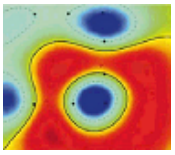
Technische Universität Berlin



Problem Set 2: Clustering and Expectation-Maximization

Report Machine Learning Lab Course
Fachgebiet Maschinelles Lernen
Prof. Dr. Klaus-Robert Müller
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

submitted by
Budi Yanto

Instructor: Daniel Bartz
Felix Brockherde

Matrikelnummer: 308819
Email: budiyanto@mailbox.tu-berlin.de

Contents

List of Figures

Chapter 1

Implementation

This chapter describes the implementation part of the second problem set. There are five assignments in this part: 1) implement *k-means* clustering, 2) implement stepwise optimal *hierarchical agglomerative* clustering, 3) implement a function which given a hierarchical clustering sets up a *dendogram* plot, 4) implement the *EM* algorithm for *Gaussian Mixture Models (GMM)* and 5) implement a function that visualizes the *GMM* for two-dimensional data.

1.1 Assignment 1: K-Means Clustering

In this assignment the *k-means* clustering algorithm should be implemented as follows:

```
mu, r = kmeans(X, k, max_iter=100)
```

The algorithm terminates when the membership and the cluster centers no longer change or after *max_iter* (default value = 100) iteration, depending on which comes first. The implemented function was tested on the test data and passed the test.

1.2 Assignment 2: Hierarchichal Agglomerative Clustering

The task in this assignment is to implement stepwise optimal *hierarchichal agglomerative* clustering with *k-means* criterion as a function. The implemented function was tested on the test data and passed the test.

```
R, kmloss, mergeidx = kmeans_agglo(X, r)
```

1.3 Assignment 3: Dendrogram Plot

The third assignment in the implementation part is to implement a function which given a hierarchical clustering sets up a *dendogram* plot:

```
agglo_dendro(kmloss, mergeidx)
```

The parameters *kmloss* and *mergeidx* correspond to the results of *kmeans_agglo*. The function *scipy.cluster.hierarchy.dendogram* is used to draw the *dendogram* plot.

1.4 Assignment 4: Expectation-Maximization

In this assignment the *EM* algorithm for *Gaussian Mixture Model (GMM)* should be implemented as a function:

```
pi, mu, sigma = em_gmm(X, k, max_iter=100,
                       init_kmeans=False)
```

The parameter *init_kmeans* determines the initialization method. If it is true, then *k-means* is used for the initialization. For random initialization, *k* data points are selected as cluster centers, the prior *pi* of each cluster is set to $1/k$. The sigma of each cluster is the identity matrix. On the other hand, the cluster centers of *k-means* are used for *k-means* initialization. The prior *pi* of each cluster is set to total data points in each cluster divided by total number of data points in the data set. The sigma of each cluster is the covariance matrix of the data points of each cluster.

The algorithm terminates when the maximal number of iterations *max_iter* has been reached or the log likelihood no longer changes (< 0.001)

1.5 Assignment 5: EM Visualization

In the last assignment a function to visualize the *GMM* should be implemented. The figure should show the data as a scatter plot, the mean vectors as red crosses and the covariance matrix as ellipses (centered at the mean).

Chapter 2

Application

In this chapter, the implementations should be applied to three datasets: *2gaussians* dataset, *5gaussians* and *USPS* dataset.

2.1 Assignment 6: *5gaussians* Analysis

In this assignment, the *5gaussians* data set is analyzed. The data set should be clustered using *k-means* and *GMM* for $k = 2, ..., 10$. Figure 2.1 shows the original *5gaussians* data set. Since the data set is constructed with five gaussians, $k = 5$ is the best choice for the number of clusters.

Question 6.1: Do both methods find the five clusters reliably?

K-means finds the five clusters reliably. Since *k-means* can have different result depending on the initialization, it was run 100 times. The loss value from each run was calculated, as depicted in Figure 2.2. The clustering which gives the lowest loss value was then picked and visualized in Figure 2.3.

GMM with random initialization also finds the five clusters reliably. The same method as *k-means* was applied to *GMM*. It was also run 100 times, but instead of loss value, the log likelihood of each run was calculated, as shown in Figure 2.4. The clustering which gives the highest log likelihood was then picked and visualized in Figure 2.5.

Finally, *GMM* with *k-means* initialization was utilized. It also finds the five clusters reliably. It was also run 100 times and the log likelihood of each run was calculated. Figure 2.6 and Figure 2.7 depict the log likelihood of each run and the clustering which gives the highest log likelihood, respectively.

As we have seen above, the three methods can find the five clusters reliably, because the *5gaussians* data set is a well-separated data set.

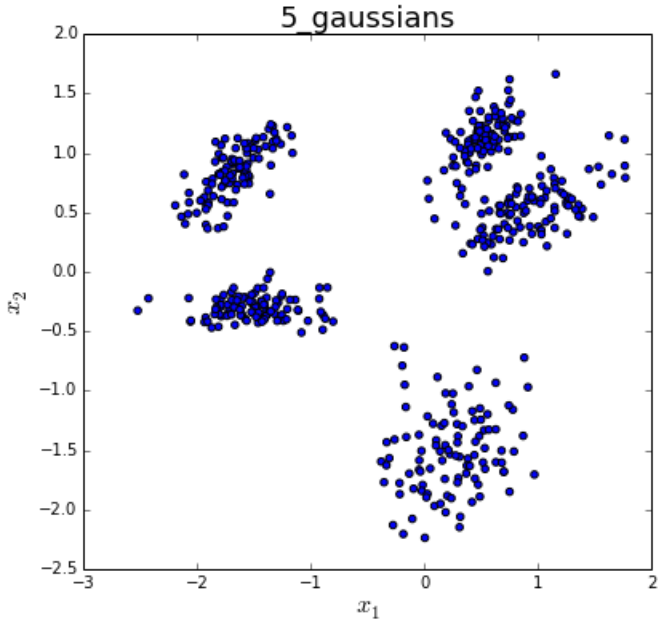


Figure 2.1: *5gaussians* data set

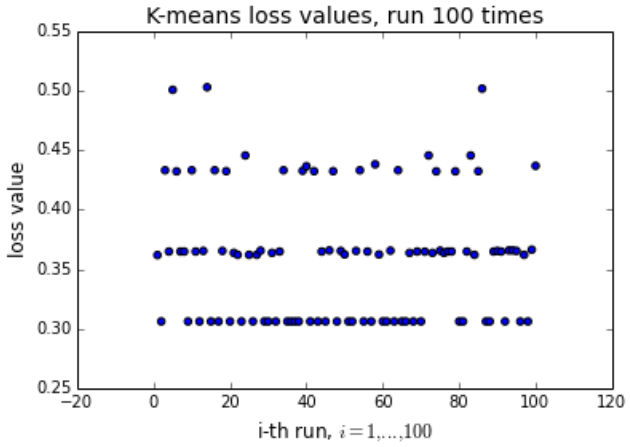


Figure 2.2: Loss values of k-means, run 100 times

Question 6.2: What role does the initialization of the GMM with a k-means solution play in the number of necessary iterations and the quality of the solution?

The initialization of the *GMM* with a *k-means* solution plays a significant role in the number of necessary iterations until the convergence is reached. *GMM* depends strongly on the initialization. With *k-means* initialization, the *GMM* converged more

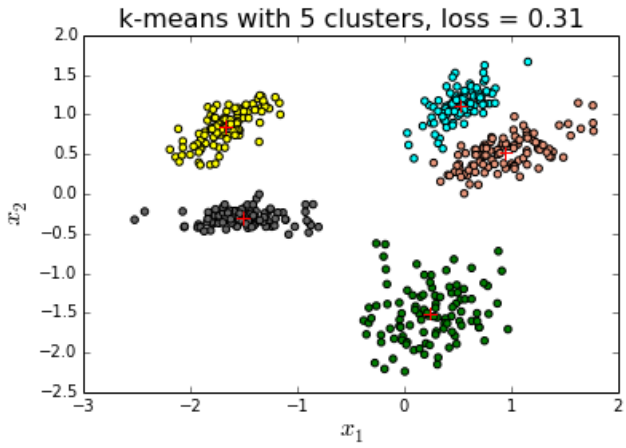


Figure 2.3: k-means clustering with $k = 5$, applied to *5gaussians* data set

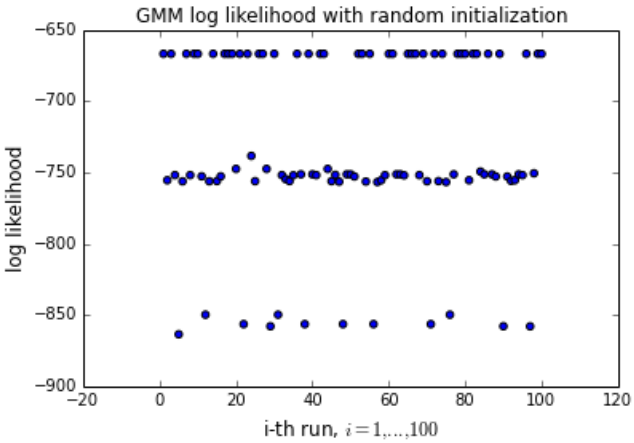


Figure 2.4: Log likelihood of GMM with random initialization, run 100 times

quickly than random initialization.

GMM was run 100 times with random and k-means initialization to avoid some bad local optima. As we can see from Figure 2.8, *GMM* with random initialization never converged with the number of iterations below 20, whereas *GMM* with k-means initialization converged mostly only with about eight iterations.

On the other hand, the initialization does not play a significant role in the quality of the solution. Both the random and k-means initialization produce almost the same log likelihoods. Figure shows the maximum log likelihoods of both initialization which were run 100 times. Both methods produce almost the same maximum log likelihood, about -666.7011.

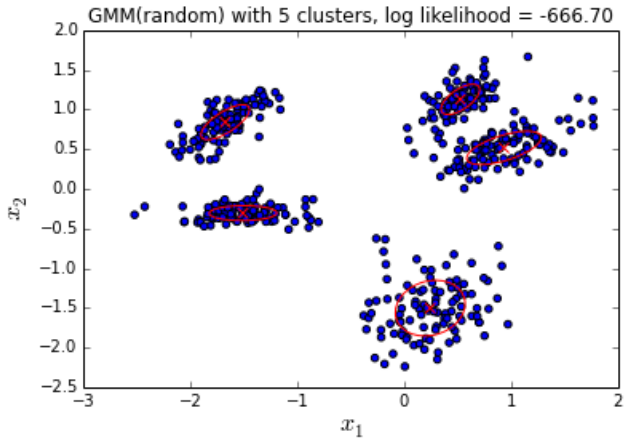


Figure 2.5: GMM clustering with random initialization, $k = 5$, applied to *5gaussians* data set

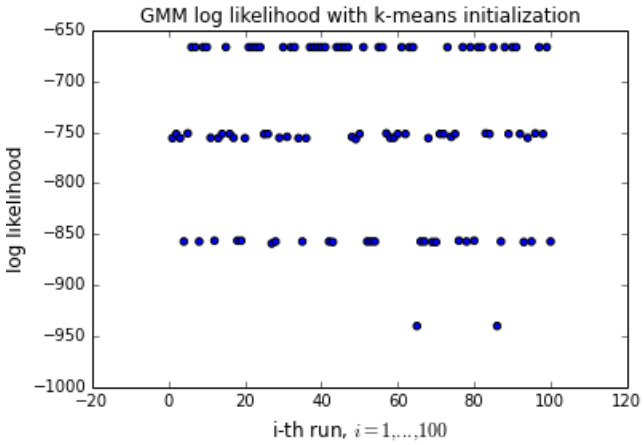


Figure 2.6: Log likelihood of GMM with k-means initialization, run 100 times

Question 6.3: What does the dendogramm of the hierarchical clustering look like and is it possible to pick a suitable value of *k* from the dendogramm?

It is possible to pick a suitable value of k from the dendogramm plot. Figure 2.10 shows the dendogramm plot of *kmeans_agglo* with initial clustering $k = 50$. From the plot, we can see that the clusters can actually be divided into five clusters (marked with yellow ellipses). Before the merge of those five clusters, there is no significant increase of the loss values. But after merging those five clusters, the loss values increase significantly.

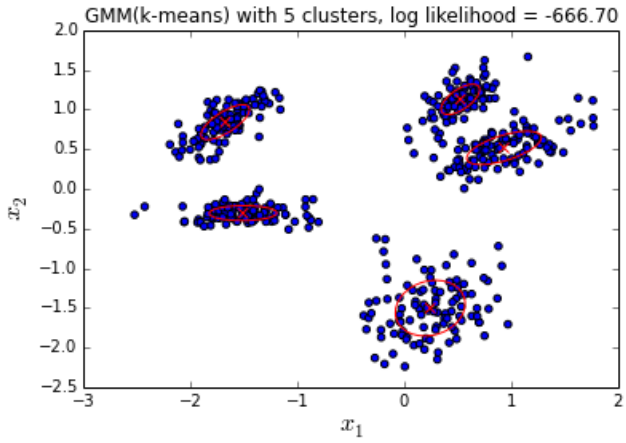


Figure 2.7: GMM clustering with k-means initialization, $k = 5$, applied to *5gaussians* data set

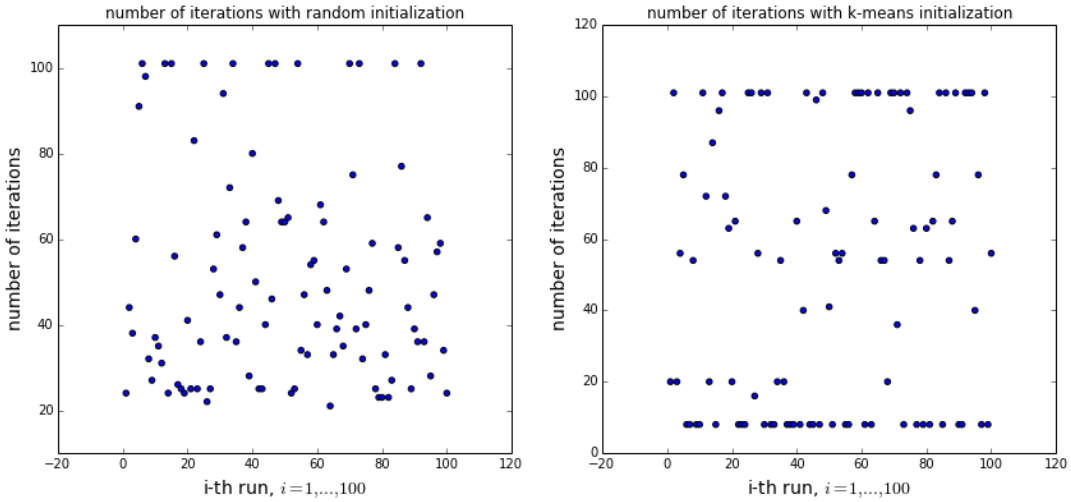


Figure 2.8: Number of iterations of *GMM* ($k = 5$) with random and k-means initialization, run 100 times

2.2 Assignment 7: *2gaussian* Analysis

In this assignment the *2gaussians* data set should be analyzed with *k-means* and *GMM*. Figure 2.11 show the original *2gaussians* data set. Since the data set is constructed by two gaussians, $k = 2$ is the best choice for the number of clusters.

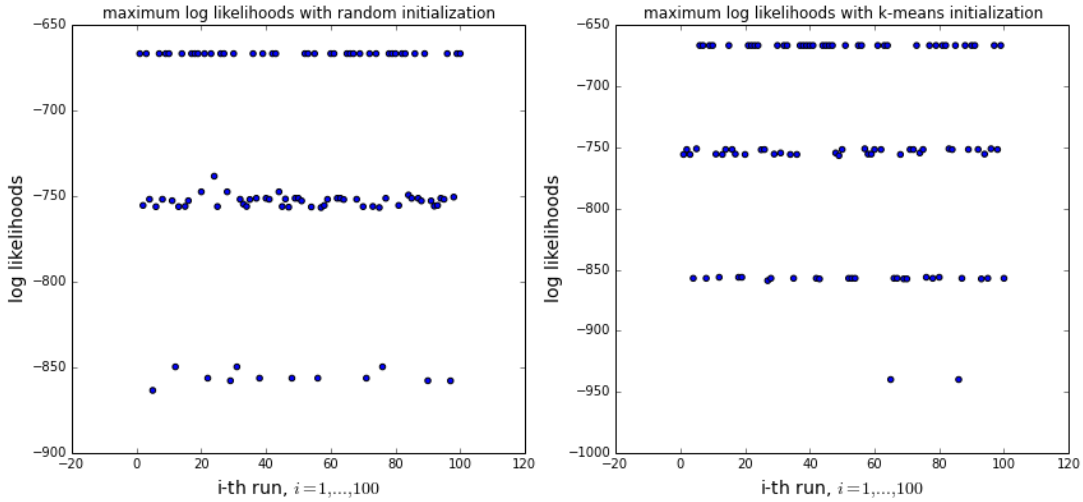


Figure 2.9: Maximum log likelihoods of *GMM* ($k = 5$) with random and k-means initialization, run 100 times

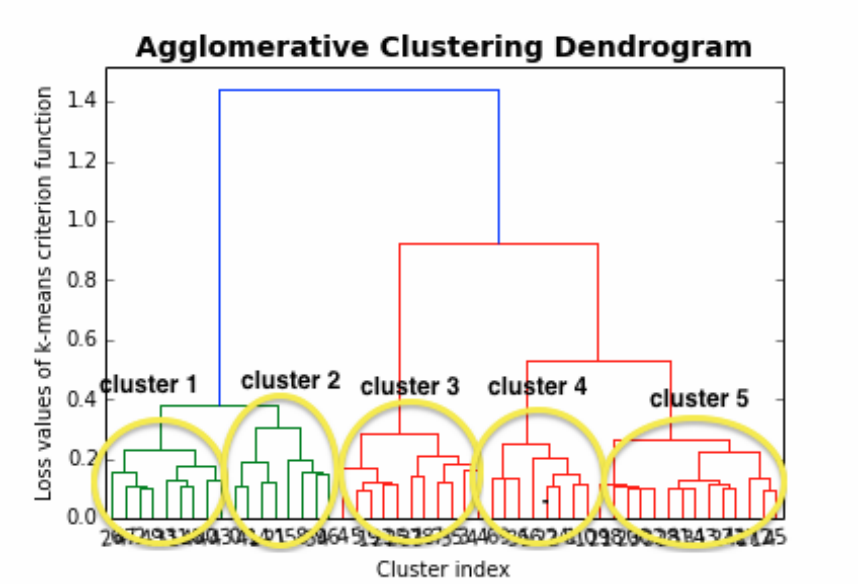


Figure 2.10: Dendogramm plot of the hierarchical clustering with initial clustering $k = 50$

Question 7.1: Compare the cluster centers. Which algorithm works better and why?

K-means, *GMM* with random and k-means initialization were used to cluster the

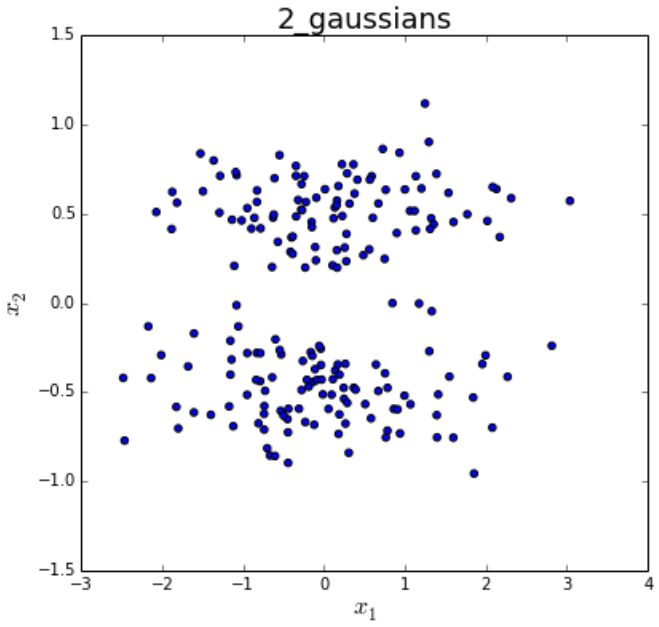


Figure 2.11: *2gaussians* data set

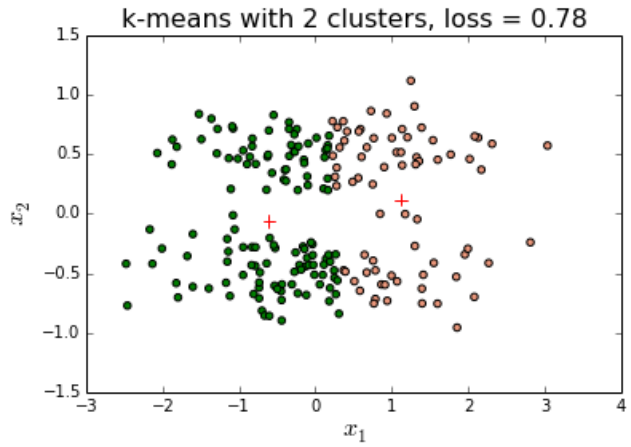


Figure 2.12: k-means clustering with $k = 2$, applied to *2gaussians* data set

2gaussians data set. Each method was run 100 times to avoid bad local optima. For *k-means* method, the clustering which gives the lowest loss value was picked, whereas for *GMM*, the clustering which gives the highest log likelihood was picked. Figure 2.12 shows the clustering using *k-means*, whereas Figure 2.13 and Figure 2.14 show the clustering using *GMM* with random and k-means initialization.

From the plots, it can be obtained that the *GMM* with random initialization is the best method to cluster the *2gaussians* data set. The *k-means* method did not perform very

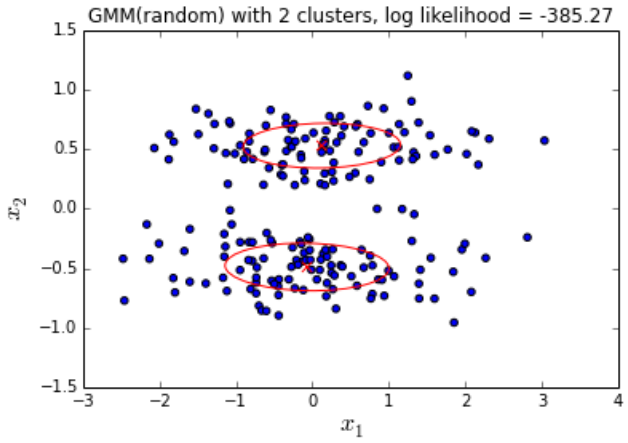


Figure 2.13: GMM clustering with random initialization, $k = 2$, applied to *2gaussians* data set

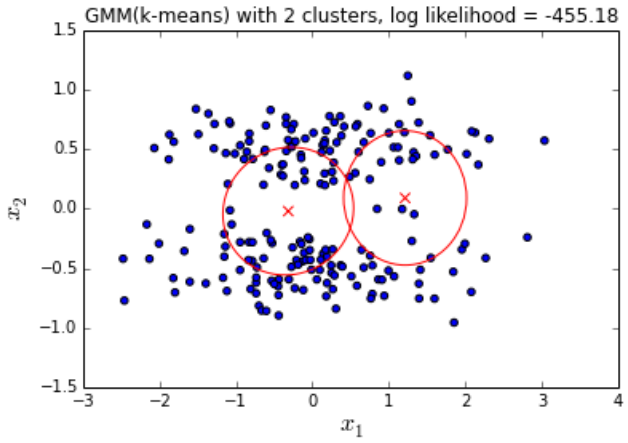


Figure 2.14: GMM clustering with k-means initialization, $k = 2$, applied to *2gaussians* data set

well, because the objective of *k-means* is to minimize the loss value, which basically depends strongly on the euclidean distances of each data point. But the data set was created using mixture of two gaussians which is separated as upper- and lower part. Therefore two data points can have a small euclidean distance although they are from two different gaussians.

The *GMM* with random initialization performed very well, because its objective is to maximize the log likelihood, which can be calculated using the gaussian function. Instead of depending on the mean of data points, it uses maximization to find the best clustering. On the other hand, the *GMM* with k-means initialization did not perform

very well, because *GMM* depends strongly on the initialization. Since it used the result of k-means to do the initialization, it performed almost in the same way as k-means.

Question 7.2: How does the GMM depends on the initialization

The *GMM* depends strongly on the initialization, as we can see in the Figure 2.13 and Figure 2.14. Although the *GMM* was applied to the same data set, but its performance was different, depending on the initialization.

2.3 Assignment 8: *USPS* Analysis

In the last assignment of this problem set, the *k-means* and *GMM* algorithm should be applied to the *USPS* data set with $k = 10$.

Question 8.1: Compare the cluster centers. Which algorithm delivers better results?

The *k-means* performed quite well on the *USPS* data set. The cluster centers that repersented as 16 x 16 images show that almost all digits were recognized, only the digit 5 was missing. Figure 2.15 shows the cluster centers of usps data set with k-means.

The *GMM* performed worse than *k-means*, as shown in Figure 2.16. Maybe because of the regularization of the covariance matrix, i did not find a good value to use for the regularization.

Figure 2.15: Cluster centers of usps data set with k-means ($k = 10$)

Figure 2.16: Cluster centers of usps data set with GMM with random initialization ($k = 10$)

Question 8.2: Set up a dendogramm to the hierarchical clustering solution and also a plot which displays the cluster centroids as a 16 x 16 image at every agglomerative step

Figure 2.17 depicts the Dendogramm to the hierarchical clustering solution, whereas Figure 2.18 depicts the cluster centroids as a 16 x 16 image at every agglomerative step.

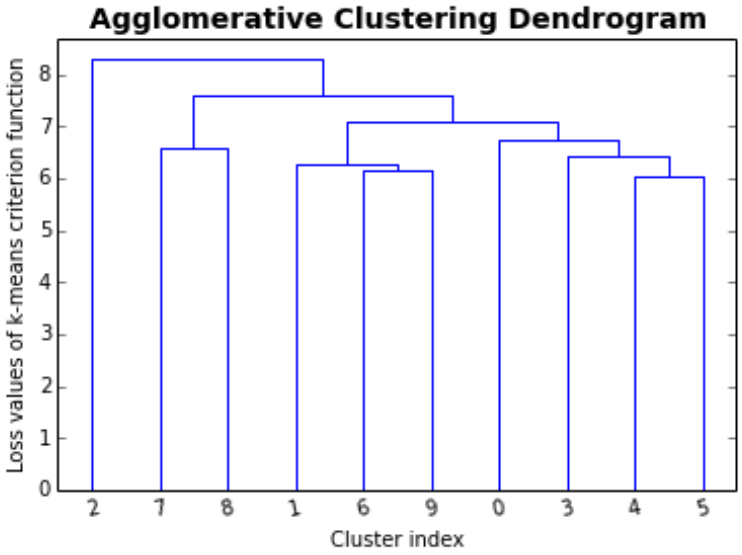


Figure 2.17: Dendogramm to the hierarchical clustering solution

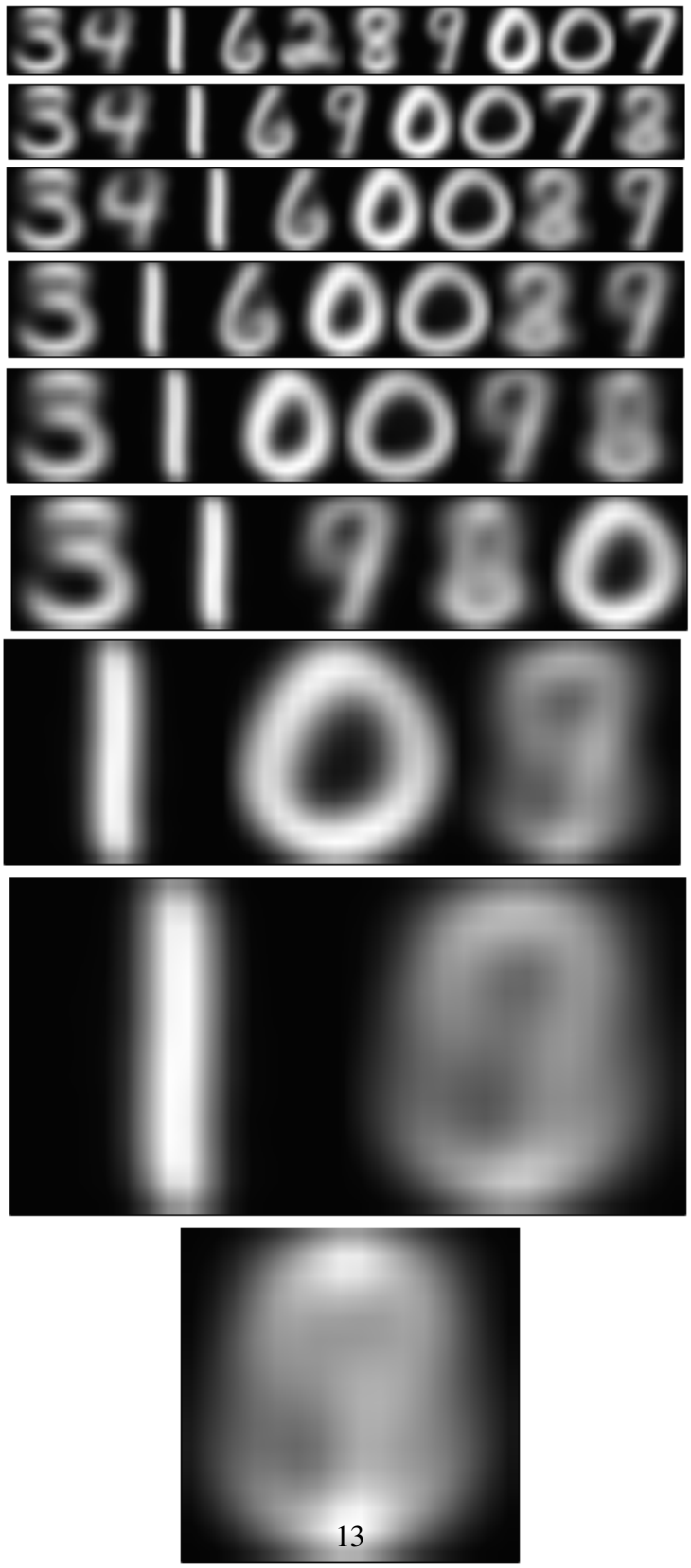


Figure 2.18: Cluster centers of usps data at every agglomerative step