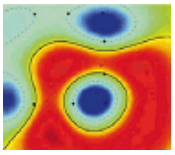
Technische Universität Berlin



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.

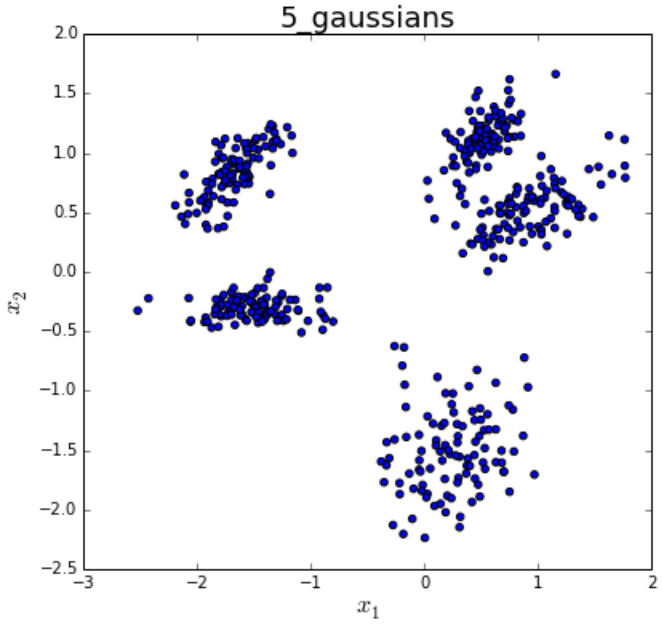


Figure 2.1: *5gaussians* data set

Question 6.1: Do both methods find the 5 clusters reliably?
K-means finds the 5 clusters reliably. Since *k-means* can have different result depending on the initialization, it is run 100 times. The loss value from each run is calculated, as depicted in Figure 2.2. The clustering which gives the lowest loss value is then picked and visualized in Figure 2.3.

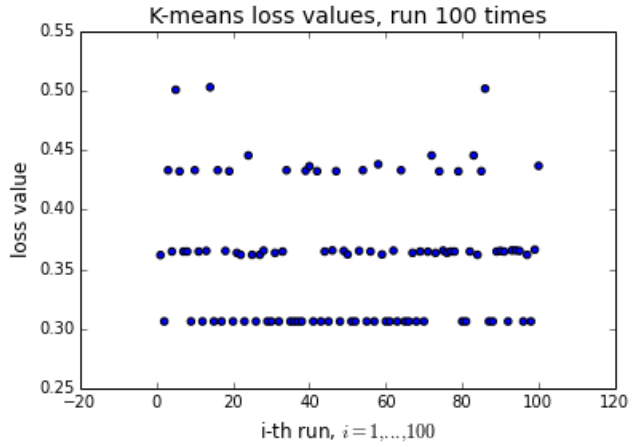


Figure 2.2: Loss values of k-means, run 100 times

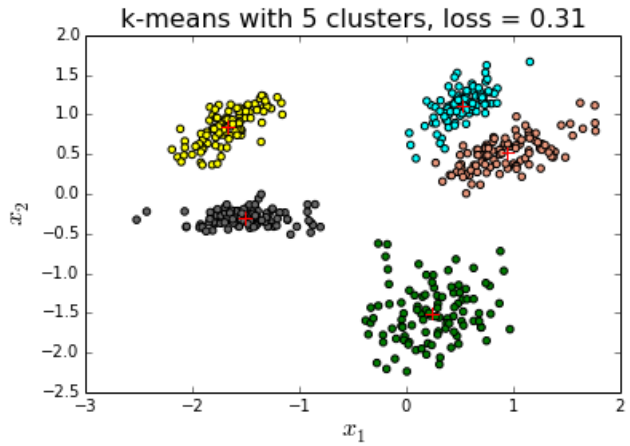


Figure 2.3: k-means clustering with $k = 5$, applied to *5gaussians* data set

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

In this case the difference in iterations seems insignificant. However, as will be seen in the presence of poorly separated data a good choice for our initial guesses can drastically reduce the number of iterations required to obtain convergence
This assignment asks us to apply *PCA* to the *usps* data set and visualizing the results. The *usps* data set consists of 2007 images with the dimension of *16 x 16*. The images are hand-written digits of zero to nine, which can be viewed as classes. Firstly, i separate the data set according to each digit into ten classes and then applied *PCA* to each class. The *PCA* was applied to the original data set and noisy data set.

2.2 Assignment 7: *2gaussian* Analysis

In this assignment, the γ-index method is utilized to detect outliers and applied it to the *banana* data set. The positive class of the data set is used as *inliers*, to which the negative class is added as outliers. The γ-index is then used to detect outliers with contamination rates of 1%, 5%, 10% and 25% relative to the positive class. Figure **??** shows the complete original data set, both positive and negative class, whereas Figure **??** shows the contaminated data set.
There are three methods that should be used to detect the outliers: (a) the γ-index with $k = 3$, (b) the γ-index with $k = 10$ and (c) the distance to the mean for each data point. All of the methods are then applied to the four contamination rates mentioned above. After that, the *AUC* (area under the *ROC*) should be calculated. Figure **??** shows the boxplots that visualize the distribution of the *AUC* values.
The boxplots show that the method using the distance to the mean for each data point performed quite bad, while both of the γ-index methods performed very well, especially for the data set with lower contamination rates. The γ-index with $k = 10$ performed slightly better than the γ-index with $k = 3$.

2.3 Assignment 8: *USPS* Analysis

In the last assignment of this problem set, *LLE* has to be applied to noisy *flatroll* data set. Two gaussian noise were added to the data set, with variance 0.2 and 1.8, respectively. After that, both noisy data sets should be unrolled using *knn* with a good value of *k* and a value which is obviously too large. Figure **??** depicts the two noisy images and their resulting embedding. It can be obtained that the noisy data set with big variance(1.8) is not very good unrolled.