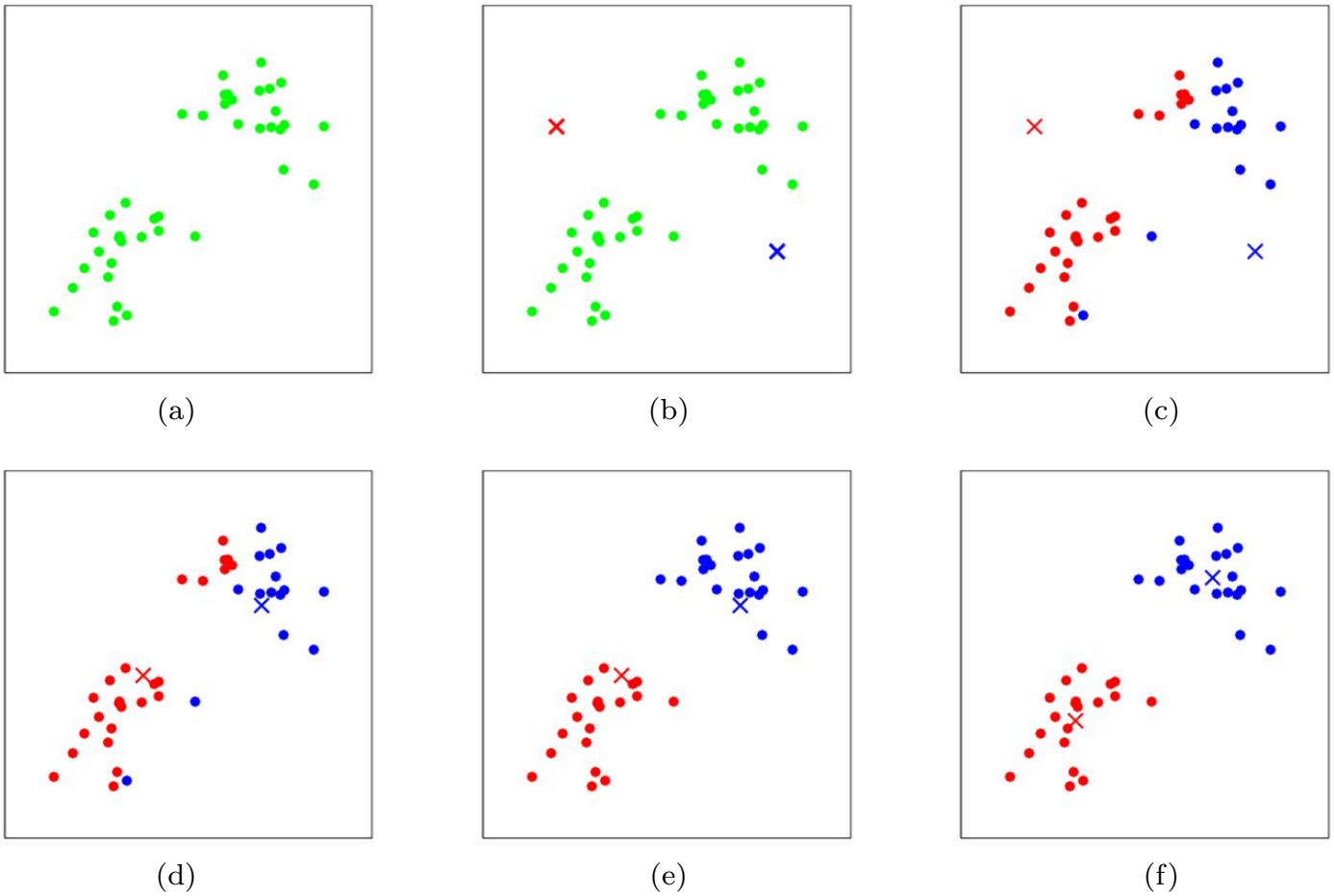
$$x_n \in C_k \text{ if } \|x_n - \mu_k\| < \|x_n - \mu_j\| \text{ for all } j \neq k$$

Step - 2 : re - compute μ_k to be the mean of the class C_k

$$\mu_k = \sum_{x_n \in C_k} x_n \Big/ \sum_{x_n \in C_k} 1$$

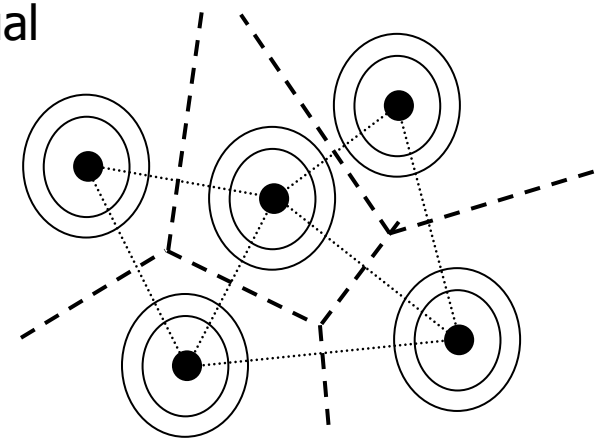
Note similarity to EM algorithm for MoGs (however K-means is usually much faster)

A graphical example



K - means MoG link

We can make the K-means/MoG link more explicit. Recall that Isotropic Gaussians (covariance = $\sigma^2 \mathbf{I}$) with equal weights (class priors) → templates model



E-step:

Step 1 = full classification (as opposed to weighting with probabilities) → $\boldsymbol{P_{n,k}} = \mathbf{0}$ or $\mathbf{1}$.

This is only true if $\sigma^2 \rightarrow \mathbf{0}$

M-step:

The only parameters to be updated are the means μ_k. So this is identical to the MoG M-step.

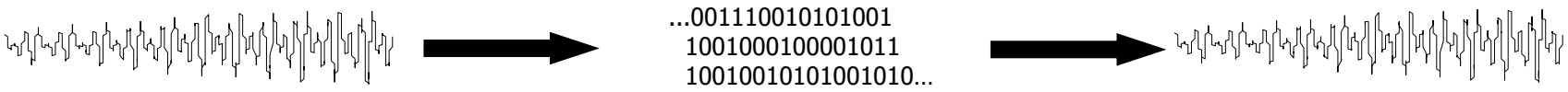
So K-means ~ MoG with isotropic covariances $\boldsymbol{\Sigma_k} \rightarrow \boldsymbol{0}$ and equal class priors.

K - means for VQ

MoG interpretation is only one way to view K-means. For one particular problem it turns out to be optimal:

> ***Problem:*** *What is the best (MMSE) way to quantize vectors of i.i.d. data using N bits per sample?*

This is called Vector Quantization (VQ). It is the basis for state-of-the-art speech codecs.



Vector Quantization involves two stages:

1. Dividing vector space into 2^N discrete regions (i.e. classification!)
2. Identifying a reconstruction value for each region

Optimality of K - means (1)

We can now show that the two steps of K-means are optimal in terms of approximating the MMSE

Optimality is easy to show in two steps:

1.The error for reconstructing $\mathbf{x_n}$ with vector $\mu_\mathbf{k}$ is simply $||\mathbf{x_n}- \mu_\mathbf{k}||$
2. Hence the MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{n}\left\|x_n - \mu_{i_n}\right\|^2$$

Where $\mathbf{i_n}$ indicates represents the cluster, $\mathbf{C_k}$, to which $\mathbf{x_n}$ belongs. Thus to minimise MSE with respect to classifications we choose $\mathbf{i_n}$ such that $||\mathbf{x_n}- \mu_\mathbf{k}||^2$ is smallest (i.e. we choose the closest centre).

Optimality of K - means (2)

2. Having assigned the data to individual clusters $\mathbf{C_k}$ what is the best vector representation in terms of MMSE?

   We can examine this for each cluster individually:

$$\text{cluster error} = \sum_{x_n \in C_k} \left\| x_n - \mu_k \right\|^2$$

   We therefore need to select $\mu_{\mathbf{k}}$ to minimise this error. This is the same as identifying the mean of the cluster:

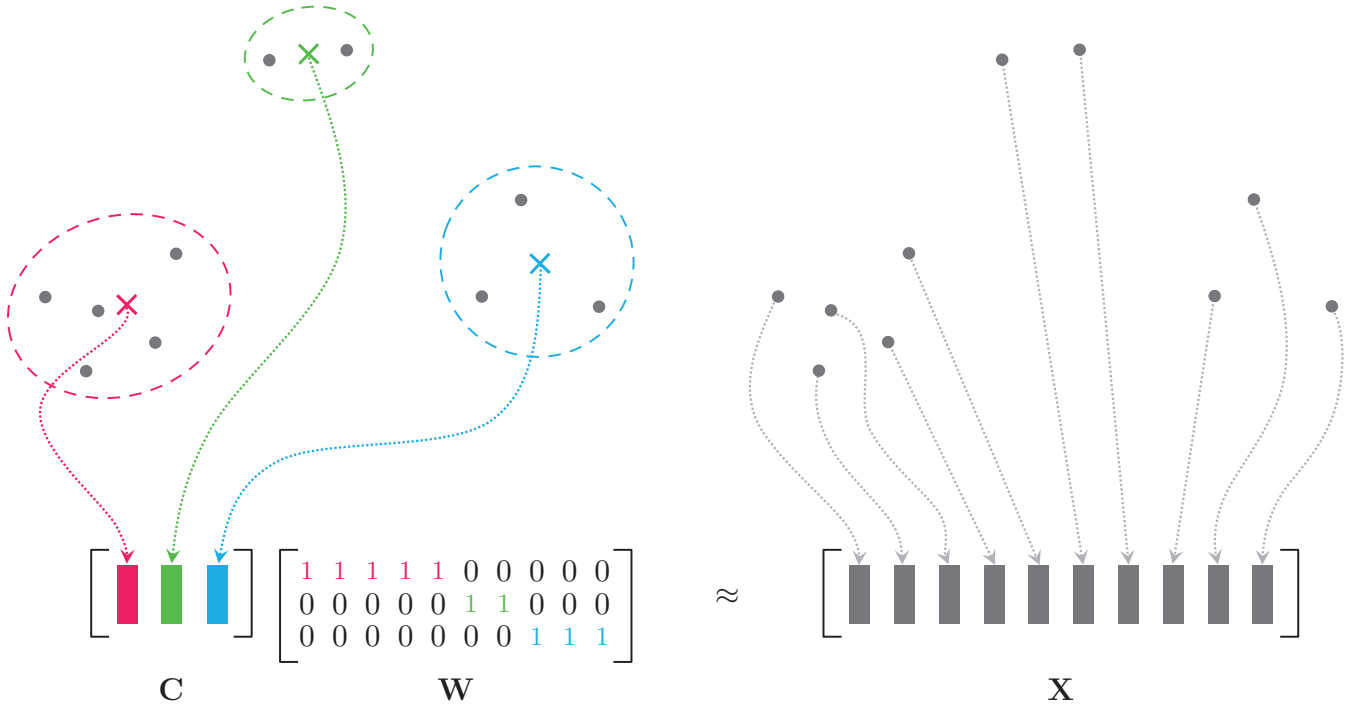
$$\text{MMSE} \rightarrow \mu_k = \sum_{x_n \in C_k} x_n \Big/ \sum_{x_n \in C_k} 1$$

Note K-means only guaranteed to find local minima.

A decomposition view of k-means

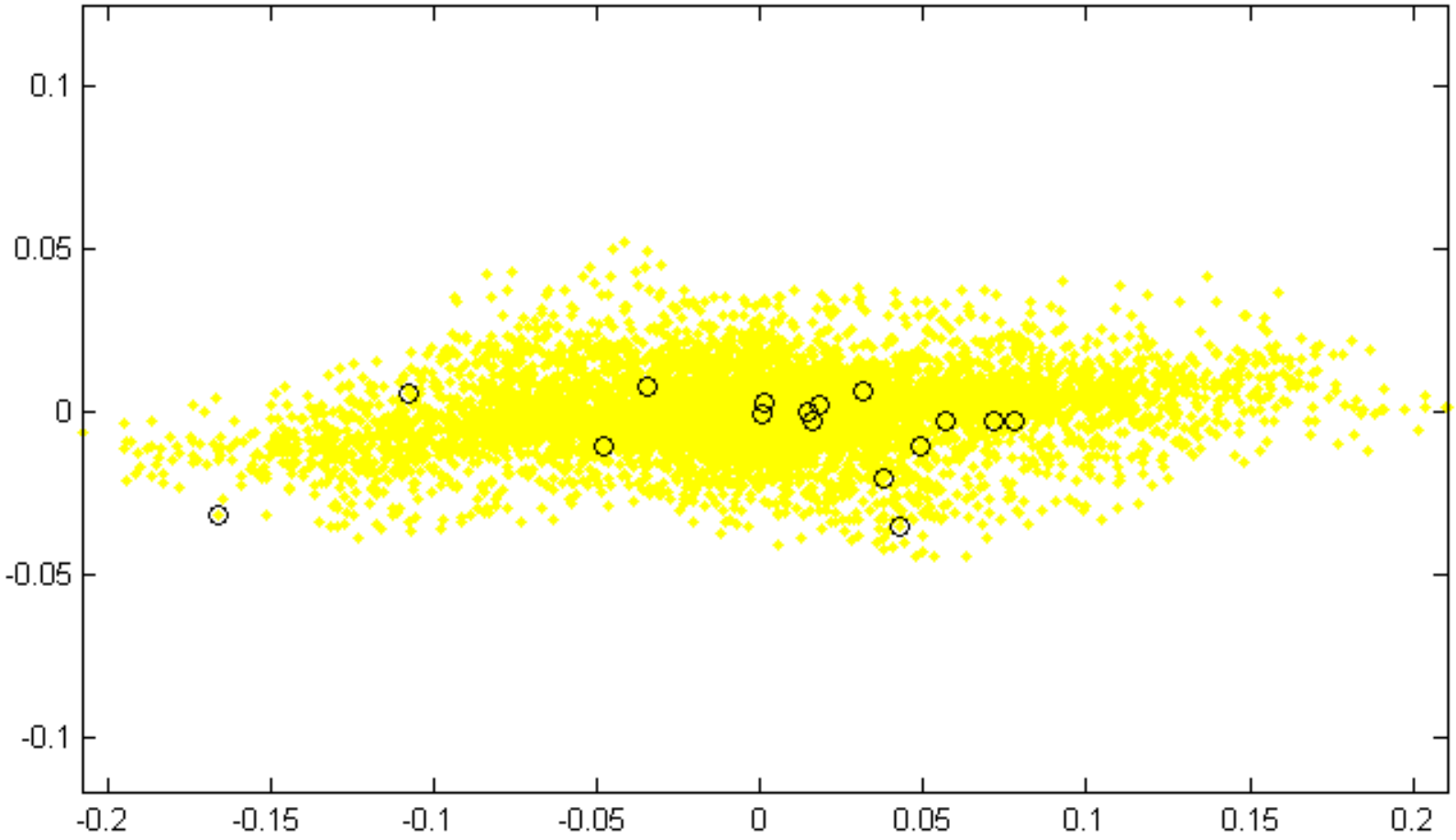
- Another way to see the k-means is as a matrix decomposition problem.
  - I want to decompose data matrix X s.t. X~CW, where C and W are special matrices, such that C contains the "centroids" and W is a sparse matrix of only zeros and ones.
- So one can see that the optimization problem is
  - find C,W that minimize $|X-CW|^2$ given specific sparsity constraints on W. What are these constraints?
- But is not easy to solve. Why?
- It can be solved in alternating fashion (keeping one matrix fixed). Then it can be shown (relatively easy) that if one keeps W fixed the solution to C is the second step of the K-means and vice versa if one keeps C fixed the best W is found by the first step of the K-means.
- So in this principle the k-means steps optimize (in alternating fashion) this matrix decomposition.
- We will see later on that also PCA, ICA can be cast as decomposition problems.

A decomposition view of k-means



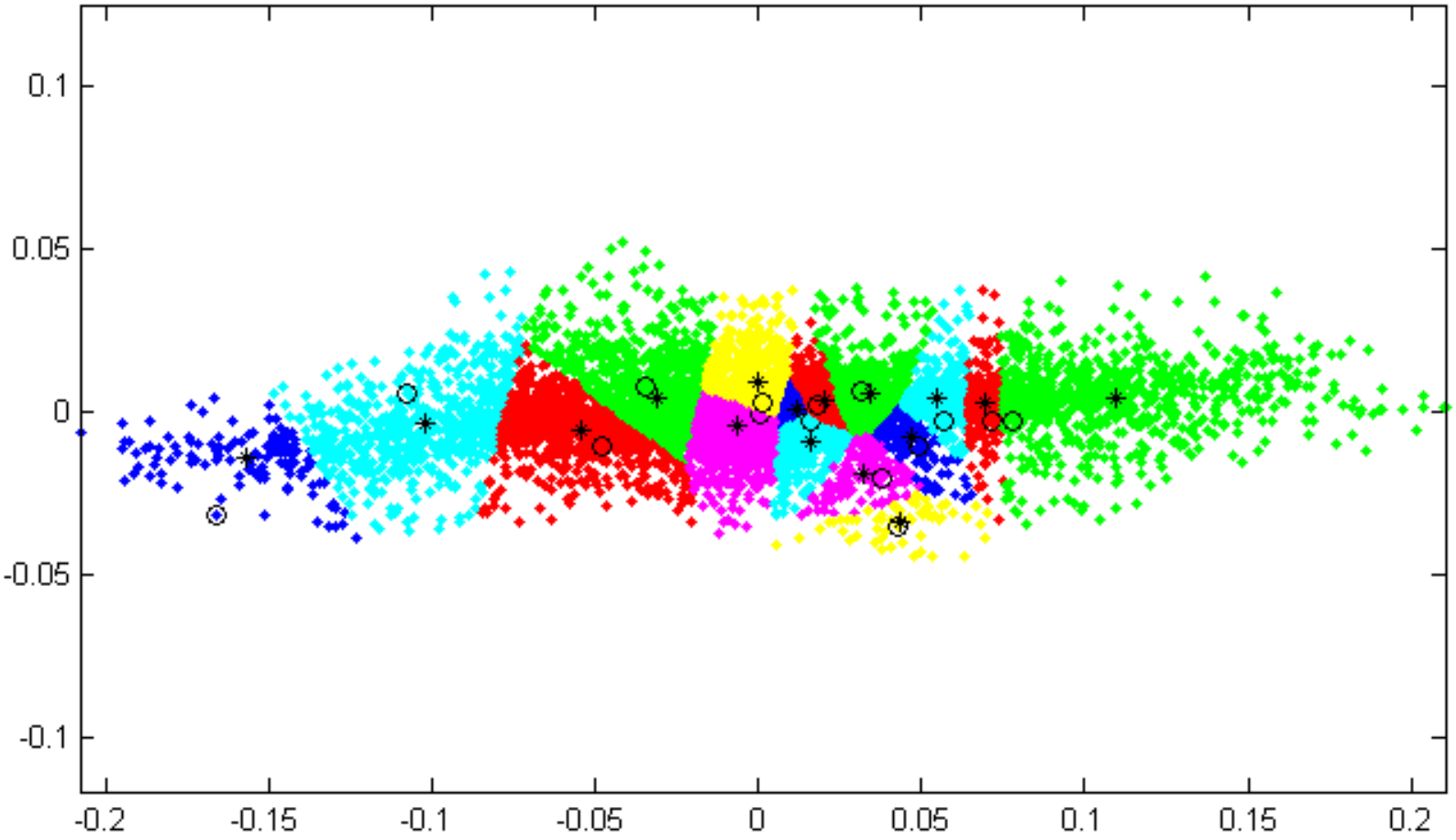
K-means clustering relations described in a compact matrix form. Cluster centroids in **C** lie close to their corresponding cluster points in **X**. The *p*th column of the assignment matrix **W** contains the standard basis vector corresponding to the data point's cluster centroid.

K - means: initialisation

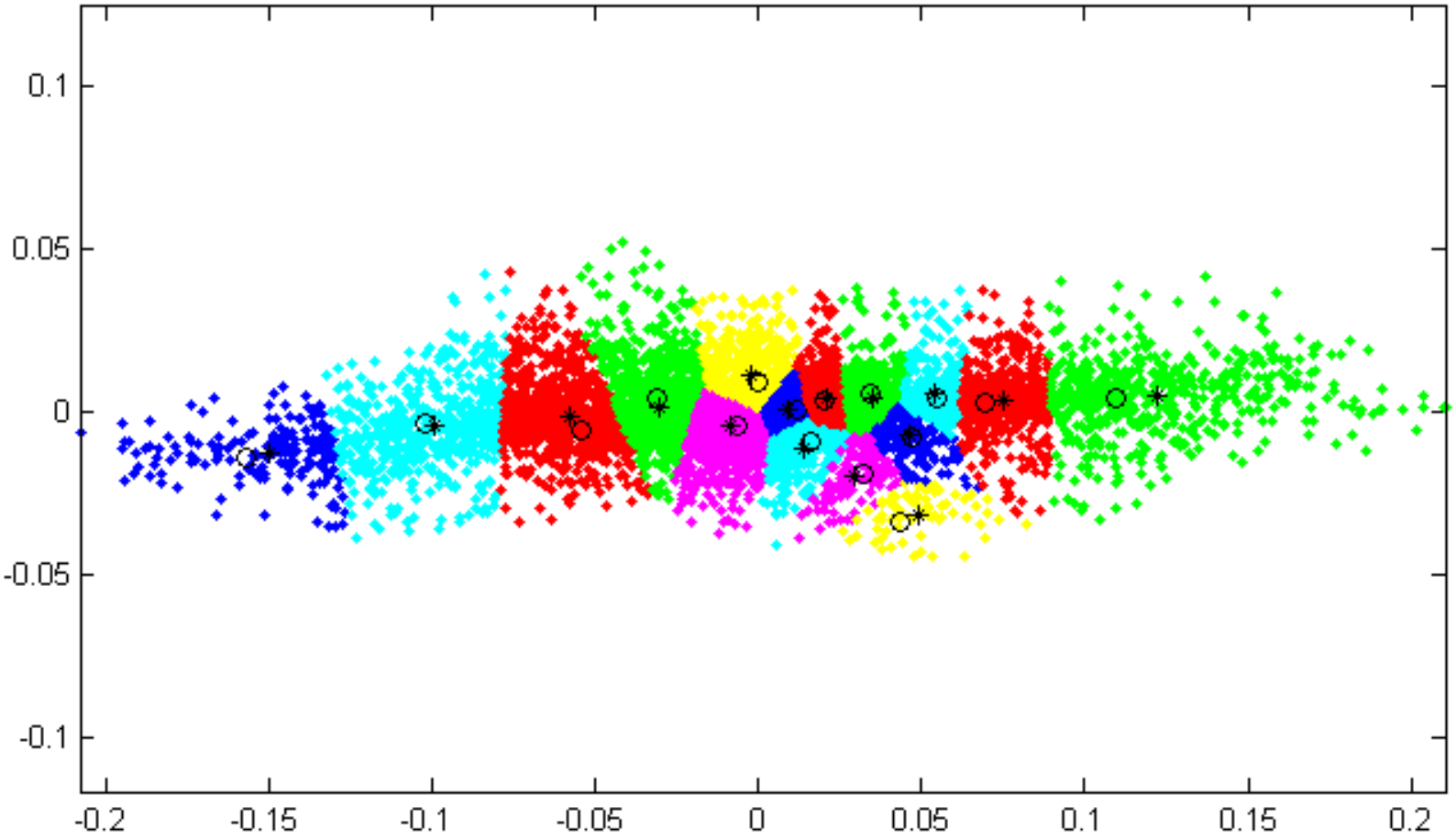


We have used random initialisation – more sophisticated schemes exist

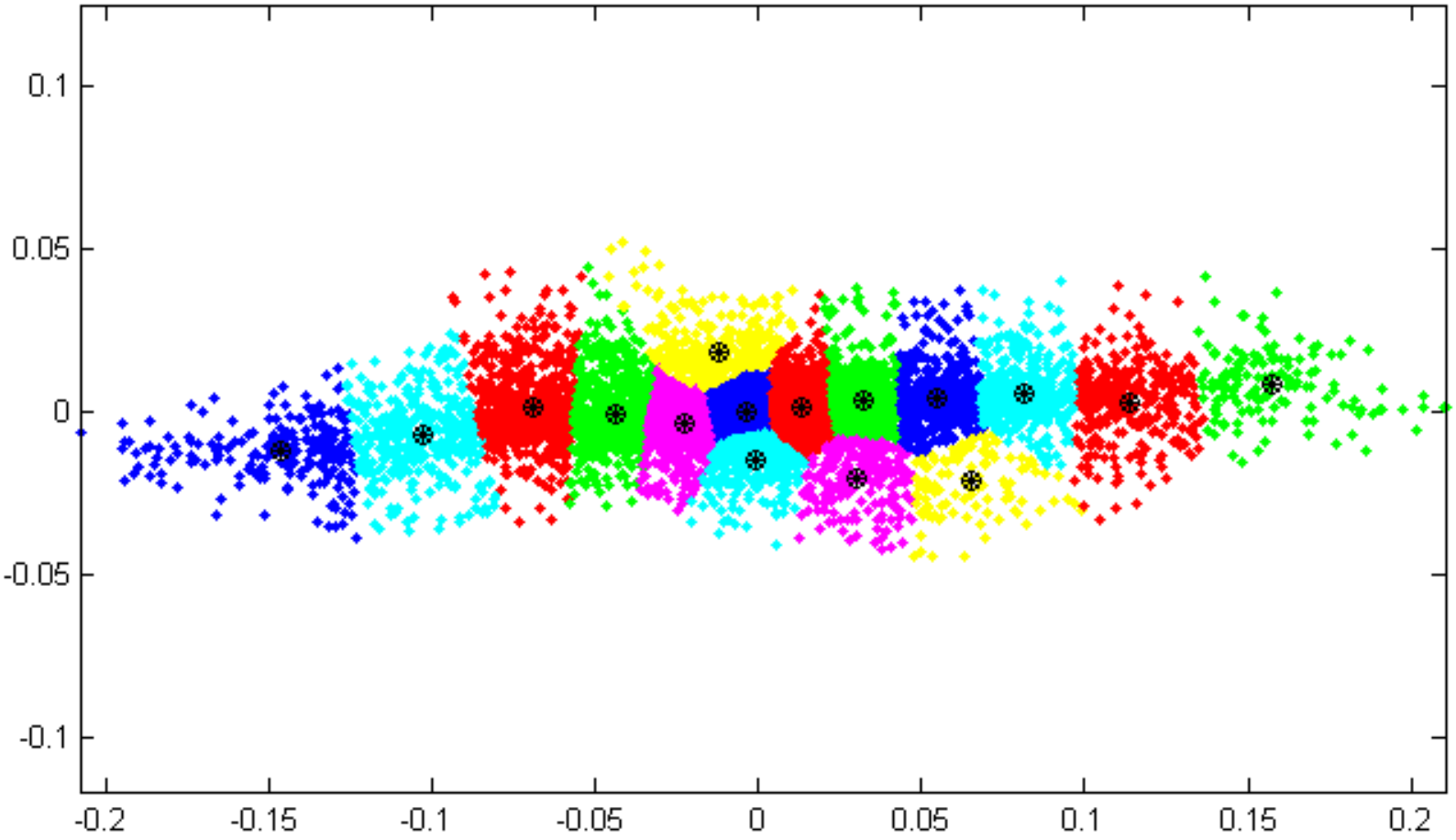
K - means: 1st iterate



K - means: 2nd iterate



K - means: converged (20th iterate)

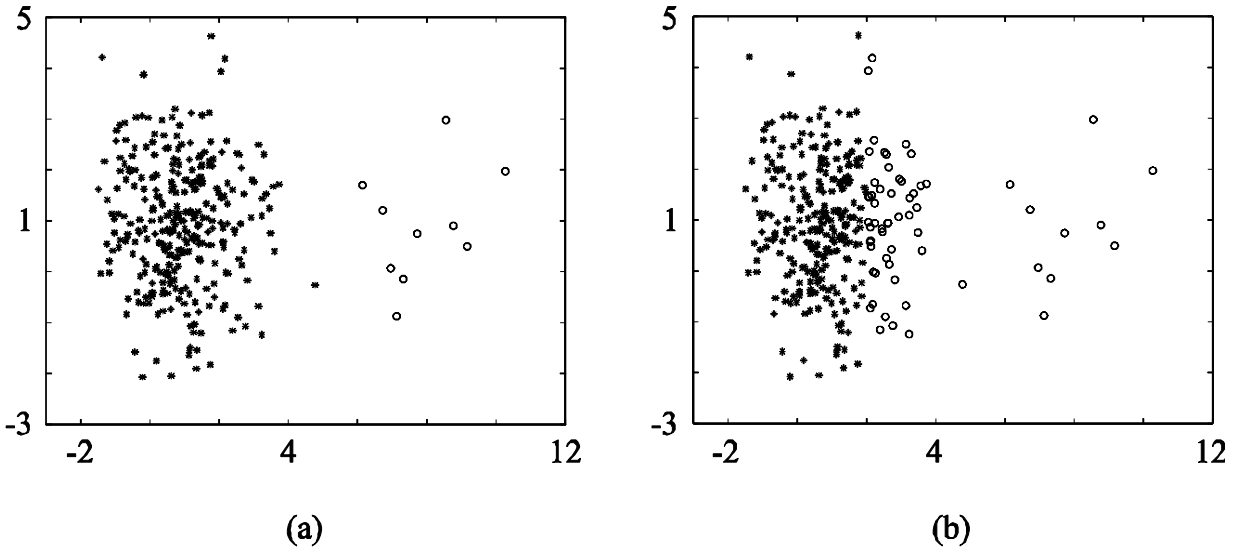


- k-means
  - An Example: (i) Consider two 2-dimensional Gaussian distributions $N(\underline{\mu}_1,\Sigma_1)$, $N(\underline{\mu}_2,\Sigma_2)$, with $\underline{\mu}_1=[1,\ 1]^T$, $\underline{\mu}_2=[8,\ 1]^T$, $\Sigma_1=1.5I$ and $\Sigma_2=I$. (ii) Generate 300 points from the 1st distribution and 10 points from the 2nd distribution. (iii) Set $k=2$ and initialize randomly *the initial centroids*.

    After convergence the large group has been split into two clusters.

    Its right part has been assigned to the same cluster with the points of the small group (see figure below).

    This indicates that k-means cannot deal accurately with clusters having significantly different sizes.



k-means (cont)

- Remarks:
  - Local minimum of cost function (result depends on initial choice)
  - k-means recovers compact clusters.
  - The computational complexity of the k-means is $O(Nmq)$, where q is the number of iterations required for convergence. In practice, m and q are significantly less than N, thus, k-means becomes eligible for processing large data sets.

- Drawbacks:

  Some drawbacks of the original k-means accompanied with the variants of the k-means that deal with them are discussed next.

k-means (cont)

- Drawback 1: *Different initial partitions may lead k-means to produce different final clusterings, each one corresponding to a different local minimum.*
  - By design of the algorithm and the alternating optimization

Example strategies for addressing drawback 1:

- Single run methods
  - Partition randomly the data set into m subsets and use their means as initial estimates for $\underline{\theta_j}'$ s.
  - Use prior knowledge
- Multiple run methods [resemble ensemble methods]
  - Create different partitions of X, run k-means for each one of them and select the best result.
  - Compute the representatives iteratively, one at a time, by running k-means mN times. It is claimed that convergence is independent of the initial estimates of $\underline{\theta_j}'$ s.

k-means (cont)

- Drawback 2: *Knowledge of the number of clusters m is required a priori.*
  - By design of the algorithm.

  Strategies for addressing drawback 2:
  - Employ splitting, merging and discarding operations of the clusters resulting from k-means.
  - Estimate m as follows:
    - Plot MSE versus the number of clusters and identify the largest plateau in the graph and set m equal to the value that corresponds to this plateau.

k-means (cont)

- Drawback 3: *k-means is sensitive to outliers and noise.*
  - This happens because a data point far away from the cluster contributes a lot on the average error due to large distance.

  Strategies for addressing drawback 3:
  - Discard all "small" clusters (they are likely to be formed by outliers).
  - Use a k-medoids algorithm, where a cluster is represented by one of its points.
- Drawback 4: *k-means is not suitable for data with nominal (categorical) coordinates.*
  - This happens mostly because while it may be possible to devise customized distance functions (albeit not rigorous), calculating averages over categorical variables is not readily possible.

  Strategies for addressing drawback 4:
  - Use a k-medoids algorithm.

The k-Medoids algorithm

- *k-Medoids Algorithm*
  - Each cluster is represented by a vector selected among the elements of X (medoid).
  - A cluster contains
    - Its medoid
    - All vectors in X that
      - Are not used as medoids in other clusters
      - Lie closer to its medoid than the medoids representing other clusters.

Let Θ be the set of medoids of all clusters, I_{Θ} the set of indices of the points in X that constitute Θ and $I_{X-\Theta}$ the set of indices of the points that are not medoids.

- Obtaining the set of medoids Θ that best represents the data set, X is equivalent to minimizing the following cost function

k-Medoids Algorithm (cont)

The PAM (partitioning around medoids) algorithm (in principle):

1. Initialize: randomly select k of the N data points as the medoids
2. Assign each data point to closest medoid (e.g., distance)
3. For each medoid m
   - For each non-medoid data point $I_{X\text{-}\Theta}$
     - Swap m and x
     - compute the total cost of the configuration
4. Select the configuration with the lowest cost

Loop between 2 and 4

- Representing clusters with mean malues vs representing clusters with medoids

Mean Values	Medoids
1. Suited only for continuous domains	1. Suited for either cont. or discrete domains
2. sensitive to outliers	2. less sensitive to outliers
3. The mean possess a clear geometrical and statistical meaning	3. No clear geometrical meaning
4. not computationally demanding	4. more computationally demanding