Lab Course Machine Learning
— Problem Set 2—

Eugeniu Vezeteu, Alberto Gonzalo Rodriguez Salgado

June 5, 2020

Part 1: Implementation

Assignment 1 k-means clustering

K-means is an iterative clustering algorithm that tries to split the dataset into k groups. It consists of 2 parts:

1. Assigning each point to its closest centroid

$$C_c = \{i | c = \arg\min_k ||x_i - \mu_k||\}$$

2. Compute new centroids by averaging over the old centroids members

$$\mu_c = \frac{1}{|C_c|} \sum_{i \in C_c} x_i$$

The main drawback of this algorithm is that it's assuming that the data has a spherical pattern. During the assigning step there may appear the problem that the centroid is empty, it means that none of the points were assigned to it. We handled this problem by reinitializing them at a random data points from dataset.

Assignment 2 kmeans-agglo

Agglomerative k-means builds hierarchy of clusters by merging of each 2 closest clusters into one cluster. It consists of these steps:

1. Define cost criteria (K-mean objective)
2. Compute the distance matrix of the clusters
3. Join two closest clusters
4. Repeat step two while no cluster left

Assignment 3 agglo-dendro

We computed dendrogram from merged index and loss from previous example, to shows the hierarchical relationship between clusters. We use dendrogram to analyze cluster relationship and to select a suitable number of clusters.

Assignment 4 Multivariate Gaussian Distribution

In this assignment, we implemented the multivariate Gaussian distribution

$$y_i = \frac{1}{\sqrt{(2\pi)^d * det\,(C)}} exp\left(-\frac{1}{2}\,(X_i - mu)^T\, C^{-1}\,(X_i - mu)\right). \tag{1}$$

Note that we can also formulate $C^{-1}\,(X_i - mu)$ as the liner systems of equations

$$Cq = (X_I - mu)\,. \tag{2}$$

Therefore, we may compute C^{-1} using the standard *inv* function or we may directly compute the solution of 2 using the *solve* function. We investigated how both methods work in order to determine which one to use. The *solve* function does not calculate the inverse C^{-1} at all since it calls one LAPACK routine that factorizes C using the LU decomposition and solves for q using a basic backward and forward substitution. On the other hand the *inv* method computes the inverse C^{-1} by solving for C^{-1} the linears system of equations $CC^{-1} = I$. It computes C^{-1} following the same factorization and forward/backward approach. The key difference is that in this case we are solving a system for a (dxd) matrix and in the other case we are solving a system for a (dx1) vector. Consequently, using the *inv* function would not only take more computation time, but also would need more floating point operations and we would have a greater numerical error. For this reason we decided to use the *solve* approach.

We also considered the possibility of using the *pinv* function which computes the pseudo-inverse. If a matrix is not invertible, it is very probable that it's pseudo-inverse still exists. Indeed, the Scipy function *multivariate_normal* takes this approach and computes the pseudo-inverse and pseudo-determinant. It computes the eigendecomposition of C and cuts the eigenvalues and eigenvectors at a given threshold. Nevertheless, we decided to solve this problem using a simpler approach that is explained in the next assignment.

Assignment 5 EM algorithm for Gaussian Mixture Models (GMM)

In this exercise, we implemented the Gaussian Mixture Model (GMM) and optimized the log-likelihood via the EM (Expectation-Maximization) algorithm resulting in the GMM parameters. For larger data sets e.g USPS, the covariance matrix C may become non invertible and therefore, the determinant would be 0. This would cause problems when computing the multivariate Gaussian distribution, since we can not divide by 0. A common and simple approach to solve this problem is to add a small parameter ϵ to the diagonal of C

$$C = C + \epsilon I. \tag{3}$$

Furtheremore, we considered the case when a cluster may become too small. In this case, one of the Gaussians collapses into a single data point of the data set. Because of this reason the variance of this Gaussian is $\sigma_j^2 = 0$ and we can easily see that this causes the $\frac{1}{\sqrt{2}\sigma}$ component to tend to ∞. One possible way of avoiding this issue is to randomly restart the Gaussian mean when this happens. This is the same approach we took in the case a k-means cluster was empty.

Assignment 6 visualizes the GMM

In this task, we plot the dataset, the GMM means and covariances as ellipses. To plot the covariances as ellipses we:

- sampled 100 points from the $[0 : 2\pi]$
- computed eigendecomposition of each covariance
- plotted the circle which is scaled by $\sqrt{}(eigenvalue)$, oriented by *eigenvector* and translated by mean value.

Part 2: Application

Assignment 7 Analyse the 5gaussians dataset

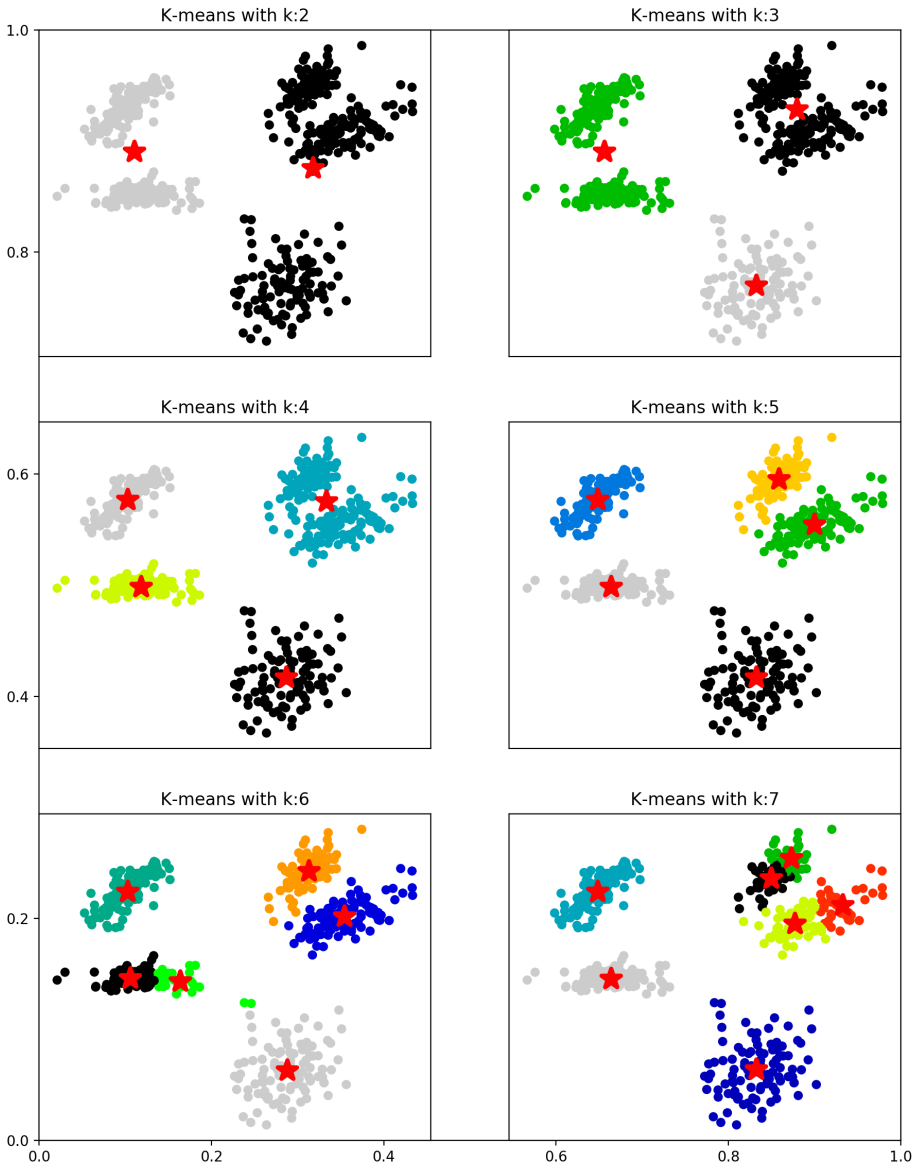


Figure 1: Analysis of 5gaussians dataset with K-means

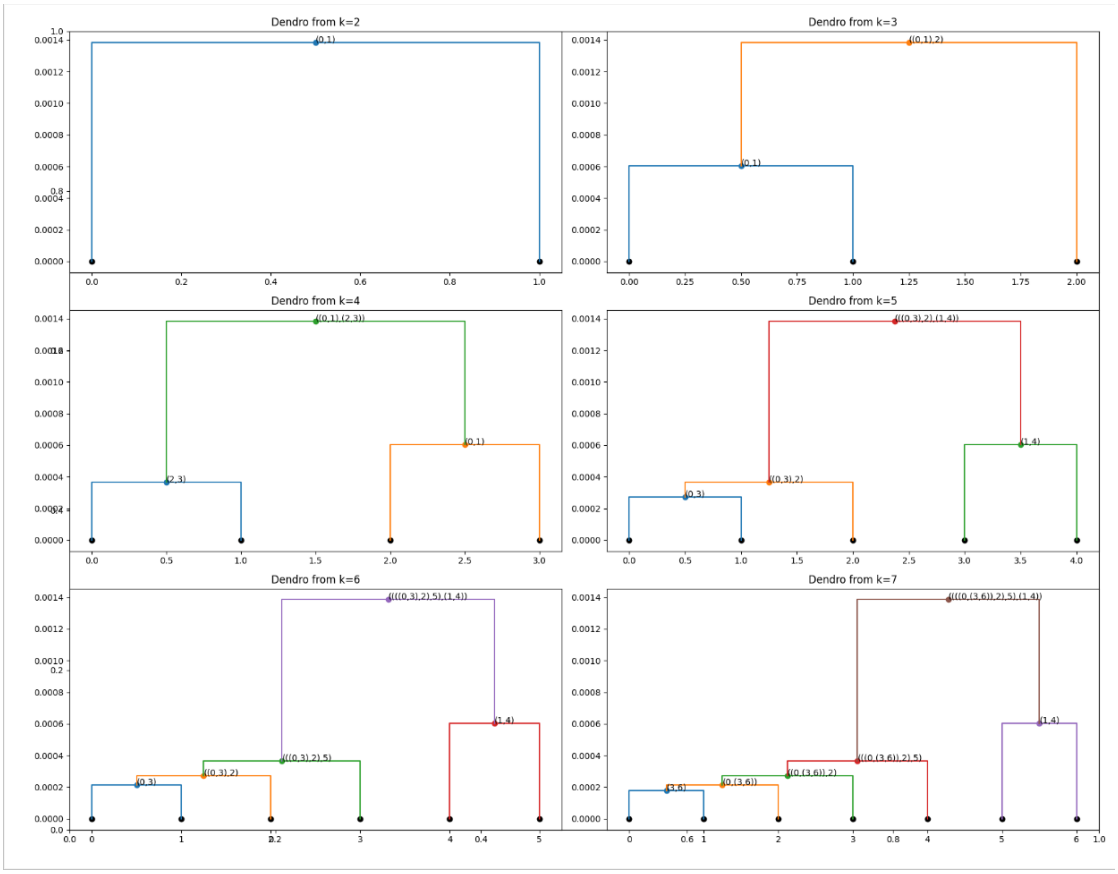


Figure 2: Analysis of 5gaussians Agglomerative K-means

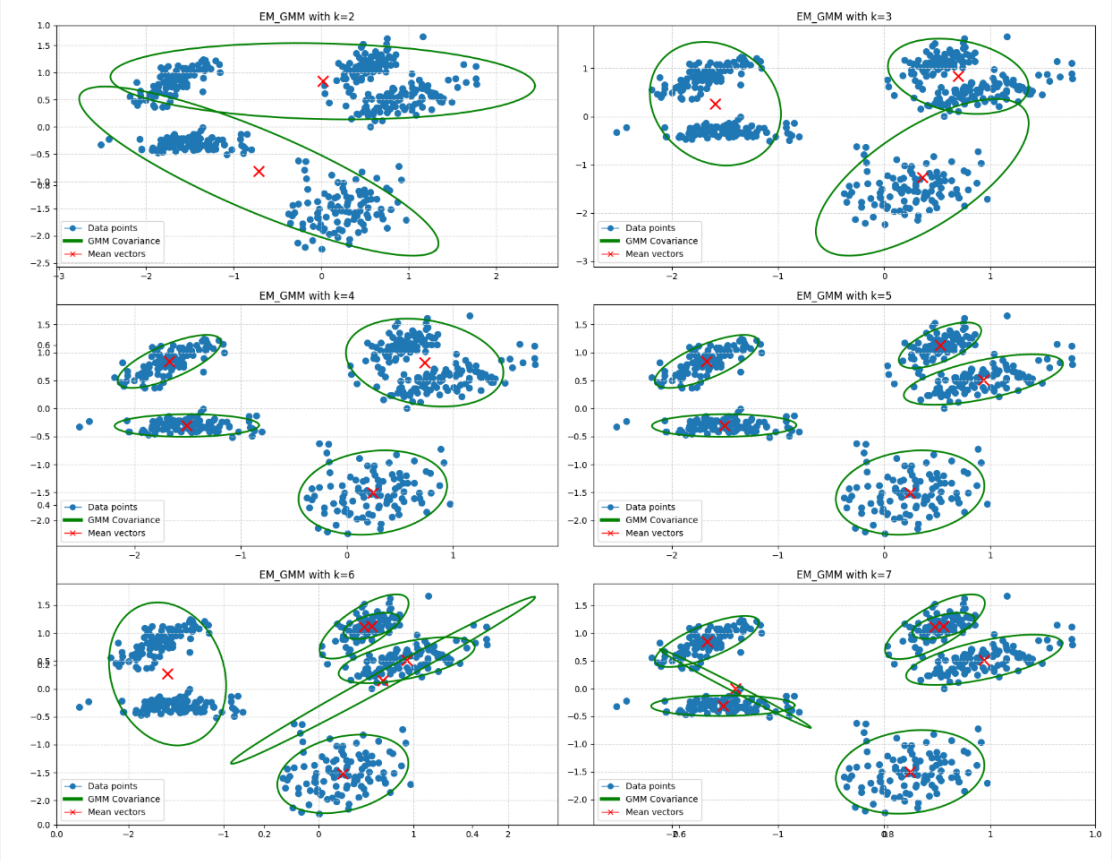


Figure 3: Analysis of 5gaussians GMM

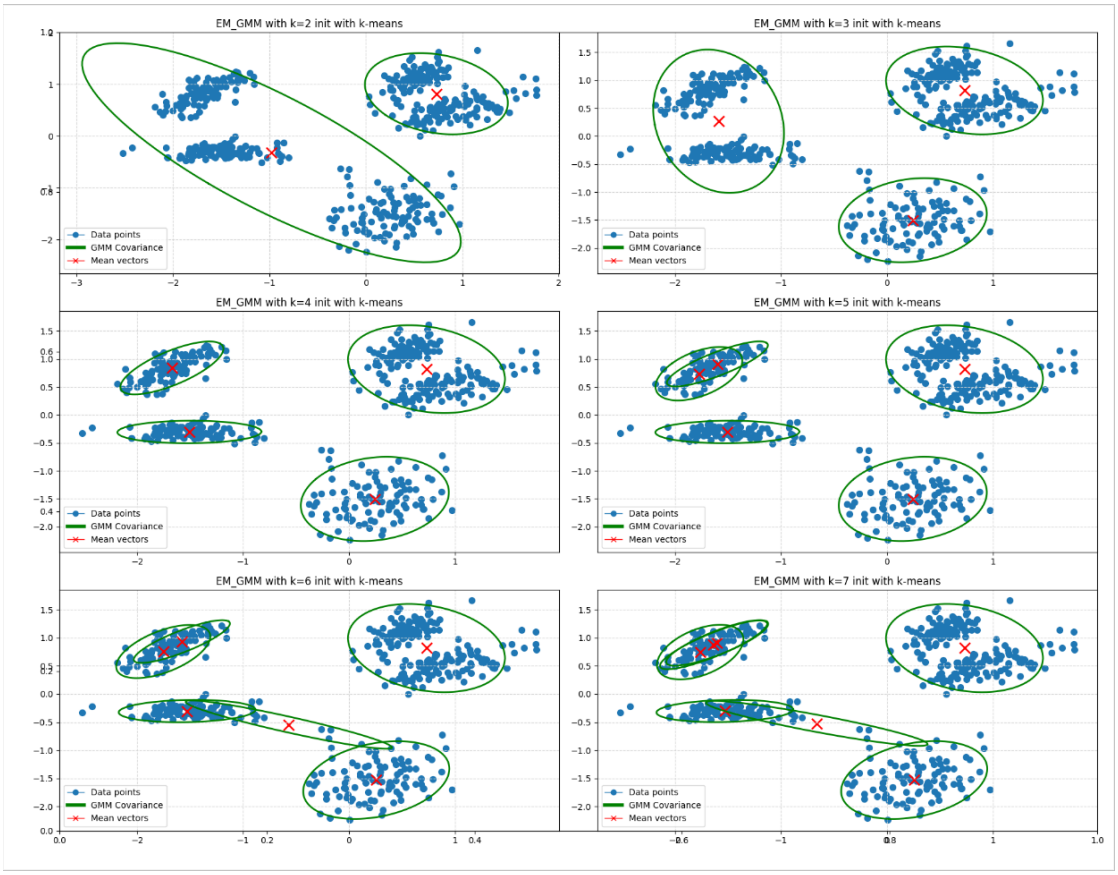
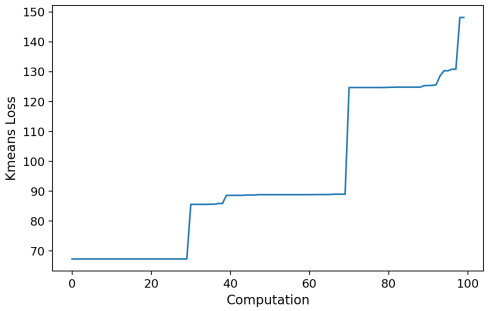
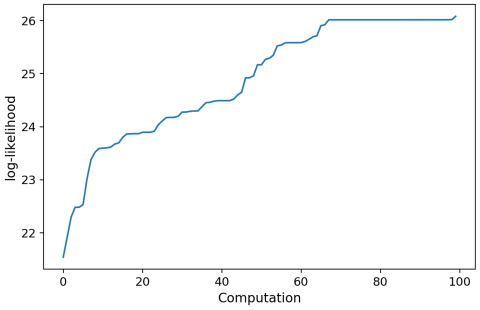


Figure 4: Analysis of 5gaussians GMM with K-means init

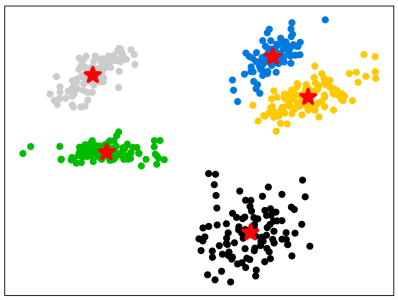


(a) Kmeans loss with $k = 5$ for 100 computations

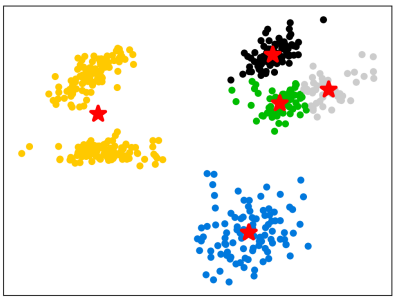


(b) GMM log-likelihood with $k = 5$ for 100 computations

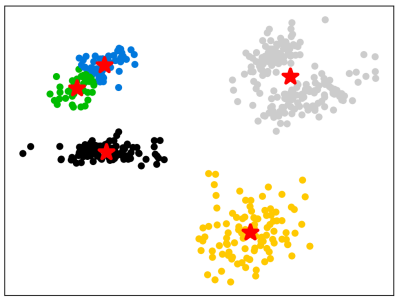
Figure 5



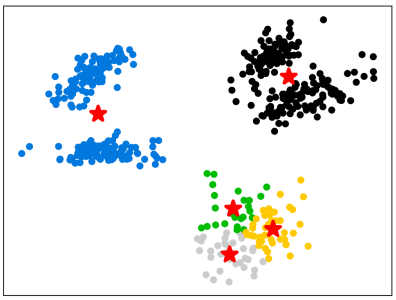
(a) Global minima, loss=67



(b) Local minima, loss=124



(c) Local minima, loss=88



(d) Local minima, loss=148

Figure 6: Clustering using K-means with k=5

1. From Figure 1 we notice that K-means tries to find the solution, but is not guaranteed that it will find it. To ensure that K-means found the optimum solution we need repeat the clustering multiple times and keep the best solution.

2. From Figure 1 we can see that K-means finds the cluster reliably, with k=5. Furthermore, in Figure 3, we notice that GMM with random initialization found the cluster successfully, with k=5.

3. From Figure 5 we can see that GMM does not gain from k-means initialization, the LogLik is almost the same. With K-means, GMM converges faster, with some anomaly at k=5 and k=6, where the number if iterations with k-means is bigger then random initialization iterations. Also, in Figure 4 we notice that with k-means initialization, GMM failed to found the clusters reliably, where k=5.

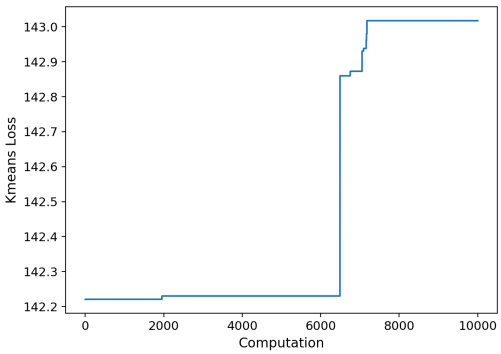
4. We plotted the resulted dendrogramm in Figure 2. We observe relationships between clusters,from the difference between each level. When we do not know the real number of clusters, we use the diagram to see the differences between the clusters, and we can apply some heuristics, by choosing the combination, which has more clusters, and small error. For example, In Figure 2, the last subplot, we noticed that the error for 5 clusters are similar to each other, contrary to a higher level.

5. In order to determine whether the K-means and GMM algorithms do find local or global optima or not, we investigated the case with $k = 5$ clusters and computed 100 times the clustering using both methods. For each computation, we stored the k-Means loss value and the GMM log-likehood value. In the case of k-Means, the algorithm is minimizing the loss function and in the case of GMM, it is maximizing the log-likelihood. In both cases, we reordered the loss/likelihood values from lowest to highest in order to better observe the global optima. Figure 5a shows that the K-means algorithm does not always find an unique, global minima. We can see that there are approximately 3 local minimas and one global one. The local minima with the greatest loss value is ≈ 145. On the other hand, the global minima value is ≈ 69. Figure 6 illustrates the clustering for these local and global optima. We can observe a similar pattern when using GMM for clustering. Figure 5b shows that GMM does neither find always the same maxima. We can observe that GMM finds

for around 30 of the 100 computations the global maxima which has a log-likelihood value of ≈ 26. Nevertheless, depending on the initialization of the algorithms, it finds numerous local maximas within the log-likelihood range of $21 - 26$. To sum up, in both cases it is extremely important to run the clustering algorithms several times and keep track of the loss/log-likelihood values. One possible way of finding the global optima is to run a specific configuration of the algorithm e.g 100 times, plot the loss/loh-likelihood values as in Figure 5 and 6, determine the global optima value from the plot or directly from the loss array and finally specifically look for the computation that produces this loss value.

Assignment 8 Analyse the 2gaussians dataset

(a) Analysis of 2gaussians K-means result



(b) K-means loss after 10.000 computations with k=2

Figure 7: K-means two 2gaussians data set

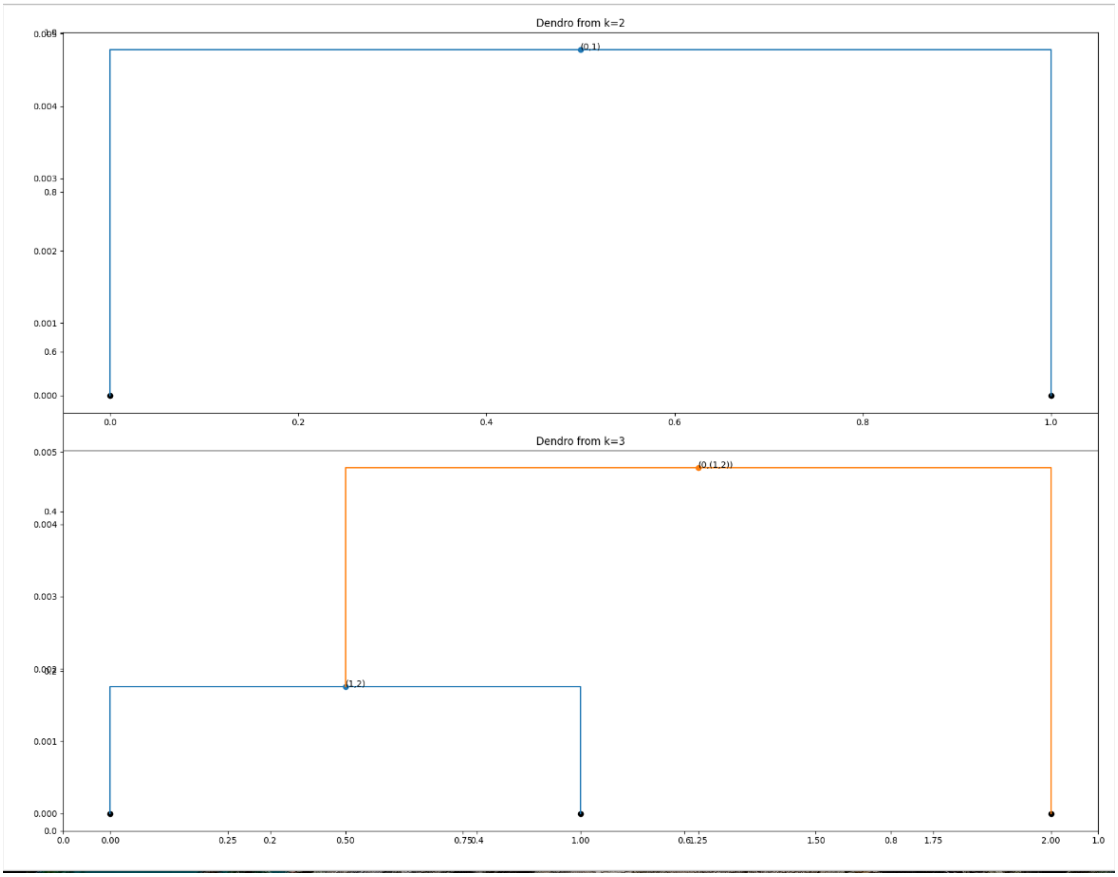
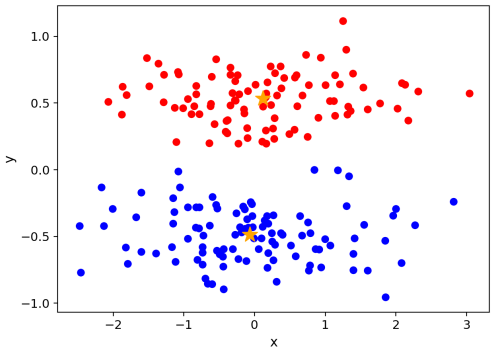
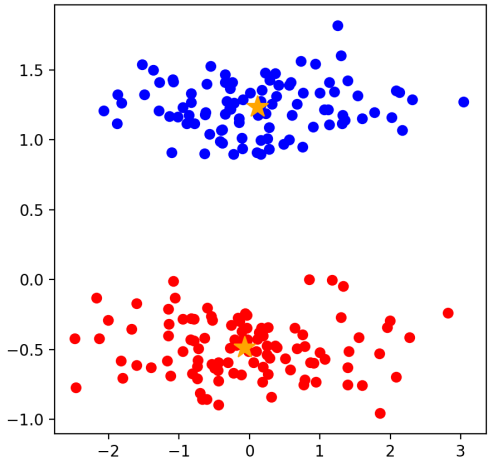


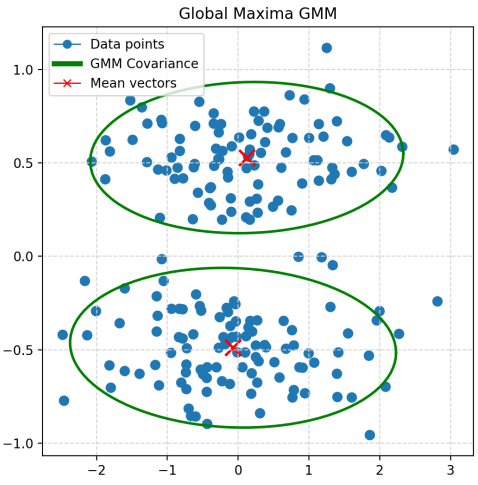
Figure 8: Analysis of 2gaussians K-means dendrogramm



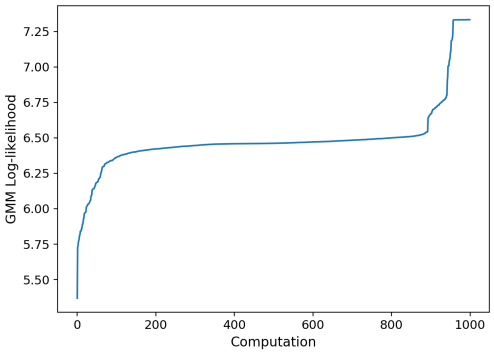
(a) Ideal separation



(b) K-means clustering after separating by y=0.7 both clusters

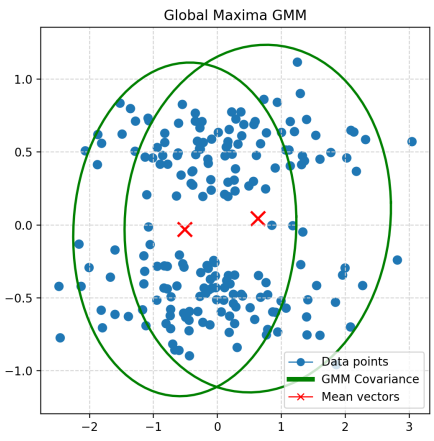


(a) Clustering with k=2

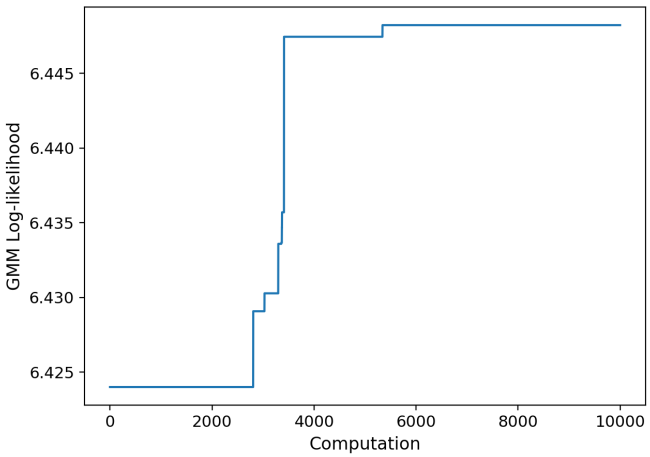


(b) log-likelihoods (local maximas) ordered values for 1000 computations

Figure 10: GMM Clustering of the 2gaussians dataset with $k = 2$ and no k-means initialisation



(a) Clustering with k=2



(b) log-likelihoods (local maximas) ordered values for 10.000 computations

Figure 11: GMM Clustering of the 2gaussians dataset with $k = 2$ and k-means initialisation

In this task we applied k-means and GMM on 2gaussians dataset. It is obviously for a human eye, that there are 2 clusters.

1. We observed that GMM ellipses that were initialized with random points, have a bigger overlapping region, contrary to K-means initialization.

2. At first, we investigated why the k-means algorithm failed to recognized the 2 clusters we expected to observe (see Fig. 9a). We computed 10.000 loss values in order to evaluate possible local and global minima. As described in the previous exercise, k-means seems to find a global minima with $loss = 142.22$ (see Fig.7b) which corresponds to the "incorrect" clustering which is shown in Figure 7a. At first, we thought the k-means algorithm could be missing the "real" global minima corresponding to the ideal clustering. In order to check this, we computed the k-means loss value for the ideal separation case. When using this clustering and the correspondent cluster centers (see Fig. 9a), we obtained $loss = 229.092$. This result shows that since, $142.22 < 229.092$, the k-means algorithm is mathematically computing the correct global minima corresponding to the loss function we are minimizing. The cause of the problem seems to be k-means itself. We know that since k-means uses the euclidean distance, the algorithm strongly tends to form spherical clusters. We can observe this in Figure 7a. The two clusters k-means finds have a much more spherical form then the two ideal clusters (see Fig.9a). We believe this problem mostly occurs when two or more non spherical clusters lie close to each other. We showed that if we separate both clusters enough, k-means recognizes both correctely again (see Fig.9a).

3. Secondly, we followed the same approach to determine if the GMM clustering with k=2 was cappable of finding the clusters we expected. In this case, we expected a positive outcome for the following reason. The Gaussian Mixture model models optimizes in the case $k = 2$ two gaussians with two different means and two different covariance matrices in order to maximize the likelihood of all the data set. Consequently, this allows the algorithm to model ellipsoids and does not tend to always form spherical patterns as k-Means does. After performing the GMM algorithm 10.000 times with its correspondent log-likelihood values (see Fig.10b) we see that the global maxima is $log - likelihood_{max} = 7.26$. We can also observe that finding the global maxima is unlikely to find. From Figure 10b we can see that there are numerous local maximas. As in the previous exercise, we can also conclude that the GMM results heavily depends on the initialisation of the algorithm.

4. Till this point we know that the K-means algorithm failed and the GMM algorithm succeeded in finding the correct clusters. Last but not least, we examined how initializing GMM with K-means affected the solution. We followed the same previous approach and computed the GMM log-likelihood 10.000 times (initializing every time with K-means). Figure 11b shows that in this case GMM is not able to find the global maxima as before

and gets stucked in the local maxima at ≈ 0.65. Consequently, the clustering initializing with K-means (see Fig.11a) is incorrect. We believe that the non optimal K-means clustering takes the GMM to the wrong path when optimizing and thus leads to a local maxima.

Assignment 9 Analyse the USPS dataset

USPS is a handwritten digits dataset, with 2007 points of shape 256. Examples of data points are plotted in Figure 12.

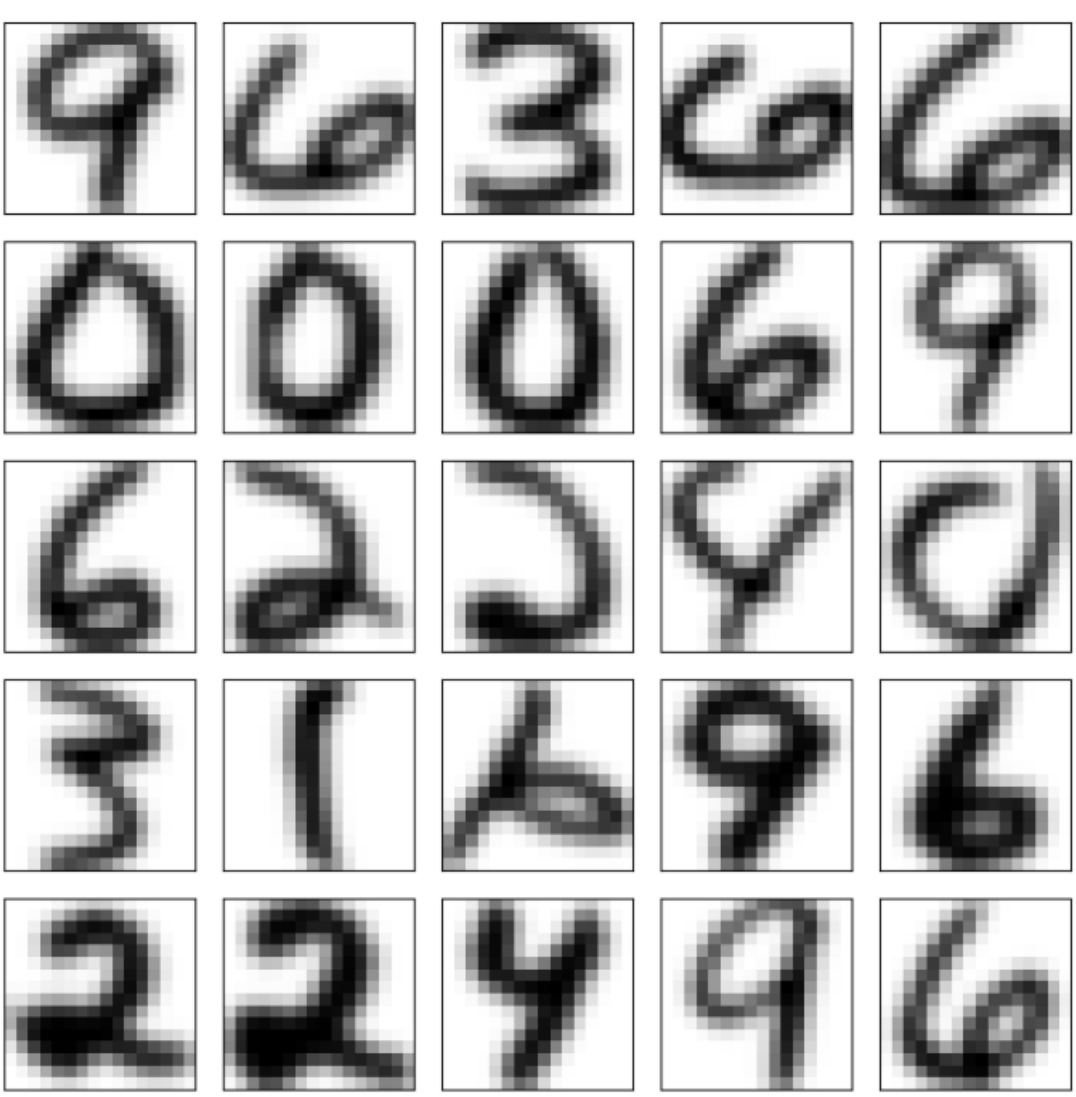


Figure 12: Points from dataset

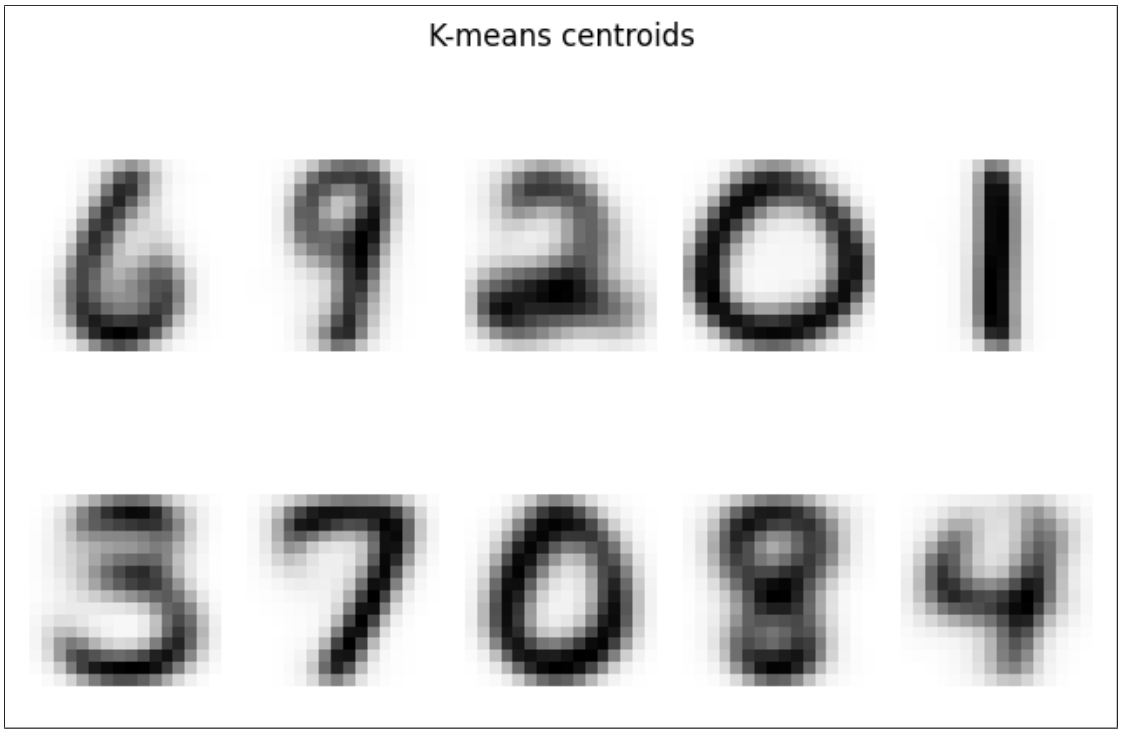


Figure 13: Centroids from K-means with k=10

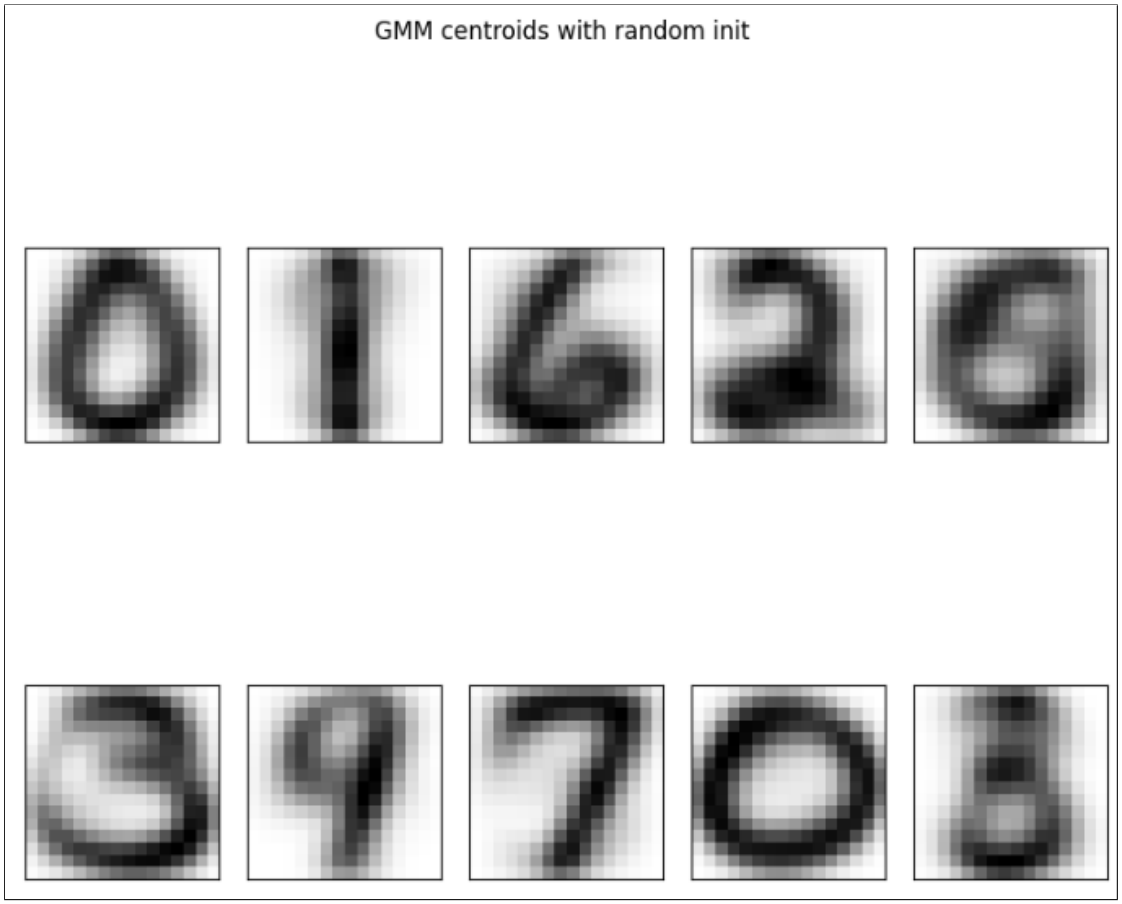


Figure 14: Centroids from GMM with k=10 and random initialization

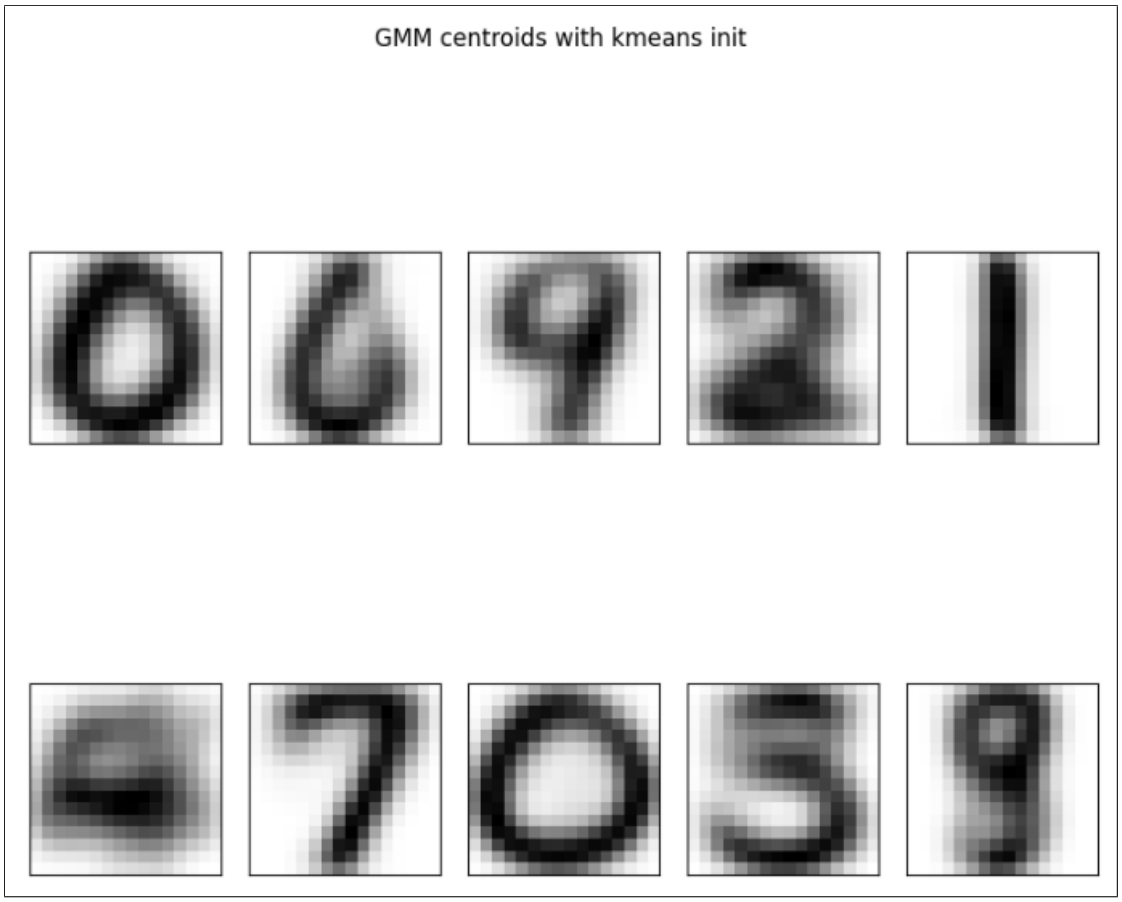


Figure 15: Centroids from GMM with k=10 and k-means initialization

Figure 13 contains the centroids found by K-means algorithm. In Figure 14 and 15 we plotted the centroids found by GMM with random initialization and with K-means initialization. We can see the both algorithms failed to find correctly all centroids, for example, centroids that correspond to number 0, are found twice.
Although both algorithms did not find the completely correct result, the K-means result is more pronounced and clearer.

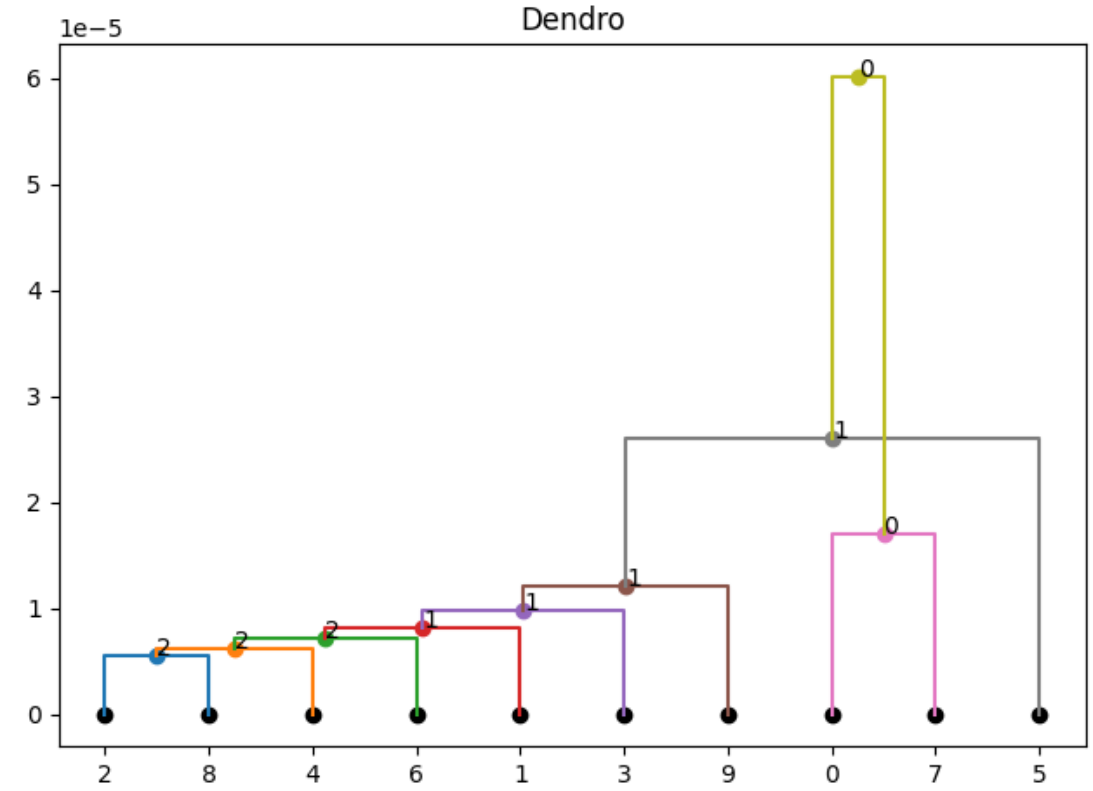


Figure 16: Dendrogramm

In Figure 16 we plotted the resulted dendrogramm of agglomerative k-means and in Figure 17 the merged centroids at every agglomerative step.

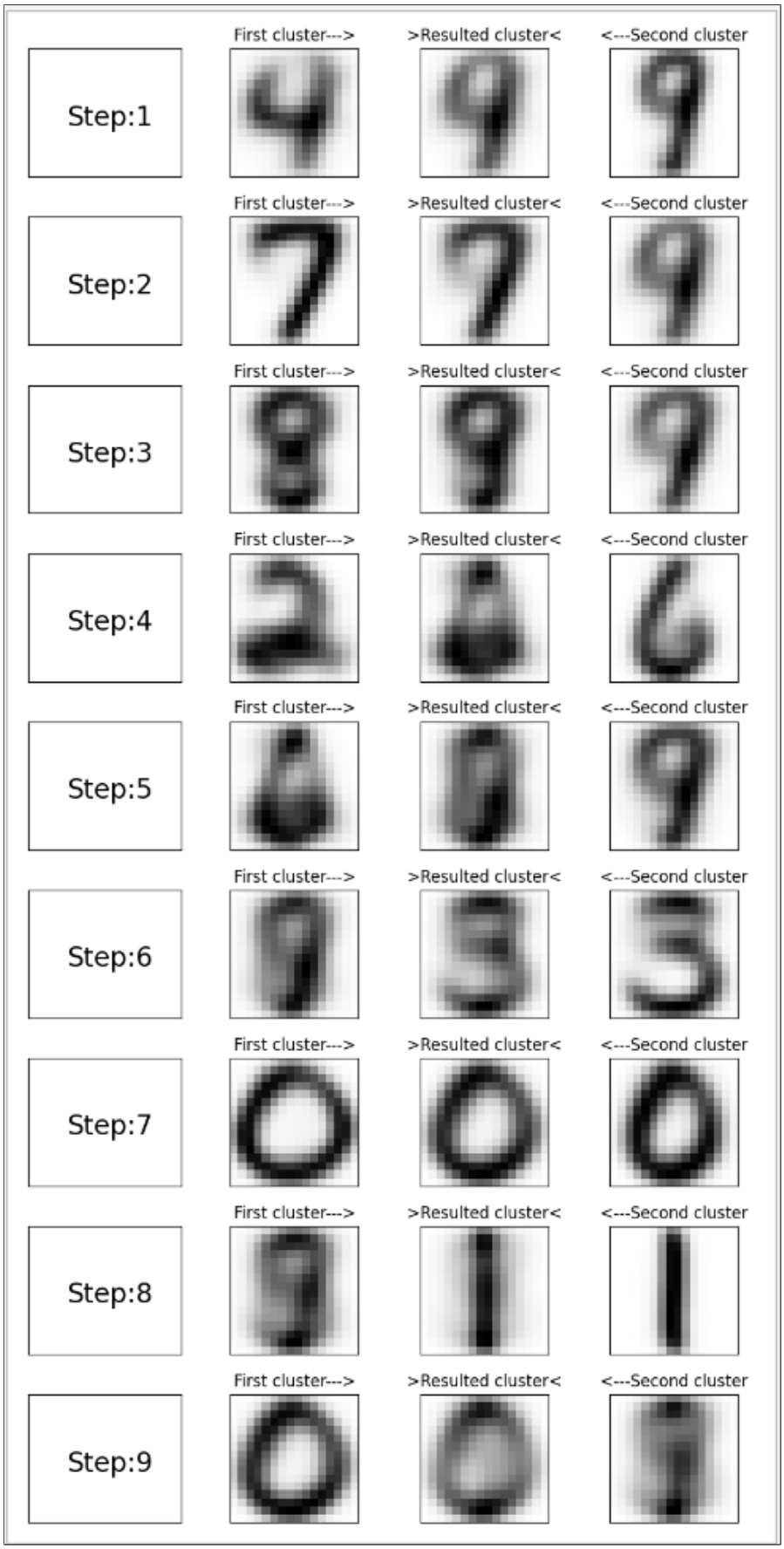
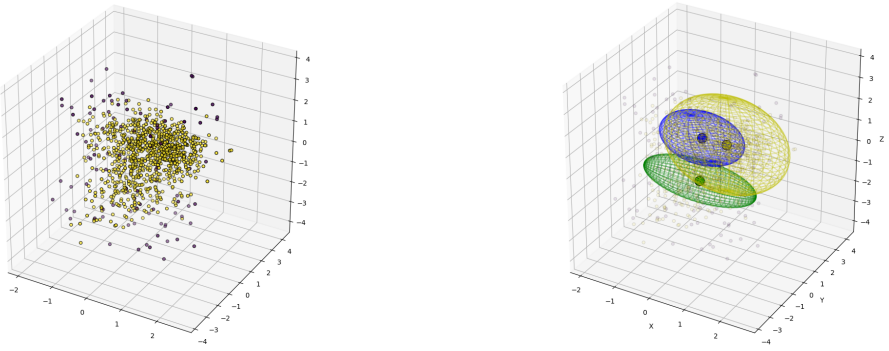


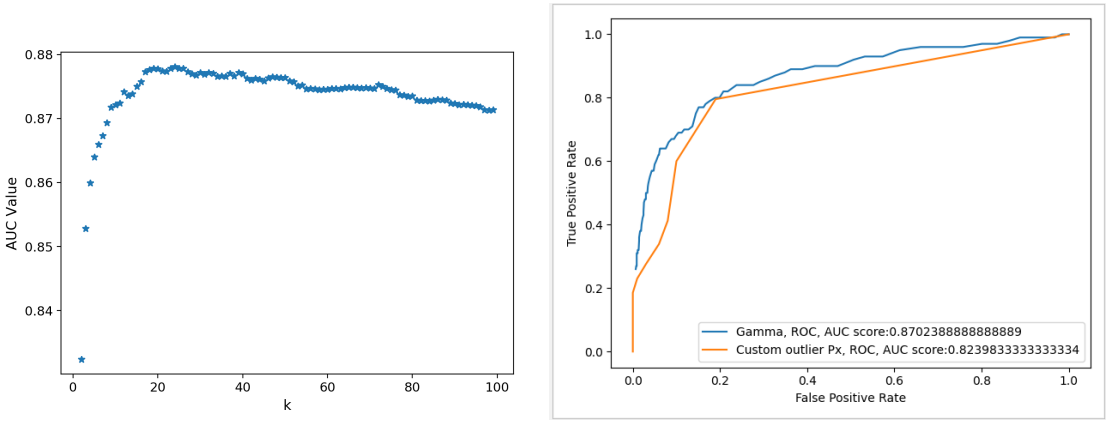
Figure 17: Cluster centroids at every agglomerative step.

Assignment 10 Analyse mixture of three Gaussians



(a) Original dataset (b) GMM means and covariances

Figure 18: LAB Data set



(a) γ index AUC values for k=2..100 (b) GammaIdx vs custom outlier detection Px

Figure 19

On this task, we analyse the mixture of 3 gaussians. The original dataset is plotted in Figure 18.a) . We plotted the resulted GMM means and covariances as ellipsoids in Figure 18.b). After computing the GMM several times in order to avoid bad local maximas, the obtained best logLike for GMM is 9.61.
In order to detect outliers, We used the $gammaIdx$ and auc functions from previous assignments sheet, and got the highest $auc = 0.87$ for k=24, see Figure 19.a).
Furtheremore, we explored how using the best GMM model worked for outlier detection. For this purpose, we used the formula: $P(x) = \sum_{k=1}^{3} \pi_k * N(x|\mu_k, \sigma_k)$, and obtained the $auc = 0.82$ value(see Figure 19.b)
The custom tailored outlier detection does not outperform the gammaidx method but also provides an optimal performance in outlier detection.