Implementation Assignment #4

Alireza Mostafizi

Introduction

I coded this assignment in MATLAB. Attached to this report, you can find the MATLAB codes. The assignment is to implement 1) Kmeans clustering algorithm, 2) assess the impact of Principal Component Analysis (PCA) dimension reduction algorithm on the Kmeans clustering, and 3) assess the improvement of Linear Discriminant Analysis (LDA) algorithm on the Kmeans clustering.

The data includes 2200 handwriting images of number 7 and 9 (1100 each) from USPS database. Each image is a 16x16 matrix which leads to 256 features. Figure 1 shows 5 example for each of the categories. The objective of this assignment is to code a classifier that differentiates these digits from each other.

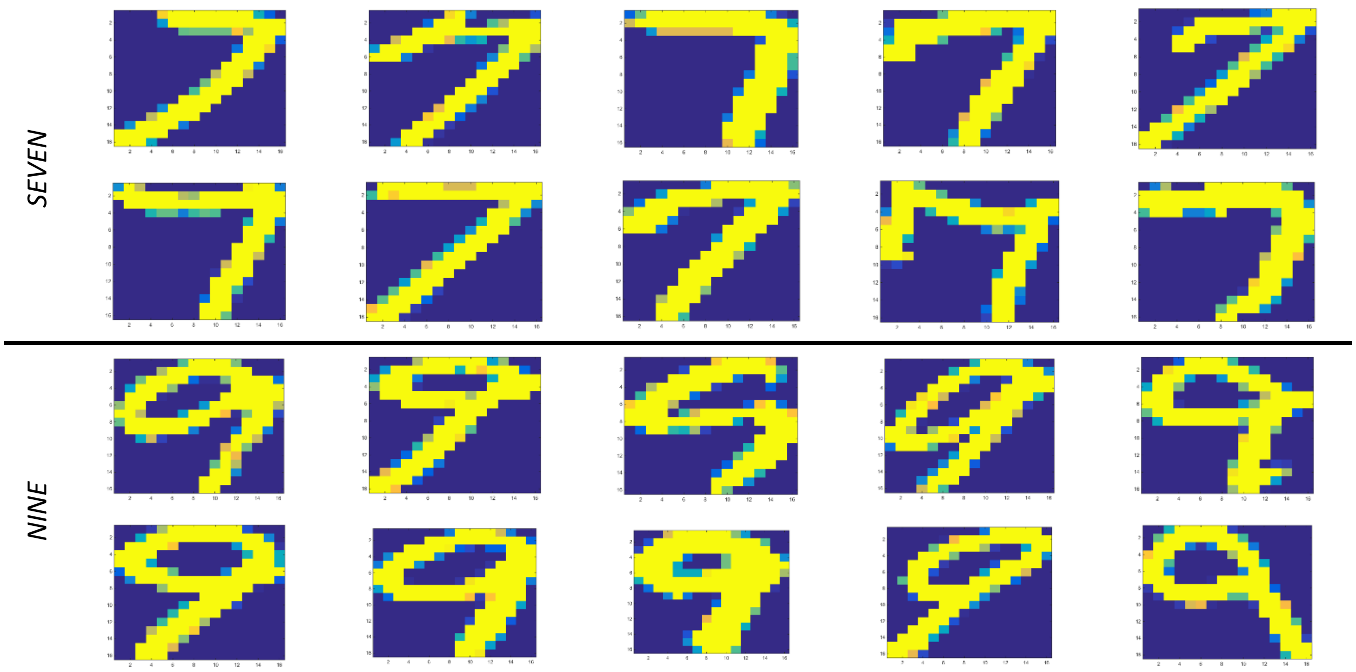


Figure 1. Examples of digits

In this report, each part of the assignment will be addressed in the following separate sections.

Part I: Kmeans Algorithm

The first part of the assignment is to apply Kmeans algorithm on the raw data which has 256 features.

Methodology

As we know that the data is classified into two categories, the K is equal to 2. The first step in the algorithm is to initialize two random vectors with the size of 256 as the mean features of two classes. Then we assign instances to the closest class, and recalculate the mean vector. At the end the majority of instances in the class determines the class label. The algorithm can be summarized as following:

- Initialize two cluster centers randomly within the range of data points
- Iterate through this steps
  - Assign each data point to the closest cluster center
  - Re-calculate the mean of the cluster with the following formula:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

- Until none of the instances change their cluster membership
- Assign the predicted label to the class based on the majority of instances

The purity of the classification can be calculated using the following formula, using ground truth labels:

$$Purity = \frac{Number\ of\ correctly\ classified\ instances}{Total\ Number\ of\ instances}$$

Results

Doing the above algorithm for 100 times, and averaging the $Purity$ over 100 iterations, the mean accuracy of the Kmeans algorithm on the raw data and without any dimension reduction technique is calculated to be:

$$\boxed{Accuracy = 50.71\%}$$

Discussion

This shows that the classifier is almost acting randomly, and it is not really classifying anything. This can be due to the facts that:

- The classes are fairly close together. As you also can see in figure 1, the images are pretty similar from classification point of view.
- The classes share common features that does not help the classification problem.
- Within each class, instances can be far away from each other due to the diversity of handwritings.
- There are lots of noises in the data, including the position of the digit in the 16x16 image, the direction of the tails, the angle of the handwritten digit, etc.

All in all, it can be concluded that Kmeans algorithm without dimension reduction, does not work for classifying these two digits.

Part II: Principal Components

In this part of the assignment, I computed the principal components, and assessed the relative variance of the data that can be explained by each component.

Methodology

The first step in calculating the principal components is to find the covariance matrix of the data centralized data using the following formula. Most often, It is also suggested to normalize the data as well, however, normalizing the data, in this case, did not change any of the results.

$$S = Cov(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

Where $n = Total\ number\ of\ instances$

$x_i = ith\ instance\ (represented\ as\ 256\ by\ 1\ matrix)$

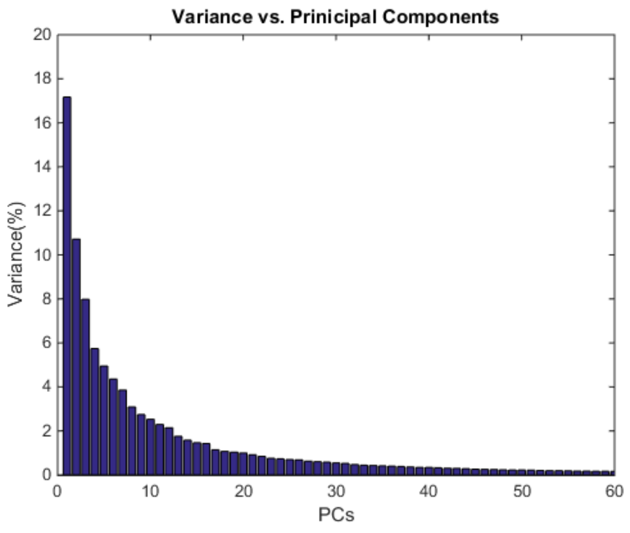
$\bar{x} = mean\ vector\ of\ features\ over\ all\ the\ instances\ (256\ by\ 1\ matrix)$

The next step is to calculate Eigen-vector and Eigen-values. I used MatLab function, "eig," for this task. Eigen-values and the corresponding Eigen-vectors were sorted in descending order to find the principal components. The relative amount of the variance that can be captured by each principal component, and be calculated with the following formula, using the Eigen-value associated with the component.

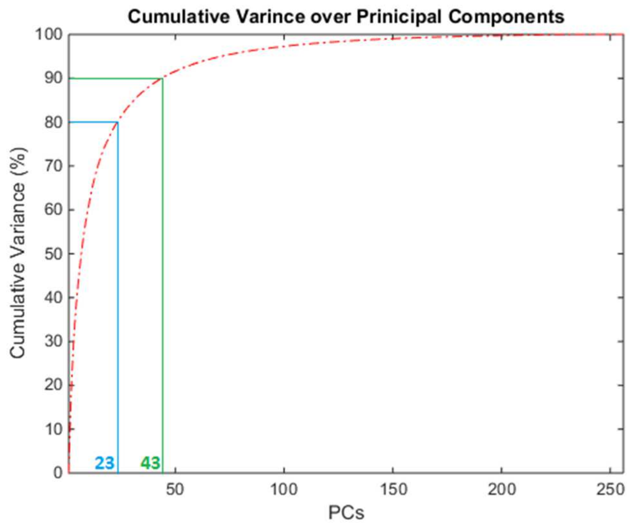
$$Relative\ Amount\ of\ Variance\ captured\ by\ PC_i = \frac{Eigen - Value_i}{\sum_{i=1}^{256} Eigen - Value_i} * 100$$

Results

Figure 2a shows the amount of variance that can be captured by principal components. In addition, you can see the cumulative variance that can be explained by the principal components in Figure 2b.



(a) Relative Variances



(b) Cumulative Variance

Figure 2. Relative Variance and Cumulative variance over the principal components

Discussion

As you can see in figure 2a, the first component can explain roughly 17 percent of the variation of the data. The second and third principal components can explain almost 11 and 8 percent of the variations in the data respectively.

Looking at Figure 2b, and focusing on the data points marked by blue and green, we can state that, 23 is the smallest dimension that we can reduce to if we want to retain at least 80% of the total variance. In addition, to maintain 90% of the variation in the raw dataset, the smallest dimension that we can reduce to is 43. Table 1, summarizes the results.

Lowest Dimension	The amount of variation captured (%)
1	17
6	50
23	80
43	90
70	95
148	99

Table 1. The lowest dimensions for corresponding variations

The results show that with almost 150 features, we are able to explain roughly 99 of the variations in the data. However, that does not mean that the principal components have high discriminative value, since high variances does not necessarily refer to higher discriminative aspects.

Part III: PCA Dimension Reduction

In this section, I reduced the data to 1, 2, and 3 dimensions, and applied Kmeans algorithm on the reduced data. Then I compared the performance of the algorithm on the reduced data to that of on the raw data calculated in the first section.

Methodology

After calculating the principal components in previous section, the dimension reduced data can be calculated with the following matrix calculation.

$$Z = Xv$$

Where $X = Raw\ Data\ (Represented\ as\ 2200\ by\ 256\ Matrix)$

$$v = Eigen - Vector\ (Represented\ as\ 256\ by\ either\ 3, 2, or\ 1\ matix)$$

$$Z = Dimension - Reduced\ Data\ (Represented\ as\ 2200\ by\ either\ 3, 2, or\ 1\ Matrix)$$

After reducing the dimension of the data, Kmeans algorithm has been applied on the new dataset. The results of clustering are represented in the next sections. Please note that the data has been reduced to 3, 2, and 1 dimensions.

Results

- *1 Dimension*

Figure 3a shows the result of the dimension reduction approach to 1 dimension along with the true labels of the data. In addition, Figure 2b shows the results of the Kmeans algorithm along with the predicted class labels for instances.

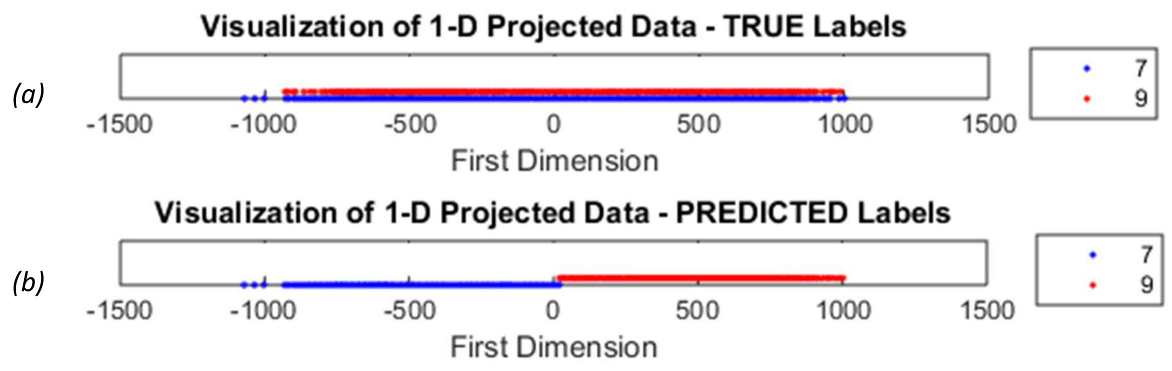


Figure 3. 1-D Projected Data along with True and Predicted Labels (PCA)

As you can see the projected data has high variance, ranging from -1000 to 1000, however, it does not have good discriminative value. You can see in Figure 3a that both categories are spread out to the whole range of the projection, which makes the classification impossible. Comparing this to Figure 3b, we see the performance of the Kmeans classifier which very poorly has classified the data.

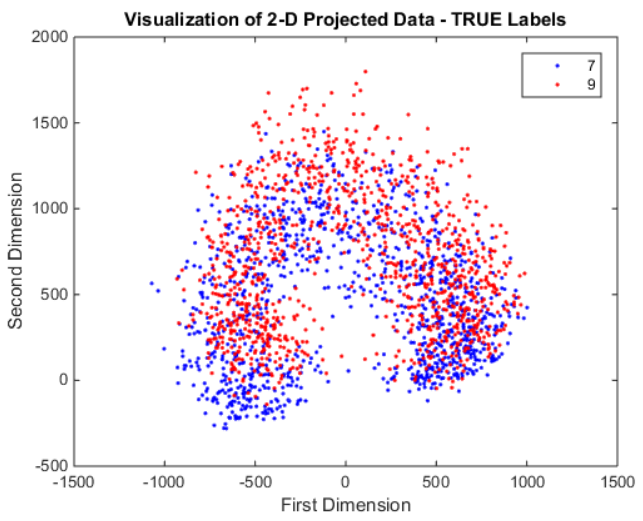
The performance of Kmeans algorithm on the 1-D projected data over 100 trials is as following:

$$\boxed{Accuracy_{1-D} = 50.21\%}$$

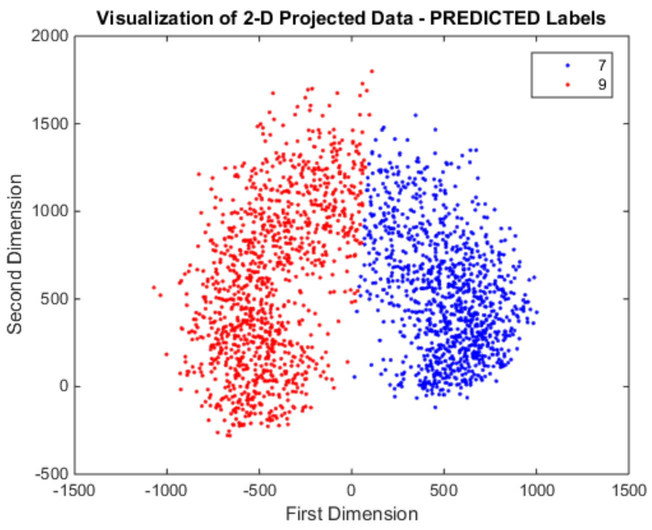
You can see that the accuracy is even worse than the accuracy of the algorithm o the raw dataset. This might be because of neglecting some of the features that have low variations, but higher discriminative values.

- *2 Dimensions*

The same problem happens in the 2 Dimensional data projection scenario. Figure 4a shows the 2-dimensional projected data, along with the true class labels. As you can see, both classes totally overlay each other, and Kmeans algorithm is expected to fail classifying. Figure 4b also shows the predicted class labels coming from Kmeans algorithm. You can see that the predicted labels are highly different than the true class labels. This is another example of the cases, where data is projected in a direction that leads to high variations, however, the projected direction does not generate any significant discriminant feature.



(a) True Labels



(b) Predicted Labels

Figure 4. 2-D Projected Data along with True and Predicted Labels (PCA)

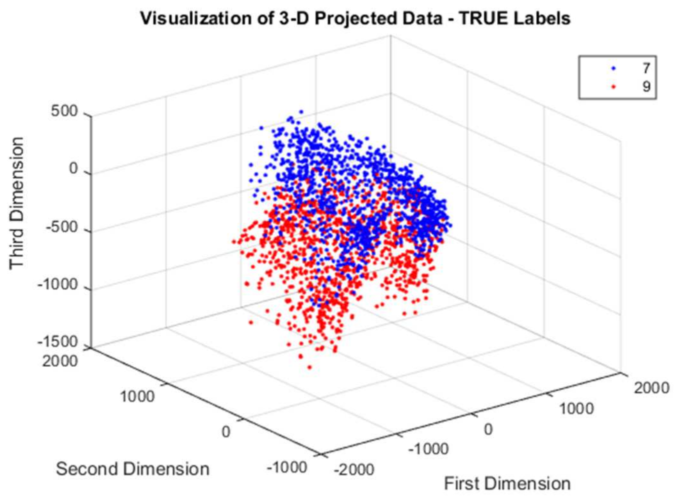
The average performance of Kmeans algorithm on 2 dimensional reduced data over 100 iterations is as following:

$$\boxed{Accuracy_{2-D} = 50.68\%}$$

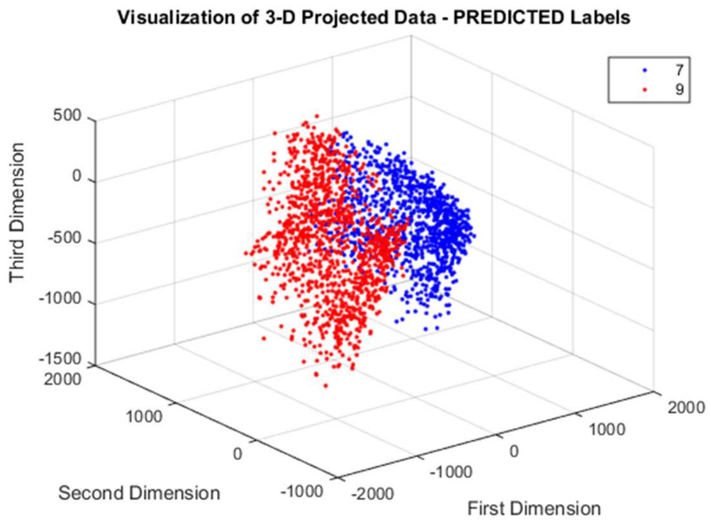
You can see that the classifier still performing randomly. The performance is a little bit better than the 1-Dimensional data, but still worse than the performance of the algorithm on the raw data.

- *3 Dimensions*

Likewise the other two cases, two categories totally overlay in 3-D projected data as well. Figure 5a shows the projected data into 3-dimensional space along with their true class labels. Figure 5b on the other hand, shows the predicted class labels using Kmeans algorithm.



(a) True Labels



(b) Predicted Labels

Figure 5. 3-D Projected Data along with True and Predicted Labels (PCA)

Again, the predicted class labels is not anywhere close to the true labels. The averaged accuracy of the Kmeans algorithm on 3-dimensional projected data over 100 trials is as following:

$$\boxed{Accuracy_{3-D} = 50.70\%}$$

It is slightly better than 2 dimensional projected data, and its performance is fairly close to that of raw data. However, still classifier is performing randomly, and it does not lead to acceptable levels of accuracy.

Discussion

As we saw, the projection of the data to 1-, 2-, and 3-dimensional space does not have any improvement of the performance of the classifier compared to that of on the raw data. This is one of the cases that higher variations does not necessarily lead to higher discriminative values. Projection of the data leads to missing some important information, in terms of classification, lied in the features with low variations. Table 2 summarizes the results of this section.

Projection Dimension	The Kmeans Accuracy (%)
1	50.21
2	50.68
3	50.70
256	50.71

Table 2. The accuracies for different projection dimension

Figure 6 also shows the accuracy of the algorithm on projected dataset to different dimensions. As you can see there is no improvement on the classification accuracy. The accuracy slightly increases as you increase the dimensions, and it remains almost constant as the dimension increases to the dimension of the raw data.

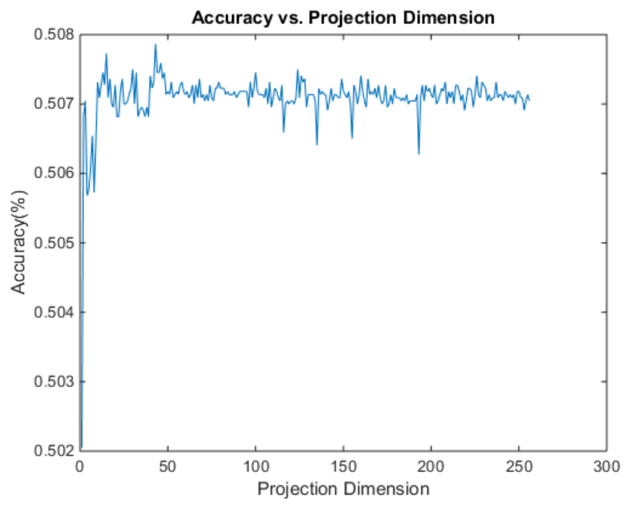


Figure 6. Accuracy vs. Projection Dimension

Part IV: LDA Dimension Reduction

For this part, I tried a supervised approach of dimension reduction using Linear Discriminant Analysis, and I compared the results of Kmeans algorithm on the reduced data.

Methodology

To compute the best projection direction that classifies the data, first we have to compute the mean feature vector of each class. To do so, I separated the data points in two datasets and computed the mean of each dataset separately. The next step is to calculate the within-class scatter matrix which can be calculated with the following equation:

$$S = \sum_{i \in Classes} \sum_{x \in c_i} (x - m_i) * (x - m_i)^T$$

Where $m_i = The\ mean\ of\ instances\ in\ class\ i$

$C_i = Class\ i$

Then the projection vector can be calculated as following:

$$w = S^{-1} * (m_1 - m_2)$$

Projecting the data to this direction leads to a 1-d dataset. Applying Kmeans algorithm on the resultant data, we get the results as presented in the next sections.

Results

Figure 7a shows the true label of the instances, projected using the projection vector calculated in the previous section. You can see that the data is perfectly separated, and therefore, the projections is highly discriminant. Figure 7b shows the predicted label of the data using Kmeans algorithm. As explained by the figures, most of the data points are correctly classified, except the ones in the middle where two datasets overlap.

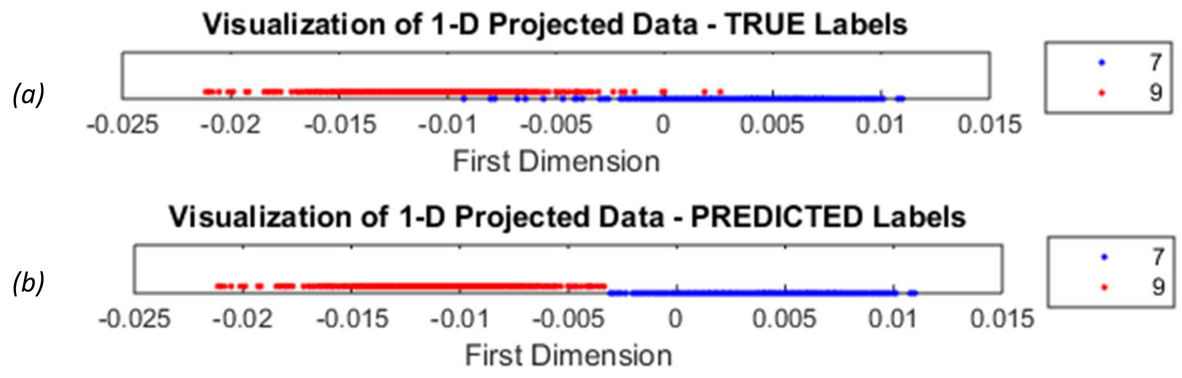


Figure 7. 1-D Projected Data along with True and Predicted Labels (LDA)

As expected the accuracy of this approach is very high since it is a supervised dimension reduction technique. The accuracy of this approach is calculated as following.

$$\boxed{Accuracy_{LDA} = 99.14\%}$$

Discussion

As we would expect, the accuracy of this supervised dimension reduction algorithm is very high, since it captures the discriminant features, rather than just projecting the data in a direction that leads to higher variations.

Part V: Discussion

As we saw in the results, PCA dimension reduction technique did not work very well. This is mainly due to the facts that

- PCA is known not to be optimal for classification purposes
- The logic behind PCA, projecting to the highest variation direction, does not necessarily leads to high discriminant features.
  - Especially in this case, the variations are caused by different handwritings, not by the difference between the digits.
- In addition, throwing out some features with high discriminative value, but with low variance, is not a good idea.
  - Throwing out the features that have low variation due to the difference between the digits
- Differentiating handwritten digits is hard by its nature, so having unsupervised learning algorithm in this case is not the best idea.
  - The classes are fairly close together and images are pretty similar from classification point of view.
  - The classes share common features that does not help the classification problem.
  - Within each class, instances can be far away from each other due to the diversity of handwritings.
  - There are lots of noises in the data, including the position of the digit in the 16x16 image, the direction of the tails, the angle of the handwritten digit, etc.

These are the reasons that justify that PCA is not an appropriate choice in this case. On the other hand, supervised machine learning, and specifically LDA approach that projects the data to the direction that has the most discriminative information seems to be the best approach.

Notes on running the code:

1. Main code is “Main.m,” and the others are functions.
2. Run the “Loading Data” part in the main code first
3. Run the code associated to each part of the assignment separately to get the results and graphs. The “Main.m” code is categorized in 5 parts.
   a. Loading Data
   b. Part 1 (Kmeans)
   c. Part 2 (Eigen-Values and Eigen-Vector)
   d. Part 3 (PCA)
   e. Part 4 (LDA)