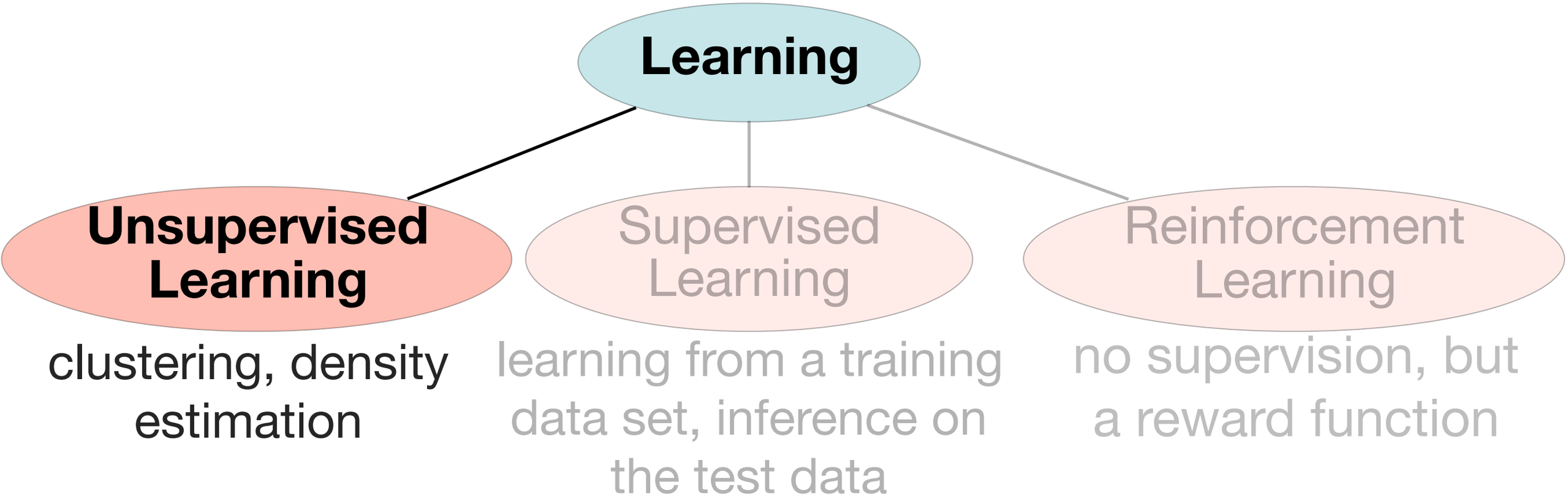
6. Clustering

Motivation

- Supervised learning is good for interaction with humans, but labels from a supervisor are sometimes hard to obtain
- Clustering is **unsupervised** learning, i.e. it tries to learn only from the data
- Main idea: find a similarity measure and group similar data objects together
- Clustering is a very old research field, many approaches have been suggested
- Main problem in most methods: how to find a good number of clusters

Categories of Learning



In unsupervised learning, there is no **ground truth** information given.

Most Unsupervised Learning methods are based on **Clustering**.

K-means Clustering

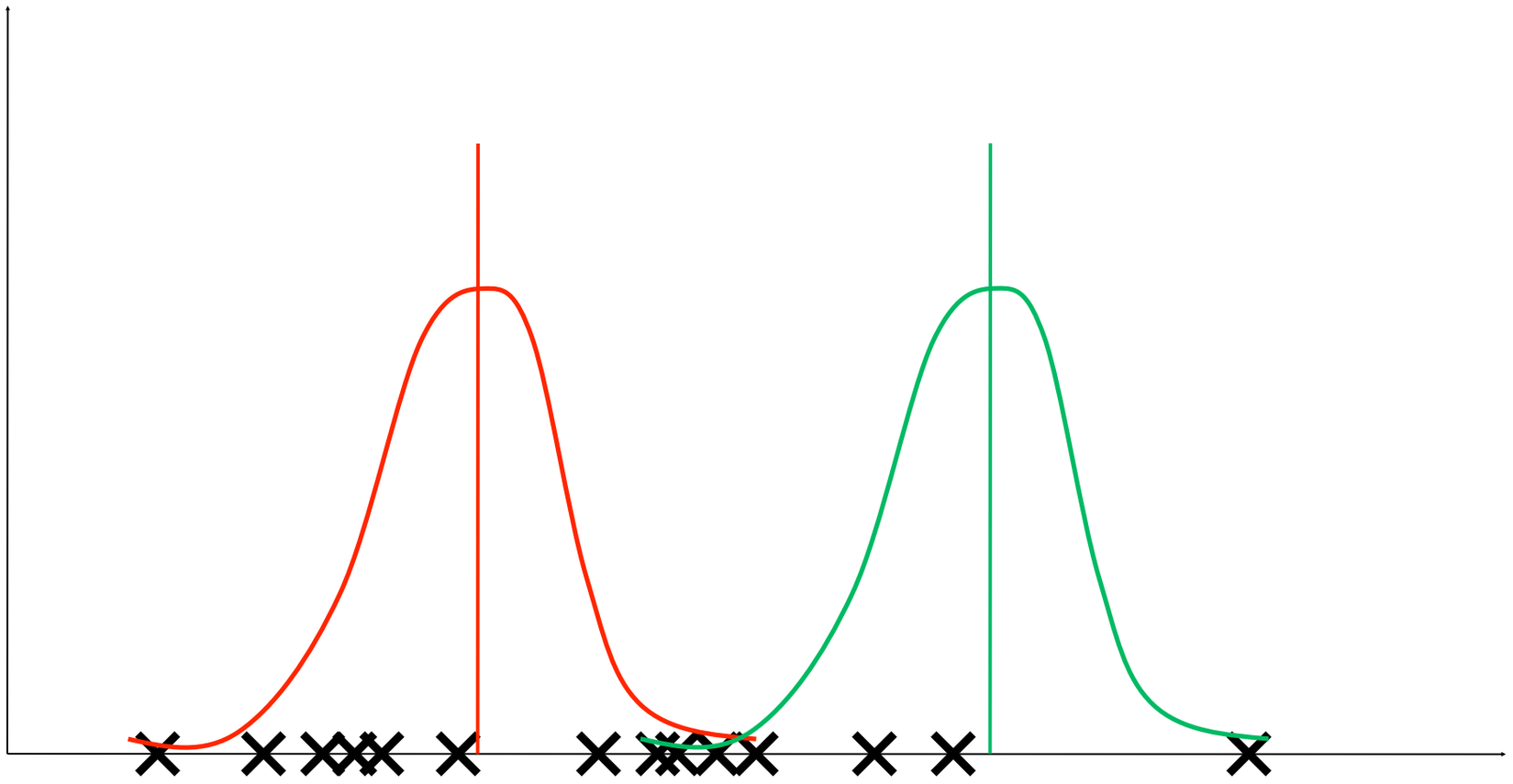
- Given: data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, number of clusters K
- Goal: find cluster centers $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ so that

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

  is minimal, where $r_{nk} = 1$ if $\mathbf{x}_n$ is assigned to $\boldsymbol{\mu}_k$
- Idea: compute r_{nk} and $\boldsymbol{\mu}_k$ iteratively
- Start with some values for the cluster centers
- Find optimal assignments r_{nk}
- Update cluster centers using these assignments
- Repeat until assignments or centers don't change

K-means Clustering

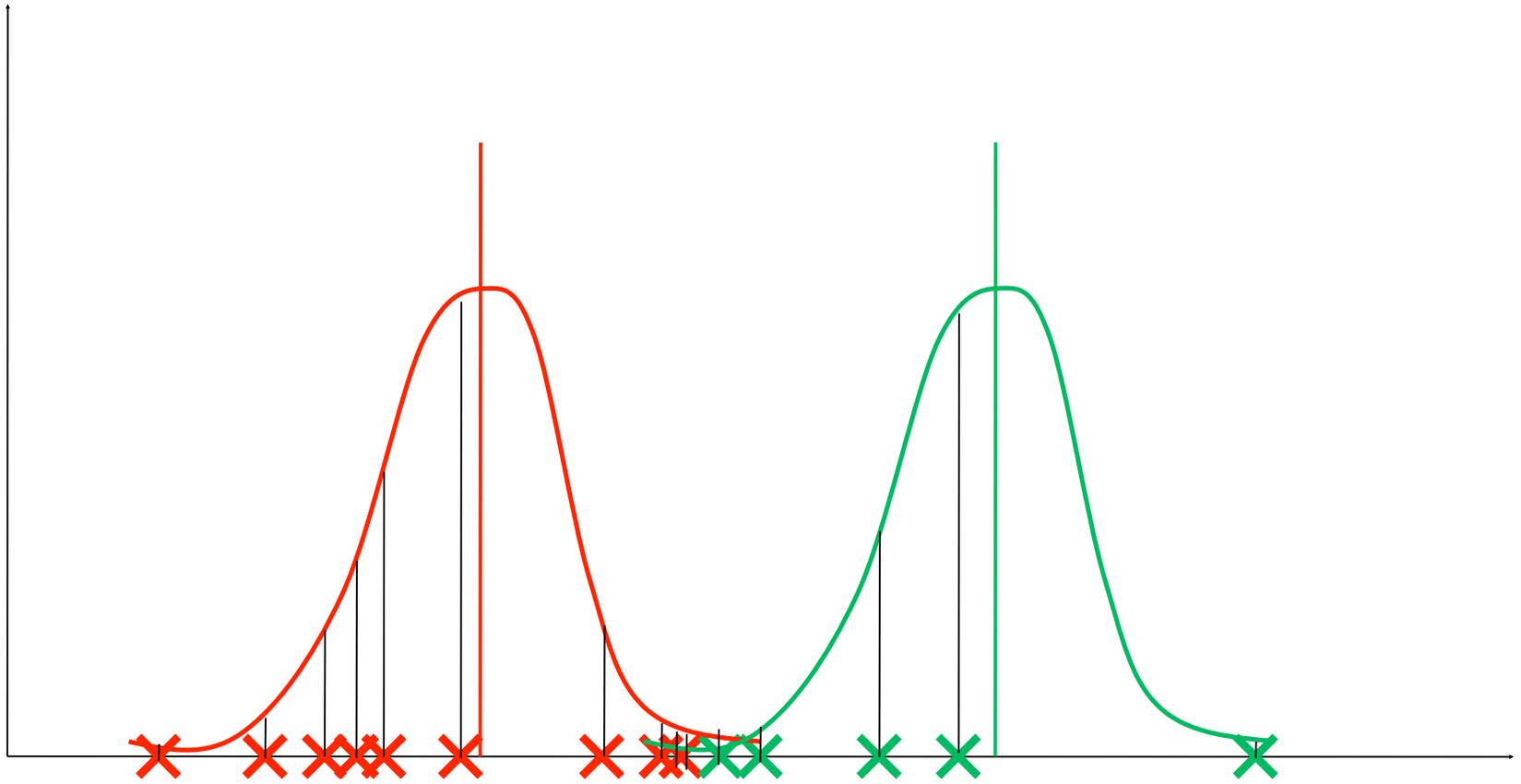
Initialize cluster means: $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$



K-means Clustering

Find optimal assignments:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\| \\ 0 & \text{otherwise} \end{cases}$$

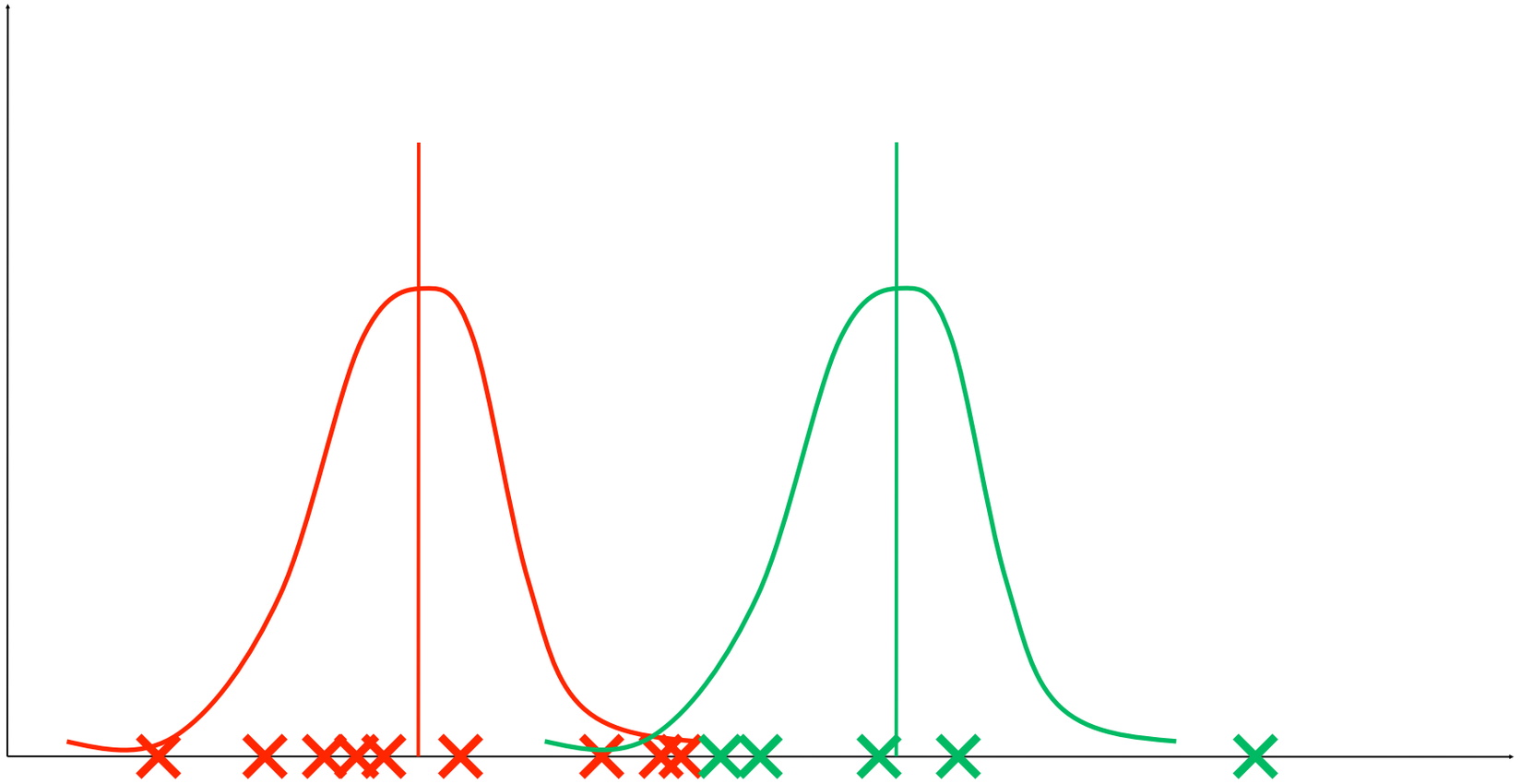


K-means Clustering

Find new optimal means:

$$\frac{\partial J}{\partial \boldsymbol{\mu}_k} = 2 \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) \stackrel{!}{=} 0$$

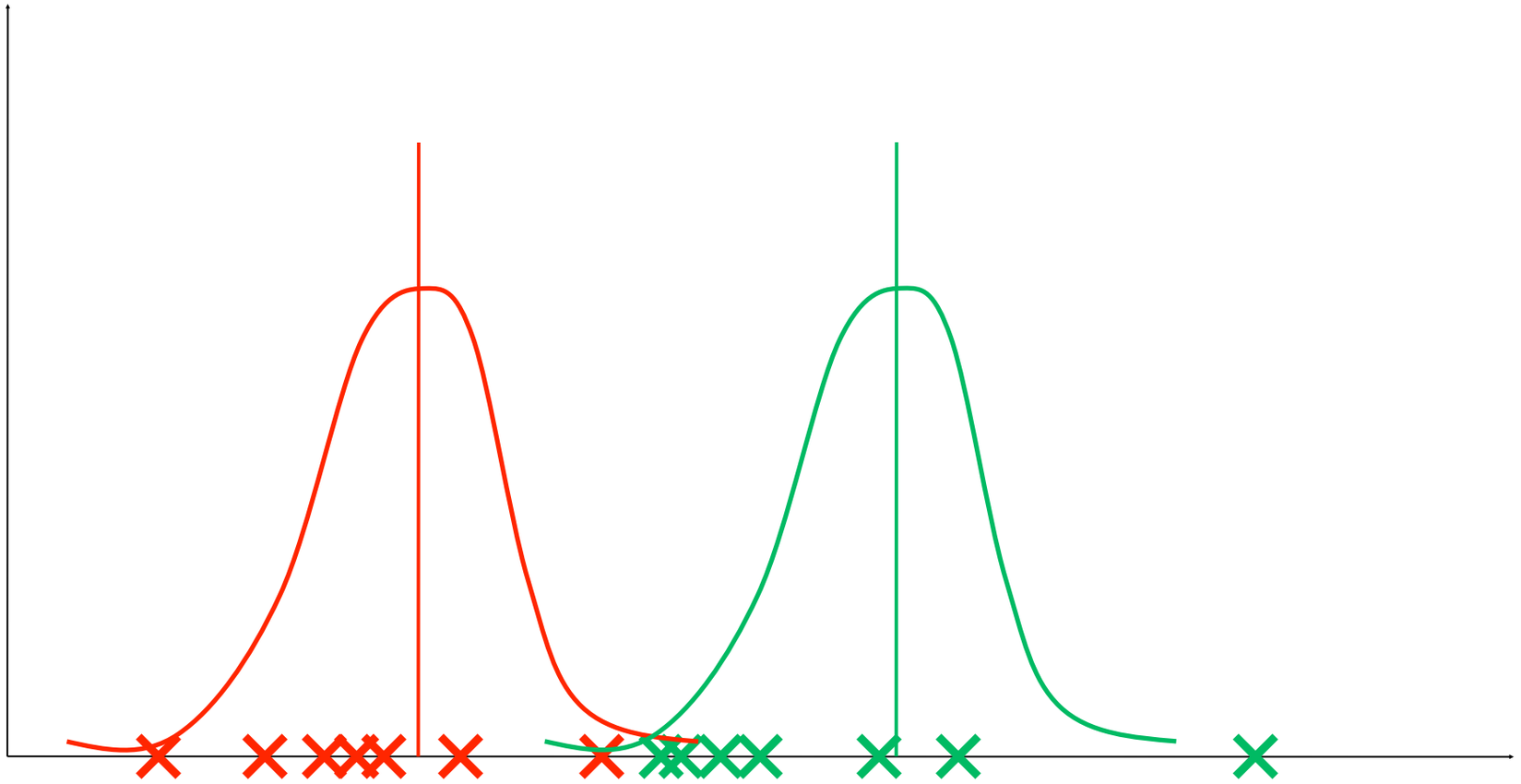
$$\Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}}$$



K-means Clustering

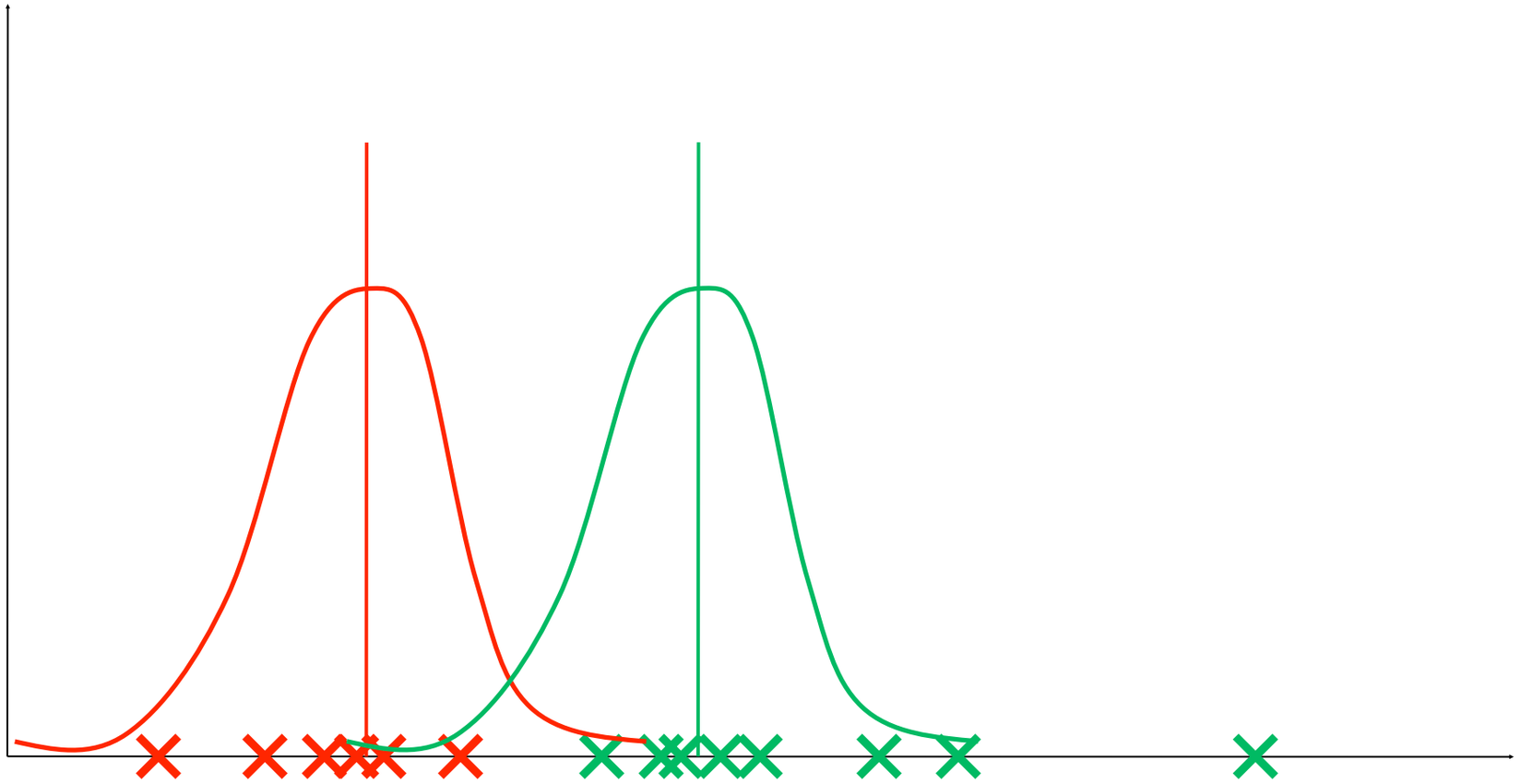
Find new optimal assignments:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\| \\ 0 & \text{otherwise} \end{cases}$$

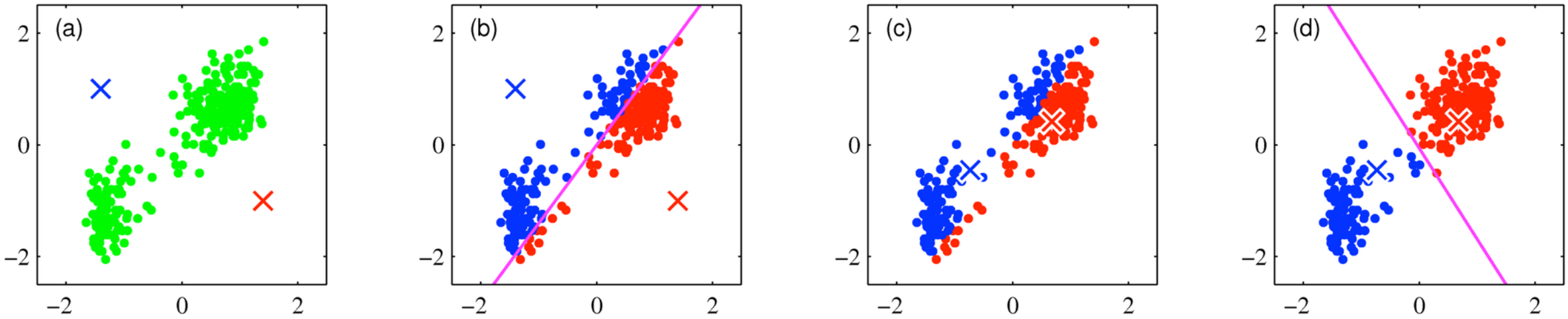


K-means Clustering

Iterate these steps until means and
assignments do not change any more



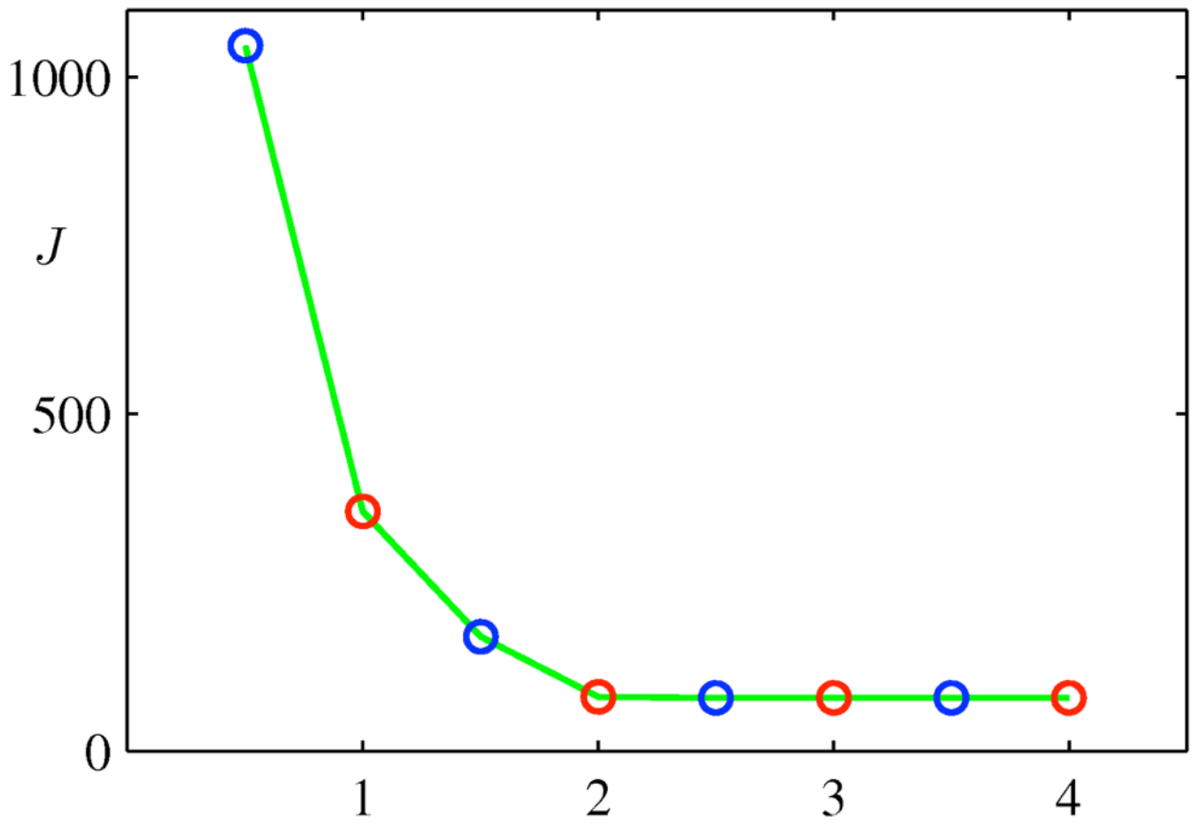
2D Example



- Real data set
- Random initialization

- Magenta line is “decision boundary”

The Cost Function



- After every step the cost function J is minimized
- Blue steps: update assignments
- Red steps: update means
- Convergence after 4 rounds

K-means for Segmentation

$K = 2$

$K = 3$

$K = 10$

Original image

K-Means: Additional Remarks

- K-means converges always, but the minimum is not guaranteed to be a global one
- There is an **online** version of K-means
  - After each addition of $\mathbf{x}_n$, the nearest center $\boldsymbol{\mu}_k$ is updated:

$$\boldsymbol{\mu}_k^{\text{new}} = \boldsymbol{\mu}_k^{\text{old}} + \eta_n(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{old}})$$

- The ***K*-medoid** variant:
  - Replace the Euclidean distance by a general measure *V*.

$$\tilde{J} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

Mixtures of Gaussians

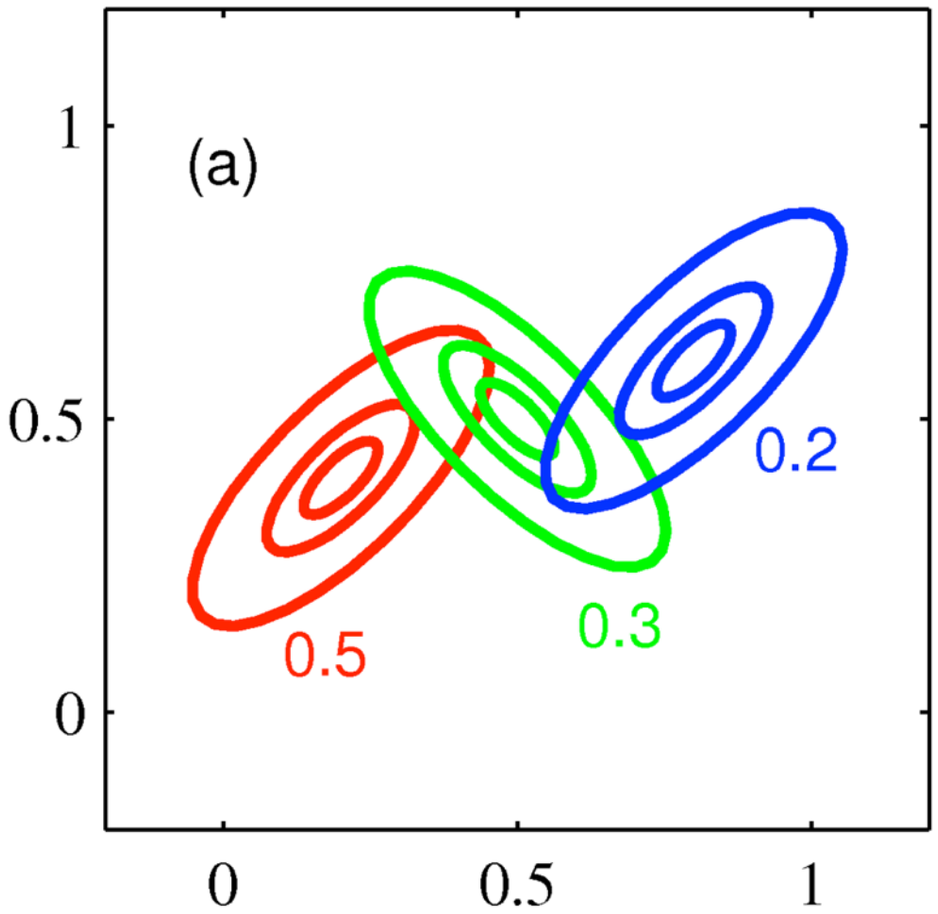
- Assume that the data consists of K clusters
- The data within each cluster is Gaussian
- For any data point $\mathbf{x}$ we introduce a K-dimensional binary random variable $\mathbf{z}$ so that:

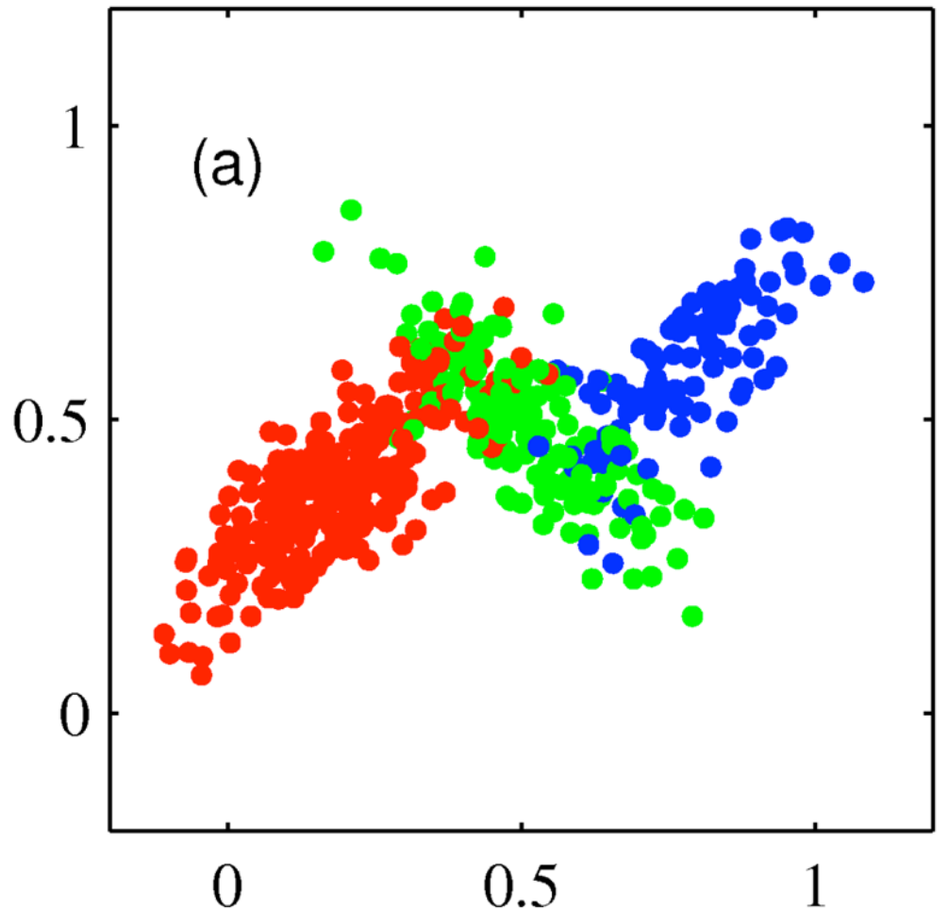
$$p(\mathbf{x}) = \sum_{k=1}^{K} \underbrace{p(z_k = 1)}_{=:\pi_k} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$$

where

$$z_k \in \{0, 1\}, \quad \sum_{k=1}^{K} z_k = 1$$

A Simple Example





- Mixture of three Gaussians with mixing coefficients
- Left: all three Gaussians as contour plot
- Right: samples from the mixture model, the red component has the most samples

Parameter Estimation

- From a given set of training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ we want to find parameters $(\pi_{1,\ldots,K}, \boldsymbol{\mu}_{1,\ldots,K}, \Sigma_{1,\ldots,K})$ so that the likelihood is maximized (MLE):

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N \mid \pi_{1,\ldots,K}, \boldsymbol{\mu}_{1,\ldots,K}, \Sigma_{1,\ldots,K}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)$$

or, applying the logarithm:

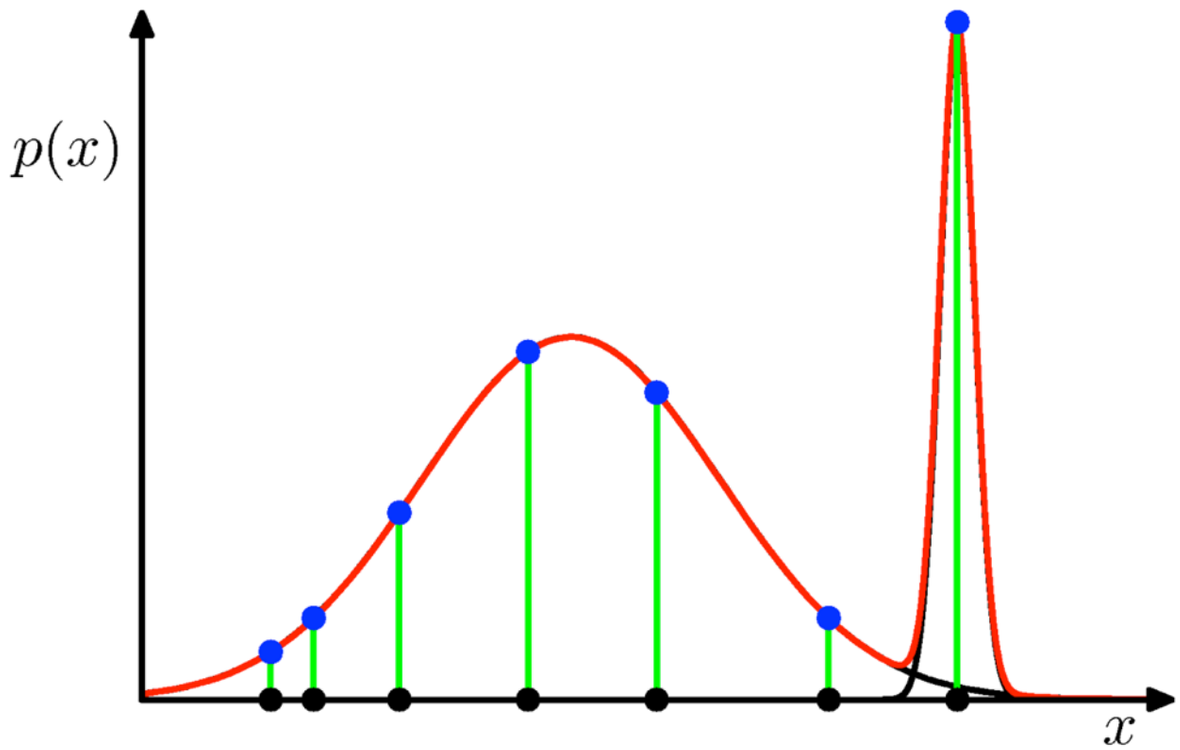
$$\log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)$$

- However: this is not as easy as maximum-likelihood for single Gaussians!

Problems with MLE for Gaussian Mixtures

- Assume that for one k the mean $\boldsymbol{\mu}_k$ is exactly at a data point $\mathbf{x}_n$
  - For simplicity: assume that $\Sigma_k = \sigma_k^2 I$
  - Then: $$\mathcal{N}(\mathbf{x}_n \mid \mathbf{x}_n, \sigma_k^2 I) = \frac{1}{\sqrt{2\pi}\sigma_k^D}$$
  - This means that the overall log-likelihood can be maximized arbitrarily by letting $\sigma_k \to 0$ (**overfitting**)
- Another problem is the **identifiability**:
  - The order of the Gaussians is not fixed, therefore:
  - There are $K!$ equivalent solutions to the MLE problem

Overfitting with MLE for Gaussian Mixtures



- One Gaussian fits exactly to one data point
- It has a very small variance, i.e. contributes strongly to the overall likelihood
- In standard MLE, there is no way to avoid this!

Expectation-Maximization

- EM is an elegant and powerful method for MLE problems with latent variables
- Main idea: model parameters and latent variables are estimated iteratively, where average over the latent variables (expectation)
- A typical example application of EM is the Gaussian Mixture model (GMM)
- However, EM has many other applications
- First, we consider EM for GMMs

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n) \qquad z_{nk} \in \{0, 1\}$$

$$\sum_k z_{nk} = 1$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

- Next, we derive the log-likelihood wrt. to $\boldsymbol{\mu}_k$:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} \stackrel{!}{=} \mathbf{0}$$

Expectation-Maximization for GMM

- First, we define the **responsibilities:**

$$\gamma(z_{nk}) = p(z_{nk} = 1 \mid \mathbf{x}_n)$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

- Next, we derive the log-likelihood wrt. to $\boldsymbol{\mu}_k$:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} \stackrel{!}{=} \mathbf{0}$$

and we obtain:

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

Expectation-Maximization for GMM

- We can do the same for the covariances:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \Sigma_k} \stackrel{!}{=} \mathbf{0}$$

and we obtain:

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

- Finally, we derive wrt. the mixing coefficients π_k :

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \pi_k} \stackrel{!}{=} \mathbf{0} \quad \text{where:} \quad \sum_{k=1}^{K} \pi_k = 1$$

Expectation-Maximization for GMM

- We can do the same for the covariances:

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \Sigma_k} \stackrel{!}{=} \mathbf{0}$$

and we obtain:

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

- Finally, we derive wrt. the mixing coefficients π_k :

$$\frac{\partial \log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \pi_k} \stackrel{!}{=} \mathbf{0} \quad \text{where:} \quad \sum_{k=1}^{K} \pi_k = 1$$

and the result is:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk})$$

Algorithm Summary

1. Initialize means μ_k covariance matrices Σ_k and mixing coefficients π_k
2. Compute the initial log-likelihood $\log p(X \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$
3. **E-Step.** Compute the responsibilities:

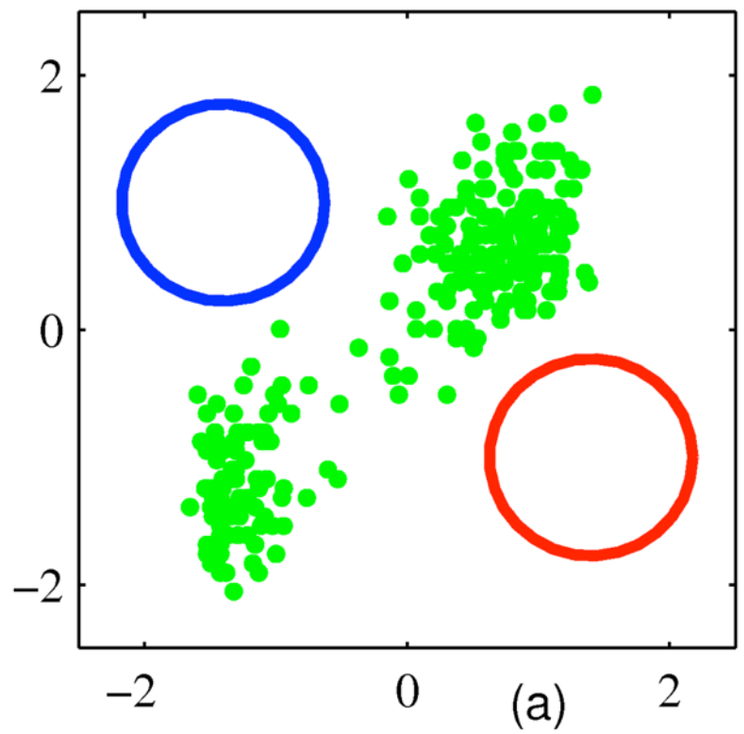
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

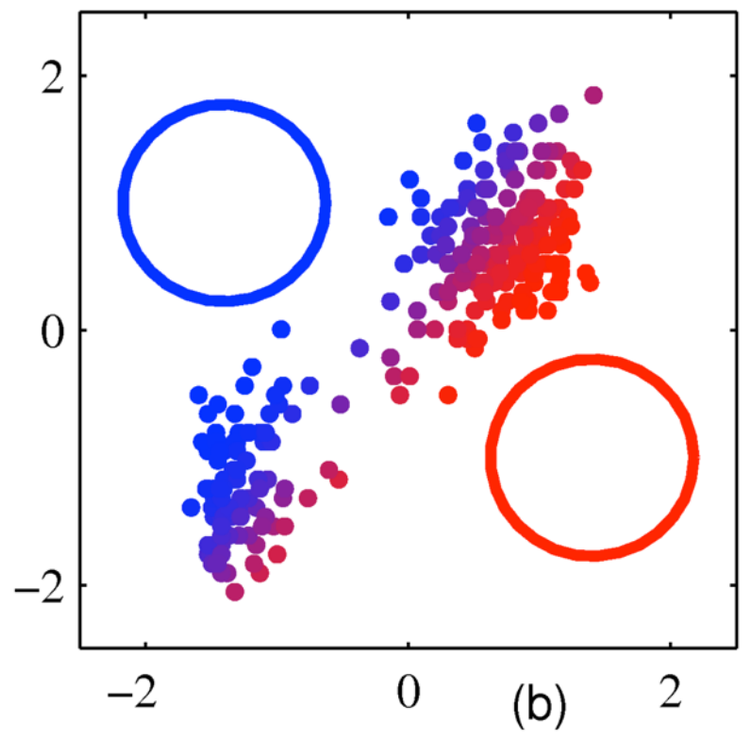
4. **M-Step.** Update the parameters:

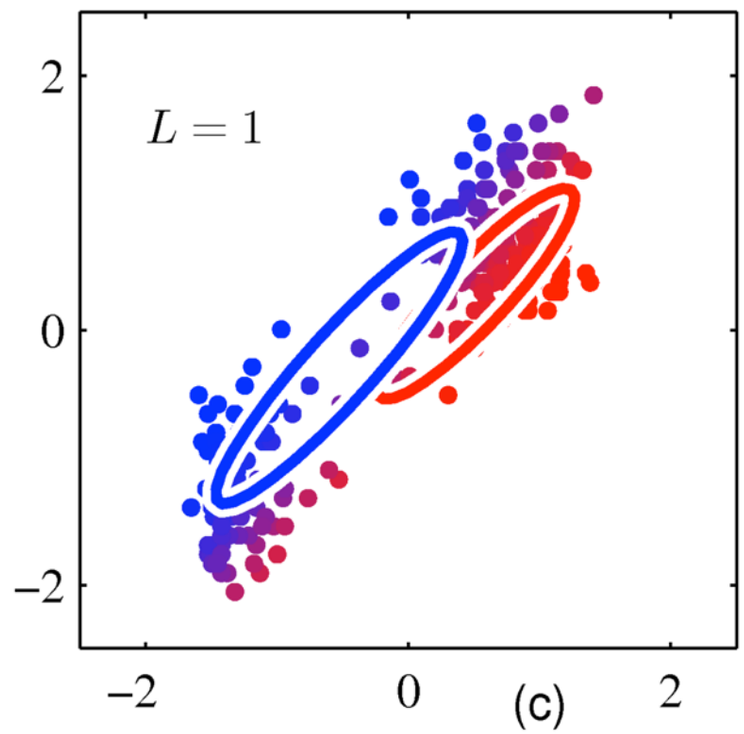
$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \quad \Sigma_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T}{\sum_{n=1}^{N} \gamma(z_{nk})} \quad \pi_k^{\text{new}} = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk})$$

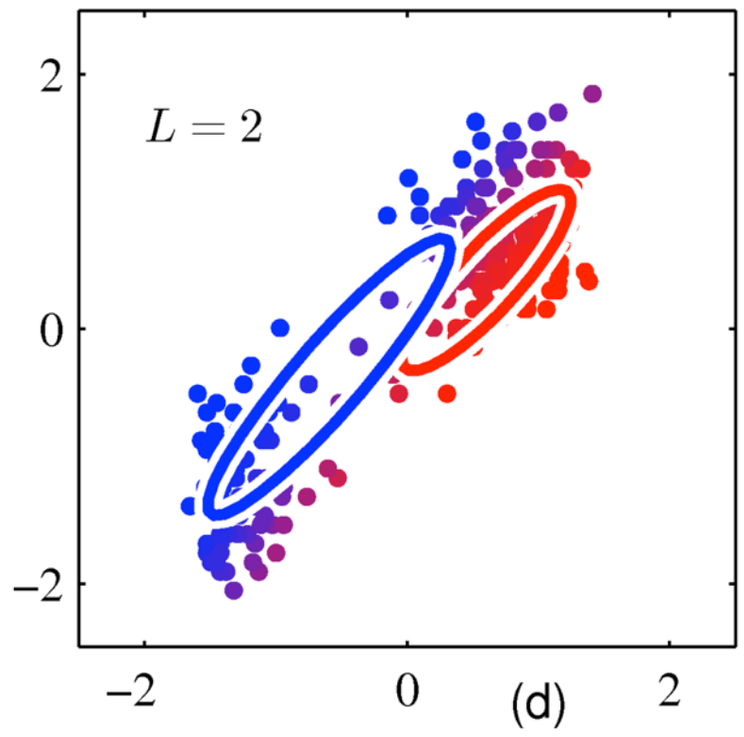
5. Compute log-likelihood; if not converged go to 3.

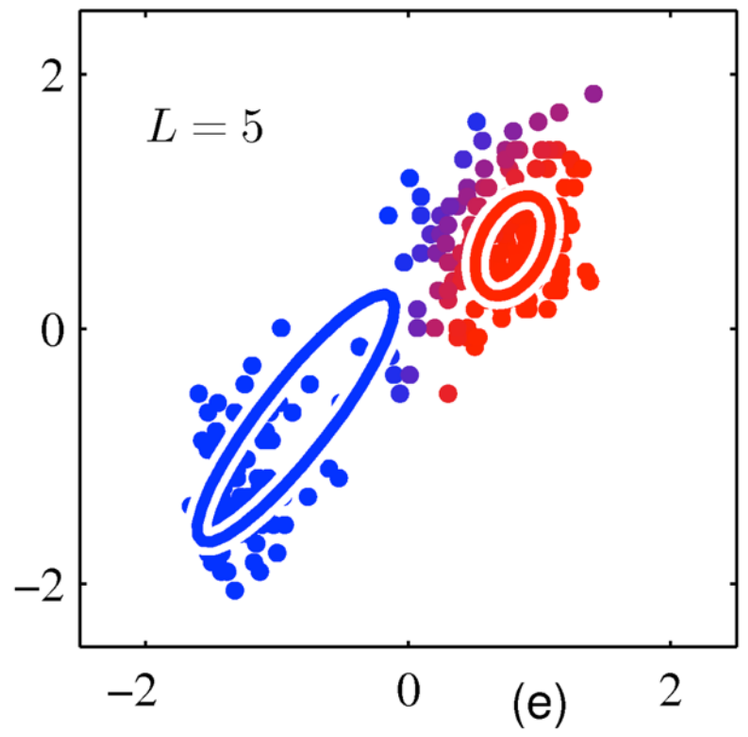
The Same Example Again

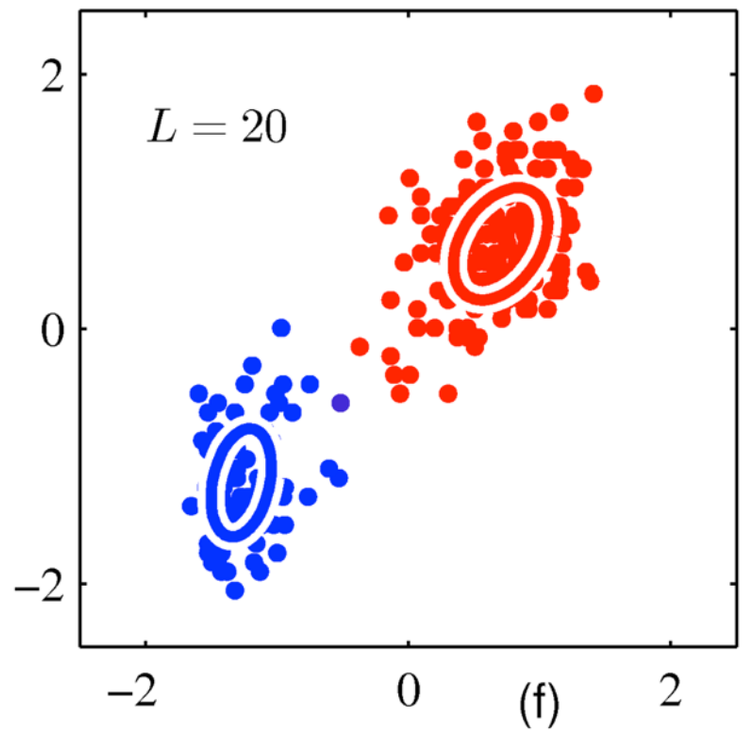










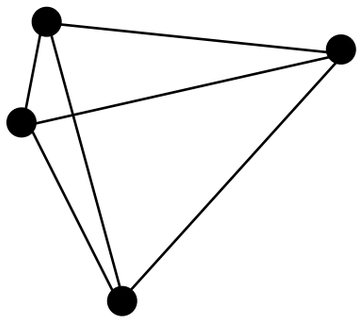


Observations

- Compared to K-means, points can now belong to both clusters (**soft assignment**)
- In addition to the cluster center, a covariance is estimated by EM
- Initialization is the same as used for K-means
- Number of iterations needed for EM is much higher
- Also: each cycle requires much more computation
- Therefore: start with K-means and run EM on the result of K-means (covariances can be initialized to the sample covariances of K-means)
- EM only finds a **local** maximum of the likelihood!

Spectral Clustering

- Consider an undirected graph that connects all data points
- The edge weights are the similarities (“closeness”)
- We define the weighted degree d_i of a node as the sum of all outgoing edges



$$W = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

$$d_i = \sum_{j=1}^{N} w_{ij}$$

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & d_3 & \\ & & & d_4 \end{bmatrix}$$

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

- This matrix has the following properties:
  - the 1 vector is eigenvector with eigenvalue 0

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

- This matrix has the following properties:
  - the 1 vector is eigenvector with eigenvector 0
  - the matrix is symmetric and positive semi-definite

Spectral Clustering

- The Graph Laplacian is defined as:

$$L = D - W$$

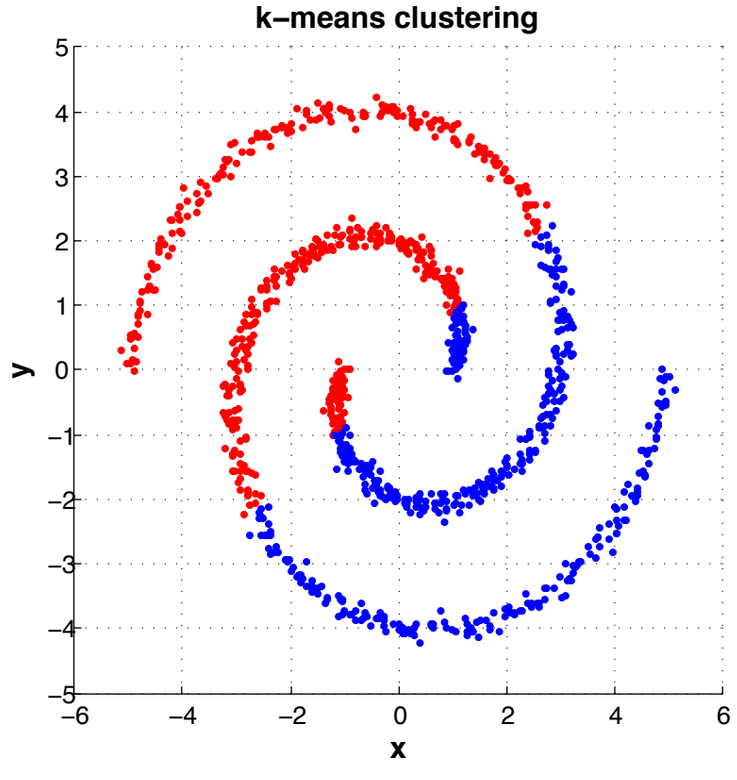
- This matrix has the following properties:
  - the 1 vector is eigenvector with eigenvector 0
  - the matrix is symmetric and positive semi-definite
- With these properties we can show:

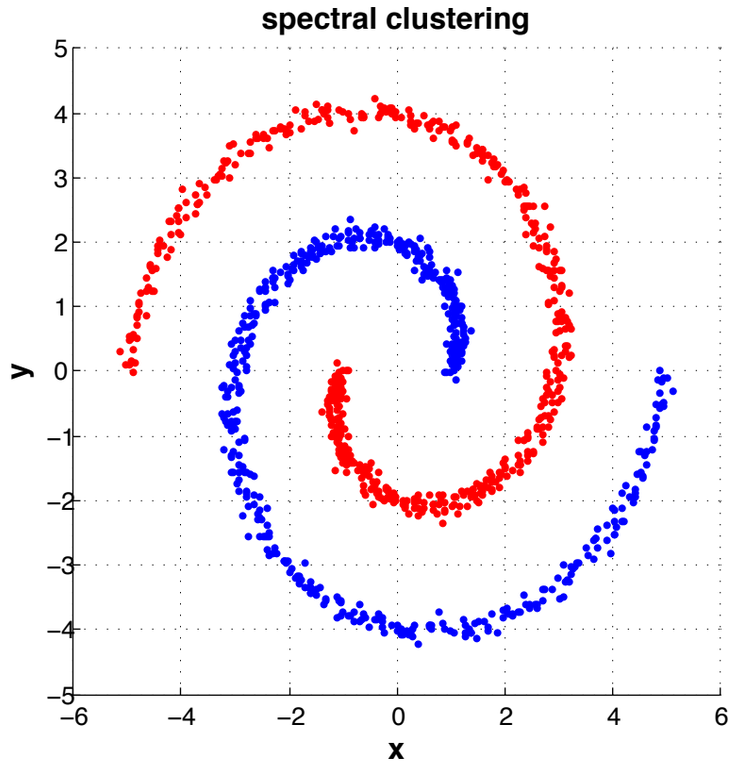
Theorem: The set of eigenvectors of L with eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_K}$, where A_k are the K connected components of the graph.

The Algorithm

- Input: Similarity matrix W
- Compute L = D - W
- Compute the eigenvectors that correspond to the K smallest eigenvalues
- Stack these vectors as columns in a matrix U
- Treat each row of U as a K-dim data point
- Cluster the N rows with K-means clustering
- The indices of the rows that correspond to the resulting clusters are those of the original data points.

An Example





- Spectral clustering can handle complex problems such as this one
- The complexity of the algorithm is $O(N^3)$, because it has to solve an eigenvector problem
- But there are efficient variants of the algorithm

Further Remarks

- To account for nodes that are highly connected, we can use a normalized version of the graph Laplacian
- Two different methods exist:
  - $L_{rw} = D^{-1}L = I - D^{-1}W$
  - $L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$
- These have similar eigenspaces than the original Laplacian L
- Clustering results tend to be better than with the unnormalized Laplacian

Summary

- Several Clustering methods exist:
  - K-means clustering and Expectation-Maximization, both based on Gaussian Mixture Models
  - K-means uses hard assignments, whereas EM uses soft assignments and estimates also the covariances
  - Spectral clustering uses the graph Laplacian and performs an eigenvector analysis
- Major Problem:
  - most clustering algorithms require the number of clusters to be given