Machine Learning

Lecture 11: Clustering

Prof. Dr. Stephan Günnemann

Data Mining and Analytics
Technical University of Munich

21.01.2019

Reading material

Reading material

- Bishop: chapters 9.1, 9.2, 9.3.0, 9.3.1

Unsupervised learning

Main idea

- Given some unlabeled data $\{\boldsymbol{x}_i\}$ we want to discover latent structure in it.

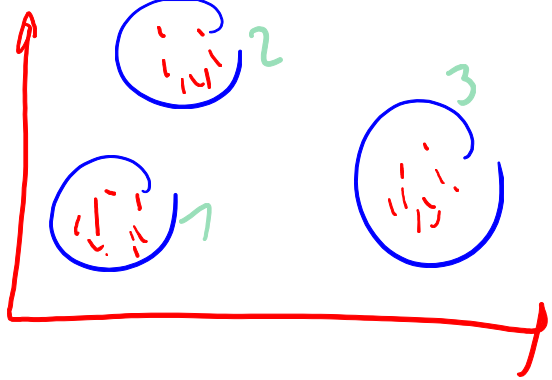
Last week: Dimensionality reduction

- Given high-dimensional data $\boldsymbol{x}_i \in \mathbb{R}^D$, find a continuous representation $\boldsymbol{z}_i \in \mathbb{R}^K$ with $K \ll D$, such that the structure is preserved.

This week: Clustering

- Group objects $\boldsymbol{x}_i$ into K groups (clusters) based on their similarity.

Problem definition



Given

- Collection of data points $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{X}$.

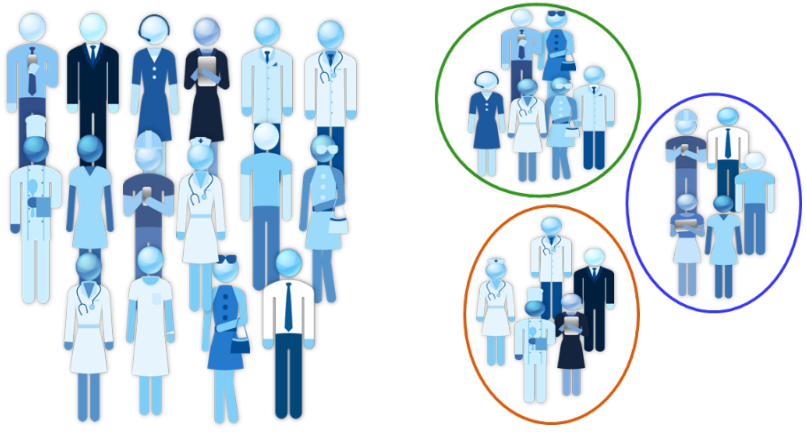
Goal

- For each object $\boldsymbol{x}_i$ find the corresponding assignment $\boldsymbol{z}_i \in \mathbb{R}^K$ to one of the K groups (clusters), such that:
  - objects within the same group are similar to each other;
  - objects between different groups are dissimilar from each other;

$\boldsymbol{z}_i$ is a one-hot indicator vector, with $z_{ik} = 1 \iff$ point i belongs to cluster k.

Clustering: Examples

- Image segmentation
- User profiling
- Data compression
- Visualization
- ...



Sources: `https://blogs.sas.com/content/subconsciousmusings/2016/05/26/data-mining-clustering/` and C. Bishop - "Pattern Recognition and Machine Learning"

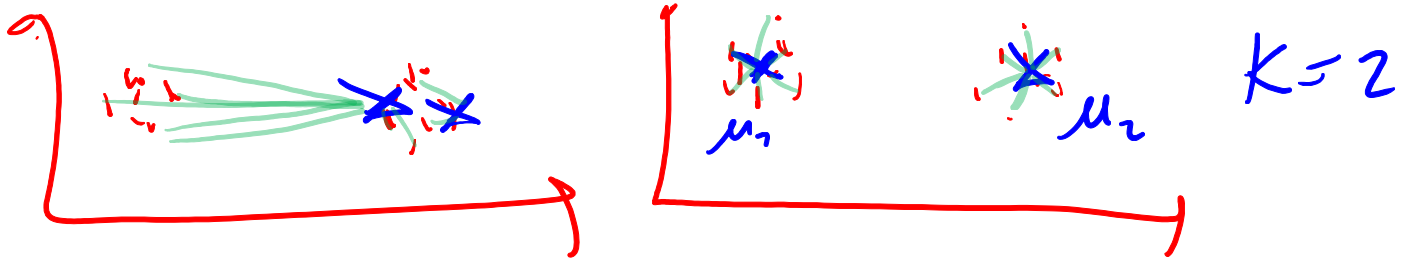
Section 1

K-means Algorithm

Distance-based clustering

- If we want to group similar objects together, we need some notion of similarity to begin with.
- Typically defined as similarity function $s(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
- Alternatively: minimize dissimilarity, usually provided by a distance function $d(\cdot, \cdot)$.
- Typical distance functions include
  - Manhattan distance: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_d |x_{id} - x_{jd}|$
  - Euclidean distance: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_d (x_{id} - x_{jd})^2}$
  - Mahalanobis distance: $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$

K-means



Model

- Consider the standard $\mathbb{R}^D$ space with Euclidean distance as metric.
- Each of K clusters is defined by its centroid $\boldsymbol{\mu}_k \in \mathbb{R}^D$.
- Cluster indicator $\boldsymbol{z}_i \in \{0,1\}^K$ with $||\boldsymbol{z}_i|| = 1$ (i.e. one-hot vector) shows which cluster point i belongs to.
- The K-means objective (also called distortion measure) is defined as

$$J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||_2^2$$

// CONSTRAINT RELAXATION
// IDEA:
// $z_i \in [0,1]^K$
// $\sum z_{ik} = 1$

Goal

- We need to find the "best" centroids $\boldsymbol{\mu}$ and cluster assignments $\boldsymbol{Z}$

$$\boldsymbol{Z}^*, \boldsymbol{\mu}^* = \arg\min_{\boldsymbol{Z}, \boldsymbol{\mu}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$$

Minimizing the K-means objective

- The joint optimization problem is difficult to solve (discrete optim.; even if $\boldsymbol{Z}$ is assumed to be continuous it is non-convex)

$$\min_{\boldsymbol{Z},\boldsymbol{\mu}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu}) = \min_{\boldsymbol{Z},\boldsymbol{\mu}} \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||_2^2$$

- However, the two subproblems

$$\min_{\boldsymbol{Z}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu}) \qquad \text{and} \qquad \min_{\boldsymbol{\mu}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$$

  have simple closed form solutions!

- **Idea:** alternating optimization - update $\boldsymbol{Z}$ and $\boldsymbol{\mu}$ in turns!

- This method for solving K-means is called Lloyd's algorithm.

Lloyd's algorithm for K-means

1. Initialize the centroids $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\}$.
2. Update cluster indicators (solve $\min_{\boldsymbol{Z}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$)

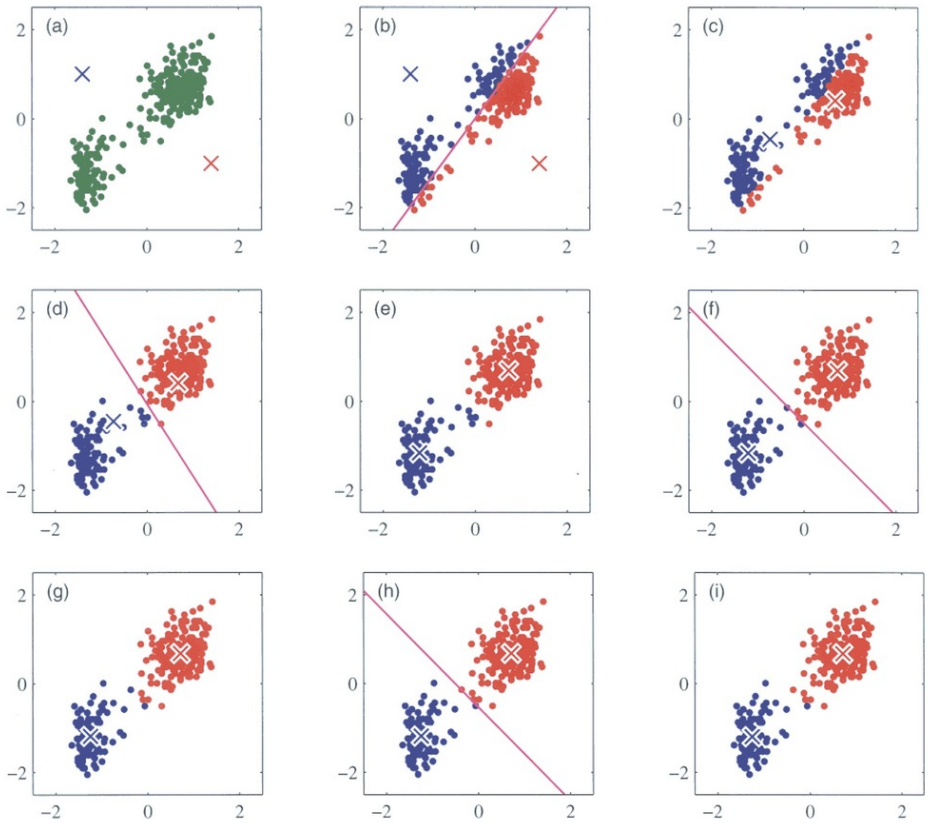
$$z_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2, \\ 0 & \text{else.} \end{cases}$$

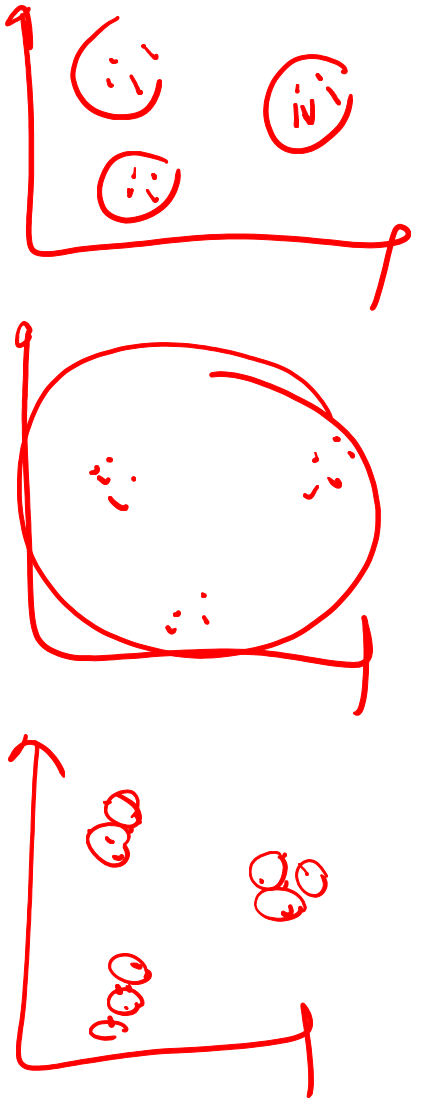
3. Update centroids (solve $\min_{\boldsymbol{\mu}} J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$)

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} z_{ik} \boldsymbol{x}_i, \qquad \text{where } N_k = \sum_{i=1}^{N} z_{ik}$$

4. If objective $J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$ has not converged, return to step 2.

K-means in action



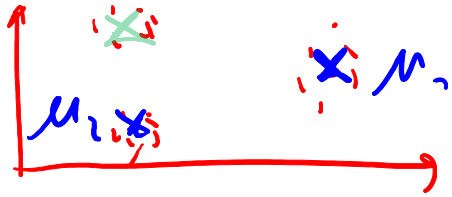


An interactive demo is available at

- `http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html`

Source: Bishop, Figure. 9.1

Initialization of the centroids



We saw in the interactive demo that the initial locations of the centroids $\boldsymbol{\mu}_k$ greatly affect the performance.

So, how do we initialize the centroids in a good way?

K-means++ algorithm

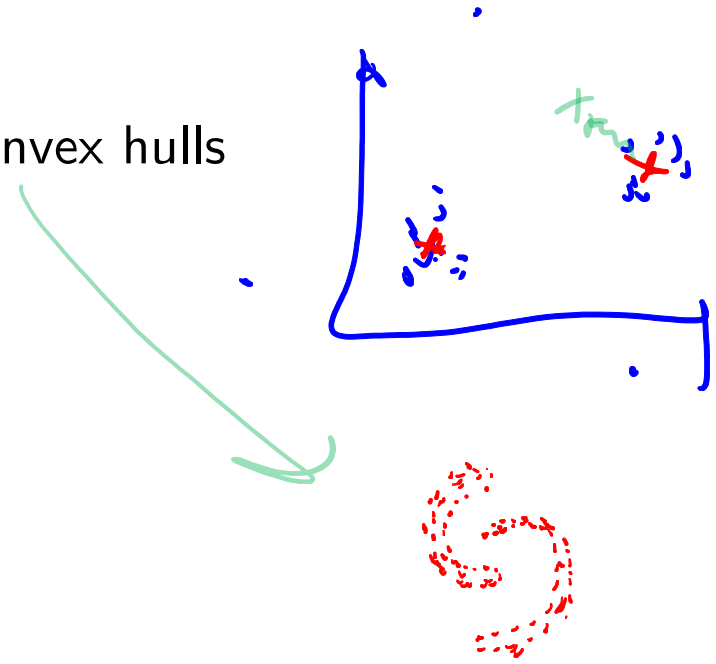
1. Choose the first centroid $\boldsymbol{\mu}_1$ uniformly at random among the data points.
2. For each point $\boldsymbol{x}_i$ compute the distance $D_i^2 = ||\boldsymbol{x}_i - \boldsymbol{\mu}_1||_2^2$.
3. Sample the next centroid $\boldsymbol{\mu}_k$ from ~~remaining~~ points with probability proportional to D_i^2.
4. Recompute the distances $D_i^2 = \min\{||\boldsymbol{x}_i - \boldsymbol{\mu}_1||_2^2, ..., ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||_2^2\}$.
5. Continue steps 3 and 4 until K initial centroids have been chosen.

Limitations of K-means

It is important to distinguish between the model (distortion $J(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\mu})$ in our case) and the algorithm used to fit it (here Lloyd's algorithm).

Modeling issues

- Underlying assumptions are not explicit
- Can't detect clusters with overlapping convex hulls
- Sensitivity to outliers
- No uncertainty measure



Algorithmic issues

- Extreme sensitivity to initialization

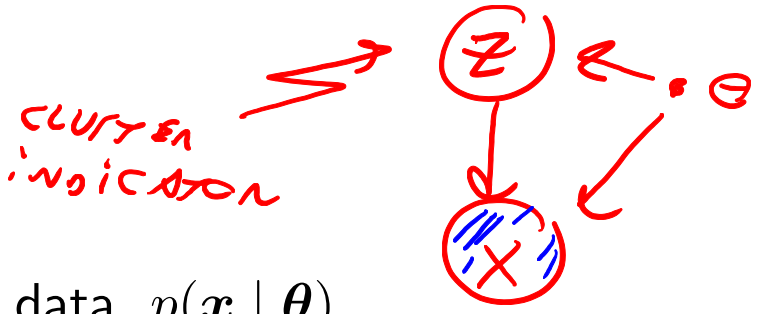
How can we address (some of) these issues?

Switch to the more principled probabilistic framework!

Section 2

Gaussian Mixture Model

Probabilistic unsupervised learning



- Goal: Design a probabilistic model for the data, $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, parametrized by $\boldsymbol{\theta}$, which we have to estimate.
- It's difficult to come up with a useful model $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ without making any assumptions about the data.
- When doing clustering, we implicitly assume that each point $\boldsymbol{x}$ belongs to some cluster (indicated by $\boldsymbol{z}$).
  $\Rightarrow$ Let's model $p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$ instead! $p(x \mid z, \theta) \cdot p(z \mid \theta)$
- Let's recall what tools we developed during our discussion of supervised learning, and see if any of them can help us here.
- For this, assume for now, that we know the true cluster indicators $\boldsymbol{z}$, and are only interested in estimating $\boldsymbol{\theta}$.

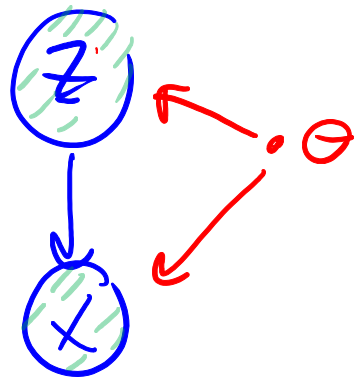
Parameter estimation

How can we estimate the parameters $\boldsymbol{\theta}$ of a given model $p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})$?

Supervised learning

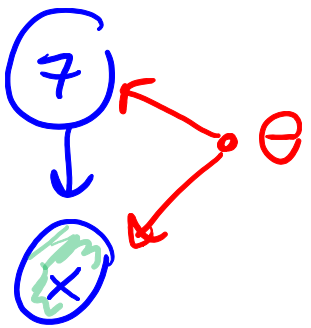
- Both $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ and $\boldsymbol{Z} \in \{0, 1\}^{N \times K}$ are known.
- Simple solution - maximum likelihood estimation

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{X}, \boldsymbol{Z} | \boldsymbol{\theta})$$



Unsupervised learning

- Only the data $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ is given.
- Is there any way to estimate $\boldsymbol{\theta}$ in this case?



Latent variables

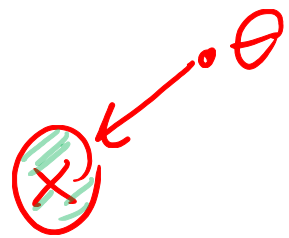
How can we estimate $\boldsymbol{\theta}$ when $\boldsymbol{Z}$ is unknown?

We can compute the joint probability $p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})$, so we can obtain $p(\boldsymbol{X} \mid \boldsymbol{\theta})$ by marginalization

$$p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{\theta}) \cdot p(\boldsymbol{Z} \mid \boldsymbol{\theta})$$

and simply optimize w.r.t $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{X} \mid \boldsymbol{\theta})$$



Since true $\boldsymbol{Z}$ are never observed, and are only our abstraction used to simplify the model, they are called latent variables. [1]

[1] Sometimes also called hidden or explanatory variables.

Gaussian mixture model (GMM)

We decompose the joint probability $p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$ using the product rule [2]

$$p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) = p(\boldsymbol{z} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta})$$

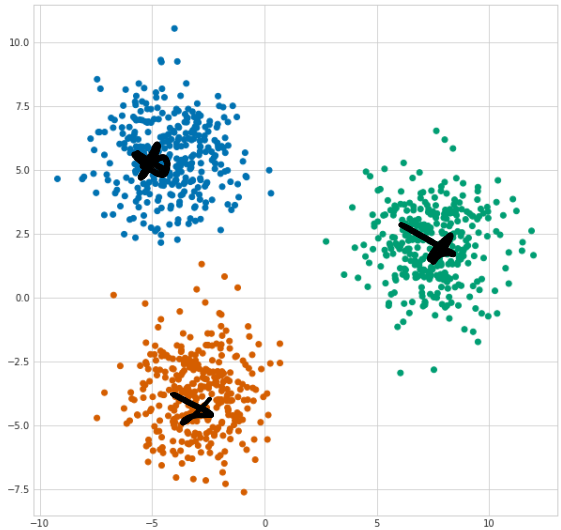
In Gaussian mixture model we assume:

- Cluster prior is categorical

$$p(\boldsymbol{z} \mid \boldsymbol{\theta}) = \mathrm{Cat}(\boldsymbol{\pi})$$

- Each cluster has its own multivariate normal distribution

$$p(\boldsymbol{x} \mid z_k = 1, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



[2] We assume that samples are drawn i.i.d., so consider only a single $(\boldsymbol{x}, \boldsymbol{z})$.

Generative process of GMM

We can draw data from a GMM as following.

Setup

We have K Gaussian components (clusters), each with its own $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$.

Prior probability of each cluster k is π_k.

Generative process

1. Draw a one-hot cluster indicator $\boldsymbol{z} \sim \mathrm{Cat}(\boldsymbol{\pi})$.

   $z_k = 1 \iff$ current point belongs to cluster k.

2. Draw the sample $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ if $z_k = 1$.

We only observe the sample $\boldsymbol{x}$, and don't know which k it came from.

GMM likelihood

Using marginalization over $\boldsymbol{z}$, the probability of a single sample $\boldsymbol{x}$ under a Gaussian mixture model can be computed as

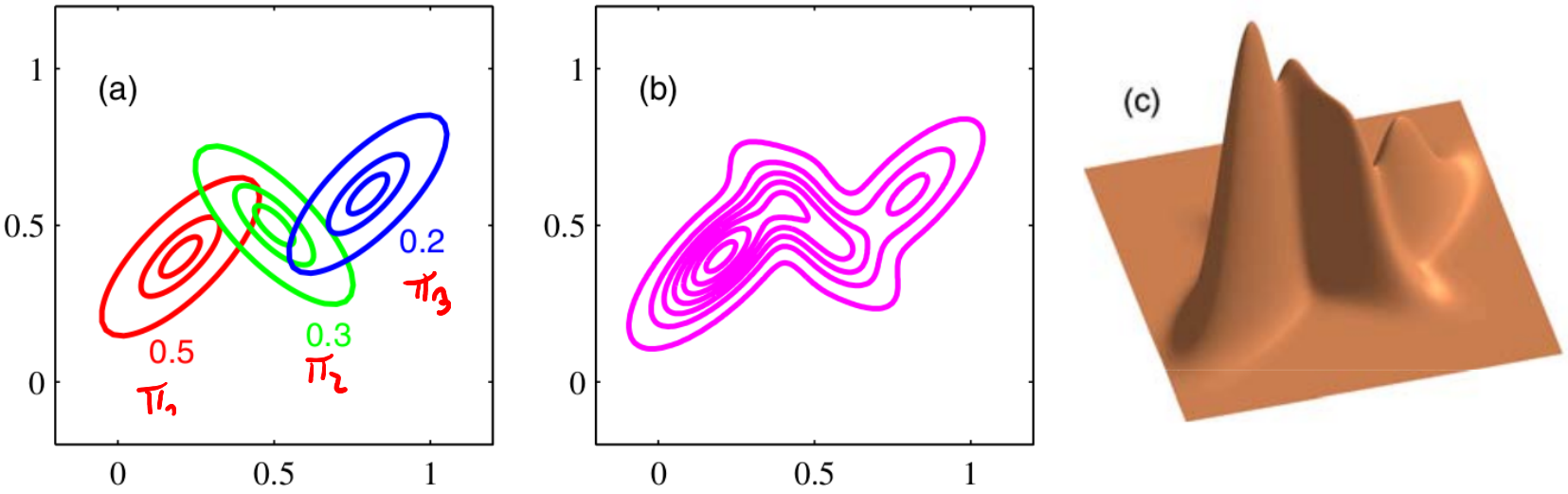
$$p(\boldsymbol{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} p(z_k = 1 \mid \boldsymbol{\pi}) \cdot p(\boldsymbol{x} \mid z_k = 1, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Hence, for the entire dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ the log-likelihood is

$$\log p(\boldsymbol{X} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

GMM density

$k = 3$



(a) Isocontours of each component $p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with respective mixing coefficients π_k.

(b) Isocontours of the GMM distribution

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

(c) Surface plot of $p(\boldsymbol{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Source: Bishop, Figure. 2.23

Learning and inference in GMM

Remember, that we are interested in two things

Learning

- We want to determine the "optimal" values of the parameters (e.g., find values that maximize the log-likelihood)

$$\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \arg\max_{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}} \log p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

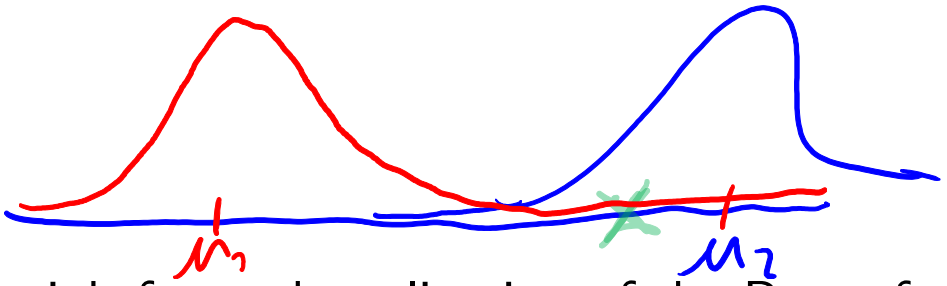
Inference

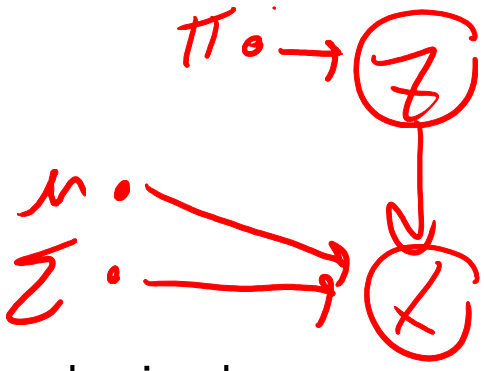
- Given the parameters $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, we want to assign the points to clusters, i.e. compute the posterior distribution of cluster indicators

$$p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

(This is the actual goal of clustering.)

Inference in GMM





By straightforward application of the Bayes formula, we obtain the posterior over the entries of $\boldsymbol{Z}$

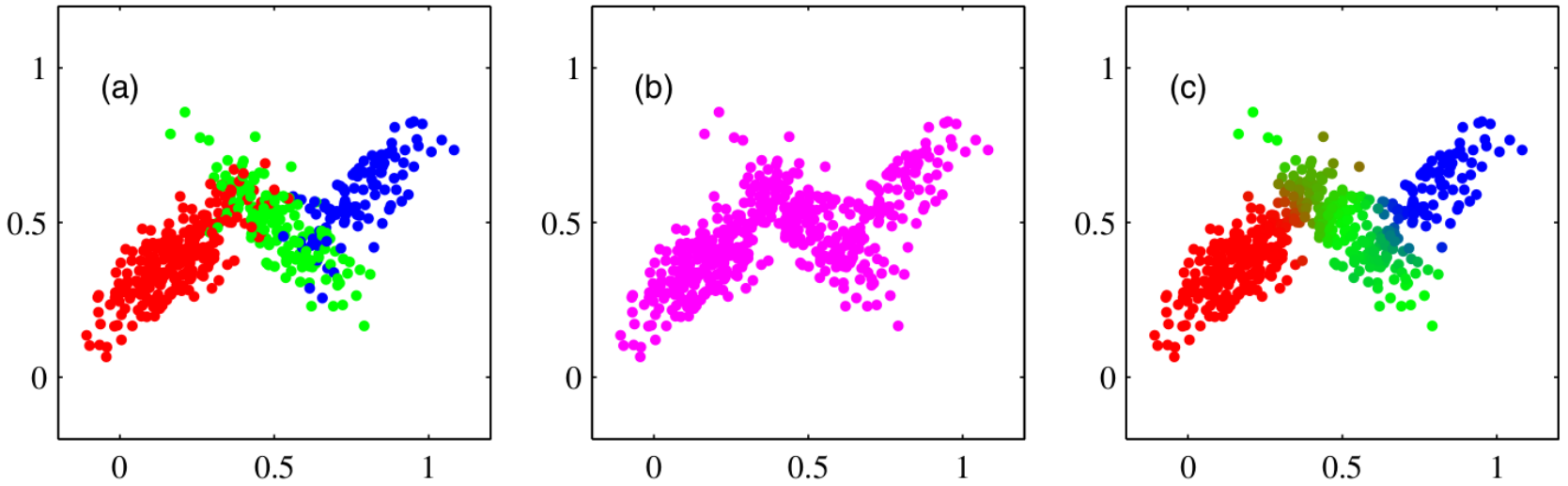
$$p(z_{ik} = 1 \mid \boldsymbol{x}_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

We will call this quantity the responsibility of component k for the observation i.

We also introduce shorthand notation for it

$$\gamma(z_{ik}) = p(z_{ik} = 1 \mid \boldsymbol{x}_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Inference: Example



(a) Data colored according to the true underlying $\boldsymbol{Z}$.

(b) Observed data $\boldsymbol{X}$.

(c) Data colored according to the inferred $\gamma(\boldsymbol{Z})$ (with $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ known).

$= p(z|x, \pi, \mu, \Sigma)$

Source: Bishop, Figure. 9.5

Learning in GMM

Maximization of the log-likelihood w.r.t. the parameters appears to be more challenging. Recall, that the log-likelihood is

$$\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Due to the summation over k, the logarithm doesn't act directly on the Gaussian density.

This happens to be a serious problem, as this means that the derivatives w.r.t. the model parameters $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ have no analytical roots. [3]

Is there anything that we can do in this case?

[3] I.e. we cannot obtain closed form expressions for the optimal values of $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$.

Learning in GMM

Linear

Recall from Linear Classification lecture (Gaussian discriminant analysis), that parameter estimation is rather straightforward if $\boldsymbol{Z}$ is known.

That is, we know how to solve the "easy" problem

$$\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \underset{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}}{\arg\max} \ \log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad (*)$$

Learning in GMM

Recall from Linear Classification lecture (Gaussian discriminant analysis), that parameter estimation is rather straightforward if $\boldsymbol{Z}$ is known.

That is, we know how to solve the "easy" problem

$$\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \underset{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}}{\arg\max} \log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad (*)$$

Too bad we don't know anything about $\boldsymbol{Z}$... Or do we?

Learning in GMM

Recall from Linear Classification lecture (Gaussian discriminant analysis), that parameter estimation is rather straightforward if $\boldsymbol{Z}$ is known.

That is, we know how to solve the "easy" problem

$$\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \arg\max_{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}} \log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad (*)$$

Too bad we don't know anything about $\boldsymbol{Z}$... Or do we?

Idea: use our current beliefs $p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})$ and solve the easier problem $(*)$.

Important: As we are dealing with a distribution over $\boldsymbol{Z}$, we have to take the expectation to account for the uncertainty

$$\boldsymbol{\pi}^*, \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \arg\max_{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}} \mathbb{E}_{\boldsymbol{Z} \sim \gamma(\boldsymbol{Z})} [\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})]$$

$$= \arg\max_{\pi, \mu, \Sigma} \sum_{z} p(z \mid X, \pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}) \cdot \log p(X, z \mid \pi, \mu, \Sigma)$$

Expectation-Maximization algorithm

This two-step procedure that we just derived is called Expectation-Maximization algorithm (EM).

In its most general form, the two steps are

- **E step** - evaluate the posterior $\gamma(\boldsymbol{Z}) = p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$.
- **M step** - maximize the current estimate of $p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{Z} \sim \gamma(\boldsymbol{Z})} [\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})]$$

Note: we do **not** get a closed form solution neither for $\gamma(\boldsymbol{Z})$, nor for the parameters, as $\gamma(\boldsymbol{Z})$ and $\boldsymbol{\theta}$ recursively depend on each other.

Therefore, we have to iterate between these two steps until the objective $\mathbb{E}_{\boldsymbol{Z} \sim \gamma(\boldsymbol{Z})} [\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})]$ converges.

EM algorithm for GMM

$N_k = \sum_{i=1}^{N} \gamma(z_{i,k})$

1. Initialize model parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\}$.
2. **E step**. Evaluate the responsibilities for current $\boldsymbol{\theta}$

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

3. **M step**. Re-estimate the parameters using current $\gamma(z_{ik})$'s

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

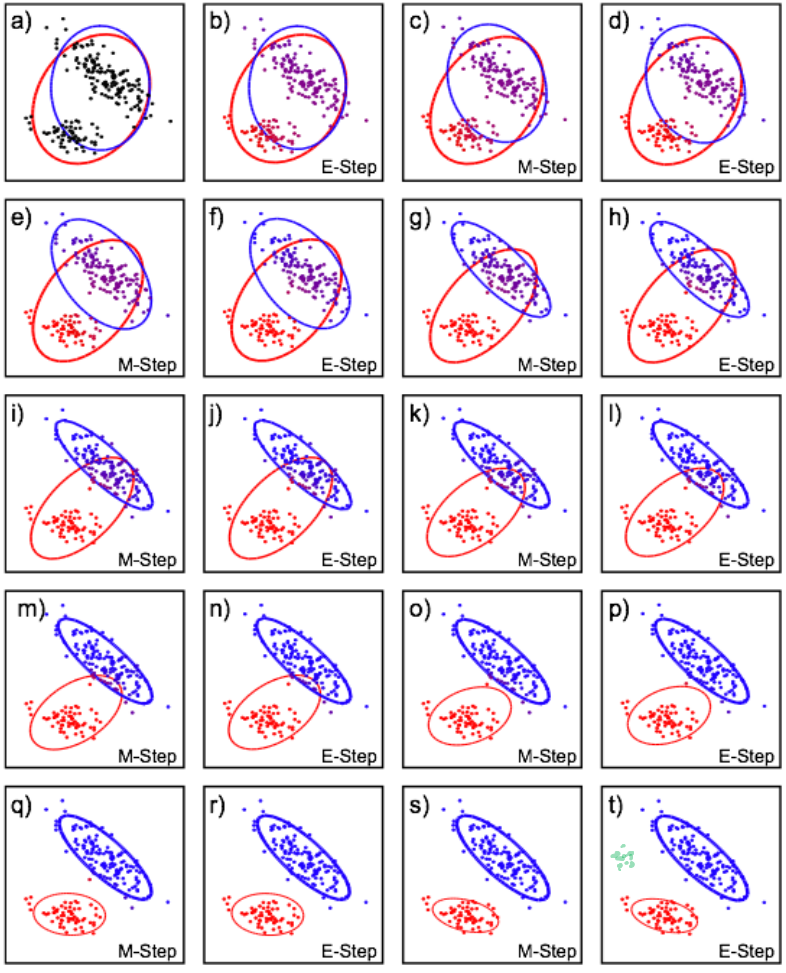
WEIGHTED MEAN

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

$$\pi_k^{new} = \frac{N_k}{N}.$$

4. If $\mathbb{E}_{\boldsymbol{Z} \sim \gamma(\boldsymbol{Z})} [\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})]$ has not converged, return to step 2.

EM algorithm for GMM: Demo



Gif: `https://upload.wikimedia.org/wikipedia/commons/6/69/EM_Clustering_of_Old_Faithful_data.gif`

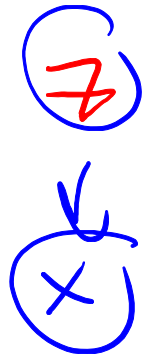
Source: Computer Vision: Models, Learning, and Inference - S.J.Prince

EM algorithm: Summary

When should we use EM?

- Intractable to optimize the log-likelihood

$$\log p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \log \left(\sum_{\boldsymbol{Z}} p(\boldsymbol{Z} \mid \boldsymbol{\theta}) p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{\theta}) \right)$$



- Easy to optimize the joint probability

$$\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) = \log p(\boldsymbol{Z} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{X} \mid \boldsymbol{Z}, \boldsymbol{\theta})$$

The main idea of EM

- Use our current beliefs $p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$ to "pretend" that we know $\boldsymbol{Z}$.
- E step - update the responsibilities $\gamma(\boldsymbol{Z}) = p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$.
- M step - update parameters

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{Z} \sim \gamma(\boldsymbol{Z})} \left[\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) \right]$$