CS7641: Assignment #3 - Unsupervised Learning and Dimensionality Reduction

Ajinkya Bagde
abagde3@gatech.edu

Abstract:
The purpose of this paper is to explore several Dimensionality Reduction algorithms - Principal Component Analysis (PCA), Independent Component Analysis (ICA), Randomized Projection, and Singular Value Decomposition (SVD). In addition, The clustering algorithms of K-Means and Expectation Maximization (EM) will be explored. Lastly, the results from both the Dimensionality Reduction algorithms and clustering algorithms will be passed through the Neural Network Learner from Assignment 1.

Overview of Datasets:

MNIST:

This is a well known algorithm in the ML space - used to train image processing systems, but also used for training/testing algorithms for ML applications - this is multi-class dataset. There are no 'discrete' features in this dataset (meaning features that are numbers, strings, etc that are understandable to humans) - a feature here is simply a pixel in a certain image. As the dataset is composed of 28x28 images (representing handwritten digits 0-9), it has 784 features - this will be an interesting concept in terms of dimensionality reduction for this assignment - what makes a pixel more 'important' than another, and how do results differ with each dimensionality reduction algorithm? For example, a certain algorithm might give more weight to pixels that are on the boundary of the handwritten digit, while for another that may not be as important.

Due to the size of the MNIST dataset (70,000 images), a 10% subset was chosen for the purposes of this report. The subset was randomly sampled to verify the class distribution would stay consistent

Spambase:

The purpose of this dataset is to help classify emails as spam, dependent on what attributes they have. Most attributes are certain keywords, but a few are also characters. Metrics wise, this dataset has ~4600 instances, with 57 features. This is also a binary classification dataset - meaning that the two categories are 'spam' and 'not spam'.

Part 1: Clustering

K-Means:

Spambase:

First to find the optimal number of clusters, the elbow method was used (Inertia vs Number of Clusters). Basically this method looks for a point or "elbow" where diminishing returns are no longer worth additional cost, shown below:

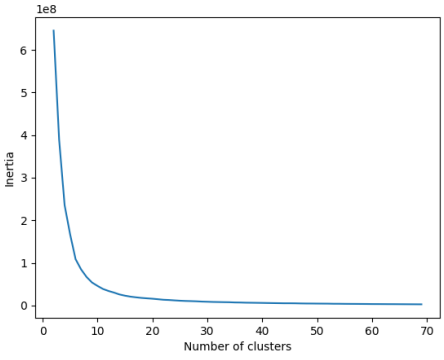

Figure1: Inertia vs # Clusters (K-means)

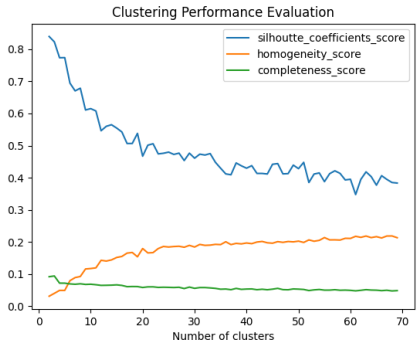

Figure2: Clustering Performance (K-means)

Since a clear "elbow" was not present in the inertia plot, different metrics were chosen in order to find optimal number of clusters - silhouette coefficients score, homogeneity score, and completeness score. The optimal number of clusters would be where all three of these scores are roughly 'maximized' - we see that happens ~8 clusters (which in turn, does sort of line up with the "elbow" in figure 1). In terms of classes vs clusters, we see a ~45% accuracy rate - this means that when the number of classes was equated with the number of clusters, they only agreed 45% of time. This can partially be attributed to the fact that the Spambase dataset had a high number of features for the relatively low amount of instances present in the dataset.

MNIST

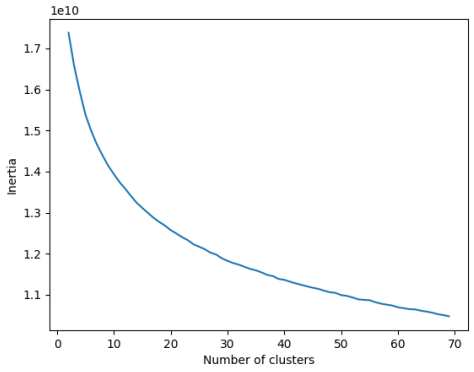

Figure3: Inertia vs # Clusters (K-means)

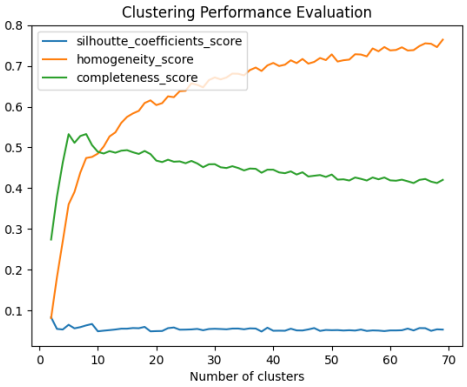

Figure4: Clustering Performance (K-means)

Even less of an “elbow” was present in the case of the MNIST dataset, so different metrics were chosen in order to find the optimal number of clusters. Here a value of ~20 clusters was chosen, as it maximizes the homogeneity score, while not providing too many diminishing returns for the completeness score. In terms of classes vs clusters, we see a ~25% accuracy rate - this means that when the number of classes was equated with the number of clusters, they only agreed 25% of time. This can partially be attributed to the fact that the Spambase dataset had a very high number of features.

Expectation Maximization:

For Expectation Maximization, just clustering performance metrics were chosen in order to find the optimal number of clusters. Following the logic discussed above to maximize these three scores, we see the optimal value is ~18 clusters for Spambase, and ~25 clusters for MNIST. In terms of classes vs clusters, we see a ~30% accuracy rate for Spambase, and ~18% accuracy rate for MNIST. Accuracy rates are low because of the same reasons stated for K-means - a high number of features for both datasets

Figure5: Clustering Performance (Spambase)

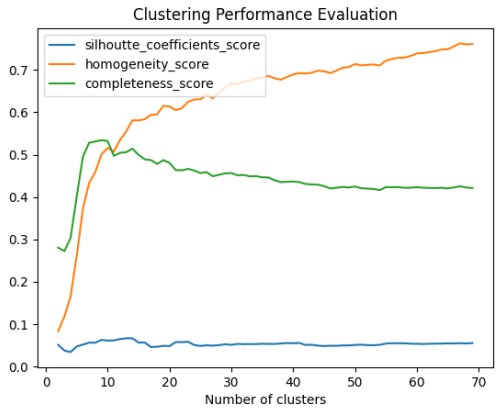

Figure6: Clustering Performance (MNIST)

It’s important to note that K-means is a form of “hard clustering” - each item is only assigned to a single cluster. In contrast, EM is a form of “soft clustering” - where an item can belong to several clusters, with a different weight in each cluster. In our case for both datasets, we see that K-means clustering returns favorable results with fewer number of clusters needed, as well as marginally better accuracy values for the classes vs clusters comparison. This could be because both our datasets are relatively large, and K-means scales well with large datasets.

<u>Part 2: Dimensionality Reduction:</u>

Spambase

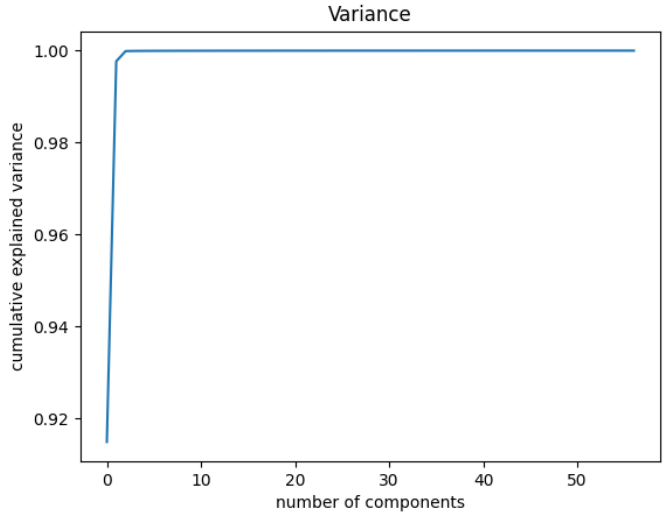

Figure7: Variance vs # Components (PCA)

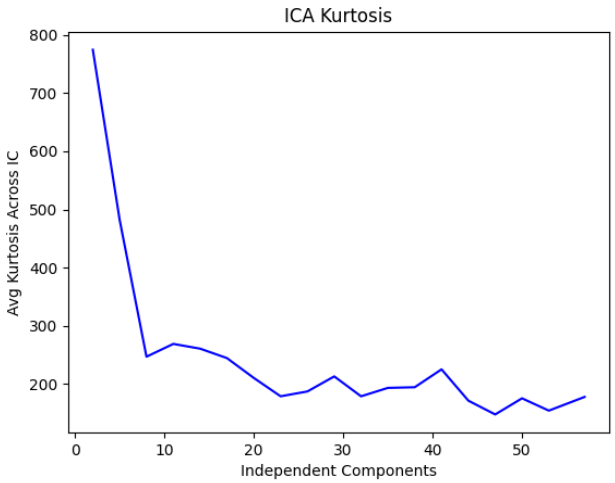

Figure8: Kurtosis vs # Components (ICA)

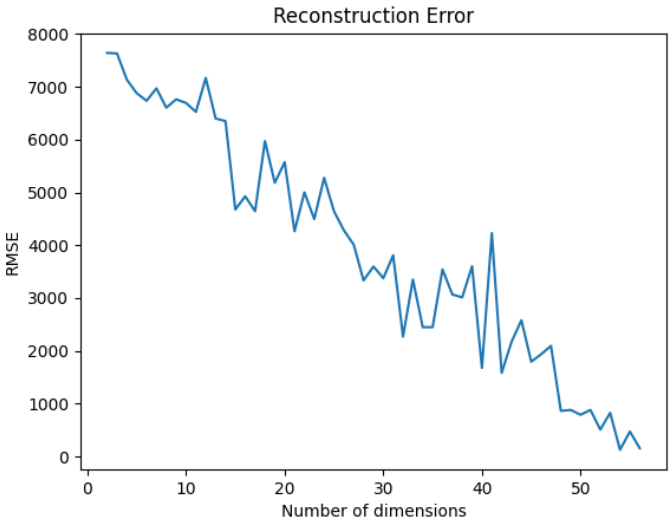

Figure9: Reconstruction Error vs # Dimensions (RP)

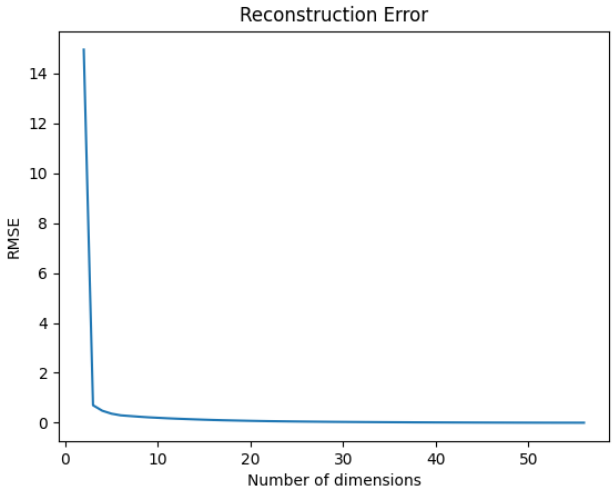

Figure10: Reconstruction Error vs # Components (SVD)

MNIST

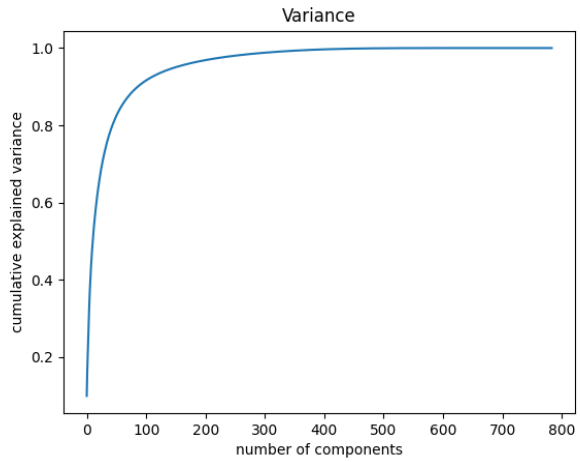

Figure11: Variance vs # Components (PCA)

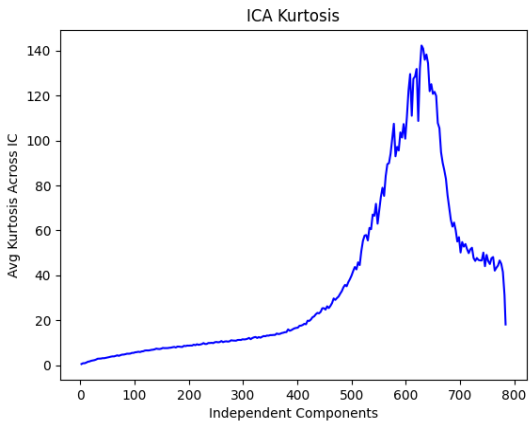

Figure12: Kurtosis vs # Components (ICA)

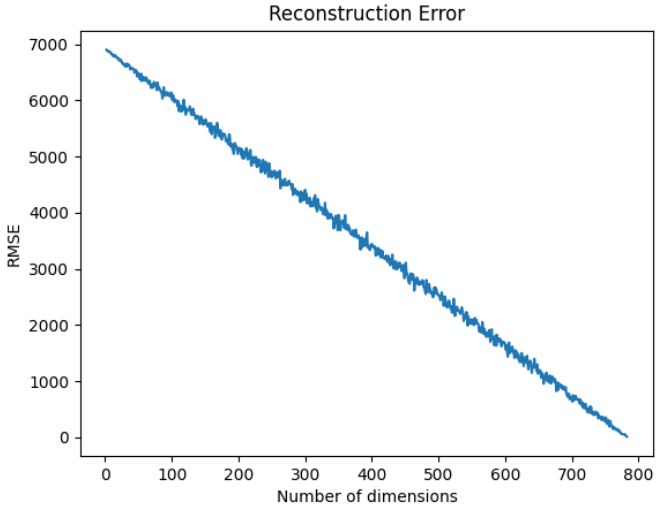

Figure13: Reconstruction Error vs # Dimensions (RP)

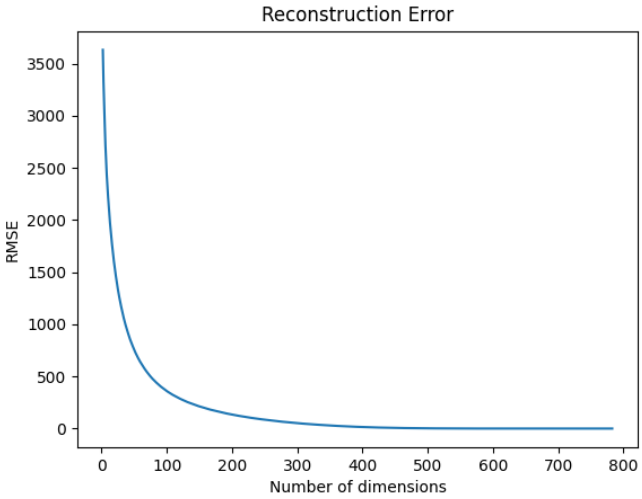

Figure14: Reconstruction Error vs # Components (SVD)

PCA Analysis

PCA performs dimensionality reduction by selecting components with the highest variance - it does this through eigenvalue decomposition of the data covariance matrix (a relatively expensive task).

Regarding Spambase, looking at Figure 7 above, we see that ~99.8% of the variance is captured in ~8 components for PCA - PCA chooses the components with highest variance. This means that we can get a really good representation of the full dataset by using just 8 components (as opposed to the 57 features in the original dataset). The first three components are plotted below (Fig 15):

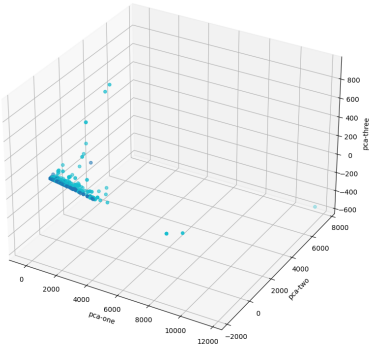

Figure15: 3-Component Plot (PCA, Spambase)

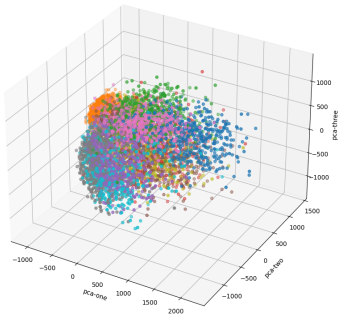

Figure16: 3-Component Plot (PCA, MNIST)

We don't really see three distinct clusters here, but this could be because these themselves don't have high enough variance to visualize the separation.

Regarding MNIST, looking at Figure 11 above, we see that ~95% of the variance is captured in ~150 components for PCA. This means that we can get a decent good representation of the full dataset by using just 150 components (as opposed to the 784 features in the original dataset). The first three components are plotted above (Fig 16) - we see that there is somewhat of a separation between the different groups/labels, but not in three distinct clusters. Again, this could be because these first three components by themselves don't have enough enough variance to visualize the separation (which lines up from Figure 11, where 3 components only captures ~40% of the variance)

ICA Analysis

For this portion, we are looking at the number of components that maximize the *statistical independence* between said components. That is to say - we want to be as anti-gaussian as possible in our distribution, which means we are plotting the statistical independence (kurtosis) between components.

Regarding Spambase, looking at Figure 8 above, we see that maximum kurtosis is captured in ~5 components. The kurtosis decreases as the number of components increases, which means that in the case of this dataset, only 5 components are needed to maximize statistical independence. The first three components are plotted below (Fig 17):

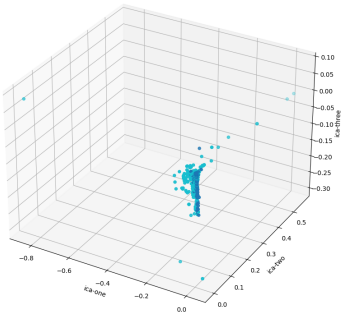

Figure16: 3-Component Plot (ICA, Spambase)

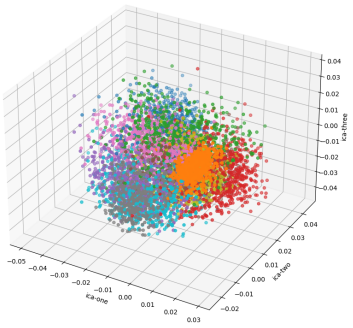

Figure17: 3-Component Plot (ICA, MNIST)

Regarding MNIST, looking at Figure 12 above, we see that maximum kurtosis is captured in ~650 components. In this case, the kurtosis increases from 0 → ~650 components, and then decreases pretty sharply. This means that after a certain point, adding components provides diminishing returns in terms of statistical independence. The first 3 components are plotted above (Fig 17). In both of the 3-Components plots, the data is seen to have a better separation than when compared to PCA.

Randomized Projections Analysis

For MNIST, this chart is extremely linear (Fig 13) - the more components/dimensions we include, the less error we get. In this case, there is no ideal number of components due to the linearity. For Spambase, it is still decently linear (Fig 9), but looking for sudden ‘dips’, we can pick out 12 components (RSME only dips to that level again at 22 components) and 30 components (RSME only dips to that level again at 40 components). Which one we pick is dependent on how much processing time we are comfortable with, since more components will take more time.

Singular Value Decomposition Analysis

This is very similar to PCA, but instead of using the covariance matrix, SVD uses the centered data matrix. In addition, PCA selects the components with the highest variance, and SVD picks components with highest singular values. Looking at Figures 10 and 14, we see that they are very similar to Figures 7 and 11 - RSME for SVD is minimized at very similar numbers of dimensions/components where variance for SVD was maximized.

<u>Part 3: Clustering (Post Dimensionality Reduction):</u>

PCA Analysis:
In terms of clustering performance, the homogeneity and completeness scores are roughly the same post dimensionality reduction, but a large difference in the silhouette coefficients score was seen for the Spambase dataset, show below:

Figure18: Clustering Performance (EM, Spambase, post DR)

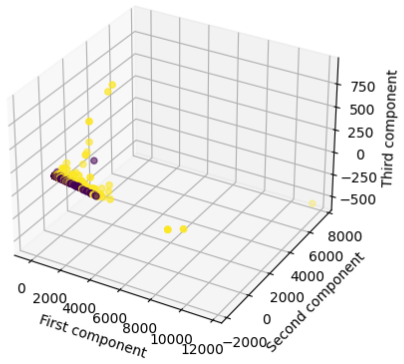

Figure19: 3-Component Plot (PCA, Spambase, post DR)

Compared to Figure 5, the dropoff for the silhouette coefficient scores for EM happens really rapidly - but the initial score for a lower amount of clusters is much higher. This means that for lower amounts of clusters, that specific datapoint was assigned to the correct cluster. This adds up, since the results from Part 1 showed that the overlap between classes and clusters (accuracy score) was 30%, but in this case was 38%. However, the plot of the primary three components looks somewhat similar, with slightly better clustering performance than the one shown in Figure 15. We would expect clusters to be more 'separated', since the accuracy score was better.

However, looking at the similar plots for K-means (instead of EM), not much difference is seen for the silhouette coefficient scores pre and post dimensionality reduction. This could be due to the K-means being a form of hard-clustering, vs EM being a form of soft-clustering. In the case of EM, it is possible that adjusting weights was enough to classify a datapoint to the correct cluster(s), vs in K-means, where the datapoint does or does not belong to a certain cluster.

ICA Analysis:

Similar to what was seen for PCA, the silhouette coefficient scores for the Spambase dataset post dimensionality reduction are better - in this case the results show more of a 'gap' than for PCA.

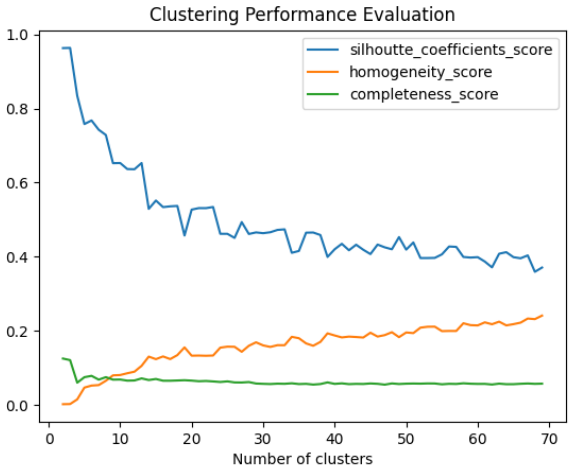

Figure20: Clustering Performance (K-means, Spambase, post DR)

Figure21: Clustering Performance (EM, Spambase, post DR)

K-means performance is slightly better, but EM performance is much better. This is supported by the fact that ICA makes an attempt at maximizing the statistical independence between components - in the case with the dimensionality reduced dataset, this was very well done. However, for the MNIST dataset, the silhouette coefficient scores were only marginally better - this could be due to ICA not being able to maximize statistical independence between components because the dimensionality reduction was not done to the extent of the Spambase Dataset.

Randomized Projections Analysis

In this case, conducting Dimensionality Reduction with Randomized Projections did not achieve much - the results for MNIST for both K-Means and EM were roughly the same, and the EM results for Spambase are actually worse. This can be explained because it was so tough to pick an ideal number of components, due to the linearity of the Reconstruction Error vs number of

components. Given more time, I would have plotted the above clustering performance plots with respect to several variations of components, just to make sure nothing was being missed.

Singular Value Decomposition Analysis

Results for SVD were pretty similar to PCA - silhouette coefficients score for EM is higher. This adds up as they are very similar dimensionality reduction algorithms. Just the numerical approach is different - PCA uses the eigenvalue decomposition (covariance) matrix, and SVD uses the centered-data matrix. One of the differences here is that the results are better for both lower and higher numbers of clusters, shown below:

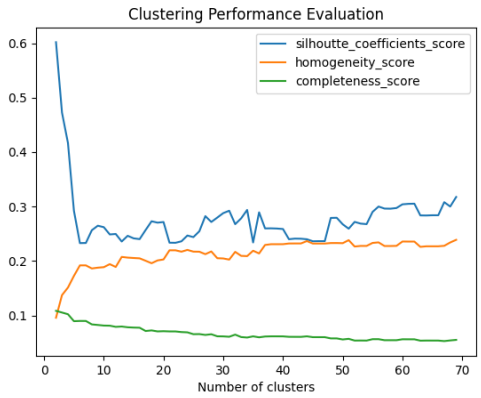

Figure22: Clustering Performance (EM, Spambase, post DR)

Compared to Figure 5, the initial score for a lower amount of clusters is much higher. This means that for lower amounts of clusters, that specific datapoint was assigned to the correct cluster, just like for PCA. This adds up, since the results from Part 1 showed that the overlap between classes and clusters (accuracy score) was 30%, but in this case was 33%. However, since the silhouette coefficients score is even better for higher numbers of clusters, this means that we should see better separation for even higher numbers of clusters.

Looking at the similar plots for K-means (instead of EM), not much difference is seen for the silhouette coefficient scores pre and post dimensionality reduction - this is probably for the same reasons as stated above in the PCA section.

Part 4: Neural Network Analysis (Post Dimensionality Reduction):

For this portion, the output of each Dimensionality Reduction algorithm was fed through our NNLearner from Assignment 1. The Spambase dataset was chosen. Below are the results:

	Assignment 1 Results (For comparison)	PCA	ICA	RP	SVD
Hidden Layer Size	(200,200)	(200)	(200,)	(200,)	(200,200)
Momentum Value	1.0	0.9	1.0	0.8	0.8
Accuracy (Tuned)	0.768	.722	.682	.510	.643
Runtime	18:56	12:12	8:45	14:06	12:40

For all four DR Algorithms, it was seen that we had slightly higher bias and lower variance compared to the results from Assignment 1. This can be explained by the fact that all these algorithms, to some capacity, only keep the “important” parts of the data. This reduces accuracy a bit (accounting for higher bias). Shown below learning curves for PCA vs those in Assignment1:

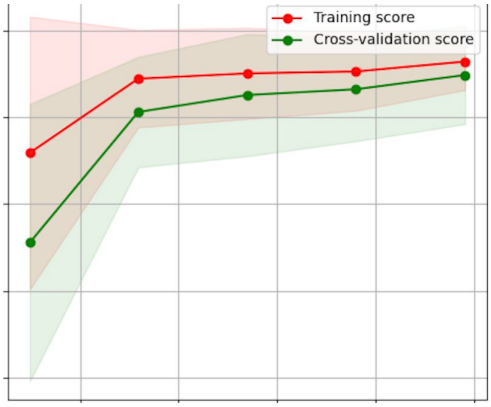

Figure23: Spambase LC

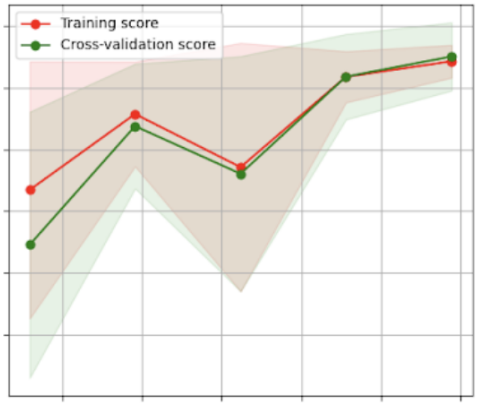

Figure24: Spambase LC (After DR)

PCA Analysis:
For PCA, 8 components were chosen because that was seen to capture 99.8% of the variance - and this reduced dataset was run through the NNLearner. The runtime was around 25% better than the original dataset, which is in line with the reduction in features. In addition, it was interesting to see that this time around a (200) Hidden Layer Size was found to be best by GridSearch - especially since a single layer of (200,200) was chosen for the original dataset, it seems that Dimensionality Reduction changed the complexity of the decision boundary so much as to take out a layer. This could have to do less layers needed to represent the now less features present in the Dimensionality Reduced dataset. Lastly, the accuracy here was ~72%, so only ~5% less than the original dataset - however given that we captured nearly 100% of the variance, I would have expected these to be almost identical.

ICA Analysis:
For ICA, 5 components were chosen because that was seen to reduce the reconstruction error within 90% of the maximum- and this reduced dataset was run through the NNLearner. The runtime was around 50% better than the original dataset, which is in line with how few components we selected. Again, it was seen that a single layer of (200) nodes was optimal for this problem, for the same reasons as stated above. Lastly, the accuracy here was ~68%, so only ~8% less than the original dataset - if more experiments were ran, I would have generated more learning curves with 6-10 components (referring to Figure 8) - and expected to see a huge drop off in accuracy, just to make sure we were on the right track.

Randomized Projections Analysis:
For RP, 12 components were chosen because that was the location of one of the bigger ‘dips’ in the RSME chart (refer to Figure 9). The runtime was around 20% better than the original dataset, which is in line with how few components we selected. Again, it was seen that a single layer of (200) nodes was optimal for this problem, for the same reasons as stated above. Lastly, the accuracy here was ~51%, so ~25% less than the original dataset. This is quite a big drop off compared to the ICA/PCA results above - if more experiments were run, I would have generated more learning curves at components numbers 30 and 48 (referring to Figure 9) - and expected to see possibly greater accuracy, and the expense of runtime. It is important to note that while Randomized Projection is the least computationally intensive of the Dimensionality Reduction algorithms (standalone, before data is passed to NNLearner), it’s performance is much worse. This could be in part because the ideal number of dimensions is rather difficult to pick, when compared to the rest of the algorithms.

SVD Analysis:
Since SVD is comparable to PCA in terms of how both approach Dimensional Reduction, results were pretty similar. Using 5 components (referring to Figure 10), runtime was 23% better than the original dataset, and accuracy was ~11% less. However, this time a double layer of (200,200) was chosen for the Hidden Layer Size. Not exactly sure why the deviance from PCA, but could have to do with how features were reduced.

Part 5: Neural Network Analysis (Post Clustering):

For this portion, the clustering algorithms were applied to the Dimensionality Reduced dataset, and that data was fed to our NNLearner from Assignment 1. The Spambase dataset was chosen. Below are the results:

	Assignment 1 Results (For comparison)	K-Means	EM
Hidden Layer Size	(200,200)	(200,)	(200,)
Momentum Value	1.0	0.9	1.0
Accuracy (Tuned)	0.768	.643	.612

For both K-Means and EM, 8 clusters were chosen to get the above results. The learning curves were very similar to the ones generated in Part 4, with slightly even lower bias, and same reduced variance when compared to the results from Assignment 1. In the case of the Spambase dataset, it was seen that clustering in place of Dimensionality Reduction did not prove to get better results.

In general for parts 4 and 5, the NNlearner ran faster, simply because there was less data to process.