CS 4786 Machine Learning Competition I

Shuying’s Part

**How did we conduct local check apart from submitting on Kaggle every time?**

1). As we were given 30 seeds that we have already known the clustering results, thus we want to check the accuracy with seeds and take a first glimpse of whether the 30 seeds are being grouped in the right clusters to indicate the potential success of our clustering model. However, later we found that a higher accuracy for seeds does not necessarily mean a higher score on Kaggle. For example, in one submission, we hit 29 out of 30 of accuracy on the seeds, but got a pretty poor score 10% on the Kaggle. The reason here might be the number of 30 seeds is still relatively too small to be a direct indication of how good our clustering model, as randomness is relatively large here. Also it much likely that our failed model led us to cluster the 12000 points in a very wrong way and the seeds themselves might be chosen intentionally.

2). Also we were given the information that the frequency of each digit in features.csv is roughly the same. So we want to calculate the frequency for each digit in our clustering result.

*(Right now, I haven’t tried it yet, if someone wants to try, that would be great.)*

1. **K-means**

**1.1 K-means on the raw data** (37% on Kaggle, accuracy = 15/30.)

We ran k-means clustering on the raw dataset(features.csv), and initialized the 10 centroids to be the average of the 3 seeds points for each digit.

**1.2 Why PCA is not a good model here?**

Normalization Vs Non-Normalization before PCA

Non-Normalization: k=3, centroid=average of seeds; accuracy=16/30; score on Kaggle = 37%;

Firstly, we performed PCA without normalizing the features.csv and reduced the dimensions to 3. The reason for choosing k=3 here is that it was noted that after we sorted the eigenvalue of covariance matrix in descending order, the first three eigenvalues are in thousand scale while the other 100 are in relatively smaller scales. So, we think there are three principle components capturing most of the variances in the features.csv.

Normalization: k=3, centroid=average of seeds; accuracy=29/30; 10% on Kaggle;

As we noticed in the features.csv, some dimensions have a broader range of values. Since k-means need to calculate the Euclidean distance between two points, we worried that the distance would be governed by those particular features who have larger range. Thus we choose to normalize the features.csv so that each dimension can contribute approximately proportionately to the final distance.

However, it is interesting to note that after we normalizing the data before PCA, the score on Kaggle dropped so much and went down to 10%.

After all, from the PCA, although we are not sure how many dimensions can capture most of the features of the original dataset. One of the significant finding was that, despite values are in different scales in different dimensions, those different scales are a good reflection of the original dataset.

**1.3 Refine Our Way of Conducting K-means:**

1). As we know that, K-means clustering is very sensitive to the initial chosen centroids. At the very beginning, as mentioned in the preliminary report, we have initialized each of the 10 centroids to the average of the 3 seed points for each digit. After that, we refine our way of choosing the centroids given the need of getting more accuracy by running k-means multiple times. And on each iteration, for every digit, randomly choose a neighbor point that is with the top 100 closest points of one of the three labeled points, also randomly choose the labeled points.

2). Run k-means multiple times (1000 times) with different initial centroids chosen as stated before.

3). After running k-means 1000 times, for each point, choose the label which is the most frequently occurring label as the clustering result.

**1.4 Random Projection on features.csv before K-means (11% on Kaggle, accuracy=16/30)**

Both Random Projection(RP) and PCA are used to conduct projection from high-dimensional space into low-dimensional space. Since PCA does not perform that well, after trying PCA, we might think that our original data does not lie in a low dimensional space.

The possible loss in accuracy derives from the fact that PCA is the optimal linear transformation in the least-square sense. On contrast, RP is picking dimensions randomly, and RP are motivated by their proven ability to preserve inter-point distances.

We performed RP on features, reduce dimension to 3, and then did k-means 1000 times, setting centroids as the random points from 100 closest neighbors for each random digit seed. However, RP didn’t generate a good performance.

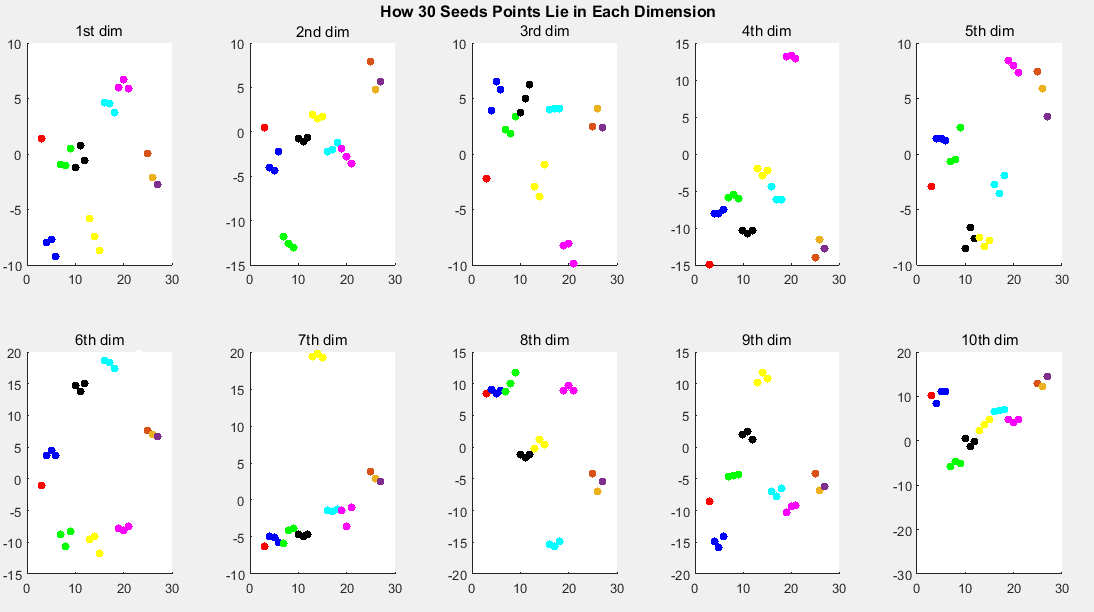
The failure might indicate we still need to pay attention to some real features in this dataset. A random projection direction does not describe the data in low dimension space very well.

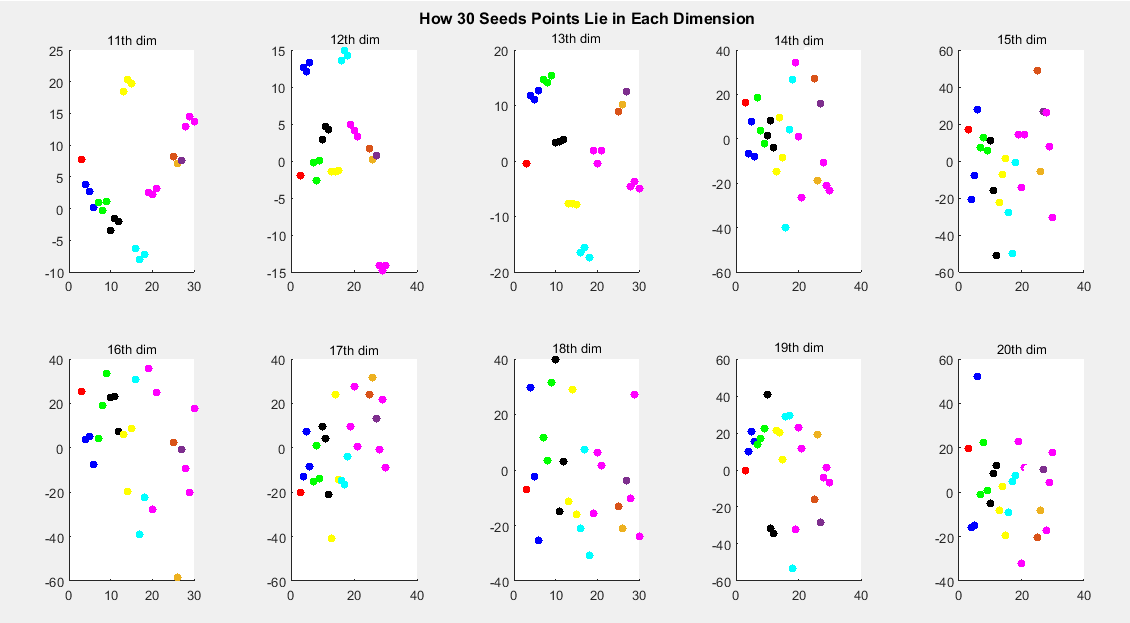
1. **Picking the Right Dimensions**

After many failed attempts described above, the team members started to wonder the potential unique characteristics of the original data.

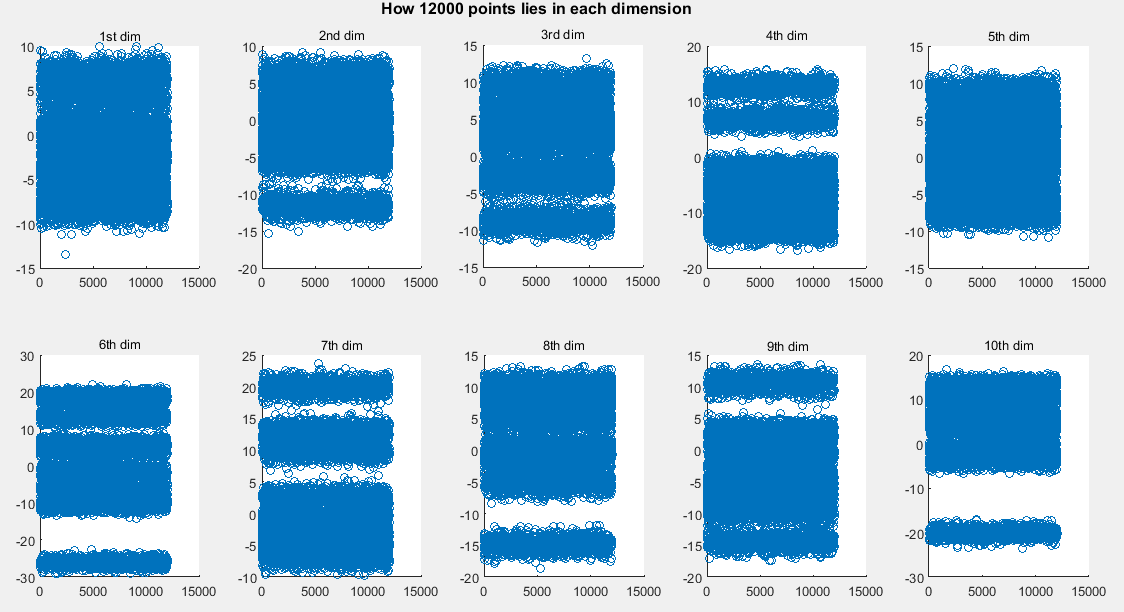
**2.1** **Extract Information from features.csv and seed.csv**

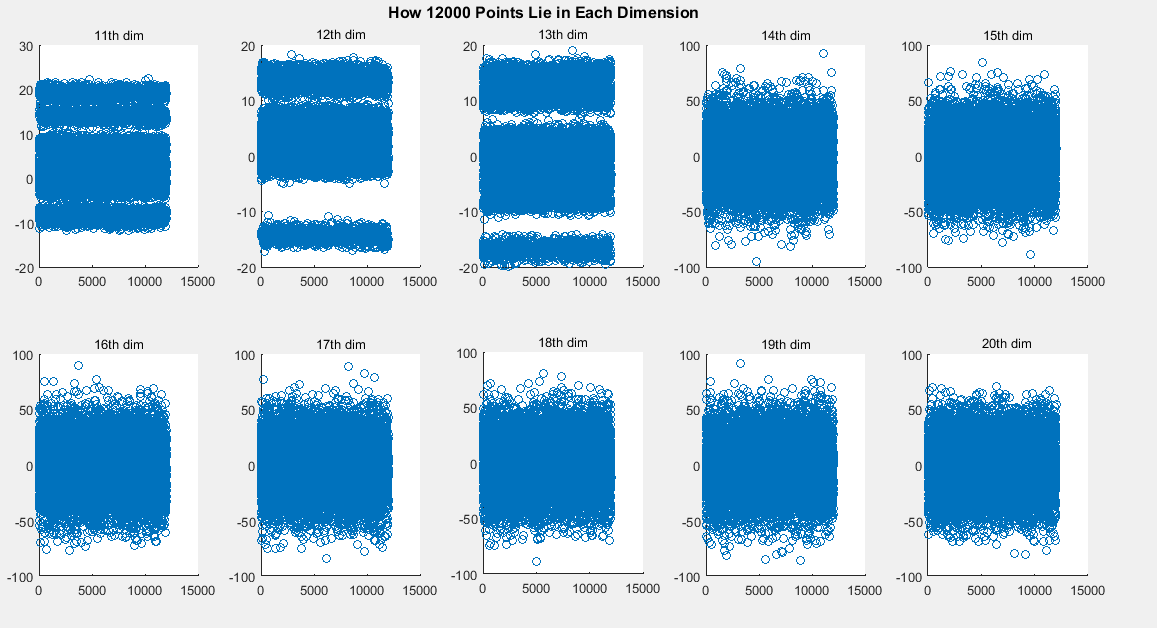
The team members wanted to visualize how the 10 clusters of 30 seeds lie in every dimensions. In order to do this, we made effort to distinguish every three seeds in the same cluster by using the same color. We first exact the 30 seed points from features.csv, and then plotted 30 seed points for each one of the 103 dimensions, there was a significant finding that in the first dimensions, every 10 clusters are well separated in that particular dimension, while in the 13th -103th dimensions, the 30 points are scattered distributed in each dimension.





In order to verify our finding, we continued to plot all the 12000 points for each dimension. From that, we can tell the first 13th dimensions are special in terms of the distribution of points. While in the 14th to 100th dimensions, data may just be in norm distributions.





**2.2 The First 3 dimensions VS. the First 13 dimensions**

When we tried both tried using just the first three columns or just the first 13 dimensions to do K-means as before, the results on Kaggle differs so much. For the first 3, it gives us the highest score 99%, while it is interesting the first 13 dimensions only gives us around 10% on Kaggle. And same failed attempt for just picking 4th-13th dimensions of the original dataset.

We guess maybe the 4th-13th dim may just the determinant of what the spatial distribution characteristics of the original data. However, the first three dimensions are the dominating features of the dataset in terms of clustering.

Why? - I still not that clearly know

**2.3 Extract Information from features.csv and Ajacency.csv---Conduct CCA during the Step in Spectral Clustering**

In spectral clustering, we got 10 eigenvectors corresponding to the top 10 eigenvalues, we multiplied Adjacency matrix by that eigenvector matrix, we can get an 12000\*10 matrix. Then, we performed CCA on the that matrix as View1 and the features.csv as View2 to get the coefficient between the 10 dimensions in View1 and 103 dimensions on View2.

It is important to find that coefficient between the first three dimensions in View2(features.csv) are highly correlated with 10 columns in View1.

Need some further explanations here.

**Final Clustering Model**

Right now, we can say that the first three dimensions determine the features of the original data well enough, while the rest 100 dimension may just be noise that does not help distinguishes the differences between points. In this way, we just choose the first three dimensions of the features.csv as the feature selection of the original datasets.

Need some further explanations here.