

# Machine Learning (IN2064)

## Lecture 12: Clustering

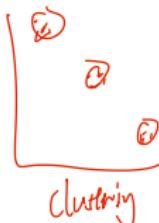
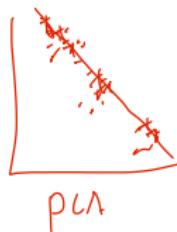
---

Prof. Dr. Stephan Günnemann

Data Analytics and Machine Learning  
Technical University of Munich

Winter term 2022/2023

# Unsupervised learning



## Main idea

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

## Last week: Dimensionality reduction

- Given high-dimensional data  $x_i \in \mathbb{R}^D$ , find a continuous representation  $z_i \in \mathbb{R}^K$  with  $K \ll D$ , such that the structure is preserved.  
*low dim representation.*

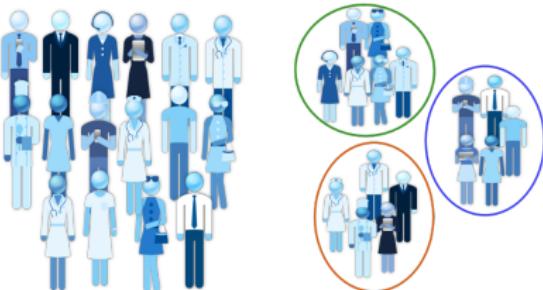
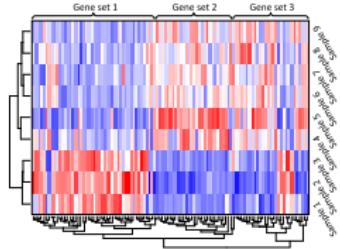
## This week: Clustering

- Group objects  $x_i$  into  $K$  groups (**clusters**) based on their similarity.

same dim

# Clustering: Examples

- Image segmentation
- User profiling
- Gene expression analysis
- Data compression
- Visualization
- ...



Sources: <https://blogs.sas.com/content/subconsciousmusings/2016/05/26/data-mining-clustering/>,  
[https://en.wikipedia.org/wiki/Transcriptomics\\_technologies#/media/File:Transcriptomes\\_heatmap\\_example.svg](https://en.wikipedia.org/wiki/Transcriptomics_technologies#/media/File:Transcriptomes_heatmap_example.svg),  
and C. Bishop - "Pattern Recognition and Machine Learning"

# Problem definition

## Given

- Collection of data points  $\mathbf{X} = \{x_1, \dots, x_N\} \subset \mathbb{X}$ .

## Goal

- For each object  $x_i$  find the corresponding assignment  $z_i \in \{1, \dots, K\}$  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;

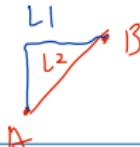
one point  $\rightarrow$  one cluster  
the point  $\rightarrow$  more cluster

## Section 1

### *K*-means Algorithm

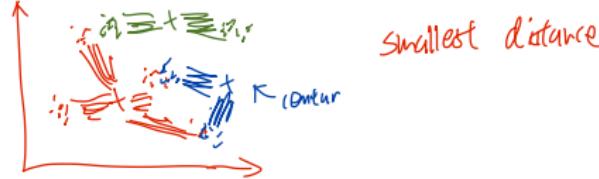
# Distance-based clustering

- If we want to group **similar** objects together, we need some notion of **similarity** to begin with.
- Typically defined as similarity function  $s(\cdot, \cdot) : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ . *Simple when one part*
- Alternatively: minimize dissimilarity, usually provided by a **distance function**  $d(\cdot, \cdot)$ .
- Typical distance functions include
  - Manhattan distance:  $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_d |\mathbf{x}_{id} - \mathbf{x}_{jd}| = \|\mathbf{x}_i - \mathbf{x}_j\|_1$
  - Euclidean distance:  $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_d (\mathbf{x}_{id} - \mathbf{x}_{jd})^2} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$
  - Mahalanobis distance:  $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$



# K-means

## Model



- Consider the standard  $\mathbb{R}^D$  space with Euclidean distance as metric.
- Each of  $K$  clusters is defined by its centroid  $\mu_k \in \mathbb{R}^D$ .
- Cluster indicator  $z_i \in \{0, 1\}^K$  with  $z_{ik} = 1 \iff x_i$  belongs to cluster  $k$  (i.e. one-hot encoding). clutter  
↓  
z\_i [ 0 0 0 0 1 0 0 ]
- The K-means objective (also called distortion measure) is defined as

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

Goal  $= \sum_i (z_{i1} \|x_i - \boldsymbol{\mu}_1\|_2^2 + z_{i2} \|x_i - \boldsymbol{\mu}_2\|_2^2 + z_{i3} \|x_i - \boldsymbol{\mu}_3\|_2^2)$

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

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

# Minimizing the $K$ -means objective

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

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

- However, the two subproblems

$$\min_Z J(X, Z, \mu)$$

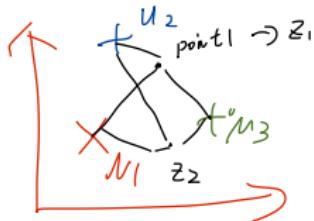
and

$$\min_{\mu} J(X, Z, \mu)$$

have simple closed form solutions!

- Idea:** alternating optimization - update  $Z$  and  $\mu$  in turns!
- This method for solving  $K$ -means is called **Lloyd's algorithm.**

# Lloyd's algorithm for $K$ -means



1. Initialize the centroids  $\mu = \{\mu_1, \dots, \mu_K\}$ .
2. Update cluster indicators (solve  $\min_{\mathbf{Z}} J(\mathbf{X}, \mathbf{Z}, \mu)$ )

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

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

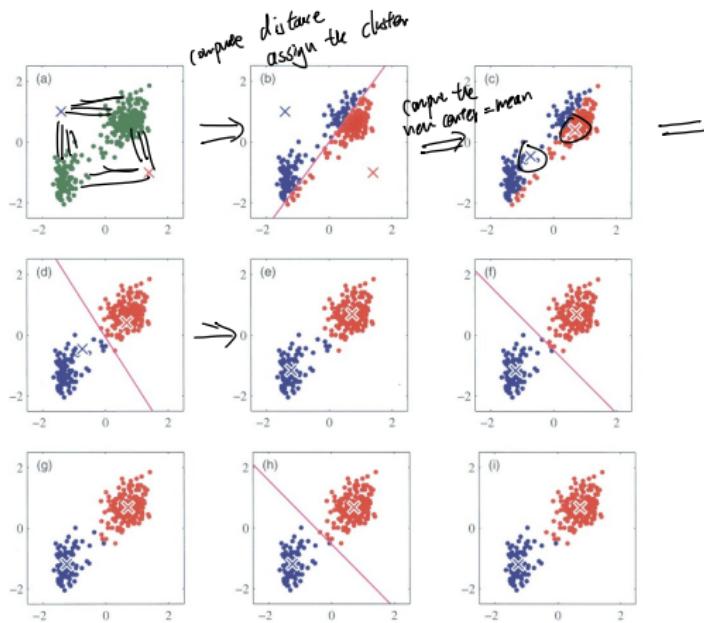
$$\mu_k = \frac{1}{N_k} \sum_{i=1}^N z_{ik} \mathbf{x}_i \quad \text{where } N_k = \sum_{i=1}^N z_{ik}.$$

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

update cluster indicator  
choose smallest distance



# K-means in action



An interactive demo is available at

- <https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/>

---

Source: Bishop, Figure. 9.1

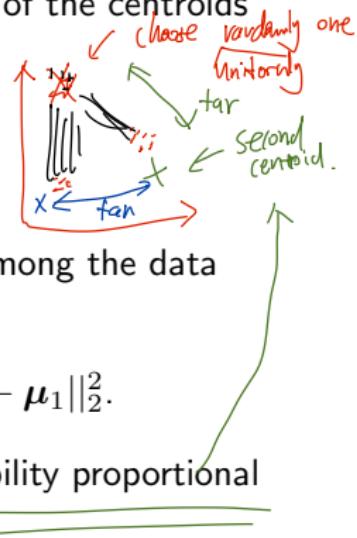
# Initialization of the centroids

We saw in the interactive demo that the initial locations of the centroids  $\mu_k$  greatly affect the performance.

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

## K-means++ algorithm

1. Choose the first centroid  $\mu_1$  uniformly at random among the data points.
2. For each point  $x_i$  compute the distance  $D_i^2 = \|\mathbf{x}_i - \mu_1\|_2^2$ .
3. Sample the next centroid  $\mu_k$  from  $\{x_i\}$  with probability proportional to  $D_i^2$ .
4. Recompute the distances  $D_i^2 = \min\{\|\mathbf{x}_i - \mu_1\|_2^2, \dots, \|\mathbf{x}_i - \mu_k\|_2^2\}$ .
5. Continue steps 3 and 4 until  $K$  initial centroids have been chosen.



# Limitations of K-means

*sensitive.*  $\Rightarrow$  distance function.

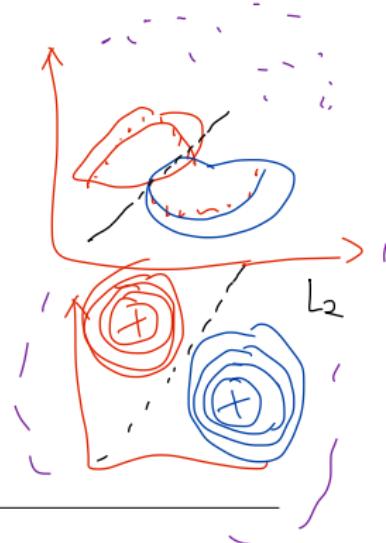
It is important to distinguish between the **model** (distortion  $J(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu})$  in our case) and the **algorithm** used to fit it (here Lloyd's algorithm). Outliers.

## 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,  $p(x | \theta)$ , parametrized by  $\theta$  which we have to estimate.
- It's difficult to come up with a useful model  $p(x | \theta)$  without making any assumptions about the data.  
*Important* →
- When doing clustering, we implicitly assume that each point  $x$  belongs to some cluster (indicated by  $z$ ).  
⇒ Let's model  $p(x, z | \theta)$  instead!
- 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  $z$ , and are only interested in estimating  $\theta$ .

# Parameter estimation

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

## Supervised learning

*c know*

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

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

## Unsupervised learning

*Z is unknown*

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

## Latent variables

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

Since we chose to model  $p(X, Z | \theta)$ , we can compute the joint probability and obtain  $p(X | \theta)$  by marginalization

$$p(X | \theta) = \sum_Z p(X, Z | \theta) = \sum_Z p(X | Z, \theta) \cdot p(Z | \theta)$$

*marginalization*  
over all  $Z$

and optimize w.r.t  $\theta$

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

Since the true  $Z$  are never observed and are only our abstraction used to simplify the model, they are called **latent variables**.<sup>1</sup>

<sup>1</sup>Sometimes also called hidden or explanatory variables.

## Gaussian mixture model (GMM)

$$p(x, z | \theta) = \prod_{i=1}^n p(x_i, z_i | \theta)$$

We decompose the joint probability  $p(x, z | \theta)$  using the product rule<sup>2</sup>

$$p(x, z | \theta) = p(x | z, \theta) \cdot p(z | \theta)$$

↙ *previous belief  
prior knowledge*

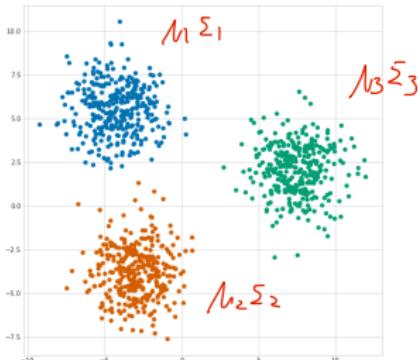
In Gaussian mixture model we assume:

- Cluster prior is categorical  $[\pi_1, \pi_2, \dots, \pi_n]$   
*all non negative*  
 $p(z | \theta) = \text{Cat}(\pi)$       *sum to 1*

- Each cluster has its own multivariate normal distribution  $\pi_i = [\mu_i \Sigma_i]$

$$p(x | z_k = 1, \theta) = \mathcal{N}(x | \mu_k, \Sigma_k)$$

*gaussian*



The parameters consist of the set of parameters of the prior and all conditional distributions  $\theta = \{\pi, \mu, \Sigma\}$ .

---

<sup>2</sup>We assume that samples are drawn i.i.d., so consider only a single  $(x, z)$ .

# Generative process of GMM

We can draw samples from a GMM as follows.

## Setup

We have  $K$  Gaussian components (clusters), each with known  $\{\mu_k, \Sigma_k\}$ , and known prior probabilities  $\pi_k$  for each cluster  $k$ .

## Generative process

$$\pi_i = [\pi_1 \ \pi_2 \ \dots \ \pi_K]$$

1. Draw a one-hot cluster indicator  $\underline{z} \sim \text{Cat}(\pi)$ .  
 $z_k = 1 \iff$  current point belongs to cluster  $k$ .
2. Draw the sample  $x \sim \mathcal{N}(\mu_k, \Sigma_k)$  if  $z_k = 1$ .

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

# GMM likelihood

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

$$\begin{aligned} p(x | \pi, \mu, \Sigma) &= \sum_{k=1}^K p(z_k = 1 | \pi) \cdot p(x | z_k = 1, \mu_k, \Sigma_k) \\ &= \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x | \mu_k, \Sigma_k) \end{aligned}$$

*marginalize  $z$ .*

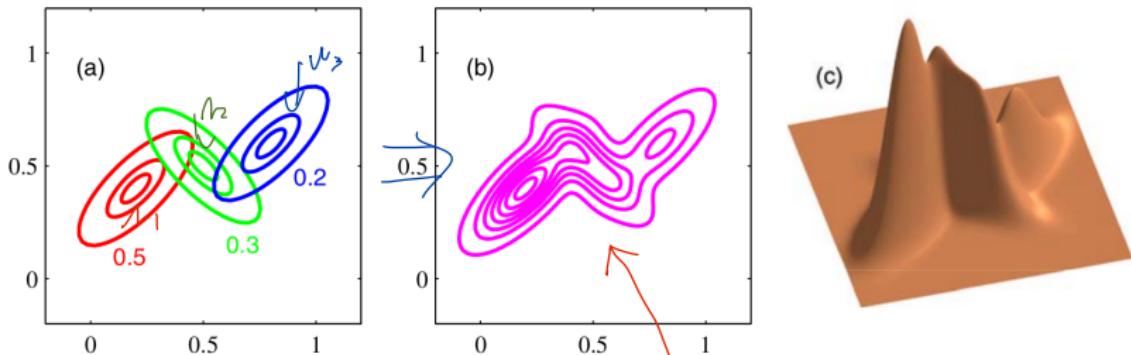
Hence, for the entire dataset  $X = \{x_1, \dots, x_N\}$  the log-likelihood is

$$\log p(X | \pi, \mu, \Sigma) = \sum_{i=1}^N \log \left( \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x_i | \mu_k, \Sigma_k) \right)$$

$$= \ln \prod_{i=1}^N \cdots$$

$$= \sum \ln \cdots$$

# GMM density



- (a) Isocontours of each component  $p(x | \mu_k, \Sigma_k)$  with respective mixing coefficients  $\pi_k$ .  
(b) Isocontours of the GMM distribution

$$p(x | \pi, \mu, \Sigma) = \sum_{k=1}^K \pi_k \cdot \mathcal{N}(x | \mu_k, \Sigma_k)$$

- (c) Surface plot of  $p(x | \pi, \mu, \Sigma)$ .

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)

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

## Inference

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

$$p(Z | \mathbf{X}, \pi, \mu, \Sigma)$$

(This is the actual goal of clustering.)

## Inference in GMM

$$p(z|x) = \frac{p(x|z) p(z)}{p(x)}$$

$\pi = [0.33, 0.33, 0.33]$     $p(z_i | x_i, \theta) = [0.1, 0.1, 0.8]$

By straightforward application of Bayes' rule, we obtain the posterior

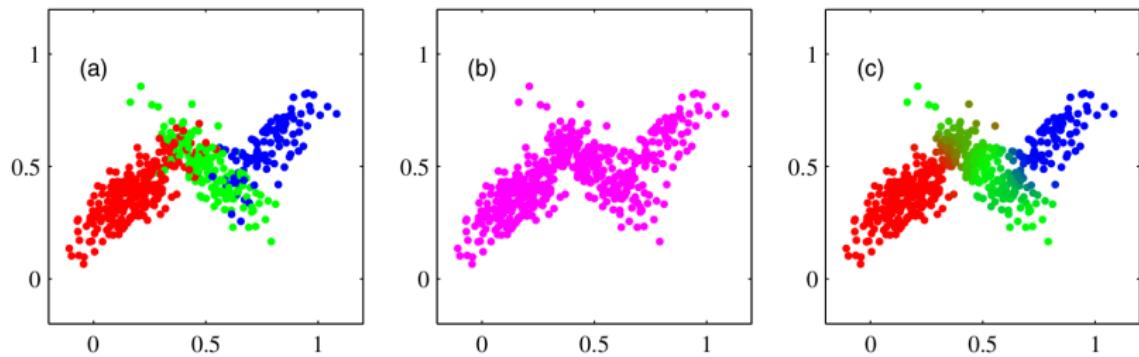
$$\begin{aligned} p(z_{ik} = 1 | x_i, \pi, \mu, \Sigma) &= \frac{p(z_{ik} = 1 | \pi) p(x_i | z_{ik} = 1, \mu_k, \Sigma_k)}{p(x_i | \pi, \mu, \Sigma)} \\ &= \frac{\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)} \end{aligned}$$

over the entries of  $Z$ . We will call this quantity the responsibility of component  $k$  for the observation  $i$ .

We also introduce shorthand notation for it

$$\boxed{\gamma(z_{ik})} = p(z_{ik} = 1 | x_i, \pi, \mu, \Sigma)$$

## Inference: Example



- (a) Data colored according to the true underlying  $Z$ .
- (b) Observed data  $X$ .
- (c) Data colored according to the inferred  $\gamma(\hat{Z})$  (with  $\pi, \mu, \Sigma$  known).

---

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

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

*difficult to compute  
other way*

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

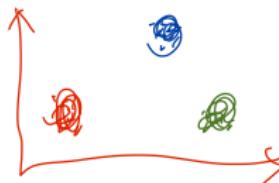
This happens to be a serious problem, as this means that the derivatives w.r.t. the model parameters  $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$  have no analytical roots.<sup>3</sup>

Is there anything that we can do in this case?

---

<sup>3</sup>i.e. we cannot obtain closed form expressions for the optimal values of  $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ .

## Learning in GMM



Recall from the Linear Classification lecture (Linear Discriminant Analysis) that parameter estimation is rather straightforward if  $Z$  is known.

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

$$\pi^*, \mu^*, \Sigma^* = \arg \max_{\pi, \mu, \Sigma} \log p(\mathbf{X}, \mathbf{Z} | \pi, \mu, \Sigma) \quad (*)$$

Too bad we don't know anything about  $Z$ ...      Or do we?

# Learning in GMM

Recall from the Linear Classification lecture (Linear Discriminant Analysis) that parameter estimation is rather straightforward if  $Z$  is known.

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

$$\pi^*, \mu^*, \Sigma^* = \arg \max_{\pi, \mu, \Sigma} \log p(X, Z | \pi, \mu, \Sigma) \quad (*)$$

Too bad we don't know anything about  $Z$ ...      Or do we?

Idea: use our initial beliefs  $\gamma_0(Z) = p(Z | X, \pi^{(0)}, \mu^{(0)}, \Sigma^{(0)})$  and solve the easier problem (\*).

## Learning in GMM: from $p(\mathbf{X})$ to $p(\mathbf{X}, \mathbf{Z})$

$$\uparrow \text{①} \log p(\mathbf{X} | \pi, \mu, \Sigma) = \log \left( \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z} | \theta) \right)$$

Instead of optimizing  $p(\mathbf{X})$  directly, we introduce the latent variables  $\mathbf{Z}$  by taking their expectation and formulate the proxy objective

$$\mathbb{E}_{\mathbf{Z} \sim \gamma_t(\mathbf{Z})} [\log p(\mathbf{X}, \mathbf{Z} | \pi, \mu, \Sigma)] = \sum_{\mathbf{z}} \gamma_t(\mathbf{z}) \cdot \log p(\mathbf{x}, \mathbf{z} | \theta)$$

start with  $\gamma_0(\mathbf{z})$

Now the log acts directly on  $p(\mathbf{X}, \mathbf{Z})$  and we can optimize the whole expression. Note, that  $\gamma_t(\mathbf{Z})$  and the optimizers  $\pi^*$ ,  $\mu^*$  and  $\Sigma^*$  recursively depend on each other.

This motivates the **expectation maximization** (EM) algorithm that iterates the following two steps until convergence:

1. Evaluate the posterior distribution  $\gamma_t(\mathbf{Z})$  with respect to the current estimate of the parameters  $\{\pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}\}$
2. Obtain the next iteration of the parameters by solving

$$\pi^{(t+1)}, \mu^{(t+1)}, \Sigma^{(t+1)} = \arg \max_{\pi, \mu, \Sigma} \mathbb{E}_{\mathbf{Z} \sim \gamma_t(\mathbf{Z})} [\log p(\mathbf{X}, \mathbf{Z} | \pi, \mu, \Sigma)]$$

# EM algorithm for GMM

1. Initialize model parameters  $\{\boldsymbol{\pi}^{(0)}, \boldsymbol{\mu}_1^{(0)}, \dots, \boldsymbol{\mu}_K^{(0)}, \boldsymbol{\Sigma}_1^{(0)}, \dots, \boldsymbol{\Sigma}_K^{(0)}\}$ . *guess*
2. **E step.** Evaluate the responsibilities

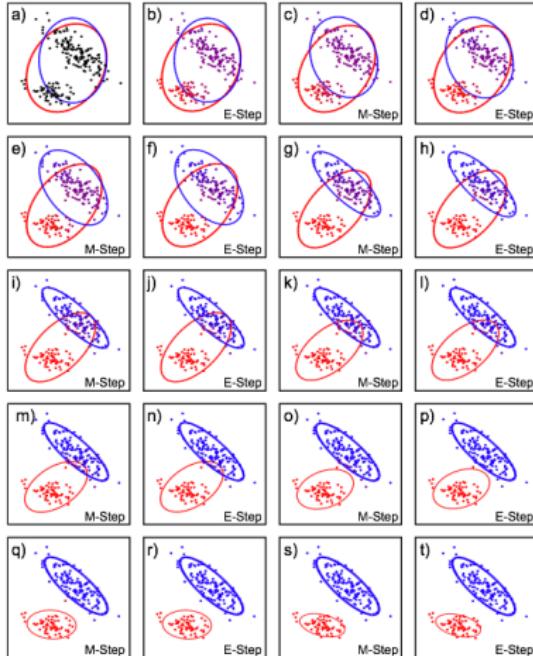
$$\gamma_t(\mathbf{z}_{ik}) = \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{j=1}^K \pi_j^{(t)} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_j^{(t)})}. \quad \text{← Bayesian rule}$$

3. **M step.** Re-estimate the parameters

$$\left| \begin{array}{l} \boldsymbol{\mu}_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_t(\mathbf{z}_{ik}) \mathbf{x}_i \\ \boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{N_k} \sum_{i=1}^N \gamma_t(\mathbf{z}_{ik}) (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})^T \\ \pi_k^{(t+1)} = \frac{N_k}{N} \quad \text{where } N_k = \sum_{i=1}^N \gamma_t(\mathbf{z}_{ik}). \end{array} \right. \quad \begin{array}{l} \text{arg } \max_{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}} E \log p(\mathbf{x}, \mathbf{z} | \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \text{based on m} \end{array}$$

4. Iterate steps 2 & 3 until  $\mathbb{E}_{\mathbf{Z} \sim \gamma_t(\mathbf{Z})} [\log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\pi}^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\Sigma}^{(t)})]$  converges

# EM algorithm for GMM: Demo



Gif: [https://upload.wikimedia.org/wikipedia/commons/6/69/EM\\_Clustering\\_of\\_Old\\_Faithful\\_data.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 in general

$$\text{EM for } \Rightarrow \log p(x|\theta) = \sum_z (\log p(x,z|\theta))$$

The Expectation-Maximization algorithm (EM) is applicable to more models than just GMMs.

In its most general form, the two steps are

- **E step** - evaluate the posterior  $\gamma_t(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta}^{(t)})$ .
- **M step** - maximize the expected joint log-likelihood  $\log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})$  w.r.t.  $\boldsymbol{\theta}$  under the current beliefs  $\gamma_t(\mathbf{Z})$

$$\boldsymbol{\theta}^{(t+1)} = \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{Z} \sim \gamma_t(\mathbf{Z})} [\log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})]$$

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

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

# EM algorithm: A justification

Let  $p(\mathbf{X} | \boldsymbol{\theta})$  be a parametric probabilistic model with latent variables  $\mathbf{Z}$  and  $q(\mathbf{Z})$  any probability distribution. Then we have

$$\begin{aligned}\log p(\mathbf{X} | \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log p(\mathbf{X} | \boldsymbol{\theta}) \\&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})} \quad p(x, z | \theta) = p(z | x, \theta) \cdot p(x | \theta) \\&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})} \cdot \frac{q(\mathbf{Z})}{q(\mathbf{Z})} \\&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \\&= \mathbb{E}_{\mathbf{Z} \sim q} \left[ \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right] + \text{KL}(q || p(\cdot | \mathbf{X}, \boldsymbol{\theta})) .\end{aligned}$$

$\mathcal{L}(q, \boldsymbol{\theta})$   
Lower-bound

$\geq 0.$

$\mathcal{L}(q, \boldsymbol{\theta})$  is a lower-bound on  $\log p(\mathbf{X} | \boldsymbol{\theta})$  for any  $q$ .

## EM algorithm: The E-step

$$\log p(\mathbf{X} | \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q || p(\cdot | \mathbf{X}, \boldsymbol{\theta}))$$

0

The lefthand side is constant with respect to q and the  $\text{KL}$  divergence is non-negative. Therefore  $\mathcal{L}(q, \boldsymbol{\theta})$  is maximal when the  $\text{KL}$  divergence is 0 which happens when  $q(\mathbf{Z}) = \gamma(\mathbf{Z}) = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$ .

This constitutes the E-step and ensures that the lower bound  $\mathcal{L}(q, \boldsymbol{\theta})$  is tight, i.e.

$$\log p(\mathbf{X} | \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}).$$

$$\hookrightarrow q = p(\mathbf{Z} | \mathbf{X}, \boldsymbol{\theta})$$

Note that

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{Z} \sim q} \left[ \log \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{q(\mathbf{Z})} \right] = \underbrace{\mathbb{E}_{\mathbf{Z} \sim q} [\log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})]}_{\text{our proxy from GMMs}} + \underbrace{\text{H}[q(\cdot)]}_{\text{independent of } \boldsymbol{\theta}}$$

## EM algorithm: The M-step

$$\log p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q \parallel p(\cdot \mid \mathbf{X}, \boldsymbol{\theta}))$$

After the E-step we have  $\log p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta})$ . So if we now perform the M-step and maximize  $\mathcal{L}(q, \boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$ , either

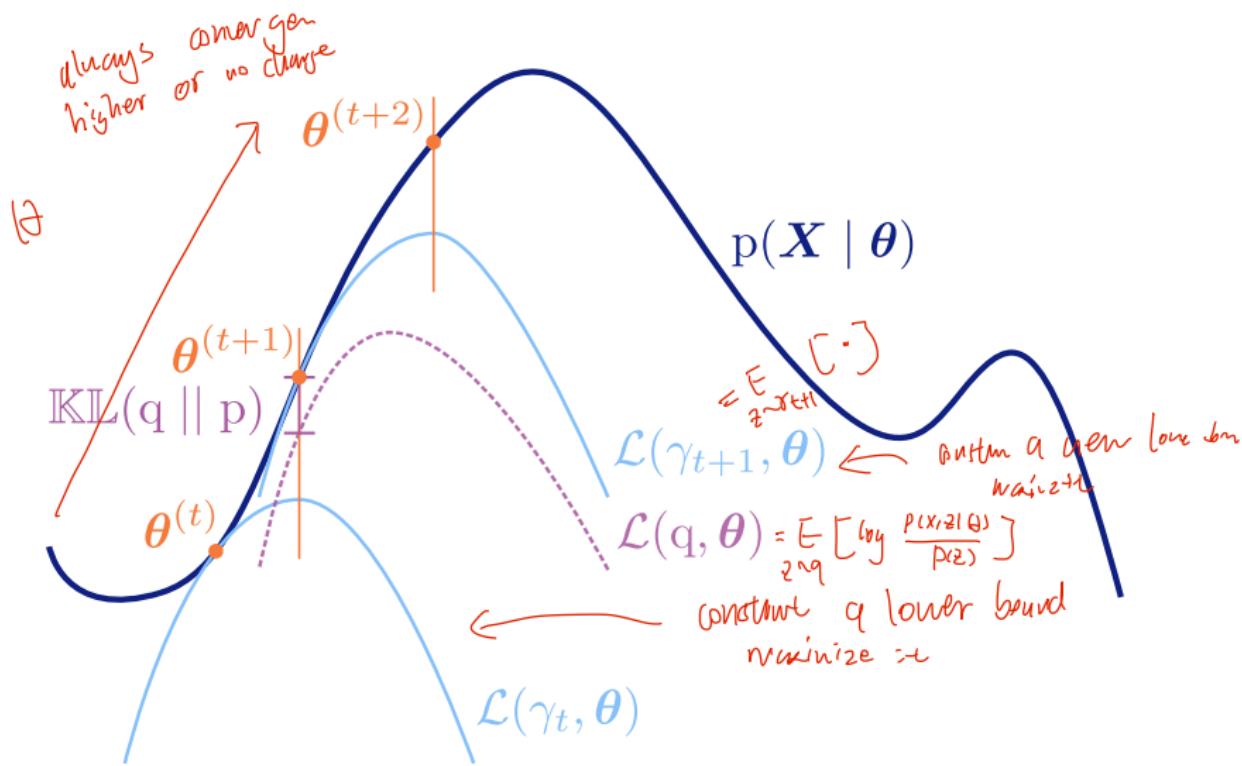
- $\mathcal{L}(q, \boldsymbol{\theta})$  does not change, we are at a local maximum of  $\log p(\mathbf{X} \mid \boldsymbol{\theta})$  and EM converges
- or  $\mathcal{L}(q, \boldsymbol{\theta})$  “pushes” up against  $\log p(\mathbf{X} \mid \boldsymbol{\theta})$ .

In the latter case, we receive parameters  $\boldsymbol{\theta}'$  that have a higher data log-likelihood. Furthermore, the same equation that we started with holds again

$$\log p(\mathbf{X} \mid \boldsymbol{\theta}') = \mathcal{L}(q, \boldsymbol{\theta}') + \text{KL}(q \parallel p(\cdot \mid \mathbf{X}, \boldsymbol{\theta}'))$$

and we can perform another iteration.

# EM algorithm: Illustration



# EM algorithm: Summary

## When should we use EM?

- Intractable to optimize the log-likelihood due to latent variables  $Z$

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

- Easy to optimize the joint probability

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

## The main idea of EM

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

$$\boldsymbol{\theta}^{(t+1)} = \arg \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{Z} \sim \gamma_t(\mathbf{Z})} [\log p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})]$$

## Choosing number of clusters $K$ : Heuristic methods



- **Elbow/knee heuristic:**

Plot within-cluster sum of squared distances for varying  $K$ . Look for a bent (knee/elbow).

- **Gap statistic:**

Compare within-cluster variation to uniform data. Choose smallest  $K$  such that gap statistic is within one std. dev. of gap at  $K + 1$ .

- **Silhouette:**

Per point difference of mean distance within cluster to points in closest other cluster. Maximize mean of all points.

Don't forget cross-validation to prevent overfitting!

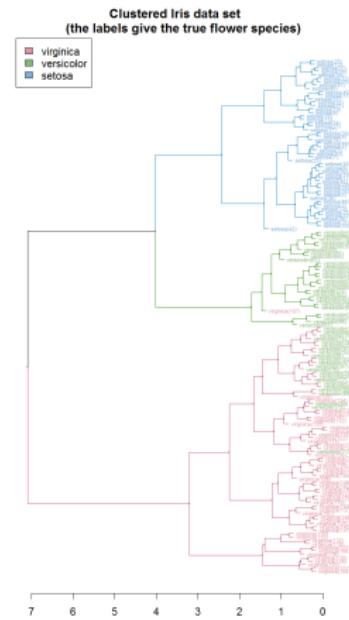
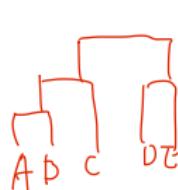
## Choosing number of clusters $K$ : Probabilistic methods

Needs generative model that defines the data likelihood  $\hat{L} = p(\mathbf{X} \mid \hat{\mathbf{Z}}, \hat{\theta})$  (with optimal parameters  $\hat{\mathbf{Z}}, \hat{\theta}$ ).

- Bayesian information criterion (BIC):  
Approximate model likelihood  $p(\mathbf{X} \mid \text{model})$ . Balances number of parameters vs. likelihood,  $\text{BIC} = M \log N - 2 \log \hat{L}$ .  
*number of free parameters*  
*number of parameters*
- Akaike information criterion (AIC):  
Estimate information lost by given model,  $\text{AIC} = 2M - 2 \log \hat{L}$
- Here  $M$  is the number of free parameters (e.g  $K \cdot (D + D^2 + 1)$  for GMM) and  $N$  is the number of samples

# Hierarchical clustering

- **Agglomerative**: Bottom-up, merge cluster pairs iteratively
- **Divisive**: Top-down, split data recursively



# Agglomerative clustering: How to choose pairs?

Cluster pairs are chosen by minimum cluster distance.

Linkage criterion defines cluster distance from sample distance:

- **Complete-linkage clustering**: Maximum sample distance
- **Single-linkage clustering**: Minimum sample distance
- **Unweighted average linkage clustering (UPGMA)**: Mean sample distance
- **Centroid linkage clustering (UPGMC)**: Centroid distance
- **Weighted average linkage clustering (WPGMA)**: Distance defined recursively as average of distances in previous level (before merging)

# Summary

- **Clustering:** Group objects into K groups
- **K-means:** Alternatingly (1) group objects by the closest centroids and (2) set the centroids to the group means  
Note: K-means update equivalent to isotropic GMM with  $\sigma \rightarrow 0!$  (exercise)
- **GMM:** Fit set of Gaussians to the data (via EM), group is the Gaussian with highest probability
- **EM for GMM:** Alternatingly update (1) responsibilities based on parameters and (2) parameters based on responsibilities.
- **EM in general:** Alternatingly (1) evaluate the expected posterior  $\gamma_t(\mathcal{Z})$  (E-step) and (2) update parameters by maximizing  $\mathcal{L}(\gamma_t, \theta)$  (M-step).
- **Agglomerative clustering:** Merge clusters recursively

# Reading material

## Reading material

- Bishop: chapters 9.1, 9.2, 9.3.0, 9.3.1
- Murphy: chapter 11.4.1, 11.4.2, 11.4.7 (optional)