**Data and Execution procedure**

The dataset considered for this experiment is taken from a file ‘Letter2Class.data’, which is present in the same folder where all other .m files are located and this data was accessed through these Matlab codes. In this data file the row header represents labels. The data consists of 16 types of attributes which represents a character ‘A’ or ‘X’. All the Matlab codes along with the data file are stored in a single folder. For all the questions same dataset is being used.

Using the ‘importdata’ function the data is being read from the file. For all the 16 types of attributes representing ‘A’ or ‘X’ a total of 1576 instances are collected of which 789 instances belong to ‘A’ and remaining 787 instances belong to ‘B’.

**Part 1**

**Implementing K-Means++ and fuzzy C-Means clustering methods using**

**MATLAB.**

**K-Means++**

K-means++ is an algorithm that is used to choose the initial values for the k-means clustering method. It improves the performance of k-means, avoiding poor clusterings due to bad initial values. The user selects the amount of clusters (k) to be computed.

The implemented algorithm consists on:

1. Selecting a random point from the data to use as first centroid
2. For each data point, the distance from the nearest, previously chosen centroid is computed
3. The next centroid is chosen such that the probability of choosing it is proportional to the distance calculated in step 2
4. Steps 2 and 3 are repeated until all k centroids have been selected

Then, conventional k-means is applied. For k-means, each cluster centroid is recalculated as the average of all the data points that belong to that cluster (a data point belongs to the cluster of its nearest centroid).

Both k-means++ and k-means algorithms were implemented in MATLAB in the kmeanspp function in ‘kmeanspp.m’ . The function takes the dataset and the number of desired clusters as input and returns the labels of the data and (optionally) the clusters centroids.

This is a hard clustering algorithm, meaning that all the data points belong to one and only one cluster.

However, both k-means and k-means++ are non-deterministic since the centroids are chosen either randomly or using a distance based probability.

MATLAB itself has an implementation of k-means from its Statistics and Machine Learning Toolbox, so that was used as cross validation.

To assess the stochastic behavior of the implemented algorithm, in ‘kmeanspptest.m’, several trials can be evaluated, the results can be seen in Table 1, where the average and standard deviation are calculated and the MATLAB centroids are also shown. It can be seen that they do not differ much from the implementation.



Table 1: Statistical behavior of the centroids in the k-means++ implementation

**Fuzzy C-Means**

Fuzzy C-means is a soft algorithm, meaning that the points do not belong to a given cluster, but rather have a membership degree to all of them. This has an advantage over hard clustering (e.g. k-means) especially for the points that are close to more than one centroid in real-life scenarios where a point actually belongs to more than one cluster with some kind of degree.

The membership function, which is the degree in which a point belongs to a given cluster is inversely proportional to the distance to the cluster’s centroid, the formula to calculate it is:

Where is the membership of point to cluster j, is the centroid of cluster j, and is the fuzziness level which in the implementation is fixed to 3. can be any real number greater than 1 (the greater, the softer the algorithm is).

Then to calculate the new centroids the following formula is used:

The formula represents the centroid of all the data points weighted by the membership to that cluster. So every data point contributes to the centroid but its proportion will be weighted.

The algorithm for fuzzy c-means that was implemented in this assignment is as follows:

1. Select k points within the variables’ ranges
2. The membership function is calculated for each point
3. The new clusters or fuzzy centers are calculated
4. Steps 2 and 3 are repeated until all k centroids have converged

The function fuzzycmeans can be found it fuzzycmeans.m, it takes the data and number of clusters as input and returns the labels and (optionally) the clusters’ centroids.

As well as with the k-means++ implementation, a test script was implemented, called ‘fuzzycmeanstest.m’ and the statistical behavior was studied in a similar manner to k-means++ and MATLAB’s k-means was also used as cross validation



Table 2: Statistical behavior of the centroids in the fuzzy c-means implementation

In Table 2 it can be seen that even that the fuzzy algorithm is non-deterministic, the centroids were all equal in all the variables in all trials. From this observation, it can be said that fuzzy c-means is more robust, in the sense that is less dependent on the random initialization of the clusters.

**Part 2**

**Implementing Expectation Maximization method for clustering using MATLAB.**

Expectation Maximization algorithm is also a soft method for clustering. It is used for finding maximum likelihood solutions for probabilistic models having latent variables. In this implementation, the used algorithm was based in Bishop’s book, Pattern recognition and machine learning (Springer 2006). In particular, mixed Gaussian PDFs were used.

Any implementation of EM has two distinct steps, the E step (Expectation) and the M step (Maximization). In the E step, the current values of the parameters are used to evaluate the posterior probabilities or *responsibilities*. Then these are used in the M step, to re-estimate the means, covariances, and mixing coefficients of the Gaussian distributions.

Responsibility is defined as:

Where , , and are the weight, mean and covariance of each Gaussian.

So the implemented algorithm goes as follows:

1. Randomly select the PDF’s parameters ( means within the variables’ ranges, random weights that are positive and add to 1, and a random number between 0 and 1 multiplied by the covariance matrix o all the data points)
2. **E step** – evaluate the responsibilities using the current parameters
3. **M step** – evaluate the new parameters using the current responsibilities
4. Repeat 2 and 3 until convergence

The algorithm can use the log likelihood to test for convergence or just use the parameters, in the implementation the log likelihood is used.

The implementation of this algorithm needed to use some heuristics to avoid singularities. Running the algorithm ‘as is’ presented scenarios where some of the Gaussian ‘collapsed’ into a single data point, that makes the covariance matrix singular and unable to be used within the script. This problem is well described in Bishop’s book and they suggest that when that situation is detected, that Gaussian’s parameters should be dismissed and new values should be used. Another observation is that the algorithm takes much longer, in both number of iterations and the time each iteration takes. And sometimes the result is not good.

The function expectationmaximization, implemented in expectationmaximization.m has the same inputs and outputs as the previous ones, data set and number of clusters; and labels and centroids respectively.

From Table 3 it can be seen that the centroids (actually the means of the gaussians) vary substantially from one trial to another, the number of iterations was also observed and it is also variable. This particular implementation of EM was not the best for this set of data. In part 3 of this report there will be a further discussion of this.



Table 3: Statistical behavior of the centroids (means) in the EM mixed gaussian

**Part 3**

**Write functions to evaluate the three methods using MATLAB.**

The test files do have some clustering results but it would be interesting to use these functions to cluster 2D data and be able to see graphically the implementation’s performance.

So a script called clusteringtest.m was implemented and a random set of two dimensional data is created, the different functions are tested and the results can be seen graphically as shown in Figures 1, 2 and 3

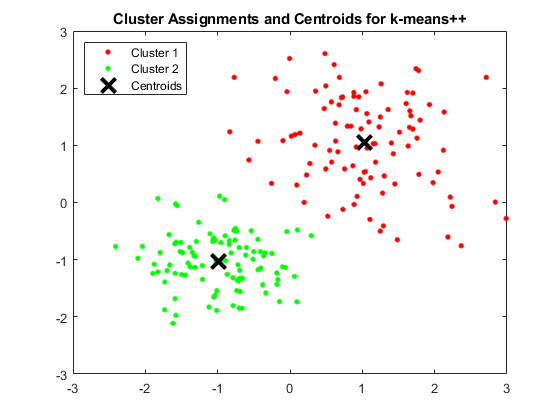


Figure 1 – Clustering algorithm performance for k-means++

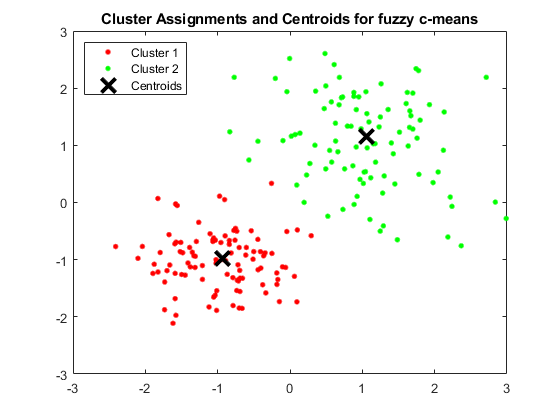


Figure 2 – Clustering algorithm performance for fuzzy c-means

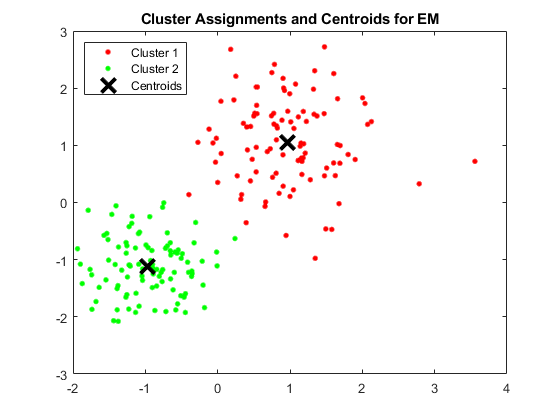


Figure 3 – Clustering algorithm performance for EM

As it can be seen graphically, all 3 algorithms work reasonably well with the randomly generated 2D data. If the script is run several times, some of the times the EM will have odd results. This may be due to some bug in the code that was not found and fixed. It can also be attributed to numerical error due to the amount of operations, especially divisions by small numbers, the resetting of the covariance matrices when a cluster collapses to a point may have unsolved repercussions on other variables. Or as Bishop himself says *“the maximization of the log likelihood function is not a well posed problem because such singularities will always be present and will occur whenever one of the Gaussian component ‘collapses’ onto a specific data point”*

So to conclude, a robust implementation of Expectation Maximization Algorithm using Mixed Gaussian functions should take all these factors into consideration.