Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

Introduction

In Part 1 of this assignment, clustering algorithms in Python sklearn are applied to the datasets from Homework 1. In part 2, dimensionality reduction algorithms are applied to both datasets to select the relevant features. In step three, clustering will be reapplied post-dimensionality reduction processing. From this step, the dimensionality reductions are run on the restaurant dataset, and the neural network learner from assignment 1 is compared to a neural network on the restaurant dataset to which dimensionality reduction has already been applied. Finally, the dataset from step three will also be used to train a neural network, which will then be compared to the results of both step four and the original neural network.

The "LSOA" dataset. Size: 6 columns, 1,000 features

This Kaggle csv (London_crime_by_lsoa) contained the LSOA, what seems to be the British equivalent of a zip code of the city, the borough, the category and description of the crime, the month and year of the crime, and the number of total incidents in that month. The problem I chose to solve for this project was to use the lsoa, borough, year, and month to decide which general classification, or category, the crime went to.

The "Restaurant" dataset. Size: 18 columns, 131 features

The restaurant dataset was a list of data for Mexican restaurants, with general location data down to the latitude and longitude, but also franchise, price, environment and other service information. Problems that this data could solve range from determining handicap accessibility based on an area, determining the clustering of certain types of restaurants in Mexico, and determining franchise information based on the region/latitude.

Part 1: Clustering Algorithms

Clustering algorithms are methods for unsupervised learning where objects are put into groups based on the distances between them, where the outputs are the group in which they belong. In these implementations, since both datasets were transformed to numerical data, labelling was not a relevant factor for analysis, although the results did show natural alignments of the groups the data would fit in given visual analysis.

k-means clustering

The k-means algorithm minimizes the within-cluster sum of squares dividing a set of N samples into clusters defined by their means, where K centers are chosen at random and the surrounding points are clustered around the K. In this implementation, the initial cluster centers were selected using sklearn's "k-means++" setting, which chose clusters in an optimized manner rather than randomly for efficient computation. The measure of similarity for this implementation was calculated using the variance of the group, where variance was calculated using the sum of squares of each data point versus the mean of each cluster. In this analysis silhouette score is used as a proxy for sum of squared error, explained below, are used to optimize the k selected.

k-means Restaurants Dataset

K's in range 2 to 12 were tested to select the ultimate k used in this problem using measures of precision, loss, and silhouette analysis, which is a method for evaluating the optimal k that sklearn offers by maximizing the distance between a data point and a neighboring cluster. Using this standard, K= 2 clusters was the grouping that maximized the silhouette metric, which indicated it was best-performing on the dataset, and that the original assumption of four main groups was incorrect. The clustering into two groups on the original classification problem makes perfect sense, as the original problem is a binary classification. The fact that this data has been previously used to create a binary classification with high accuracy makes sense given that it is easily and best separated into two clusters. My best performing model took .1363 seconds to compute, and as Fig 1 shows, performed quickly on a variety of n's, proving this method performed efficiently this dataset of 138 rows and 18 columns.

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

Figure 1: Silhouette Score, SSE vs Cluster Number

# of Clusters	Average Silhouette Score	Computation Time
2	.09288	.1363 sec
3	.08938	.1482 sec
4	.08542	.1442 sec
6	.04388	.2072 sec
8	.05151	.2319 sec
12	.04239	.2649 sec

Figure 2: Learning Curve on k-Means

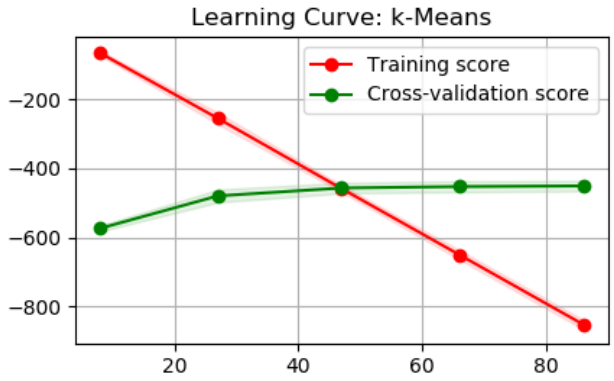
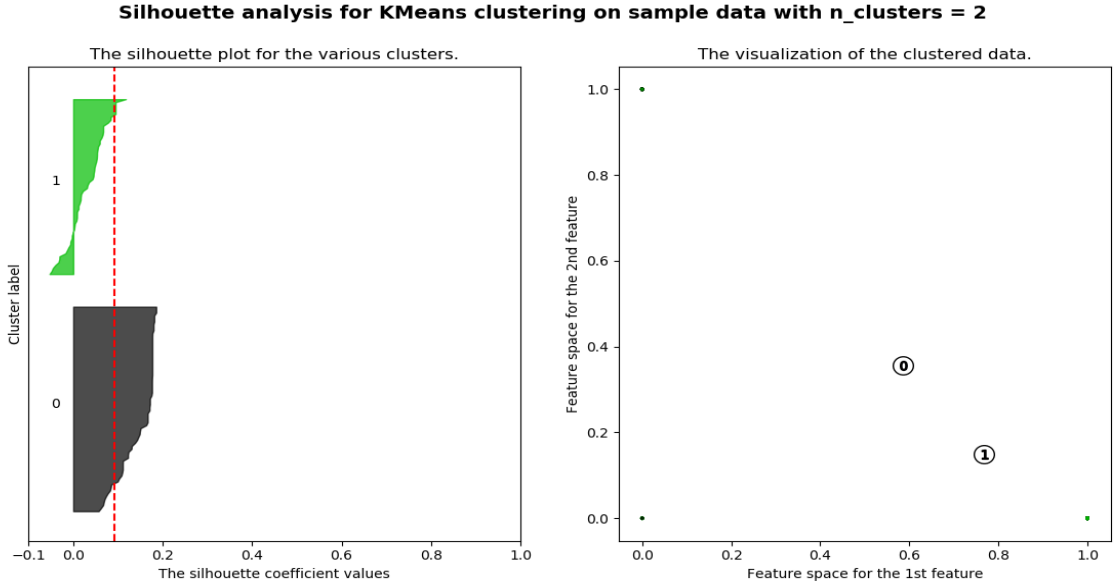


Figure 3: Silhouette analysis output for 2 clusters



As Figure 3 demonstrates, the clusters resulting from this process were labeled strictly with either [0,1] or [1,0], and centroids for the two clusters were approximately [.8, .2], and [.6, .4]. The low silhouette values in the above graphs and these two groupings can be explained by looking at the data- this was mostly qualitative data, so I transformed it to series of zeroes and ones. I think this processing and resulting the lack of diversity in my data but the magnitude of data points causes the silhouette function to be unable to definitively find an objectively good fit for this function, although 2 clusters was the best fit available. This problem is further exacerbated in the LSOA dataset below, which had very low silhouette values because the analysis was performed using 20,000 data points.

LSOA Dataset

As the figures above show, the performance of k-means on the LSOA analysis demonstrated similar trends as on the restaurants dataset, but again, certain problems in this approach were exacerbated by the size of the dataset. At 20,000 points, this method became very computationally expensive with the increased size, taking over ten minutes to cluster with n=6, so the dataset was reduced to a sample of 1,000, and k-means was run again with the same results. Using the silhouette method, k = 4 was selected as the optimal clustering. Looking over the data, and based on the logic from my decision tree from assignment part 1, this is a feasible value for k.

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

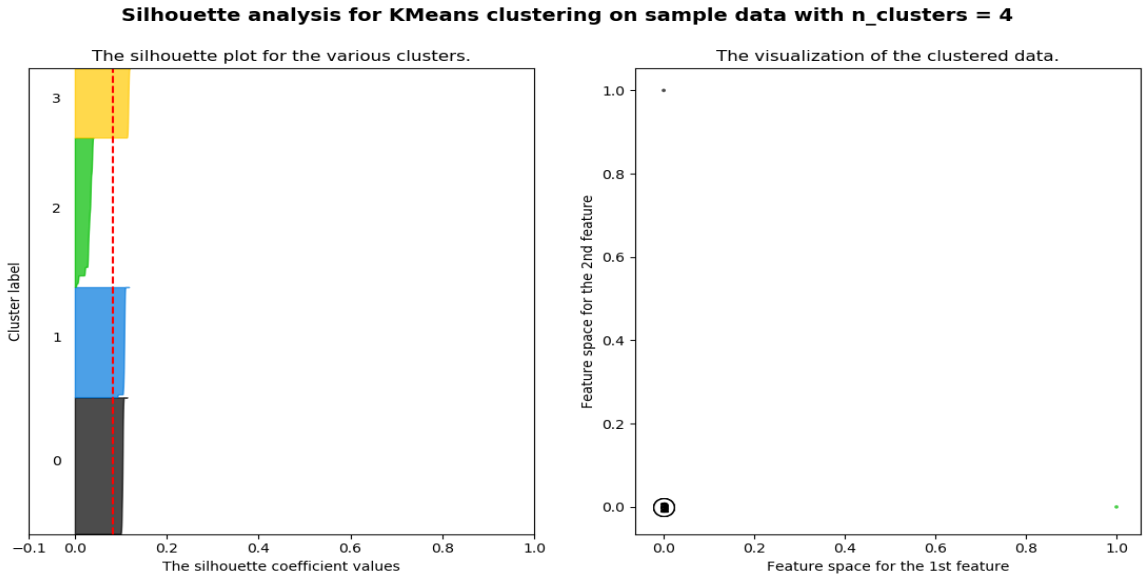
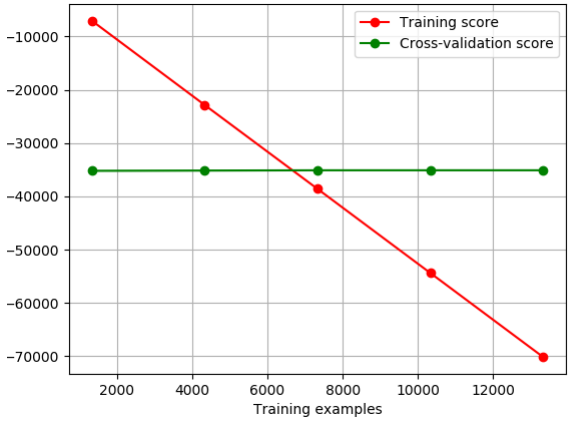


Figure 4: Learning Curve on kMeans

Figures 5, 6: Silhouette Score, vs. Number of Clusters



# of Clusters	Average Silhouette Score
2	.04715
4	.08266
6	.10347
8	.058037

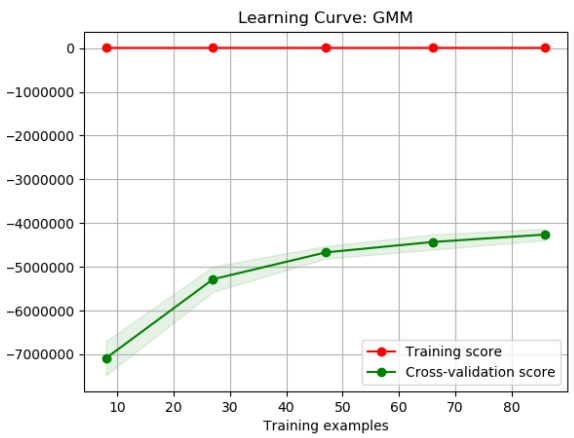
Expectation Maximization

The expectation maximization algorithm allows a process to identify the source of a data point through a probabilistic approach, which in this implementation takes the form of Gaussian mixture models. The distance metric used in this implementation is log likelihood, which in this implementation is calculated at the lower bound of the best fit of the EM.

Expectation Maximization Restaurants Dataset

Figure 7: Component Number vs. Log Likelihood

Figure 8: Cross Validation for GMM



Number of Components	Log Likelihood	Time (s)	Converged
2	4514.2148	1.47 s	True
5	4698.5689	2.76 s	True
7	4775.5406	3.97 s	True
10	4810.227	5.97 s	True

The data above demonstrate the fast convergence time of the GaussianMixture algorithm in sklearn, and readily converged on every test despite a variety of inputs. These results were very interesting on my data, as my implementation seemed to create a very small cluster, in a small range, which is not the result that expectation maximization resulted on this data.

Expectation Maximization LSOA Dataset

Figure 9: Component Number vs Log Likelihood

Number of Components	Log Likelihood	Time (s)	Converged
2	26756.3	93 s	True
4	28139.0	181 s	True
6	28543.8	300 s	True
10	28661.9	8023 s	True

The data in Figure 9 oppose the result of the k-means- because the log likelihood keeps increasing, it seems that it would converge around a little higher than 10 components, however, the convergence time was so high that clustering in this scenario is computationally less effective than not clustering at all, and doesn't effectively identify the useful clusters because of the high number of components selected.

Comparison of Expectation Maximization and k-Means

K-means was a more effective strategy for both datasets for clustering, as evidenced by the lower computation time, and lower error rate, although this could also be due to my implementations of both algorithms. For k-means, silhouette analysis was effective methods of choosing components, and effectively demonstrated that the problems in part 1 and part 2 were all relatively weak learners. The results of expectation maximization corroborated these results for the restaurant dataset, but diverged on the result of the LSOA dataset, choosing to segment the population into two groups as opposed to the k-means result of 6. To optimize the performance of LSOA, additional testing with different implementations of Gaussian mixture models, would need to be done. To optimize the performance of kmeans, I would like to re-run the tests with a larger dataset, as that would better allow me to visualize the features and their scores.

Because both problems were chosen to be geared towards classification of only a few parts- the first problem was binary and the LSOA problem had three to four categories the data could be classified into, the clustering seemed to reflect the problem. Because it's likely that the results were very influenced by the problem statement in this way, and hopefully will serve to reinforce the trends seen in the data more forcefully when applied to the classification problems.

Part 2: Dimensionality Reduction Algorithms

Dimensionality, or feature, reduction, is a process through which the number of variables, or features, is lessened, and the most important features are extracted. In this implementation, vectors are projected to a lower dimensional space through a. In this section, the parameters and components for each algorithm are fine tuned. In later parts of this assessment, the data will be transformed using all dimensionality reduction algorithms using these tuned parameters.

Principal Components Analysis (PCA)

Principal components analysis is a technique for feature reduction that separates features by their orthogonal distance. The eigenvalues created through PCA account for the explained variance, and the first principal component has the largest possible variance, meaning that it accounts for the largest amount of variability in the data possible to visualize distance between populations.

The explained variance in this implementation is a proxy for both the distribution of eigenvalues and the significance of each variable – if a variable is a large contributor to the explained variance ratio, then it is a significant component. This implementation also orders the features by importance, which explains the results seen in explained variance and explained variance ratio in Figure 9. In this implementation, a higher PCA score is better, and this factor is used in conjunction with the explained variance to select the number of features.

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

PCA on Restaurants Dataset

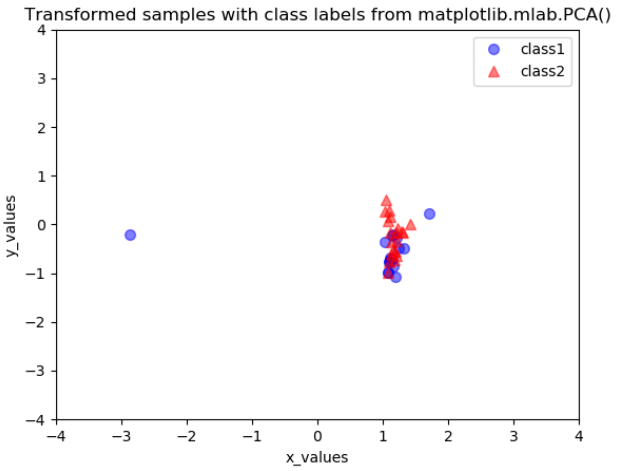
Figure 9: Analysis of Component Number on PCA outcome

Number of Components	Explained Variance	Explained Variance Ratio	PCA Score	Computation Time(s)
2	Var1 = 1.2655 Var2 = .57841	Var1=.111981 Var2= .051181	251.902	.355
3	Var1 = 1.2655 Var2 = .578417 Var3 = .535489	Var1 = .111981 Var2 = .051181 Var3 = .0473297	271.6	.209
4	Vars 1- 3 same as above Var 4 = .03969	Vars 1-3 same as above Var 4 = .035127	304.56	.308
6	Vars 1- 4 same as above Var 5 Var 6	Vars 1- 4 same as above Var 5 Var 6	318	.256

The data in Figure 9 demonstrate that past three significant variables, the explained variances of the features show low significance– variables 4, 5, 6 and onward contribute to a very small ratio of the explained variance. Due to the significance of the variables and low PCA score, this analysis shows that either two or three components are sufficient for analysis, and therefore, that the dataset can be reduced to two or three dimensions.

This result is mirrored in Figure 10, which gives a visualization of the two most significant clusters, selected because PCA optimizes for the lowest PCA score. These results are clustered around different features located at x=1 or 1 and y = 1 or 1, which makes sense because almost the entire dataset is binary.

Figure 10: Graph of PCA Reduced Outcome



PCA on LSOA Dataset

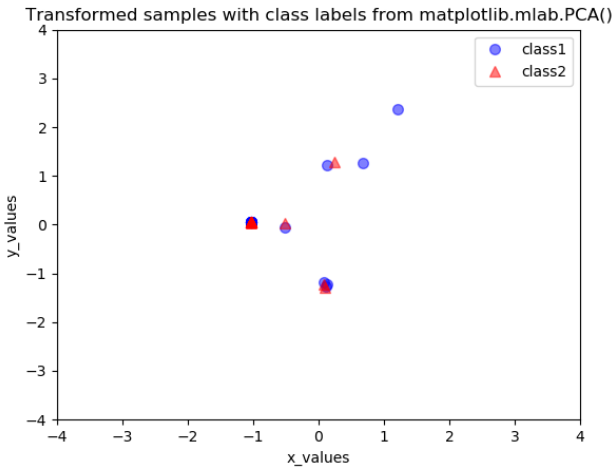
The data in figure 11 are a better representation than the data in figure 10 of the power of PCA, since there are 10 clusters for the 1,000-feature dataset. The resulting features of these data can be visualized in figure 12, where the individual data seems much more spread out into individual features. This is supported by the explained variances being more spread out amongst different features, suggesting that the LSOA dataset contains 5 or 6 weak learners that each contribute to the explained variance, whereas the restaurant dataset above only contains four or five important features.

Figure 11 : Analysis of Component Number on PCA outcome

Number of Components	Explained Variance	Explained Variance Ratio	PCA Score	Computation Time(s)
2	Var1 = .3234 Var2 = .2196	Var1=.0583 Var2= .0396	8031.3871	12.228
4	Vars 1- 2 same as above Var 3 = .1522 Var 4 = .1409	Vars 1-2 same as above Var 3 = .0275 Var 4 = .0254	8172.0281	9.974
6	Vars 1- 4 same as above Var 5 = .1312 Var 6 = .1277	Vars 1- 4 same as above Var 5 = .0236 Var 6 = .0230	8303.392	10.007
8	Vars 1-6 same as above Var 7= .1215 Var 8= .1196	Vars 1-6 same as above Var 7= .0219 Var 8= .0216	8433.0737	10.278

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

Figure 12 : Graph of PCA Reduced Outcome



Autoencoders

Autoencoders learn on an approximation function to the identity function for dimensionality reduction. They mirror the structure of a neural network, but with the objective of reconstructing or modelling its own given inputs, rather than for classification.

In this implementation, stochastic gradient descent is used, and the optimal learning rate was varied to determine the best performance. All autoencoders scored very high in terms of their testing score on an arbitrary neural network using the test and train data.

Autoencoders on Restaurants Dataset

This implementation used stochastic gradient descent to come to its conclusion. The optimal learning rate for this implementation was found by graphing different rates and finding the data with the lowest loss function on a basic neural net implementation. A rate of .01 was the implementation in Figure 13, which reduced the dimensionality of the vectors to a single vector. Computing the autoencoder alone took 25 seconds, which was longer than the original neural network took alone, so it will likely not increase efficiency on a larger scale model.

Figure 13: AE on Restaurants Dataset

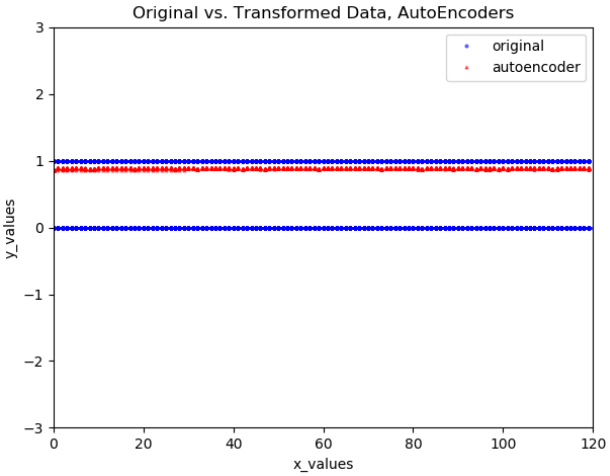
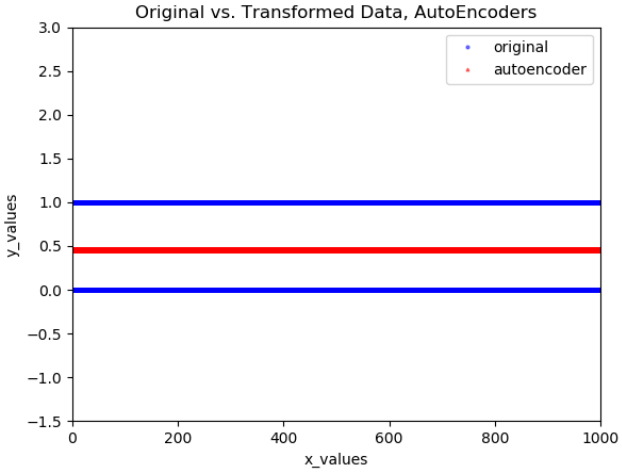


Figure 14: AE on LSOA Dataset



The implementation in figure 14 was computationally efficient, but did not produce a result that was particularly enlightening, as represented was the mean of the x value and y value as a single vector.

The model in Figure 14 took 554 seconds to compute, which was longer than the neural network took to train alone, and although the test accuracy was higher, the fact that the learning rate was .1 and the test score was 100 indicated over-fitting and likely a large amount of loss. This is likely because, for both datasets, this implementation took sparse vectors and made them almost entirely uniform and not sparse, which reduced the dimensionality but also caused significant loss in both datasets. This is especially prominent in Figure 14, where the dimensionality seems to have been reduced to a single feature that is .5 for all 1000 data points.

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

Randomized Projections

Randomized projection reduce dimensionality by trading accuracy for variance, thereby allowing the model to compress. This implementation of randomized projections used Gaussian random projections to project points in a higher dimension space to a representation in a lower space while preserving the distances between them as a function of 1/components, or the inverse of the number of components. Variance between Gaussian trials for both datasets was negligible, with accuracy measures differing in thousandths of a percent, and the number of components in each random projection was varied to arrive at an accuracy score on a simple SVM model implemented just for scoring the model.

RP on Restaurants Dataset

Figure 15: Transformed Data from RP

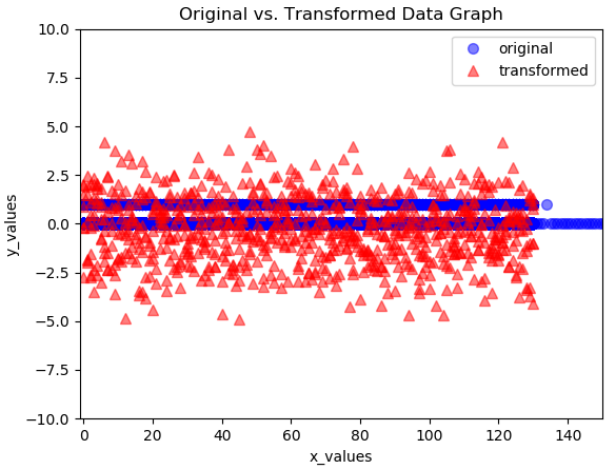


Figure 16: Transformed Data at different components, Restaurants

Random Projection Component #	Accuracy Score on SVM	Time(s)
none	.97	.053
Comp = 2	.96	.018
Comp = 3	**.98**	**.042**
Comp = 4	.93	.043
Comp = 5	.92	.045
Comp = 6	.89	.046

The transformed dataset is shown in Figure 15, and a user can see that the randomized projections added more complexity to the dataset, which originally appeared to be a vector of zeroes and ones in this dimension. Creating a two-component random projection allowed for increased computational efficiency of the SVM, since convergence times for the model were decreased for the small dataset.

RP on LSOA Dataset

Figure 17: Transformed Data at different components

Random Projection Component #	Accuracy Score on SVM	Time(s)
none	.97	19342
Comp = 2	**.67**	**23003**
Comp = 3	.45	26432
Comp = 4	.34	33,452

The randomized projections themselves were not necessarily computationally inefficient, but the SVM was, and this inefficiency outpaced any benefit of dimensionality reduction as the complexity of the features increased. This test was largely unsuccessful, as the model show that only two components were necessary to model this dataset, which shows severe testing error, likely in the form of the number of components selected. I realized that these differ from the clustering algorithms, as component numbers can model the number of features in your dataset total, and not just the X values, so additional analysis would graph the spectrum of x_values from 1 to 1000 as RP component numbers and their respective scores using a less computationally expensive scoring model.

Part 3: Clustering after Dimensionality Reduction Algorithms

Principal Component Analysis

Figure 18: k-means clusters after dimensionality reduced with PCA

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

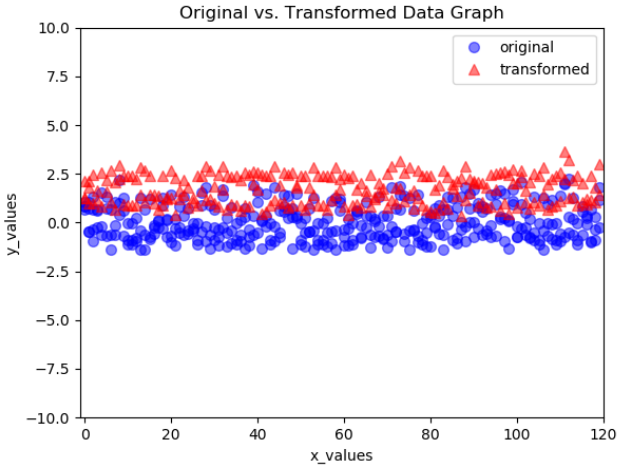
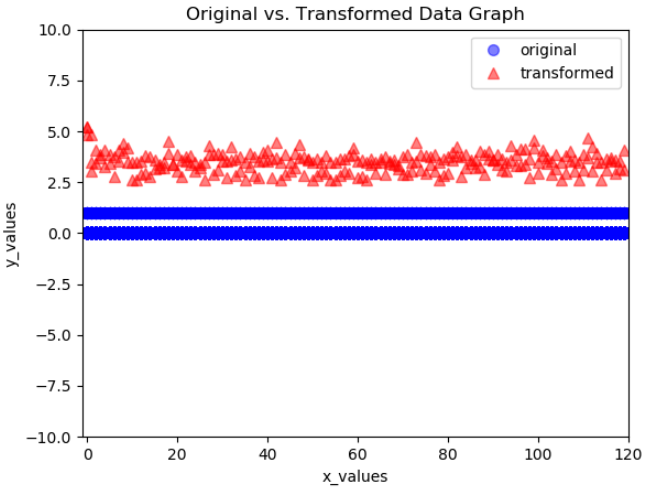


Figure 18 demonstrates that the transformed data clusters into 2 main points as opposed to the three originally seen in the original data. This originates from from the two principal components identified in part 1, and unlike the clusters in the original dataset, these are clustered horizontally, and range from .5 to 2.5.

Expectation maximization on PCA reduced data allowed my EM models to perform the same as they originally did, because the number of features were already reduced significantly, and therefore the scores went up. This implementation only reinforced the idea that there were two principal components, of 0 and 1 in this case, but EM still centered around 5.

Randomized Projections

Figure 19: k-means clusters after dimensionality reduced with RP

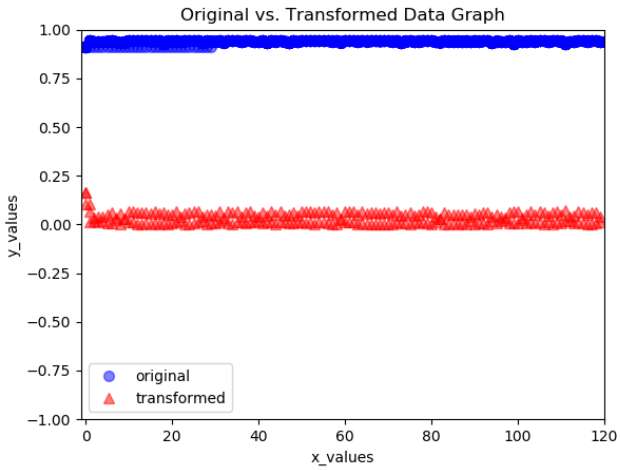


The K-means result also supports the notion that there are two clusters, as evidenced in the data labelled “original “- in this case, it was the RP-reduced data- because of the RP dimensionality reduction to 2 dimensions, but they aren’t as well defined as the two clusters in part 1 because of the fuzzy nature of random projections.

Expectation maximization also showed similar results as before, identifying one major component that was the mean of the two clusters, centered at about 3.5. This was an interesting result, since it did not reflect any of the data at all, and would likely be a very erroneous assumption if applied to a classification problem.

Autoencoders

Figure 20: k-means clusters after dimensionality reduced with AE



The RP dimensionality reduction also seemed to oversimplify the data, clustering the data in what seems to be one to two clusters in the 0-.25 range. This indicates a likely error with my autoencoder implementation, as this should not properly identify weights in higher dimensions, but perhaps the clusters are centered around the weights of the neural networks themselves for this implementation, and not the actual data. The expectation maximization for this problem identified one cluster at .05, which also seemed to oversimplify the data, and will produce a very erroneous result.

Sheena Ganju
Unsupervised Learning
CS 4641 Fall 2017

Part 4 and 5: Neural Network Learner on post-processed restaurant dataset

Figure 21 : Data Comparison of Various NN implementations

	Training Accuracy	Test Accuracy	Time Taken	Comments
i. Neural Network- Raw Data	100	94.44	5.779	Precision: 89.19 Loss: 191.881
ii.Neural Network - RP	100	95.412	5.554	Precision: 90.91 Loss: 160.45 Did not converge at 500 or 750 iterations, converged at 1500 iterations, accuracy exceeded raw data accuracy after convergence, but not before
iii.Neural Network - PCA Analysis	100	95.920	10.345	Precision: 92.421 Loss: 112.131
iv.Neural Network - Autoencoder	100	100	16.472	Precision: 92.93 Loss: 174.376
v. Part 4: Dimensionality Reduction Data – all algorithms	100	100	20.545	Precision: 94.432 Loss: 350.398
vi. Neural Network- Dimensionality and k-means	100	97.542	5.423	Precision: 97.342 Loss: 131.432
vii. Neural Network – Dimensionality and Expectation maximization	100	89. 412	200.3432	

NN with Dimensionality Reduction

Figure 20 part I was the baseline neural network on a dataset of 120 of the 138 features originally in the dataset used in assignment one, and had a test accuracy of 94.44 percent and took 5.779 sec, which are metrics to be compared to the rest of the section of this analysis. The first neural network I applied was Gaussian random projections to the neural network, which took .2 fewer seconds on this very small dataset, as demonstrated in fig. 21 part ii. This resulted in an improvement on the accuracy of the dataset, which increase to 95.34% after the NN could converge. This was initially counterintuitive because I would expect an increase in randomness to decrease the time to convergence- however, because RP's helped project the features of higher importance to the first dimension, the time complexity was ultimately reduced on the dataset of 120 features that was used.

The PCA analysis performed in figure 21 part iii with the ideal number of three components also improved the outcome of the neural network, but not significantly, and was computationally less efficient than the neural network by itself, so was not considered a significantly better method of training the neural net than the neural net alone, although it would be interesting to compare performance bigger dataset with fewer important features to see how the relative importance of the features affects accuracy.

In figure 21 part iv, autoencoders to train the weights of the MLP likely over trained the data or reduced the features too much on my restaurant dataset, which is why any implementation that combined neural networks and autoencoders had such a high testing accuracy. This is because the weights were already fitted to the training set for the neural network, so the MLP classifier had an even greater head start than for the other algorithms.

In part v, just for kicks, I applied Gaussian projections, then PCA analysis on the Gaussian data, and then created three layers of autoencoders before feeding the model into the neural network, which also took longer than the raw data to

converge, and likely overfit after the dimensionality was reduced to essentially one or two features, and resulted in a high measure of loss.

Clustering Algorithms treated as Dimensionality Reduction

When k-means was applied to neural network, the expected increase from the higher emphasis on the centroids/influential points on the outcome of my dataset, as supported by Figure 20. The addition of k-means with the optimal value of 3, as found before in this paper, allowed the three most important features to be given the most emphasis in the implementation, therefore slightly lessening the convergence time of the data and slightly increasing the accuracy.

When expectation maximization was applied to the neural network with the same parameters as part 2, where every data point was represented as .5, the accuracy of the NN decreased significantly, since this grossly oversimplified the data and the resulting classification was very erroneous. As stated before, this implementation and its parameters strongly need to be revisited.