Gaussian Mixture Modelling for Clustering

CS342: Machine Learning
2021-2022
Department of Computer Science, University of Warwick, UK

I. Introduction

The coursework focuses on implementing, testing and analyzing a clustering approach based on Gaussian Mixture Modelling (GMM) and the Expectation-Maximization (EM) algorithm. In the lectures, we covered two types of clustering approaches: k-means and soft k-means. Both approaches are based on the core concept of assigning data-points to the cluster with the closest centroid. One of the most important limitations of these approaches is that they do not account for the variance of the clusters, they only consider the mean (i.e., the centroid). Let us take Fig. 1 as an example to explain this further. This figure depicts the result computed by k-means for the case of $k = 2$ and 2D data-points. If we think of k-means as a probabilistic method, each data-point can then be assigned a probability of belonging to one of the two clusters depicted in this figure. Such a probability can be defined by the Euclidean distance between the data-point and each cluster centroid. Therefore, for several of the 2D data-points in Fig. 1, the probability of belonging to one cluster is greater than the probability of belonging to the other cluster. Hence, we can assign each data-point to a single cluster. Once this assignment is done, we can depict each cluster as an area of probability, where the centroid is the point that has the highest probability and as we move away from it in every direction, this probability decreases monotonically. This will produce areas of probability that have circular shapes, where the radius is defined by the point that is farthest from the centroid. The same idea can be applied to the results produced by soft k-means, where each data-point can be also assigned to a single cluster; i.e., the cluster that has the largest membership value for the data-point. Like k-means, soft k-means would produce circular clusters. Such a circular shape is a consequence of accounting only for the mean of the cluster and not its variance.

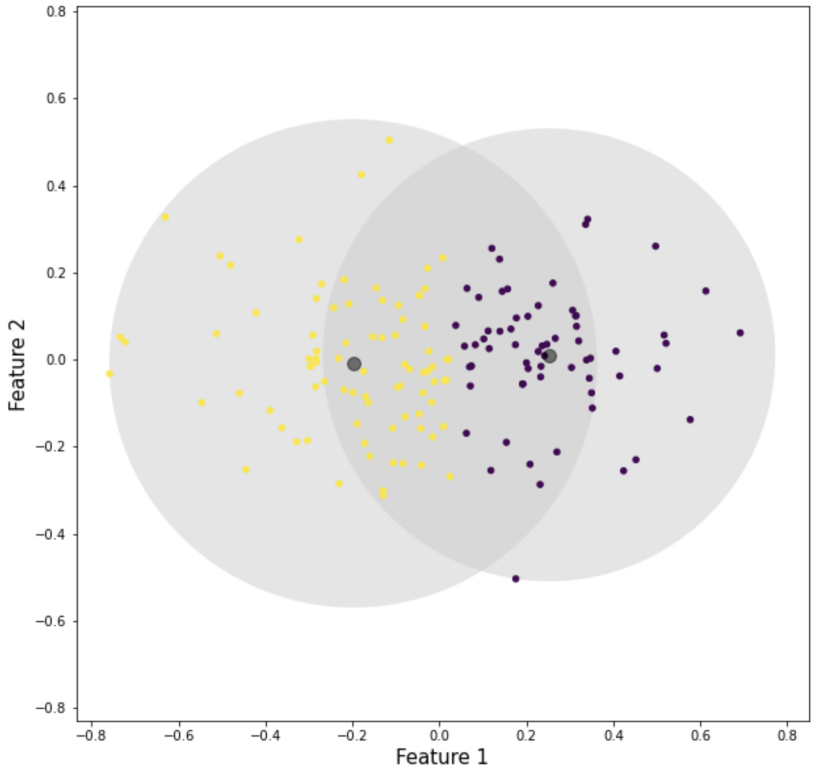


Fig. 1: Two clusters found by k-means for a set of 2D data-points. The shape of each cluster is a circle whose radius is determined by the position of the corresponding cluster members in the 2D space. The centroids are depicted in black.

A cluster, however, may not necessarily have a circular shape, but an elliptical shape. The shape of a cluster is also known as its distribution, where the centroid is considered to be the point with the highest probability. GMM is an approach that can solve this issue. As its name implies, each cluster is modeled according to a different Gaussian distribution. Moreover, GMM offers a *probabilistic* approach to modeling the data by allowing for soft assignments into clusters like soft k-means. This means that each data-point could be part of any of the possible clusters with a certain probability. Just like soft k-means, each cluster, or distribution, has some *responsibility* over a particular data-point. However, unlike soft k-means, which only considers the mean

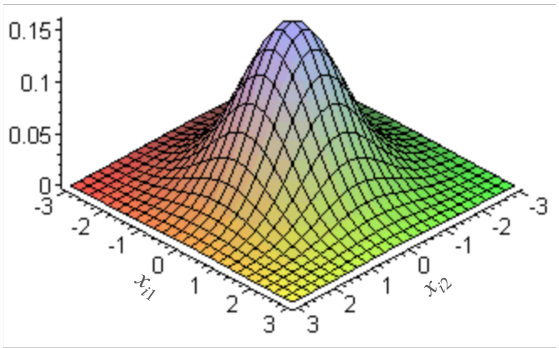


Fig. 2: Sample bivariate Gaussian distribution.

of the clusters, GMM also accounts for their variance.

To complete this coursework, the following subsections provide the necessary technical background to understand GMM and the EM algorithm.

A. Multivariate Gaussian Distribution

Let us consider a dataset $\mathbf{X}$ with one-dimensional data-points $x_i \in \mathbb{R}$, for example, $\mathbf{X}$ may contain data-points that represent grades, velocities, temperatures, etc:

$$\mathbf{X} = \begin{bmatrix} 1.5 \\ 3.8 \\ 7.3 \\ 2.9 \end{bmatrix}. \tag{1}$$

Even with $d = 1$ dimensions, there are many possible distributions that can be used to model $\mathbf{X}$. The most common distribution is the Gaussian (or normal) distribution (see Lecture 2 - The Basics):

$$p(x_i|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \text{ or } x_i \sim \mathcal{N}(\mu, \sigma^2), \tag{2}$$

for mean $\mu \in \mathbb{R}$ and standard deviation $\sigma > 0$ (or variance $\sigma^2 > 0$). Recall that Eq. 2 is the probability density function (PDF) of the normal distribution or likelihood for x_i (see Lecture 10 - MLE and MAP estimation).

If we have $d > 1$ dimensions, we could make each dimension j follow an independent Gaussian, $x_{ij} \sim \mathcal{N}(\mu_j, \sigma_j^2)$. In this case, the joint density $p(\mathbf{x}_i|\mu_1, \mu_2, \ldots, \mu_d, \sigma_1^2, \sigma_2^2, \ldots, \sigma_d^2)$ for a d-dimensional feature vector $\mathbf{x}_i$ can be written as follows:

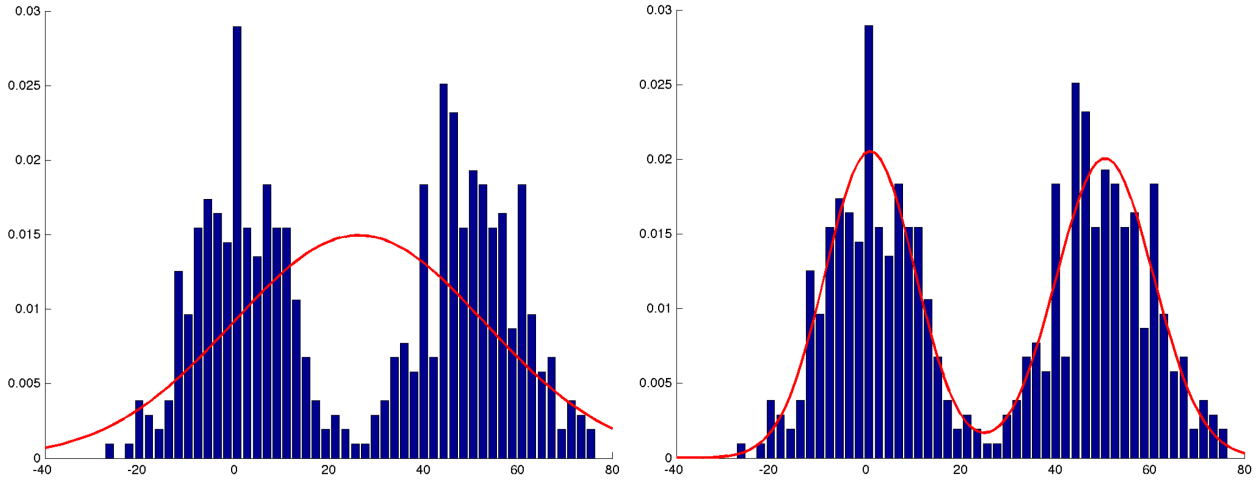
$$\begin{aligned} \prod_{j=1}^{d} p(x_{ij}|\mu_j, \sigma_j^2) &\propto \prod_{j=1}^{d} \exp\left(-\frac{(x_{ij} - \mu_j)^2}{2\sigma_j^2}\right) \\ &= \exp\left(-\frac{1}{2}\sum_{j=1}^{d} \frac{1}{2\sigma_j^2}(x_{ij} - \mu_j)^2\right) \text{ (using } e^a e^b = e^{a+b} \text{)} \\ &= \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu})\right) \text{ (vector/matrix notation),} \end{aligned} \tag{3}$$

where $\boldsymbol{\mu}^{\mathrm{T}} = [\mu_1, \mu_2, \ldots, \mu_d]$ and Σ is a $d \times d$ diagonal matrix with diagonal elements $\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_d^2\}$ (all entries outside the diagonal are all zero). Distributions with this form are known as multivariate Gaussian. Figure 2 shows a sample bivariate Gaussian distribution; i.e., 2D data-points, $\mathbf{x}_i^{\mathrm{T}} = [x_{i1}\ x_{i2}]$.

Using Eq. 3, the PDF for the multivariate Gaussian for d-dimensional feature vectors, $\mathbf{x}_i$, is given by:

$$p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu})\right), \tag{4}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\Sigma \in \mathbb{R}^{d \times d}$, and $|\Sigma|$ is the determinant.



(a) (b)

Fig. 3: (a) The distribution of some data modelled by (a) using a single Gaussian and (b) two Gaussians.

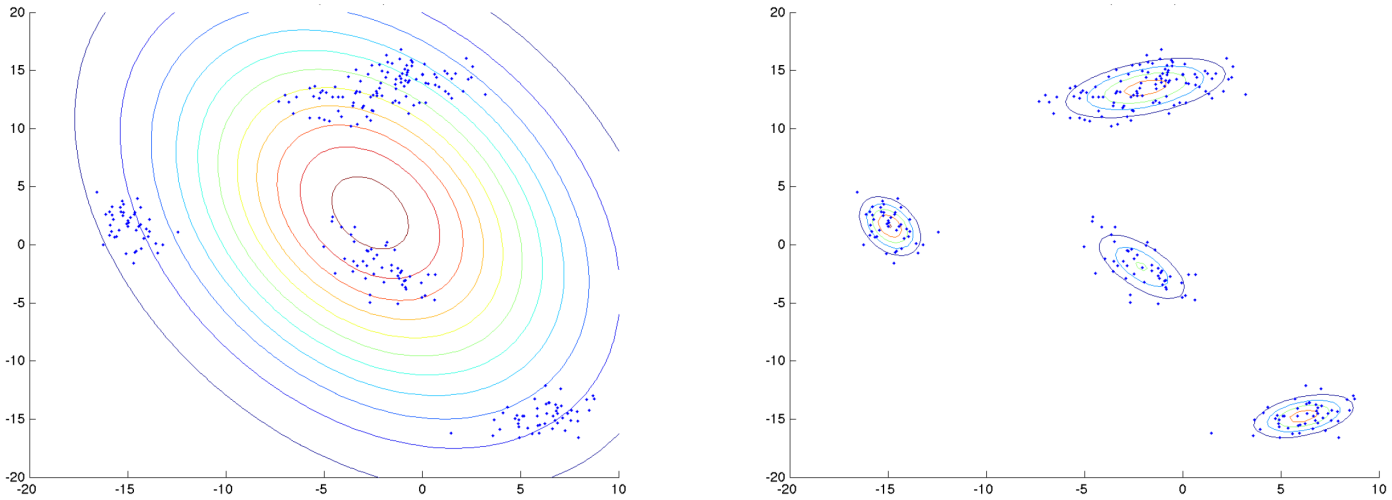


Fig. 4: A single Gaussian (left) vs. a mixture of 4 Gaussians (right) to model 2D data-points. The distribution is depicted as concentric ellipses, where the color goes from intense red (highest probability) to blue (lowest probability).

B. Mixture of Gaussians

Consider the histogram depicted in Fig. 3(a). If we want to model the distribution of the data using a single Gaussian distribution, also known as a uni-modal Gaussian, we obtain a poor fit (see red line). Instead, we can model this distribution by using two Gaussians, where half of the samples are from Gaussian 1, and the other half are from Gaussian 2, as depicted in Fig. 3(b).

For d-dimensional feature vectors, $\mathbf{x}_i$, the PDF for the example in Fig. 3(b) is given by:

$$p(\mathbf{x}_i|\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \frac{1}{2}p(\mathbf{x}_i|\boldsymbol{\mu}_1, \Sigma_1) + \frac{1}{2}p(\mathbf{x}_i|\boldsymbol{\mu}_2, \Sigma_2), \tag{5}$$

where we need the $\left(\frac{1}{2}\right)$ factors so the proportions for the two components add up to 1.

If our data comes from one Gaussian more often than the other, we could also use:

$$p(\mathbf{x}_i|\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma_1, \Sigma_2) = \pi_1 p(\mathbf{x}_i|\boldsymbol{\mu}_1, \Sigma_1) + \pi_2 p(\mathbf{x}_i|\boldsymbol{\mu}_2, \Sigma_2), \tag{6}$$

where π_1 and π_2 are non-negative values that add up to 1. In general, it is possible to have a mixture of k Gaussians with different weights, π_c:

$$p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma) = \sum_{c=1}^{k} \pi_c p(\mathbf{x}_i|\boldsymbol{\mu}_c, \Sigma_c), \tag{7}$$

where π_c represents the probability that we take a sample $\mathbf{x}_i$ from the c^{th} Gaussian. The set of values $\{\pi_1, \pi_2, \ldots, \pi_k\}$ are categorical distribution parameters (non-negative and add up to 1). See Fig. 4 for an example of a set of 2D data-points modeled with a single Gaussian and a mixture of 4 Gaussians.

C. The Expectation-Maximization algorithm

Let us consider a dataset $\mathbf{X}$ to be clustered into k clusters. Let z_i be the cluster number of data-point $\mathbf{x}_i$ in such a dataset; i.e., $z_i \in \{1, 2, \ldots, k\}$ specifies in which cluster data-point $\mathbf{x}_i$ can be found. Furthermore, let us assume that these k clusters may be modeled as a mixture of k Gaussians, like the example in Fig. 4(right), where we can see $k = 4$ clusters. Under this assumption, the probability of data-point $\mathbf{x}_i$ is given by:

$$\begin{aligned} p(\mathbf{x}_i) &= \sum_{c=1}^{k} p(\mathbf{x}_i, z_i = c) \\ &= \sum_{c=1}^{k} p(z_i = c) p(\mathbf{x}_i | z_i = c), \end{aligned} \tag{8}$$

where $p(\mathbf{x}_i, z_i)$ has the form of a Gaussian distribution. Equation 8 can be interpreted as first generating the cluster z_i according to $p(z_i)$, and then given z_i, sampling the data-point $\mathbf{x}_i$ from that cluster.

For our clustering task, we do not know the z_i values. However, if we treat the z_i values as latent variables (or unobserved variables), we can find the $z_i \in \{1, 2, \ldots, k\}$ that generated data-point $\mathbf{x}_i$. Generally, such latent variables are denoted by H. Our data-points represent the data that is observed. Generally, such observed data is denoted by O. To find the z_i values, our goal is then to **maximize** the following conditional probability:

$$p(O|\Theta), \tag{9}$$

where Θ is the set of parameters that define the mixture of Gaussians representing the k clusters. This set of parameters is denoted by $\Theta = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^{k}$. If we maximize the probability in Eq. 9, we can then find Θ. If we know Θ, we can then easily determine in which cluster a data-point may be found. The probability in Eq. 9 is commonly known as the **marginal likelihood**. From Lecture 2 (The Basics), you may recall the marginalization rule, which can also be applied to the marginal likelihood to simplify its optimization:

$$\begin{aligned} p(O|\Theta) &= \sum_{H_1} \sum_{H_2} \cdots \sum_{H_m} p(O, H|\Theta) \\ &= \sum_{H} p(O, H|\Theta) \end{aligned} \tag{10}$$

where we sum over all possible $H = \{H_1, H_2, \ldots, H_m\}$ and $p(O, H|\Theta)$ is called the **complete likelihood**.

Recall from Lecture 10 (MLE and MAP estimation) that we can equivalently minimize the negative log-likelihood:

$$-\log\left(p(O|\Theta)\right) = -\log\left(\sum_{H} p(O, H|\Theta)\right). \tag{11}$$

Unfortunately, the minimization problem in Eq. 11 involves working with a non-convex function and is usually NP-hard. The good news is that there is an approximation to simplify this minimization problem:

$$-\log\left(\sum_{H} p(O, H|\Theta)\right) \approx -\sum_{H} \alpha_H \log\left(p(O, H|\Theta)\right). \tag{12}$$

where α_H is some probability for the assignment H to the hidden variables. Eq. 12 is an expectation over the complete log-likelihood.

For our clustering task under GMM, the observed data are the set of data-points in matrix $\mathbf{X}$. The probability α_H then becomes $p(z|\mathbf{X}, \Theta)$, i.e., the conditional probability of having a cluster z given our observed data $\mathbf{X}$ and a set of parameters Θ defining a mixture of Gaussians. Therefore, Eq. 12 has the following form:

$$-\sum_{z} p(z|\mathbf{X}, \Theta) \log\left(p(\mathbf{X}, z|\Theta)\right). \tag{13}$$

For n data-points in $\mathbf{X}$ and k clusters, Eq. 13 can be expressed as follows:

$$-\sum_{z} p(z|\mathbf{X}, \Theta) \log\left(p(\mathbf{X}, z|\Theta)\right) = -\sum_{i=1}^{n} \sum_{z_i=1}^{k} p(z_i|\mathbf{x}_i, \Theta) \log\left(p(\mathbf{x}_i, z_i|\Theta)\right). \tag{14}$$

When working with a mixture of k Gaussians, the value of $p\left(z_i = c|\mathbf{x}_i, \Theta\right)$ in Eq. 14 is the probability that data-point $\mathbf{x}_i$ comes from cluster c. We call this the responsibility r_c^i of cluster c for data-point $\mathbf{x}_i$. This responsibility can be computed by the Bayes rule as follows:

$$r_c^i \triangleq p\left(z_i = c|\mathbf{x}_i, \Theta\right) = \frac{p\left(\mathbf{x}_i|z_i = c, \Theta\right) p\left(z_i = c|\Theta\right)}{\sum_{c'=1}^{k} p\left(\mathbf{x}_i|z_i = c', \Theta\right) p\left(z_i = c'|\Theta\right)}, \tag{15}$$

where $\triangleq$ denotes "equal by definition to", and $p\left(z_i = c|\Theta\right)$ is the prior (recall Lecture 10 - MLE and MAP estimation).

Since $r_c^i = p\left(z_i = c|\mathbf{x}_i, \Theta\right)$, we can then express Eq. 14 as:

$$-\sum_{i=1}^{n}\sum_{z_i=1}^{k} r_{z_i}^i \log\left(p(\mathbf{x}_i, z_i|\Theta)\right). \tag{16}$$

Equation 16 is the objective function to minimize in order to find the set of parameters $\Theta = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^k$ that define the mixture of k Gaussians, each Gaussian representing a cluster. To make the minimization of Eq. 16 much simpler, we can use the Expectation-Maximization (EM) algorithm. This algorithm aims to minimize Eq. 16 over several iterations using two steps, the E-step or Expectation step and the M-step or Maximization step. Each iteration is denoted by t and for each iteration, a new (and more accurate) set of parameters is found, denoted by Θ^t. The EM algorithm is then very similar to the heuristic approach used by k-means to cluster data. In the E-step, the algorithm:

1) Computes the responsibility of each cluster c for data-point $\mathbf{x}_i$ based on parameters at iteration t, i.e., Θ^t:

$$r_c^i = p\left(z_i = c|\mathbf{x}_i, \Theta^t\right). \tag{17}$$

From Eqs. 15 and 17, we can see that the E-step uses a *soft assignment* for each data-point $\mathbf{x}_i$ (like the soft-assignment used by soft k-means). For example, data-point $\mathbf{x}_i$ could have a membership of 40% with cluster 1, 35% with cluster 2, and 25% with cluster 3, for a total of $k = 3$ clusters and a 100% membership.

From a more formal point of view, the E-step defines the expectation of the complete log-likelihood given the current parameters Θ^t. Mathematically, this is expressed as $Q(\Theta|\Theta^t) = \mathbb{E}_{z|\mathbf{X},\Theta^t}\left[p(\mathbf{X}, z|\Theta)\right]$. The M-step then maximizes this expectation to generate new and more accurate parameters Θ^{t+1}. Mathematically, this is expressed as $\Theta^{t+1} = \arg\max_{\Theta} Q(\Theta|\Theta^t)$. The M-step maximizes this expectation by updating the following:

1) The cluster probabilities based on the n data-points *soft-assigned* to each cluster c (recall Eq. 7):

$$\pi_c^{t+1} = \frac{1}{n}\sum_{i=1}^{n} r_c^i. \tag{18}$$

2) The cluster centroids based on the means of the n data-points *soft-assigned* to each cluster c:

$$\mu_c^{t+1} = \frac{1}{\sum_{i=1}^{n} r_c^i}\sum_{i=1}^{n} r_c^i \mathbf{x}_i. \tag{19}$$

3) The cluster covariances based on the covariances of the n data-points *soft-assigned* to each cluster c:

$$\Sigma_c^{t+1} = \frac{1}{\sum_{i=1}^{n} r_c^i}\sum_{i=1}^{n} r_c^i \left(\mathbf{x}_i - \mu_c^{t+1}\right)\left(\mathbf{x}_i - \mu_c^{t+1}\right)^{\mathrm{T}}. \tag{20}$$

The EM algorithm at the first iteration requires initial values, $\{\pi_c\}_{c=1}^k$, $\{\mu_c\}_{c=1}^k$, and $\{\Sigma_c\}_{c=1}^k$, for the E-step. It can then move to the second step (M-step). This process is repeated over several iterations until the algorithm converges.

II. Methods

A. Dataset

The dataset for this coursework is the Iris Dataset from the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/iris. The dataset contains 3 classes of 50 examples each, where each class refers to a type of iris plant. Each example is represented by a 4D feature vector with the following features:

1) sepal length in cm
2) sepal width in cm
3) petal length in cm
4) petal width in cm

The three classes are:

1) Iris Setosa
2) Iris Versicolour
3) Iris Virginica

The dataset is available on the module webpage.

B. Data visualization

Use PCA to visualize the data-points of the Iris Dataset in 2 dimensions (2D) with their correct label. This requires projecting the data onto the top two principal components (PCs). You may use your implementation of PCA based on SVD from Lab 5.

C. Clustering using GMM and the EM algorithm

Using the 2D data-points computed by PCA in part II-B, complete the following:

1) Implement the EM algorithm to cluster the dataset into $k = 3$ clusters. Recall that the E-step requires initial values at the first iteration. Use k-means clustering to compute these initial values. You can use the python implementation of k-means.
2) Based on the set of initial values, compute the initial responsibilities $\{r_c\}_{c=1}^k$ using the Bayes rule (see Eq. 15). To compute these responsibilities at any iteration of the EM algorithm, you can use the python implementation of the multivariate normal distribution. **Hint:** recall that $p(z_i = c|\Theta)$ is the prior, our prior knowledge about the cluster probabilities.
3) Determine the appropriate number of iterations to run the EM algorithm so it converges.
4) Once the EM algorithm converges, display the final set of parameters $\Theta = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^k$.
5) Recall that each data-point $\mathbf{x}_i$ is soft assigned to all clusters according to values $\{r_c^i\}_{c=1}^k$. However, we can easily assign each data-point $\mathbf{x}_i$ to a single cluster based on the final values $\{r_c^i\}_{c=1}^k$. Doing this *hard* assignment allows computing the accuracy of the EM algorithm to cluster the data-points based on the available labels. Based on this idea, compute how accurately the cluster assignment matches the three classes in the dataset.

III. Deliverables

Submit the following via Tabula:

1) A brief report as a PDF with the figures, plots, and discussions requested (you may use snippets of your code to explain results).
2) Your code as a Jupyter file. Please make sure that the submitted code works correctly.

The breakdown of marks is as follows:

- Section II.B: **5 marks**.
- Section II.C:
  - Results computed by k-means as a figure displaying the cluster centres, the cluster assignments, and the shapes of the clusters (see Fig. 1 for an example). To display the shapes of the clusters, you may use the function `add_patch`: **5 marks**.
  - Accuracy of the cluster assignment provided by k-means with respect to the ground truth labels: **5 marks**.
  - Computation of the initial values for the EM algorithm using k-means: **5 marks**.
  - Implementation of the EM algorithm: **40 marks**.
  - Final set of parameters $\Theta = \{\pi_c, \boldsymbol{\mu}_c, \Sigma_c\}_{c=1}^k$ displayed by the code after convergence: **5 marks**.
  - Accuracy of the cluster assignment provided by the EM algorithm with respect to the ground truth labels (see section II.C.5): **10 marks**.
  - Cluster assignment provided by the EM algorithm (according to section II.C.5) as a figure displaying the cluster centres, the cluster members, and the shapes of the clusters. To display the shapes of the clusters, use the function `draw_ellipse` available on the CS342 website: **10 marks**.
  - Discussions about the appropriate number of iterations for convergence: **5 marks**.
  - Discussions about the accuracy (with respect to the ground truth labels) of k-means with respect to that of the EM-algorithm: **5 marks**.

A total of **5 marks** are available for the quality and presentation of the report, as well as the organization and explanation of your code (in the form of comments within the code). The document should be clearly and logically structured, well written and adequately proofread before submission. The suggested length is between 800-900 words. The standard department late penalties and plagiarism policies are in effect.

Please note that there are several ready-to-use implementations of the EM algorithm and clustering using GMM. You must implement your own solution and **must not** use these libraries. Your code from Lab 5 may be used as a starting point to complete this coursework. The following libraries that implement basic operations may be used for the coursework:

- mean, variance, centre data
- plot matrices, data-points
- matrix and vector multiplication, addition, inverse, transpose
- implementation of argmax and argmin
- computation of Euclidean distance
- SVD - singular value decomposition
- implementation of the multivariate normal distribution
- implementation of k-means

The following libraries that implement the main solutions must not be used for the coursework:

- libraries that implement PCA
- libraries that implement the EM algorithm
- libraries that implement GMM clustering

If you are not sure whether to use a ready-to-use implementation, please email Victor to double-check.