Author: Apurva Sharma CS7641
GTID: 902490301

Assignment 3 : Unsupervised Learning and Dimensionality Reduction

In this assignment, we focus on clustering algorithms like k-means clustering and EM algorithm. Once that is done, we explore dimensionality reduction techniques like PCA, ICA, Randomized Projections and Latent Semantic Analysis. After that, I will attempt to join the dots and make comparisons of applying/augmenting these strategies on Neural Nets and make comparisons with the experiments carried out in Assignment 1. For this very reason, I do not change the datasets so its easier to make these comparisons.
A brief recap of the data sets chosen is given below. For a detailed description including histogram plots, please refer to Assignment 1.

Datasets Description:

Soybean:
This dataset is collected from a survey of soybean diseases from an inspection of 683 instances and 36 (35 for description and 1 for class) attributes are used to describe each instance. Based on the symptoms, the plant is classified to have one out of the nineteen possible diseases. The data set contains some missing values.

SpamBase:
This is a dataset consisting of 4601 instances and (58) 57 + 1 class attributes with all of them continuous except the class label which is nominal. As opposed to Soybean, this dataset is complete. A majority of the columns are dedicated to indicate the frequency of select keywords/characters encountered mails. Attributes 55-57 measure run lengths of consecutive capitals. Based on this information, a mail can be marked either as spam (1) or non-spam (0).

Clustering Algorithms:

1. *k-Means Clustering:*

k-Means tries to form k clusters for N data points where each point is assigned to a the cluster whose centroid is the closest. A key aspect, then become the metric of distance being used. Ideally I would have liked to use Squared Euclidean distance as for it, the centroid is the arithmetic mean. However, since Weka currently has support only for Euclidean and Manhattan distances, I decided to go for Euclidean distance. Another key selection choice is the value of k itself. Since I have the luxury of knowing the labels, one value of k that I will definitely try is the number of classes itself. However, I do realize that clustering algorithms are typically used for unsupervised learning methods and hence will try a range of k values. Another point worth pondering is how do we know what is a good clustering? There are two ways of going about this : one is the visual inspection of the clusters w.r.t. the instances, clusters and the class/the attribute with the maximum information (calculated using Information Gain evaluation for unlabeled data) however, this is not always simple. Hence, another measure than can be used is the 'Within cluster sum of squared errors'. However, we can't generalize that lower this value, the better as take the case where each point is defined as a cluster in itself thereby giving a zero value. Very large values of this are not suitable too, as that means that the clusters may not be dense at all. A good measure maybe observing the trend in this parameter for variations in k and picking k where the value starts decreasing at a visibly slower rate (Elbow Analysis).

Intuitions before experiment:

1. Since I know the number of classes before hand, I expect good clustering results around this value of k.
2. I also expect the clusters to look similar w.r.t. (instances, clusters and class) and (instances, clusters and the attribute with the maximum information [calculated using Information Gain evaluation]).

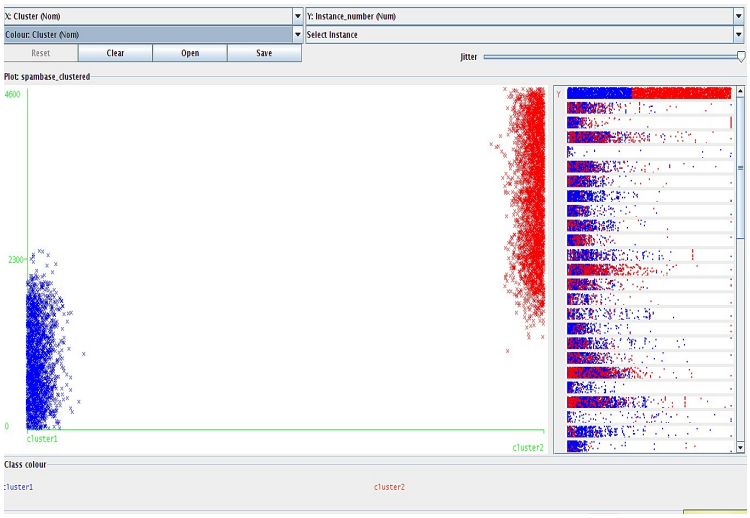
Experiment Setup:
The main experiment is variations in k and observing clustering formed. Then these are evaluated on basis of the approach described above.

Analysis, Possible Explanations and Lessons Learnt:

In the process of experimentation, I found out another feature in Weka which does classes to clusters evaluation. This again gives a good idea of what k values are: but once again, this only for validation purposes as in unsupervised learning, classes are not known.
For k=2 for the spambase dataset, we get a fairly good clustering. If you notice carefully in figure 1 (a), none of instances

belonging to class 1 appear in the cluster of class 0 and vice versa. When I try to do the same visualization w.r.t the attribute with the most information gain, I do not get a very conclusive picture as the attribute takes continuous values and hence the color gradient is very smooth and difficult to identify.
The elbow analysis seems to suggest a value of k~10 for the spambase dataset. For soy bean it is a little tougher to spot the elbow but it seems to be near 25. In case of spambase it seems to be a little off the target as for validation, when runs for classes to cluster

evaluation are done, it assigns classes to only 2 of the clusters. It does a little better for the soybean dataset with reasonable clustering for k=25. Worth noting is that for cases like k=10, still there is not a single presence of blues(spam) in reds(non spam) indicating that the clustering is still clean. For soybean, on the other hand, there are many cases of such overlaps which makes sense because the number of classes is reasonably large with fewer instances.



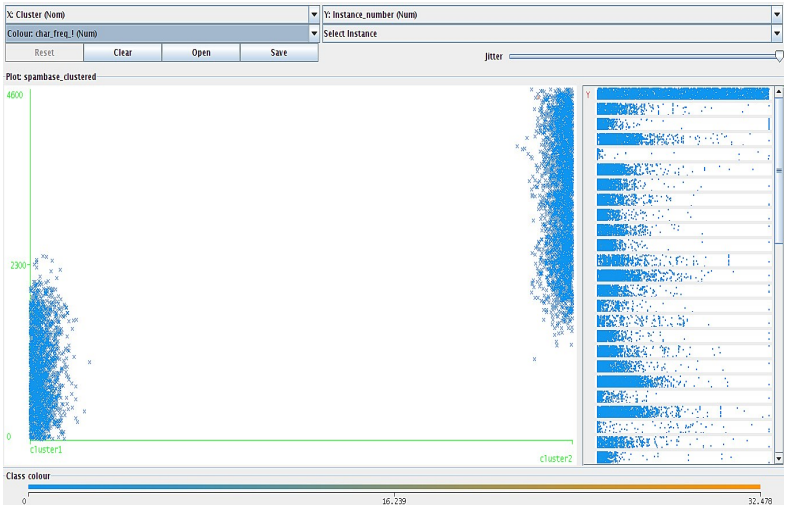
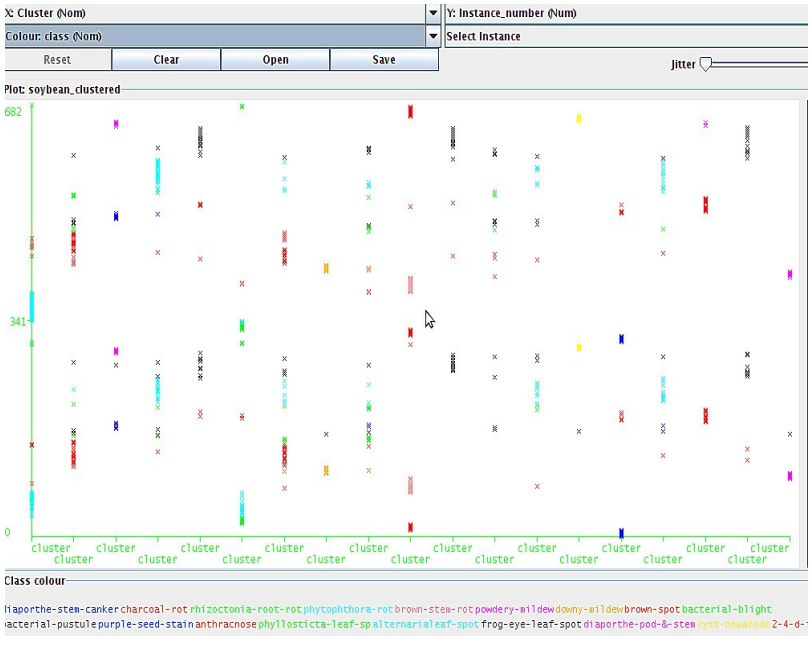


Fig 1(a)(b) Clustering Visualization for (cluster,instance,class) vs (cluster,instance, best attribute) for spam dataset. K=2



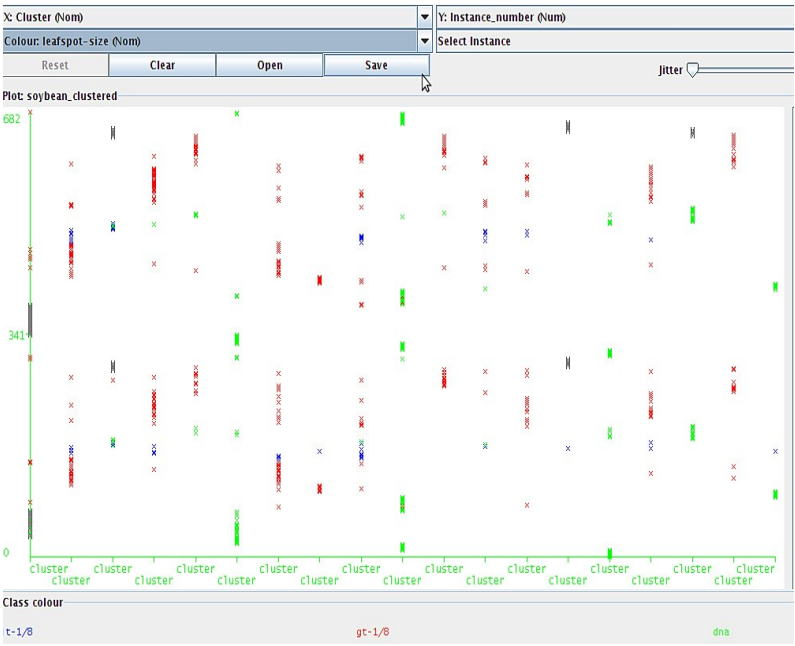


Fig 1(c)(d) Clustering Visualization for (cluster,instance,class) vs (cluster,instance, best attribute) for soy dataset k=19

Lesson learnt is that it is definitely not a simple task to choose k and as I explored more literature about this, there seems to be seems to be ways of getting k (ranging from simple thumb rules to the elbow analysis and far more complex models). Based on my experiments, I am not entirely convinced about the elbow analysis as of now. Another lesson learnt is that since k-means assigns the initial clusters randomly, it is a good idea to run multiple iterations. I did so and picked the best results.

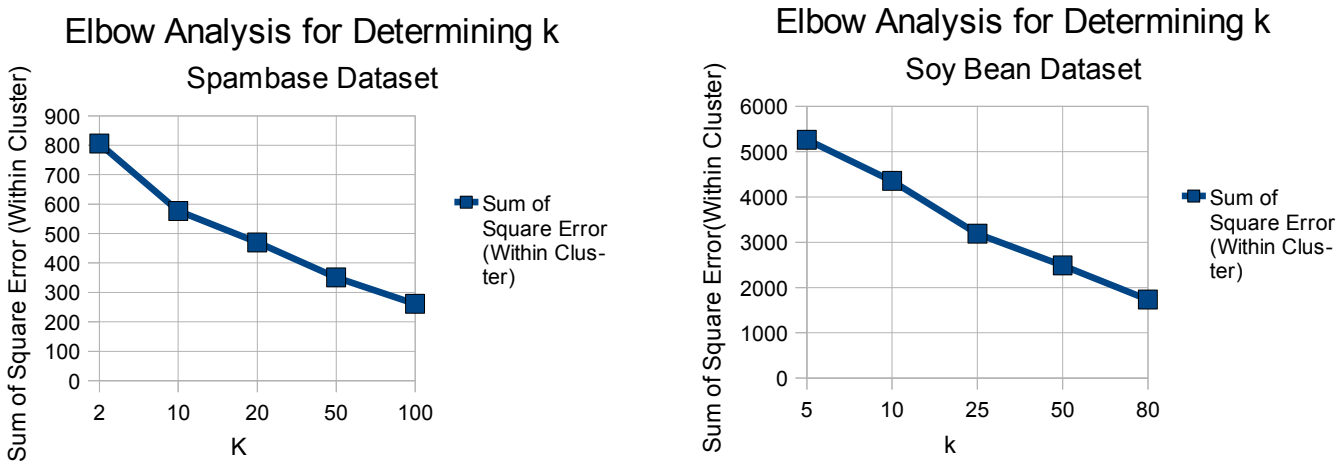


Fig 2 (a)(b) Elbow analysis for picking k for Spam and Soy Dataset

EM Algorithm:

Since EM Algorithm tries to find the m.l.e. of parameters in statistical models, it is relatively easier to evaluate. Higher values of expected maximized likelihood would indicate better results. Also cross validation support for determining the best number of clusters is also present in weka. This feature is used and compared with the actual number of classes.

Intuitions before experiment:

1. I expect the number of clusters suggested by cross validation to be close to the number of classes.

Experiment Setup:

1. The main setup is visualization of clustering as per the number of clusters suggested by cross validation.
2. It will also be interesting to observe clustering for other values of k.
3. Also, the trend of log likelihood w.r.t. k should be interesting to see.

Analysis, Possible Explanations and Lessons Learnt:

One of the surprising outcomes is that the suggested number of clusters for the spam dataset is 13! This is considerably different from the expected value (close to 2). To verify this, I ran the EM algorithm witk k=2 to see if the clusters look any better for k=2. Visually, k=13 seems to present a better clustering, with some of the clusters having a fairly dominant class present. K=2 seems to convey very little information, with each cluster containing a fair share of instances from either class.

For the soy dataset, the results are more or less on the expected lines. The suggested value is k=14 which is fairly close to the number of classes (19). Also, the clustering looks fairly separated with almost every cluster being dominated by a particular class. The cluster diagram for it has been omitted due to space restrictions but its sufficient to say they behave as expected.

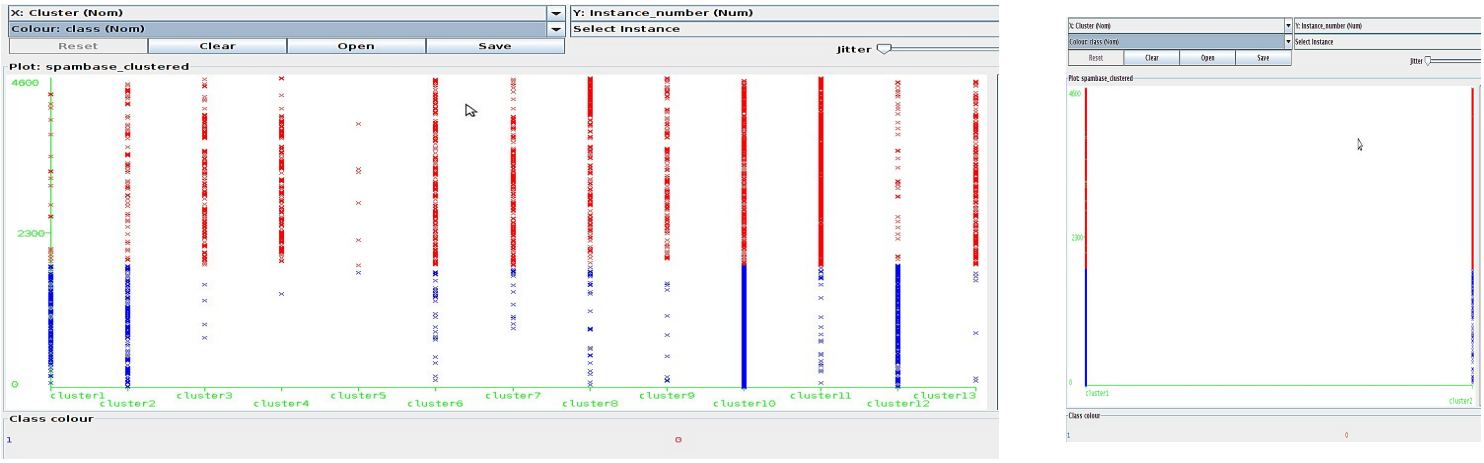


Figure 3 : EM Clustering Comparison for k=13 (suggested by cross validation) vs k=2 (number of classes)

Variations in k have the expected results (similar to the elbow analysis of k-means). As the value of k increases the log likelihood increases but the rate slows down considerably after a certain value of k. In EM algorithm, elbow analysis seems to make a lot of sense. First of all, for the spam dataset, there is a very prominent elbow near k=10 which is confirmed by the cross validation suggesting k=13. For the soy dataset, it's a little more tricky as the growth is very gradual. Yet, it's not hard to see that growth rate
retards significantly after k=25. This can be seen in Figure 4 very clearly. A visual inspection of the clusters seems to support this argument.
A key lesson learnt from these clustering algorithms is that the results depend a lot on the nature of the datasets chosen. What we can do, however is apply techniques like the one's I used (and many more) to try and get a value of k which gives reasonably good clustering results.

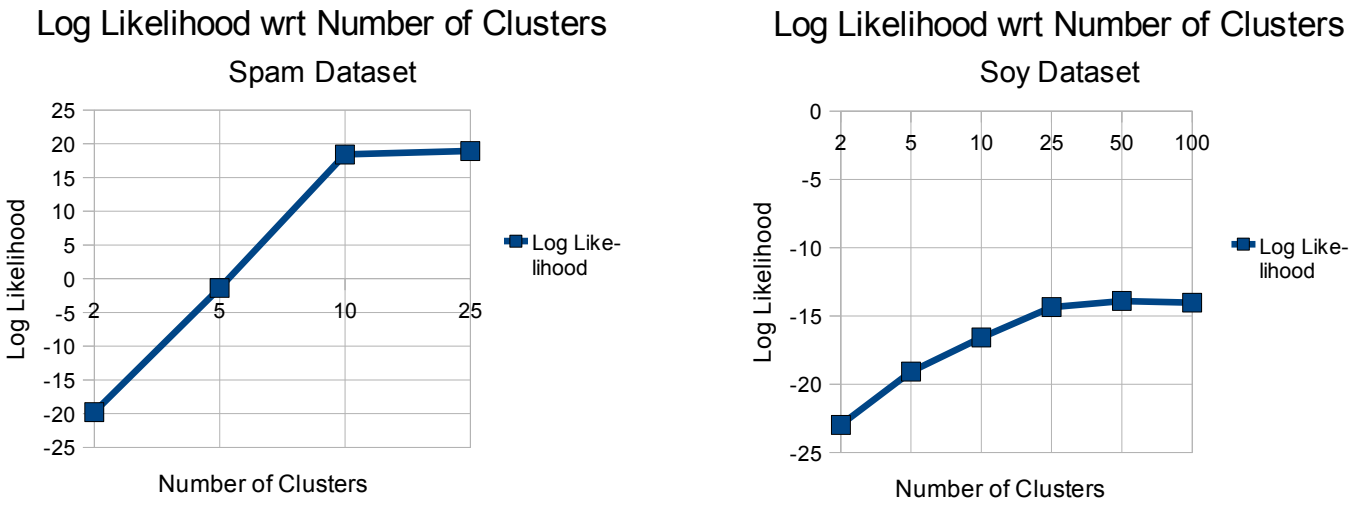


Figure 4 Elbow Analysis in EM Algorithm

Dimensionality Reduction:

PCA:

PCA reduces dimensionality by means of attribute transformation as opposed to attribute selection. Also noteworthy is the fact that the largest eigen values are the one's that capture the most variance and hence should be preferred while selecting transformed attributes. As expected, there is loss of information when transforming to reduced dimensionality, however, one of the strengths about PCA is supposed to be that there is minimum error while projecting back to the N dimensional space f

rom the k-dimensional space(k<<N).

Intuitions before experiment:

1. I have already read that as the eigen values decrease, lesser variance is captured and that this rate of variance decrease is generally pretty sharp. So, I expect k should indeed be <<N for a reasonable amount of variance capture (say 95%).
2. I expect a near full recovery when trying to project back to the N dimensional space as PCA is supposed to be good at this.

Experiment Setup:
1. The main setup is trying to vary the variance retained and seeing how many dimensions is the problem set reduced to.
2. Another setup will try to inspect the recovery skills of PCA.

Results:

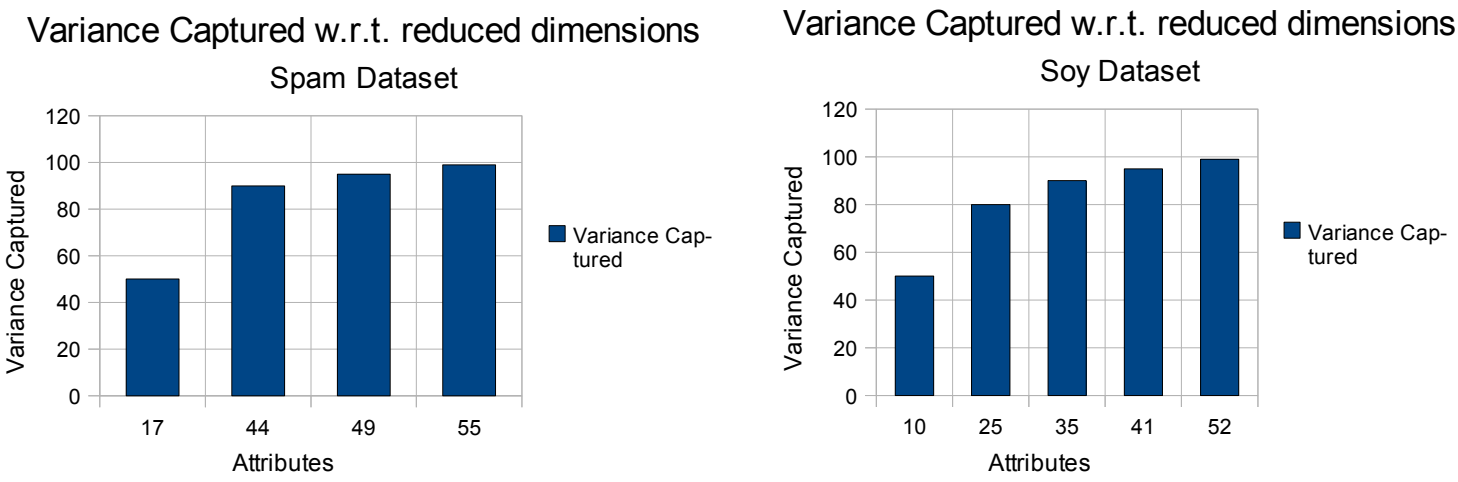


Figure 5: Distribution of Principal Components required w.r.t. variance captured

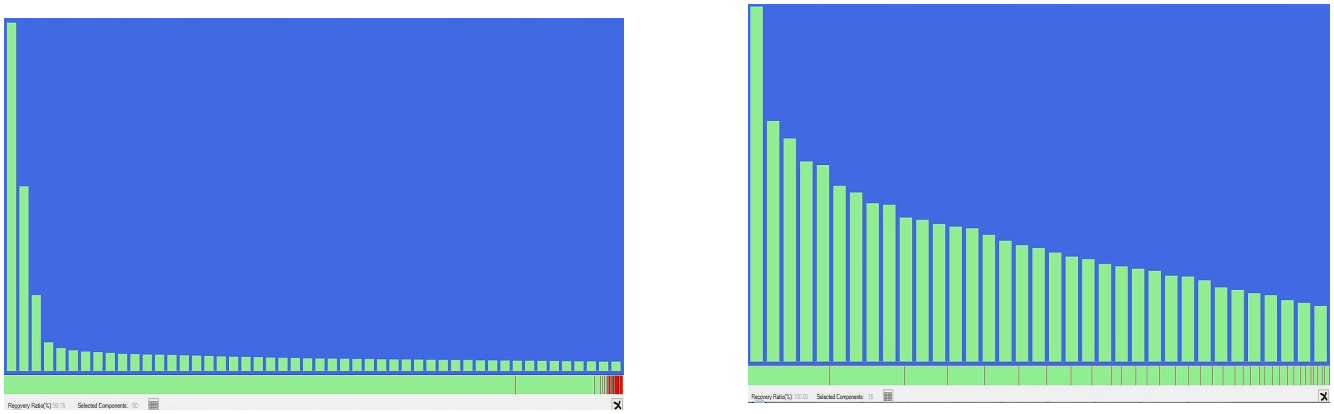


Figure 6: Recovery w.r.t. Principal Components

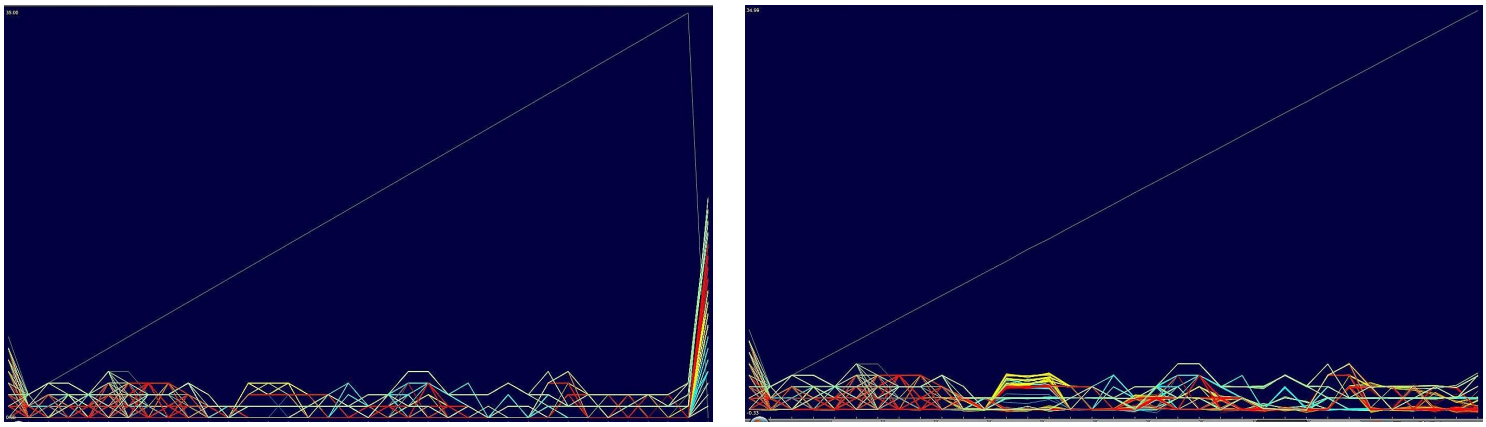


Figure 7: Actual Sample Recovery for Soybean through Value Diagrams. Left panel shows original value diagram. Right panel shows 97.6% recovered value diagram by using 48 principal components.

Analysis, Possible Explanations and Lessons Learnt:
Figure 5 brings in a very interesting find. The soybean dataset has only 36 attributes but according to PCA, to preserve more than ~80% variance, we need more than 36 projections! This can be seen from Figure 6 too. As opposed to Spambase, for soybean, the cumulation variance capture is not as high and in fact, for a full 100% variance capture, more projections than original number of dimension are needed. This is where we need to take a call as to how much variance loss is admissible. If we are willing to lose upto 20% variance, we can reduce the dimensions from 36 to 25. What it's repercussions are will be explored when we run the neural network on these reduced datasets.
In terms of recovery, as expected, PCA does recover well. Figure 6 and 7 support this thought. A similar plot for spambase (as Fig 7) was calculated, but since the behavior was similar, it has been omitted due to space restrictions.

ICA:

For this experiment, I use the Non Gaussianity family of ICA instead of MMI(Minimization of Mutual Information). This means that I rely on measures like kurtosis of distributions for varying number of k (k<<N). The projections on these projection axes are statistically independent for ICA.

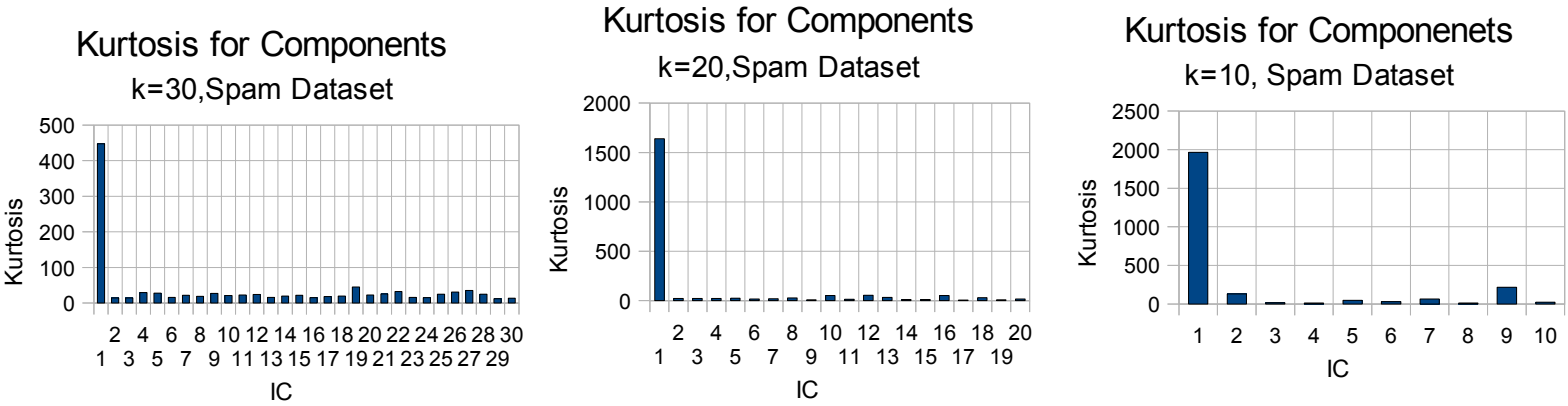


Figure 8 : Component wise Kurtosis for varying k

ICA tries to rotate the projection axes so that Gaussianity of projections on all axes is minimum. This can be verified by studying the kurtosis of projections on each IC axis. This is precisely what is done in Figure 8. There is not a single IC axis with kurtosis value =0 implying no Gaussian projections. In fact for IC1, kurtosis value is significantly higher than any other axis across variations in k.

The motivation for reducing this Gaussianity is to enable recovery of “original sources” which are statistically independent

(derived from Central Limit Theorem).
How well these reduced datasets perform will be further known when I run the Neural Nets on these reduced datasets. A good performance would indicate a good reduction as we would have captured the important features which aid in classification and thus capture the true essence of the dataset. Similarly, a bad performance there would imply that in the process of reducing dimensionality, we have missed out on some crucial information. Often we may agree to some drop in accuracy provided that the dimensionality reduction is significant enough.

Randomized Projections:

For randomized projections, I had to write my own code which is attached as a part of the submission. Here, I create a Randomized Projection Filter which is nothing but a multiplier for taking projections. This multiplier is a matrix of dimensions (k,N) and consists of randomly generated values. Then svd analysis is done to get the final projection multiplier matrix. Once this is done, projections are taken to reduce dimensionality to k attributes. A recovery operation is implemented too which uses the transpose of the projections and the filter as well as gives the error in recovery which is calculated as log(Σ(i,j) abs(recovered(i,j))-abs(input(i,j)))). I chose this over the sum of squares error as due to large number of components and random projections, sometimes the errors were too huge and exceeding the float capacity.

Experiment Setup:

I ran randomized projections over a 100 iterations for the spambase data set and observed the recovery errors. For soybean dataset, first I had to apply the nominal to numeric filter.
Since the whole approach is random, there are very limited evaluations possible for this approach. One is actually testing the neural net on the reduced data set to see how it works. Worth keeping in mind is that there is no guideline here to help us choose k. Another measure which we can take is the recoverability error which can be calculated as described above.I

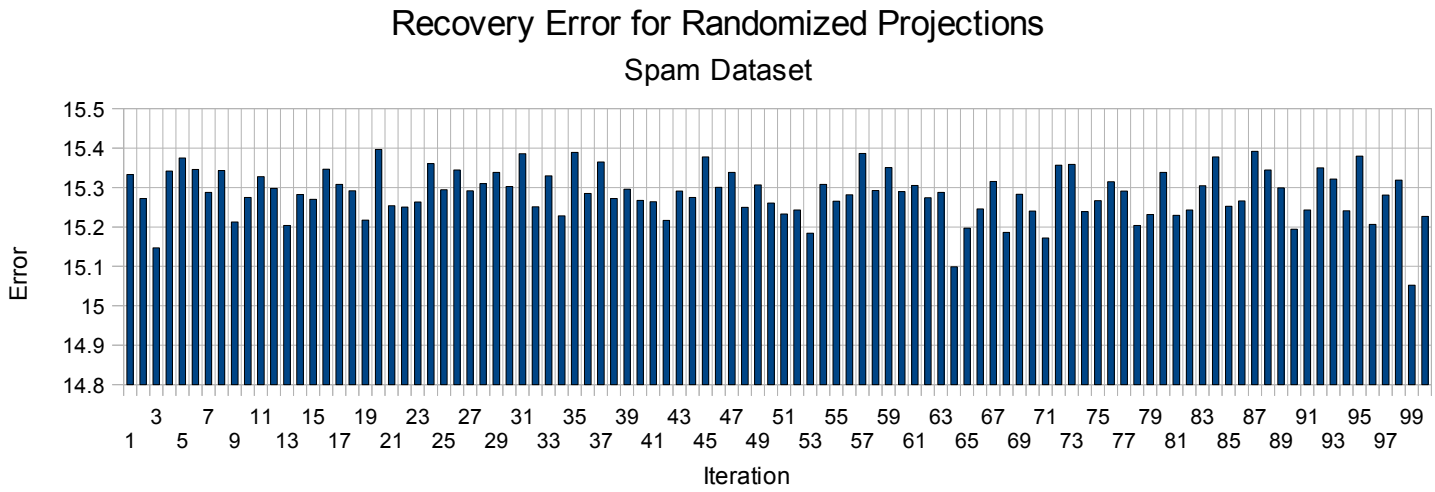


Figure 9: Recovery Errors over 100 iterations for Spam Dataset

The same analysis can also be done for the Soybean dataset also, but there is no real trend here to explore. One approach that can be taken though, is for a recovery, we can make multiple iterations and then choose to give either the average recovered input or the one with the least error.

Latent Semantic Analysis:

LSA typically finds a low rank approximation of the full data. The parameter it accepts is the proportion to indicate the desired coverage. This is the matrix rank to be used for data reduction. As this proportion decreases, there is a very rapid decline in the number of attributes required in the reduced dimension space.

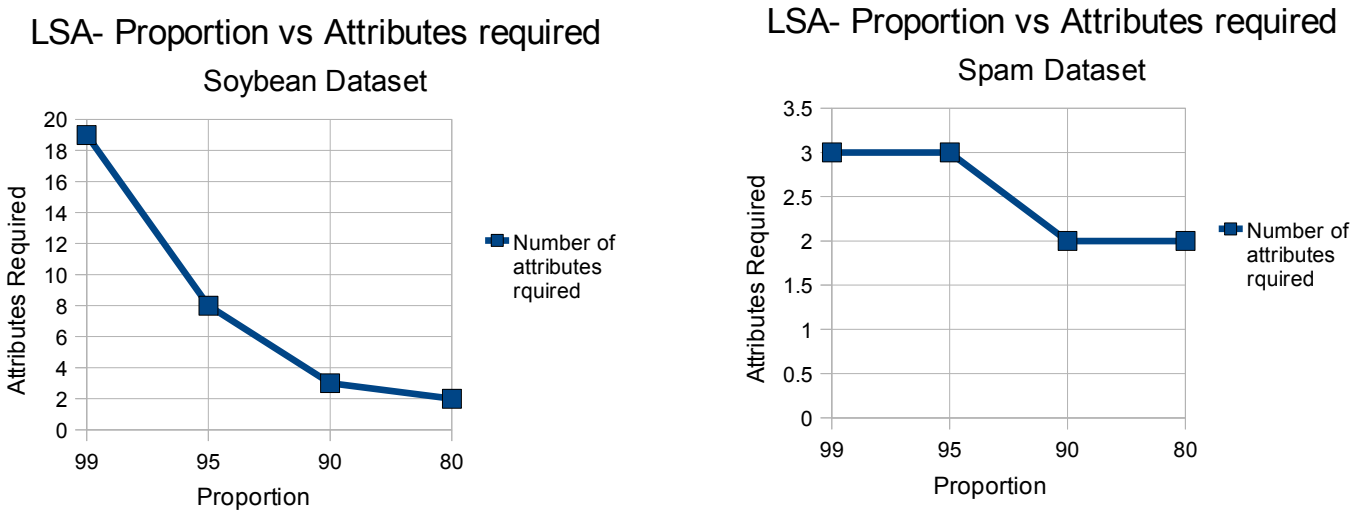


Fig. 10 Proportion vs Attributes Required

On the other hand, for Spam dataset, for the proportion of low rank matrix as high as 99, only 3 attributes are required. While this may sound like a very good thing to happen- dimensions being reduced from 58 to 3, however, how good a performance this gives can only be known once we are able to run the neural network on it.

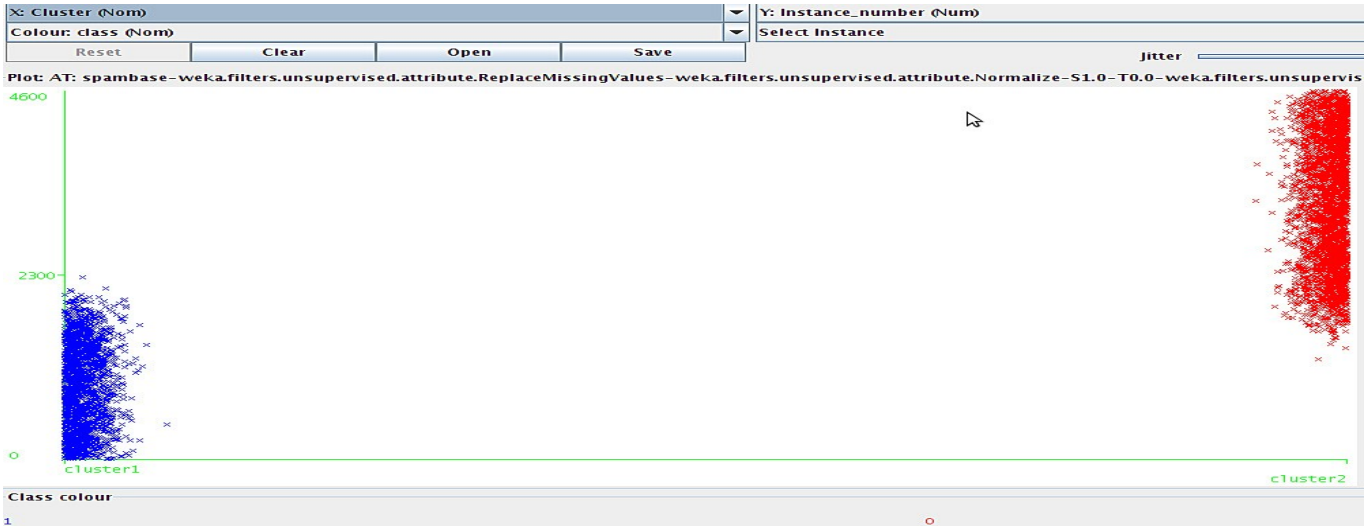
Clustering on Reduced Datasets:

To see if the nature of clustering changes on reduced datasets, I perform k-Means and EM on the reduced datasets produced.

K Means on Reduced Datasets:

Dataset Reduced by PCA (90% Variance Captured):

For a 90% variance capture, for the Spambase Dataset, the number of attributes comes down from 58 to 44. However, there seems to be no visible change in the clustering.



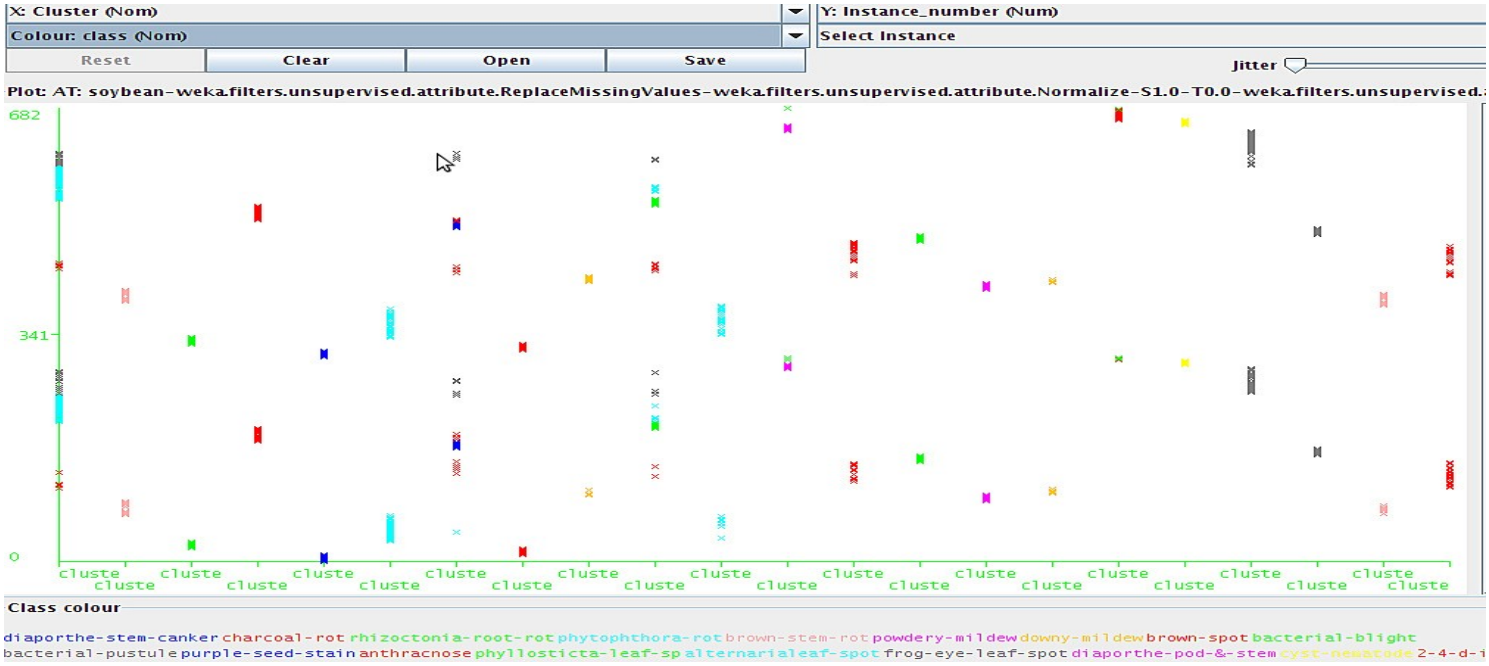


Figure 11 k-Means on Reduced Dataset

Comparing these figures with Fig 1(a) and Fig1(c), there is not much to choose between, especially for the spam dataset. In fact, I tried clusterings on LSA, ICA, RP reduced datasets as well, but the clustering looked the exact same. If we notice carefully though, there are subtle changes in the clusterings for the soybean dataset. Some clusters appear cleaner for the reduced dataset indicating possible elimination of noisy data during the process of dimensionality reduction.

EM on Reduced Datasets:

EM on reduced datasets has slightly more interesting results. One feature that differs straight away is the number of clusters suggested by using cross validation. For example, on the PCA-reduced data set above for spam, the suggested number of clusters
went down from 13 to 6 which is a lot closer to the number of classes present (2). Similarly, for the soy dataset, the predicted number of clusters goes down from 25 to 22 which is a lot closer to the number of classes (19). This seems to be a good sign.

The effect described for k-Means is even more pronounced in case of EM. I could observe greater number of clusters which did not have any instances from other classes which again is a good sign showing how clusters are tending to stick cohesively on basis of class. This characteristic is already captured by the k-means figure 11(b) and hence similar clusterings for EM are not displayed because of space restrictions.

Applying Neural Network on Reduced Datasets:

In particular, I choose the spambase dataset over the soybean dataset since it has no missing values

Figure 12 actually conveys a lot of information – it shows a clear trade off between how much of accuracy are we willing to sacrifice given an option of reducing the dimensions. This is not always a bad thing. Often there may be systems where computation costs are very high due to very high dimensionality of data. In such cases, it may be advisable to compromise(even this is not always necessarily the case) a bit on the amount of information captured while transforming to a lower dimensionality dataset. For example PCA with 44 principal components gives a classification accuracy of ~91% which is roughly the same as what the whole dataset with 58 attributes gave in assignment 1.

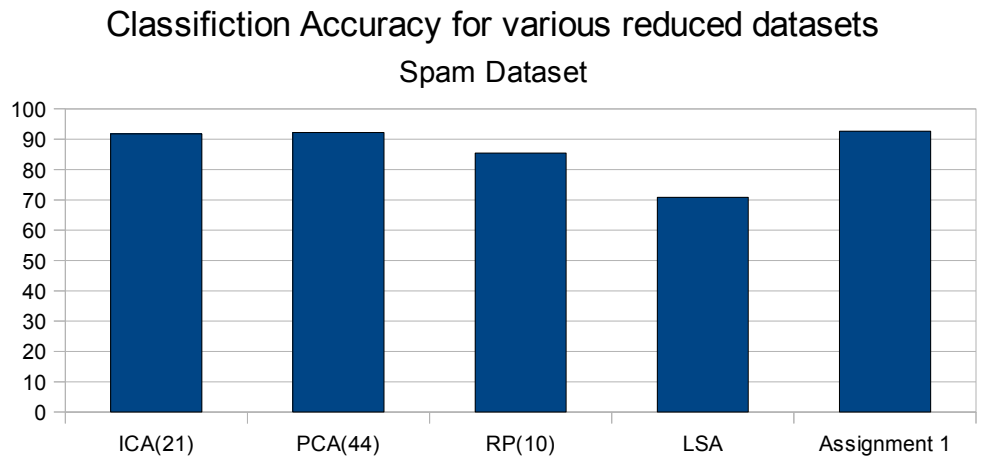


Fig 12 Performance of Neural Nets on reduced data

LSA gives the least accuracy but this is because it drastically cuts down the number of attributes- from 58 to 3! Tweaking the parameter values to strike a better balance may help get better performances from this algorithm.
Another very obvious trend seen while running this test is observing the wall clock time. Neural Net on LSA-reduced dataset finishes significantly faster than the original N dimensional dataset. As always techniques like boosting can be applied to further improve the accuracy if the system gives bad results.

Incorporating Clustering Information in the Dataset:

When we perform clustering on the reduced dataset, we can choose to utilize this new information about which cluster an instance belongs to. Now we can run two tests: only using the clustering information as the sole feature and training the neural network on it. If that performance is below par, we can augment the existing dataset with this new information and see if the accuracy increases.

When I use only a single attribute in the dataset- the cluster the instance belongs to, for the spam dataset reduced by PCA and clustered by EM, I got an accuracy of 67.28% which is not very good. But then again, this is an attribute we derived and it single handedly can classify data much better than chance. This naturally gives us enough motivation to try and augment the existing dataset with this information and then try to do classification. This strategy works wonderfully well and gives the best results I got in any of my experiments so far!! For the reduced spam dataset, I get a classification accuracy of 99.7827% (with 10 fold cross validation).

If the clustering scheme chosen is k-Means, the results are even more surprising. If only the clustering information Is used as the sole attribute in the dataset, even then the neural net classifies with 100% accuracy (using 10 fold cross validation). This should ideally leave us with no incentive of augmenting this information with the reduced dataset, but I want to confirm that the accuracy won't be adversely affected.

This gives us what can prove to be a very effective strategy- augmenting datasets with cluster values for improving accuracies.