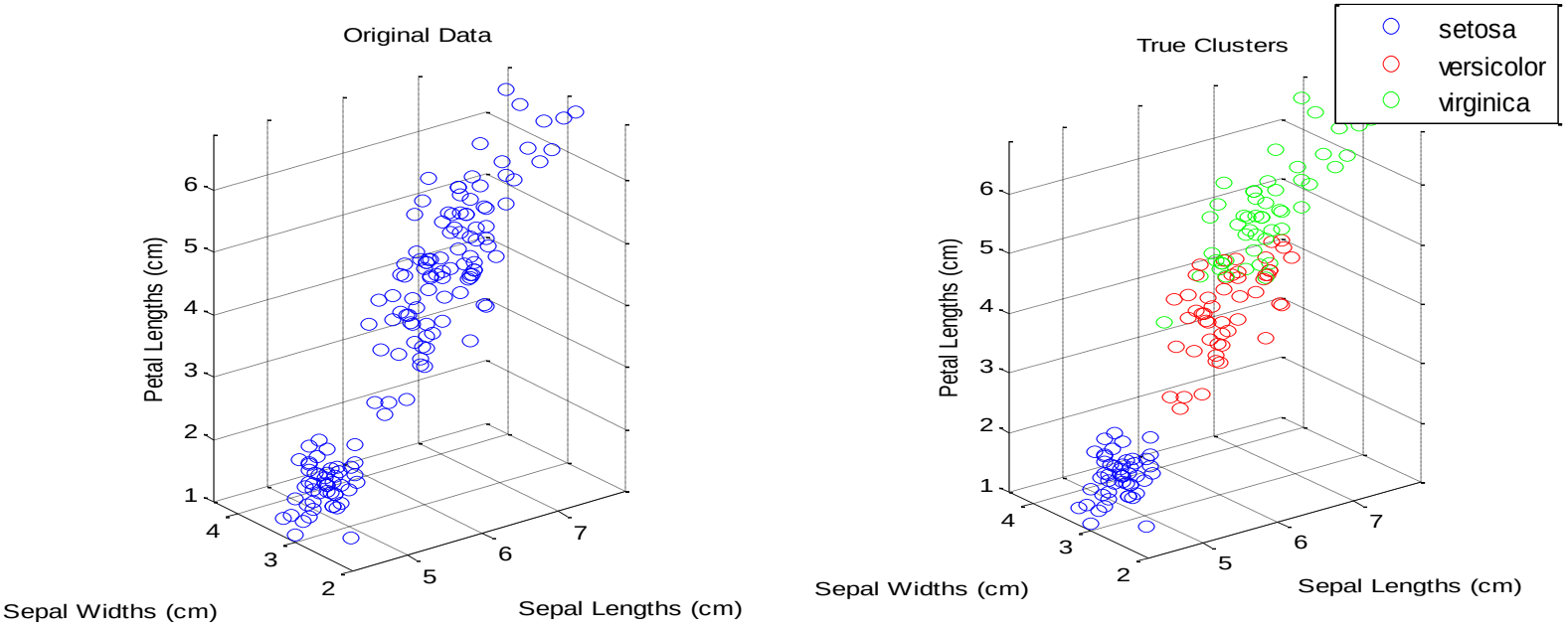
1 DATASETS

1.1 IRIS FLOWER DATASET (3 FEATURES AND 150 SAMPLES)

The Iris flower data set or Fisher's Iris data set is a multivariate data set having four features: the length and the width of the sepals and petals, in centimeters. However for the purpose of visualization I modified the dataset and the modified dataset contains only three features: sepal length, sepal width and petal length. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Following are the plots of the samples with respect to the features.



The scatterplot on the left-hand is without the labels, while right-hand plot uses true labels. As we can see from left-hand plot the data set only contains two clusters with rather obvious separation. One of the clusters contains *Iris setosa*, while the other cluster contains both *Iris virginica* and *Iris versicolor* and is not separable without the species information. It will be interesting to see how k-means clustering and Expectation Maximization perform when k= 3.

1.2 LEAF DATASET (14 FEATURES AND 340 SAMPLES)

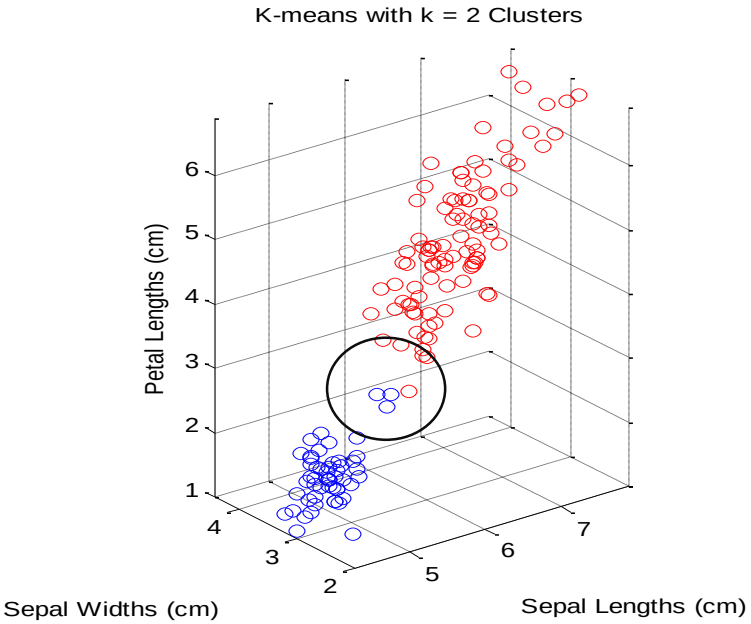
This is same dataset from Assignment 1. This dataset consists of collection of shape and texture features extracted from digital images of leaf specimens originating from a total of 30 different plant species (Description of the dataset says that it has 40 species, but is not the case). The classification goal is to predict the name of species (Label) based on the digital image of the leaf. Dataset has 14 attributes/features but only 340 instances.
Information about the attributions can be found under Dataset folder.

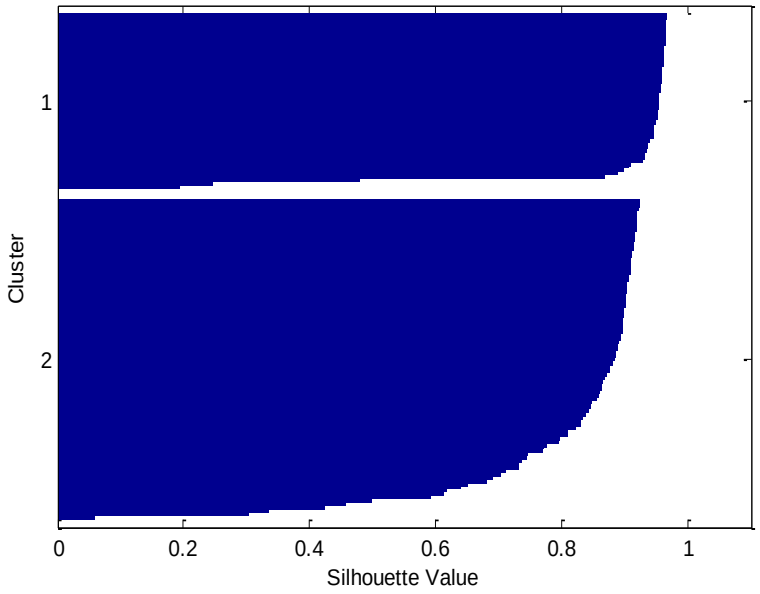
2 CLUSTERING ALGORITHMS

2.1 K- MEANS CLUSTERING

2.1.1 Iris Flower Dataset

I started with k = 2, and got the following plot. The distance metric used for squared Euclidean. Notice the circled region, some points from versicolor are considered to be setosa. This is not the case when I use correlation distance metric and is shown at the end of this section.
I replicated the algorithm 5 times, but it converged to same results for this setting (K =2).



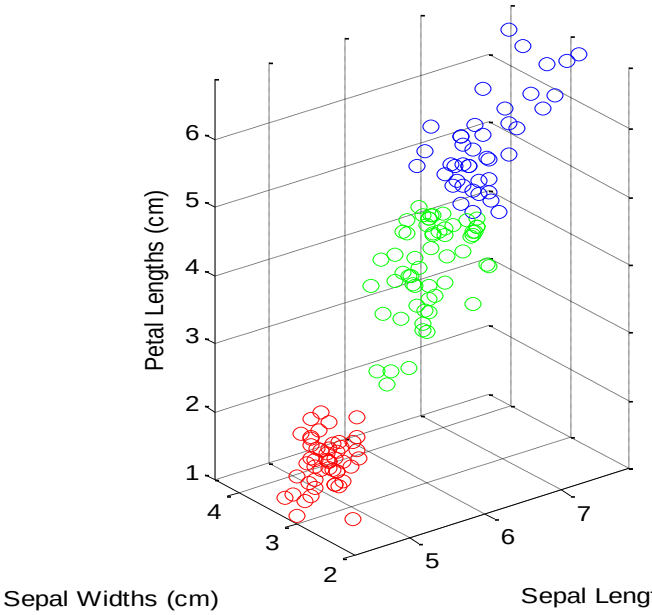


To get an idea of how well-separated the resulting clusters are, I made use of a silhouette plot using the cluster indices output from k-means. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. This measure ranges from +1, indicating points that are very distant from neighboring clusters, through 0, indicating points that are not distinctly in one cluster or another, to -1, indicating points that are probably assigned to the wrong cluster. As we can see from scatter plot and can verify from silhouette value that clusters are very well separated. The mean silhouette value was 0.8523. I increased the number of clusters to see if k-means can find a better grouping of the data and to see how well it represents true labels.

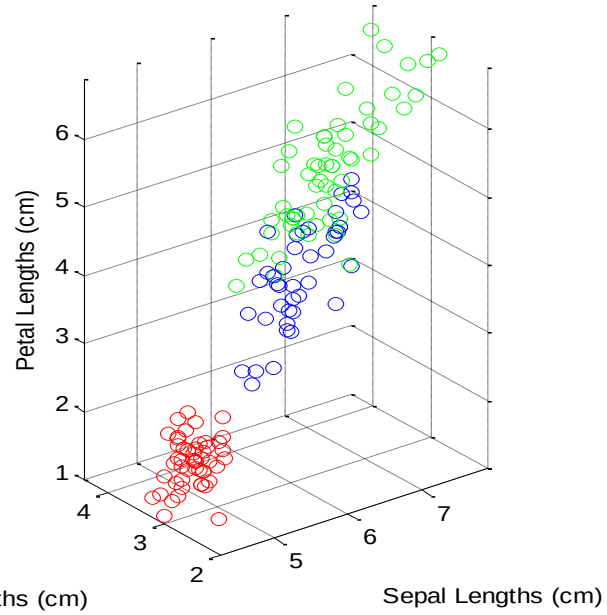
For k=3, I performed the algorithm using two distances- squared Euclidean and correlation. Clusters generated using correlation distance seems to line up with true labels as compared to squared Euclidean distance.

I plotted the silhouette plot for both the cases and due to space constraints plots were eliminated. Looking at the plots, it was clear that one cluster was well separated from other two, but other two were very close to each other. For correlation, silhouette plot looked good.

K-means with k = 3 Clusters and Squared Euclidean distance



K-means with k = 3 Clusters and Correlation distance

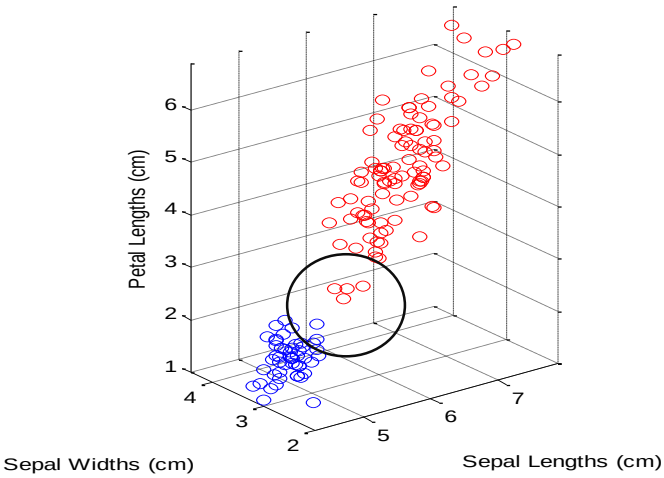


A more quantitative way to compare the two solutions is to look at the average silhouette values and the table shows the same. K =2, and correlation distance performs well and its graph is plotted below. The correlation distance has better performance as compared to Euclidean distance for this dataset and because only two clusters have obvious separation k=2 performs better than k = 3.

I ran algorithm for 5 times, and selected the best total sum of distances for all cases.

K	Distance Metric	Mean Silhouette Value	Elapsed Time	Iterations
2	Squared Euclidean	0.8523	0.200814	6
2	Correlation	0.9624	0.193206	4
3	Squared Euclidean	0.733	0.194892	4
3	Correlation	0.8564	0.19653	5

K-means with k = 2 Clusters and Correlation distance

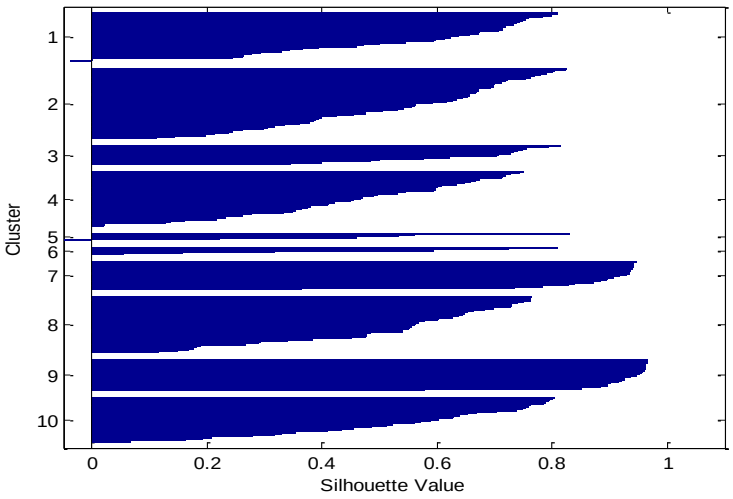


2.1.2 Leaf Dataset

I ran the algorithm 300 times and selected the one which gave best total sum of distances. This dataset has 30 different labels, so I selected k = 30, k less than 30 and k greater than 30. I ran algorithm for Squared Euclidean and Correlation distance metric and following is the table of the result.

K	Distance Metric	Mean Silhouette Value	Elapsed Time	Iterations
10	Squared Euclidean	0.5331	3.573759	12
25	Squared Euclidean	0.4905	7.304447	13
30	Squared Euclidean	0.4584	7.92366	15
50	Squared Euclidean	0.5164	9.021434	21
10	Correlation	0.5915	2.76889	20
25	Correlation	0.5182	4.243614	15
30	Correlation	0.5018	4.550542	26
50	Correlation	0.517	8.810445	11

It is surprising to see that for k=30, mean silhouette value is less than other cases. K = 10, and correlation distance metric performs the best and following is the silhouette plot for the same. Cluster 5 and 6 has less points. Cluster 7 and 9 are well separated from others.



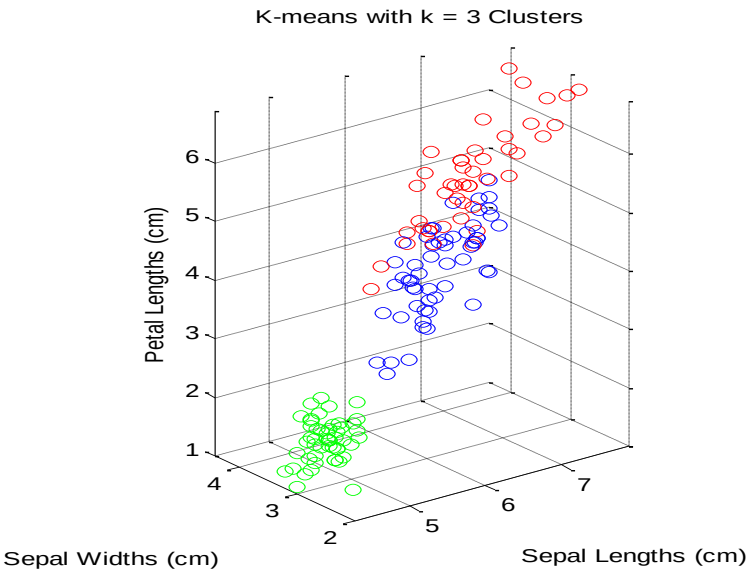
Notice that time increases when k increases.

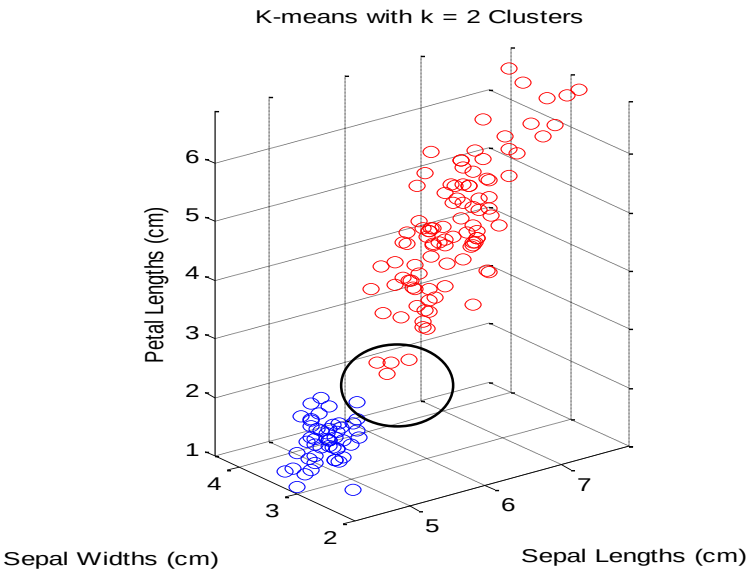
2.2 Expectation Maximization

2.2.1 Iris Flower Dataset

I ran the algorithm 10 times. Each time it converged to same log-likelihood value and same centers when k=2, but took varying number of iterations depending upon the starting points. Since there is obvious separation between two clusters, it converged each time to same centers. This was not the case when k=3. I took the best result. I defined 'best' in the case of the EM-algorithm as the highest value of the maximized expected log likelihood that was obtained among these runs

K	Silhouette Mean value	Steps required to Converge	Elapsed Time	Log-likelihood
2	0.9624	24	0.089113	-1.822692092
3	0.81	110	0.11575	-1.746178486





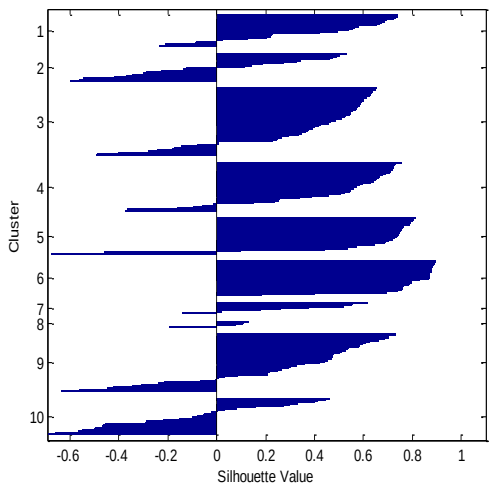
EM performed better as compared to K-means when K=2 (squared Euclidean distance metric). Notice the circled region for k=2. K-means had included those points into blue, but EM did a better job clustering them. For K= 3, both methods performed equally good and lined up with the true labels.

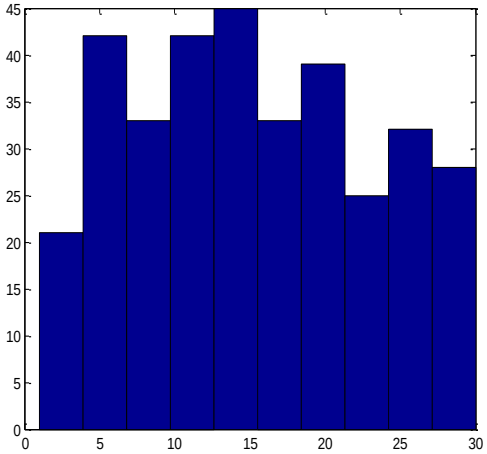
2.2.2 Leaf Dataset

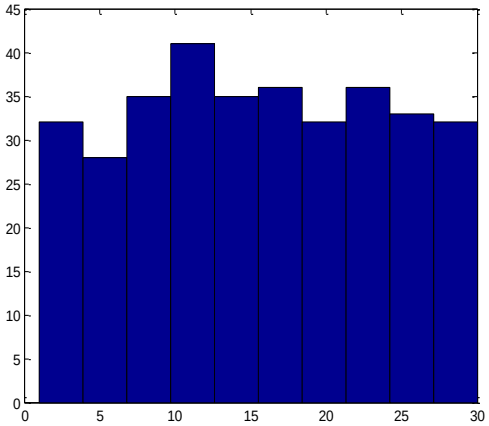
I ran algorithm for 10 times and selected the result which has highest log-likelihood value. EM didn't performed good for this dataset.

K	Silhouette Mean value	Steps required to Converge	Elapsed Time	Log-likelihood
10	0.341	83	0.24	40.58971084
25	0.2606	12	0.10918	46.3894218
30	0.2937	8	0.091648	45.54171375

Silhouette value is very less, and says that clusters are not separated. This is also visible from the plot (k =10) shown below. Many of them are misclassified and is verified from histogram plot. Following are the histogram plot of true labels (right) and one obtained when k=30(center). This shows that the labels obtained don't line up with the true ones.







3 DIMENSION REDUCTION ALGORITHMS

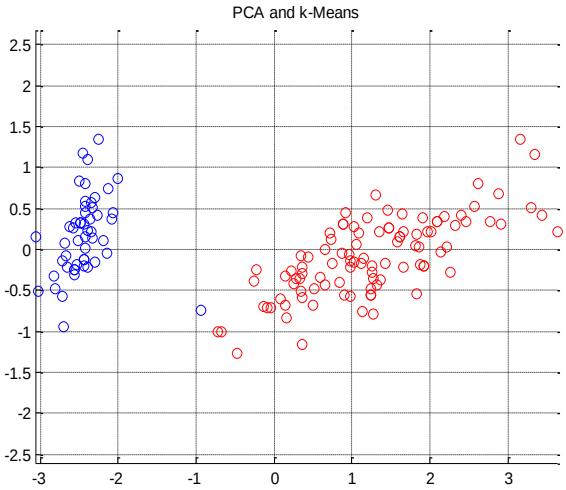
3.1 PCA AND ICA

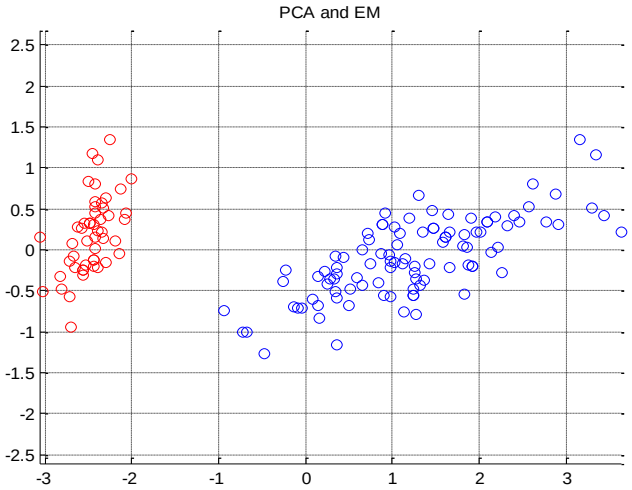
3.1.1 Iris Flower Dataset

I ran the PCA on this dataset which has 3 features in MATLAB, and following were the eigen values I got.

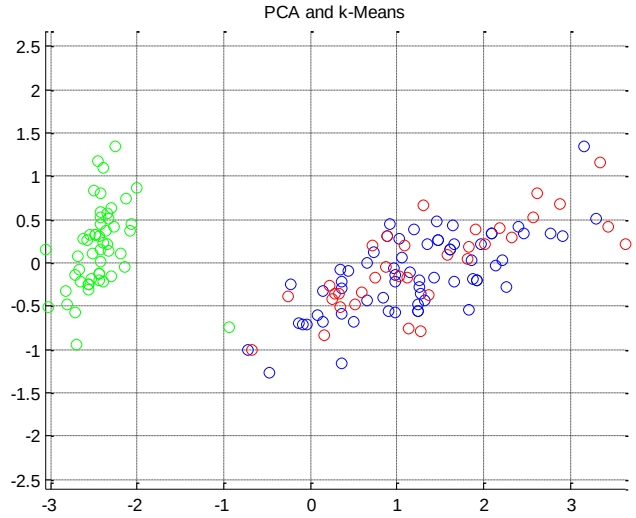
$$\{3.69111978893677|\ 0.241377272788921|\ 0.0594537212720684\}$$

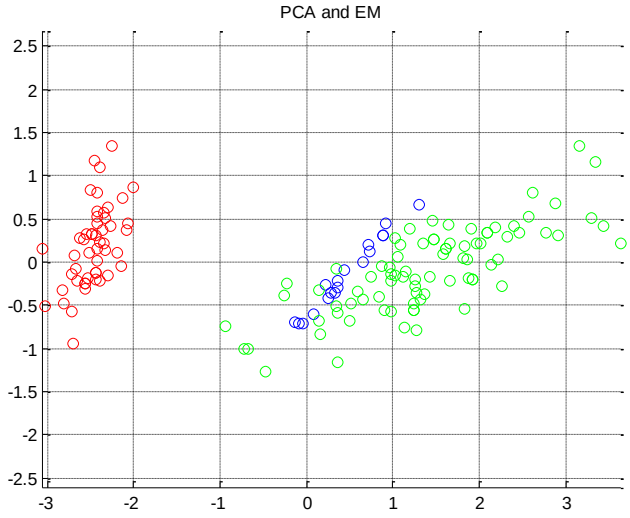
I decided to neglect the last eigen value and generated the reduced data and ran clustering on them. Following were the results obtained





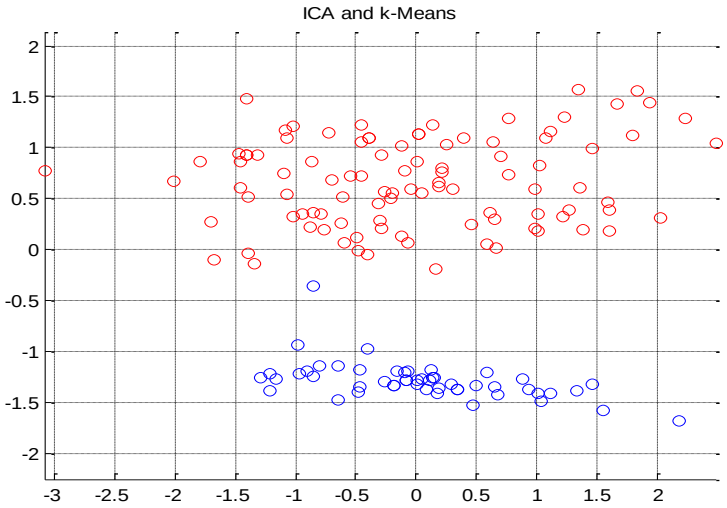
Both clustering algorithm work well for k =2 and I got the same result as in section 2.1.1. This was not the case with k=3 and following are the results.

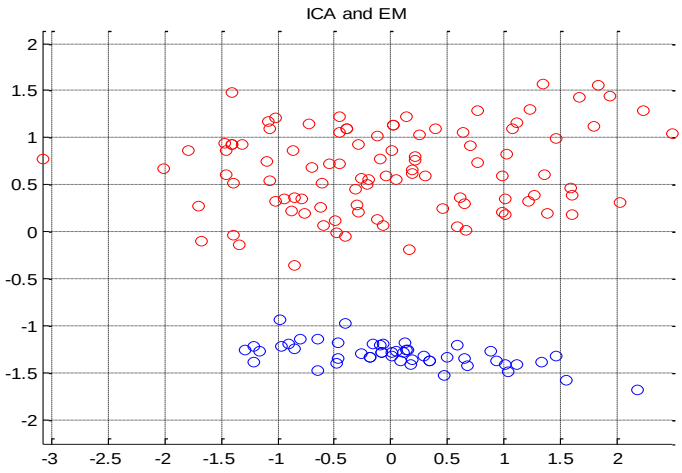




I ran ICA with dimensions =2 for this dataset and following was kurtosis of two features.
Kurtosis = [2.8093 1.5751]

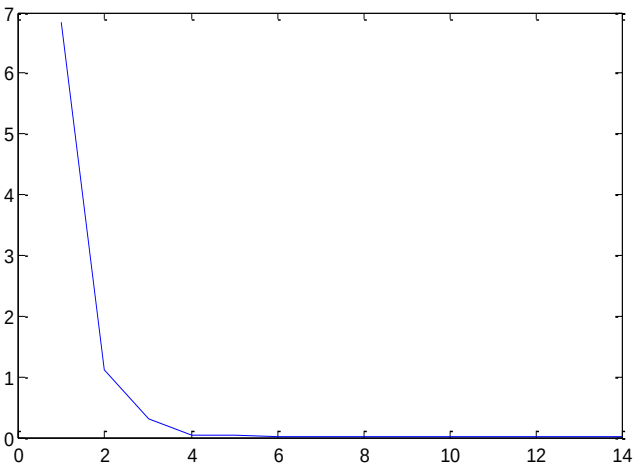
Each time I ran the ICA and clustering algorithm, K-means gave me different clusters while EM gave same.





3.1.2 Leaf Dataset

It is not possible to analyze this data set visually.
Following is the plot of the eigen value distribution for PCA and looking at the plot we can conclude that four features capture the data very well.



The kurtosis values for 6 ICA features are
[2.0768 1.9290 1.8954 2.2785
1.8914 2.2016]

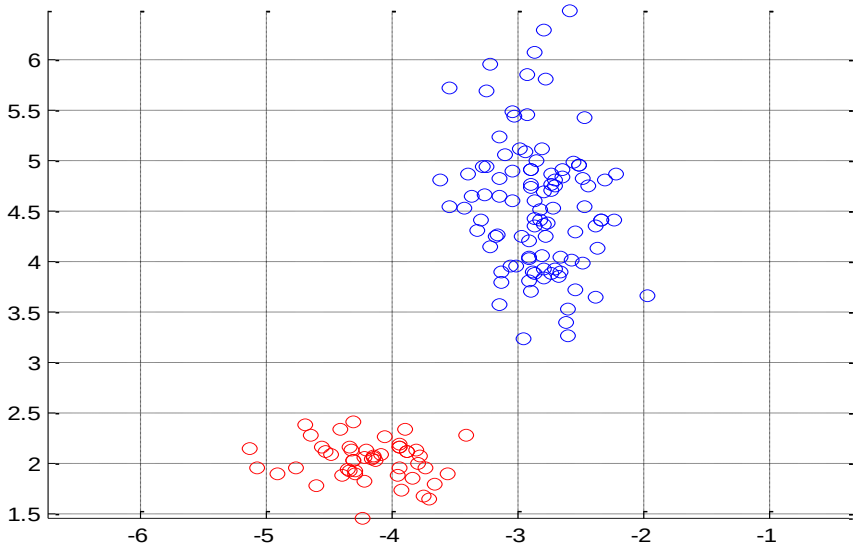
Dataset	Method	Dimensions used	Reconstruction Error
Iris Flower	PCA	2	52.4775
Iris Flower	ICA	2	546.0199
Leaf	ICA	6	3.08E+03
Leaf	ICA	4	3.45E+03
Leaf	ICA	14	5.88E-26
Leaf	PCA	6	261.5275
Leaf	PCA	4	1.03E+03
Leaf	PCA	14	8.81E-27

Above is the reconstruction error for various cases. As expected as we go on ignoring features, reconstruction error goes on increasing. Notice for all dimensions(14), error is almost negligible.

3.2 RANDOMIZED PROJECTION

I use the following technique to construct randomized projections. Assume our we have n samples of our data, each has d dimensions. We arrange our data column-wise in a data matrix $X \in R^{d*n}$. Our goal is to reduce each of our n samples to $k << d$ dimensions. That corresponds to a projections matrix $A \in R^{k*d}$, with the rows of A containing the base vectors of the subspace we project on. The question arises: What makes a good projection matrix? While PCA and ICA construct A w.r.t. to optimize certain goals described above, the randomized projection technique just generates A randomly. It is preferable to be A an orthonormal matrix, so that the projection is although not isometric but at least norm-decreasing. I used MATLAB functions randn() and orth() to produce A. I created several random matrices and then selected best depending upon the reconstruction error.

Following is the plot for Iris Flower Dataset with reconstruction error of 15.7815. I created 1000 random matrices and selected this one, as it had minimum error. RP was fast as it didn't had to compute anything as compared to PCA and ICA. Clustering was performed using K-means. EM also gave me similar results, plot for which is omitted due to space constraints



Even for Leaf Dataset I created 1000 random matrices and selected one with minimum error. Following table is for varying number of dimensions and reconstruction error. As expected, as number of dimensions goes on increasing error decreases

Number of Dimensions	Reconstruction Error
4	1.86E+03
6	1.15E+03
8	573.2249
14	1.79E-27

3.3 SEQUENTIAL FEATURE SELECTION

This method has two components:

An objective function, called the criterion, which the method seeks to minimize over all feasible feature subsets. Common criteria are mean squared error (for regression models) and misclassification rate (for classification models).

A sequential search algorithm, which adds or removes features from a candidate subset while evaluating the criterion. Since an exhaustive comparison of the criterion value at all 2n subsets of an n-feature data set is typically infeasible (depending on the size of n and the cost of objective calls), sequential searches move in only one direction, always growing or always shrinking the candidate set.

The method has two variants:

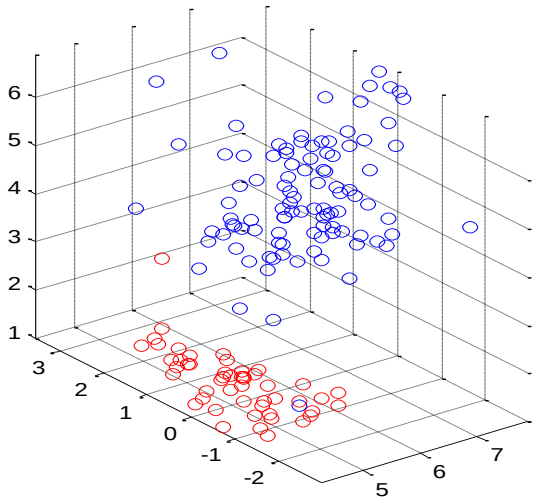
Sequential forward selection (SFS), in which features are sequentially added to an empty candidate set until the addition of further features does not decrease the criterion.

Sequential backward selection (SBS), in which features are sequentially removed from a full candidate set until the removal of further features increase the criterion.

This method comes under wrapper type. I used SFS and quadratic fit as ML model. In order to demonstrate this method, I created a random matrix with 150 samples and 10 predictors. I inserted the original 3 predictors from Iris flower dataset at position (1, 3 and 7). I ran the algorithm. Following was the output

Step 1, added column 7, criterion value 0.0466667
Step 2, added column 1, criterion value 0.04
Step 3, added column 2, criterion value 0.0333333
Final columns included: 1 2 7

Each time we get different output, but most of the time feature 1 and 7 is selected. Feature 1 corresponds to sepal length while Feature 7 corresponds to petal length.



I ran clustering using K-means. Similar results were obtained for EM. Same method was extended to Leaf Dataset. But due to less number of samples and more labels, the quadratic discriminant analysis model cannot fit to the data. The covariance matrix no longer remains positive semi-definite. I think this issue can be handle by either changing the model or by increasing the number of samples. I changed the model to LDA and it selected only one feature (1st feature).

4 NEURAL NETWORKS

In assignment 1, I had shown that this dataset performs better for 6 layers with size of each being 50. As in assignment 1, I used Gradient Descent Backpropagation algorithm with Momentum Learning rate was 0.4 and momentum was also 0.4.
I ran NN for dimension reduction algorithms and following were the results. They didn't perform better than the original one but there was reduction in time requirement which should be the case.

Method	Dimensions used	Training Error	Validation Error	Testing Error	Training Time	Testing time
PCA	4	12.4	23.6	19.8	71.226278	0.821069
ICA	4	7.8	23.4	19.6	40.600987	0.836613
RP	4	11.8	21.5	21.6	70.422861	0.865408
Original	14	1.48789	10.8	10.4	153.664770	0.838638

I then ran the algorithm for clustering algorithm, treating them as dimension reduction algorithm and following were the results. Training error went further down, suggesting that the results are not very great. EM performs bad then k-means. As only one feature is being used, time require is very less.

Clustering Method	Training Error	Validation Error	Testing error	Training Time	Testing Time
k-means(30 Clusters)	26.3	29.3	28.4	16.271957	0.845836
EM (30 Clusters)	32	32.5	32.4	4.201707	0.861398

I also applied clustering algorithms on reduced set, and then treated the new labels as inputs to NN. Following were the results obtained. The performance for some condition were better while for some other were bad.

Clustering Method	Dimensionality Reduction	Training Error	Validation Error	Testing error	Training Time	Testing Time
k-means(30 Clusters)	PCA	31.8	32.5	31.8	3.664035	0.82199
EM (30 Clusters)	PCA	21.1	21.2	20.8	16.62799	0.839973
k-means(30 Clusters)	ICA	20.5	26.1	27.8	40.609564	0.981218
EM (30 Clusters)	ICA	31.9	32.1	32.5	4.367144	0.808012