UNIVERSITÄT BONN

Clustering and Density Estimation

Elena Trunz

September 20, 2023

Uni Bonn

Overview

Introduction

K-means

Gaussian Mixture Models

Expectation Maximization

Introduction

Unsupervised learning

- Sometimes our data consists of input vectors without corresponding target labels.
  - $\Rightarrow$ Unlabled data
- Unsupervised learning is concerned with discovering structure in unlabeled data.
- Goals:
  - $\rightarrow$ Discover groups of similar examples within the data (*Clustering*)
  - $\rightarrow$ Determine the distribution of data within the input space (*Density estimation*)
  - $\rightarrow$ Project the data onto a lower dimensional space (*Dimensionality reduction*)

What is Clustering?

- *Clustering* is the process of organizing a set of physical or abstract objects into classes (called *clusters*), such that there is
  - High intra-class similarity
  - Low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally: Finding natural groupings among objects.

What is a natural grouping among these objects?

What is a natural grouping among these objects?

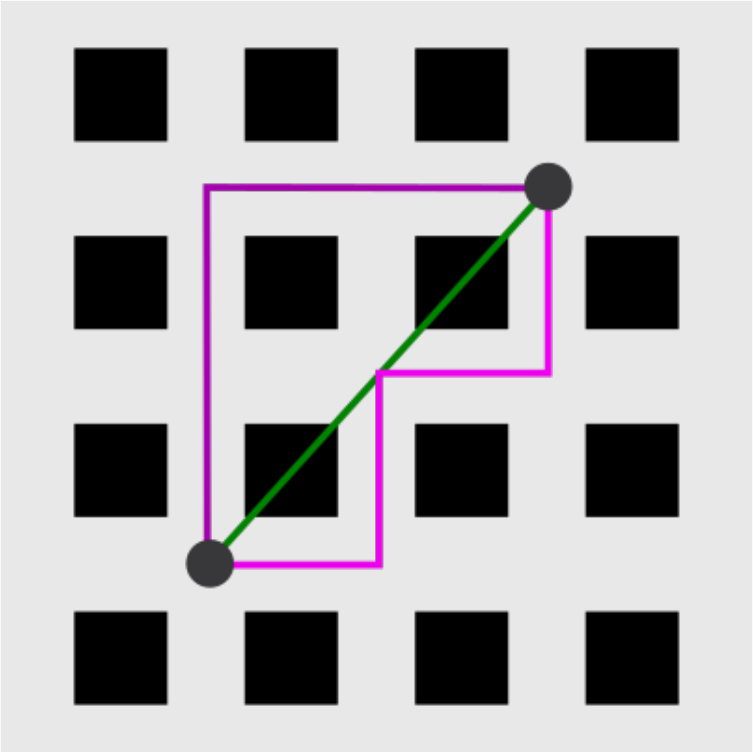
Clustering is subjective

What is similarity?

Definition: Let U be the universe of possible objects. The *distance* (*dissimilarity, metric*) on U is a function $d : U \times U \to \mathbb{R}$ that satisfies the following conditions (axioms):

$d(a, b) = 0 \Leftrightarrow a = b$	*Identity of indiscernibles*
$d(a, b) = d(b, a)$	*Symmetry*
$d(a, c) \leq d(a, b) + d(b, c)$	*Triangle inequality*

L1 vs. L2 distance



L2 on images

Distance metrics on pixels are not informative!

All 3 images have same L2 distance to the one on the left

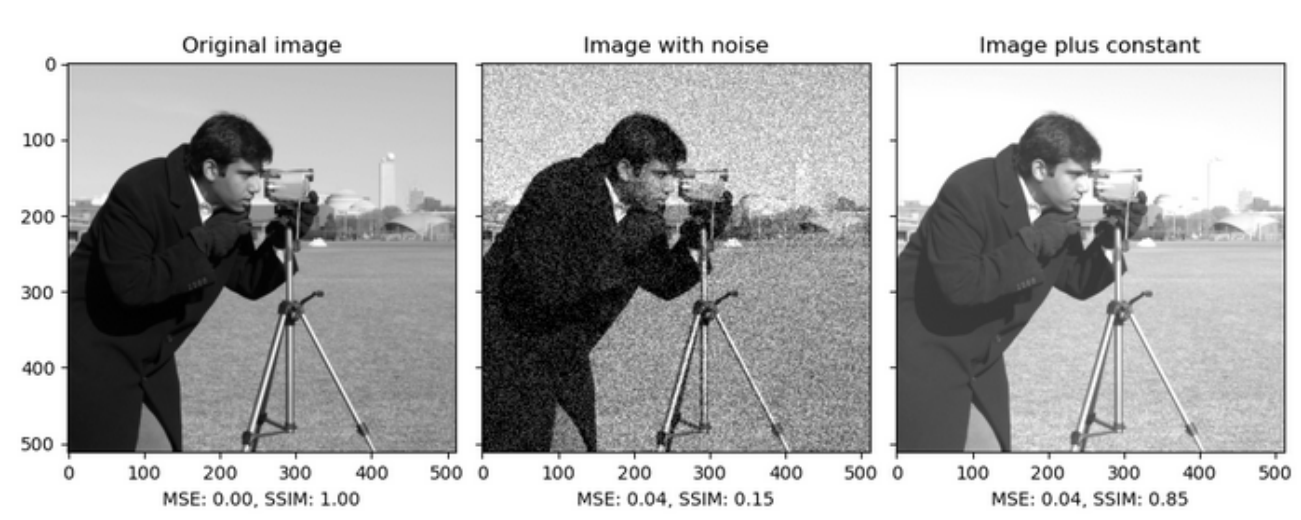
Other distance measures

Data dependent!

- Images:
  - Structural similarity (SSIM)
  - Histogram of Oriented Gradients (HOG)
  - Cosine similarity
  - Normalized Cross-Correlation
- Shape based data: Chamfer distance
- Sphere: Haversine formula
- Text: Edit distance, Cosine similarity
- Many more...

SSIM

- Measures percieved similarity between images
- Measurement of image quality is based on an initial uncompressed image as a reference
- Compares luminance, contrast and structure



Clustering procedure

Input:

- Data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x} \in \mathbb{R}^m$ of unlabeled observations of random m-dimensional variable $\mathbf{x}$.

Procedure:

1. Make suitable assumptions (e.g. about the structure of the data)
2. Find optimal cluster representations with respect to the assumptions
3. Assign data points to the clusters

K-means

K-means: problem formulation

Assumptions:

- Euclidean distance as similarity measure
- Number K of clusters

Find:

- Set of cluster centers $\{\boldsymbol{\mu}_k\}_{k=1}^K$ we will write $\{\cdot\}$ instead of $\{\cdot\}_{k=1}^K$
- Assignment of data points from X to a cluster

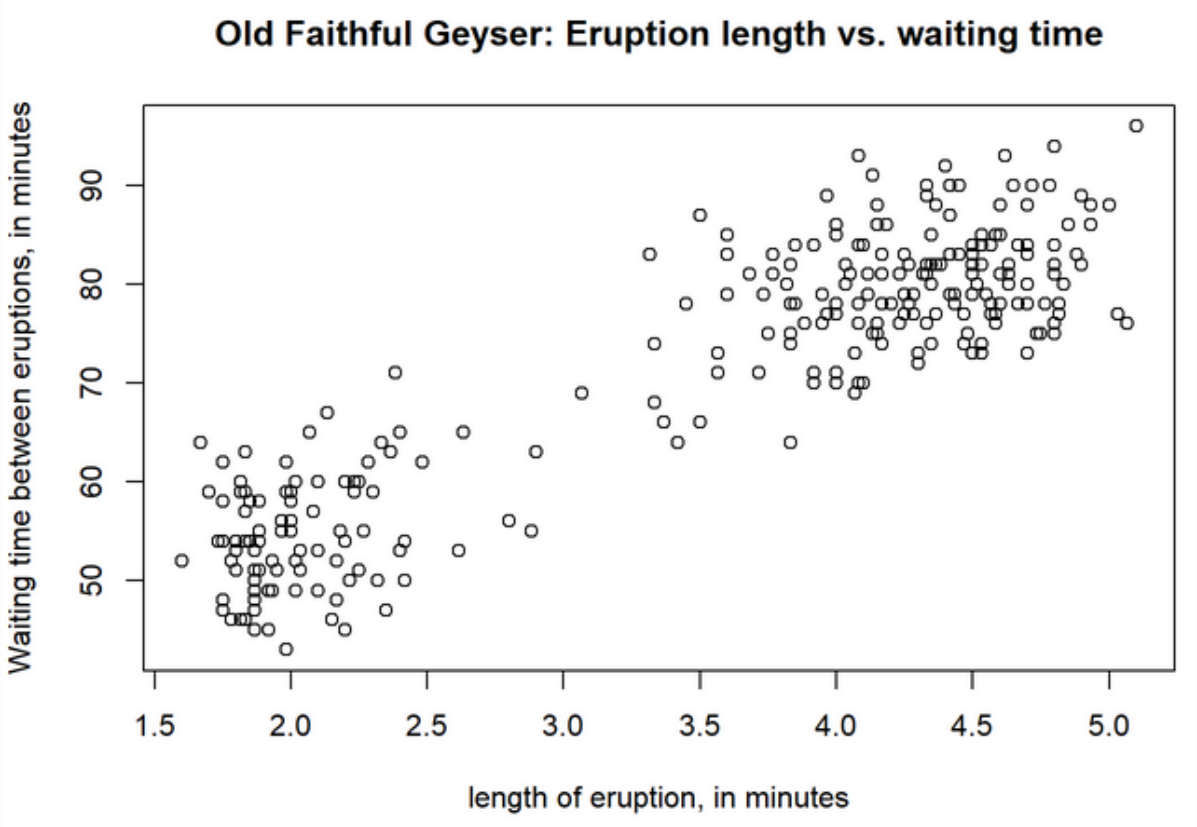
in order to minimize the following objective function:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2,$$

where $r_{ik} \in \{0, 1\}$ is a binary variable, called the assignment, which is 1 iff $\mathbf{x}_i$ is assigned to cluster k.

Example data set

- "Old Faithful" dataset
- 272 measurements of the eruption of the Old Faithful geyser at Yellowstone National Park (USA)



K-means algorithm:

How can we find the values for $\{r_{ik}\}$ and $\{\boldsymbol{\mu}_k\}$ so as to minimize

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2?$$

Idea: Choose some initial values for the $\{\boldsymbol{\mu}_k\}$. Then iterate between two steps until convergence:

1. Minimize J with respect to the $\{r_{ik}\}$, keeping the $\{\boldsymbol{\mu}_k\}$ fixed.
2. Minimize J with respect to the $\{\boldsymbol{\mu}_k\}$, keeping the $\{r_{ik}\}$ fixed.

Updating $\{r_{ik}\}$ corresponds to the E (expectation) step and updating $\{\boldsymbol{\mu}_k\}$ corresponds to the M (maximization) step of the *Expectation-Maximization* algorithm (will be discussed later).

E step: determination of the $\{r_{ik}\}$

- J is a linear function of r_{ik}
  - $\rightarrow$ Easy to give a closed form solution
- The terms involving different i are independent, so we can optimize for each data point separately
  - $\rightarrow$ Simply assign each data point to the closest cluster center
- Formally:

$$r_{ik} = \begin{cases} 1 \text{ if } k = \arg\min_j ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2 \\ 0 \text{ otherwise.} \end{cases}$$

M step: optimization of the $\{\boldsymbol{\mu}_k\}$

- J is a quadratic function of $\boldsymbol{\mu}_k$
- To find the minimum we compute the derivative with respect to $\boldsymbol{\mu}_k$ and set it to zero:

$$2\sum_{i=1}^{N} r_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k) = 0$$

- Solving this equation for $\boldsymbol{\mu}_k$ gives

$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}} = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik}\mathbf{x}_i,$$

where N_k = number of points assigned to cluster k.

$\rightarrow$ $\boldsymbol{\mu}_k$ = mean of all data points $\mathbf{x}_i$ assigned to cluster k

$\rightarrow$ For this reason it is called *K-means* algorithm.

K-means

- E step and M step are repeated until there is no further change in the assignments (or some maximum number of iterations is reached)
- Algorithm converges, because each step reduces the value of the objective function J
- However it may converge to a local minimum of J instead of global one

K-means: preprocessing

Features (dimensions) can have different scales

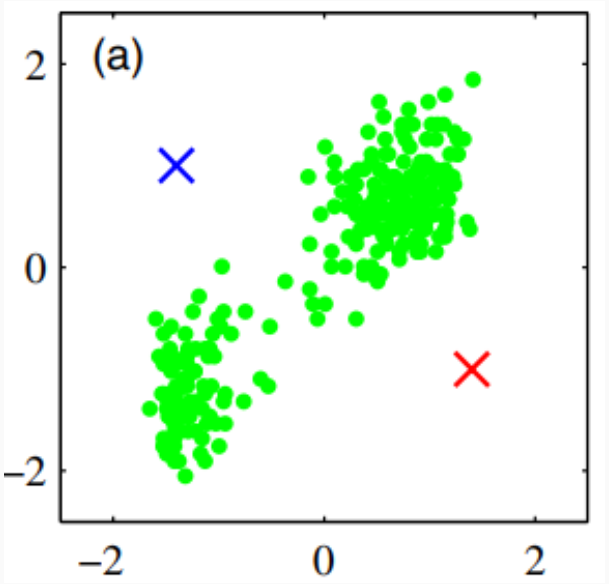
$\rightarrow$ One or more dimensions will dominate the distance calculations

$\rightarrow$ It is important to preprocess the data

Standardization: linear re-scaling of the data, so that it has zero mean and unit variance

Example: initialization

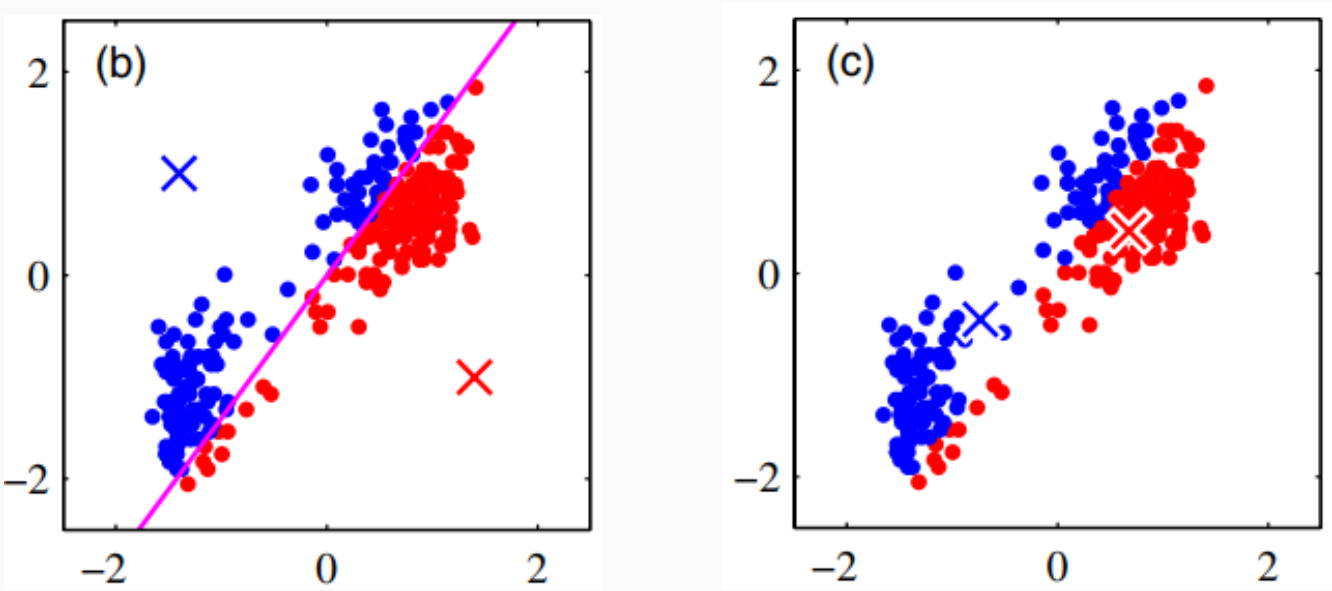
Standardized Old Faithful dataset:



Initialization with two means

Image was taken from [Bis06]

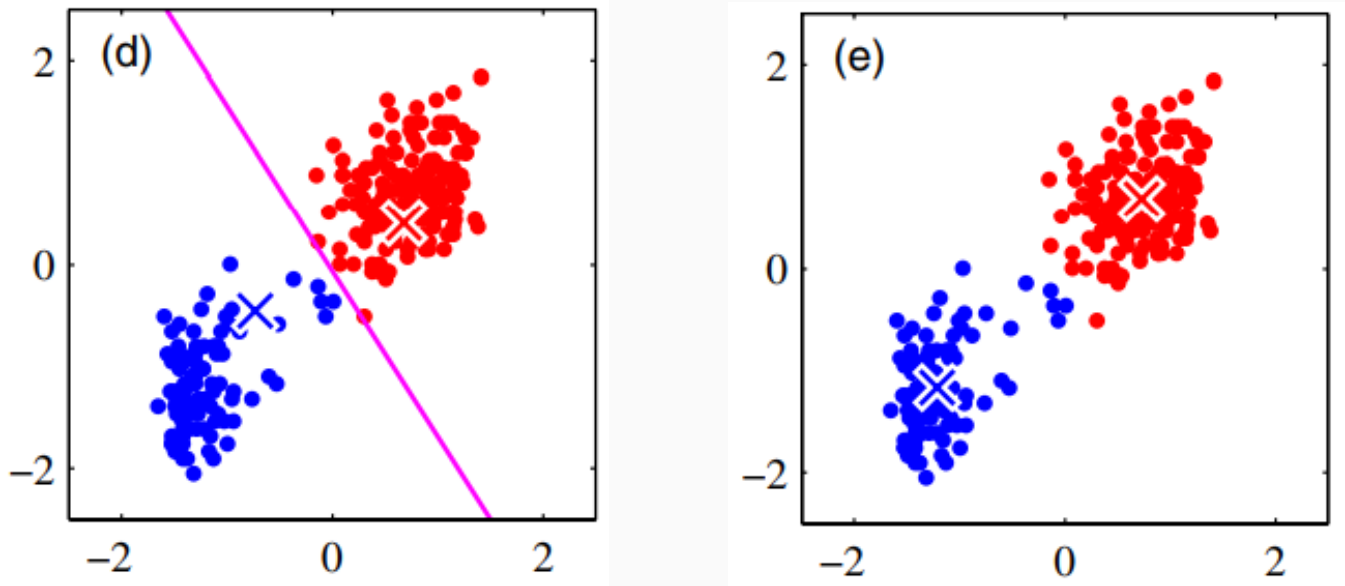
Example: first iteration



E step (left) and M step (right)

Image was taken from [Bis06]

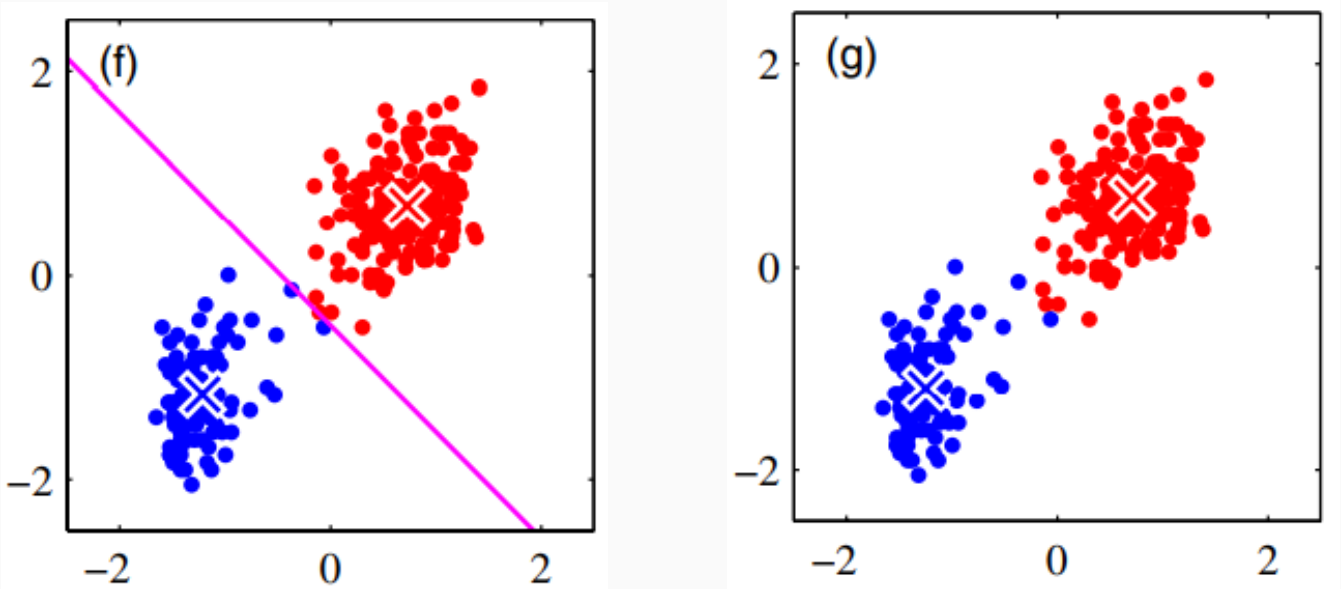
Example: second iteration



E step (left) and M step (right)

Image was taken from [Bis06]

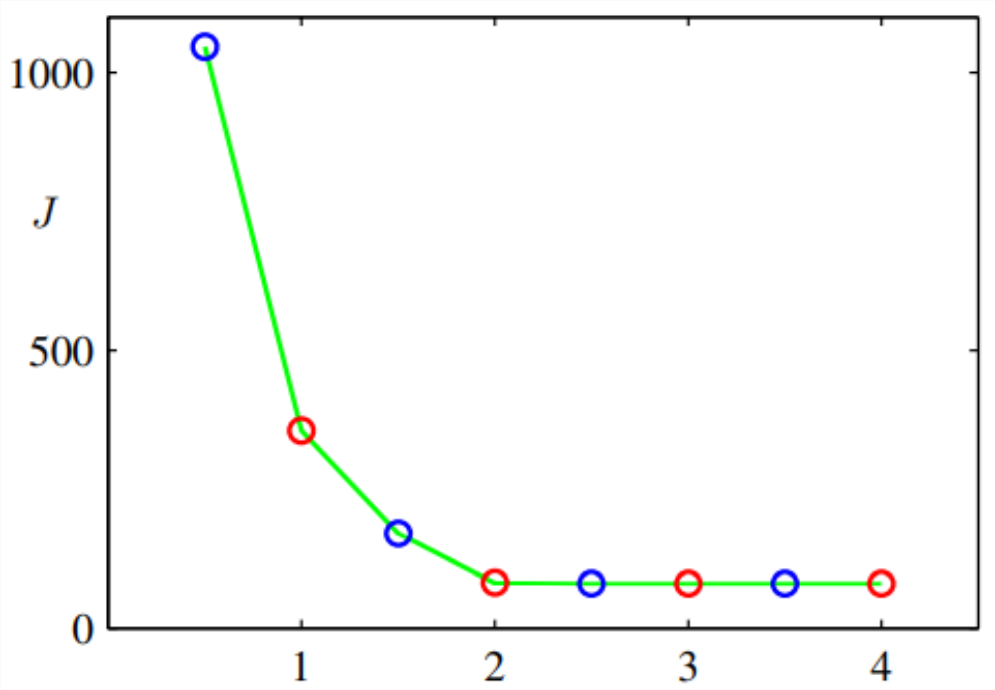
Example: third iteration



E step (left) and M step (right). Algorithm has converged, positions of the means didn't change.

Image was taken from [Bis06]

Example: objective function



Plot of the function J after each E step (blue points) and M step (red points).

Image was taken from [Bis06]

Initialization

- A good initialization of the cluster centers is to choose a random subset of K data points
- J is generally a non-convex function
  - $\rightarrow$ Run the algorithm with different initializations and post-select the best local minimum

K-means for segmentation

Image was taken from [Bis06]

Strengths and Weaknesses of the K-means method

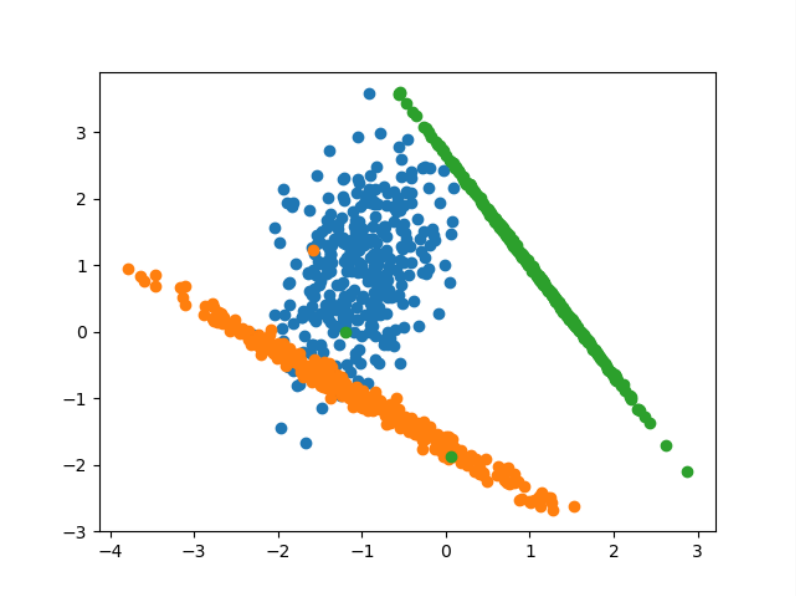
Strengths:

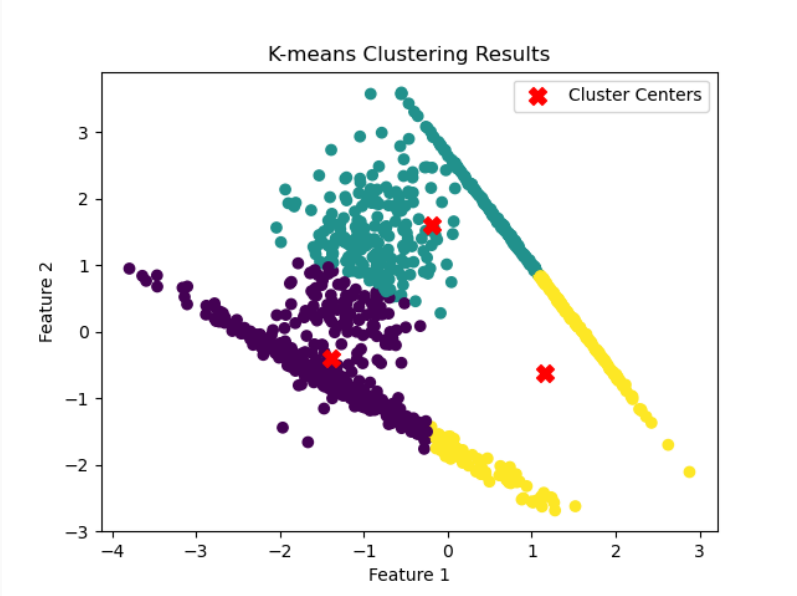
- Guaranteed to converge, although often to a local minimum
- Relatively efficient: computational complexity scales linearly in the size of the data set ($O(KN)$ per iteration)
  $\rightarrow$ Can deal with very large datasets

Drawbacks:

- Applicable only when mean is defined
  $\rightarrow$ Not suitable for categorical data
- Need to specify K in advance
- Non-robust to noisy data and outliers
- Not suitable for clusters, which have very different variances, since the underlying assumption is that all clusters have uniform variances

Limitation: different variances





K-means tries to build circular clusters, but the data distribution is elliptical.

Hard cluster assignments

- At each iteration K-means assigns each data point exactly to one of the clusters.
- There may be data points that lie roughly midway between cluster centers
- Idea: Use probabilistic approach to obtain "soft" assignments of data points to clusters and reflect the level of uncertainty over the most appropriate assignment

  $\rightarrow$ Mixture Models

Gaussian Mixture Models

Idea:

- Each point of the dataset X was drawn from an unknown distribution
- Assumption about the structure of the dataset: the data X was drawn from K Gaussians
- Each cluster is then characterized by some mean and variance
- Rather than identifying clusters by nearest centroids, fit a set of K Gaussians to the data

  $\rightarrow$ *Density estimation*

Multivariate Gaussian distribution

Recall: one-dimensional (univariate) Gaussian distribution:

$$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

with mean μ and variance σ^2.

Multivariate Gaussian for a m-dimensional vector $\mathbf{x}$:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m|\boldsymbol{\Sigma}|}} exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

$$\uparrow$$

determinant

with mean $\boldsymbol{\mu} \in \mathbb{R}^m$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times m}$.

Why do we need a set of Gaussians?

Often a simple Gaussian distribution is unable to capture the structure of a data set, whereas a linear superposition of several Gaussians gives a good characterization

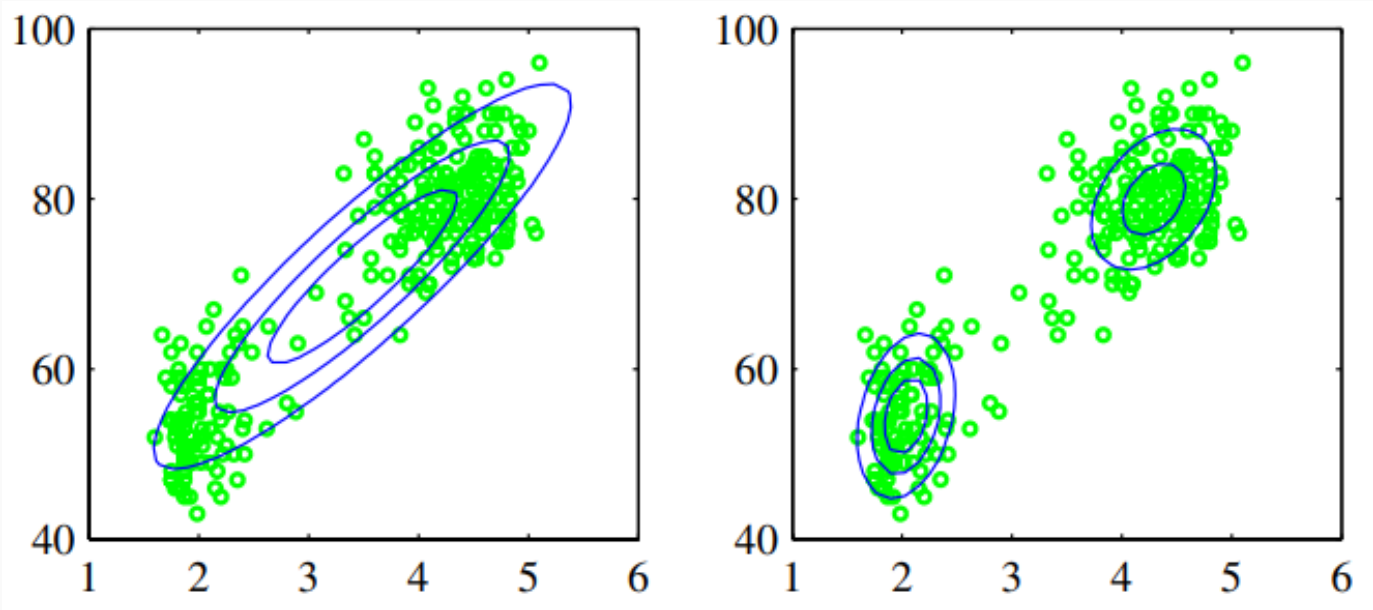


Image was taken from [Bis06]

Gaussian mixture distribution

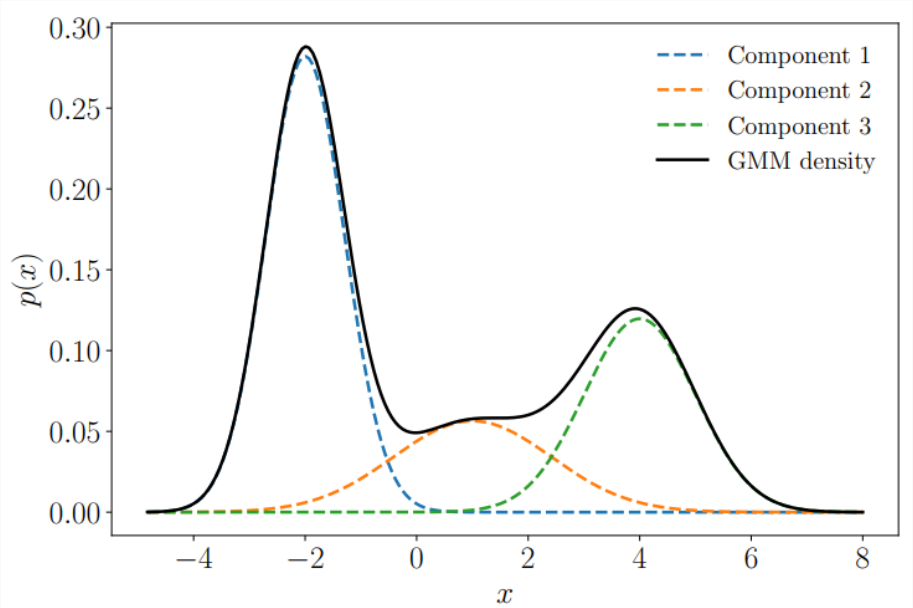
Linear superposition of K Gaussian densities is called *mixture of Gaussians* and has the form

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

The parameters π_k are called *mixture weights* or *mixing coefficients*.

Gaussian mixture distribution: example

With sufficient number of Gaussians (with adjusted means, covariances and weights), almost any continuous density can be approximated to arbitrary accuracy.



Weighted components and the mixture density, which is given as

$$p(x|\{\mu_k, \Sigma_k, \pi_k\}) = 0.5\mathcal{N}(x| -2, \frac{1}{2}) + 0.2\mathcal{N}(x|1, 2) + 0.3\mathcal{N}(x|4, 1)$$

Image was taken from [DFO20]

GMM for clustering

In order to use GMM for clustering we need to solve two problems

1. Find a good representation (approximation) of the real density by a GMM
2. Find cluster assignments

GMM for density estimation

Let π_k denote the probability that a data point is drawn from a mixture component k.

Then the probability of generating a point $\mathbf{x}$ in a GMM is given by

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

We can write likelihood of our dataset X as

$$L = p(\mathbf{X}|\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\})$$

GMM for density estimation (2)

Taking the logarithm helps to get rid of the products:

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

In order to find the best GMM approximation of the real density from which X was drawn, we need to find such parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ that maximize this likelihood.

GMM for clustering

In order to use GMM for clustering we introduce a K-dimensional binary random variable $\mathbf{z}$ for each data point $\mathbf{x}$.

Its k-th component is 1, if point $\mathbf{x}$ was generated from the k-th cluster and zero otherwise (one-hot encoding).

Then $p(z_k = 1) = \pi_k$ and $\sum_{k=1}^{K} \pi_k = 1$.

GMM for clustering (2)

Suppose, we already know the parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ of our GMM.

Then in order to assign clusters to our data points, for each data point $\mathbf{x}_i$ we are interested in the conditional probability of $\mathbf{x}_i$ being in the k-th cluster:

$$\begin{aligned} p(z_k = 1|\mathbf{x}_i) &= \frac{p(z_k = 1)p(\mathbf{x}_i|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}_i|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} =: \gamma(z_{ik}) \end{aligned}$$

π_k is the prior probability of $z_k = 1$ and $\gamma(z_{ik})$ is the posterior probability once $\mathbf{x}_i$ was observed.

GMM for clustering (3)

$\gamma(z_{ik})$ is viewed as the *responsibility* that component k takes for "explaining" the observation $\mathbf{x}_i$.

In order to determine the final assignment, each data point is assigned to the cluster with the largest probability: $arg\, max_k \gamma(z_{ik})$ over the responsibilities.

The problem is, we do not know the parameters of the underlying GMM.

$\rightarrow$ Expectation-Maximization algorithm

Expectation Maximization

EM: Problem formulation

Assumption:

- Gaussian mixture model as underlying distribution
- Number K of mixture components (clusters)

Find:

- Set of K mixture components defined by means $\{\boldsymbol{\mu}_k\}$ and covariances $\{\boldsymbol{\Sigma}_k\}$
- Mixture weights $\{\pi_k\}$
- Assignments of data points from X to clusters

in order to maximize the following objective function:

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

EM algorithm

1. Initialize $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ and evaluate the initial value of the log likelihood.
2. E step: Evaluate the responsibilities $\{\gamma(z_{ik})\}$, keeping the $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ fixed.
3. M step: Maximize L with respect to the $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$, keeping the $\{\gamma(z_{ik})\}$ fixed.
4. Repeat E step and M step until convergence.

Maximizing the likelihood

Calculating partial derivatives of L with respect to the means $\boldsymbol{\mu}_k$, setting them to zero and rearranging we obtain:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \mathbf{x}_i$$

where $N_k = \sum_{i=1}^{N} \gamma(z_{ik})$ is the number of points assigned to cluster k.

With similar calculations we get:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top$$

and

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

Density estimation example: Initialization

Given 1-D dataset $X = \{-3, -2.5, -1, 0, 2, 4, 5\}$ and $K = 3$.

Initialize the mixture components as

$$p_1(x) = \mathcal{N}(x| -4, 1)$$
$$p_2(x) = \mathcal{N}(x|0, 0.2)$$
$$p_3(x) = \mathcal{N}(x|8, 3)$$

and their weights as $\pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$.

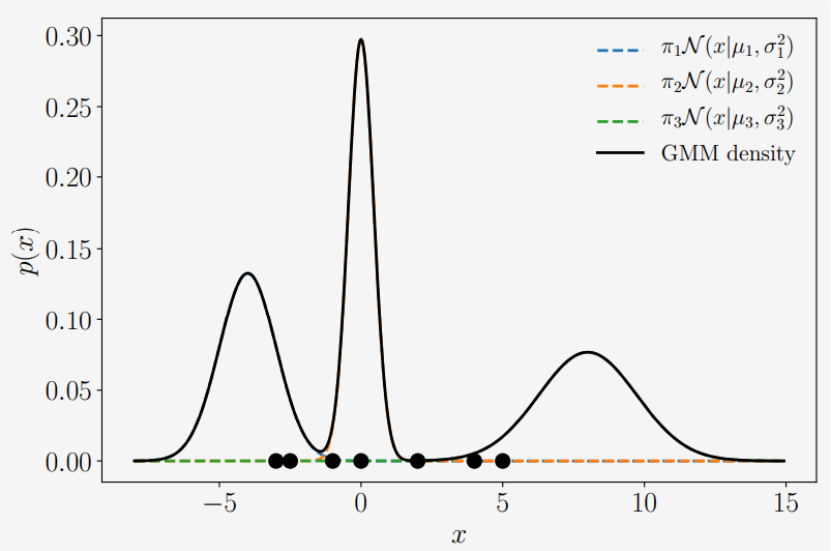
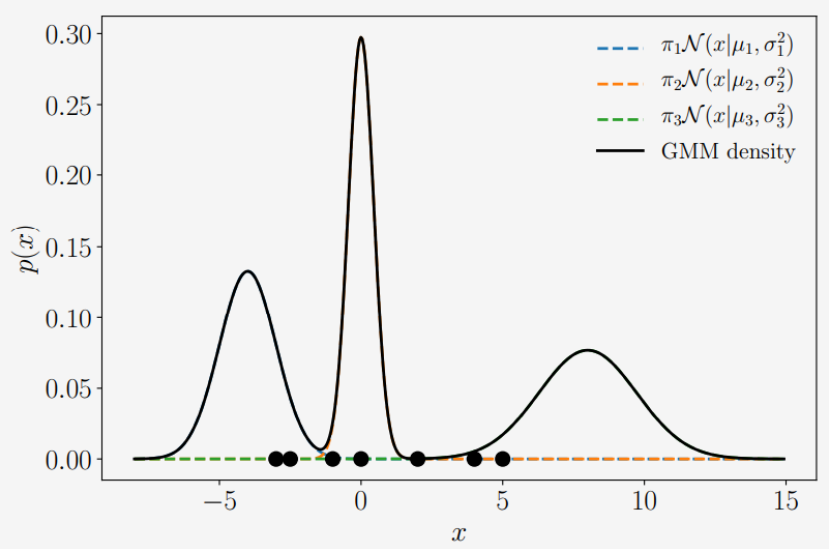


Image was taken from [DFO20]

Density estimation example: E step



Responsibilities are $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0.057 & 0.943 & 0 \\ 0.001 & 0.999 & 0 \\ 0 & 0.066 & 0.934 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times K}$

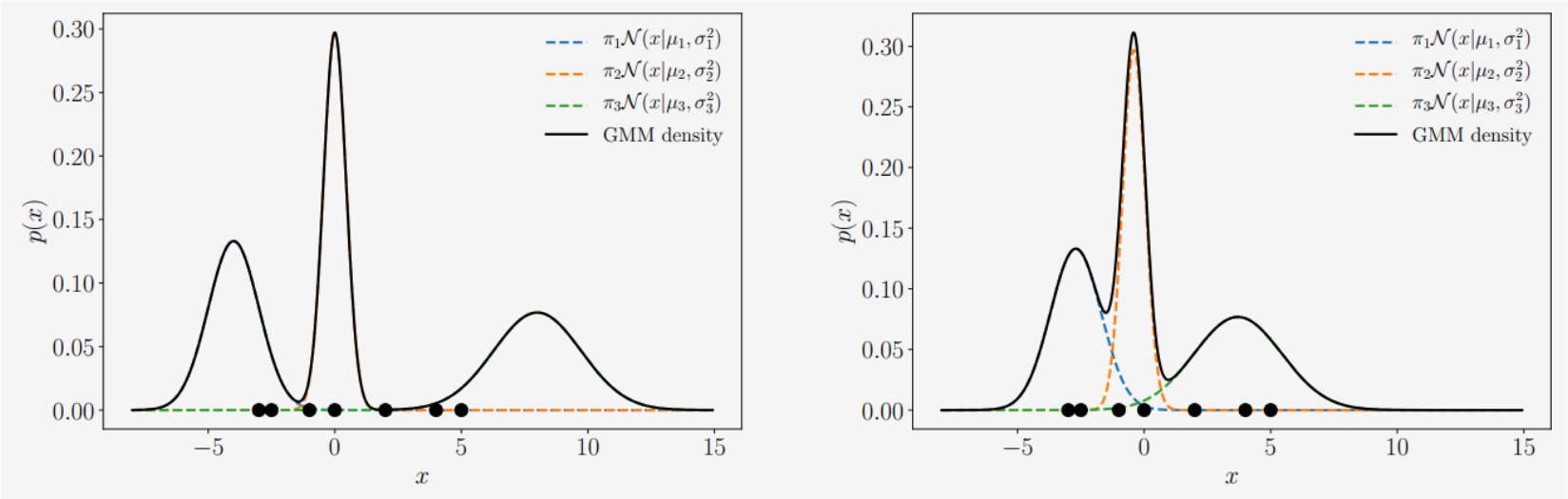
Image was taken from [DFO20]

M step: Updating the means

$$\mu_1 : -4 \to -2.7$$

$$\mu_2 : 0 \to -0.4$$

$$\mu_3 : 8 \to 3.7$$



Density before updating the means (left) and after (right).

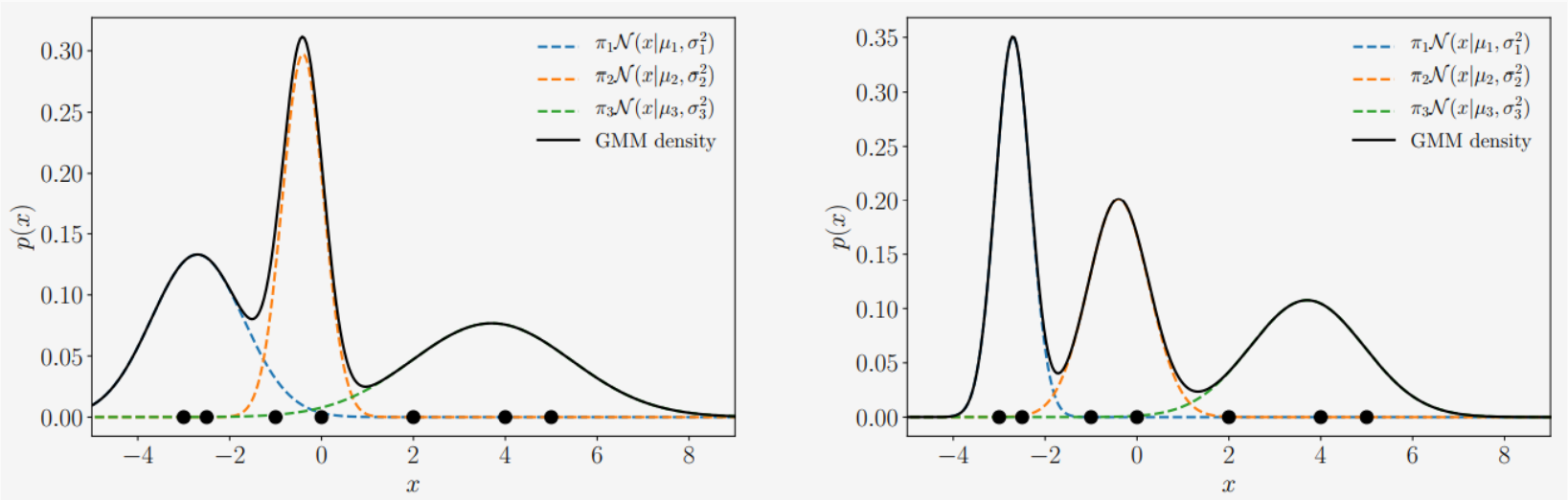
Image was taken from [DFO20]

M step: Updating the variances

$$\sigma_1^2 : 1 \to 0.14$$

$$\sigma_2^2 : 0.2 \to 0.44$$

$$\sigma_3^2 : 3 \to 1.53$$



Density before updating the variances (left) and after (right).

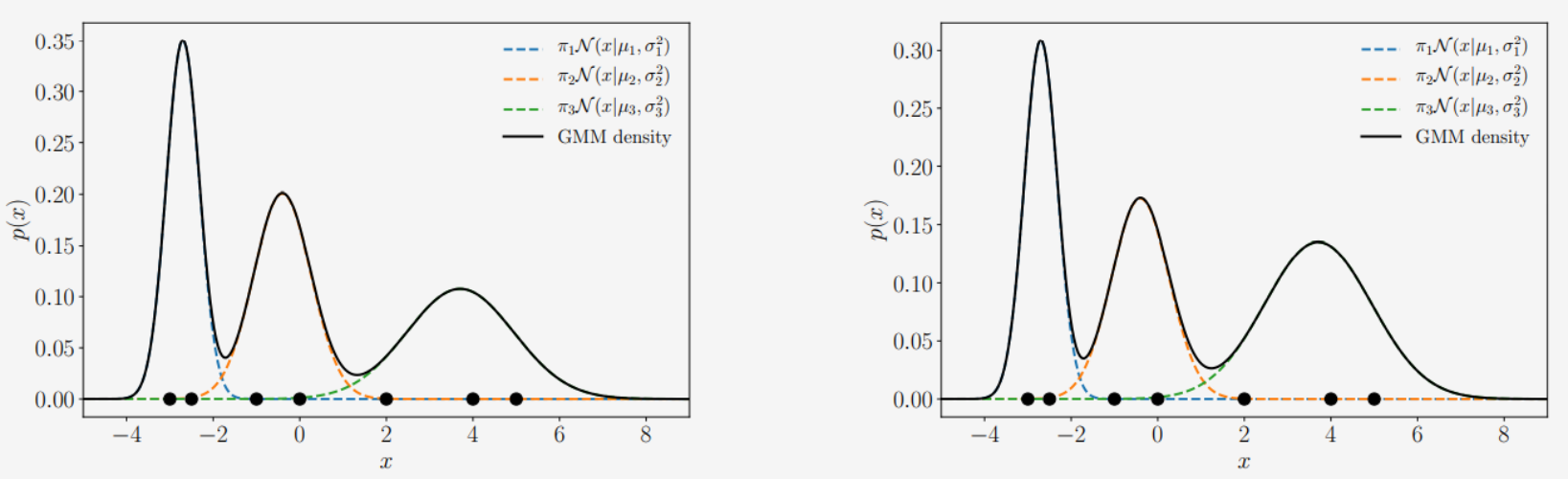
Image was taken from [DFO20]

M step: Updating the weights

$$\pi_1 : \frac{1}{3} \rightarrow 0.29$$

$$\pi_2 : \frac{1}{3} \rightarrow 0.29$$

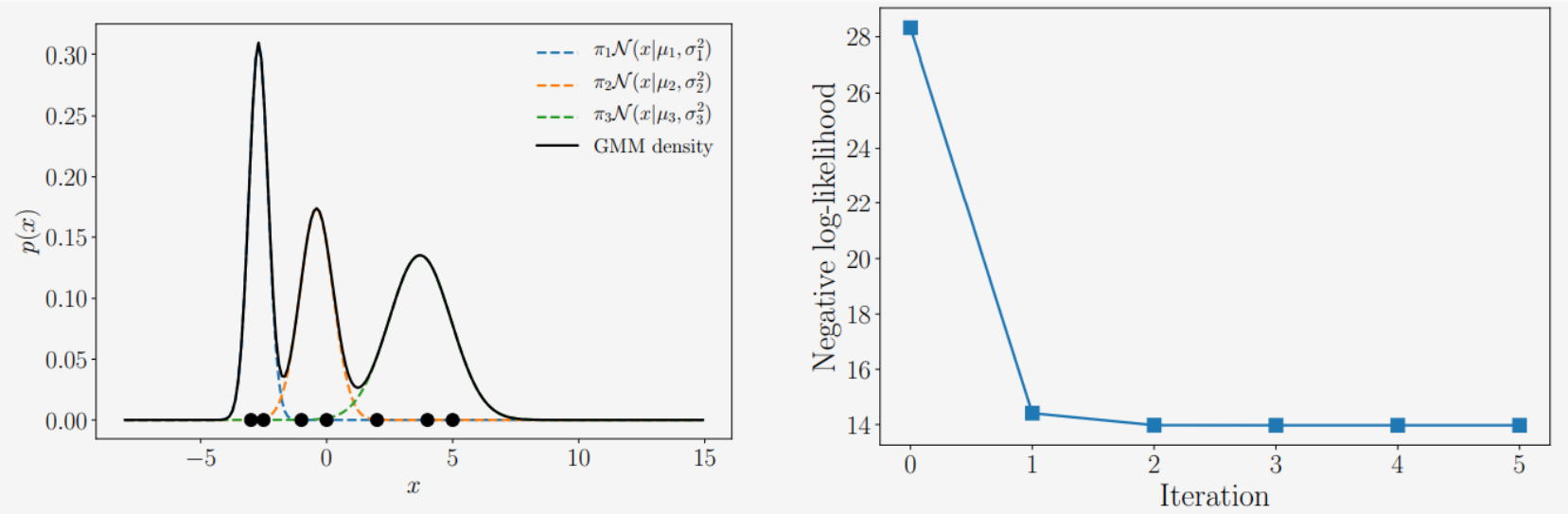
$$\pi_3 : \frac{1}{3} \rightarrow 0.42$$



Density before updating the weights (left) and after (right).

Image was taken from [DFO20]

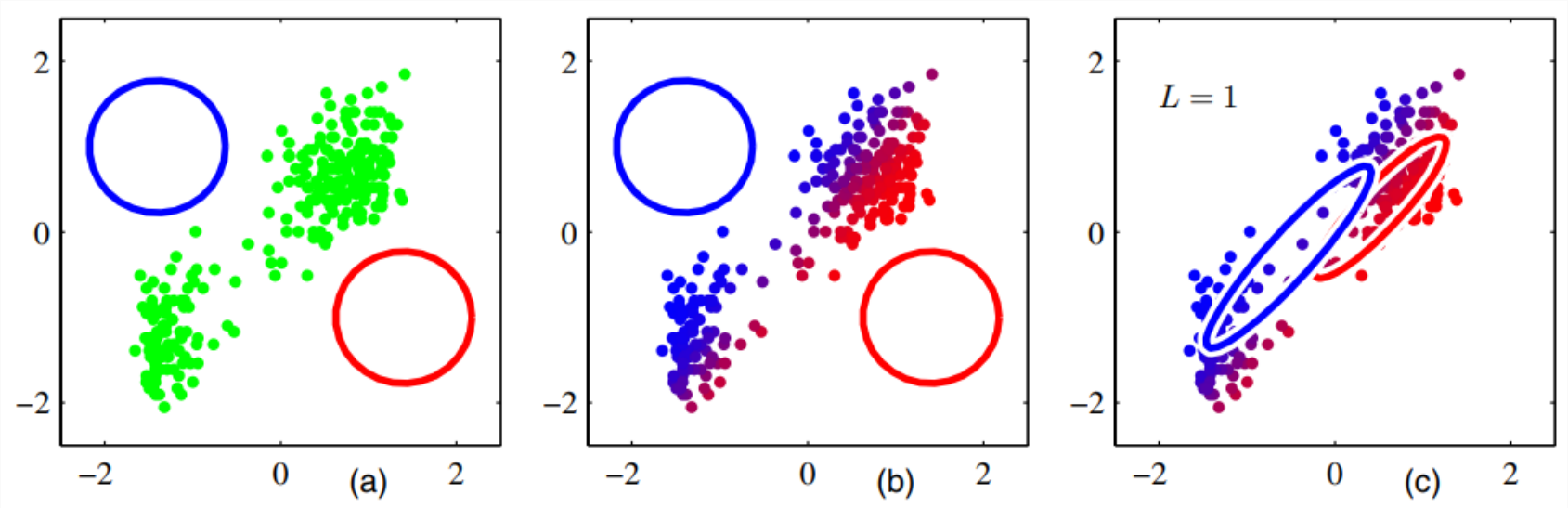
Density estimation example: final GMM fit



Left: After five iteration the EM algorithm has converged. Right: Negative log-likelihood

Image was taken from [DFO20]

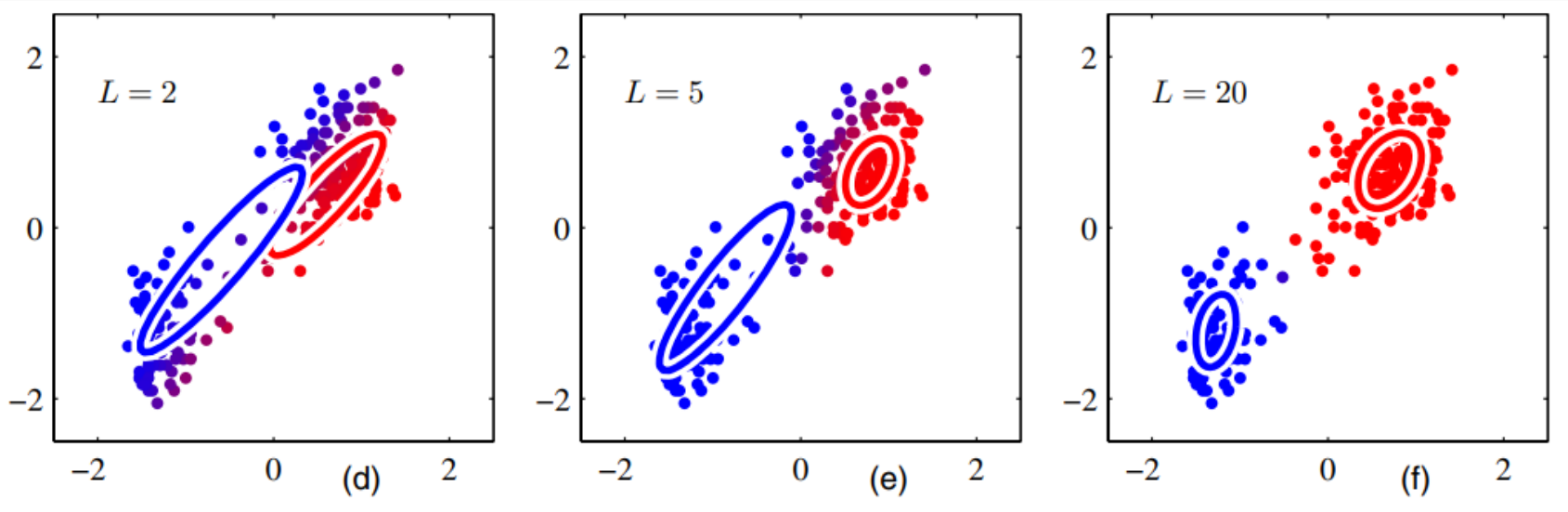
Clustering example



(a) Initialization, (b) E step, (c) M step

Image was taken from [Bis06]

Clustering example (2)



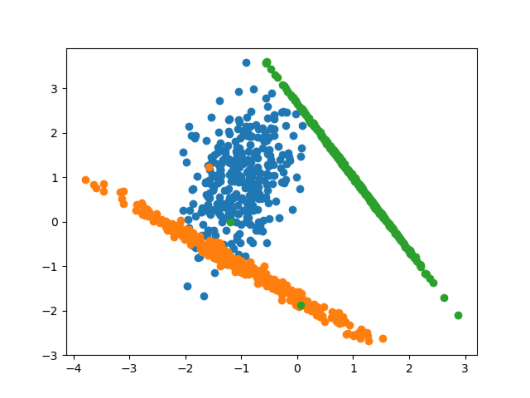
Results after 2, 5 and 20 iterations. After 20 iterations the algorithm is close to convergence.

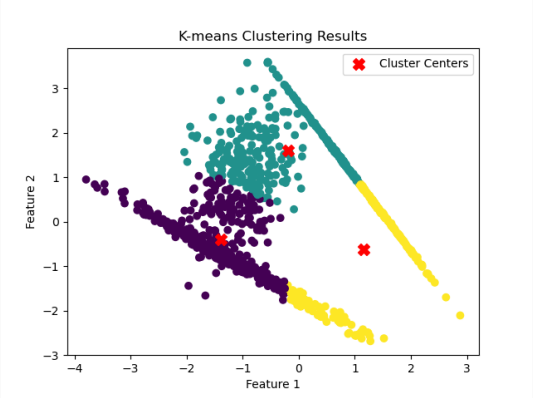
Image was taken from [Bis06]

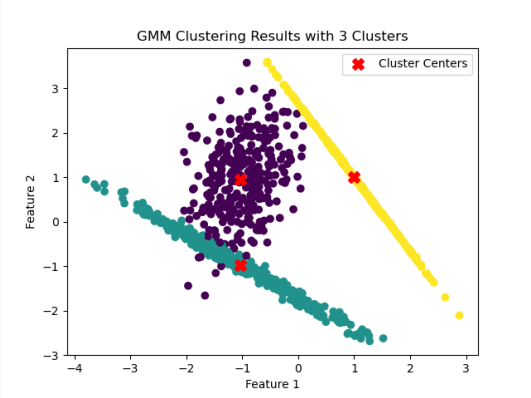
GMM vs. K-means

- In order to assign points to clusters, GMM does not use a distance measure, but applies a probability distribution around the cluster centers
  - $\rightarrow$ GMM handles outliers easier than K-means
  - $\rightarrow$ GMM can analyze more complex and mixed data
- EM algorithm requires many more iterations to reach convergence than K-means
- Each iteration of EM requires significantly more computations

GMM vs. K-means (2)







Initialization

K-means is "cheap", EM is "expensive"

$\rightarrow$ Run K-means to find suitable GMM initialization for EM

- Initialize the means with the means found by K-means
- Initialize the covariance matrices with the sample covariances of the clusters found by K-means
- Initialize the weights with the fractions of data points assigned to respective clusters