Competition 1 Write-Up
CS 4786

Saarthak Chandra (sc2776)
Shweta Shrivastava (ss3646)
Vikas P Nelamangala (vpn6)
Jae Young Chang (jc2379)

November 2, 2016

1 Introduction

In this write-up, we explain the motivation behind our model which groups 12000 data points into ten distinct clusters, and predicts which of the ten digits does each data point represent. We were provided with the following data sets:

- A **features** data set that describes the image of handwritten digits.
- An **adjacency** graph that contains information of how similar two data points are.
- 3 **label**ed points for each of the 10 classes to aid with proper classifications.

2 Pre-analysis on Individual Data Sets

Before we started developing an algorithm to cluster the data points, we first performed some preliminary tests on the given individual data sets to best identify the optimal method for combining the two.

2.1 Features Data

The features data is a list of 103 dimensional data points that were extracted from the raw data. Our first intuition was to apply PCA to this data set, and see what we can achieve. Before we applied PCA, we first decided to look at the eigenvalue spectrum of the covariance matrix of the features, as shown in Figure 1.

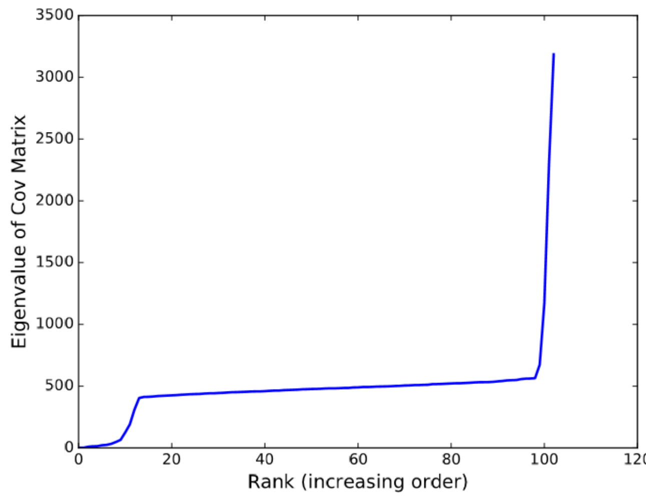

Figure 1: Eigenvalues of covariance matrix of feature matrix.

We noticed a very sharp increase in the eigenvalues at high rankings, suggesting that there were subdimensions within the data set that could yield us valuable data for clustering. Since the top 3 eigenvalues of this data set were much higher than the rest, PCA with K=3 was performed on the features, and the result projected down to two dimensions is shown in Figure 2.

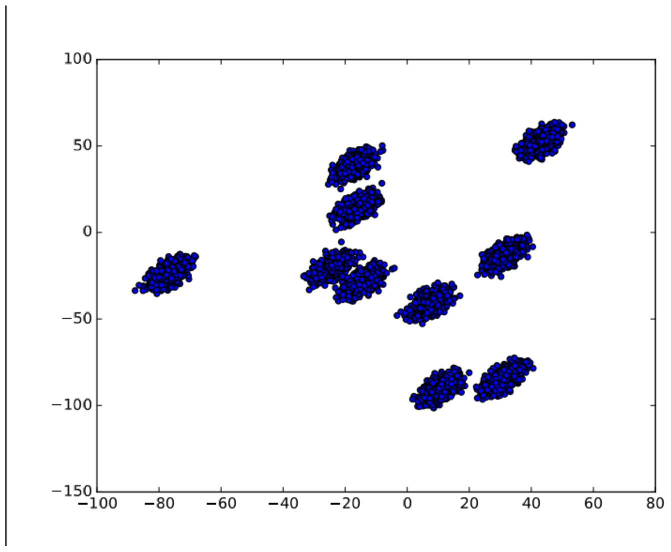

Figure 2: Result of PCA with K=3 on Features matrix.

Figure 2 shows 10 distinct clusters, even in the projected 2D space, therefore this indicated that the final algorithm would involve applying some sort of PCA to some modified data set from the Features. However, just applying PCA on the features alone would not be sufficient as we could not be sure which digit each point within any given cluster corresponds to. We therefore started matching the data points with the 30 seed data points that were given to us. We found that these seeds were mixed between these 10 clusters, as shown in Figure 3. This meant that additional information from the adjacency matrix was required for proper clustering.

2.2 Adjacency Matrix Data

For our adjacency matrix, we decided to extract the spectral features using the Normalized Laplacian matrix, as descibed in [1]. The normalized symmetric Laplacians are defined as

$$L_{sym} := D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \tag{1}$$

Where D is the degree matrix, and W is the original agjacency graph matrix. To extract the spectral embedding from this matrix, we simply calculate the first K eigenvectors of the Laplacian, and normalize each row such that you end up with 12000, K dimensional vectors with norm of 1. From this point, one could perform a clustering method like k-means or single link to group nearby data points together.

Just plotting the first two dimensions of the spectral data gives us Figure 4, and we can see that this data set alone fails to yield meaningful clustering for our goal. This is mostly due to the fact that we are projecting down a high-dimensional data set into just two. Although the boundaries between the clusters will be difficult to visualize using the spectral data, we expect that merging this distance-based data along with the features will be important for improving our accuracy.

2.3 Combining Both Data Sets with Seeds

From the previous sections, we found that the PCA and the spectral data from the Laplacian separately were not sufficient to achieve 10 clusters with the seeds placed appropriately. Therefore, we decided to apply an algorithm that finds similarities between the two data sets – Canonical Correlation Analysis. After performing CCA between the two, we had to project the result to lower dimensions, and then perform some sort of distance-based clustering method such as k-means.

Our justification for using CCA between the Laplacian eigenvectors and the features is that both of these datasets are distance based datasets, meaning that any correlation between the two would be extracted out using linear transformations.

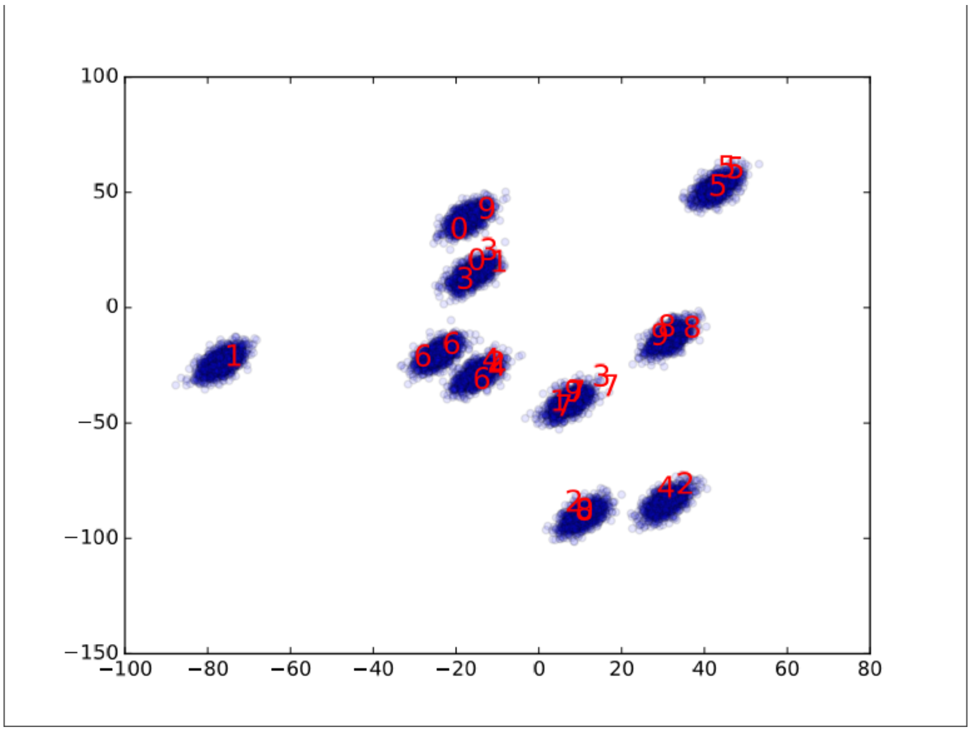

Figure 3: Result of PCA with K=3 on Features matrix, plotted with the seeds. The red numbers are randomly scattered throughout the clusters, demonstrating that more information is needed to properly cluster.

However, we still haven't taken into consideration on how to label each cluster with a specific number. In order to do this, we can perform k-means with initial means of the clusters as the mean positions of each seed group. After the clusters converge, we can use these means to properly categorize each cluster with a digit.

3 Main Algorithm

Our main algorithm is as follows:

- Calculate the Normalized Laplacian of the Adjacency matrix, and calculate its first K_1 eigenvectors. Transpose this matrix to get N rows of K_1 dimension data points.
- Perform CCA with dimensions K_2(reducing it to this dimension) on the first K_3 columns of the features matrix and the spectral embedding.
- Perform k-means on the result to obtain the initialization of the means,variances of the clusters.
- Perform Gaussian Mixture Model using the results of k-means to obtain more refined clusters, with prior weights as 0.1 for each of the 10 clusters.
- Properly categorize each group using the initial seeds.

The parameters K_1,K_2, and K_3 are adjustable parameters for our model, and we tried several values before achieving good accuracy on Kaggle.

Parameter values that achieved good results were $K_1 = 1500$, $K_2 = 8$, and $K_3 = 12$. Plotting the results of the CCA in 3 dimensions in Figure 5, we observed ten distinct and separated clusters with all of the values of the seeds in appropriate positions.

Overall Accuracy achieved (private) : 99.85%

3.1 Motivations for Our Parameters

- K_1=1500
  This choice was made based on our Kaggle Attempt 4.6 (written below), where we obtained a very

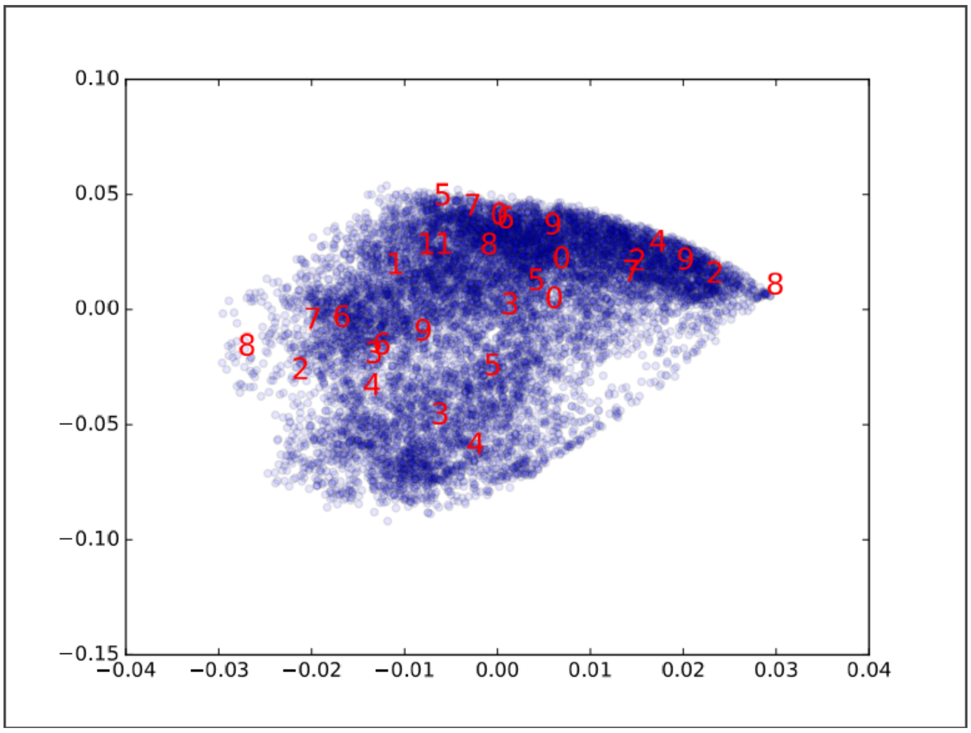

Figure 4: 2D Projection of the Eigenvectors of the Laplacian Matrix. Just from this data, we fail to see meaningful clusterings.

low accuracy when we chose just the top 10 eigenvectors. Once we made our model, we started off with K_1 as 200, since it had to be more than the dimensionality of our 103 dimension feature set. We then increased this to 500,1000 and finally 1500 in order to preserve more dimensions of the embedding so that we can get better results when we perform CCA with the features data. We stopped at 1500 since we obtained good cluster sizes, all nearing the expected size of 1200.

- K_2=8
  This choice was made since after we performed the spectral embedding, we wanted CCA to bring down our dimensionality, while still retaining relevant information of our dataset. K_2 as 2/3 gave us bad cluster sizes, after we performed KMeans algorithm on it, which meant that we were losing out on most of the clustering information in such low dimensionality. While running our final model, we started off with K_2 as 3 and moved up till 8 where we got our best results.

- K_3=12
  The image data that we were given,in the form of features.csv was noisy. When we plotted the variance of the dataset, we noticed that dimensions 1-13 had the least variance, dimensions 14-100 had slightly more and dimensions 101-103 were extremely noisy. Hence we begun by choosing K_3 as 13 and brought it down to 12 where we got our best results. We referred to figure 1 to visualize the values of the eigenvalues and determine our parameters.
  We then performed GMM clustering, with KMeans algorithm for pre-clustering, in order to obtain our final accuracy.

3.2 Partial Supervision

- For some pre-final model submissions : We did not always get all the seeds in the same cluster. In such cases, we chose the labels based on the most frequent cluster number across that row of seed points in the matrix. We made a matrix, where we plotted the cluster numbers against each of the seed points.

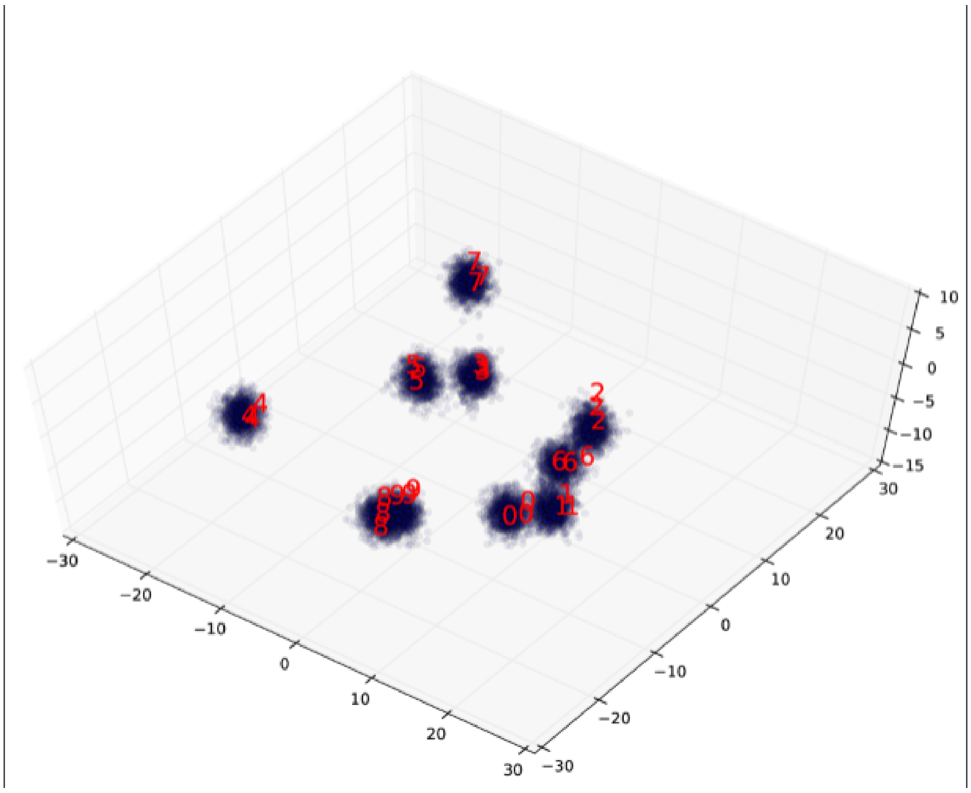

Figure 5: 3D Projection of the results of CCA using $K_1 = 1500$, $K_2 = 12$, and $K_3 = 8$. We notice 10 distinct clusters with the positions of the seeds in appropriate locations.

Label0 seed1 :	ClusterNum	Label0 seed2 :	ClusterNum	Label0 seed3 :	ClusterNum
Label1 seed1 :	ClusterNum	Label1 seed2 :	ClusterNum	Label1 seed3 :	ClusterNum
Label2 seed1 :	ClusterNum	Label2 seed2 :	ClusterNum	Label2 seed3 :	ClusterNum,

and so on for all 10 digits

Thereafter, for the first row (Label 0), we noted the most frequent occurring cluster number (common for atleast 2 out of 3 seed points) and replaced all occurrences of that cluster number in our clustering algorithm result, with 0. We proceeded row-by-row in a similar fashion for assigning cluster numbers for all the 10 digits from 0-9.

- For the Final model submission: We used a combination of k-means and Gaussian Mixture model to classify each cluster with a digit. First, we calculated the means of each group of 3 seeds, whose indices were given to us, and used those as the first guesses for our k-means algorithm. After running k-means, which returned the adjusted means of clusters, we used those new means as the initial guess for running the Gaussian Mixture Model. What resulted was a vector of size 12000 that classified each index to a cluster number.

  However, we had to do some minor cleaning of the data at the very last step. Since GMM labels the first cluster as "1", and the last cluster as "10", we simply had to subtract 1 from every classification value so that the classifications range from 0 to 9 inclusively.
  Reason : We checked that all the seeds from the 1st row seed.csv were in cluster 1, 2nd row of seeds were in cluster 2 and so on.

4 Failed Attempts

Overall on Kaggle, we had **31 submissions.** Here we mention the ones we had most learning from :

4.1 First Attempt : PCA(3 dim),K-Means with seed data as centroids

Features dimension - 3, k-means Algorithm: lloyd, 10 clusters :

As per our initial write-up, where we saw PCA plot in nearly 2 dimensions, we could see some clear clusters, and some were 'muddled'. Hence we did a PCA to 3 dimensions, since there was a drastic drop in eigenvalues after the first 3 eigenvectors. Then, for clustering, we tried k-means WITHOUT using the seed data as the centroid, and giving the number of clusters as 10. Despite being a naive approach, we wanted to get a hunch of the clustering present in the data. Surely, we got 10 roughly equally sized clusters - but they were not matching up with the seed points so we were not able to label them correctly for a submission.

Then we used the starting centroid for k-means (as learnt in class) as the centroid of the 3 seed points that were given to us. The results we got were not very promising, since not all the seed points were in the right clusters, for example - out of 3 seed points for cluster 0 , 2 were in cluster 0 but one was in cluster 1. Similarly, we got few seed points where none of the seeds belonging to one cluster were put into the same cluster by the k-means algorithm.

Accuracy : 10%

Learning/Intuition : Chosing the top 3 eigenvectors to do PCA, may not be the right choice.

4.2 Second Attempt : PCA(100 dim),K-Means with seed data as centroids

Features dimension - 100, k-means Algorithm: lloyd, 10 clusters :

Having learnt from our previous attempt, we chose to look at the cumulative variance represented by the eigenvalues/vectors. We noticed that it was at the 100th eigenvector that gave us near 99.9% value as the cumulative variance. Hence, we repeated k-means after bringing down the feature space to 100 dimensions (using top 100 eigenvectors). Looking at the results, all the seed points were still not assigned correctly, but a majority of them were and we decided to submit the clustering we obtained.

Accuracy : 37.15%

Learning/Intuition : Cumulative variance is a good measure to decide the cutoff point for the eigenvalues/vectors after PCA.

4.3 Third Attempt : PCA(94 dim),K-Means with GMM

Features dimension - 94, k-means Algorithm: llyod, 10 clusters :
From our understanding, GMM is often better than k-means since it is 'soft' clustering unlike k-means which is a 'hard' clustering algorithm. Hence, having seen improvements in our results, we decide to apply GMM on the feature set, after having reduced it to 94 dimensions. GMM algorithms that we found -

- Started with random assignments of means,variance as well as priors/'initial weights'.
- Required some pre-clustering,in order to decide an initial value of means,variance within a cluster as well as prior(weights or a notion of size of clusters). For this approach, we used k-means with cluster center initialization to 'pre-cluster' and obtain means/variances that were fed into the GMM algorithm. For init-weights, we chose them to be 0.1 (all even), instead of (size of current cluster)/(1200), since the latter would give us more skewed cluster assignments (already large sized clusters from the kmeans output could grow larger after GMM)

We pursued both the approaches(i&ii), and noticed that with the first approach(random initialization), we did not get results very different from k-means only, which meant that there were not too many outliers in 94 dimensions, which the GMM model could 'fit-in'. The second approach gave us better results, with cluster seeds more accurately labeled (not all seeds were correctly assigned still).

Accuracy : 39.217%

Learning/Intuition : GMM does not give very different results from kmeans, but can help cluster outliers better. This means, that in our case, we do not have many outliers, since GMM did not drastically improve(or even change) the results.

4.4 Fourth Attempt : PCA(94 dim),K-Means with GMM

Features dimension - 94, k-means Algorithm: llyod, 10 clusters :
Since we got an improvement in our results, we continued using the PCA+K-Means+GMM approach. We chose the 94th eigenvalue as the cutoff in this case, since we obtained nearly 99% variance at this eigenvalue. This was the only difference from our last approach. After PCA, we followed the same approach of using k-means and GMM to cluster the resulting reduced dataset.
Accuracy : 55.08%
Learning/Intuition : Since a mere change in the 'K' value for the PCA algorithm made a difference,we believe that our final algorithm/model will need to wisely choose a value for K, or more simply put, we would need to use a reduced dataset and not use all 103 dimensions.

4.5 Fifth Attempt : Spectral Clustering

Spectral Embedding - choosing top 10 eigenvectors of the Laplacian , k-means Algorithm: llyod, 10 clusters :
One approach we had not tried was using the adjacency matrix provided. The first thing that we thought of was using this to perform spectral clustering. For this, we tried 3 algorithms, choosing eigenvectors **2-11** from the Laplacian :

- Unnormalized spectral clustering Using this approach, when we tried clustering using KMeans, we obtained only one cluster. Hence we dropped this approach.
- Normalized spectral clustering - Shi and Malik : After the spectral embedding, we tried k-means on the embedding and we ran into an error stating 'kmeans works only on real values'. We noticed that our Laplacian had some complex numbers as well, and hence we decided to drop this approach.
- Normalized spectral clustering - Ng,Jordan, and Weiss : We tried k-means algorithm, with 10 clusters, initializing the start centroids using the seed points, and the results did not look promising. We even applied GMM, in order to fine-tune the results of kmeans algorithm. We did not obtain the correct assignments even for the seed points, most of the seeds belonging to one cluster were spread across different clusters. **Accuracy : 10.47%**
  Learning/Intuition : Spectral Clustering alone will not give us the right clustering. We dropped the approach of spectral clustering.

4.6 Sixth Attempt : CCA with Spectral Embedding, k-means

Having used the features dataset and adjacency matrix separately, we decided to use a combination of the two. One major failed attempt that we countered was an algorithm that was fairly similar to our main algorithm.

- Calculate the Normalized Laplacian of the Adjacency matrix, and calculate its first **10** eigenvectors. This yields a matrix to get N rows of **10** dimension data points.
- Perform CCA with dimensions K_2 on the first K_3 columns of the features matrix using the spectral data.
- Perform k-means on the result to obtain the means of the clusters.
- Perform Gaussian Mixture Model using the results of k-means to obtain 10 clusters.
- Properly categorize each group using the initial seeds.

The main difference with this failed method is that we set the dimension of the spectral data as 10, since we assumed that there would be 10 clusters. Every step after is exactly the same as our main algorithm. After adjusting the other two parameters, K_2 and K_3, to yield a data set with the groups of seeds placed in appropriate clusters, we performed K-means to cluster the sets. We plot the results of the CCA in Figure 6. Even though the seeds of each group are fairly close with each other.
Accuracy : 10%
Learning/Intuition : That this model does no better than random guesses, embedding in 10 dimensions loses out on a lot of the information that the data may have.

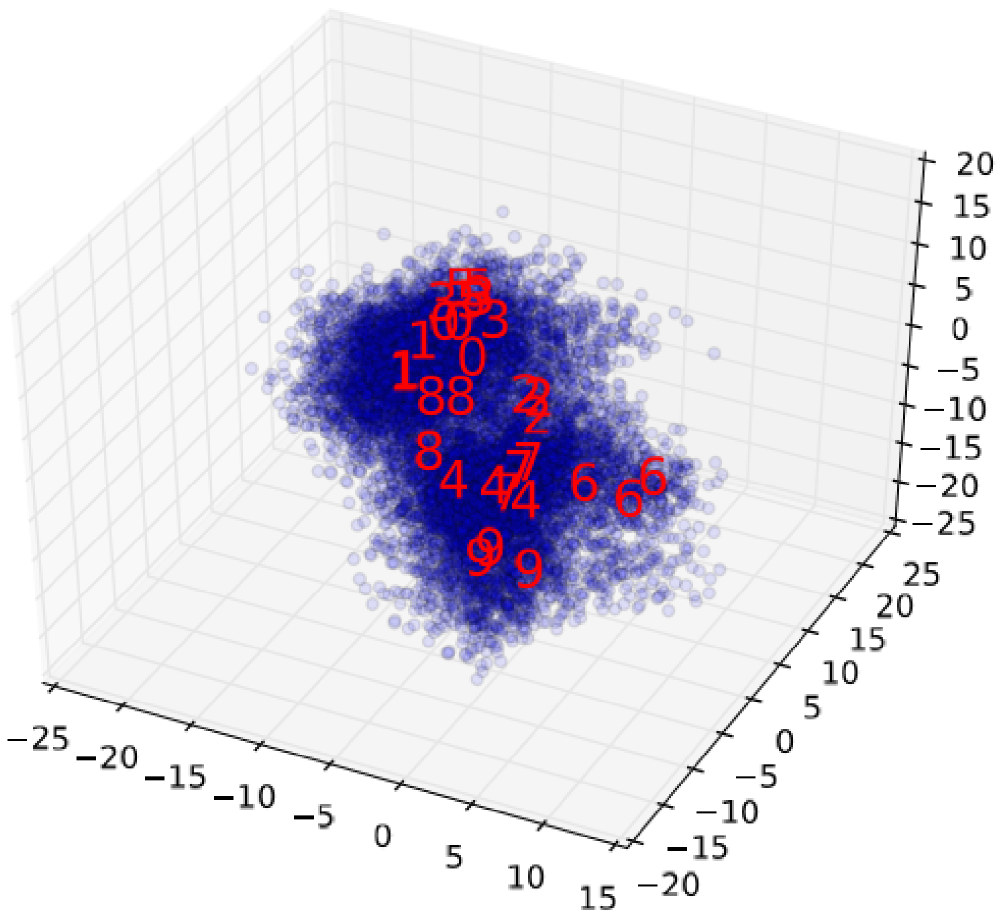

Figure 6: Results of CCA using our failed model. Although we were able to get the seeds of each group close together, this model failed to accurately predict the data sets.

4.7 Pre-Final Attempt : Spectral Embedding + CCA(3 dim) with GMM

Spectral Embedding - 1000 dimensions, GMM, 10 clusters, with prior initialization :
Since we were now pursuing CCA with spectral embedding, for this attempt we chose to choose the top 1000 eigenvectors of the Laplacian in-order to retain information within the spectral embedding,unlike our previous failed attempt where we chose 10 eigenvectors only. With this data, we did CCA with all 103 dimensions of the features dataset and reduced the dimensions to 3, post CCA. To this data, we applied GMM clustering with prior initializations (kmeans for pre-clustering as described earlier, as well as init-Weights as 0.1 for all 10 clusters). This gave us 29/30 seedpoints correctly assigned. We now realized that this would be our model - we would look at data-cleanup (which we did not pursue in this attempt) , as well as tweaking the dimensions of the spectral embedding.
Accuracy : 94.45%
Learning/Intuition : Data Cleanup will be needed to increase accuracy, which we learnt from our earlier failed attempts. However, the overall model in place was working and hence only the parameters were to be tweaked.

5 Improvements on Main Method Using Failed Attempts

- We should always begin with a preliminary analysis of the dataset - features.csv. We should study its variance, or in simpler terms - find out how noisy it is. Often data-set we get in the real world will not fully convey what we need, and hence we must try to do some initial pre-processing/cleaning before the model that we built takes over. In our final approach, we decided to use the first 12 dimensions of the features dataset. We obtained this by fine-tuning it down from 13 dimensions. We noticed that the first 13 dimensions had the least variance, then the variance increased till the 100th dimension, and finally it was maximum for 101-103 dimensions.

- Given a features dataset, it is possible to construct our own adjacency matrix. Hence, since we are given an adjacency matrix, we would need to use this as 'extra' information about the features dataset. This is not redundant data, rather, we can think of this as a different view of our dataset.
- We were not using ALL of the provided data, together. We had build models where we used only the features set, independent of the adjacency matrix, and vice-versa. We learnt that we had to come up with a model involving all of the provided data - features set + adjacency matrix. Thus, we would need to use CCA in our model. We would also need to use the sample seed labelling , which we eventually used as the initialization for the cluster centroids.
- We should build a model, where we feed the output of data into the next algorithm and so on till we finally reach a convergence. Earlier, we would fine-tune each part of the model separately, which was a BIG mistake. We wrote code which would have all the aspects in-place - reduction of dimensions,CCA with embedding as well as k-means so that we can tweak the parameters to obtain better results in the form of more evenly sized clusters. This approach helped us a lot to move from 94% accuracy to our final 99.81% accuracy, as we fine-tuned the feature set parameters to finally 8 dimensions after CCA, where we got more evenly-sized clusters.
- Our main problem with this method was that we didn't retrieve the majority of the information embedded in the spectral data by just calculating the first 10 eigenvectors of the normalized Laplacian matrix. A solution to this problem is by setting the number of eigenvectors as just another parameter, and fine-tuning it with other parameters to obtain a better model with higher accuracy.

6 Group Logistics

One of our group members, Jae Young (jc2379), joined our group after the preliminary write-up was due. He emailed Dr. Sridharan about this, and he has emailed his copy of the prelim write-up directly to Dr. Sridharan.

Jae Young has not made any submissions on Kaggle apart from submissions with our own group, in order to not break the Academic Code of Integrity.

References

[1] U. von Luxburg *A Tutorial on spectral clustering*, Statistics and Computing, 2007 Vol 17, 4