Jeannie Fu (jf586)
Nikita Gupta (ng354)
Richa Deshpande (rd382)
Tori Seidenstein (tbs52)

November 1, 2016

Machine Learning for Data Science: Competition 1

Overview

We tried several different approaches to the problem of classifying handwritten digits. We began by working with the individual data sets to form clusters, and then we worked to combine them and fine-tuning the parameters. As a rough measure of how accurate our models were, we looked at the sizes of each cluster and the agreement between the cluster assignments for the three seed points for each digit. The example below illustrates this as "Columns 1 through 10" show the size of each cluster and the chart below displays the cluster assignments for the three seeds of digit 0 up to digit 9.

```
>> eig10 = spectralOnly(A, seed, 10);

bin =

  Columns 1 through 8

       1561         873        1008        1649         980        3398         651         664

  Columns 9 through 10

          9        1207

  0: 1 1 6
  1: 8 6 3
  2: 1 4 6
  3: 6 4 6
  4: 1 6 5
  5: 4 6 6
  6: 6 1 6
  7: 10 1 1
  8: 1 10 9
  9: 6 6 10
```

We used seed assignments as only a rough measure of accuracy for our different models, and submitted some of our approaches to Kaggle to verify our accuracies.

For all approaches, we used our seeds to label our clusters as digits, as a form of partial supervised learning. We took the average of the 3 seeds for each digit as our initial centroid points for running k-means clustering. Based on which cluster the seeds were assigned, we labeled that cluster with the corresponding digit.

Approach 1: Spectral clustering on the adjacency matrix alone

Our first approach was to perform spectral clustering with k = 10 on the adjacency matrix alone. We selected spectral clustering because it is effective when the formation of clusters is based on the similarity between points and not any prior knowledge. Spectral clustering works on the adjacency matrix since it is is symmetric (if Point A is deemed similar to Point B, then Point B is deemed similar to Point A).

Explanation of Algorithm: Spectral clustering works by partitioning the data so as to "cut through" as few edges as possible when dividing the points between clusters. Therefore, it does not rely on any assumptions about the size or shape of the clusters.

Once we had reduced the data into 10 clusters, we used a voting approach to determine which digit each cluster represented since there was a lot of disagreement among our seeds. For example, if all the seed points for digit "6" are in cluster one, then cluster one represents the digit "6." After assigning all clusters for which the three seeds agreed, we assigned clusters where two of the seeds agreed, and proceeded using process of elimination. We only used this method with approaches that had very low accuracy since there was a lot of disagreement with the seeds. See below:

```
>> eig10 = spectralOnly(A, seed, 10);

bin =

  Columns 1 through 8

       1561         873        1008        1649         980        3398         651         664

  Columns 9 through 10

          9        1207

  0: 1 1 6
  1: 8 6 3
  2: 1 4 6
  3: 6 4 6
  4: 1 6 5
  5: 4 6 6
  6: 6 1 6
  7: 10 1 1
  8: 1 10 9
  9: 6 6 10
```

Figure 1. Command window screenshot showing cluster sizes and seed assignments.

However, spectral clustering on the adjacency matrix alone was not effective because there was a lot of disagreement, with the seeds being assigned to different clusters. In the screenshot above, we see that many seeds were assigned to cluster 6, and the sizes varied drastically. The cluster plot on the next page further illustrates how low our accuracy is when analyzing the adjacency graph alone.

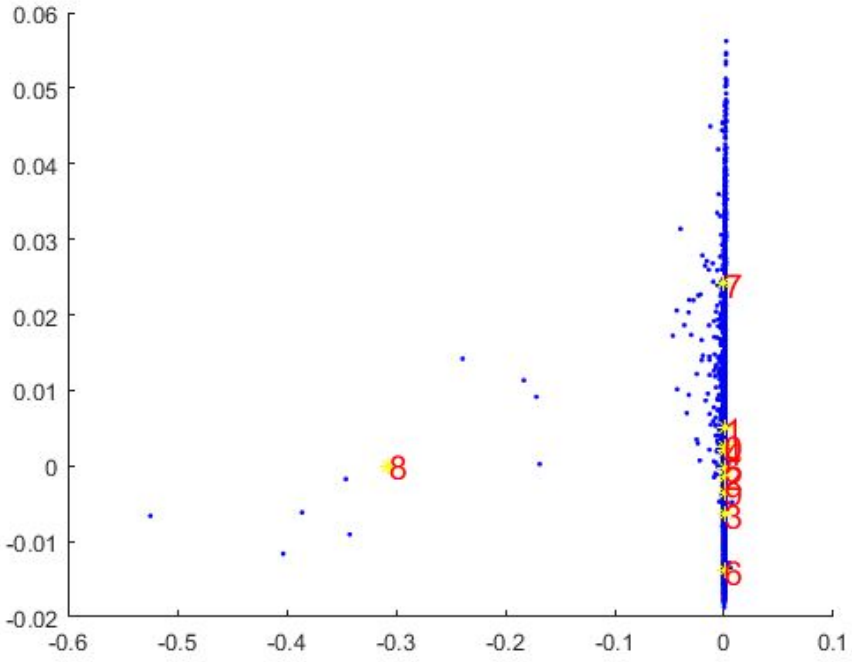

Figure 2. Spectral clustering only with 10 smallest eigenvectors.
The clusters are not easily distinguishable.

Approach 2: K-means clustering on the features matrix alone

Next we decided to do k-means on the features matrix alone. K-means is particularly effective when used on spherical, equal sized clusters and the competition stated that there is an equal distribution of each digit.

Additionally, k-means handles outliers better than alternate methods such as single-link clustering. Handwriting is a good situation in which this is valuable since someone with messy handwriting might make their 4's look extremely similar to 9's, for example.

Explanation of Algorithm: K-means is done in two iterative steps. First, it fixes the means and computes new cluster assignments for each point. Then it fixes the clusters that each point is assigned to and adjusts the means. This process is repeated until termination when the points do not change clusters anymore and a final assignment of points to clusters is complete.

Running k-means on the features matrix alone gave a 35.6% accuracy on Kaggle. We can easily see from our command window screenshot below that there was still a significant amount of disagreement between seeds in the clusters (though only a rough measure of accuracy), and we can see that the cluster sizes still vary substantially.

```
>> kmeansOnly (feats, seed)

bin =

  Columns 1 through 8

       1177        1323        2395         651        1186        1146        1150        1165

  Columns 9 through 10

       1162         645

  0: 1 8 4
  1: 10 2 3
  2: 5 8 3
  3: 3 4 4
  4: 5 5 3
  5: 6 6 6
  6: 7 7 7
  7: 3 3 3
  8: 9 9 8
  9: 3 9 1
```

Figure 3. Cluster sizes and seed assignments for k-means only.

Additionally, we can see from our plot below of clusters and labels why our accuracy was so low. Although points seem to cluster well, the centroids and labels did not group well.

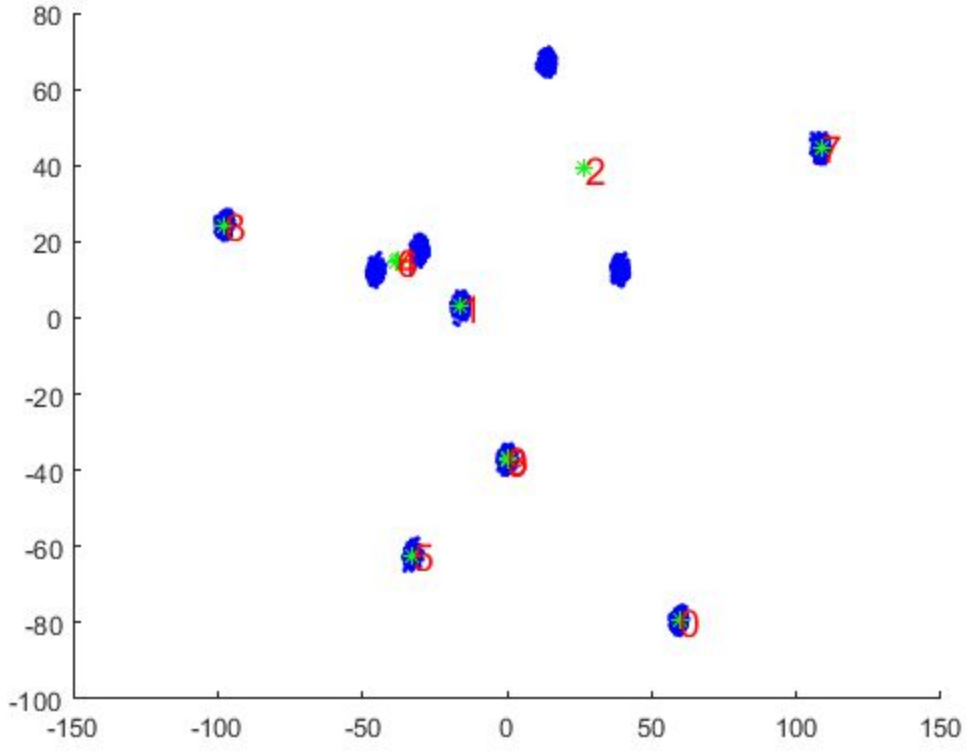

Figure 4. Cluster assignment for k-means only.

Approach 3: Combining spectral clustering on the adjacency matrix (10 eigenvectors) with k-means clustering on the features matrix using CCA with 10 dimensions

Next we decided to take an approach that utilized both the adjacency matrix and the features matrix since that provides more information on which to base the classification.

We performed spectral clustering on the adjacency matrix to get a spectral embedding based on the smallest 10 eigenvectors. Then we used CCA to combine the features matrix with the spectral embedding to reduce to 10 dimensions. Finally, we ran k-means on our combined data set (we decided to pick the dataset produced by CCA that was based on the features matrix).

Explanation of Algorithm: CCA is used to combine data from two views - such as visual and audio data or features and similarity in our case. It works by picking the direction of projection in each view for which the data is maximally correlated. One positive aspect of CCA is that because it maximizes the correlation coefficient, it is scale free. This means that the fact that the adjacency matrix is all ones and the feature matrix has numbers with much larger magnitudes does not throw off the dimensionality reduction.

However, this approach performed worse than before (32.9%). This is perhaps because we did such a drastic dimensionality reduction before CCA that CCA had a very difficult time finding the correlated columns in the two views. In the below screenshot, we see that cluster sizes were relatively the same but seeds disagreed.

```
>> combinedData(feats, eig10, seed, 10)

bin =

  Columns 1 through 8

       1132        1212        1195        1203        1074        1261        1180        1318

  Columns 9 through 10

       1181        1244

  0: 7 8 4
  1: 2 8 5
  2: 10 2 6
  3: 1 6 2
  4: 5 6 5
  5: 6 10 10
  6: 4 1 7
  7: 4 4 8
  8: 8 9 9
  9: 8 10 10
```

Figure 5. Cluster sizes and seed assignment for approach 3.

In the plot below, we can see that this method did not separate the clusters very well at all.

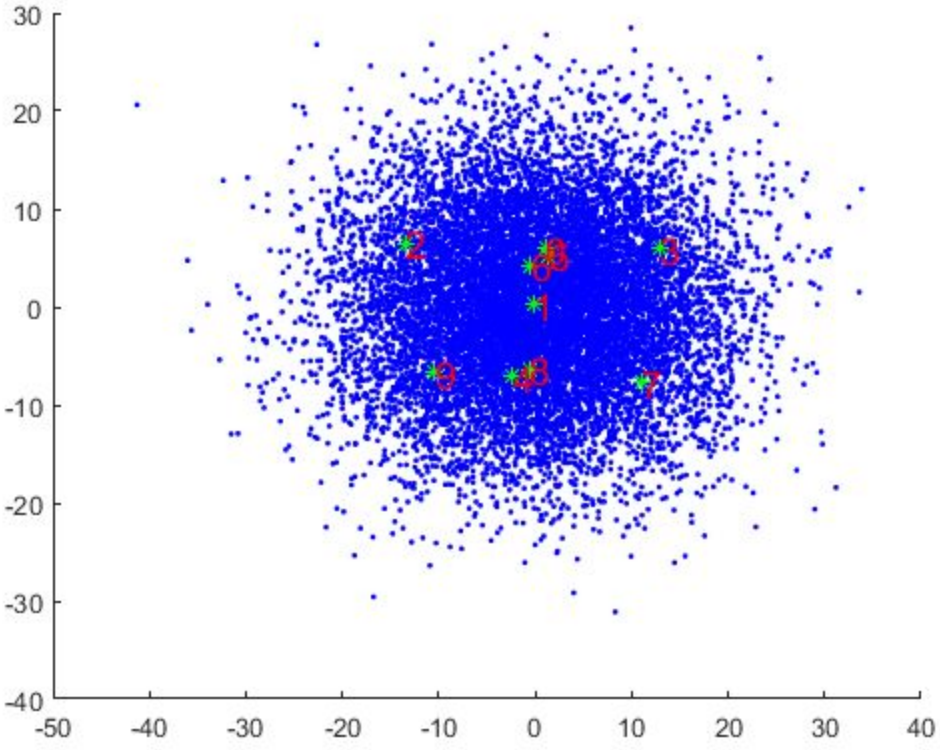

Figure 6. Clusters for approach 3

Approach 4: Combining spectral clustering on the adjacency matrix (150 eigenvectors) with k-means clustering on the features matrix using CCA with varying dimensions

To solve the previous problem, of giving CCA too little to work with, we decided to create a spectral embedding based on 150 eigenvectors instead of 10. Then we experimented with different dimensions for CCA (10, 2, 3, 4, 5, 6, 7) and found that 3 had the highest accuracy at 90.4%. See the following page for how we decided to sample these different dimensions.

How we found the dimension:

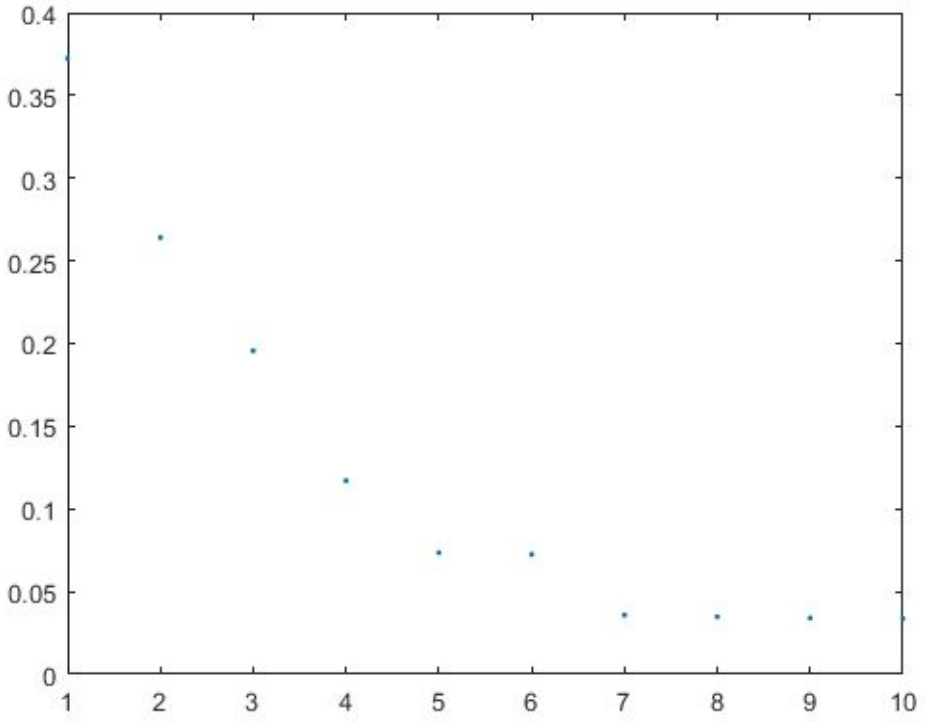

Figure 7. Largest 10 Eigenvalues for CCA with spectral embedding (150 eigenvectors) and features data set.

We first tried reducing to 10 dimensions again for CCA, but then decided to use this graph in order to determine the appropriate value for k (dimension) for CCA. We chose a k based on a cutoff for when the points in the graph start to level off or "plateau". We tried several k's where we thought the k might be from k=2...7.

We found that k = 2 and k = 3 as a cut off resulted in the best separation of clusters in the plots on the following page. Looking at relative cluster sizes and seed agreement and using Kaggle, we found k = 3 was the best, with 90.4% accuracy.

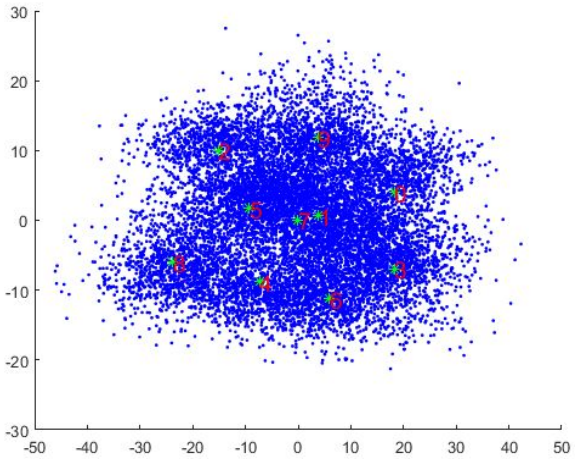

Figure 8. Spectral clustering (150 eigenvectors) with CCA with 10 dimensions

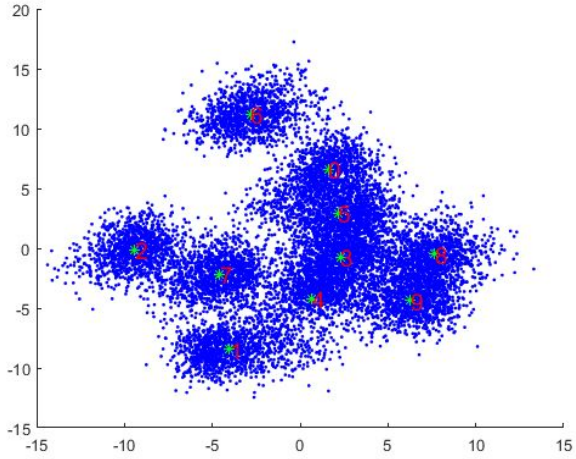

Figure 9. Spectral clustering (150 eigenvectors) with CCA with 2 dimensions

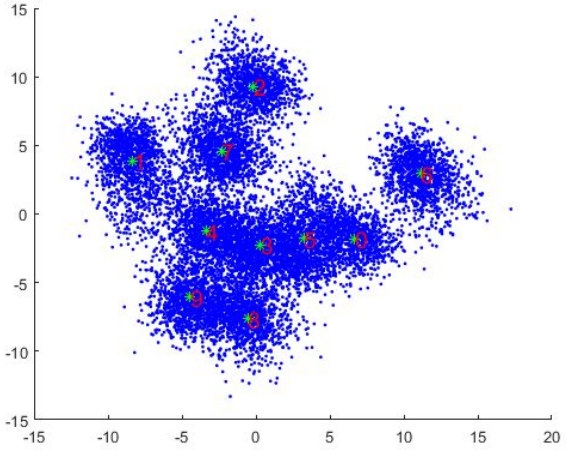

Figure 10. Spectral clustering (150 eigenvectors) with CCA with 3 dimensions

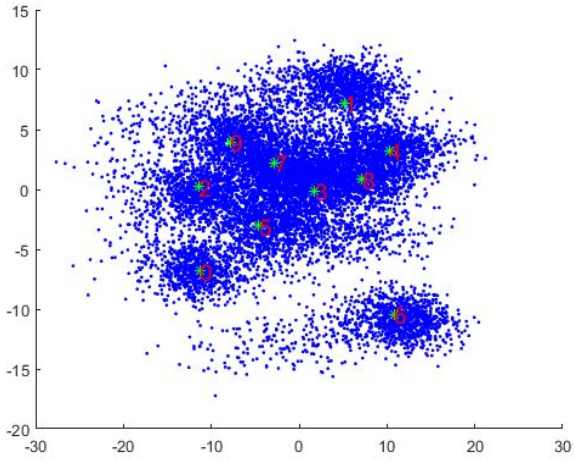

Figure 11. Spectral clustering (150 eigenvectors) with CCA with 4 dimensions

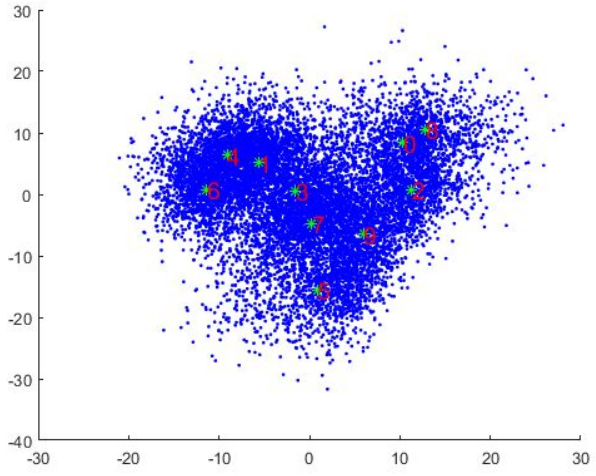

Figure 12. Spectral clustering (150 eigenvectors) with CCA with 5 dimensions

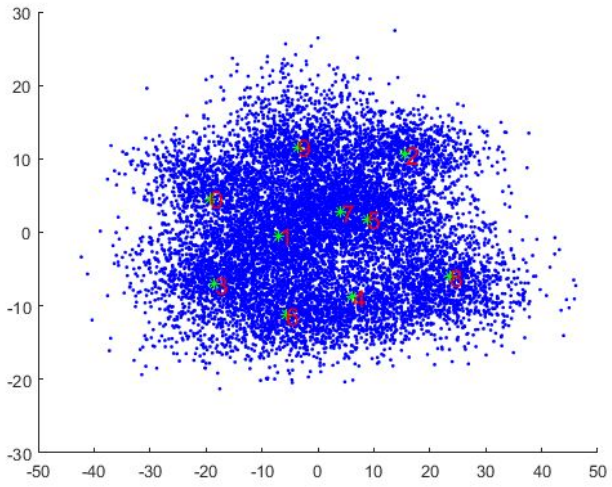

Figure 13. Spectral clustering (150 eigenvectors) with CCA with 6 dimensions

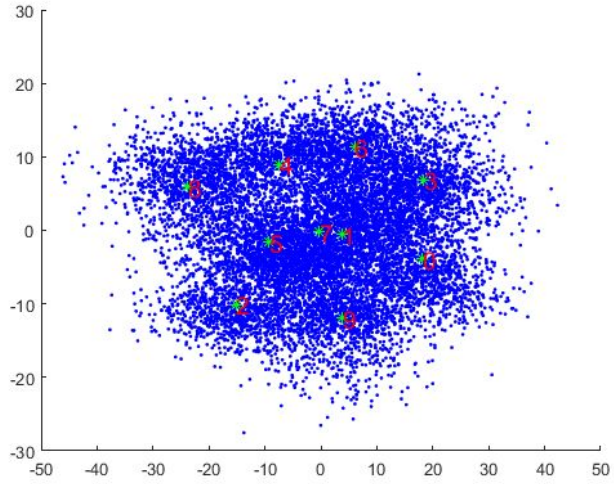

Figure 14. Spectral clustering (150 eigenvectors) with CCA with 7 dimensions

Approach 5: Combining spectral clustering on the adjacency matrix (250 eigenvectors) with k-means clustering on the features matrix using CCA with 3 dimensions

To make CCA even more effective, we used spectral clustering to produce 250 eigenvectors and then conducted CCA with the 3 dimensions, as determined in the previous experiment. This produced 92.6% accuracy.

With this high accuracy, the assignment of digit to cluster is a direct mapping. Since we provide initial centroids based on the seeds, cluster 1 corresponds to digit “0”, cluster 2 corresponds to digit “1,” etc.

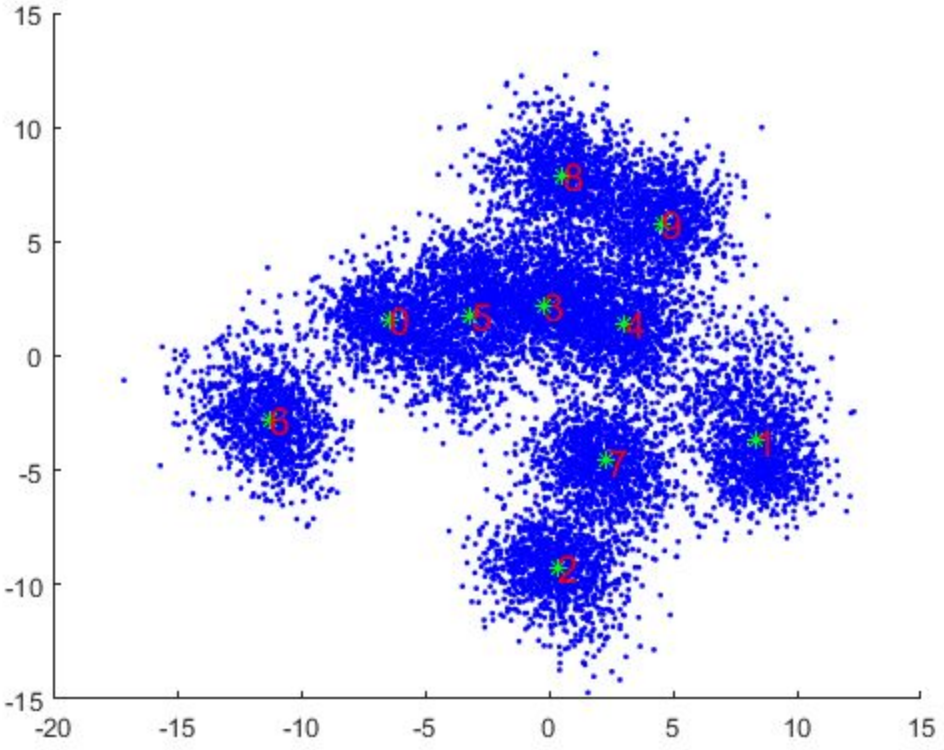

Figure 15. Spectral clustering (250 eigenvectors) with CCA with 3 dimensions

Other Failed Attempts

To refine our number of eigenvectors for spectral embedding, we tried to use the plot below which shows a graph of the smallest 250 eigenvalues. We attempted to use this to determine the appropriate value of k for the spectral embedding. The proper k should correspond to the cutoff of where the points start to plateau in this graph. However, by trial and error of several appropriate k's we did not get a reasonable result from the clusters. We tried numbers ranging from 7-20, 150, and 250. We used our clusters with 250 as it was the best result.

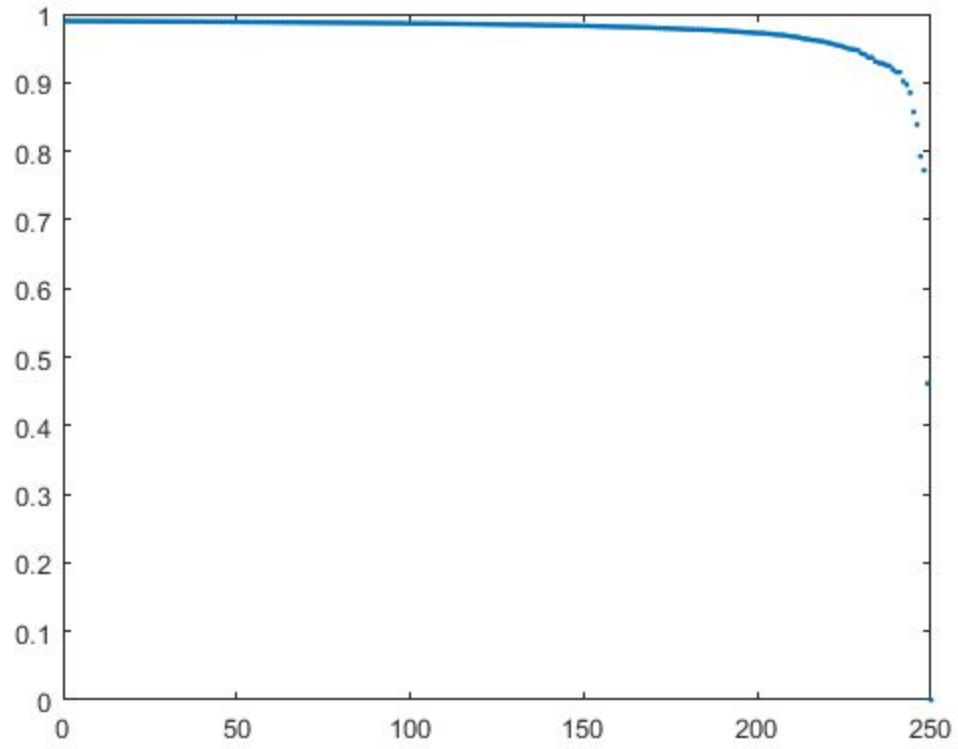

Figure 16. Smallest 250 eigenvalues for spectral embedding

Summary

Our final Kaggle submission was approach #5: using the adjacency matrix to create a spectral embedding based on 250 eigenvectors, combining the embedding with the features matrix using CCA to reduce dimensionality to 3, and running k-means on the combined data.

Visuals helped us develop our model in several ways. First, they provided insight into the effectiveness of each approach at a general level. For example, were points not being differentiated into distinct clusters? Were the centroids not matching up with the clusters? The plots also helped us answer specific questions such as which k value should select for CCA?

We used a combination of knowledge about the pros and cons of various algorithms, visual indicators, and iterative experimentation to arrive at our final solution.