Lecture 11

Non-parametric unsupervised learning

- all kinds of clustering

Parametric unsupervised learning

- Mixture models
- EM

Supervised learning

To date have focused on learning classifiers from the following type of data:

Training examples consist of a *pair* of variables $(\mathbf{x}, y)$ where

$\mathbf{x}$ a feature vector **AND** y the feature vector's label

This is called **supervised learning** since the system is given **BOTH** the feature vector and its correct label.

Unsupervised learning

Today we will investigate **unsupervised learning** methods that operate on unlabeled data.

> Given a collection of feature vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ without class labels y_i, unsupervised methods attempt to build a model that captures the structure of the data.

Unsupervised learning

Such methods, if powerful enough, are extremely useful as

- Labeling large data sets can be a costly procedure.
- Class labels may not be known beforehand (i.e., data mining)
- Large datasets can be compressed by finding a small set of prototypes.
- One can train with large amount of unlabeled data, and then use supervision to label the groupings found.
- Unsupervised methods can be used for feature extraction.
- Exploratory data analysis can provide insight into the nature or structure of the data.

Classification of unsupervised learning methods

Non-parametric (clustering)

No assumptions are made about the underlying densities, instead we seek a **partition of the data into clusters**.

Parametric (mixture models)

These methods model each underlying class-conditional densities with a parametric probability density function. Thus the overall pdf of the data is a mixture of these class-conditional densities:

$$p(\mathbf{x} \mid \boldsymbol{\Theta}) = \sum_{i=1}^{c} p(\mathbf{x} \mid \boldsymbol{\theta}_i, \omega_{\mathrm{x}} = i)\, P(\omega_{\mathrm{x}} = i)$$

Objective is to find the model parameters.

Non-parametric unsupervised learning

Basic properties of non-parametric unsupervised learning

- **No density functions** are considered in these methods.
- Concerned with finding natural groupings (clusters) in a dataset.

What is a cluster?

A cluster is a number of similar objects collected or grouped together.

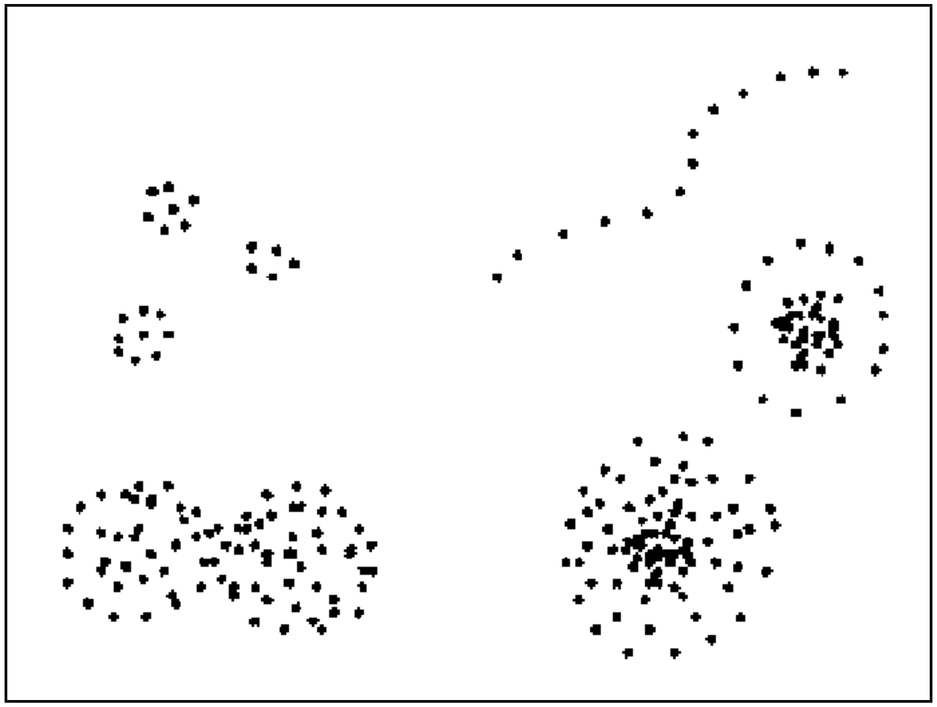
Other definitions of clusters (from Jain and Dubes, 1988)

1. A cluster is a set of entities which are alike and entities from different clusters are not alike.

2. A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.

3. Clusters are connected regions of a multidimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

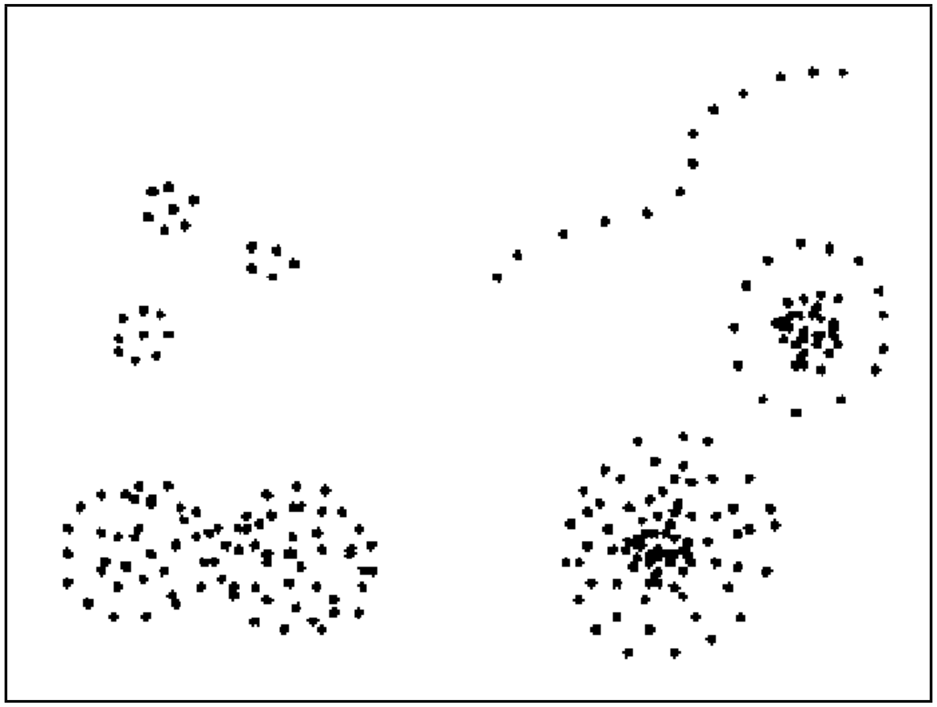
Clustering



How many clusters do you see in this figure ?

Clustering

Remember: Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes.



How many clusters do you see in this figure ?

Clustering

Clustering algorithms can be divided into several groups

- *Exclusive* **Vs** *Non-exclusive*

  Exclusive — each pattern belongs to only one cluster,
  Non-exclusive — each pattern can be assigned to several clusters.

- *Hierarchical* **Vs** *Partitional*

  Hierarchical — nested sequence of partitions,
  Partitional — a single partition

Implementations of clustering algorithms can also be grouped

- *Agglomerative* **Vs** *Divisive*

  Agglomerative — merging atomic clusters into larger clusters,
  Divisive — subdividing large clusters into smaller ones.

- *Serial* **Vs** *Simultaneous*

  Serial — processing patterns one by one,
  Simultaneous — processing all patterns at once.

- Graph-theoretic **Vs** Algebraic

  Graph-theoretic — based on connectedness,
  Algebraic — based on error criteria.

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.
- Most of these algorithms are based on the following two popular techniques:
  - Iterative squared-error partitioning, k-means,
  - Agglomerative hierarchical clustering.
- One of the main challenges is to select an appropriate measure of similarity to define clusters that is often both data (cluster shape) and context dependent.

Clustering

Non-parametric clustering involves three steps:

- Defining a **measure of (dis)similarity** between examples
- Defining a **criterion function** for clustering
- Defining an **algorithm to minimize** (or maximize) the criterion function

Similarity measures

- The most obvious measure of similarity (or dissimilarity) between two patterns is the distance between them.
- If distance is a good measure of dissimilarity, then we can expect the distance between patterns in the same cluster to be significantly less than the distance between patterns in different clusters.
- In this case, a very simple way of doing clustering is to choose a threshold on distance and group the patterns that are closer than this threshold.

Criterion functions

- The next challenge after selecting the similarity measure is the choice of the criterion function to be optimized.
- Suppose that we have a set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n samples that we want to partition into exactly k disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
- Each subset represents a cluster, with samples in the same cluster being somehow more similar to each other than they are to samples in other clusters.
- The simplest and most widely used criterion function for clustering is the sum-of-squared-error criterion.

Squared-error partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $\mathcal{D}_1, \ldots, \mathcal{D}_k$.
- Let n_i be the number of samples in $\mathcal{D}_i$ and let $\mathbf{m}_i$ be the mean of those samples.

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}.$$

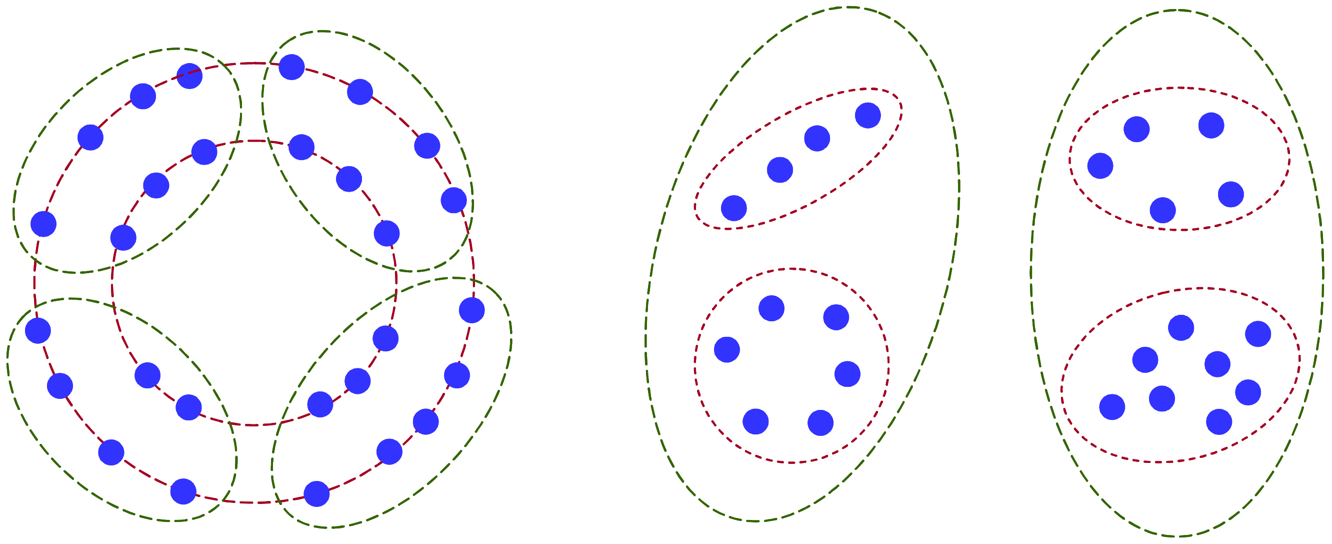
- Then, the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{D}_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

- For a given cluster $\mathcal{D}_i$, the mean vector $\mathbf{m}_i$ (centroid) is the best representative of the samples in $\mathcal{D}_i$.

Cluster validity

The choice of (dis)similarity measure and criterion function has a major impact on the final clustering produced by the algorithms.



The validity of the final cluster solution is highly subjective.

This contrasts with supervised training, there is a clear objective function - Bayes' Risk.

Iterative optimization

Given a criterion function, how do we find the partition of the data set that minimizes it ?

- Exhaustive enumeration of all partitions, which guarantees the optimal solution, is infeasible.

  For example, a problem with 5 clusters and 100 examples yields 10^{67} possible partitions.

Iterative optimization

The common approach is to proceed in an iterative fashion.

1. Find some reasonable initial partition and then
2. Move samples from one cluster to another in order to reduce the criterion function

These iterative methods produce sub-optimal solutions but are computationally tractable.

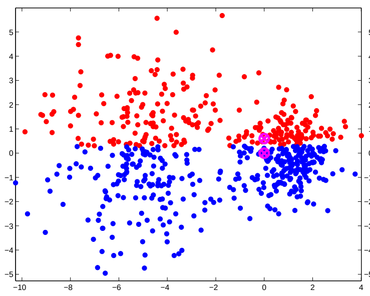
The k-means algorithm

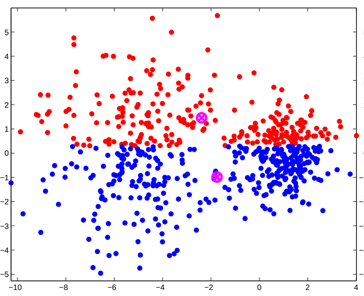
The k-means algorithm is a simple clustering procedure that attempts to minimize the squared-error function J_e in an iterative fashion.

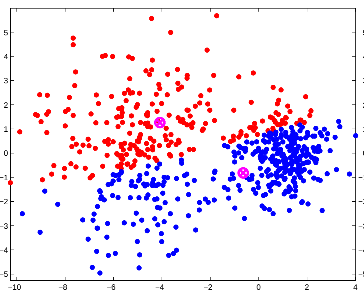
k-means algorithm

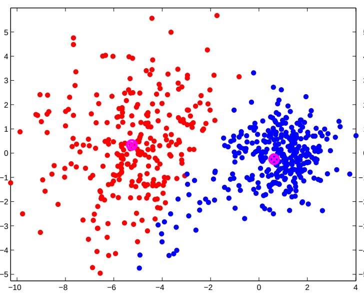
1. Define, k, the number of clusters.
2. Initialize the clusters by
   - arbitrary assignment of examples to clusters or
   - arbitrary set of cluster centers
3. Compute the sample mean of each cluster.
4. Reassign each example to the cluster with the nearest mean.
5. If the classification of all samples has not changed, stop, else go to step 3.

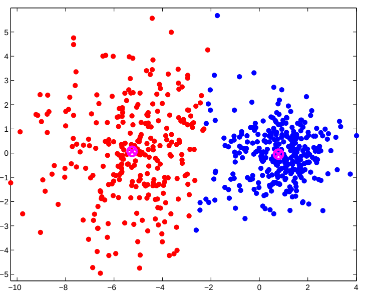
The k-means algorithm example

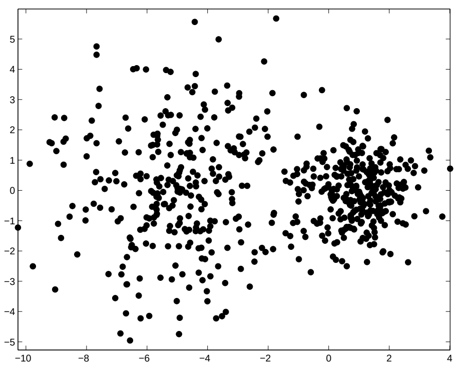

iteration 0

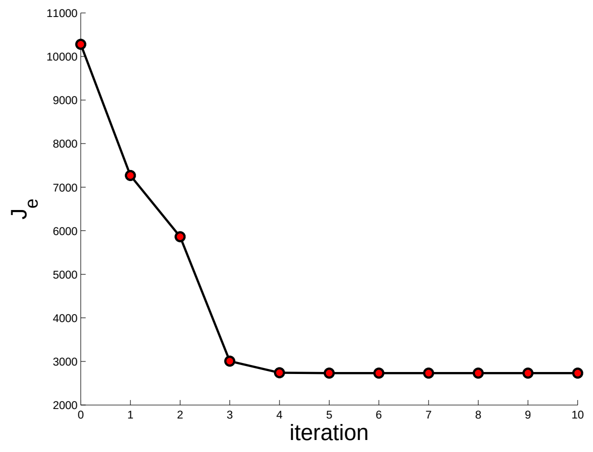

iteration 1


iteration 2


iteration 3


iteration 4


data



Squared error after each iteration

The k-means algorithm

Pros

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape, and well-separated in the feature space.

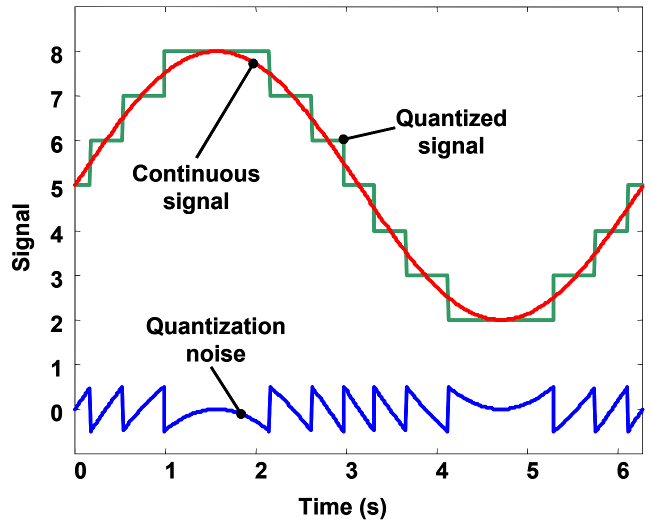
Cons

- However, choosing k and choosing the initial partition are the main drawbacks of this algorithm. The value of k is often chosen empirically or by prior knowledge about the data.
- Suffers from problems of local minima.

The k-means algorithm

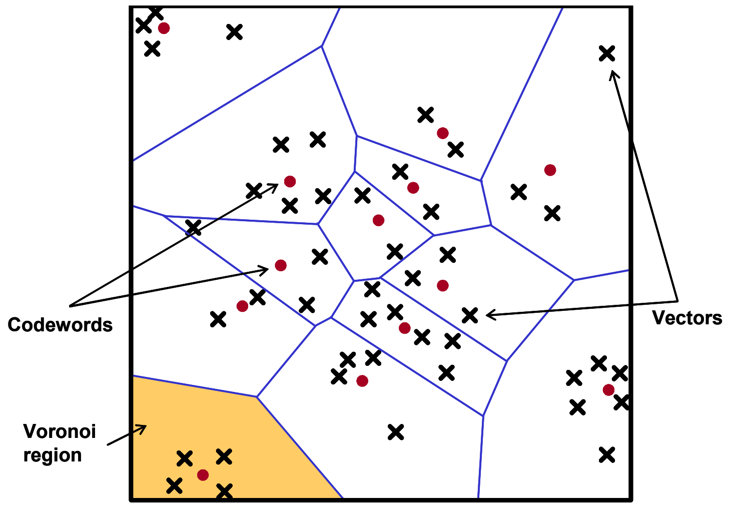
The k-means algorithm is widely used in the fields of signal processing and communication for **Vector Quantization**.

- One dimensional signal values are usually quantized into a number of levels.



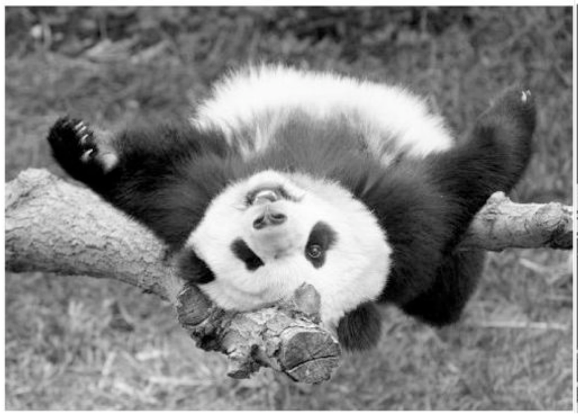
- The same idea can be extended for multiple channels
  - However, rather than quantizing each separate channel, we can obtain a more efficient signal coding if we quantize the overall multidimensional vector by finding a number of multidimensional prototypes (cluster centers)

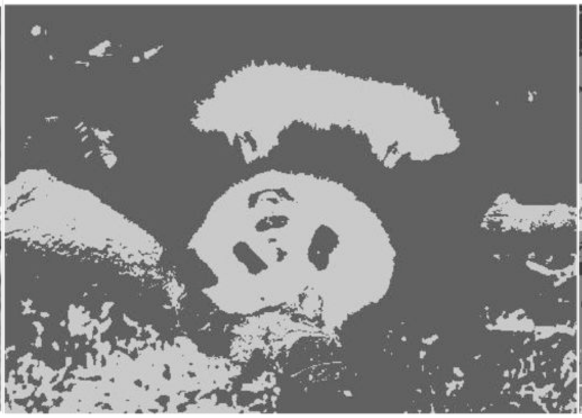
- The set of cluster centers is called a *codebook*, and the problem of finding this codebook is normally solved using the k-means algorithm.

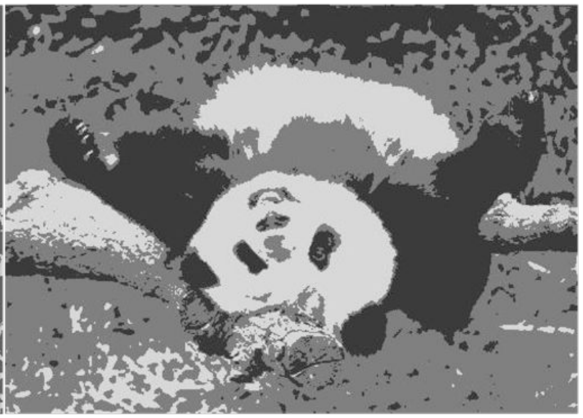


Segmentation via clustering

Take the pixel intensity data from the image. Apply k-means clustering to this data. For each pixel, set its intensity value to the mean value of the cluster it is assigned to.


Original Image

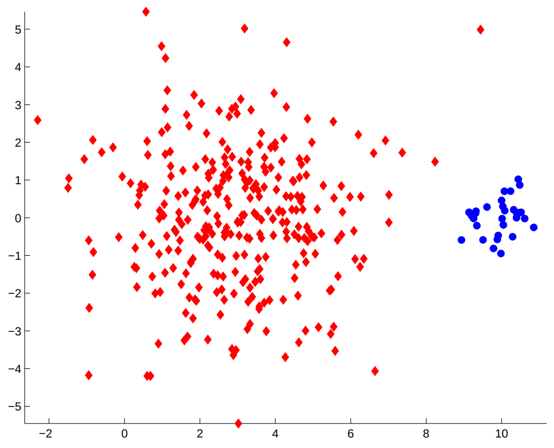

two clusters


three clusters

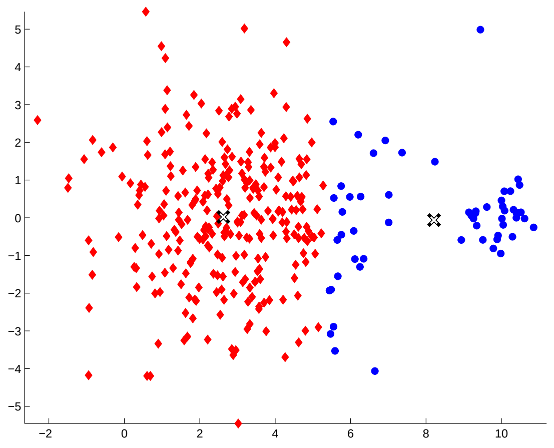
Weaknesses of K-means clustering

Can't deal effectively with clusters containing very different number of points

Ground truth clusters



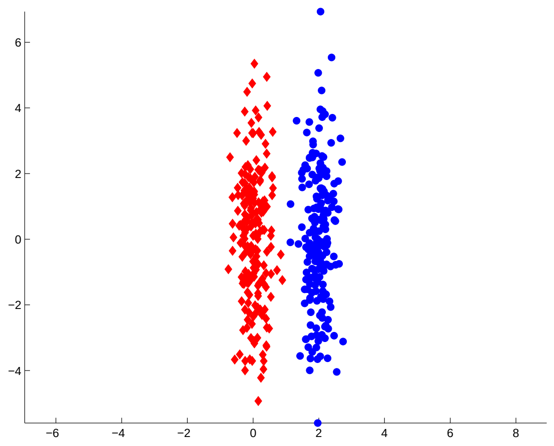
K-means clusters



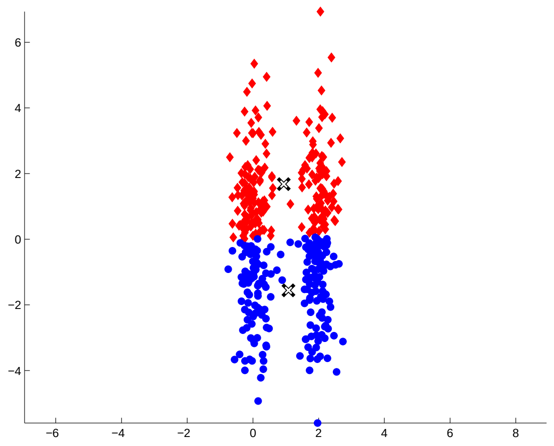
Weaknesses of K-means clustering

Can't deal effectively with elongated clusters

Ground truth clusters



K-means clusters



Weaknesses of K-means clustering

Also K-means makes **hard** assignments instead of **soft** assignments. Therefore points on the boundary have as much influence as those in the centre on the calculation of the mean.

Can **upgrade** K-means clustering to have soft assignment....

Soft K-means clustering I

Assignment Step

Each data point $\mathbf{x}_i$ is given a soft *degree of assignment* to each of the means.

Call the degree to which $\mathbf{x}_i$ is assigned to cluster k the responsibility $r_i^{(k)}$ of cluster k for point i:

$$r_i^{(k)} = \frac{\exp(-\beta\ d(\boldsymbol{\mu}_k, \mathbf{x}_i))}{\sum_{k'} \exp(-\beta\ d(\boldsymbol{\mu}_{k'}, \mathbf{x}_i))}$$

where $d(\boldsymbol{\mu}_k, \mathbf{x}_i)$ is the Euclidean distance.

Note:

$r_i^{(k)}$ is between 0 and 1.

What happens when $\beta \to \infty$?

Soft K-means clustering I

Update Step

The means are adjusted to match the sample means of the data points they are responsible for

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_i^{(k)} \mathbf{x}_i}{R^{(k)}}$$

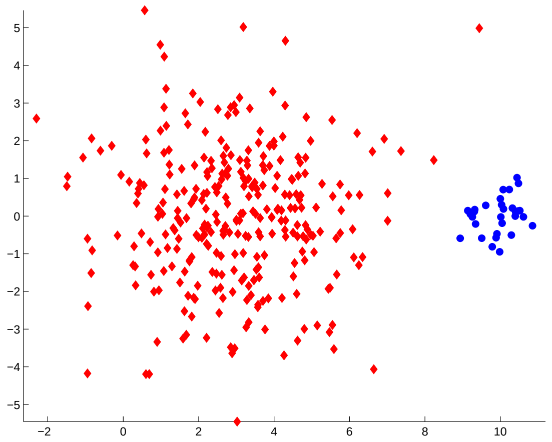
where $R^{(k)}$ is the total responsibility of mean k

$$R^{(k)} = \sum_{i=1}^{n} r_i^{(k)}$$

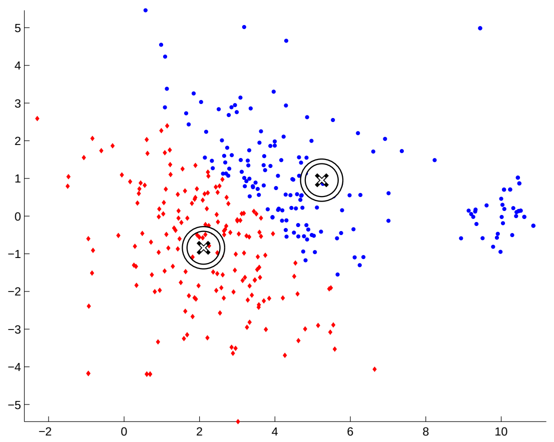
Soft K-means clustering I

Let $\sigma \equiv \frac{1}{\sqrt{\beta}}$ then the assignment function can be considered as modelling each cluster as a multi-variate Gaussian with covariance matrix proportional to $\sigma^2 I$.

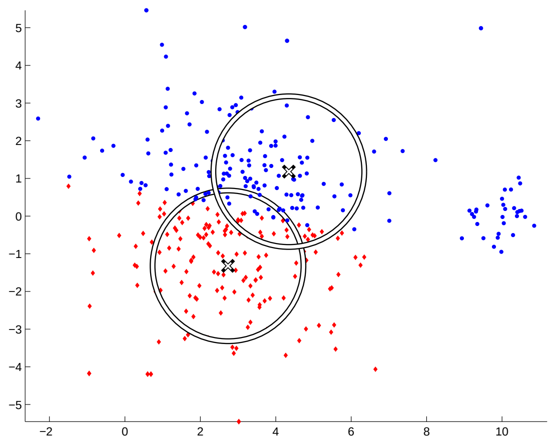
Ground truth clusters



Soft K-means $\sigma = .5$



Soft K-means $\sigma = 2$



Can **upgrade** algorithm again to take into account that each cluster will have a different shape and number of elements....

Soft K-means clustering II

Assignment Step

The *responsibilities* are

$$r_i^{(k)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})}$$

Update Step

Each cluster's parameters

- $\boldsymbol{\mu}_k$ mean
- π_k proportion of total points belonging to cluster k
- Σ_k covariance matrix

are adjusted to match the data points that it is responsible for

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_i^{(k)}\,\mathbf{x}_i}{R^{(k)}}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} r_i^{(k)}\,(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^t}{R^{(k)}}$$

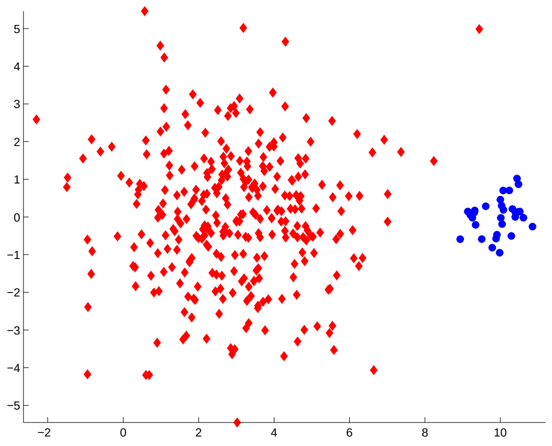
$$\pi_k = \frac{R^{(k)}}{\sum_k R^{(k)}}$$

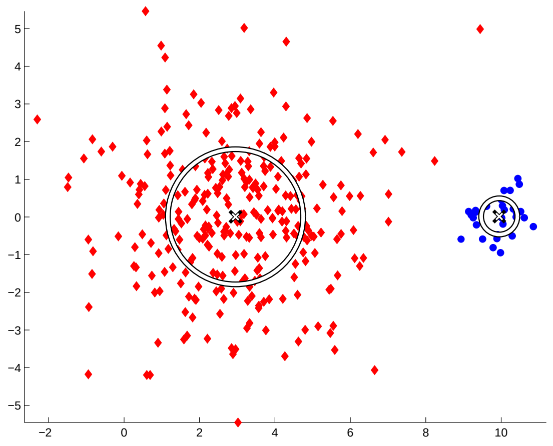
where $R^{(k)}$ is the total responsibility of mean k

$$R^{(k)} = \sum_{i=1}^{n} r_i^{(k)}$$

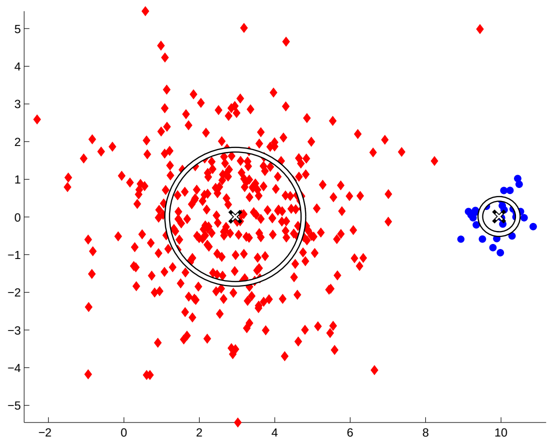
Soft K-means clustering II

Ground truth clusters **Results of soft clustering**





$\Sigma_k = \sigma_k^2 I$

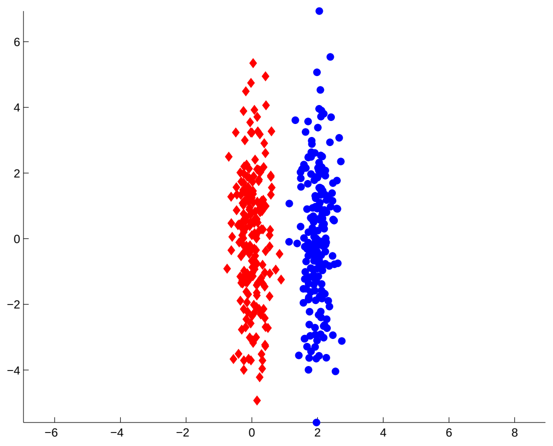


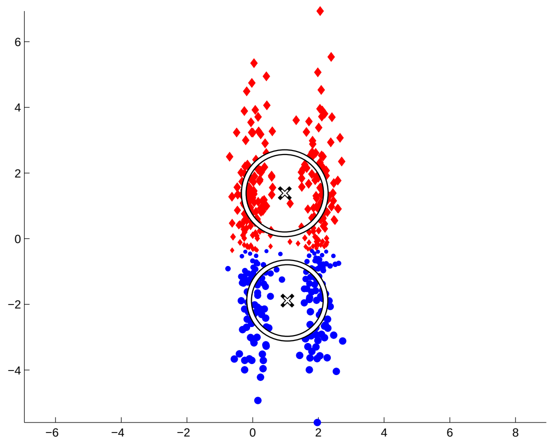
diagonal covariance matrices

Soft K-means clustering II

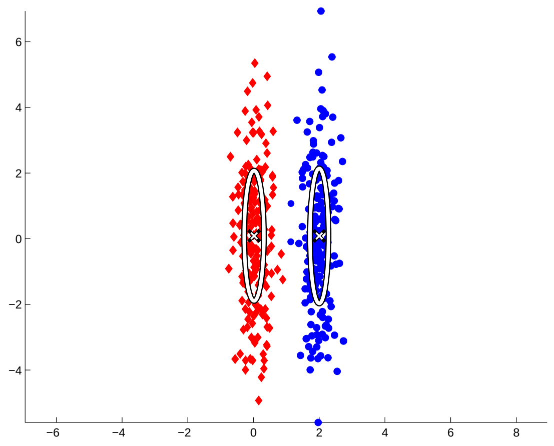
Ground truth clusters

Results of soft clustering





$\Sigma_k = \sigma_k^2 I$



diagonal covariance matrices

Hierarchical Clustering

- The k-means algorithm produces a flat data description where the clusters are disjoint and are at the same level.
- In some applications, groups of patterns share some characteristics when looked at a particular level.
- Hierarchical clustering tries to capture these multi-level groupings using hierarchical representations rather than flat partitions.

Hierarchical Clustering

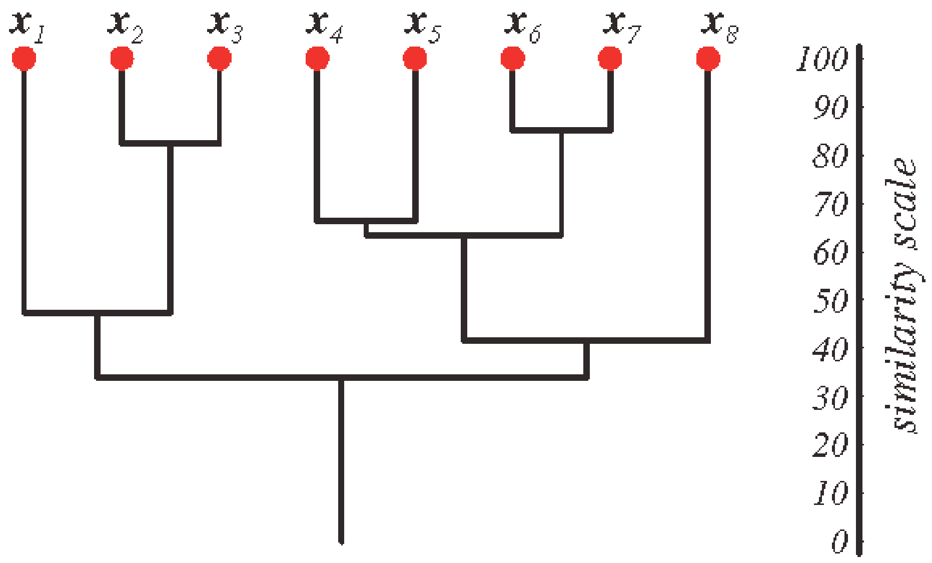
Hierarchical clustering methods can be grouped in two general classes

- **Agglomerative**
  - Also known as bottom-up or merging
  - Starting with n singleton clusters, successively merge clusters until one cluster is left
- **Divisive**
  - Also known as top-down or splitting
  - Starting with a unique cluster, successively split the clusters until n singleton examples are left.

Dendrograms

The preferred representation for hierarchical clusters is the dendrogram.

- The dendrogram is a binary tree that shows the structure of the clusters.
- The vertical axis shows a generalized measure of similarity among clusters.



Divisive clustering

Divisive Clustering

Have n training examples.

1. Start with one large cluster. Set $n_c = 1$.
2. Find *worst* cluster.
3. Split it, set $n_c = n_c + 1$.
4. If $n_c < n$ go to 2.

How to choose the *worst* cluster Largest number of examples, largest variance, largest sum-squared-error, ...

How to split clusters Mean-median in one feature direction, perpendicular to the direction of largest variance,...

Agglomerative clustering

The computations required by divisive clustering are more intensive than for agglomerative clustering methods. Hence..

Agglomerative Clustering

Have n training examples.

1. Start with n singleton clusters. Set $n_c = n$.
2. Find the two *nearest* clusters.
3. Merge them, set $n_c = n_c - 1$.
4. If $n_c > 1$ go to 2.

Popular distance measures (for two clusters $\mathcal{D}_i$ and $\mathcal{D}_j$):

$$d_{\min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|,$$

$$d_{\max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} \|\mathbf{x} - \mathbf{x}'\|$$

$$d_{\mathrm{avg}}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{|\mathcal{D}_i||\mathcal{D}_j|} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} \|\mathbf{x} - \mathbf{x}'\|,$$

$$d_{\mathrm{mean}}(\mathcal{D}_i, \mathcal{D}_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

Agglomerative clustering

Minimum Distance

- When $d_{\min}$ is used to measure the distance between clusters, the algorithm is called the nearest neighbour clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the single linkage algorithm.
- This algorithm favours elongated classes.

Max Distance

- When $d_{\max}$ is used to measure the distance between clusters, the algorithm is called the farthest neighbour clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between

nearest clusters exceeds a threshold, it is called the complete linkage algorithm.

- This algorithm favors compact classes.

Average and mean distance

- The minimum and maximum distance are extremely sensitive to outliers since their measurement of between-cluster distance involves minima or maxima.
- The average and mean distance approaches are more robust to outliers.
- Of the two, the mean distance is computationally more attractive.

Agglomerative clustering example

Problem Perform agglomerative clustering on the following one dimensional dataset using the $d_{\min}$ metric

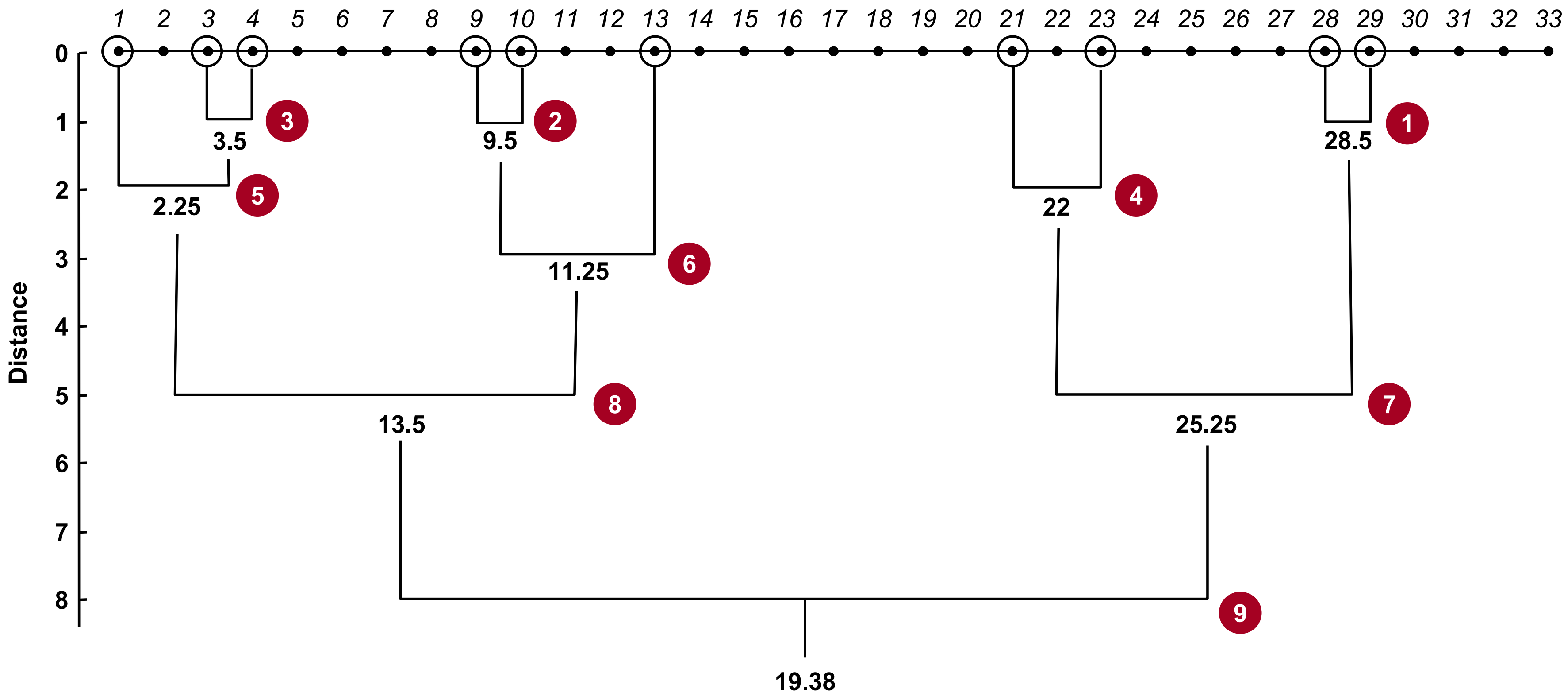
- $\mathcal{D} = \{1, 3, 4, 9, 10, 13, 21, 23, 28, 29\}$.
- In case of ties, always merge the pair of clusters with the largest mean
- Indicate the order in which the merging operations occur.

Partial Solution

- The initial sets are $\mathcal{D}_1 = \{1\}, \mathcal{D}_2 = \{3\}, \mathcal{D}_3 = \{4\}, \ldots, \mathcal{D}_{10} = \{29\}$.

- What are the i, j such that $d_{\min}(\mathcal{D}_i, \mathcal{D}_j)$ is minimum ? Note $d_{\min}(\mathcal{D}_1, \mathcal{D}_2) = 2, d_{\min}(\mathcal{D}_1, \mathcal{D}_3) = 3, \ldots$

Agglomerative clustering example



Parametric Methods

Mixture densities

A mixture model is a linear combination of m densities

$$p(\mathbf{x} \,|\, \boldsymbol{\Theta}) = \sum_{j=1}^{m} \alpha_j \, p_j(\mathbf{x} \,|\, \boldsymbol{\theta}_j)$$

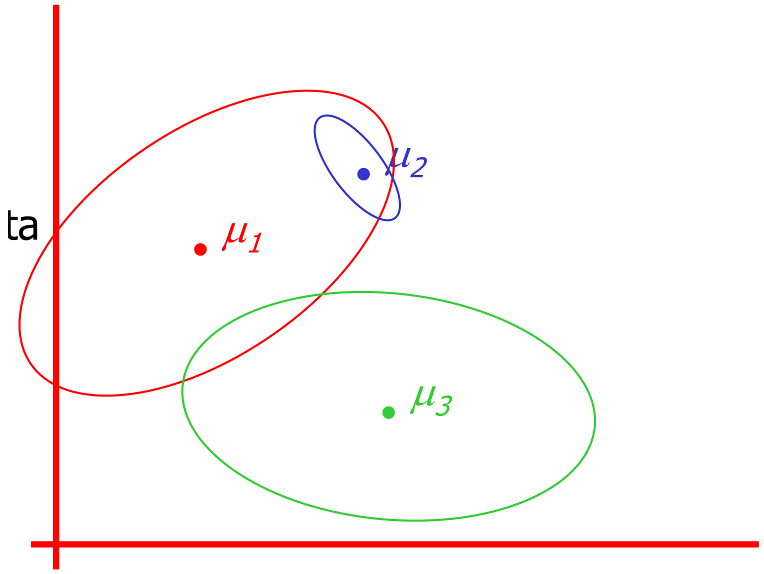
where $\boldsymbol{\Theta} = (\alpha_1, \ldots, \alpha_m, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m)$ and each $\alpha_j \geq 0$ and $\sum_j \alpha_j = 1$.

- $\alpha_1, \ldots, \alpha_m$ are called the mixing parameters.
- $p_j(\mathbf{x} \,|\, \boldsymbol{\theta}_j)$, $j = 1, \ldots, m$ are called the component densities.

Using probability density functions of this type is the alternative to the non-parametric methods. $p_j(\mathbf{x}|\boldsymbol{\theta}_j)$ is a parametric pdf (e.g., Gaussian).

The general Gaussian mixture model

- There are m components.
- The ith component has an associated mean vector $\boldsymbol{\mu}_i$ and covariance matrix Σ_i.
- Each component generates data from $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$.



Each data-point $\mathbf{x}$ is generated by:

- Choose component i with probability $p(\omega_{\mathbf{x}} = i)$.
- $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$.

Mixture model

Have n feature points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.

Imagine each example $\mathbf{x}$ is drawn from the jth class of the m different classes with probability $P(\omega_{\mathbf{x}} = j)$ and $\sum_{j=1}^{m} P(\omega_{\mathbf{x}} = j) = 1$.

$$p(\mathbf{x}) = \sum_{j=1}^{m} p(\mathbf{x}, \omega_{\mathbf{x}} = j) = \sum_{j=1}^{m} p(\mathbf{x} \,|\, \omega_{\mathbf{x}} = j)\, P(\omega_{\mathbf{x}} = j)$$

Assume the class-conditional probability density functions are multi-variate Gaussians each with its own parameters:

$$p(\mathbf{x} \,|\, \omega_{\mathbf{x}} = j) = \mathcal{N}(\mathbf{x}\,;\, \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{Z_j} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_j)^t\, \Sigma_j^{-1}\, (\mathbf{x} - \boldsymbol{\mu}_j) \right),$$

Then returning to our original definition of a mixture distribution then

$$P(\omega_{\mathbf{x}} = j) = \alpha_j \quad \text{for } j = 1, \ldots, m$$

Have to estimate from $\mathcal{X}$ these parameters and the prior class probabilities

$$(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m, \Sigma_1, \ldots, \Sigma_m, \alpha_1, \ldots, \alpha_m)$$

How to estimate the parameters of the GMM ?

Assume one has n training examples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.

From this data we want to estimate the parameters of

$$p(\mathbf{x}) = \sum_{j=1}^{m} P(\omega_{\mathbf{x}} = j)\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \Sigma_j) = \sum_{j=1}^{m} \alpha_j\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \Sigma_j)$$

How??

Maximum Likelihood Estimate ???

E.M. for general GMMs

Introduce hidden variables:

For each training example introduce hidden variables

$$\mathbf{Z} = (z_1, z_2, \ldots, z_n) \quad \text{and each } z_i \in \{1, 2, \ldots, m\}$$

indicating which component of the GMM generated each example.

Have a current estimate for the parameters:

At the t-th iteration the estimate of the GMM's parameters is:

$$\Theta^{(t)} = \{\boldsymbol{\mu}_1^{(t)}, \boldsymbol{\mu}_2^{(t)}, \ldots, \boldsymbol{\mu}_m^{(t)}, \Sigma_1^{(t)}, \ldots, \Sigma_m^{(t)}, \alpha_1^{(t)}, \ldots, \alpha_m^{(t)}\}$$

then

Iterate

E-step

Compute "expected" classes of all datapoints for each class

$$p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)}) = \frac{p(\mathbf{x}_i \mid z_i = k, \Theta^{(t)})\, p(z_i = k \mid \Theta^{(t)})}{p(\mathbf{x}_i)} = \frac{p(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{(t)}, \Sigma_k^{(t)})\, \alpha_k^{(t)}}{\sum_{j=1}^{m} p(\mathbf{x}_i \mid \boldsymbol{\mu}_j^{(t)}, \Sigma_j^{(t)})\, \alpha_j^{(t)}}$$

M-step

Compute Maximum likelihood of the parameters of the mixture model given our data's class membership distributions.

$$\mu_k^{(t+1)} = \frac{\sum_i p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)})\, \mathbf{x}_i}{\sum_i p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)})}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_i p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)})\, (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})^t}{\sum_i p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)})}$$

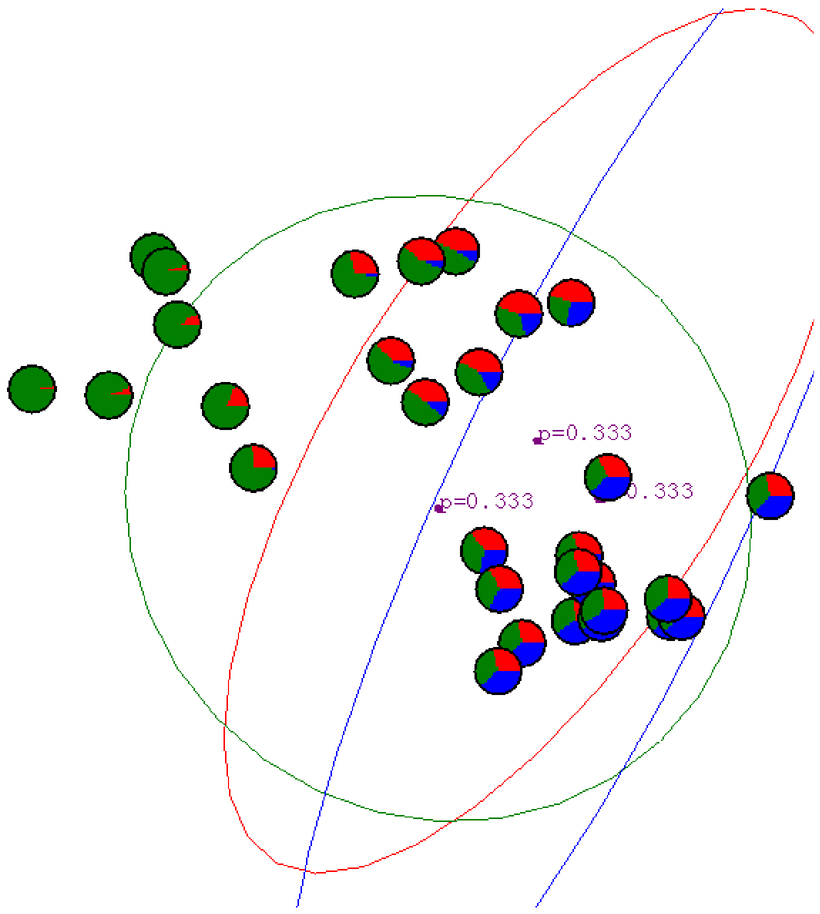
$$\alpha_k^{(t+1)} = \frac{\sum_i p(z_i = k \mid \mathbf{x}_i, \Theta^{(t)})}{n}$$

Here EM is the same as soft clustering

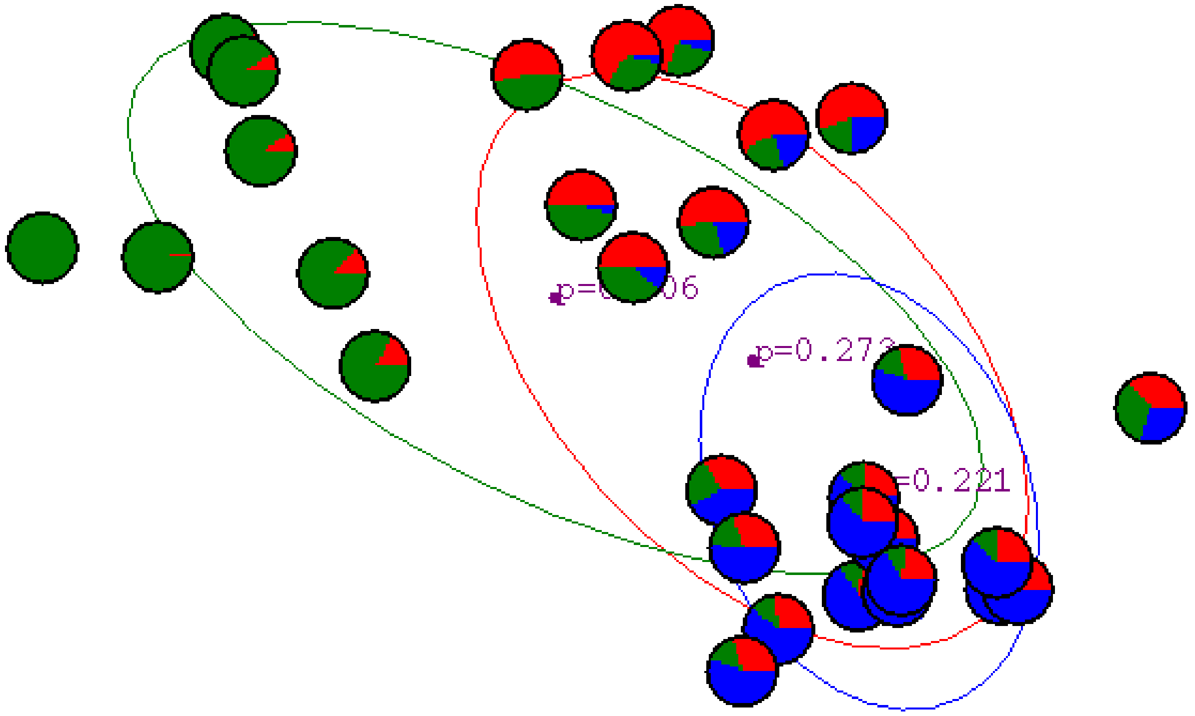
Have just exchanged the responsibilities with the posterior probabilities of the hidden variables, that is

$$r_i^{(k)} \equiv p(z_i = k \,|\, \mathbf{x}_i, \Theta^{(t)})$$

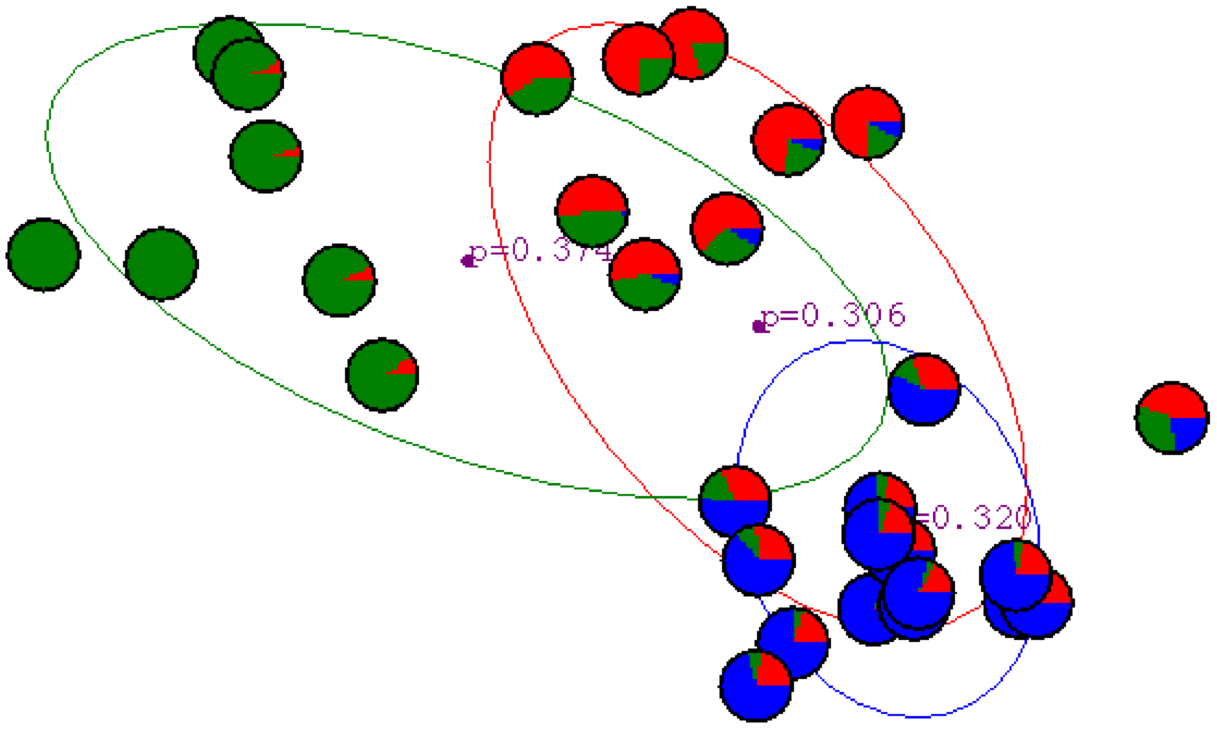
Gaussian mixture example: Start



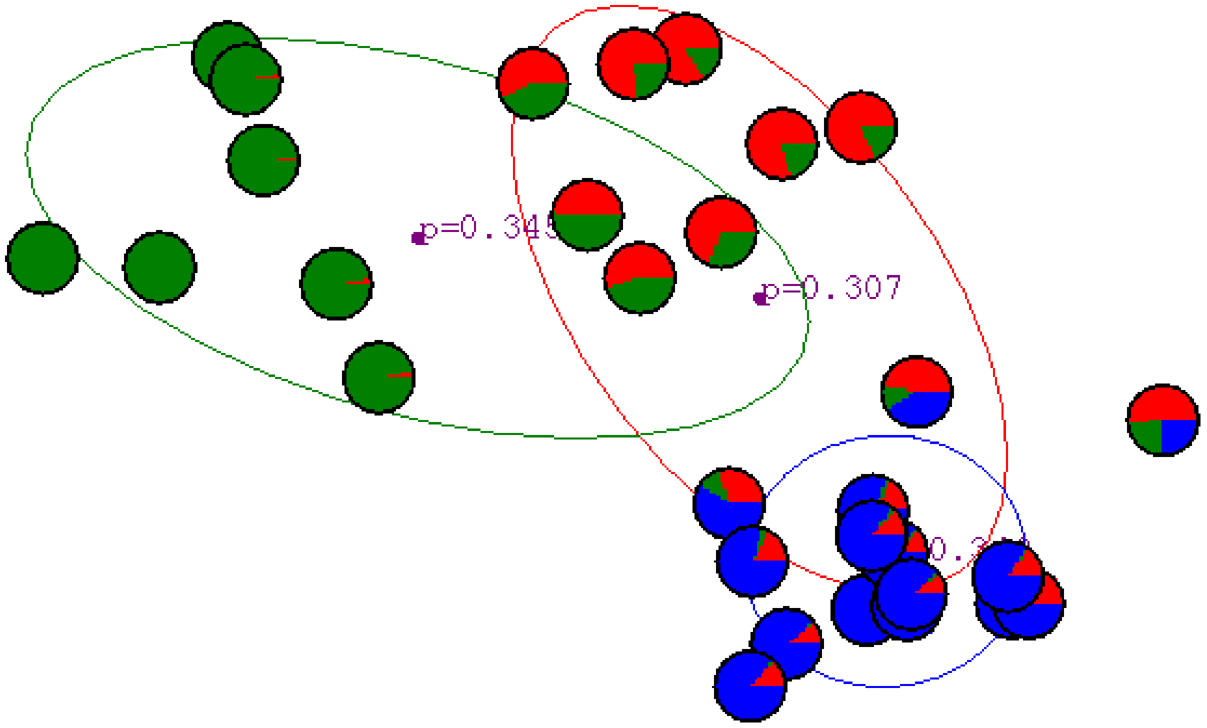
After 1st iteration



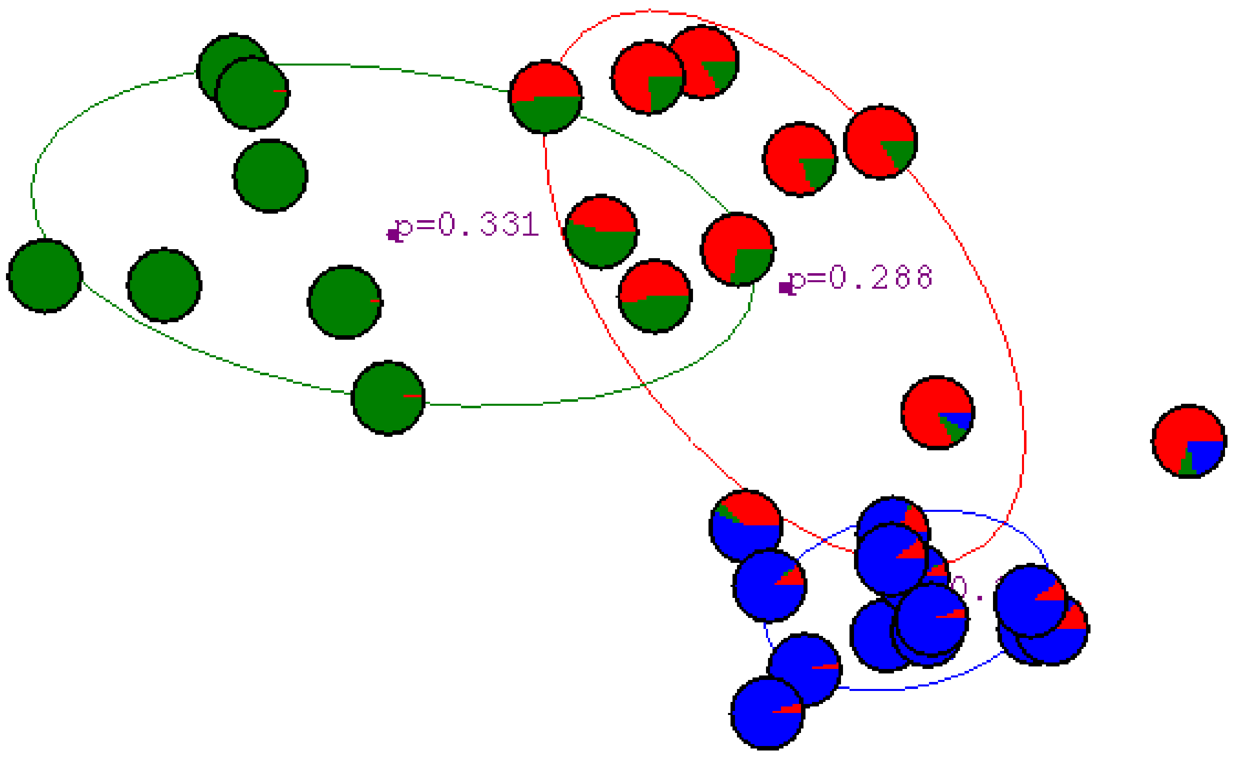
After 2nd iteration



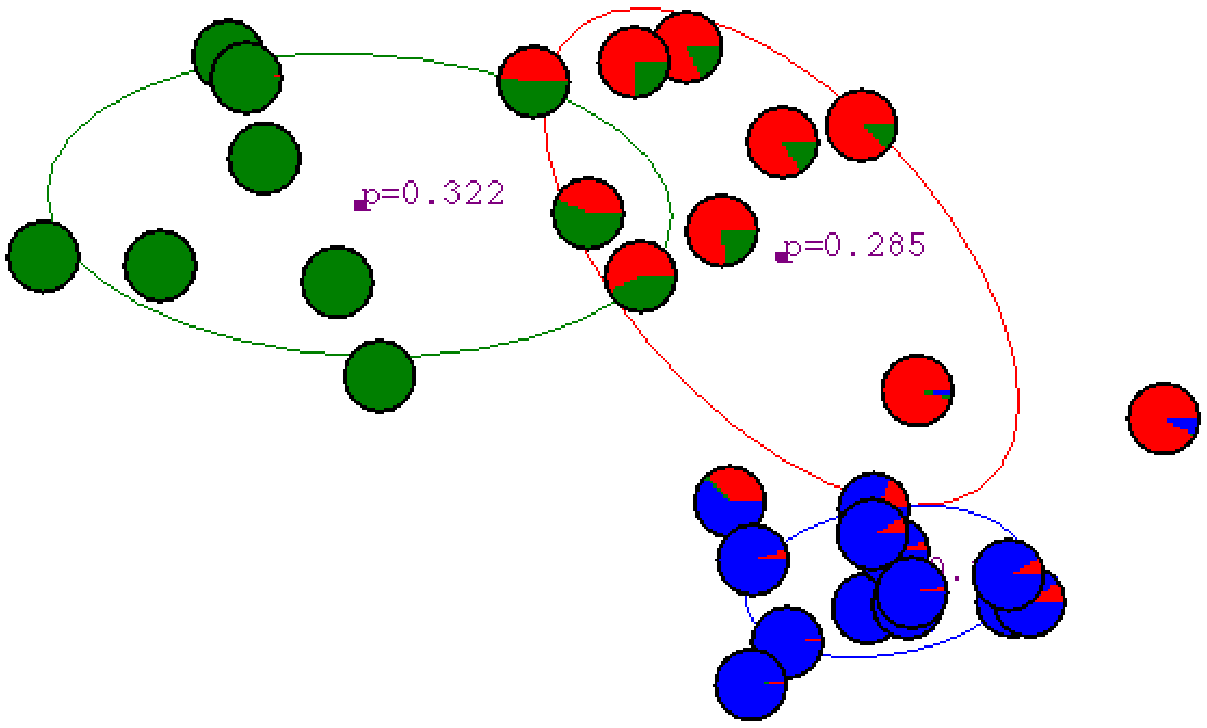
After 3rd iteration



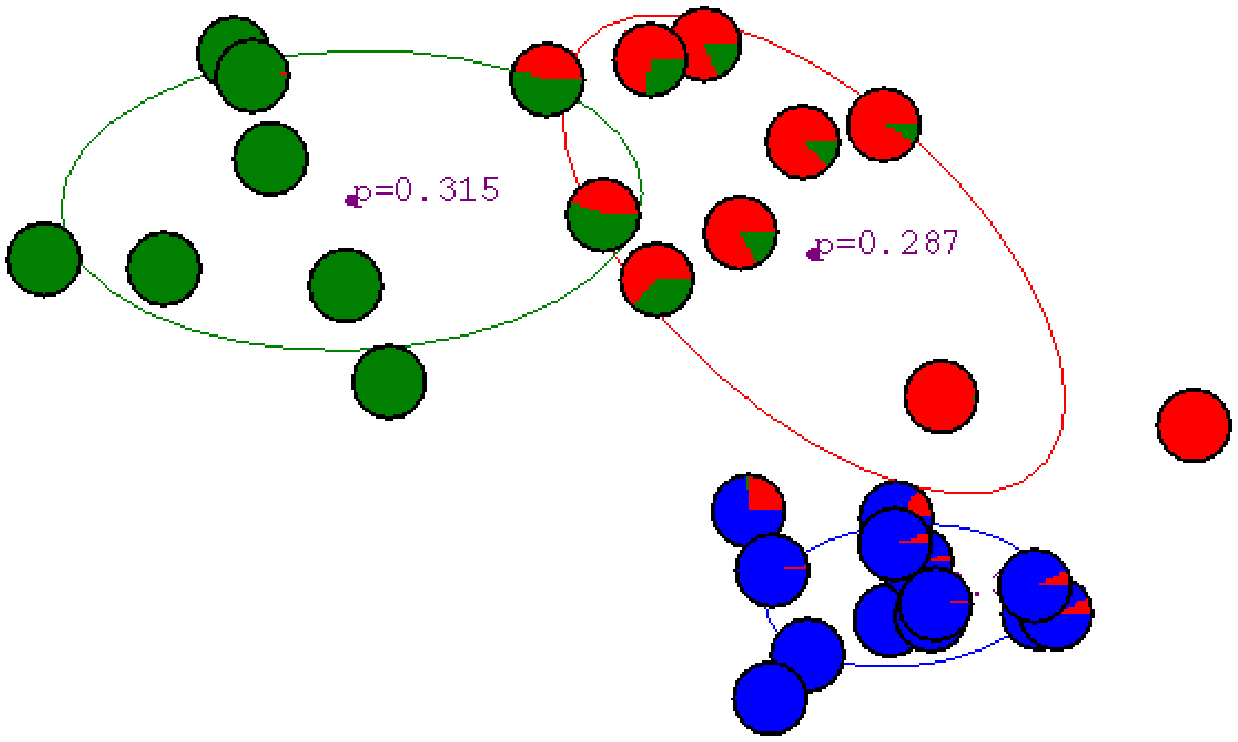
After 4th iteration



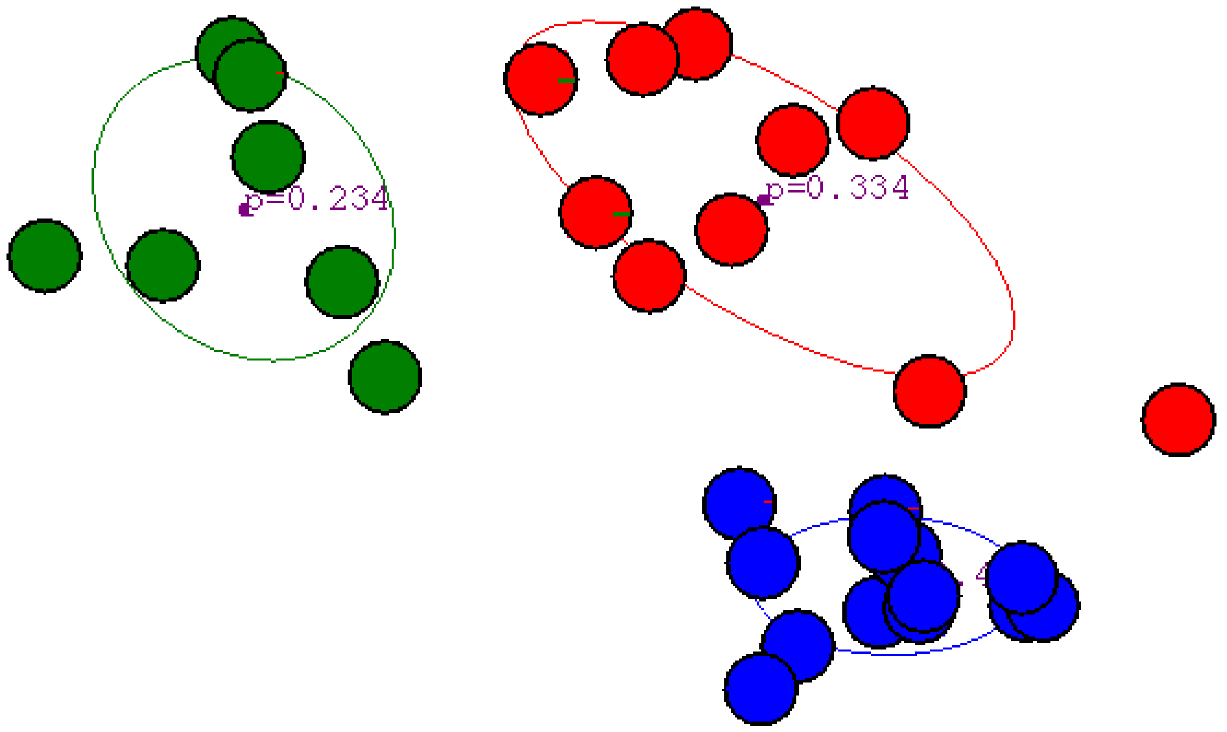
After 5th iteration



After 6th iteration



After 20th iteration



Fatal flaw of maximum likelihood estimation

Assume for each cluster $\Sigma_k = \sigma_k^2 I$.

What happens if

for one data point $\mathbf{x}_i$ we have $\mu_k = \mathbf{x}_i$ and σ_k^2 is very small?

Fatal flaw of maximum likelihood estimation

Assume for each cluster $\Sigma_k = \sigma_k^2 I$.

What happens if

for one data point $\mathbf{x}_i$ we have $\mu_k = \mathbf{x}_i$ and σ_k^2 is very small?

Soft clustering can blow up!!

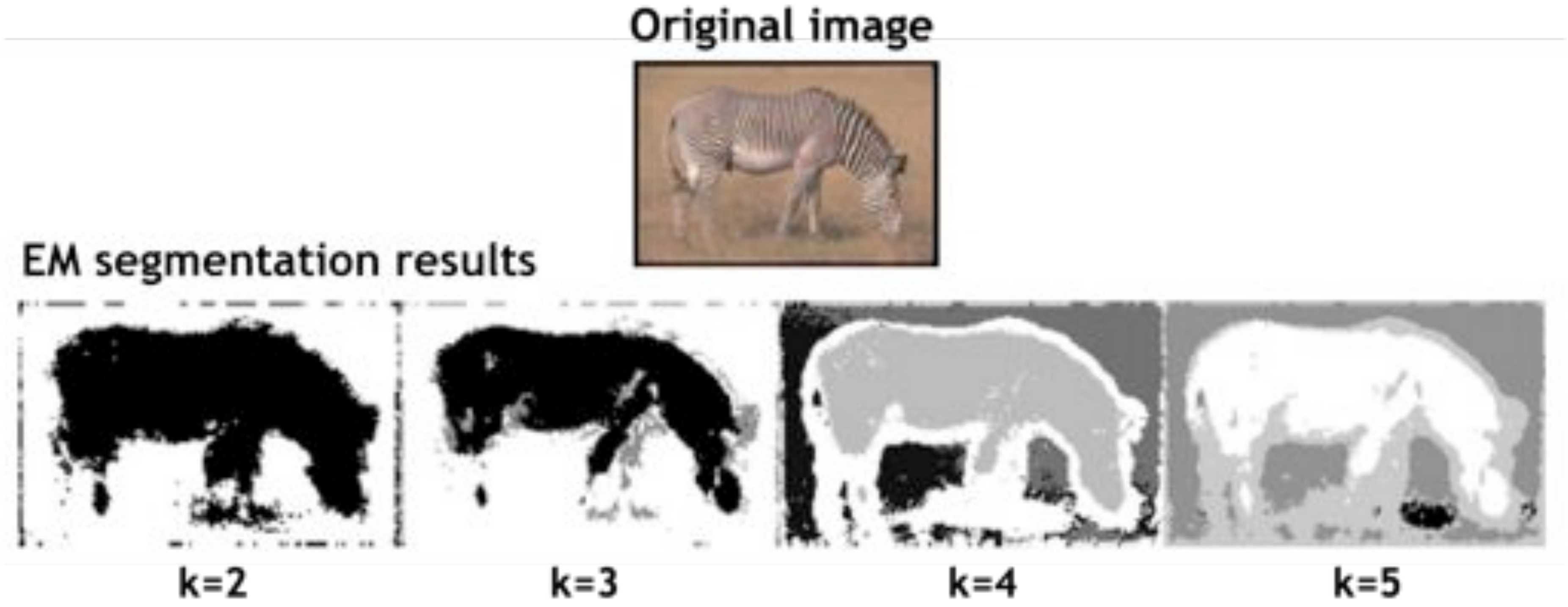
Put one cluster exactly on one data point and let its variance go to zero $\Longrightarrow$ you can get an arbitrary large likelihood!

Applications of EM

Turns out this type of procedure is uesful for lots of problems

- Clustering problems
- Model estimation problems
- Missing data problems
- Find outliers
- Segmentation problems

Segmentation with EM



Summary: GMMs and EM

Pros

- Probabilistic interpretation
- Soft assignments between data points and clusters
- Generative model, can predict novel data points
- Relatively compact storage

Summary: GMMs and EM

Cons

- Local minima
- Need an initial guess of the parameters

  Often good idea to start with an k-means clustering

- Need to know number of components
- There can be numerical problems.

  If $\boldsymbol{\mu}_1 = \mathbf{x}_1$, $\Sigma_1 = \sigma I$ and $\sigma \to 0$, what happens to the likelihood score?

Pen & Paper (and Programming) assignment

- Details available on the course website.
- The compulsory assignment is a very simple clustering problem. There is also an optional programming exercise which involves exploring k-means clustering with *Matlab*.
- Mail me about any errors you spot in the Exercise notes.
- I will notify the class about errors spotted and corrections via the course website and mailing list.