CSE517A Machine Learning **Spring 2025**

Lecture A1: Unsupervised Learning – Clustering

Instructor: Marion Neumann

Reading: FCML Ch6 (Intro), 6.2 (k-Means), 6.3 (Mixture Models); [optional]: ESL 8.5 (EM), PDSH Ch5[1]

1 Introduction

1.1 Unsupervised Learning

Main Question: Can we learn the **hidden structure** in (unlabeled) data?

Problems in USL:

(1) Clustering: partitioning of data into a finite number of *typically* disjoint (or overlapping) groups

(2) Projection/Dimensionality Reduction: project high-dimensional data to lower-dimensional space

Use cases:

[U1] data analysis, data exploration
[U2] data summarization, data visualization
[U3] feature extraction for supervised learning
[U4] data compression
[U5] data/image denoising
[U6] prototype learning (e.g., RBF networks, or to make kernel methods more tractable for big data)
[U7] recommendation systems
[U8] topic modeling

Exercise 1.1. Decide which of the use cases [U1-U8] are *clustering* problems and which are *dimensionality reduction* problems. Justify your decisions with a brief explanation.

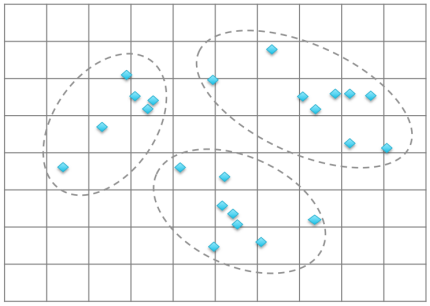
Main Techniques:

(1) Clustering: k-means, Gaussian Mixture Models (GMMs)

(2) Dimensionality Reduction: PCA/SVD/MDS, spectral methods

1.2 Clustering

Goal: partition data points $\{x_i\}_{i=1}^n$ into k groups (clusters) such that points within a cluster are **close** and points in different clusters are **far** away.



Given: $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$

Task: identify clusters and cluster assignment for each point.

[1] [PDSH Ch5] Python Data Science Handbook, J. VanderPlas, 2016, O'Reilly

2 k-means Clustering

k-means is one of the simplest and arguably the most popular clustering algorithm.

Assumptions:

- number of clusters k is known
- each input can only belong to a single cluster

Notation:

- cluster assignment $\mathbf{z}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \in \mathbb{R}^k$ is *one-hot encoding* and the index α of the "1" means the i-th point belongs to α-th cluster, with $i = 1, \ldots, n$ and $\alpha = 1, \ldots, k$
- cluster representative/prototype is the *cluster center* (also *cluster mean*):

$$\boldsymbol{\mu}_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha \mathbf{x}_i}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{1}$$

2.1 Objective and Algorithm

Clustering is inherently *hard* and *subjective*, since we do not have the true labels as for classification. Hence, we cannot compute an error rate to assess the result and in turn to learn the clustering using the approaches from supervised ML.

→ In (unsupervised) clustering, we **do not have class labels** given for the training data.

One heuristic generally accepted is that points in the same cluster should be tight and points in different groups should be as far apart as possible. This heuristic can be modeled by attempting to minimize the **total within-cluster distances** between each data point and its corresponding cluster center leading to the following *objective function*:

$$D(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k, \mathbf{z}_1, \ldots, \mathbf{z}_n) = \sum_{i=1}^n \sum_{\alpha=1}^k [\mathbf{z}_i]_\alpha \, d_{i\alpha}, \tag{2}$$

where $d_{i\alpha}$ is the distance of the i-th input point to cluster α, for instance $d_{i\alpha} = \|\mathbf{x}_i - \boldsymbol{\mu}_\alpha\|^2$.

Unfortunately, this objective is *non-convex* and *non-continuous* and therefore hard to optimize. The brute-force solution of enumerating all possible clusterings, computing their score for Eq. (2), and picking the minimum is not feasible (HINT: this scales exponential in n).

So, we have to resort to an approximate solution for instance by performing *alternate minimization*: for fixed cluster centers we can find the optimal cluster assignments and for fixed cluster assignments we can compute the exact cluster centers.

This algorithm known as **k-means clustering** is stated in Algorithm 1. After initializing the k cluster centers randomly, the first *for* loop updates cluster assignment and the second *for* loop updates cluster centers.

Algorithm 1 k-means clustering

Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ to distinct data points in D
repeat
  for all $i = 1, \ldots, n$ **do**

$$[\mathbf{z}_i]_\alpha = \begin{cases} 1 & \text{if } \alpha = \arg\min_\alpha \underbrace{\|\mathbf{x}_i - \boldsymbol{\mu}_\alpha\|^2}_{=d_{i\alpha}} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

  end for
  for all $\alpha = 1, \ldots, k$ **do**

$$\boldsymbol{\mu}_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha \mathbf{x}_i}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{4}$$

  end for
until convergence

Exercise 2.1. Consider the following two possible termination conditions for k-means:

(1) STOP if cluster assignments do not change anymore.

(2) STOP if cluster centers do not change anymore.

(a) Write out the convergence criterion for each condition.

(b) Proof that (1) and (2) are equivalent.

Exercise 2.2. One of the main ingredients in the k-means algorithm is $d_{i\alpha}$. What do we need to keep in mind with respect to data pre-processing when using distances? Name a concrete strategy to preprocess your data prior to running the k-means algorithm.

2.2 Visualization of the k-means Algorithm

Figure 1 illustrates the k-means algorithm using a data set of erruptions of the Old Faithful geyser in Yellowstone NP. The features are *eruption time* (in min on x-axis) and *waiting time to next eruption* (in min on y-axis).

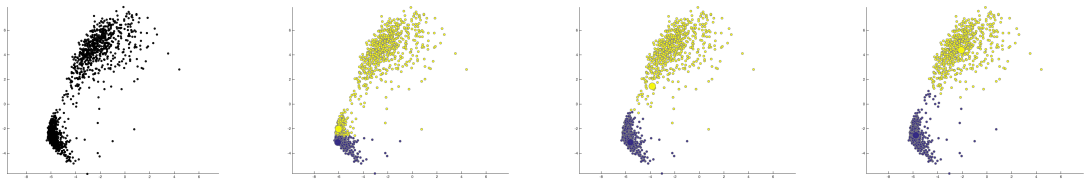


Figure 1: Illustration of k-means. Left: 2D input data (standardized). Middle-1: two randomly chosen initial cluster centers (they are close together) and cluster assignments. Middle-2: after one update the two clusters drift apart. Right: final assignment (captures the two clusters fairly well).

Interpreting the results

Since clustering is inherently unsupervised, we typically have to interpret the results and verify if they makes sense. For $2d$ input data this can happen visually. In our example application, the two clusters reveal that

there are two series of eruptions of the Old Faithful geyser: eruptions with short intervals and eruptions with long intervals. The eruptions with long intervals last longer than short interval eruptions, most likely because longer eruptions require more effort than short interval discharges. Furthermore, the geyser has a higher number of long eruptions than shorter eruptions and the variance in that cluster is higher. Now, we leave it up the the geologists to interpret and use this result in more depth.

For higher-dimensional data we can examine the objective function value or perform *silhouette analysis* (https://en.wikipedia.org/wiki/Silhouette_(clustering)).

2.3 Choosing the Number of Clusters k

Typically, the number of clusters k is unknown. So, we could pick the k that leads to the minimal value of our objective function D, cf. Eq. (2).

→ However, D **always decreases** as k increases.

Verify for yourself that D is minimized at the trivial and uninteresting case of $k = n$, where each input has its own cluster. So, in order to find an appropriate k we can use the following strategy: apply k-means repeatedly with an increasing value of k and observe the value of the objective function D (ideally average over several different initialization). Now, plot the values for D and find the "*kink*" in the curve. This is called the **elbow-method**.

Figure 2 shows a dataset with two inherent clusters and how the objective D, cf. Eq. (2), varies as k increases. We can see that the objective value D will always decrease (on average) as k increases, however, the amount of additional improvement diminishes once the "true" underlying number of clusters has been reached. This effect leads to a *kink* in the decreasing curve of D and points us to a good value for k.

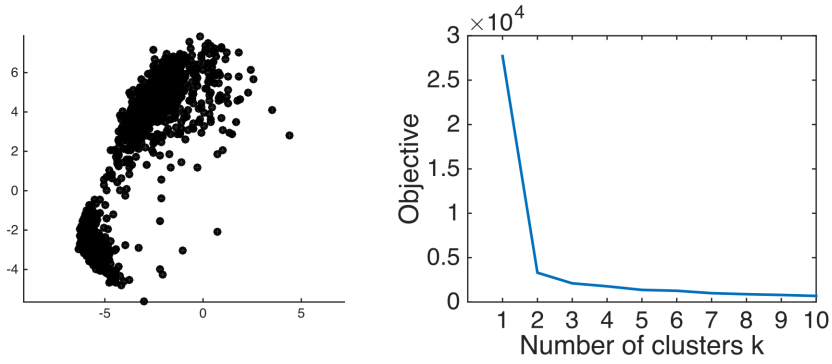


Figure 2: A 2D data set with two inherent clusters

Figure 3 is another 2D toy data set with four inherent clusters.

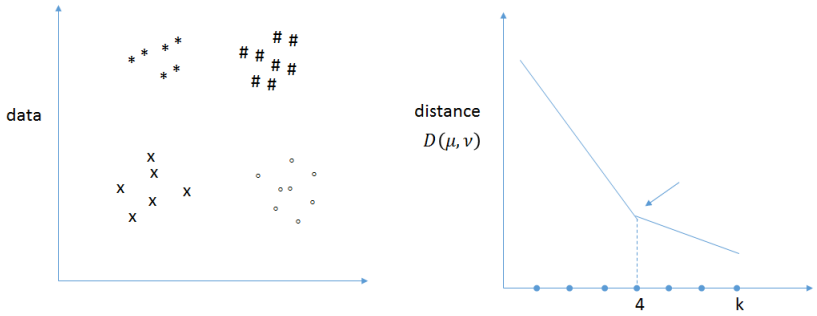


Figure 3: A 2D data set with four inherent clusters.

Silhouette analysis can also be used to select the number of clusters.

2.4 Applications of k-means

[A1] Use cluster centers as **data representatives/prototypes** as illustrated in Fig. 4 (e.g. in RBF networks).

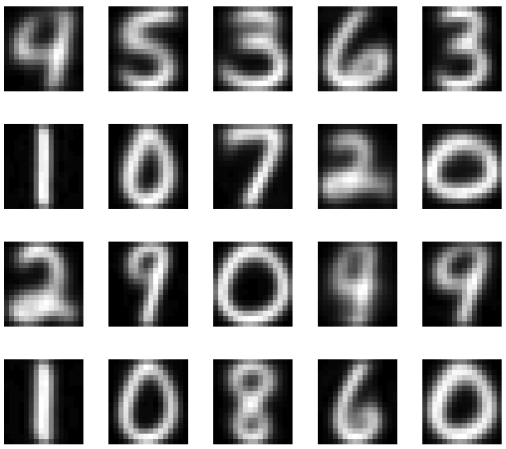


Figure 4: The k = 20 cluster centers of k-means applied to a data set of handwritten digits. The cluster centers consist of the average image in each cluster and are not actual data points in the original data set. However, they clearly show that the data set has strong cluster structure defined by the different digit types.

[A2] Use clusters as **compressed information** as illustrated in Fig. 5 (cf. vector quantization).

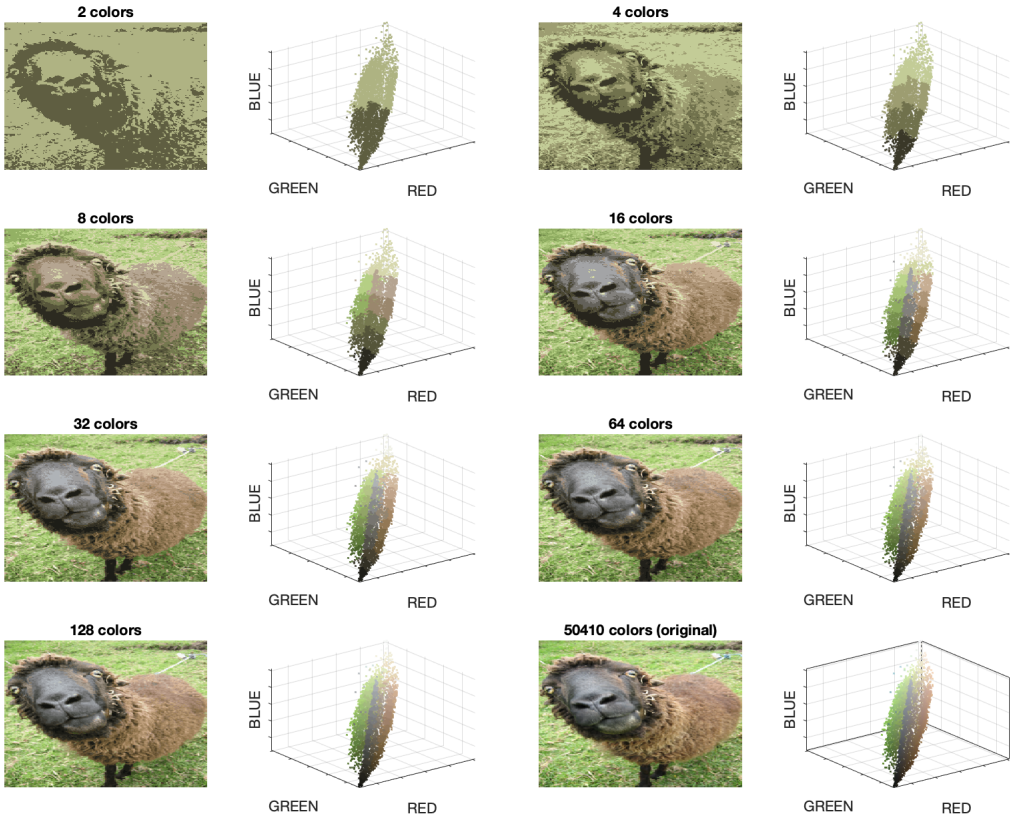


Figure 5: An example application of k-means used for color reduction. The image pixels are represented in a 3D space (red,green,blue) and clustered with k-means. After clustering all pixels are substituted by their corresponding cluster centers, effectively reducing the number of different colors to only k colors.

[A3] Use distance to cluster centers as (low-dimensional) feature representation for supervised learning.

2.5 Summary

+ easy to implement
+ fast for small k
+ easily parallelizable
- can be dominated by outliers and initialization
- only allows hard cluster assignments
- works well when clusters are *spherical*, cf. Figure 6 for an illustration

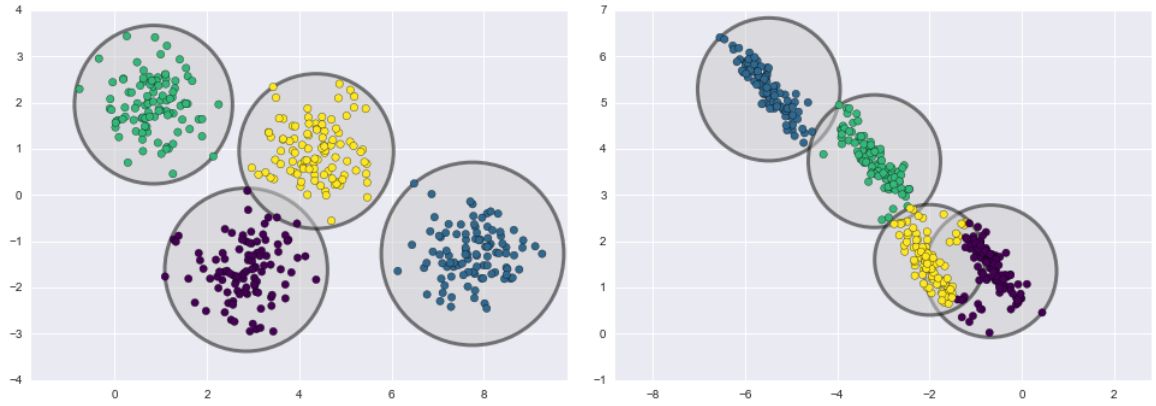


Figure 6: k-means works well for spherical clusters (left), but fails to give good results for oblong or elliptical clusters (right) [PDSH Ch5].

3 Gaussian Mixture Model Clustering

To overcome this last drawback, we will now discuss **Gaussian mixture model** (GMM) clustering. This clustering approach can be thought of as a generalization of the k-means algorithm for *non-spherical clusters* using soft cluster assignments, cf. Figure 7.

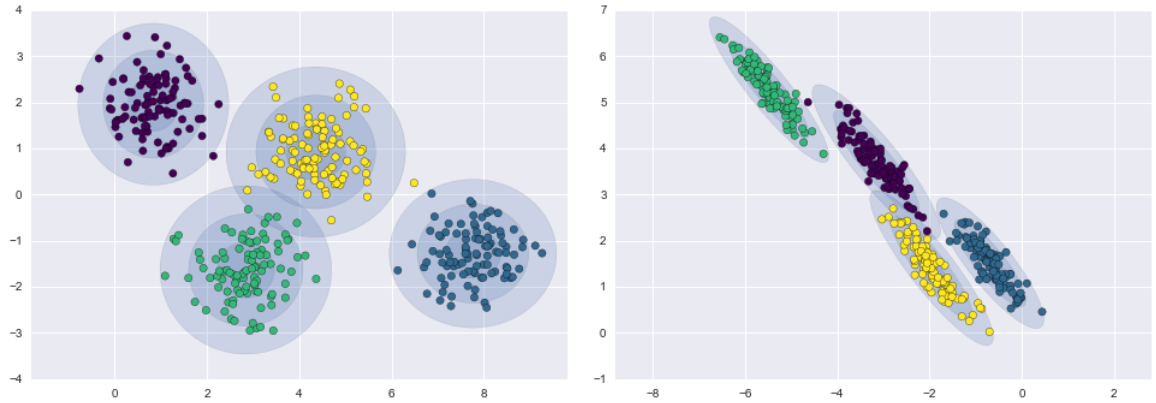


Figure 7: Illustration of GMM soft cluster assignments for spherical clusters (left) and non-spherical clusters (right) [PDSH Ch5].

3.1 Setup

Define $[\mathbf{z}_i]_\alpha \in [0,1]$ as the probability that $\mathbf{x}_i$ belongs to cluster α.

$$\sum_{\alpha=1}^{k} [\mathbf{z}_i]_\alpha = 1 \tag{5}$$

$\Rightarrow$ *soft* cluster assignments

Assumption: dataset D comes from a mixture of Gaussians

$\Rightarrow$ points in each cluster c_α come from a different multivariate Gaussian:

$$\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\alpha}}, \Sigma_\alpha) = p(\mathbf{x}_i \mid c_\alpha) \tag{6}$$

$\Rightarrow$ cluster center = mean of the Gaussian $\boldsymbol{\mu}_\alpha$
$\Rightarrow$ "radius" = covariance of the Gaussian Σ_α

The probability density function (PDF) of the Gaussian Mixture Model (GMM) is given by:

$$p(\mathbf{x} \mid \boldsymbol{\pi}, \{\boldsymbol{\mu}, \Sigma_\alpha\}_{\alpha=1}^k) = \sum_{\alpha=1}^k \pi_\alpha \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha) \tag{7}$$

where π_α are the mixing coefficients acting as the prior probability of each cluster, $\pi_\alpha \in [0,1]$ and $\sum_\alpha \pi_\alpha = 1$. A more careful view of this PDF is as follows (using $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \{\boldsymbol{\mu}_\alpha, \Sigma_\alpha\}_{\alpha=1}^k\}$ and $\boldsymbol{\theta}_\alpha = \{\pi_\alpha, \boldsymbol{\mu}_\alpha, \Sigma_\alpha\}$):

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{\alpha=1}^k p(\mathbf{x}, c = \alpha \mid \boldsymbol{\theta}_\alpha) = \sum_{\alpha=1}^k p(\mathbf{x} \mid c = \alpha, \boldsymbol{\theta}_\alpha)\, p(c = \alpha \mid \boldsymbol{\theta}_\alpha) = \sum_{\alpha=1}^k p(\mathbf{x} \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\, p(c = \alpha \mid \pi_\alpha) \tag{8}$$

which is equivalent to Eq. (7). Figures 8 and 9 illustrate a GMM on 1D and 2D data respectively.

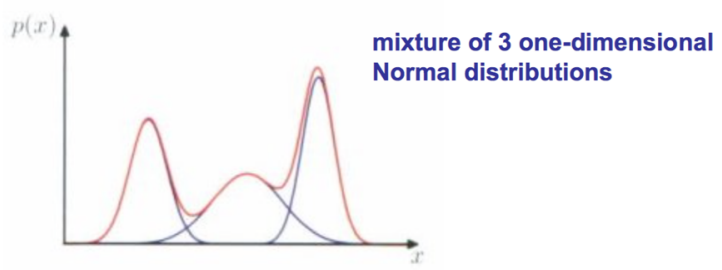


Figure 8: Mixtrue of three Gaussians on 1D input data; The three *mixture components* are shown in blue and the *mixture distribution* $p(x)$ in red.

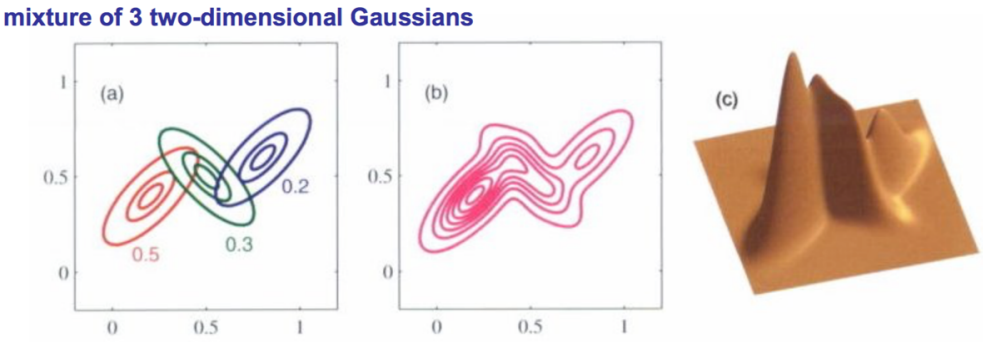


Figure 9: Mixtrue of three Gaussians on 2D input data. (a) shows the *level-sets* of the three *mixture components*. (b) shows the *level-sets* of the *mixture distribution* and (c) shows the *mixture distribution* $p(\mathbf{x})$.

3.2 Generative Model and GMM Algorithm

GMMs are *generative models*, meaning that we can model how the points in D are generated:

- pick a cluster α according to some probability $\boldsymbol{\pi}$ over k categories.
- pick a data point from $\mathcal{N}(\boldsymbol{\mu}_\alpha, \Sigma_\alpha)$

To derive GMM clustering, we can now model the cluster assignments using this generative model:

$$[\mathbf{z}_i]_\alpha = p(c_i = \alpha \mid \mathbf{x}_i) = \frac{p(\mathbf{x}_i \mid c_i = \alpha)\, p(c_i = \alpha)}{\sum_l p(\mathbf{x}_i \mid c_i = l)\, p(c_i = l)} = \frac{p(\mathbf{x}_i \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\, \pi_\alpha}{\sum_l p(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \Sigma_l)\, \pi_l}, \tag{9}$$

where c_i is a random variable that is never observed, which thus is also called a *hidden/latent* variable.

→ to **get the cluster assignments** $\{\mathbf{z}_i\}_i$, we need to estimate the following parameters: $\pi_\alpha, \boldsymbol{\mu}_\alpha, \Sigma_\alpha \ \forall \alpha$

→ to **estimate the parameters** $\pi_\alpha, \boldsymbol{\mu}_\alpha, \Sigma_\alpha \ \forall \alpha$ we need the $\mathbf{z}_i$'s

To resolve this issue, we start by fixing the parameters and computing the cluster assignment. Then we update the parameters based on the new clustering and alternate these steps until convergence leading to the GMM clustering algorithm as given in Algorithm 2. This algorithm is also called **Expectation-Maximization** (EM) with two steps, an (E) step and an (M) step:

(E) step calculates cluster membership probabilities by using the **(E)xpectation** of which cluster the data point belongs to, that is the data-point likelihood evaluated using the current estimate for the parameters normalized by the likelihoods across all clusters

(M) step estimates the parameters by **(M)aximizing** the expected log-likelihood, that is the log-likelihood with the expected cluster assignments

Algorithm 2 GMM Algorithm

Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ to be distinct random data points in D
Initialize $\Sigma_1, \ldots, \Sigma_k$ to $\sigma^2 I$ for any $\sigma > 0$
Initialize $\pi_1, \ldots, \pi_k$ to $\frac{1}{k}$ (uniform prior)
repeat
  for all i & α **do**
    E-step:

$$[\mathbf{z}_i]_\alpha = \frac{p(\mathbf{x}_i \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\pi_\alpha}{\sum_l p(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \Sigma_l)\pi_l} \tag{10}$$

  end for
  for $\alpha = 1, \ldots, k$ **do**
    M-step:

$$\boldsymbol{\mu}_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha \mathbf{x}_i}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{11}$$

$$\Sigma_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha (\mathbf{x}_i - \boldsymbol{\mu}_\alpha)(\mathbf{x}_i - \boldsymbol{\mu}_\alpha)^\top}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{12}$$

$$\pi_\alpha = \frac{1}{n} \sum_{i=1}^n [\mathbf{z}_i]_\alpha \tag{13}$$

  end for
until convergence

3.3 Summary and Remarks

- k-means is a special case with $\Sigma_\alpha = I \ \forall \alpha$
- GMM clustering is a *latent variable model* (LVM) since the $\mathbf{z}_i$ are variables that are never observed (*latent* variables)
- EM actually performs **maximum likelihood estimation** (MLE) to estimate the parameters $\boldsymbol{\theta}$ for a model that depends on *unobserved/latent variables* $c_i \in \{1, ..., k\}$. The likelihood ℓ that we would like

to maximize is given as:

$$\ell(\boldsymbol{\theta}) = p(D \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{x}_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{\alpha=1}^{k} \pi_\alpha p(\mathbf{x}_i \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha) \tag{14}$$

Unfortunately this likelihood cannot be easily optimized. One easy to spot issue here is that once we apply the log to get the sum of the logs, there is yet another sum (over α) "in the way". Try to take the derivatives with respect to the parameters, set them to zero and solve as an exercise.

However, if we **fix the cluster assignments** as given (as in the GMM algorithm), we can derive a likelihood formulation that we can easily optimize. Let's start with the log likelihood:

$$\log \ell(\boldsymbol{\theta}) = \log p(D \mid \boldsymbol{\theta}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i, c_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\mathbf{x}_i, c_i \mid \boldsymbol{\theta}) = \dots \tag{15}$$

Let's **decompose** the sum over all our data points to go through the points in each cluster in groups (since we assume c_i is known) and let's use the cluster indicator variable $\mathbf{z}_i$'s (let's imagine they are hard cluster assignments for now):

$$\begin{aligned} \dots &= \sum_{\alpha=1}^{k} \sum_{i:c_i=\alpha} \log p(\mathbf{x}_i, c_i = \alpha \mid \boldsymbol{\theta}_\alpha) \\ &= \sum_{\alpha=1}^{k} \sum_{i=1}^{n} [\mathbf{z}_i]_\alpha \log p(\mathbf{x}_i, c_i = \alpha \mid \boldsymbol{\theta}_\alpha) = \dots \end{aligned}$$

now, let's use Eq. (8):

$$\begin{aligned} \dots &= \sum_{\alpha=1}^{k} \sum_{i=1}^{n} [\mathbf{z}_i]_\alpha \left[\log p(c_i = \alpha \mid \pi_\alpha) + \log p(\mathbf{x}_i \mid c_i = \alpha, \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\right] \\ &= \sum_{\alpha=1}^{k} \sum_{i=1}^{n} [\mathbf{z}_i]_\alpha \left[\log \pi_\alpha + \log \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\right] =: \mathcal{L}(X \mid \boldsymbol{\theta}) \end{aligned} \tag{16}$$

$\Rightarrow$ maximize log likelihood w.r.t. $\boldsymbol{\mu}_\alpha$ to get

$$\boldsymbol{\mu}_\alpha = \frac{\sum_{i=1}^{n} [\mathbf{z}_i]_\alpha \mathbf{x}_i}{\sum_{i=1}^{n} [\mathbf{z}_i]_\alpha}$$

Note, that intuitively $\boldsymbol{\mu}_\alpha$ is the weighted average of points in cluster α. Mathematically it is the MLE solution that can be derived by taking the derivatives of the *decomposed log likelihood* (Eq. (16)) with respect to the $\boldsymbol{\mu}_\alpha$'s and setting them to zero. Σ_α and π_α can be derived the same way. See FCML 6.3.3 for the full derivation!

Exercise 3.1. Consider the log-likelihood $\mathcal{L}(X \mid \boldsymbol{\theta})$, for the data X given the parameters $\boldsymbol{\theta}$ for our **Gaussian mixture model**.

(a) State $\mathcal{L}(X \mid \boldsymbol{\theta})$. What are the parameters $\boldsymbol{\theta}$?

(b) Derive the parameter update for $\boldsymbol{\mu}_\alpha$ by maximizing $\mathcal{L}(X \mid \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\mu}_\alpha$.

(c) Derive the parameter update for Σ_α by maximizing $\mathcal{L}(X \mid \boldsymbol{\theta})$ w.r.t. Σ_α.

(d) Derive the parameter update for π_α by maximizing $\mathcal{L}(X \mid \boldsymbol{\theta})$ w.r.t. π_α. HINT: when optimizing over π_α we have the constraint that the prior probabilities for the clusters must add up to 1: $\sum_\alpha \pi_\alpha = 1$ (cf. Eq. (7)).

4 Kernel k-means

If our input data has non-convex clusters as for instance the illustrated in Figure 10, we can apply the same trick as for supervised learning: **kernelize the learning method** (*kernel-trick*).

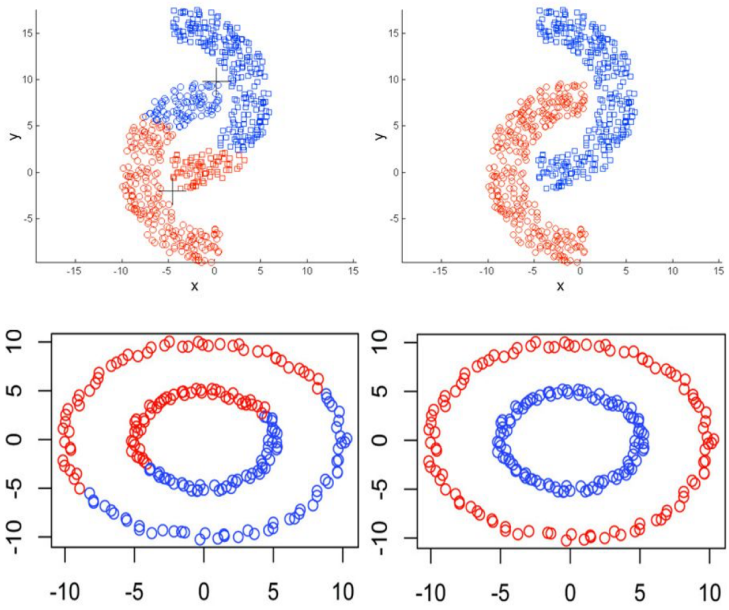


Figure 10: Illustrations of kernel k-means. (left) clusters found by k-means. (right) clusters found by kernel k-means. Note that we cannot visualize the cluster centers for kernel k-means as they "live" in the (*explicit* or *implicit*) transformed feature space defined by the kernel and not in the original input space shown here.

To kernelize k-means, we need to rephrase the equations of Algorithm 1 such that $\mathbf{x}_i$ only appears in inner products. Then, all inner products can be replaced by a kernel function. This will be a homework problem, once we covered kernel methods in-class.

Note, that kernel k-means can cluster *non-vectorial data* (text, graphs, molecules, ...). We only need to be able to compute kernel values among the data objects.

Exercise 4.1. Practice Retrieving!
For this summary exercise, it is intended that your answers are based on **your own** (current) understanding of the concepts (and not on the definitions you read and copy from these notes or from elsewhere). Don't hesitate to **say it out loud** to your seat neighbor, your pet or stuffed animal, or to yourself before **writing it down**. Research studies show that this practice of retrieval and phrasing out loud will help you retain the knowledge!

(a) Using *your own words*, summarize each of the concepts listed above in 2-3 sentences by retrieving the knowledge from the top of your head.

(b) Describe the data points within a cluster as retrieved by a clustering algorithm, such as k-means or GMM clustering.

(c) Name the differences of k-means clustering versus GMM clustering. Identify 2-3 shortcomings of either approach.

(d) Name 3 use cases of clustering.

(e) How do we use the (log) likelihood in clustering?

And always remember: It's not bad to get it wrong. *Getting it wrong is part of learning!* Use your notes or other resources to get the correct answer or come to our office hours to get help!