CS5567 Spring 2016 Assignment III. (70+30) pts.

NAME(s) (use Pathway format):

E-mail IDs of TEAM MEMBER(s):

Raghunandan Rao Malangully: rm9vf Santhosh Kumar Gattu: sg6n6

Hirenbhai Harshadbhai Shah: hhstm4 Rakesh Reddy Bandi: rb8xf

Word processed electronic submission on Blackboard is due latest by 11 pm Monday April 18th. Submissions received after the deadline will be graded only for effort for a maximum of 70% of the total grade (Refer to class syllabus for detailed grading policy). State any assumptions you make, justify your answers, show intermediate steps and explain your results for maximum credit. For machine learning questions, the answer should be written in the form of a concise scientific report with a focus on rationale and evaluation. The report should be a polished depiction of what you did, why you did it, how exactly you did it, and how well it worked.

All answers should be in your own words with any sources you refer to cited at the appropriate places. Any knowledge you acquire from the Internet should be written in your own words and be appropriately referenced. Copying and pasting from the Internet, each other or any other source will not count as your effort (Refer to class syllabus for detailed policy on plagiarism).

You may submit this assignment in groups of up to 4 each. Write your names on this sheet and include it as the cover page for your submission. The submission should consist of a standalone cogent word-processed report file and additional files (code, data, instructions to run your code). Code should be submitted as a text file (not copied and pasted into Word) annotated with comments (include your names within the code files).

Compress all files into an archive before uploading on Blackboard. Be sure to name your archive file using the names of group members, e.g., HW1\_RadhaKrishnaRomeoJuliet (and not just “HW1”). If you upload the wrong file, don’t panic. You may upload upto 3 times.

You may use any programming language or package of your choice for programming assignments in the homework. Possible choices are R, MATLAB, Python (e.g., sci-kit or code on textbook website) or JAVA (e.g., Weka). Be sure to cite your source if you use pre-existing code. Clearly distinguish between pre-existing code and your contributions.

Q1. (15) Create a synthetic dataset that is a mixture of multiple Gaussians. Use the Expectation Maximization algorithm to recover the underlying Gaussians. How is the performance of the algorithm by the nature of the dataset?

Q2. (15) Create a synthetic dataset of points in 3 dimensions. Project the data onto 2 dimensions such that the original distances are preserved as closely as possible. How is the performance of the algorithm affected by the nature of the dataset?

**Q1)**

**Solution:**

**Creation of a multiple Gaussian mixture dataset:**

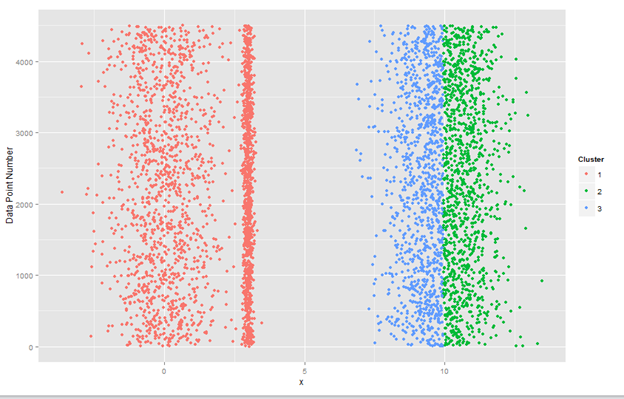
To create a dataset containing multiple Gaussian mixtures we performed the following steps in the algorithm mentioned below. We used this algorithm from the reference mentioned at the end of the report [1]. This algorithm was coded in R language and we generated the dataset with 4500 samples.

1. Generate a random variable U∼Uniform(0,1). This gives a set uniform random numbers. We pick samples from this set.
2. If U ∈ [∑ki=1 pk , ∑k+1i=1pk+1)interval, where pk correspond to the the probability of the kth component of the mixture model, then generate from the distribution of the kth component. This signifies the rule that if a sample belongs to one of the mixture model then generate distributions for that model. Eg: if U<0.3, then samples[i] = rnorm(1,0,1). If U>0.3 && U<0.8, then samples[i] = rnorm(1,10,1).
3. Repeat steps 1 and 2 until we have the desired amount of samples from the mixture distribution. This would ensure that we get samples from different normalization distributions.

**EM algorithm to find out the underlying Gaussians**:

After generating the data set now we move onto the EM algorithm for fitting the dataset into underlying Gaussians. In our case we have dataset that had 4500 samples and we assumed that there were 3 clusters in the data. The following were the steps performed to find out the underlying Gaussians.

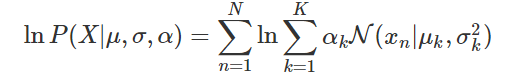
1. Initialization step where the goal is to find out the initial parameters for the model. This was done by applying the K-means algorithm on the data set with K=3 clusters. K-means also provides the parameters of Mean, Standard deviation and these are used to calculate mixing weight alpha = No.of. points in cluster k / Total no.of. points. The following figure shows our implementation of the k-means clusters.



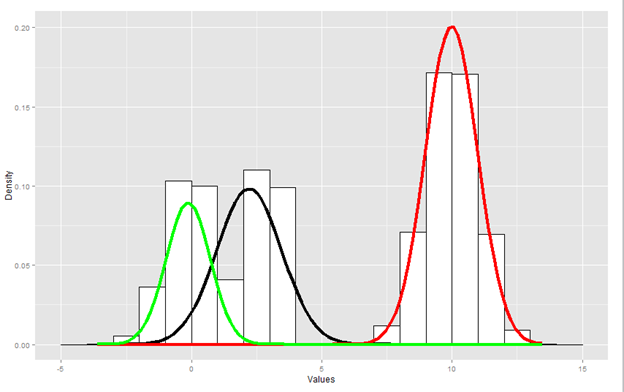
1. Then the next step is Expectation step where the goal is to find soft labels, i.e. the probability that a data point p belongs to cluster k. This challenge is handled using the Bayes’ rule which is used to calculate the posterior probability. In our case we are using the Gaussian PDF for estimating the posterior values. We use the dnorm function with the parameters of mean, standard deviation and multiply with mixing weight alpha to calculate the gaussian PDF. Eg: To estimate the probability of point x belonging to cluster 1 we do the following steps. Assuming we have 3 clusters and μk is the mean of cluster k, 𝜎k is standard deviation of cluster k, and 𝛼k is the mixing weight for cluster k.
   1. Cluster1 part, C1 = dnorm(x, μ1, 𝜎1) \* 𝛼1
   2. Cluster2 part, C2 = dnorm(x, μ2, 𝜎2) \* 𝛼2
   3. Cluster3 part, C3 = dnorm(x, μ3, 𝜎3) \* 𝛼3
   4. Normalizer = C1 + C2 + C3
   5. Probability of point ‘x’ belonging to cluster k, P(k) = Ck / Normalizer

This way we arrive at the posteriors for each point in the sample. We calculate the posterior for each point with respect to each cluster.

1. We follow the Expectation step with the Maximization step where the goal is to maximize the initial parameters which are Mean, standard deviation and mixing weights alpha of each cluster to ensure that we have the best values possible. This is achieved by replacing the value of Nk(The no.of. points present in cluster k) with the posterior probability of point xk in the equations for calculating Mean, Standard deviation and alpha. Which states that instead of using the value of point Xi we use the posterior value of Xi in calculation of the parameters.
2. The final step is the convergence checking step where the goal is to check if the algorithm has reached the point of convergence or not. For this we have implemented the technique where we calculate the log likelihood value for the point x being generated by the estimation parameters(μ, 𝜎, 𝛼) obtained from every EM iteration steps. If the log likelihood value does not change significantly then we conclude that the algorithm has converged and stop the process. The following is the formula to calculate the log likelihood value. The assumption that we have made is that the larger the likelihood value, better the model parameters fit the data.



This way we have performed the EM algorithm on the Gaussian mixture model data set to identify the underlying Gaussians in the data. The algorithm runs in a loop of 50 iterations where for each iteration we work the E-step and M-Step.The algorithm stops at the end of the 50 iterations or if at any iteration the log likelihood difference between two iterations is less than 1e-6. The following screenshot is the plot of the value of a point versus its density. The plot is a histogram plot containing the Gaussians identified by the EM algorithm. In our case we have three Gaussians.



The performance of the EM algorithm for identifying the underlying Gaussians from a1 data set can be based on two factors. Speed and the Error rate.

For the first performance measure speed of convergence, it depends on the factor that defines the amount of overlapping present in the mixture components. Having unbalanced mixing coefficients would ensure that the components have overlapping nature in them. Then when we perform the EM algorithm on such a data set it would take more iterations to converge on the other hand balanced mixing coefficients resulted in much faster convergence. The EM algorithm also provided accurate approximation of parameters on increasing iterations count when we are sampling the data is is well separated. For advocating this we initially tried the algorithm with 100 samples in the data set with no.of.clusters to be 2. The EM algorithm took 11 iterations to converge and get the parameters. Then the dataset was increased to 500 samples. This time the EM algorithm took 40 iterations to converge and provided the parameter approximations.

1. If we know the number of clusters from which the data is being sampled but none of the data in the clusters follow a normal distribution. Although the EM algorithm might identify the clusters but it is not something that can be guaranteed always. Sometimes the EM algorithm might determine the clusters present but the clusters might be inaccurate and not clearly separated.

The following figure depicts the story where on the left hand side we have a plot of the distribution of clusters that are non Gaussian. On the right hand side is the clusters identified by the EM algorithm. As you can see the algorithm wrongly categorizes class 2 cluster as class 1 and part of class 2 to be class 1. This provides support to the point that if the data is non- Gaussian then the algorithm does not perform well.



2. If the EM algorithm is not provided with the number of clusters in the beginning or if the number of clusters change based on different applications. In this case we could either use algorithms to identify the ideal number of clusters for that data set provided the dataset follows normal distribution. One example would be the usage of Bayesian Information Criterion for finding the number of clusters.

3. The error rate of the EM algorithm tends to increase with the increase in the difference of the size of the clusters. There is a direct association between the above two entities. We can conclude this fact on the basis of the experiment done where the size of one cluster was k times the other cluster, where 0<k<1. As the value of k decreased(one cluster getting smaller than the other which in case increases the difference between clusters) the error rate of the EM algorithm increased.   
The following figure is the plot of the error rate for varying values of decrease in size of one cluster, k.



References : Algorithm referred from

1. <http://tinyheero.github.io/2016/01/03/gmm-em.html>
2. <http://stats.stackexchange.com/questions/70855/simulating-random-variables-from-a-mixture-of-normal-distributions>

**Q2. Create a synthetic dataset of points in 3 dimensions. Project the data onto 2 dimensions such that the original distances are preserved as closely as possible. How is the performance of the algorithm affected by the nature of the dataset?**

**Creating a synthetic Dataset of points in 3 dimensions:**

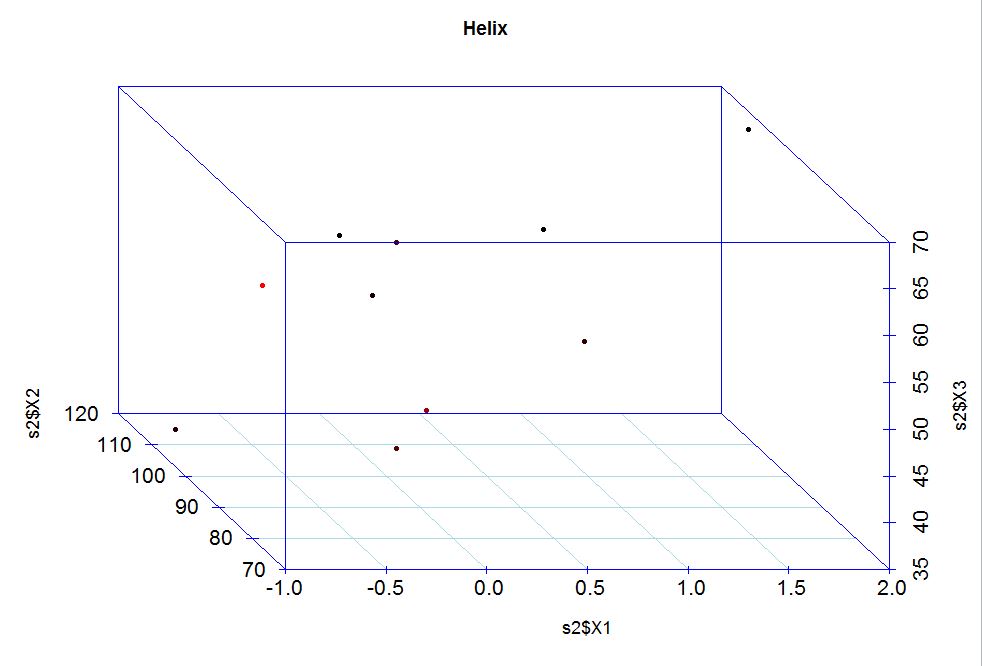
We have chosen the gendata package in R to create a synthetic data sets .

We have created the 3 dimensional data space with 10 points as sample and we tested the performance of Sammon’s mapping.

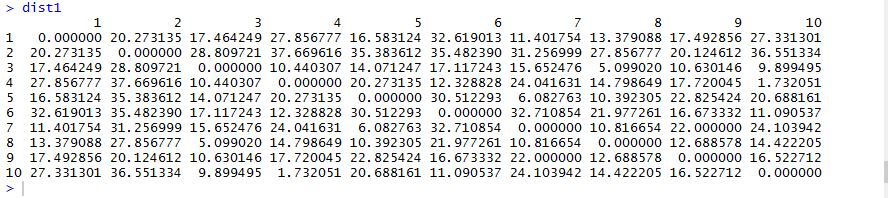
We created the data sets with three features having correlation of 0.7 , 0.2 & 0.3 . We have chosen this values randomly to make the points fall in three Dimensional space.

We have calculated the Euclidean distance pair wise distances between all these points. And We used this distance matrix between the points to check how well the distances in lower dimensional space are preserved.

Data in 3-Dimensional space



Initial Euclidean distances between the all points



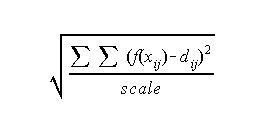
**Projecting Data onto 2 Dimension: Multi-Dimensional Scaling - Sammon's Nonlinear Mapping:**

We have chosen the Multi-Dimensional Scaling because the original distances in the High dimensional is preserved as closely as possible.

Aim is to project the every item in 3-Dimensional data to 2- Dimensional Data space. Since it doesn’t not depend on the linearity and normality, we found that it is the more powerful technique.

Stress function is measure of degree correspondence between the similarities in the 3- dimensional space and the similarities in the 2- dimensional space.

Formula:



dij – Euclidean Distance between the items i and j on the Reduced Space(2D)

f(xij) – function of the input data (3D)

scale- constant value which is used to keep the stress value between 0 and 1

**Sammon's Nonlinear Mapping:**

It is one of the variation of multidimensional scaling nonlinear metric.

Following the paper Sammons Mapping Paper introducing the mapping,

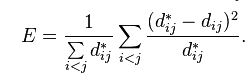
It represents the original dataset as 10 vectors in 3-dimensional space.

Original vector Xi , i = 1, . . . , 10.

We seek to map these into 2-dimensional space to give below data points

Reduced Vector Yi , i = 1, . . . , 10.

The Sammon’s mapping uses the following Stress function formula to minimize the error which is called the Sammon’s Stress.



d\*ij – denotes distance between i and j objects in Original space

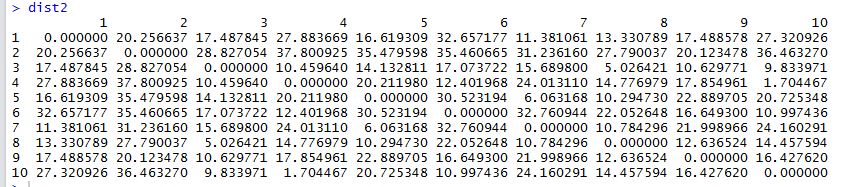
dij- distance between the projections of I and j in Reduced space.

We have used the MASS package sammon function to reduce the points to lower dimension.

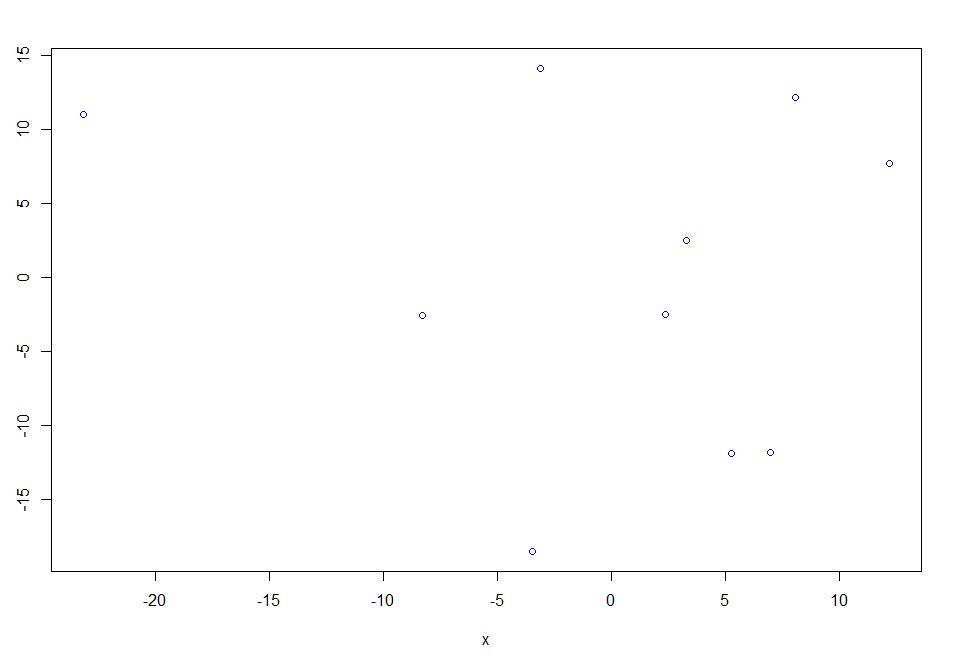
We set the initial configuration to classical multidimensional because the sammon mapping doesn’t run if the initial reduced stress from classical MDS is best fit.

We have kept the initial value of step size as 0.01 which is very low.

We have kept the tolerance very very low which is used to stop the iterations.Euclidean Distances between all points in Reduced space



Data points in 2-Dimensional Space(Reduced Space)

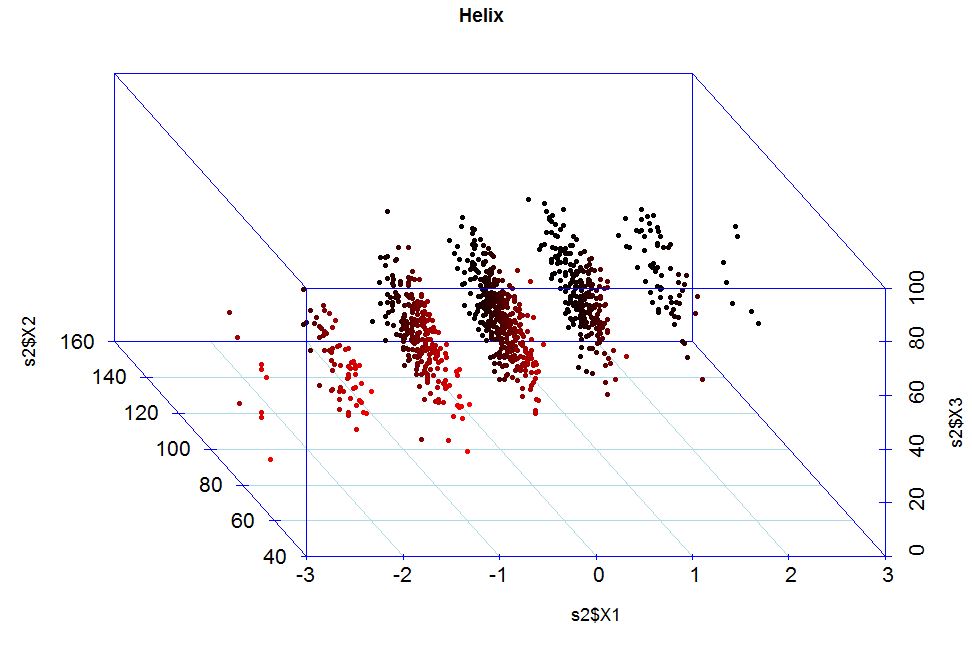


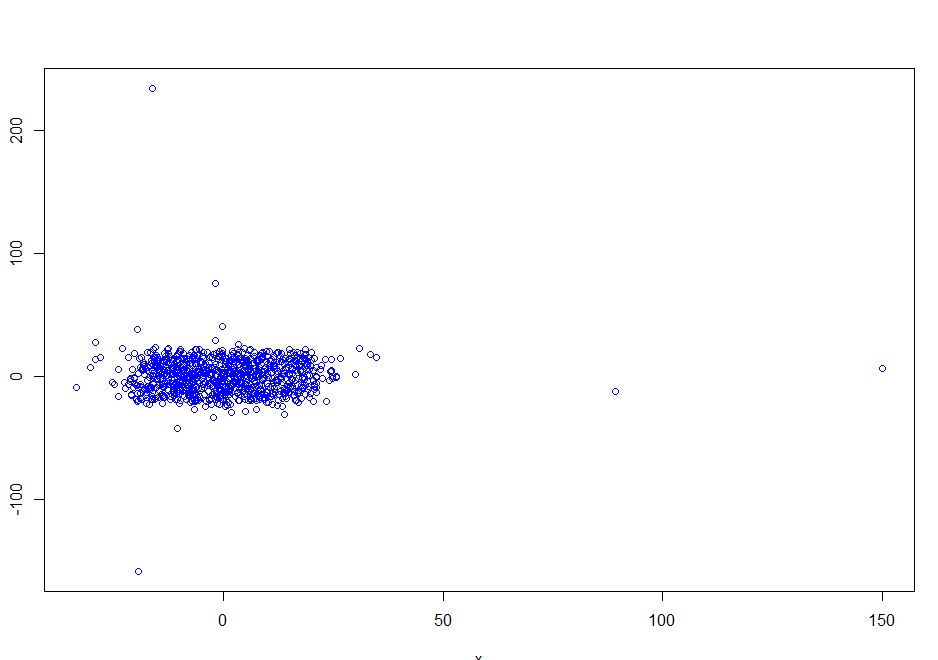
**Performance of the Algorithm affected by nature the Dataset:**

From the above dataset Euclidean distances, we can see the distances in the original and reduced data sets are preserved as closely as possible.

Consider the data set which is cylindrical, they are chances to stuck between the local minima while using the Sammons mapping. Hence Nature of Data set effects the Sammons mapping.

Consider there is a bigger proportion of data pattern in the data and we will be getting more nodes in that pattern using the Sammons mapping. Dense data conquers a larger area in the map than sparse data, even if their distribution volumes are equal in a data space. We may loose some important data when projected using the sammons mapping.





References:

<http://www.ee.oulu.fi/~nisu/projects/publications/dimred.pdf>

https://cran.r-project.org/web/packages/gendata/gendata.pdf