CSE517A Machine Learning **Spring 2020**

Lecture 12: Unsupervised Machine Learning

Instructor: Marion Neumann *Scribe: Jingyu Xin*

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 selection for supervised learning (make d smaller)
[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

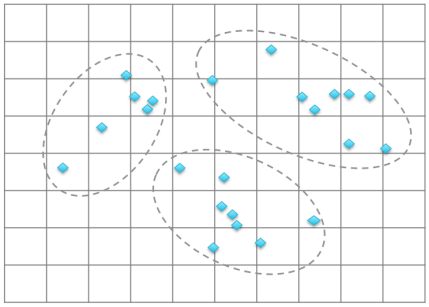

Figure 1: Illustration of a clustering with 3 clusters for 2D input data.

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

We aim to minimize the distance of each input point to its closest 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.

So, we have to resort to an approximation solution for instance by performing alternate minimization. This algorithm known as k-means clustering is stated in Algorithm 1. 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

2.2 Visualization of the k-means Algorithm

Figure 2 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).

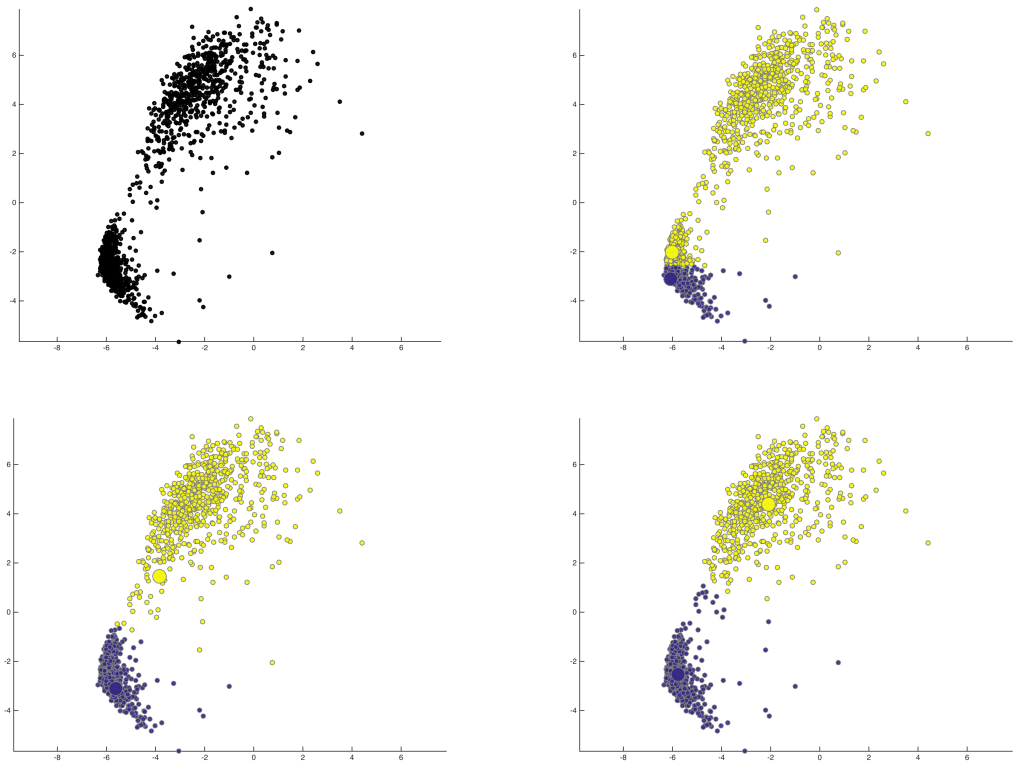

Figure 2: Illustration of k-means. Top/left image: 2D input data (standardized). Top/right: two randomly chosen initial cluster centers (they are close together). Bottom/left: after one update the two clusters drift apart. Bottom/right: final assignment (captures the two clusters fairly well).

Interpreting the results

Since clustering is inherently unsupervised, we typically have to interpret the results and verify if it makes sense. 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.

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 minimual vlue of our objective function D, cf. Eq. (2). However, D always decreases as k increases.[2]
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.

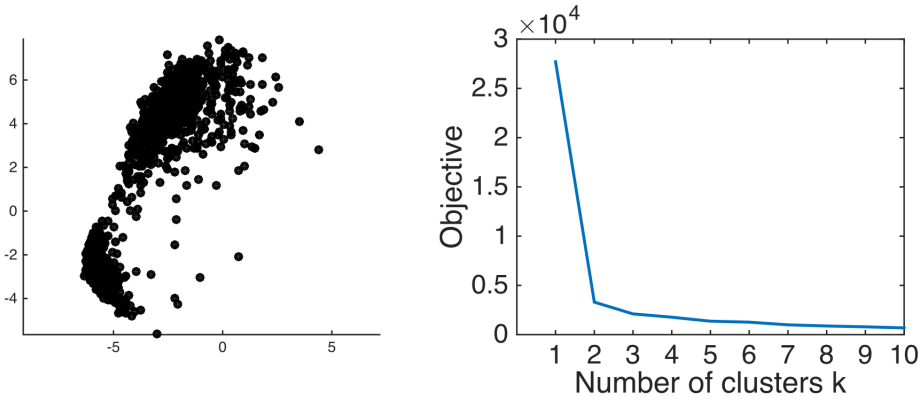

Figure 3: A 2D data set with two inherent clusters

Figure 3 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 4 is another 2D toy data set with four inherent clusters.

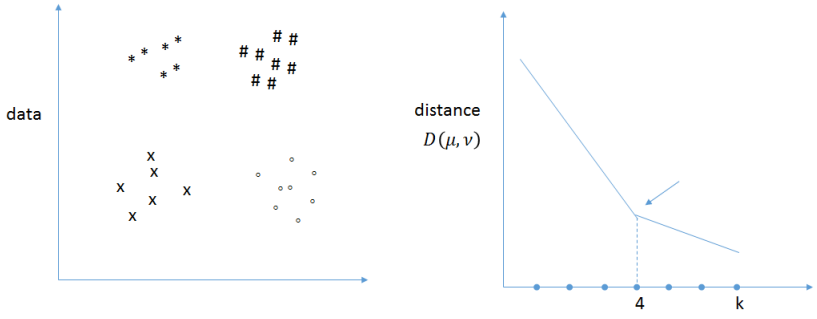

Figure 4: A 2D data set with four inherent clusters.

[2]Note that D is of course minimized at the trivial and uninteresting case of $k = n$, where each input has its own cluster.

2.4 Applications of k-means

[A1] use cluster centers as **data representatives/prototypes** as illustrated in Fig. 5 (e.g. in RBF networks)

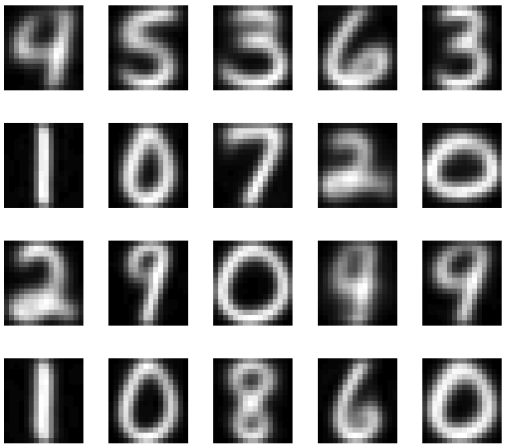

Figure 5: The $k = 20$ cluster centers of k-means applied to a data set of handwritten digits. The cluster centers consist of the average image in each cluster and are not actual data points in the original data set. However, they clearly show that the data set has strong cluster structure defined by the different digit types.

[A2] use clusters as **compressed information** as illustrated in Fig. 6 (cf. vector quantization)

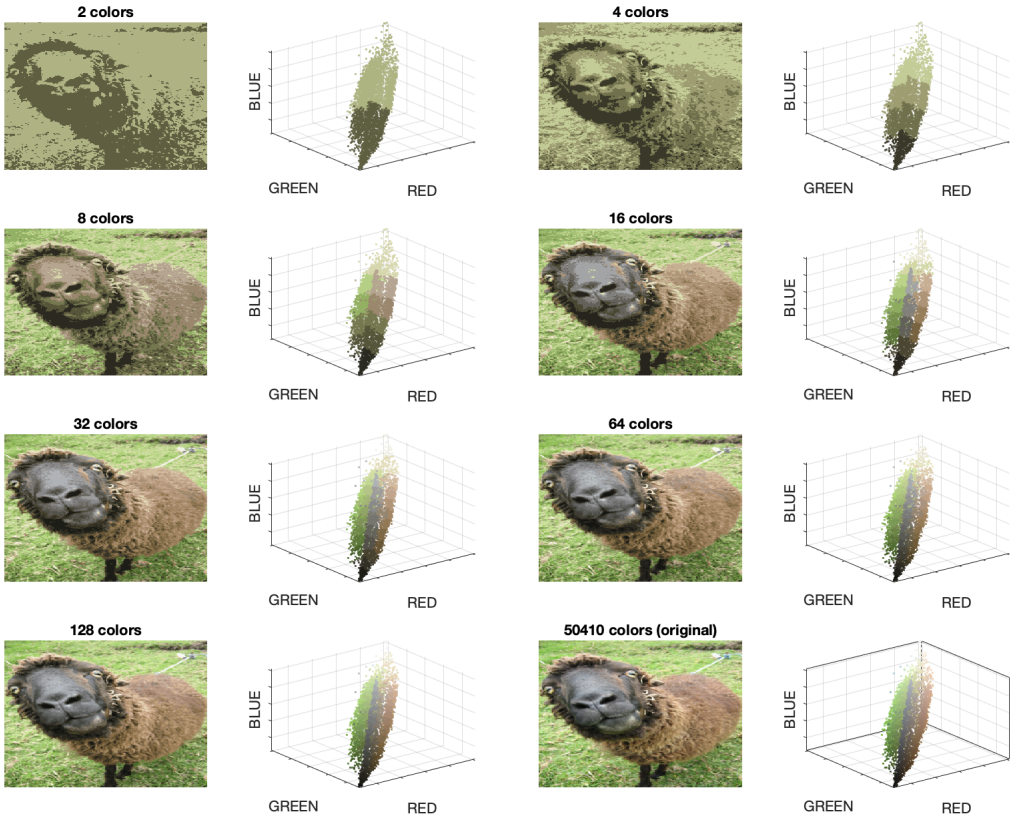

Figure 6: An example application of k-means used for color reduction. The image pixels are represented in a 3D space (red,green,blue) and clustered with k-means. After clustering all pixels are substituted by their corresponding cluster centers, effectively reducing the number of different colors to only k colors.

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 7 for an illustration

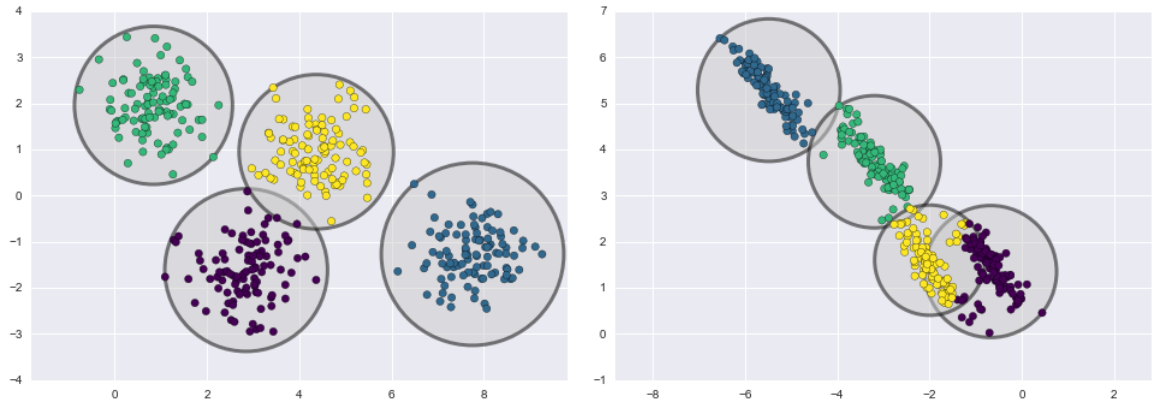

Figure 7: k-means works well for spherical clusters (left), but fails to give good results for oblong or elliptical clusters (right) [PDSH Ch5].

To overcome this last drawback, we will now discuss two generalizations of the k-means model: **kernel k-means** and **Gaussian mixture model clustering**.

3 Kernel k-means

If our input data has non-convex clusters as for instance the illustrated in Figure 8, we can apply the same trick as for supervised learning, kernelize the learning method. 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 works out nicely for the **cluster assignment** in Eq. (3):

$$
\begin{aligned}
d_{i\alpha} &= ||\mathbf{x}_i - \boldsymbol{\mu}_\alpha||^2 = (\mathbf{x}_i - \boldsymbol{\mu}_\alpha)^\top (\mathbf{x}_i - \boldsymbol{\mu}_\alpha) = (\mathbf{x}_i - \frac{\sum_{j=1}^n [\mathbf{z}_j]_\alpha \mathbf{x}_j}{\sum_{j=1}^n [\mathbf{z}_j]_\alpha})^\top (\mathbf{x}_i - \frac{\sum_{j=1}^n [\mathbf{z}_j]_\alpha \mathbf{x}_j}{\sum_{j=1}^n [\mathbf{z}_j]_\alpha}) \\
&= \underbrace{\mathbf{x}_i^\top \mathbf{x}_i}_{\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle} - \frac{2}{\sum_{j=1}^n [\mathbf{z}_j]_\alpha} \sum_{j=1}^n [\mathbf{z}_j] \underbrace{\mathbf{x}_i^\top \mathbf{x}_j}_{\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle} + \frac{1}{(\sum_{j=1}^n [\mathbf{z}_j]_\alpha)^2} \sum_{j=1}^n \sum_{l=1}^n [\mathbf{z}_j]_\alpha [\mathbf{z}_l]_\alpha \underbrace{\mathbf{x}_j^\top \mathbf{x}_l}_{\langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_l) \rangle}
\end{aligned}
\tag{5}
$$

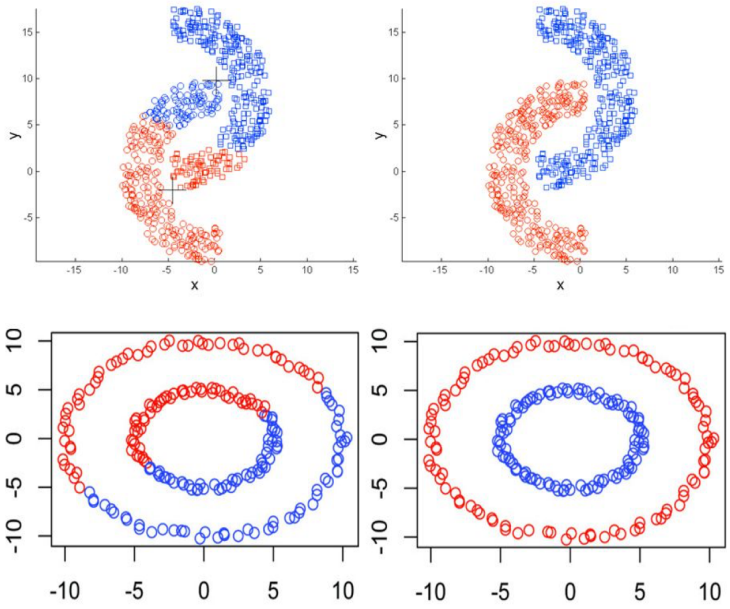

Figure 8: Illustrations of kernel k-means. (left) clusters found by k-means. (right) clusters found by kernel k-means. Note that we cannot visualize the cluster centers for kernel k-means as they "live" in the (*explicit* or *implicit*) transformed feature space defined by the kernel and not in the original input space shown here.

However, in Eq. (4) the **cluster centers** $\boldsymbol{\mu}_\alpha$ cannot be computed for infinite dimensional $\phi(\mathbf{x})$:

$$\boldsymbol{\mu}_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha \phi(\mathbf{x}_i)}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{6}$$

So, we have to come up with an alternative k-means algorithm that does not compute the cluster centers. Instead, it computes the closest cluster α for each data point $\mathbf{x}_i$ based on the minimal $d_{i\alpha}$, cf. Algorithm 2.

Algorithm 2 Kernel k-means

Randomly initialize $\mathbf{z}_1, \ldots, \mathbf{z}_n$ (or use result of standard k-means)
repeat
  for all i & α **do**
    $d_{i\alpha} \leftarrow$ Eq. (5) with $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$
  end for
  update $\mathbf{z}_i$ using $\alpha = \arg\min_\alpha d_{i\alpha}$
  if $\mathbf{z}_i$ do not change **then**
    STOP
  end if
until convergence

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

4 Gaussian Mixture Model Clustering

The Gaussian Mixture Model (GMM) clustering can be thought of as a generalization of the k-means algorithm for non-spherical clusters using soft cluster assignments, cf. Figure 9.

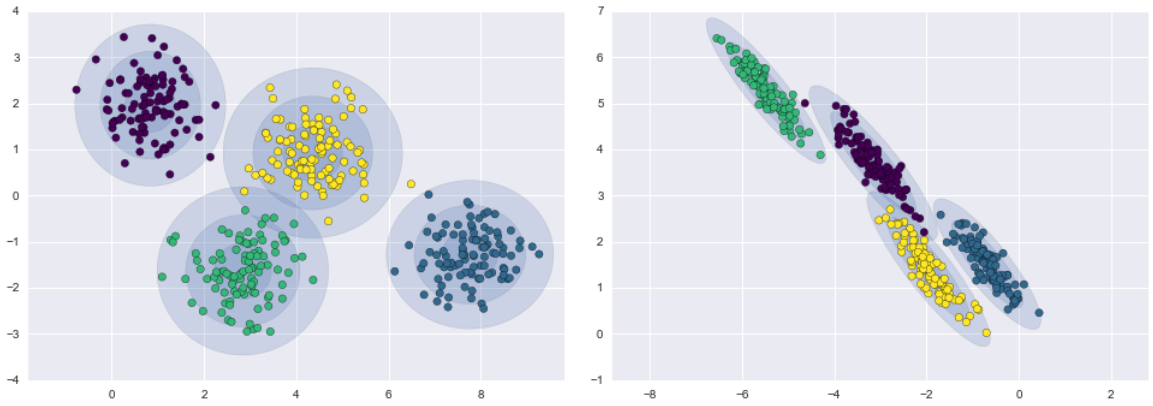

Figure 9: Illustration of GMM for spherical clusters (left) and non-spherical clusters (right) [PDSH Ch5].

4.1 Setup

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

$$\sum_{\alpha=1}^{k} [\mathbf{z}_i]_\alpha = 1 \tag{7}$$

$\Rightarrow$ *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:

$$\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_\alpha, \Sigma_\alpha) = p(\mathbf{x}_i \mid c_\alpha) \tag{8}$$

$\Rightarrow$ 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:

$$p(\mathbf{x} \mid \boldsymbol{\pi}, \{\boldsymbol{\mu}, \Sigma_\alpha\}_{\alpha=1}^{k}) = \sum_{\alpha=1}^{k} \pi_\alpha \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha) \tag{9}$$

where π_α are the mixing coefficients acting as the prior probability of each cluster, $\pi_\alpha \in [0, 1]$ and $\sum_\alpha \pi_\alpha = 1$. Figs. 10 and 11 illustrate a GMM on 1D and 2D data respectively.

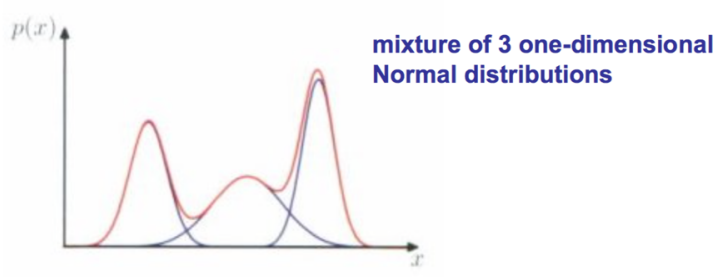

Figure 10: Mixtrue of three Gaussians on 1D input data; The three *mixture components* are shown in blue and the *mixture distribution* $p(x)$ in red.

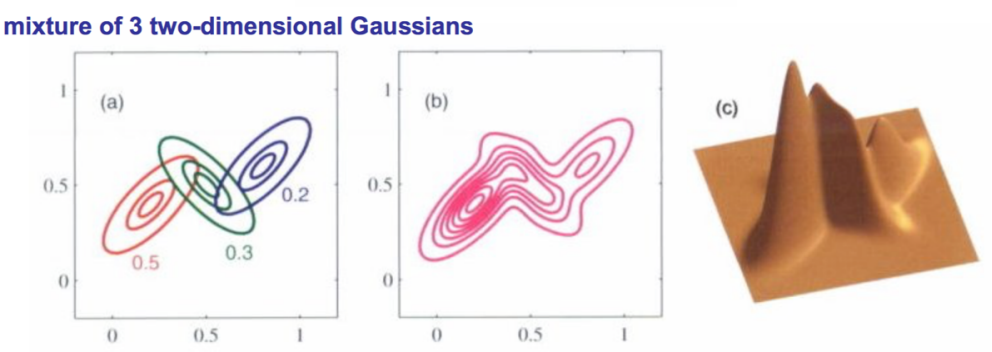

Figure 11: Mixtrue of three Gaussians on 2D input data. (a) shows the *level-sets* of the three *mixture components*. (b) shows the *level-sets* of the *mixture distribution* and (c) shows the *mixture distribution* $p(\mathbf{x})$.

4.2 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:

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

where $[\mathbf{z}_i]$ is a random variable that is never observed, which thus is also called a *hidden/latent* variable.

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

$\rightarrow$ to **estimate the parameters** $\pi_\alpha, \boldsymbol{\mu}_\alpha, \Sigma_\alpha \ \forall \alpha$ using the sample mean, covariance and sample cluster probability estimates, 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 3. 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 the log-likelihood evaluated using the current estimate for the parameters

(M) step estimates the parameters by **(M)aximizing** the expected log-likelihood

Algorithm 3 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:

$$[\mathbf{z}_i]_\alpha = \frac{p(\mathbf{x}_i \mid \boldsymbol{\mu}_\alpha, \Sigma_\alpha)\pi_\alpha}{\sum_l p(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \Sigma_l)\pi_l} \tag{11}$$

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

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

$$\Sigma_\alpha = \frac{\sum_{i=1}^n [\mathbf{z}_i]_\alpha (\mathbf{x}_i - \boldsymbol{\mu}_\alpha)(\mathbf{x}_i - \boldsymbol{\mu}_\alpha)^\top}{\sum_{i=1}^n [\mathbf{z}_i]_\alpha} \tag{13}$$

$$\pi_\alpha = \frac{1}{n} \sum_{i=1}^n [\mathbf{z}_i]_\alpha \tag{14}$$

  end for
until convergence

4.3 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 for a model that depends on *unobserved/latent variables*. The likelihood ℓ that is maximized is given as:

$$\ell(D) = p(D \mid \boldsymbol{\pi}, \{\boldsymbol{\mu}_\alpha, \Sigma_\alpha\}_{\alpha=1}^k) = \prod_{i=1}^n \sum_{\alpha=1}^k \pi_\alpha p(\mathbf{x}_i \mid \boldsymbol{\mu}, \Sigma_\alpha) \tag{15}$$

  $\Rightarrow$ 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 ℓ 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!