Project 3: Unsupervised Learning

Tyler Bobik
Georgia Institute of Technology

Description of Problem 1

Data	Attributes	Instances	Classes
diabetes	8	768	2

Description of Problem 2

Data	Attributes	Instances	Classes
Spam	57	4601	2

1.0 Description of Problem #1 (Diabetes)

I found this data set on the UCI machine learning database labeled "The Pima Indians Diabetes Data Set" it is multivariate and it 8 contains continuous type attributes, 1 discrete class and 768 instances. The purpose of the data set is to predict if someone will get diabetes or not. This data set was interesting to me because it has a pretty low number of attributes and a low number of instances. Also, there seems to be noise in this data set and I would like to see how the different learning algorithms are affected by this I was curious to see which of the different classification algorithms are affected by this. preform on this data given those facts. Additionally, this kind of problem is interesting to me personally because I am in the Intro to Health Informatics class and we are developing a web app to help diagnose diabetes and thought I could gain information and maybe even apply a machine learning technique I learn here to it.

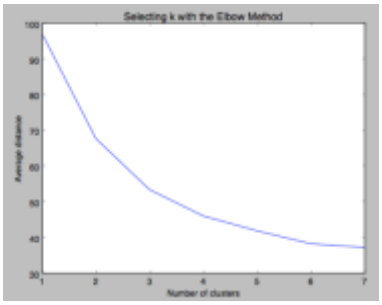
2.0 Description of Problem #1 (Spambase)

I found this data set on the UCI machine learning database labeled "Spambase Data Set" it is multivariate and it contains 57 continuous type attributes, 1 discrete class (spam or not) and 4601 instances. The purpose of the data set is to predict if an email is spam or not. Most of the attributes indicate whether a particular word or character was frequently occurring in the email. This data set was interesting to me because it has a pretty high number of attributes and a high number of instances. I was curious to see how different classification algorithms preform on this data given those facts.

Being new to machine learning, these two data sets were interesting me and drew my attention because, the diabetes data set had a low number of attributes and a low number of instances when compared to the spam dataset, where it has a high number of attributes and a high number of instances. With these two data sets together I was able discover many other interesting points when comparing the different unsupervised learning algorithms which I will point out throughout this paper. On the left I will display the results of different classifiers for the diabetes data set and on the right I will display the results of the spambase data set.

1.1 Finding Appropriate K for K-means

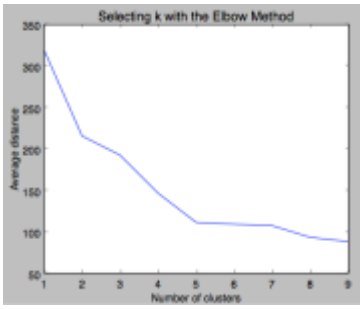
My first step in using K-means is to find an appropriate K value. For this I did not split my data up for train and test sets because I already knew what the classes are. I still ran a few tests to see if my results of K-means lined up. To first analyze this I used the elbow test which uses Euclidian distance to calculate the distance between each observation in the dataset and the cluster centroids. I then Summed the minimum distances across all observations, then divide the sum of this distances by the number of observations in the dataset. This was done from values K:1-7.



Here we can see a clear elbow at K=2 where the cost drop dramatically. I decided to run a few more tests to see what else I can derive.

2.1 Finding Appropriate K for K-means

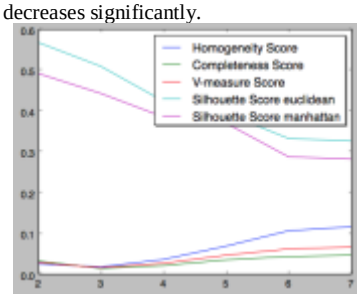
My first step in using K-means is to find an appropriate K value. For this I did not split my data up for train and test sets because I already knew what the classes are. I still ran a few tests to see if my results of K-means lined up. To first analyze this I used the elbow test which uses Euclidian distance to calculate the distance between each observation in the dataset and the cluster centroids. I then Summed the minimum distances across all observations, then divide the sum of this distances by the number of observations in the dataset. This was done from values K:1-9.



Here we can see a clear elbow at K=2 where the cost drop dramatically. I decided to run a few more tests to see what else I can derive.

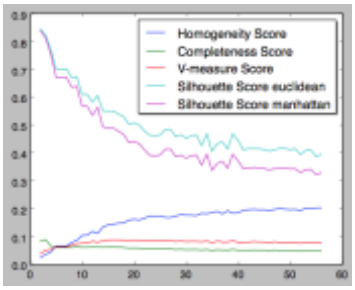
1.2 Finding Appropriate K for K-means

I also used extrinsic and intrinsic measures for evaluating the clusters. In this graph when comparing these scores in tandem with the elbow graph I believe the best K value to evaluate on is a K value of 2. I say this because the Silhouette score is maximized here and if we move to a K value of 4 we can see that the other measures that we would like to maximize only increase by a fraction while the Silhouette score decreases significantly.

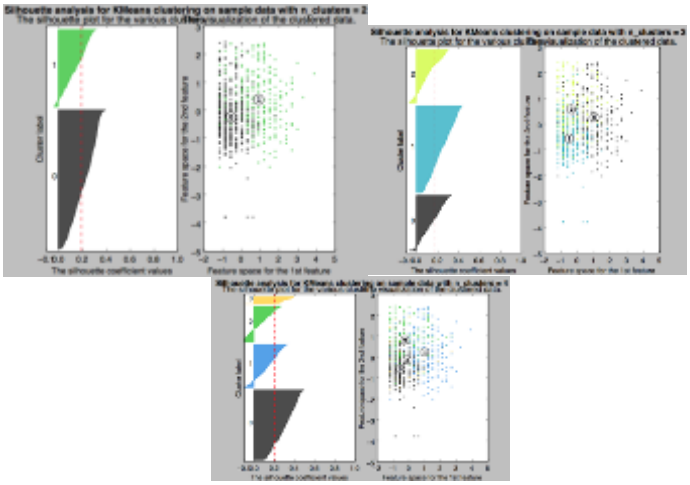


2.2 Finding Appropriate K for K-means

I decided to run these k-means with clusters 2-57 and this results demonstrated the most accurate number of clusters lay between 2 and 4. I also used extrinsic and intrinsic measures for evaluating the clusters.

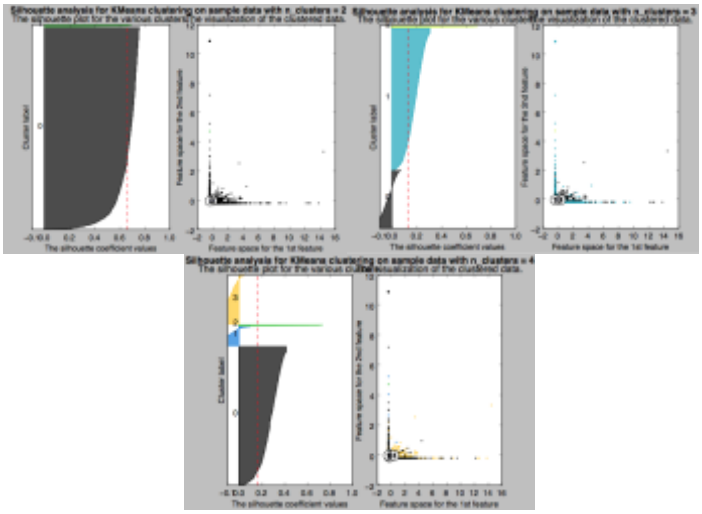


Here I used silhouette analysis to see what amount of clusters fit the data the best for K:2-4.



Here I believe that K=2 is still the best choice than the others because it has the least observations that are assigned to the wrong clusters and it seems to have a somewhat even distribution of the cluster size.

Here I used silhouette analysis to see what amount of clusters fit the data the best for K:2-4.



Here I believe that K=2 is still the best choice than the others because, although the cluster sizes are not homogeneous for K=2, none of the observations are assigned to the wrong cluster and both clusters for K=2 have higher than average silhouette scores.

The number of clusters. Surprisingly the silhouette score, the intrinsic metric, does a much better job helping us select the right number of clusters. Although I was only able to make this observation because of my prior domain knowledge that these datasets only have two classes. I wonder if there could be any other clusters in these datasets that could explain other class labels for different numbers of clusters. I will leave this question to solve at another time, it is not in the scope of this assignment.

1.3 Comparing against the True Labels

For K: 2

K	Percent Correctly Classified Given True Labels Vs. Predicted Labels
2	66.02%

For K = 2
Description of clusters:
Percent of positive diabetes diagnoses given true labels:
Cluster 1 30.18% with 603 instances
Cluster 2 52.12% with 165 instances

2.3 Comparing against the True Labels

For K: 2

K	Percent Correctly Classified Given True Labels Vs. Predicted Labels
2	63.59%

For K = 2
Description of clusters:
Percent of positive classified spam given true labels:
Cluster 1 37.28% with 4357 instances
Cluster 2 78.28% with 244 instances

I would say that clustering just using K means with both datasets does not a superb job with aligning the data with the true labels, but just because it got only 63.59% and 66.02% accuracy on the true labels does not mean it is bad at clustering, it is actually pretty impressive that it is able to get this high given no other information.

1.4 Expectation Maximum

I started by finding out what covariance type to use as one of the parameters to run the Gaussian Mixture Models(GMM) which is a probabilistic approach to clustering. I tested the number of clusters from 2-6 on covariance types: Spherical, Diagonal, Tied and Full to see what got the highest percentage of correct labels. In order to find a good number of clusters I decided to run the Silhouette score from 2 – 6 clusters.

# of Clusters	Silhouette score **Spherical**	Silhouette score **Diagonal**	Silhouette score **Tied**	Silhouette score **Full**
2	.3592	.4934	.3437	.465
3	.4713	.3385	.2031	.3567
4	.2213	.2174	.3802	.2565
5	.1966	.3854	.2539	.3802
6	.1575	.2161	.1041	.3437

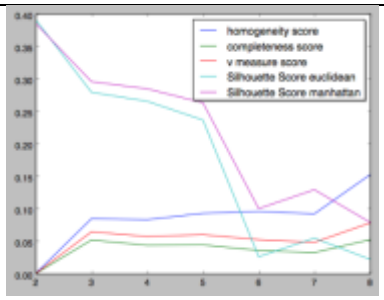
We can see that the covariance type "Diagonal" generated the highest score when the clusters equal two, so I will explore that a bit more.
What is very interesting from these scores is that if I did not know that from prior domain knowledge that the correct number of clusters is two, I might have thought that with spherical that the correct number would be 3. Although I probably would not have chosen that because Diagonally gives a higher percent correct when the clusters equal 2. I next, decided to test diagonal on a few more metrics to see what I uncover.

2.4 Expectation Maximum

I started by finding out what covariance type to use as one of the parameters to run the Gaussian Mixture Models(GMM) which is a probabilistic approach to clustering. I tested the number of clusters from 2-6 on covariance types: Spherical, Diagonal, Tied and Full to see what got the highest percentage of correct labels. In order to find a good number of clusters I decided to run the Silhouette score from 2 – 6 clusters.

# of Clusters	Silhouette score **Spherical**	Silhouette score **Diagonal**	Silhouette score **Tied**	Silhouette score **Full**
2	.5695	.0075	.7912	-0.04
3	.3854	-0.107	.7268	-0.031
4	.4035	-0.247	.6332	-0.098
5	.3668	-0.274	.3089	-0.201
6	.3676	-0.309	.3758	-0.201

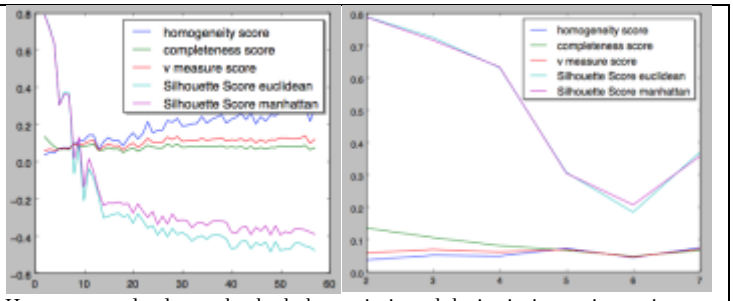
We can see that the covariance type " Tied " generated the highest score when the clusters equal two, so I will explore that a bit more.



After analyzing this graph for GMM, I decided to choose the number of clusters as 2 because you can see at 3 clusters the silhouette score decreases more significantly, while the other measure do not increase as much. This does make sense because we know that there are 2 data labels for this dataset, but I wanted to visualize what the clusters looked like for four different attribute combinations.

Cluster #	Count	Percent of positive diabetes diagnoses given true labels
1	394	32.99%
2	374	36.89%

This resulted with an overall accuracy score of 52.34%, which was a decrease in accuracy over just using k-means alone.



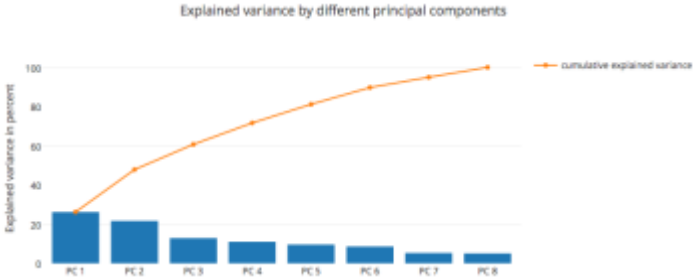
Here you can clearly see that both the extrinsic and the intrinsic metrics are in agreement that the correct number of clusters for EM with covariance_type "tied" is two clusters.

Cluster #	Count	Percent of positive diabetes diagnoses given true labels
1	4388	37.01%
2	213	88.73%

This resulted with an overall accuracy score of 35.81%, which was a decrease in accuracy over just using k-means alone.

1.5 Principal Component Analysis

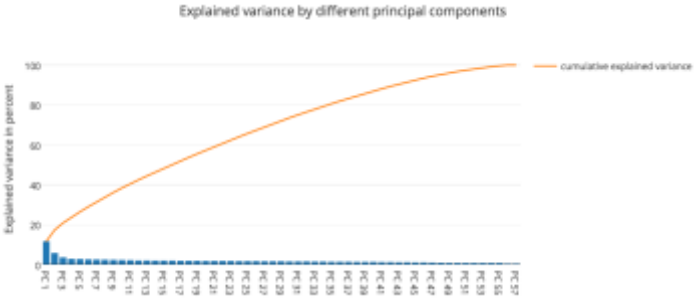
I started by calculating the eigenvalues, then ordering them from highest to lowest to choose the top k eigenvectors.
From high to low the descending eigenvectors were:
[2.09, 1.73, 1.02, 0.87, 0.76, 0.68, 0.41, 0.40], you can see what principal component they correspond to in the graph below.
I then plotted these against the explained variance to tell me how much information can be ascribed to each component.



From this you can see that the first 6 principal components contain 90% of the data so I would say we can safely drop the last two components without losing too much information.

2.5 Principal Component Analysis

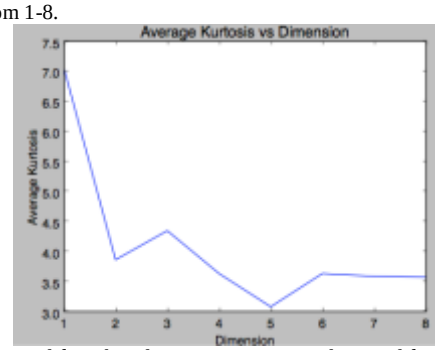
I started by calculating the eigenvalues, then ordering them from highest to lowest to choose the top k eigenvectors.
From high to low the top 8 descending eigenvectors were (I only did the top 8 because 57 of them were too long to list here:
[6.59, 3.26, 2.00, 1.61, 1.54, 1.46, 1.41, 1.37], you can see what principal component they correspond to in the graph below.
I then plotted these against the explained variance to tell me how much information can be ascribed to each component.



From this you can see that the first 43 principal components contain 90% of the data so I would say we can safely drop the last two components without losing too much information.

1.6 Independent Component analysis

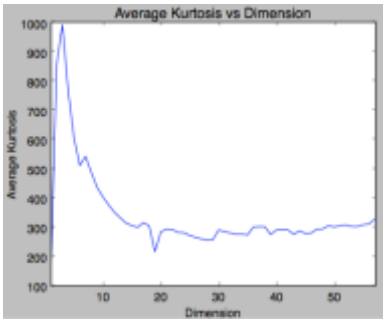
Here I used fastICA in sklearn and plotted the total kurtosis for the number of components from 1-8.



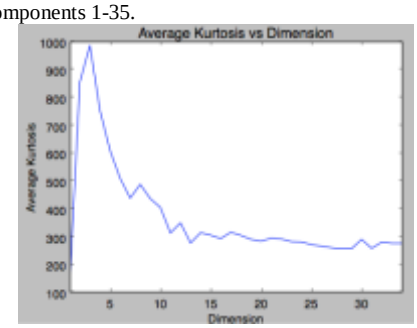
ICA is typically good for when data sources are mixed up, and for this dataset this is not the case. Therefore, I do not believe that ICA is going to do a good job with this dataset. Again, we come back to our domain knowledge of this dataset, because we have prior knowledge that these data sources are not mixed up. Kurtosis is a statistical modes of a distribution after the variance. Variance is the second mode, give you an idea of the non-Gaussian components of the distribution, can take into account the non-Gaussianity of the data, in order to identify the separate sources. Data should have some non-Gaussian components for ICA to work properly. For this dataset it is clear that independent components, 1, 2 and 3 have the highest kurtosis.

2.6 Independent Component analysis

Here I used fastICA in sklearn and plotted the total kurtosis for the number of components from 1-57.



Here, I wanted to see where the kurtosis kind of "bottoms out" so I decided to zoom in from components 1-35.



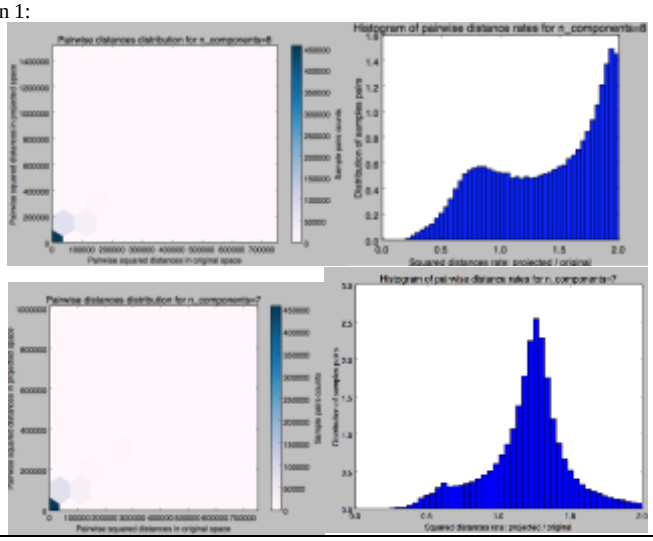
The rational for ICA is the same for this dataset as the diabetes dataset I described to the left. For this dataset it is clear that the data is nonguassian from the high kurtosis scores. kurtosis scores are decreasing the most from 5 all the way to 15, with the highest kurtosis value being at components equaling 3.

When comparing the two kurtosis scores it is clear that the Spambase dataset is much more nongussian than the diabetes dataset.

1.7 Randomized Projections

Here I ran randomized projections in order to see if I could reduce my features. To see how this performed I used SparceRandomProjections is sklearn. The goal of random projections is to preserve the pairwise distances between any 2 points in your data. My hypothesis is that is not going to do well on my dataset because it does not have many dimensions or features. I started this by calculated the Johnson Lindenstrauss minimum dimensions required to preserve the pairwise distances. This dataset has 768 points and the Johnson Lindenstrauss minimum dimensions to preserve the pairwise Euclidean distances in a tolerance of .01 epsilon calculated that the random projection matrix should have 5,694 components, we can't reduce 8 features to 5,694. This just means that we can't make any assumptions regarding the preservation of pairwise distances between data points. So, I decided to see what the pairwise distance distribution looked like for 3 different runs showing components [8,7,6,4,2].

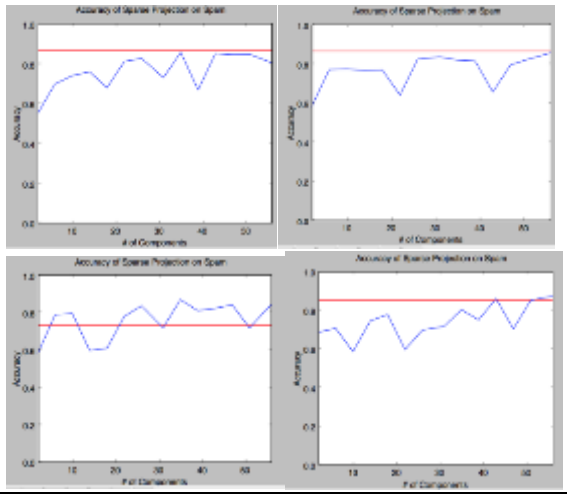
Run 1:

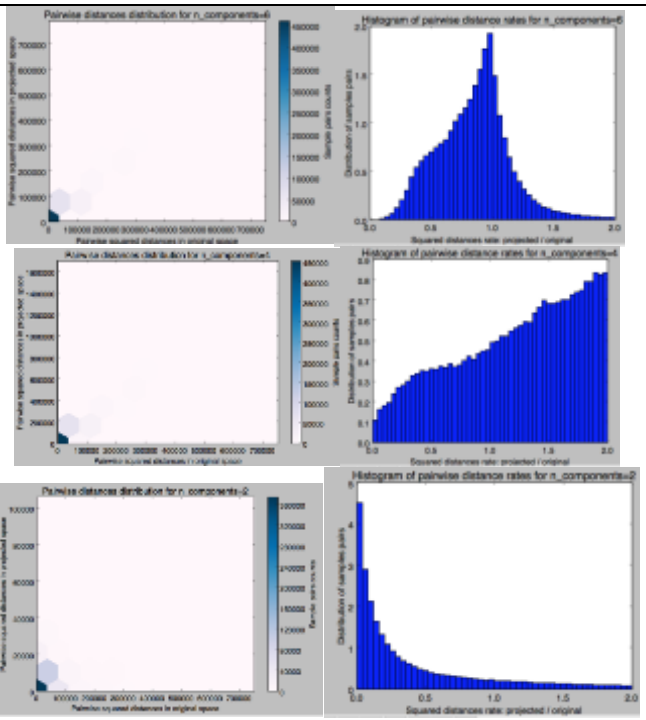


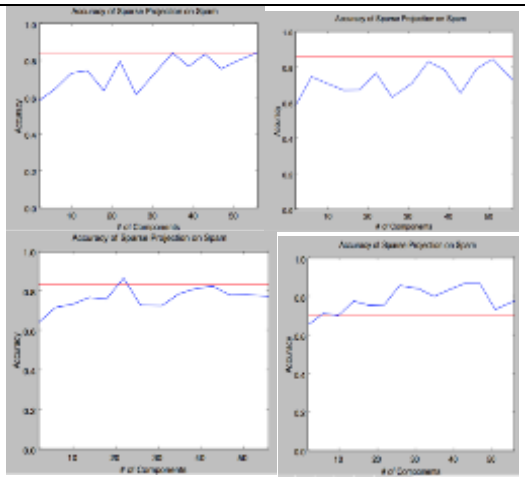
2.7 Randomized Projections

I started this by calculated the Johnson Lindenstrauss minimum dimensions required to preserve the pairwise distances. This dataset has 4,601 points and the Johnson Lindenstrauss minimum dimensions to preserve the pairwise Euclidean distances in a tolerance of .01 epsilon calculated that the random projection matrix should have 7,229 components, we can't reduce 57 features to 7,229. This just means that we can't make any assumptions regarding the preservation of pairwise distances between data points. For this dataset I decided to try a different way to display what a good number of components are because there are too many graphs to show all the pairwise distance rates like it did with the diabetes dataset to the left.

I decided to compare the accuracy of the randomized projections with a baseline accuracy of a SVM (because knew SVM performed well on this dataset from previous assignments), against the number of components. I did this for 8 runs.





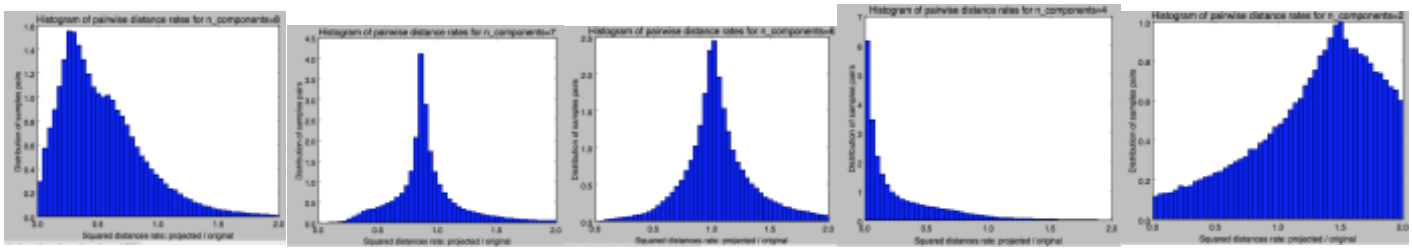


At first look it seems that for each run randomized projections returned results with lots of variations in it. These results were not extremely clear to me on what number of components bets fits, but after more comparison it seems that most of the tests eventually get close or outperform the original SVM with around 45 components.

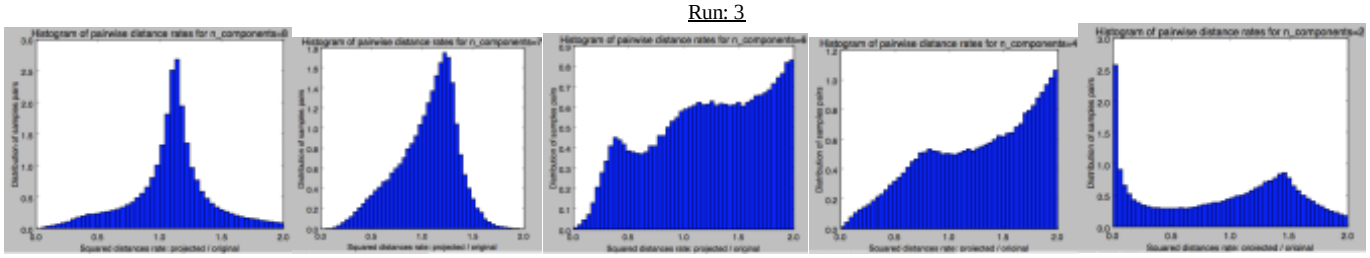
1.7 Randomized Projections Diabetes (Continued from left column)

Run: 2

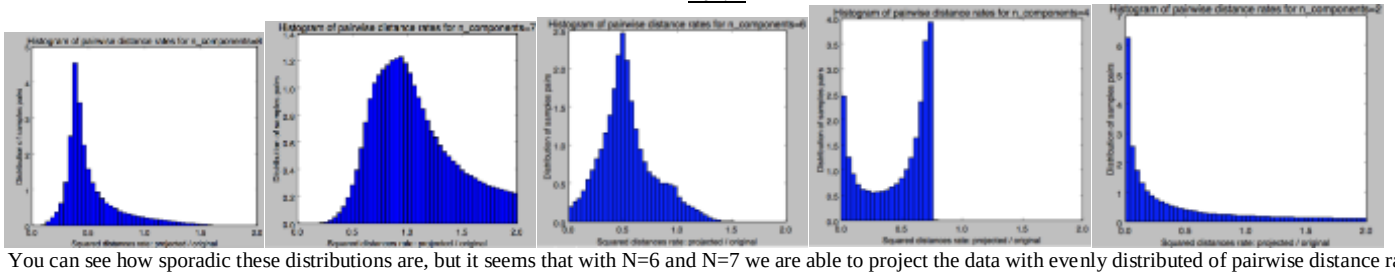
Note: stopped producing the pairwise distribution (the left graph) for the rest because they ended up all looking the same.



Run: 3



Run: 4



You can see how sporadic these distributions are, but it seems that with N=6 and N=7 we are able to project the data with evenly distributed of pairwise distance rates which preserves the pairwise distances between the data points.

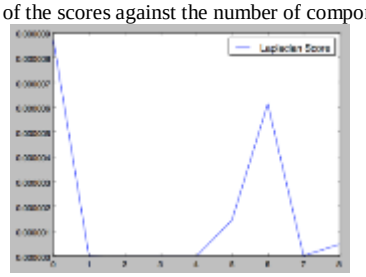
1.8 **Laplacian Score**

I decided to try a feature selection algorithm that uses a score called the Laplacian Score. The Laplacian Score is fundamentally based on Laplacian Eigenmaps and Locality Preserving Projection. The basic idea of the Laplacian Score is to evaluate features according to their locality preserving power. For each feature, its Laplacian score is computed to reflect its locality preserving power. LS is based on the observation that; two data points are probably related to the same topic if they are close to each other. In fact, in many learning problems such as

2.8 **Laplacian Score**

Explanation of the LS score, is the same as the column on the left.
Here is the graph all of the scores against the number of components.

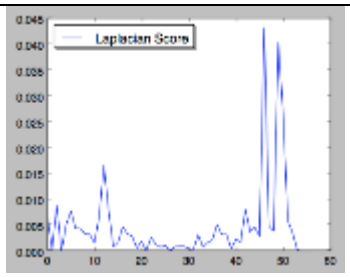
classification, the local structure of the data space is more important than the global structure. In order to model the local geometric structure, we construct a nearest neighbor graph. LS seeks those features that respect this graph structure. Here is the graph all of the scores against the number of components.



I then use normalized mutual information score (NMI) and accuracy (ACC) to measure the performance of unsupervised feature selection algorithm Laplacian Score.
I decided to test these metrics on 1-2 and 4-8 features compared to the accuracy on all the components.

Selected Features	Normalized Mutual Information Score	Accuracy
All	0.0297	66.01%
1-2 and 4-8	0.0468	66.40%

According to the LS score we get a better NMI score and a bit .4% greater accuracy while using 3 less features! I would say these are pretty great results.



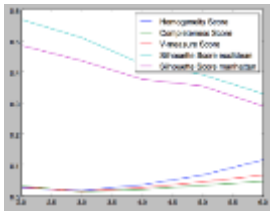
From this graph it is clear the best LS score lie from components 45-50, and also 3-12.
I decided to use just those components and test the mutual information score and accuracy against using all of the features to see if I get decent results.

Selected Features	Normalized Mutual Information Score	Accuracy
All	0.0472	63.59

Selected Features	Normalized Mutual Information Score	Accuracy
3-12 and 45-50	0.0162	60.75

It is pretty clear that the LS score did a good job of finding what the important features are. It was able to get in the same accuracy range by +/- 3% while using only 14 of the features.

1.9 Dimension Reduction with PCA then Cluster with K-Means

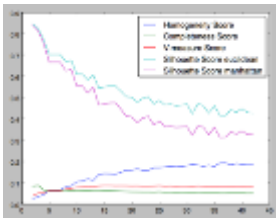


After running k-means with the PCA reduced to n-components equal to 6 I ended up getting the best accuracy with K clusters equaling 2 with a percent correct given true labels of 66.02%. You can see the silhouette score using the euclidean distance is still the highest with K=2.

Cluster #	Count	Percent of positive diabetes diagnoses given true labels
1	603	30.12%
2	165	52.12%

These results were very surprising to me because PCA did not increase or decrease my accuracy, for k means, it ended up staying the exact same. This meant that those two attributes did not contain any good information.

2.9 Dimension Reduction with PCA then Cluster with K-Means



After running k-means with the PCA reduced to n-components equal to 6 I ended up getting the best accuracy with K clusters equaling 2 with a percent correct given true labels of 57.92%.

Cluster #	Count	Percent of positive spam given true labels
1	4478	40.48%
2	123	0%

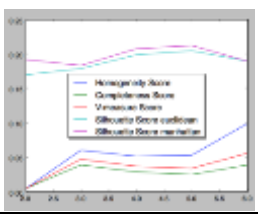
This did a bit worse than just using k-means alone, my guess is after reviewing the previous PCA analysis is that there are a lot of values at the end of the dataset that contribute only a small bit of extra information, but all those little component add up and make a notable difference when clustering. It seems that 90% explained variance is not enough for this dataset.

1.1.0 Dimension Reduction with ICA then Cluster with K-Means

After running multiple experiments, I found that I was getting the highest Silhouette scores for when K=2 and K=4, but from previous domain knowledge I knew that the correct K value should be 2 for k-means. So I decided to see what the best number of components would be knowing this previous domain knowledge. For ICA I decided to list the accuracies I got against components equaling to 2-7.

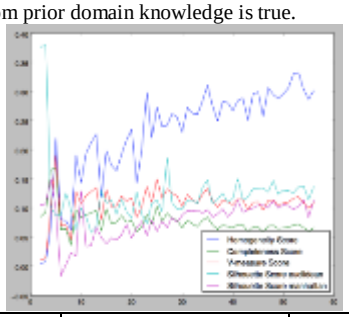
N_components	2	3	4	5	6	7
% Accuracy Given True Labels	36.50%	36.46%	46.88%	52.87%	59.64%	46.61%

Here you can see that the accuracy increases up until n_components equal 6, I next plotted the intrinsic and extrinsic metrics for clusters 2-6 with fastICA equaling 6 components.



2.1.0 Dimension Reduction with ICA then Cluster with K-Means

I after running k-means with ICA n_componets equaling 45 on k-means we can see that the highest silhouette score using the Euclidean distance is at K=2. Which we know from prior domain knowledge is true.



Cluster #	Count	Percent of positive spam given true labels
1	4502	38.29%
2	99	89.89%

Here with n_components equaling 6 we can see that the highest score for the clusters is 5, but we know from prior domain knowledge that there are only 2 clusters to correspond to the labels we have so I decided to run a test with K=2.

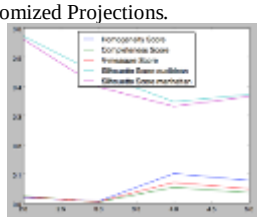
Cluster #	Count	Percent of positive diabetes diagnoses given true labels
1	538	32.34%
2	230	40.87%

This resulted with an overall accuracy score of 60.70%, which was a decrease in accuracy over just using k-means alone.

This resulted with an overall accuracy score of 62.30% given true labels, which was just a small decrease in accuracy over just using k-means alone.

1.1.1 **Dimension Reduction with Randomized Projections Then Cluster with K-Means**

After running multiple tests with different K values I still got the best Silhouette score when K=2. They all came out to look a bit like this for n_components equaling 6 and 7 for Randomized Projections.



I then wanted to see what the best average score with K =2 for k means and focus n_components equaling 6 and 7 for Randomized Projections. These results are averaged after 10 runs because there is a lot of variation with randomized projections.

N_components	6	7
% Average Accuracy on Given True Labels	65%	43%

You can see clearly here that with n_components equal to 6 we get much higher accuracy.

2.1.1 Dimension Reduction with Randomized Projections Then Cluster with K-Means

Explanation of the LS score, is the same as the column on the left.

Here is the graph all of the scores against 45 of the components.

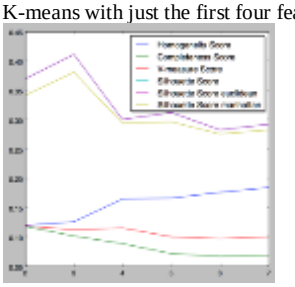
Average cluster sizes and accuracy over 10 runs:

Cluster #	Count	Percent of positive spam given true labels
1	4480	40.49%
2	121	0%

Total average accuracy over 10 runs: 57.97% given true labels, which has been the worst score of all the dimension reduction algorithms so far.

1.1.2 Lapacian Score with K-means

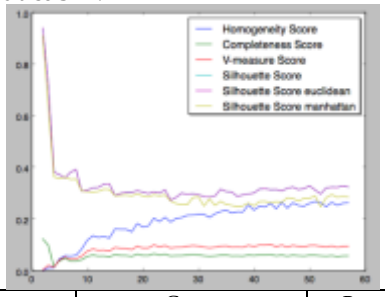
For the Laplacian Score I got the best accuracy and mutual information with just the first four features. So I ran K-means with just the first four features to see the results.



Here it seems that K could equal 2 or 3 because the silhouettes score is lower with K=2 compared to K=3 but the other scores either stay the same or go down a bit when K=3, but as stated before we know that K should equal 2 from prior domain knowledge so I decided to find the accuracy knowing that. I ended up finding that with K=2 on the first four features, I got an accuracy on the true labels of 28% which is not very good at all.

2.1.2 Lapacian Score with K-means

For the Laplacian Score I got the best accuracy and mutual information with features, 45-50, and also 3-12.

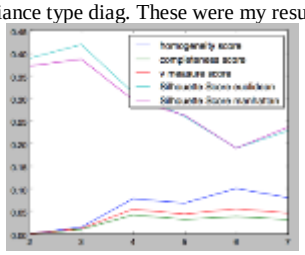


Cluster #	Count	Percent of positive spam given true labels
1	4594	39.312%
2	7	100%

Total average accuracy of 60.75% given true labels, which has been the second best score of all the dimension reduction algorithms so far.

1.1.3 **PCA with EM**

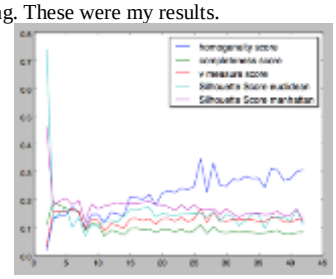
I ran EM with PCA= 2 components and with all of the following EM tests I ran them with covariance type diag. These were my results.



It is clear here that for EM with PCA equaling 2 components that the best number of clusters is either 2 or 3. From prior domain knowledge we know that the correct number of clusters for the labels should be 2, so I decided to run the stats knowing that.

2.1.3 PCA with EM

I ran EM with PCA= 43 components and with all of the following EM tests I ran them with covariance type diag. These were my results.



It is clear here that for EM with PCA equaling 43 components that the best number of clusters is 2.

Cluster #	Count	Percent of positive diabetes given true labels
1	372	37.09%
2	396	32.82%

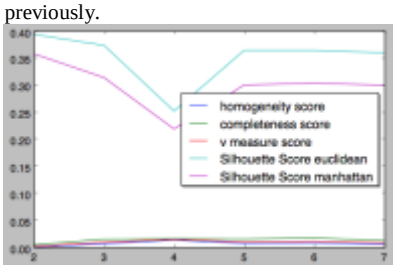
With an overall accuracy score of 47.36% on the corresponding set of true labels. This was not as good as using the whole dataset.

Cluster #	Count	Percent of positive spam given true labels
1	4478	40.48%
2	123	0%

Total average accuracy of 57.92% given true labels

1.1.4 ICA with EM

Next I tested EM clustering with ICA equaling to 6 components like which I found the best results for previously.



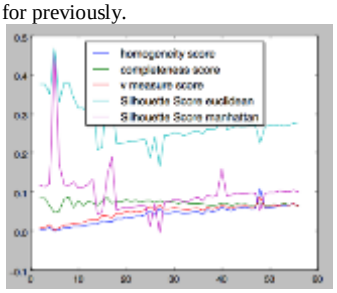
Here it is clear that the best clustering with ICA equaling to 6 components that EM does best with the amount of clusters equaling to two, which is consistent with our prior domain knowledge.

Cluster #	Count	Percent of positive diabetes given true labels
1	729	34.29%
2	39	46.15%

With an overall accuracy score of 64.47% on the corresponding set of true labels.

2.1.4 ICA with EM

Next I tested EM clustering with ICA equaling to 45 components like which I found the best results for previously.



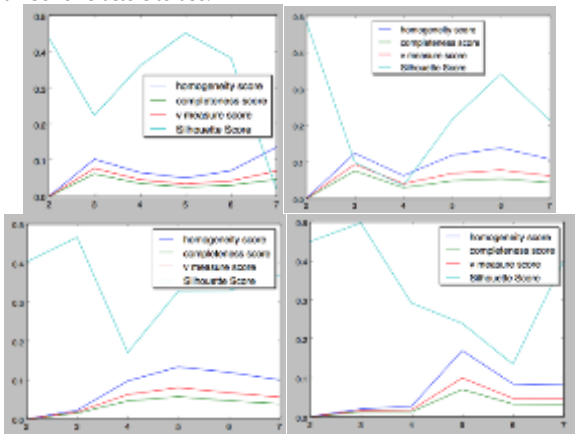
You can see here that the highest scores for the number of clusters appears around 2 and 4. From prior domain knowledge we know the correct number of clusters for this dataset to test against the true labels is two, so I will find the results knowing this.

Cluster #	Count	Percent of positive spam given true labels
1	4566	39.70%
2	35	0.00%

With an overall accuracy score of 39.40% on the corresponding set of true labels.

1.1.5 RP with EM

Next I tested EM clustering with RP equaling to 6 components like which I found the best results for previously. After running the test multiple times, I got the best scores for K values equaling to 2 and 3. Here are some of the results from testing which number of clusters to use.



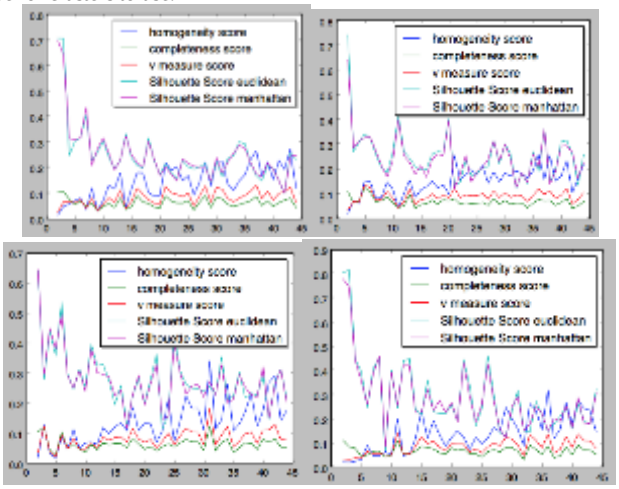
We know from prior domain knowledge that the correct number of clusters should be equal to two. So I decided to find the accuracy of RP with the components equaling to 6 and EM with 2 clusters. Here are the averages of the % correct against the true labels over 10 iterations.

Cluster #	Count	Percent of positive diabetes given true labels
1	398	33.17%
2	370	36.75%

The overall accuracy score of 10 runs was 54.21% on the corresponding set of true labels. Compared to ICA this result is better, but the runtime takes much longer because you have to do 10 runs to get the average.

2.1.5 RP with EM

Next I tested EM clustering with RP equaling to 6 components like which I found the best results for previously. Here are some of the results from testing which number of clusters to use.



After running the test multiple times, I got the best scores for K values equaling 2, which we know prior domain knowledge is correct.

Here are the averages of the % correct against the true labels over 10 iterations.

Cluster #	Count	Percent of positive spam given true labels
1	123	100.00%
2	4478	40.48%

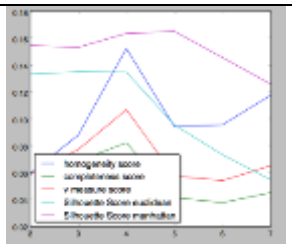
The overall accuracy score of 10 runs was 58.14% on the corresponding set of true labels.

1.1.6 LS with EM

Next I tested EM clustering with the LS score, where we just test the first four features in the dataset. Here the results I got were interesting.

2.1.6 LS with EM

Next I tested EM clustering with the LS score, using features 45-50, and also 3-12.

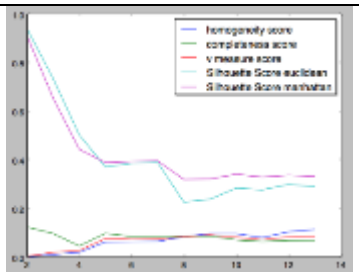


These scores show that the number of clusters should equal 4 with EM, but we know that is wrong from prior domain knowledge. So I decided to see what the accuracy comes out to for when the number of clusters equal to 2 because we know the true labels for that.

EM clusters = 2

Cluster #	Count	Percent of positive diabetes given true labels
1	492	44.51%
2	276	17.75%

With an overall accuracy score of 58.07% on the corresponding set of true labels.



It is clear here that for EM using the LS scores to select features 45-50, and also 3-12 that the best number of clusters is 2.

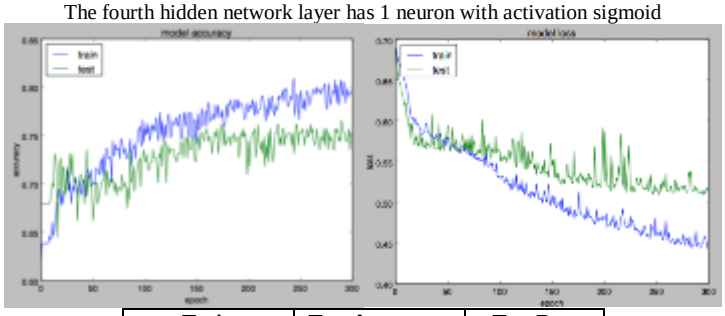
Cluster #	Count	Percent of positive diabetes given true labels
1	4594	39.31%
2	7	100.0%

With an overall accuracy score of 60.74% on the corresponding set of true labels.

Neural Networks with Diabetes Dataset

Neural Network Result from Assignment 1

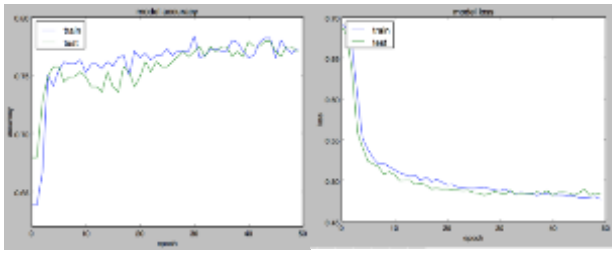
The first hidden network has 15 neurons with activation relu
The second hidden network layer has 6 neurons with activation linear
The third hidden network layer has 4 neurons with activation relu
The fourth hidden network layer has 1 neuron with activation sigmoid



Train Accuracy	Test Accuracy	Test Run Time (Seconds)
0.7932	0.7532	7.44

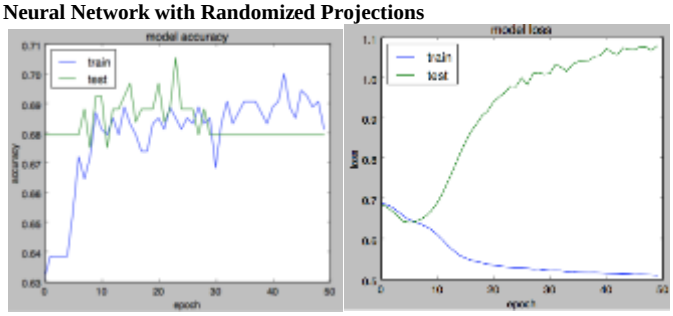
I kept the structure of my network the same for the following tests, only adjusting the initial input size when need be.

Neural Network with PCA



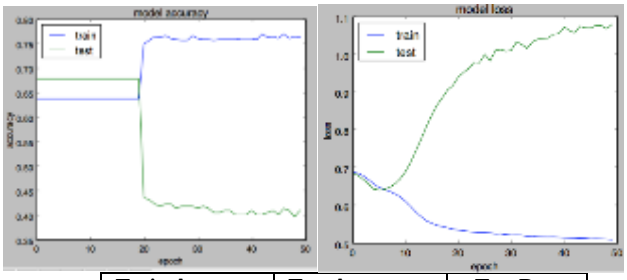
Train Accuracy	Test Accuracy	Test Run Time (Seconds)
77.28%	77.06%	3.61

Neural Network with Randomized Projections



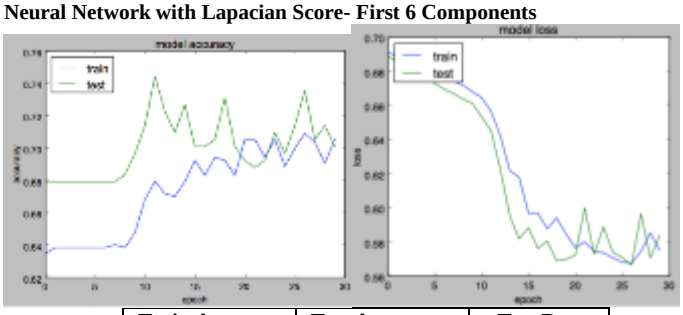
Train Accuracy	Test Accuracy	Test Run Time (Seconds)
68.91%	67.97%	2.96

Neural Network with ICA



Train Accuracy	Test Accuracy	Test Run Time (Seconds)
76.35%	41.13%	3.16

Neural Network with Lapacian Score- First 6 Components



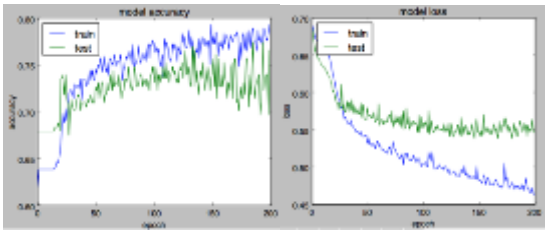
Train Accuracy	Test Accuracy	Test Run Time (Seconds)
70.57%	70.13%	2.74

Conclusion of the Dimensionality Reduction Algorithms with NN

The best performer by far was the Neural Network with PCA, although it was a bit slower than the rest it generated much better results with minimal overfitting and underfitting. It even outperformed my original model in assignment 1 which had a train accuracy of 79.32%, test accuracy of 75.32% and a run time of 7.44 seconds.

7.1 Clustering Algorithms Applied as New Features with NN

Here I applied my two clustering algorithms as new features. The first was k-means with k=2, and EM as a Gaussian mixture model with n_components equaling to two with a covariance type as diag.



I got the best results with 200 epochs with a batch size of 30.

Train Accuracy	Test Accuracy	Test Run Time (Seconds)
79.32%	76.62%	4.33

These results are very interesting, they scored the second highest train and test accuracy among all the other tests I ran, while minimizing model loss, this is interesting because I would not have even thought to add the clustering algorithms in as features, let alone expect to get good results from it. Although it took more time than some of the other tests, it produced significantly better results than them.

Conclusion

	Best Stats of Diabetes Unsupervised Learning Algorithms	Best Stats of Spambase Unsupervised Learning Algorithms
Clustering Algorithm	**Accuracy on True Labels**	**Accuracy on True Labels**
K-Means	66.02%	63.59%
EM(GM)	52.34%	35.81%
Dimension Reduction with K-means		
PCA	66.02%	57.92%
ICA	60.70%	62.30%
RP	65%	57.97%
LS	28%	60.75%
Dimension Reduction with EM		
PCA	47.36%	57.92%
ICA	64.47%	39.40%
RP	54.21%	58.14%
LS	58.07%	60.74%

In conclusion, the best clustering algorithm for the Diabetes dataset is, K-means. As for dimension reduction, for k-means I would say that PCA did pretty good because it got the same score with less dimensions which results in lower computational time. For EM, ICA got pretty close to the accuracy of K-means with two less features which I think is pretty impressive as well. The best clustering algorithm for the Spambase dataset is, K-means by a wide margin. As for dimension reduction, for k-means I would say that ICA did a good job because it was able to get very close to the same accuracy of just k-means with just 45/57 of the components, lowering computational time, while retaining accuracy. For EM, LS got pretty close to the accuracy of K-means with a lot less features, 14/57 which is extremely impressive. Overall, none of the dimensionality algorithms with clustering did better in terms of accuracy than just using the clustering algorithms on their own, but some of them got very close, while using less dimensions thus resulting in less computing time, this is a very important property, especially when you get very large datasets.