

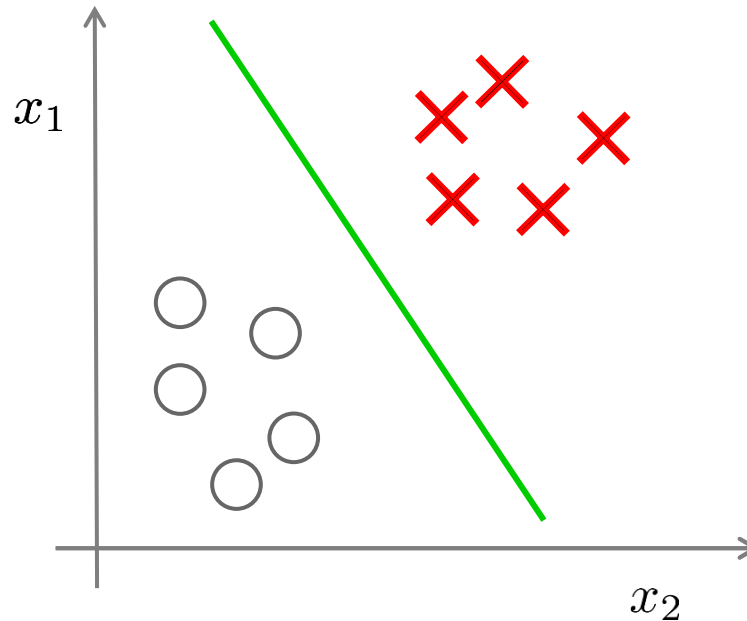
Aprendizaje Automático

Unsupervised Learning Clustering



Máster en Bioinformática y Biología Computacional

Supervised Learning

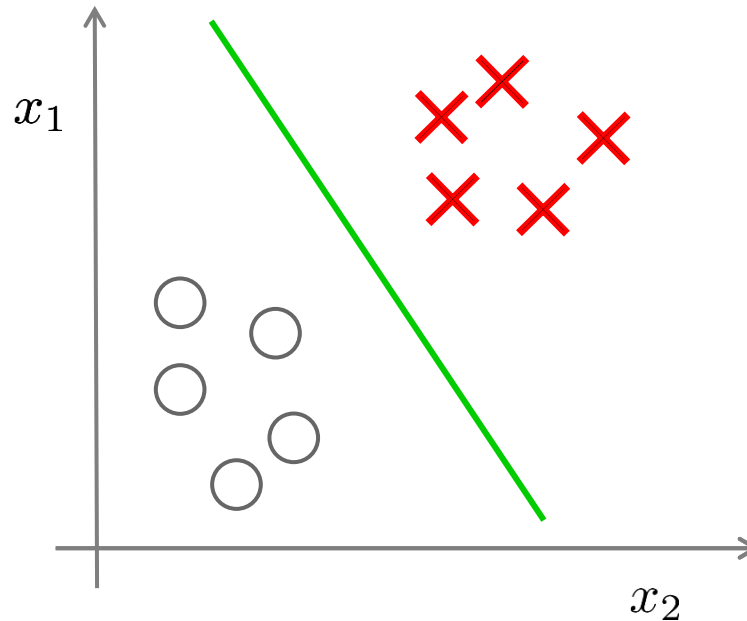


Training set:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})\}$$

Labels!

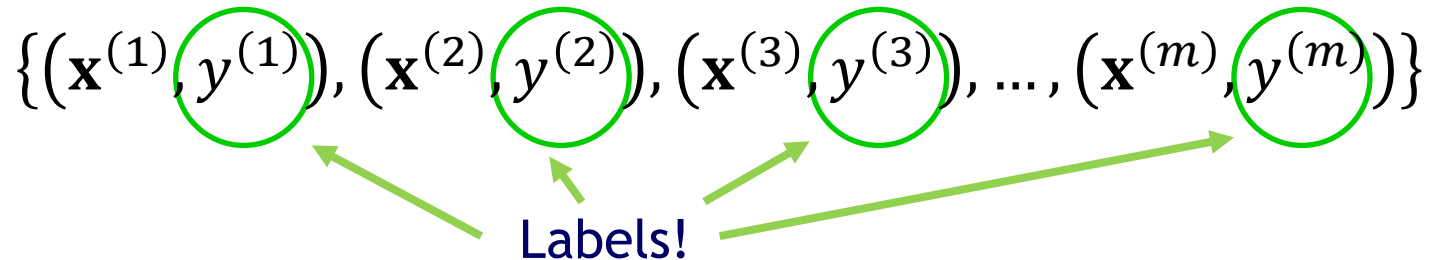
Supervised Learning



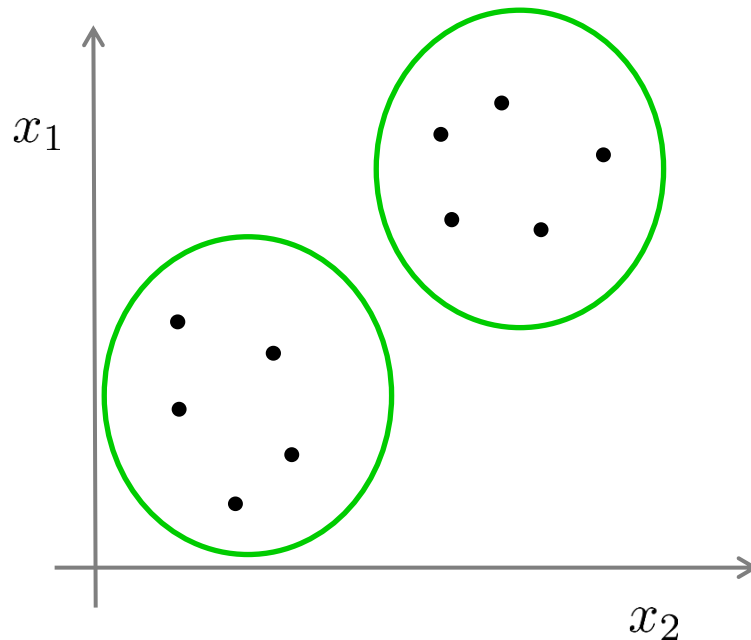
Training set:

$$\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$$

Labels!



Unsupervised Learning



Clustering
Algorithms
(grouping)

Training set:

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \dots, \mathbf{x}^{(m)}\}$$



No Labels!

Clustering: What is it?



- Proximity between items
- What do we mean by proximity?
- Are they really related?
- Can we perform such a methodology in an automatic way?

Clustering: What is it?

- Classification 
 - There exist labels for some points
 - Rule such that new points are assigned labels properly
 - Supervised learning
- Clustering
 - No labels
 - Points assign to clusters according to “how close” they are to one another
 - Identify structure in data 
 - Unsupervised learning

Clustering: What is it?



- **Cluster analysis** = organization of a “collection of patterns into clusters based on similarity” (Jain, Murty, et al. 1999)
- The definition and the scope of “cluster” in the data set are not easy to define
- According to (Jain and Dubes 1988), a cluster is:
 - a. set of similar objects
 - b. set of points aggregated in the testing environment | the distance between two points in a cluster is less than the distance between any point in the cluster and any point of other clusters
 - c. densely connected regions in a multi-dimensional space separated by loosely connected points

Clustering: Distance

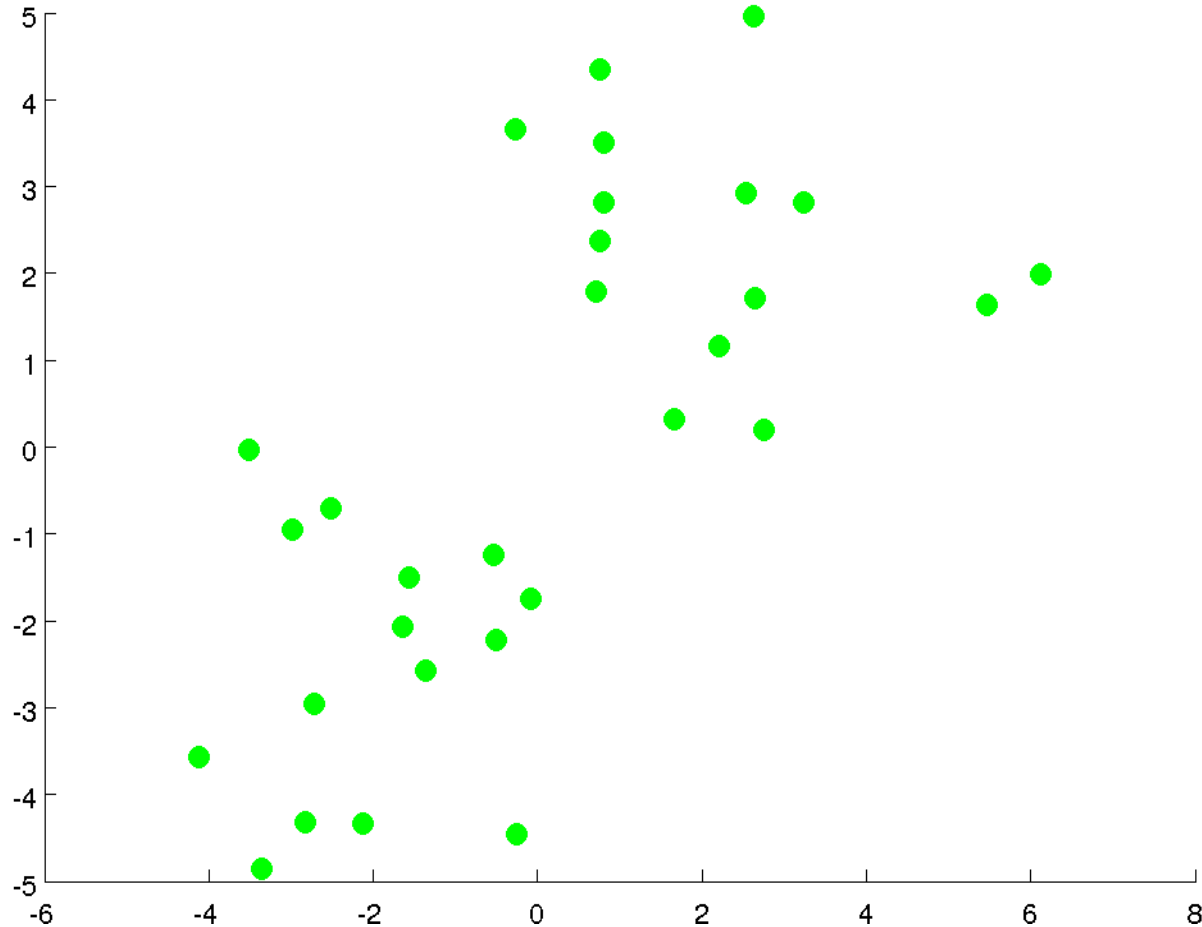
- The distance between two instances $x^{(i)}$ and $x^{(j)}$, which is a metric distance measure if it satisfies the following properties:

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \leq d(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) + d(\mathbf{x}^{(k)}, \mathbf{x}^{(j)}), \quad \forall \mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)} \in \mathcal{R}^n$$

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 0 \rightarrow \mathbf{x}^{(i)} = \mathbf{x}^{(j)}, \forall \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathcal{R}^n$$

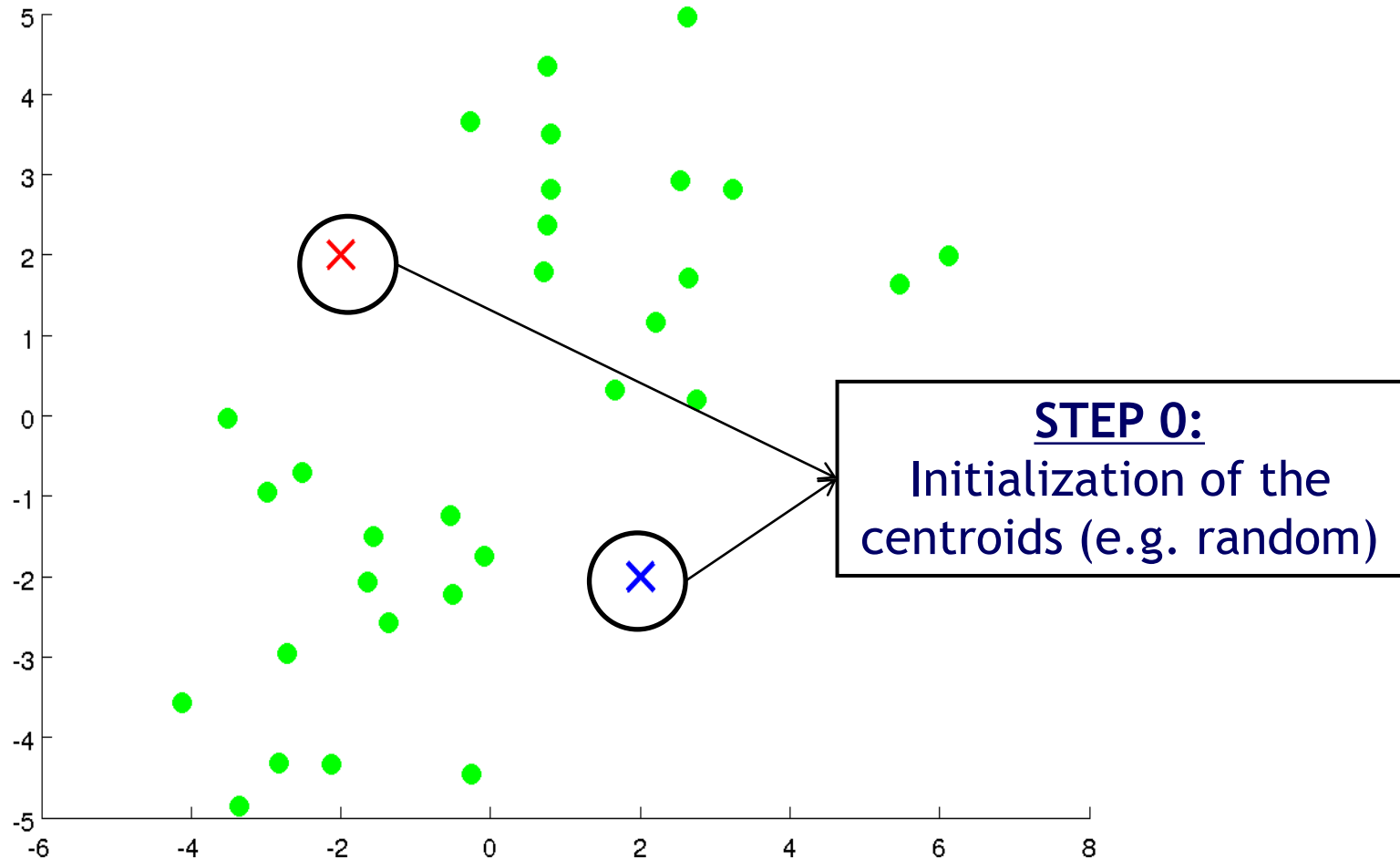
e.g. Minkowski distance for numeric attributes (Euclidean, Manhattan,...)

K-means: How it works?



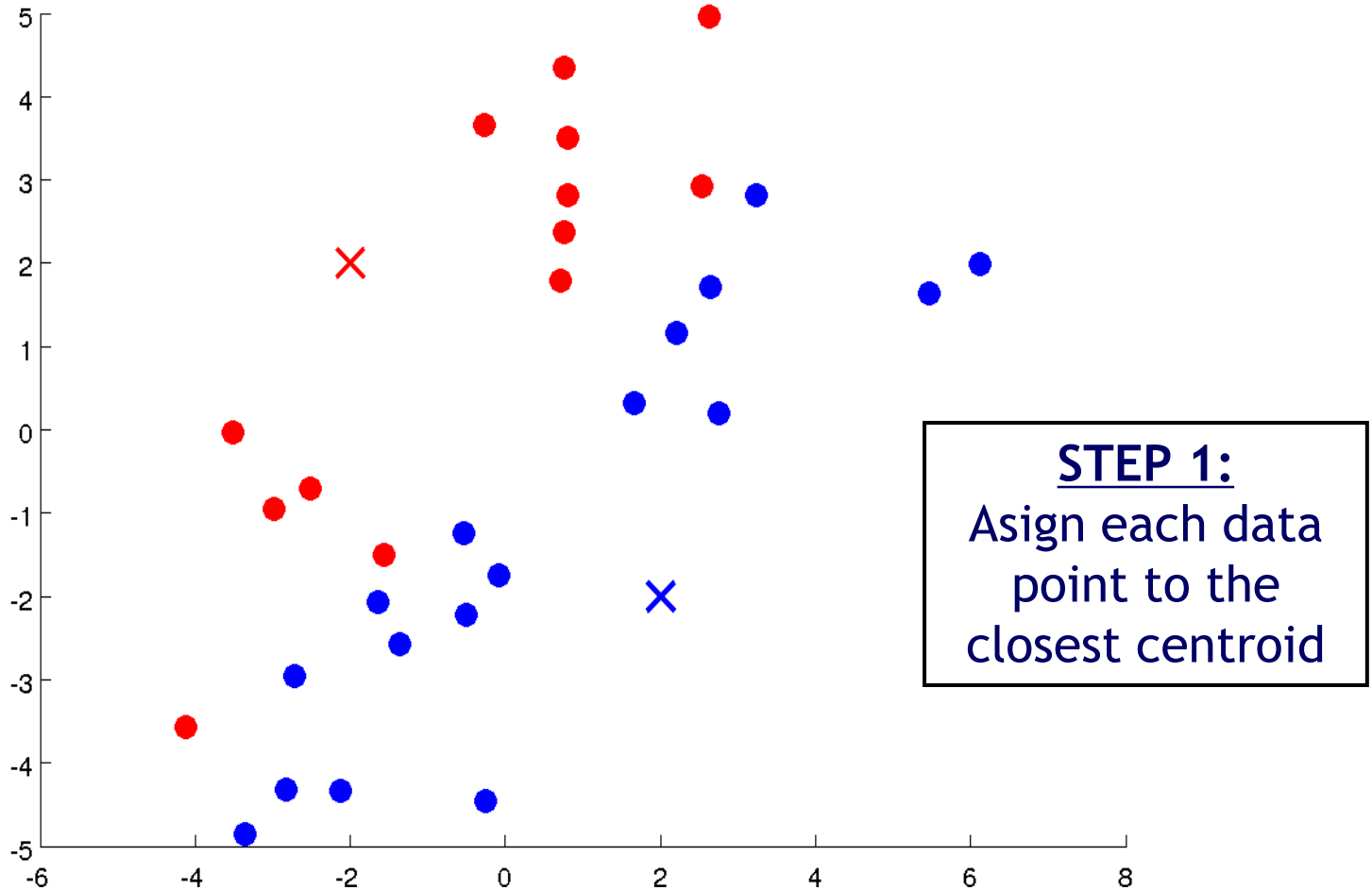
Example: We have an unlabeled database, and we want to group the data into two clusters. K-means is an iterative process

K-means: How it works?



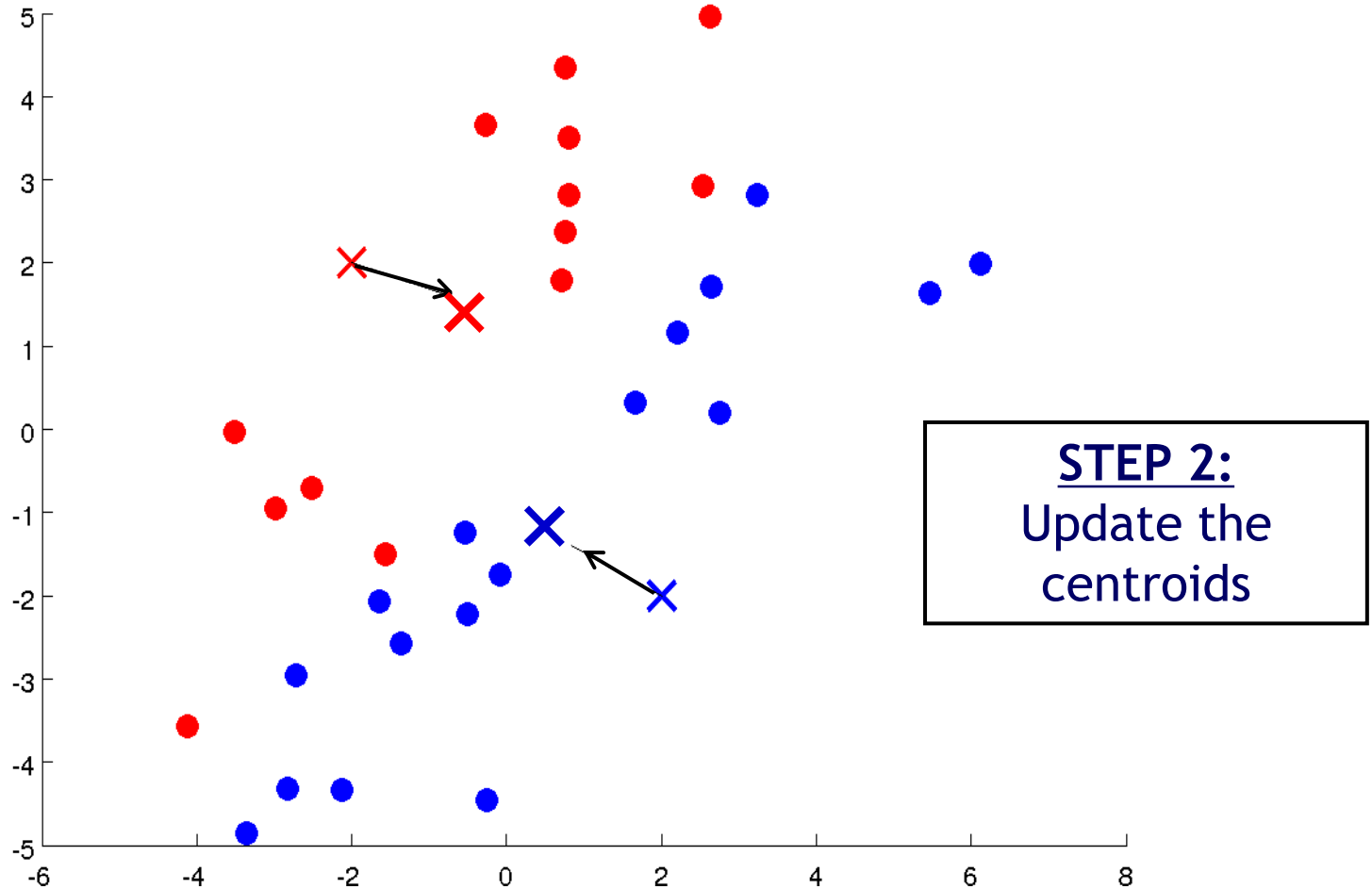
Two cluster centroids because we want to have two clusters

K-means: How it works?



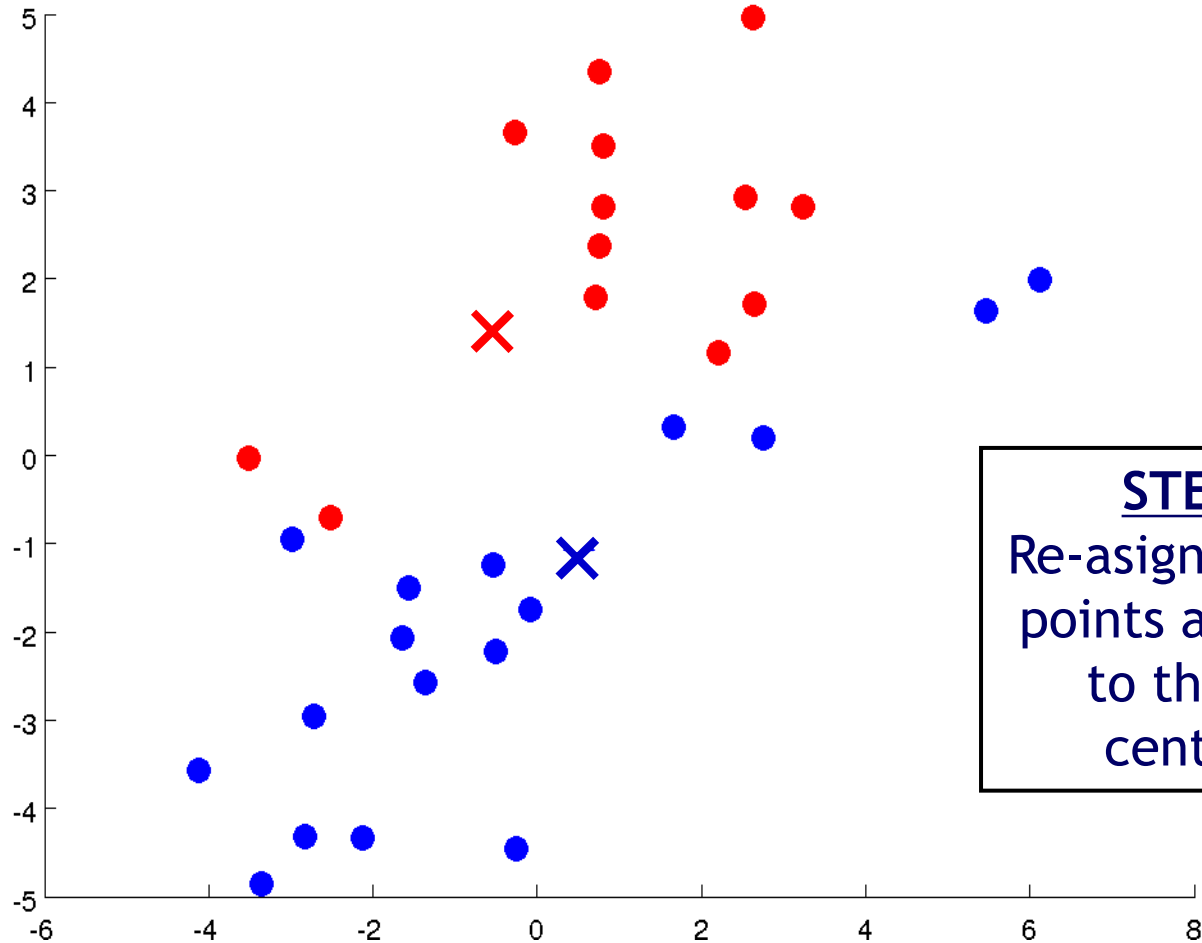
Compute the distance from each point to each cluster to assign the label to the closest one

K-means: How it works?



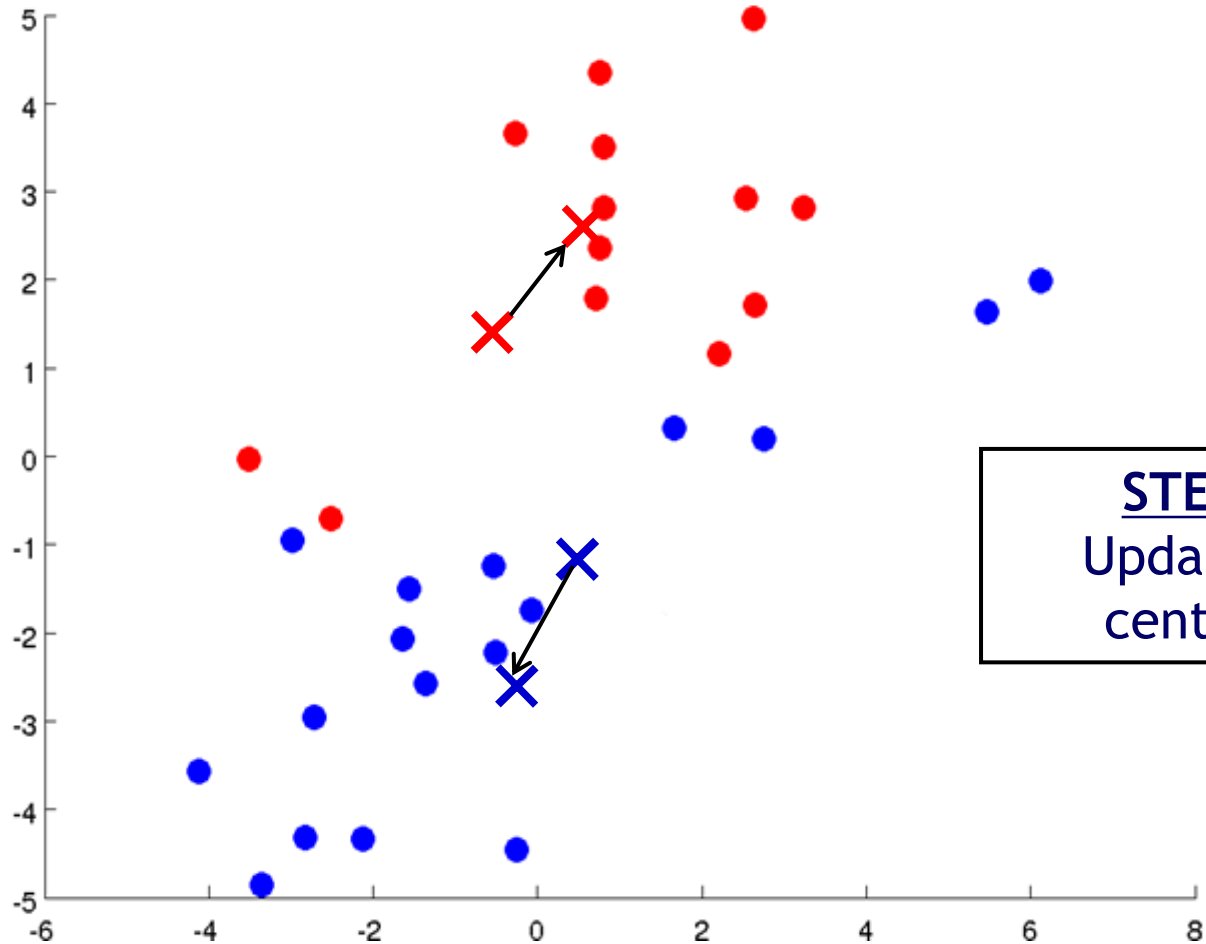
Move the cluster centroids to the mean location of the points colored the same colour

K-means: How it works?



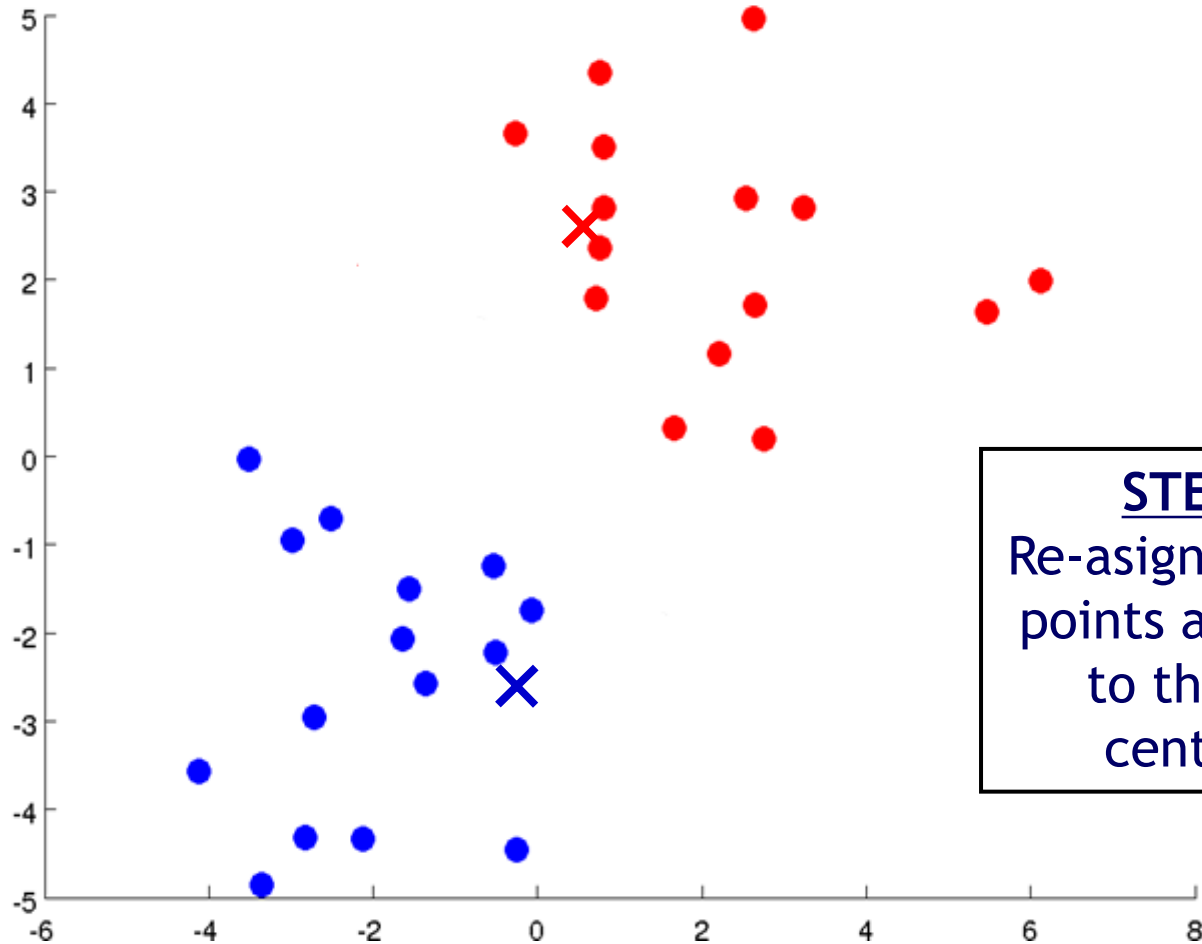
STEP 1:
Re-assign the data
points according
to the new
centroids

K-means: How it works?



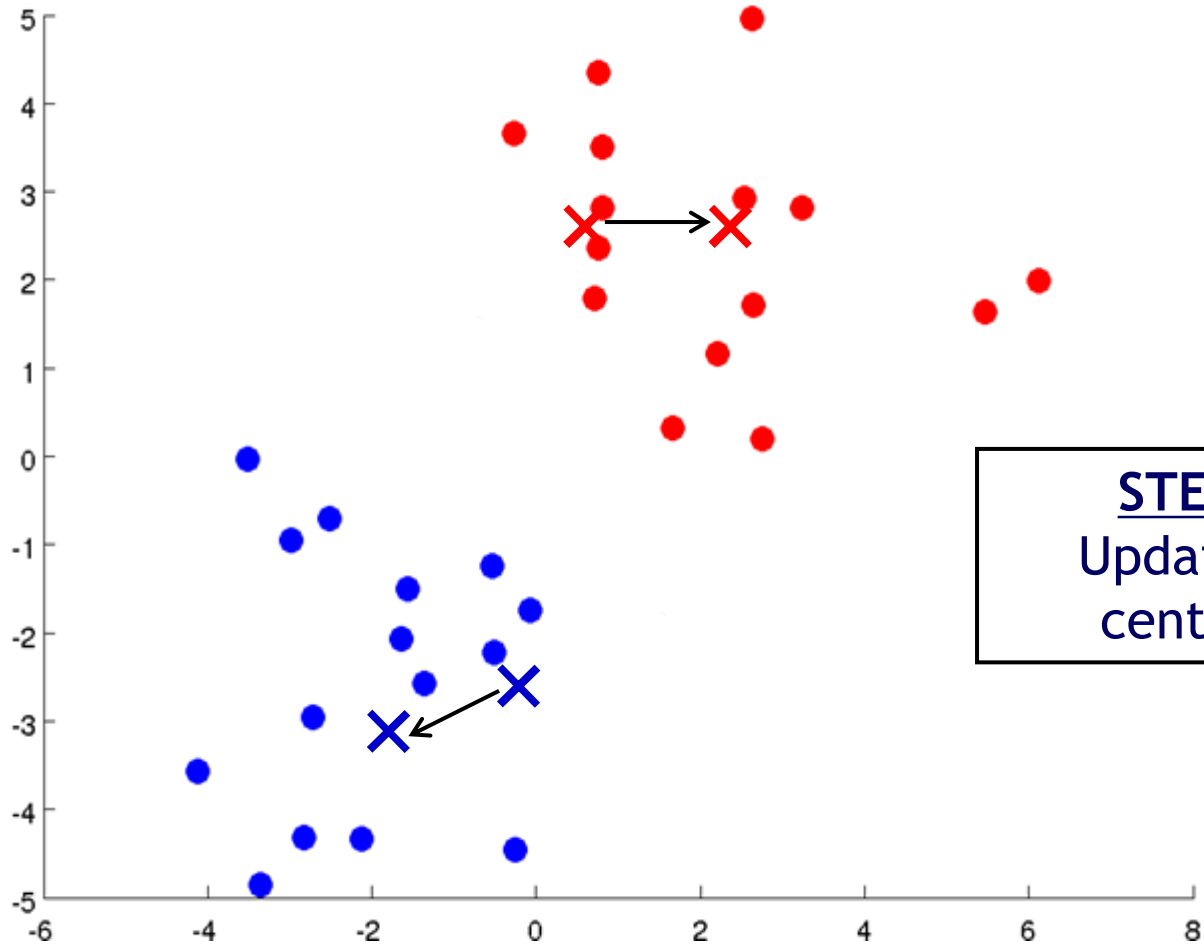
STEP 2:
Update the
centroids

K-means: How it works?



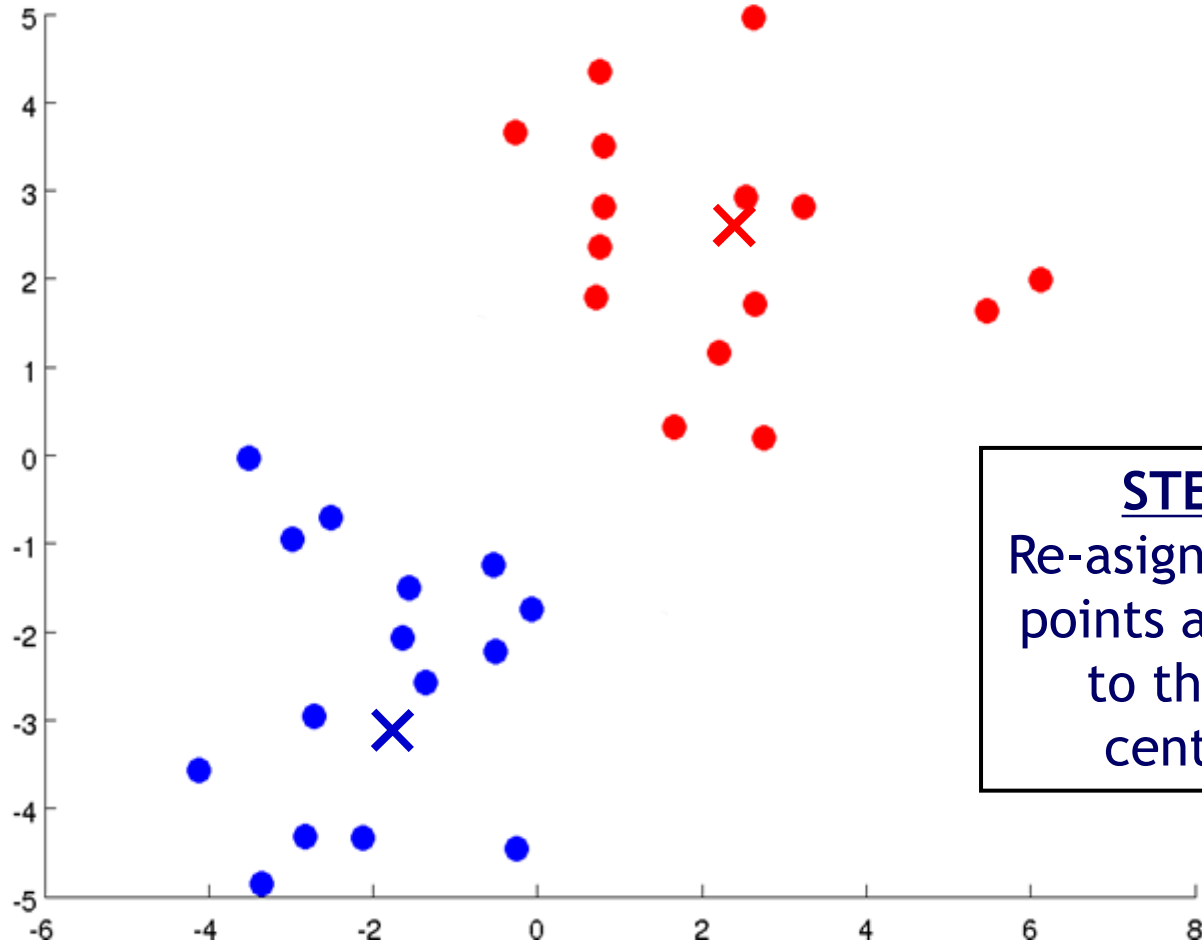
STEP 1:
Re-assign the data
points according
to the new
centroids

K-means: How it works?



STEP 2:
Update the
centroids

K-means: How it works?



STEP 1:
Re-assign the data
points according
to the new
centroids

If you keep running additional iterations of K means from here the cluster centroids will not change from here in this example. K-means has converged

K-means: Algorithm

Input:

- K : number of clusters
- $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}\}$ training set $\mathbf{x}^{(i)} \in \mathbb{R}^n$

STEP 0: Random initialization K centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

for $i = 1$ to m

STEP 1:

$c^{(i)} :=$ index of the cluster (from 1 to K) closest to $\mathbf{x}^{(i)}$

$$c^{(i)} := \arg \min_k \|\mathbf{x}^{(i)} - \mu_k\|^2$$

STEP 2:

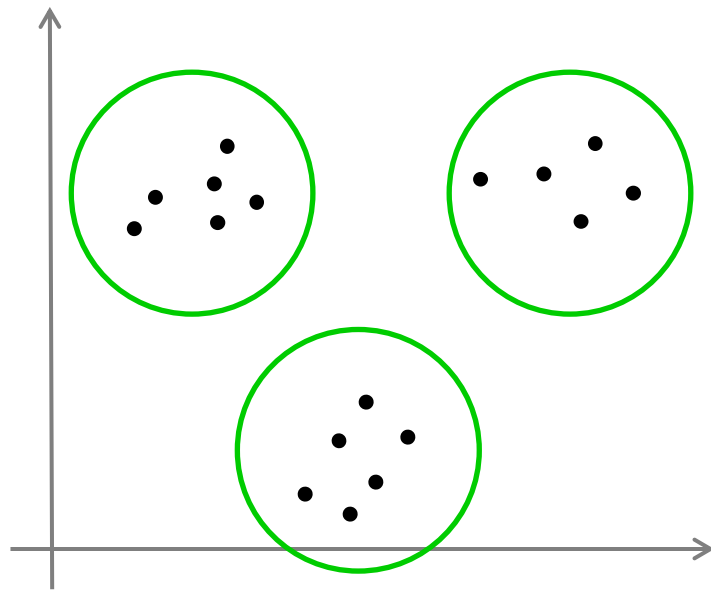
for $k = 1$ to K

$\mu_k :=$ mean value of the data assigned to the cluster k

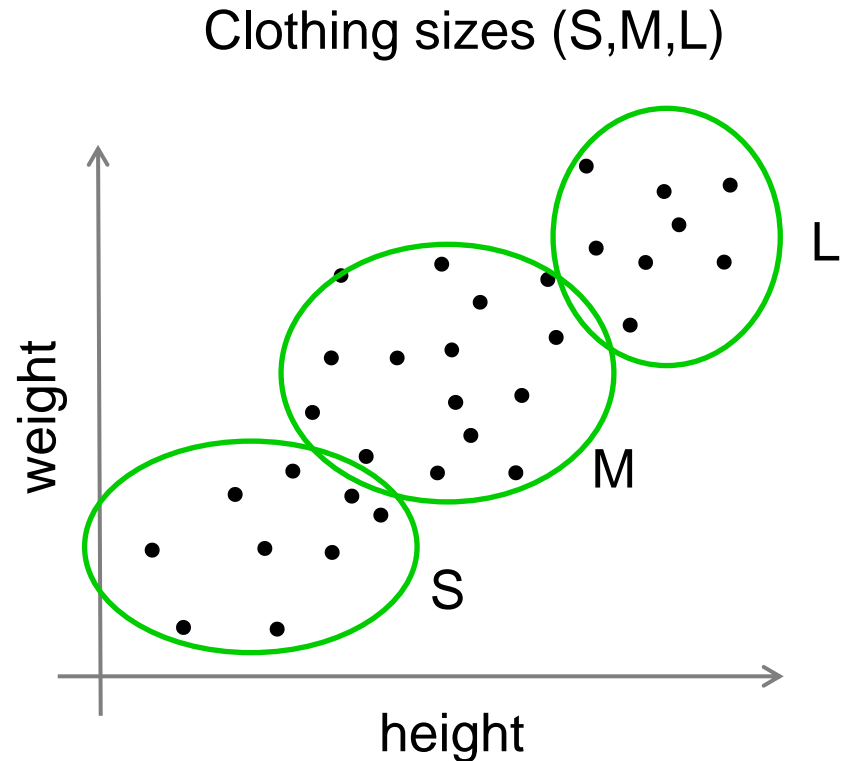
}

$$\mu_k := \frac{\sum_{i=1}^m 1\{c^{(i)} = k\} \mathbf{x}^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = k\}}$$

K-means: overlap between clusters



Example of well separated clusters, like the one before



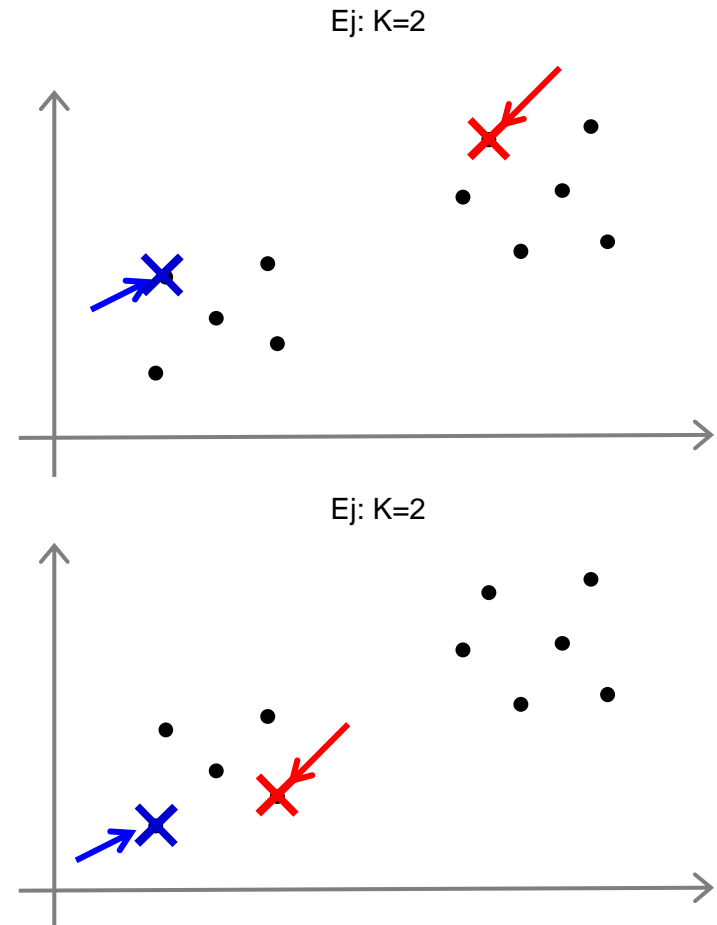
Example of diffuse clusters. Database of the height and weight of a population. You want to design T-shirts based on that into S,M,L sizes

K-means: Random initialization

Should have $K < m$

Randomly pick K training samples

Set μ_1, \dots, μ_K equal to these K samples

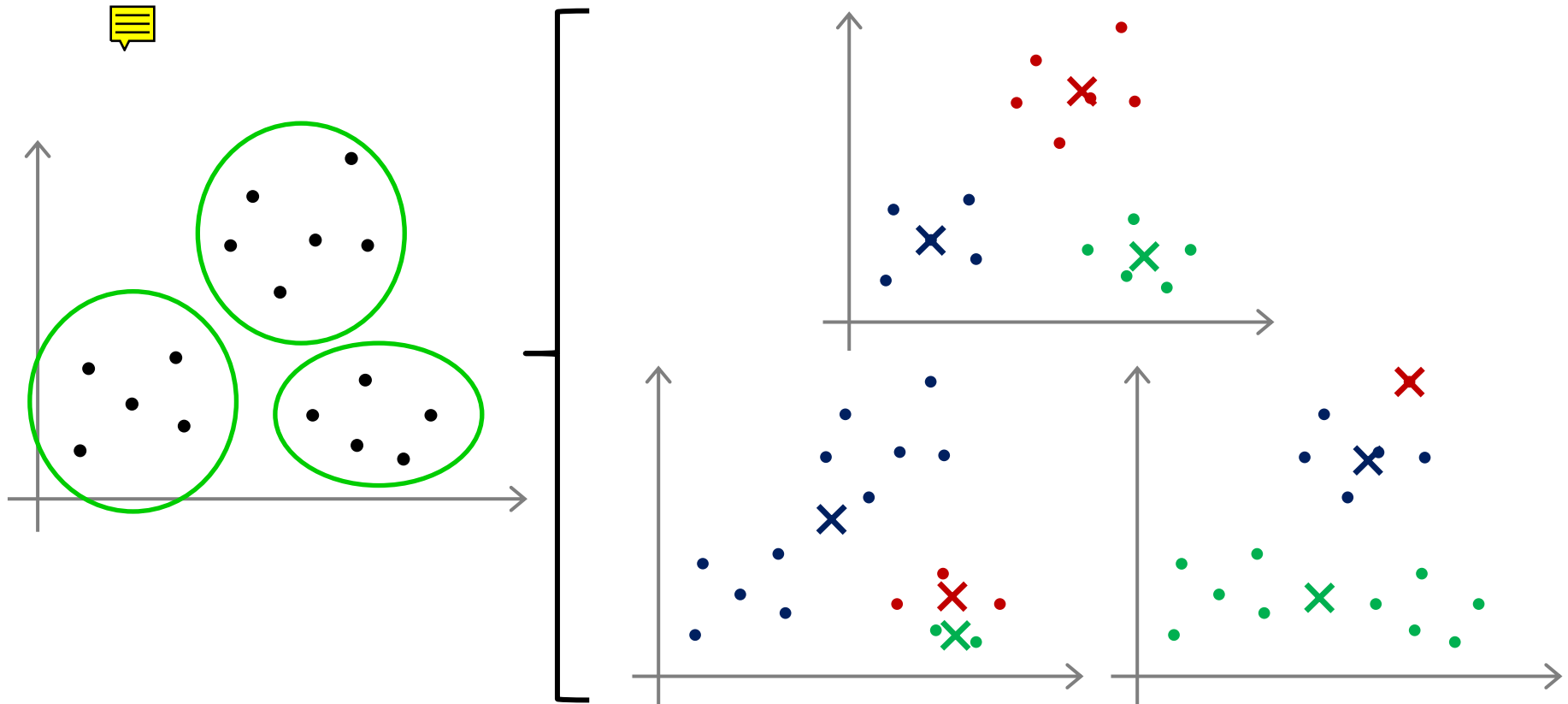


Examples of two different initializations. K-means could end up converging to different solutions depending on the first initialization.

K-means: local optima

K-means does NOT guarantee to find the global minimum of the cost function

The result is highly dependent of the initialization



Solution: Try multiple random initializations, run K-means several times to make sure we get a good solution


K-means: cost function

$c^{(i)}$ = index of the cluster $(1, 2, \dots, K)$ to which it is assigned $\mathbf{x}^{(i)}$

μ_k = centroid of the cluster k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = centroid of the cluster to which it is assigned $\mathbf{x}^{(i)}$. This is a matrix of m elements, one for each example.

The objective of K-means is to optimize (minimize) the following cost function (also known as distortion function):

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|\mathbf{x}^{(i)} - \mu_{c^{(i)}}\|^2$$


$$\min_{\substack{c^{(1)}, \dots, c^{(m)} \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

Minimize the squared distance between each example $\mathbf{x}^{(i)}$ and the location of the cluster centroid to which $\mathbf{x}^{(i)}$ has been assigned.

K-means: cost function



STEP 0: Random initialization of K centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

STEP 1: for $i = 1$ to m
 $c^{(i)} :=$ index of the cluster (from 1 a K) closest to $\mathbf{x}^{(i)}$

$$\min_{c^{(1)}, \dots, c^{(m)}} J(c^{(1)}, \dots, c^{(m)})$$

STEP 2: for $k = 1$ to K
 $\mu_k :=$ mean value of the data assigned to the cluster k

$$\min_{\mu_1, \dots, \mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

K-means: local optima (solution)



For $i = 1$ to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

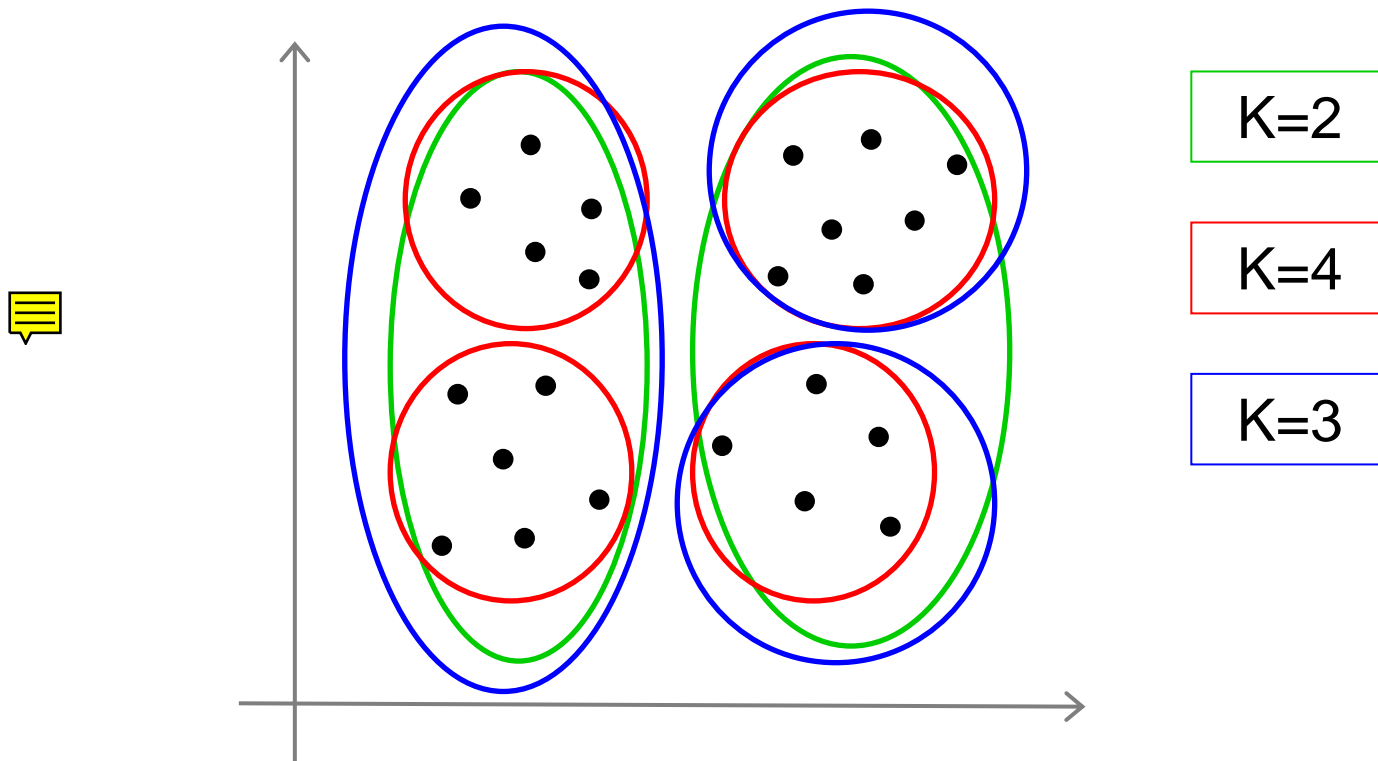
Compute cost function (distortion)

$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$
}

Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

K-means: choosing k

Unfortunately, there is NOT a perfect solution

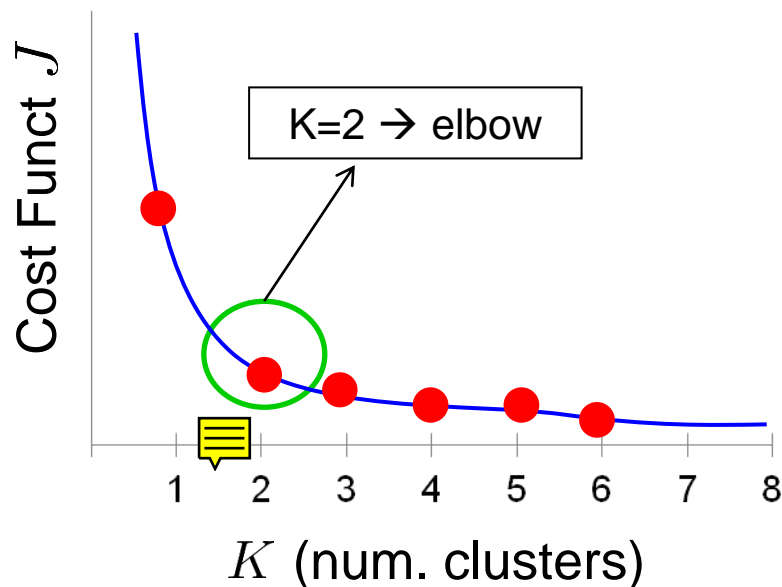


How many clusters do you see?

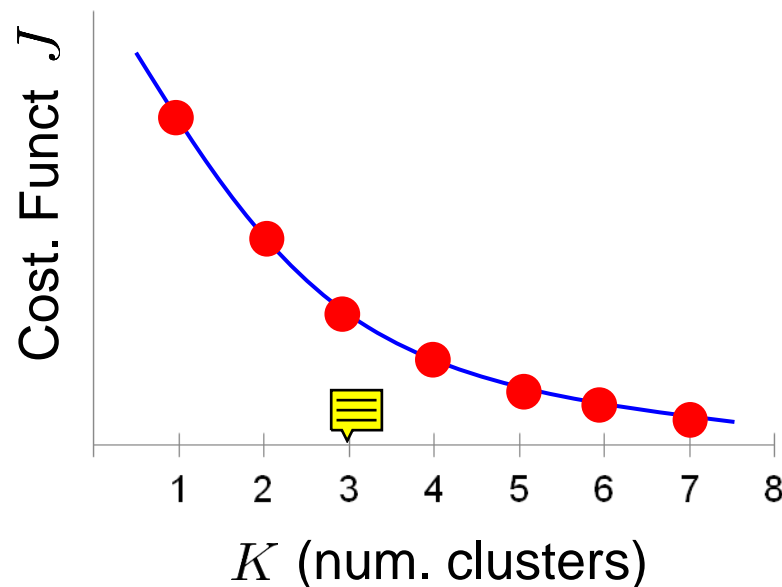
This can be very ambiguous

K-means: choosing k

1. Know your data. i.e., you know the number of K
2. Elbow method: run K-means several times varying K



e.g. real life



In this case you don't find a real elbow, but something ambiguous

Aprendizaje Automático

Unsupervised Learning

Clustering



Máster en Bioinformática y Biología Computacional

Extended Material
(Optional, not evaluable)

GMM and
Expectation-Maximization algorithm (EM)

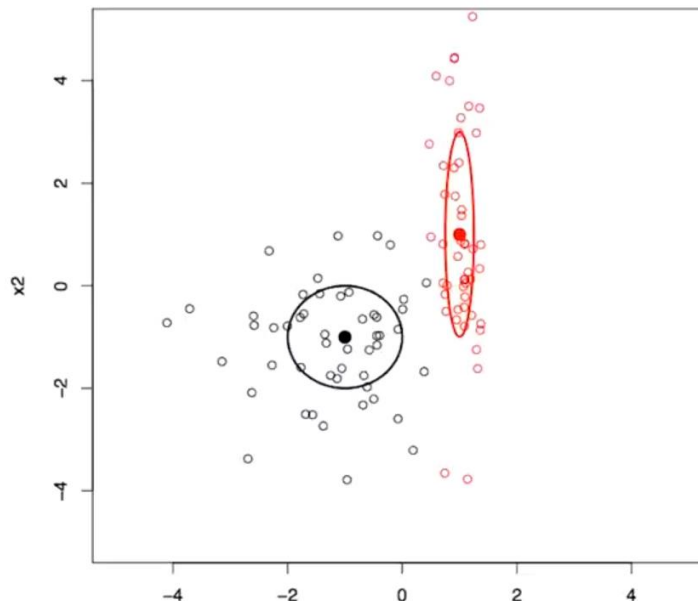
GMM and EM algorithm

With K-means the clusters only could have circular shape. Only in the overlapping areas the distance to the centroid determines the assigned cluster.

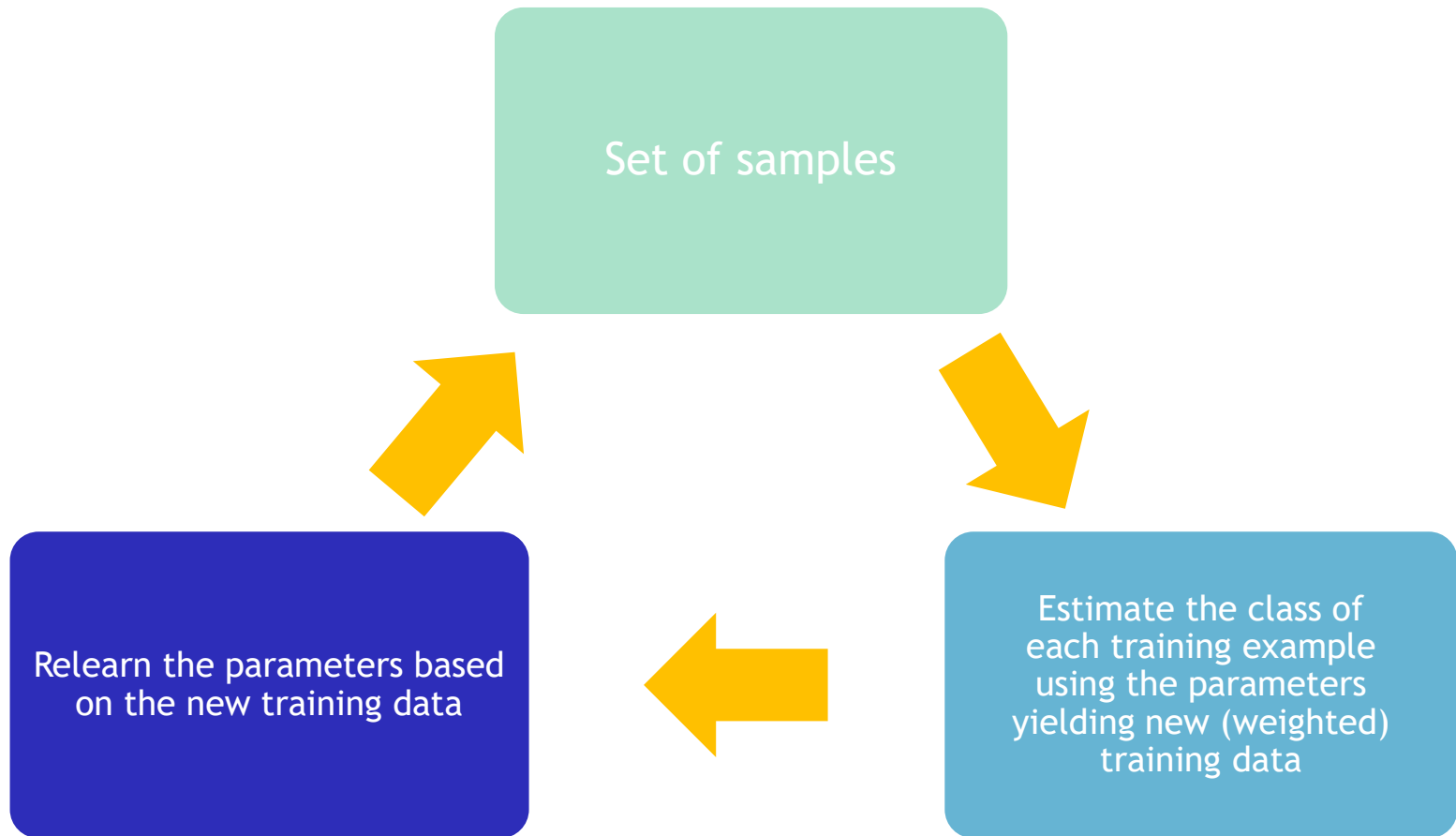
With GMM we are going to model the data. For each cluster we are going to model a Gaussian Mixture, that do not need to have a circular shape.

Mixture Models

Gaussian mixture model ($K = 2$):



Expectation-Maximization algorithm (EM)



Expectation-Maximization (EM) algorithm

- One of the most popular approaches to maximize the likelihood is to use the Expectation-Maximization (EM) algorithm
- Basic idea of EM algorithm:
 - **E-Step: Estimate** the distribution of the hidden variable given the data and the current value of the parameters
 - **M-Step: Maximize** the joint distribution of the data and the hidden variable

Gaussian Mixture Models

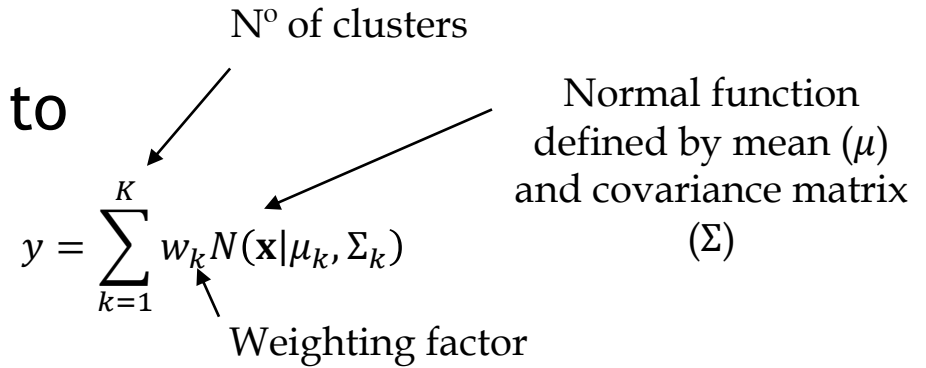
Combination of simple models to generate a complex one:

$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

N° of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

Weighting factor



Gaussian Mixture Models

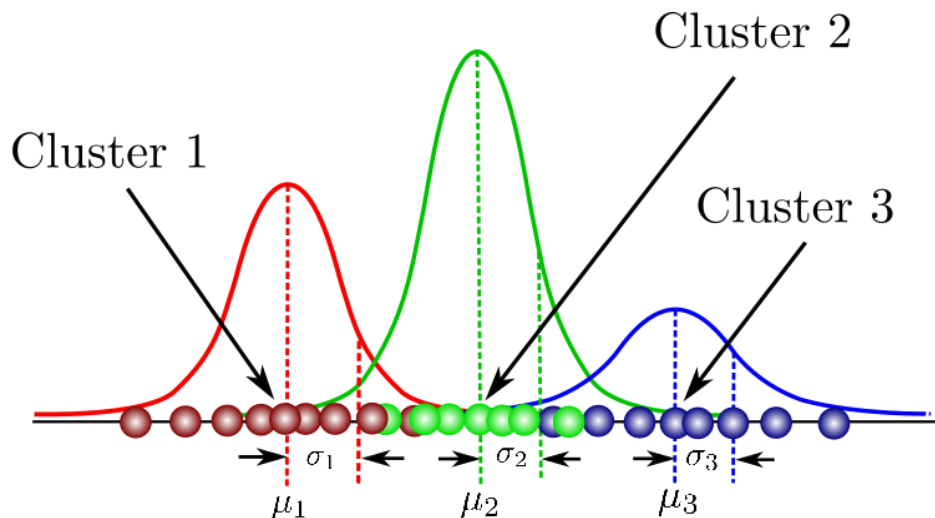
Combination of simple models to generate a complex one:

$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

N° of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

Weighting factor



Gaussian Mixture Models

Combination of simple models to generate a complex one:

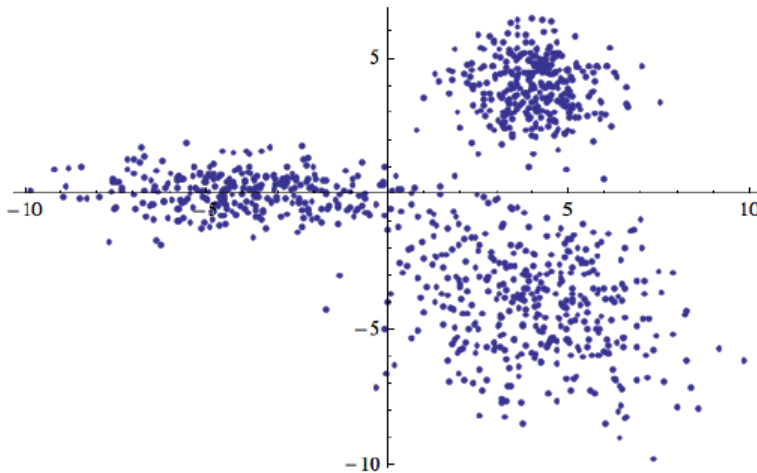


$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

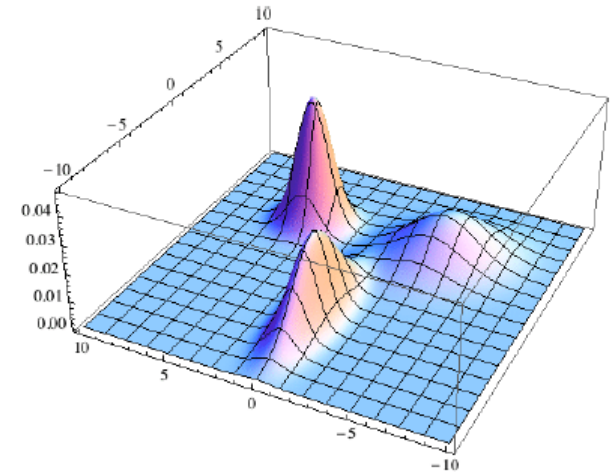
N^o of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

Weighting factor



Data from an unknown function



Data distribution modelled with GMM

Gaussian Mixture Models

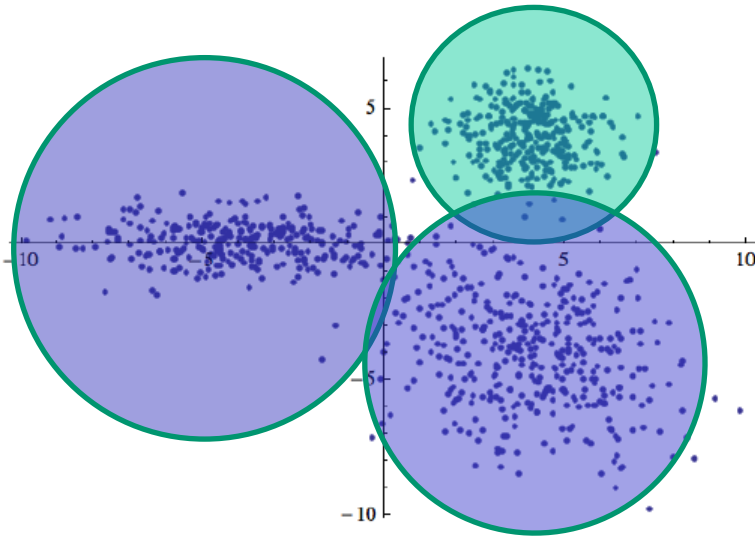
Combination of simple models to generate a complex one:

$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

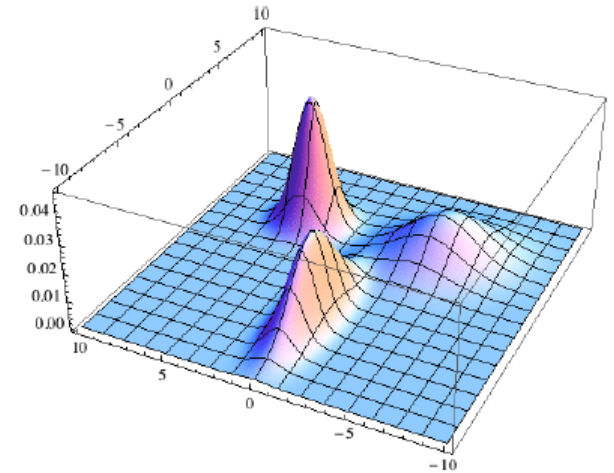
N° of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

Weighting factor



Data from an unknown function



Data distribution modelled with GMM

Gaussian Mixture Models

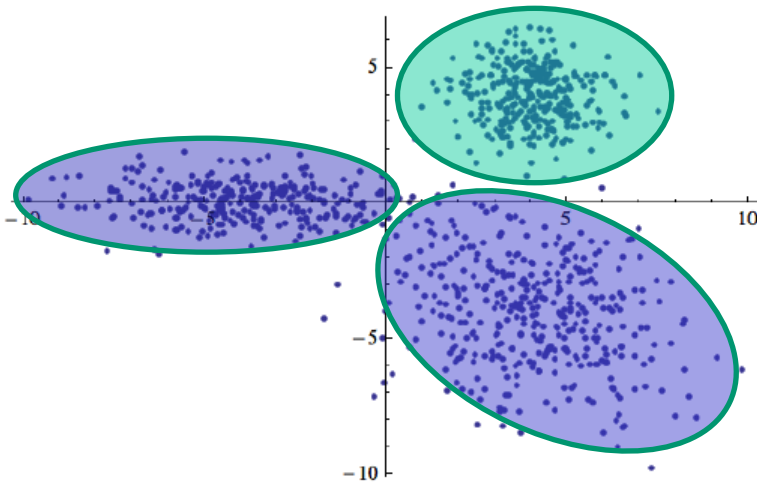
Combination of simple models to generate a complex one:

$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

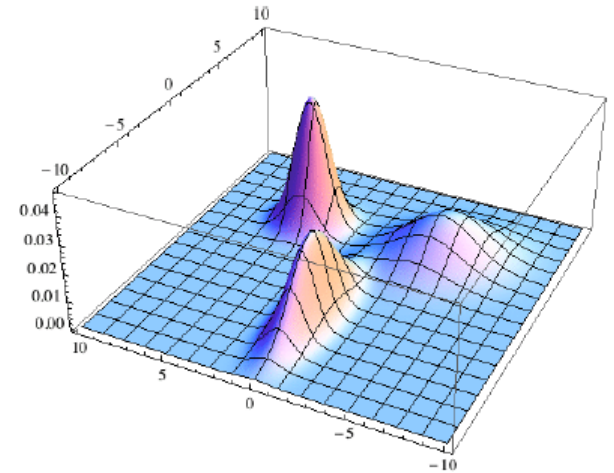
N^o of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

Weighting factor



Data from an unknown function



Data distribution modelled with GMM

Gaussian Mixture Models

Combination of simple models to generate a complex one:

- Rather than identifying clusters by “nearest” centroids
- Fit a Set of k Gaussians to the data
- Maximum Likelihood over a mixture model

Parameters (for the Gaussian Mixture):

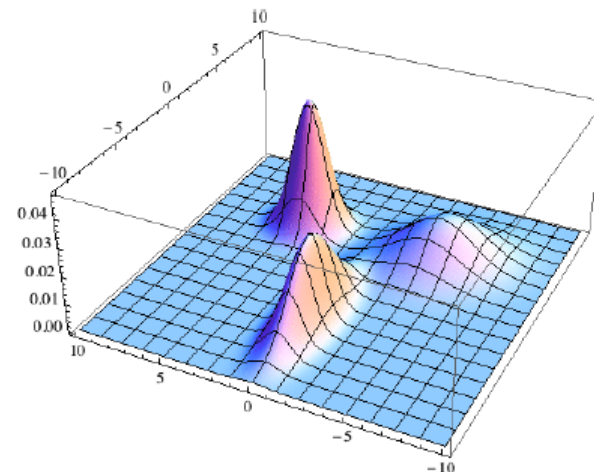
- Mean vector:
- Covariance matrix: Σ_k
- Weight vector: w_k

$$y = \sum_{k=1}^K w_k N(\mathbf{x} | \mu_k, \Sigma_k)$$

N^o of clusters

Normal function defined by mean (μ) and covariance matrix (Σ)

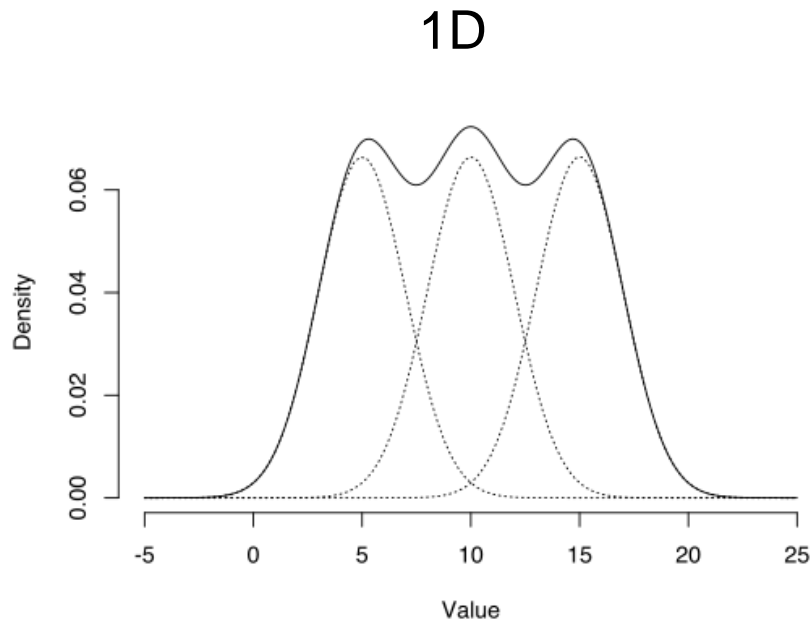
Weighting factor



Data distribution modelled with GMM

Gaussian Mixture Models

GMM can be used also for Probability Density Estimation

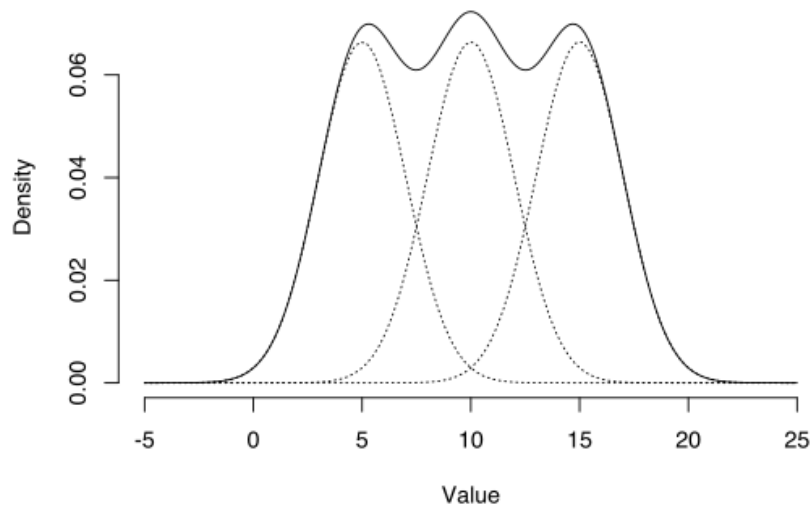


1D features, 3 Gaussians

Gaussian Mixture Models

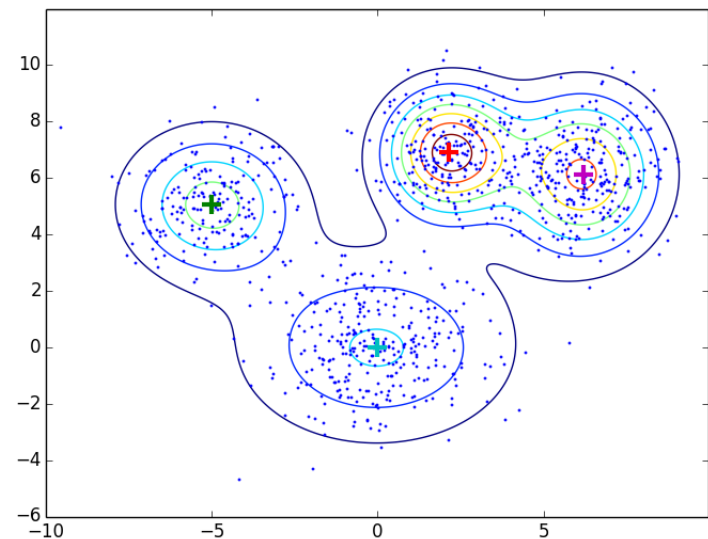
GMM can be used also for Probability Density Estimation

1D



1D features, 3 Gaussians

2D

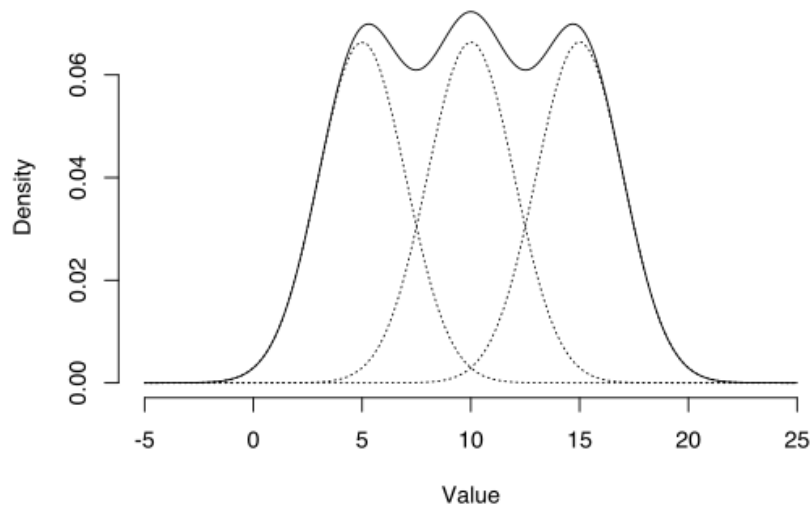


2D features, 4 Gaussians

Gaussian Mixture Models

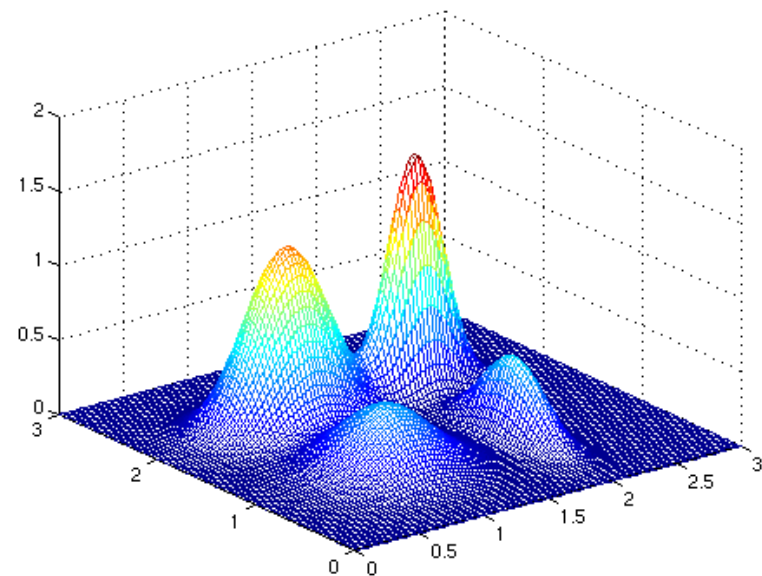
GMM can be used also for Probability Density Estimation

1D



1D features, 3 Gaussians

2D



2D features, 4 Gaussians

Gaussian Mixture Models

- GMM: the weighted sum of a number of Gaussians (K) with different weights:

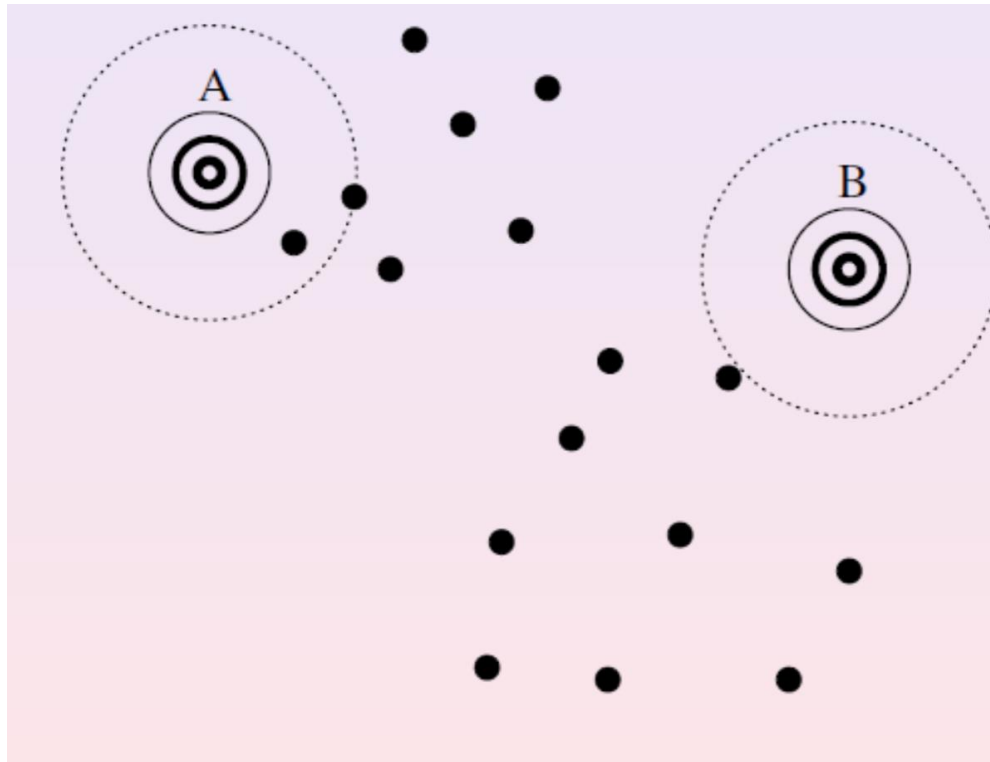
$$\begin{aligned} GMM &= \sum_{k=1}^K w_k N(\mathbf{x}|\mu_k, \Sigma_k) = \\ &= w_1 N(\mathbf{x}|\mu_1, \Sigma_1) + w_2 N(\mathbf{x}|\mu_2, \Sigma_2) + \cdots + w_K N(\mathbf{x}|\mu_k, \Sigma_k) \end{aligned}$$

where $\sum_{k=1}^K w_k = 1$

- We can compute the probability P_{ik} of data point $\mathbf{x}^{(i)}$ in cluster k , given parameters (w_k, μ_k, Σ_k) .

Gaussian Mixture Models: Algorithm

- **E-Step:** for each point, which Gaussian generated it?



Gaussian Mixture Models: Algorithm

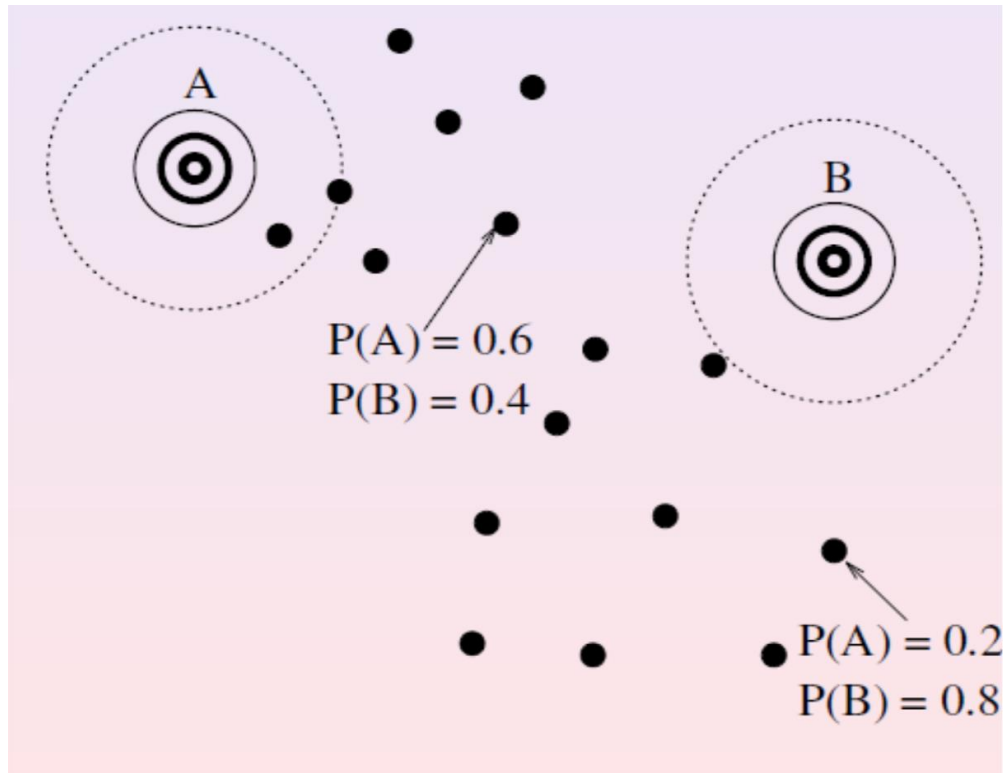
- **E-Step:** for each point, estimate the probability that each Gaussian generated it (compute P_{ik}). Each P_{ik} is calculated as:

$$P_{ik} = \frac{w_k N(\mathbf{x}^{(i)} | \mu_k, \Sigma_k)}{\sum_{j=1}^K w_j N(\mathbf{x}^{(i)} | \mu_j, \Sigma_j)}$$

where P_{ik} is the probability that $\mathbf{x}^{(i)}$ is generated by component C_k .

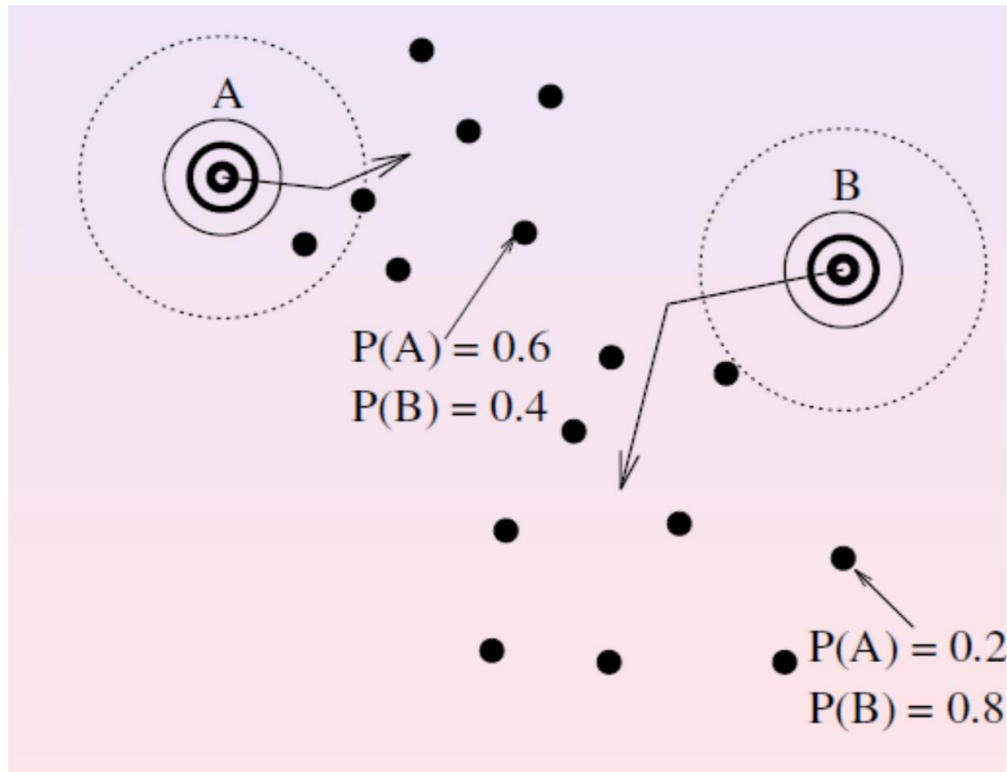
Gaussian Mixture Models: Algorithm

- **E-Step:** for each point, estimate the probability that each Gaussian generated it (compute P_{ik}).



Gaussian Mixture Models: Algorithm

- **M-Step:** modify the parameters (w_k, μ_k, Σ_k) to maximize the likelihood of the data



Gaussian Mixture Models: Algorithm

- Let $M_k = \sum_{i=1}^m P_{ik}$, i.e., the sum of the probabilities of each example to the k th component—this is the effective number of data points assigned to component k .

$$w_k^{new} = \left(\frac{M_k}{m} \right)$$

- The updated mean is calculated in a manner similar to how we could compute a standard empirical average of the probability of each i th data vector $\mathbf{x}^{(i)}$ to the k th component.

$$\mu_k^{new} = \left(\frac{1}{M_k} \right) \sum_{i=1}^m P_{ik} \mathbf{x}^{(i)}$$

$$\Sigma_k^{new} = \left(\frac{1}{M_k} \right) \sum_{i=1}^m P_{ik} (\mathbf{x}^{(i)} - \mu_k^{new})(\mathbf{x}^{(i)} - \mu_k^{new})^T$$