Report for Assignment 1

ECE 657A
Group 11
Shala Chen, 20698485
Jinghang Wang, 20676432
Zhao Zhang, 20689699

February 12, 2017

Tools we use: MATLAB
Version: 2014b or 2016b
We used libraries in in MATLAB for Part1 and Part2, besides, we used drtoolbox for Part3 and Part4. Since the drtoolbox had some functions with the same name as libraries in MATLAB, so it is needed to set the drtoolbox as the lowest priority compared with other toolboxes in MATLAB.

1 Part I

1.1 Detect problems in dataset A

For dataset A, we detected the missing values and outliers

(1) Missing values

In dataset A, missing values was represented by NaN. So we calculated numbers of missing values for each feature. And then represented percent of missing values for each feature.

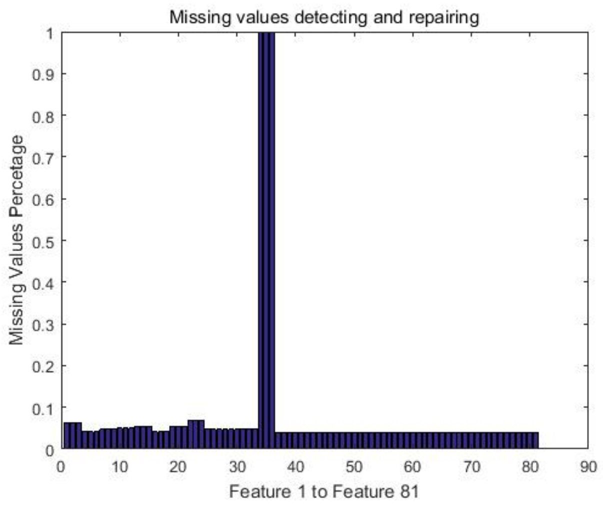

Figure 1: The percentage of missing values for each feature @data set A

From fig.1 we could see that in dataset A, feature #34 #35 and #36 have around 90% missing values. So we deleted the values of feature #34 #35 and #36. There were less than 10% missing values for other features, we replaced them with mean values of each feature.

(2) Outliers

After repairing all the missing values, we detected the outliers for each feature using hampel function. Since the datasets were time-based series, we used slide windows to detect outliers. We set the window size to be 20. Fig.2 shows the number of outliers we detected for each feature when we set the window size to be 20.

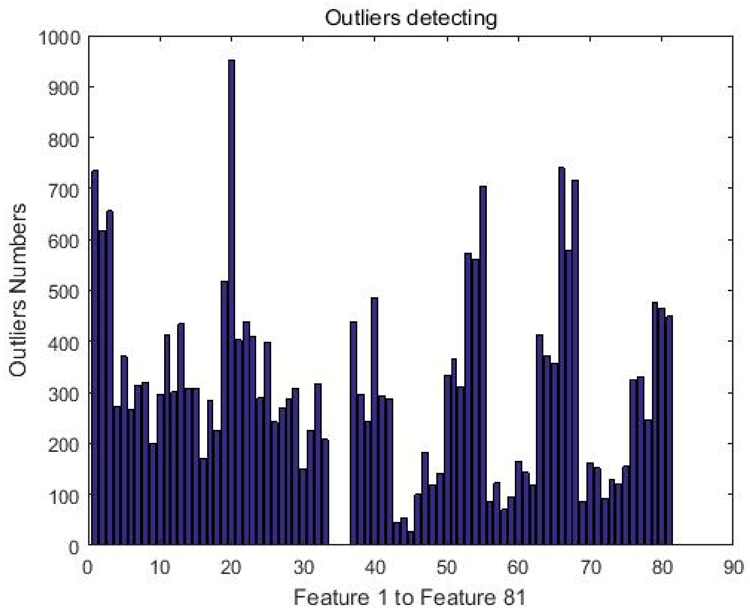

Figure 2: The outlier numbers for each feature @data set A

1.2 Fix the detected problems

Two ways to fix:

(1) For missing data, we used the mean value of each feature to replace the missing values.

(2) For the outliers, we used window values mean to replace the outliers.

1.3 Min-max normalization and Z-score normalization

(1) Min-max and z-corse Normalization

Fig.3 showed the values distributions of feature #9 before and after normalization. Fig.4 showed the values distributions of feature #24 before and after normalization. From fig.3 and fig.4 we could see that the original data which was not normalized distributed between large ranges. After applying min-max normalization, the data transferred to [0, 1] range. And after doing the z-score normalization, the data values distributed in a small scope. However, data had a similar distribution trend before and after normalization.

The difference between min-max normalization and z-score normalization was that min-max normalization could transfer data into values from 0 to 1. However, after applying min-max normalization, we could no longer input a feature value that fell outside of the original data range. We could use z-score method in this situation. Z-score method would transfer values into a small scope around zero. Normalization might or might not be desirable in some cases. However in some cases it might make samples that were dispersed in space closer to each other and hence were difficult to separate.

Compare fig.3 and fig.4, we could realize that data for feature #24 was more stable than data for feature #9. Data for feature #24 was more concentrated on the center of the range. And the third image in fig.4 showed most of the values had a lower deviation, so its data was more stable.

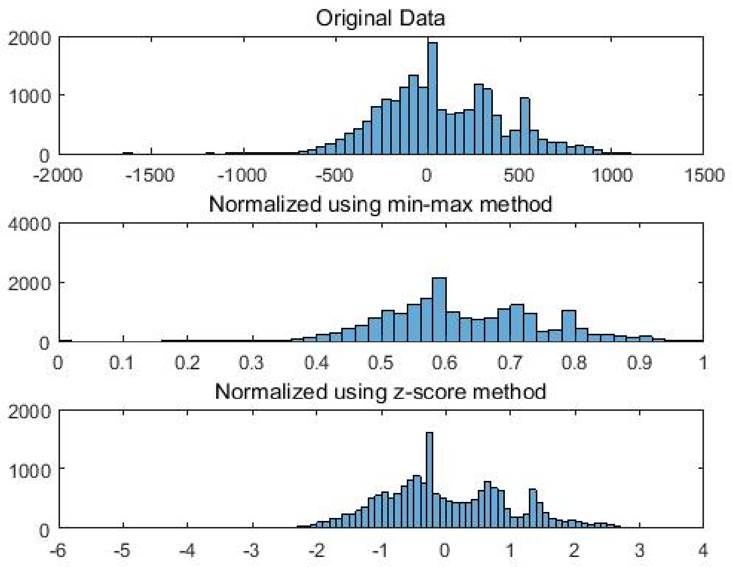

Figure 3: Values distribution before and after normalization @feature #9, data set A

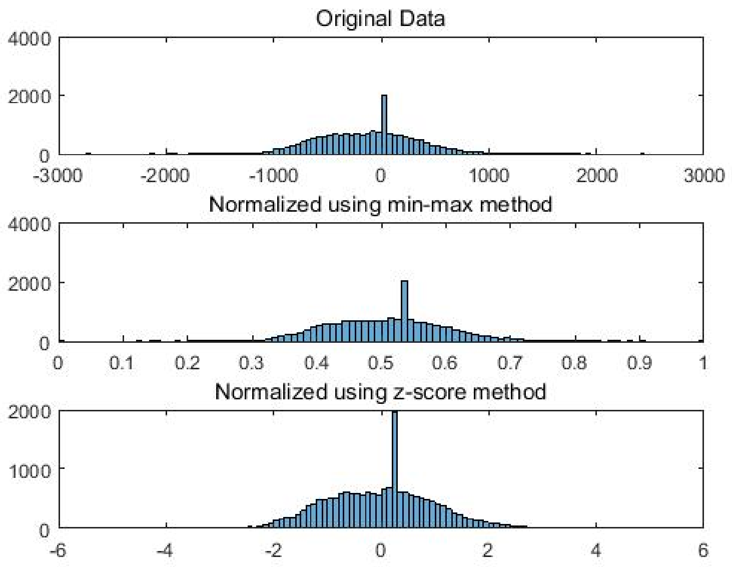

Figure 4: Values distribution before and after normalization @feature#24, data set A

(2) Auto-correlation

In fig.5 and fig.6, we plot the auto-correlation line for #9 and #24 features before and after normalization. From the two figs we can see that before and after normalization they have the same auto-correlation lines. So we can conclude that min-max normalization and z-score normalization dont change the auto-correlation.

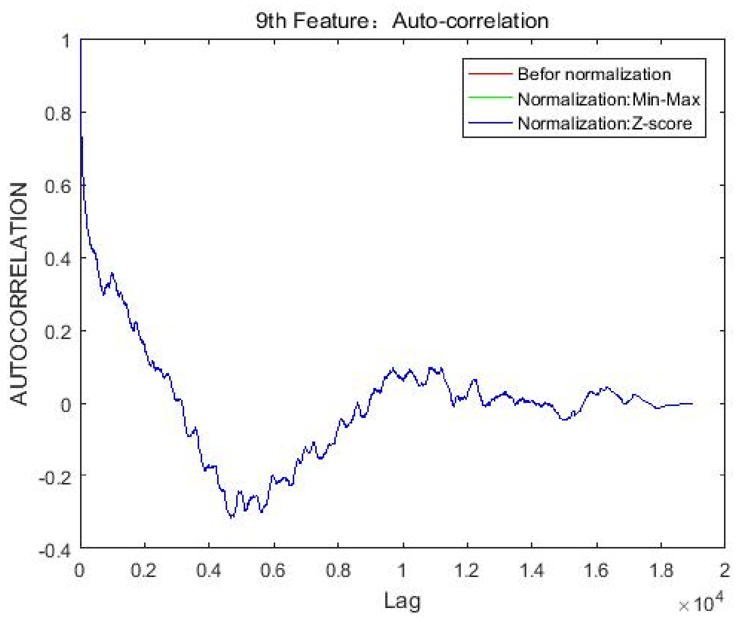

Figure 5: Auto-correlation @feature #9, data set A

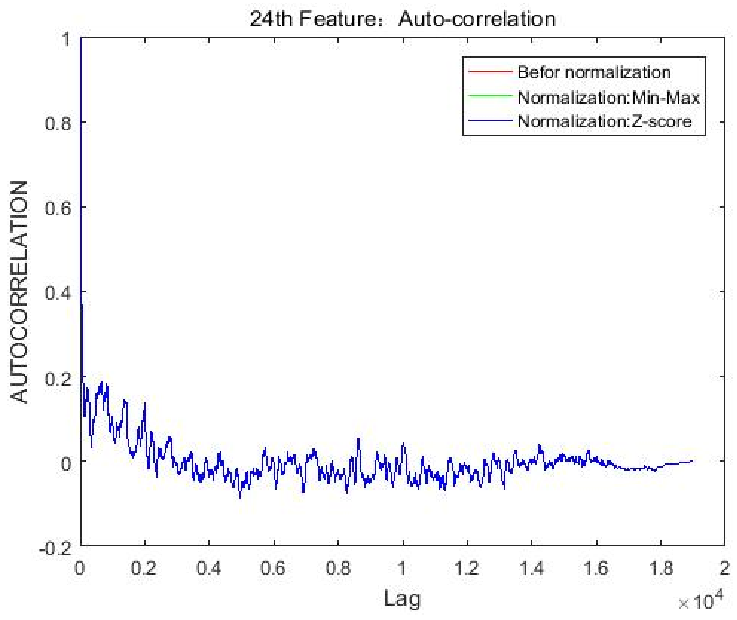

Figure 6: Auto-correlation @feature #24, data set A

In fig.5 we could see that the auto-correlation dropped from 0.6 to -0.3 and then rise to 0. So the time series data from feature #9 had a strong correlation. The data collected earlier could strongly affect the data collected later. In fig.6 we could see the auto-correlation was around zero. So the correlation would be much weak.

2 Part II

2.1 Compute the eigenvectors and eigenvalues

We used the MATLAB API: pca() function to calculate eigenvectors and eigenvalues. And then stored eigenvectors and eigenvalues into our eigen.mat in local folder.

2.2 The first and second principal components

We used pca() function to get the first two components and then used them as point values to plot data points. Fig.7 showed the results we got. There were five classes in this fig. For class 1, the points were closer to each other. For class 0(the red points) were more dispersed with each other. They had different distances and locations, so it would be easier to classify them.

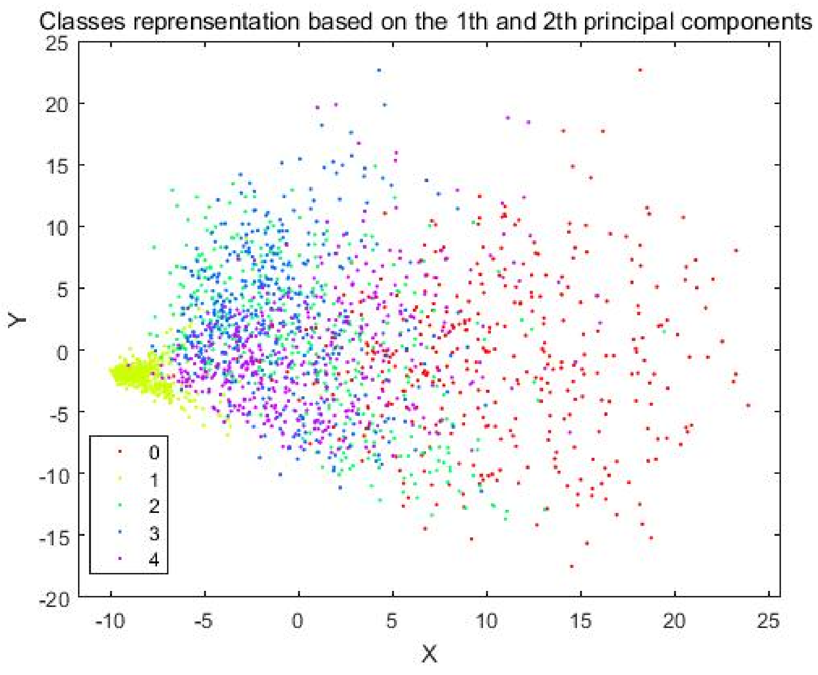

Figure 7: Data points @ the first two PCA components, data set B

2.3 The 5th and 6th principal components

Fig.8 showed the data points for the 5th and 6th principal components. Compared with fig.7, the data points for class 1 were blended in other classes data points. The classes were too close. It would be difficult to do the classification obviously.

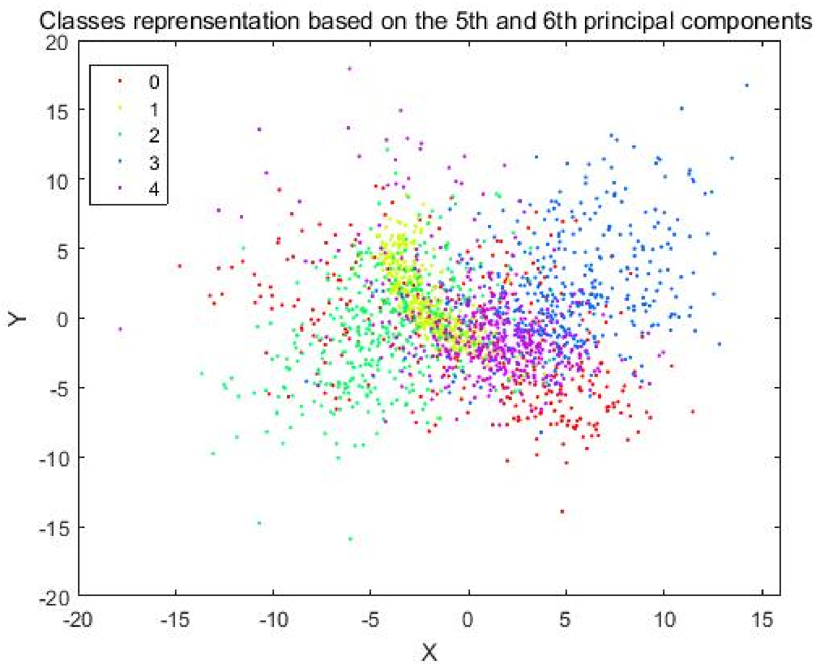

Figure 8: Data points @the 5th 6th PCA, data set B

2.4 The classification error and retained variance

Fig.9 showed the classification errors and retained variance of 8 sets (using the first 2, 4, 10, 30, 60, 200, 500, and all 784 PCA components as 8 sets of dimensionality reduced data to classify). From fig.9 we could see that the error was lower when the sum of the eigenvalues the data set became larger. We usually wanted to ensure that 95% of the variance was retained in the new space so that the classification error would be low enough.

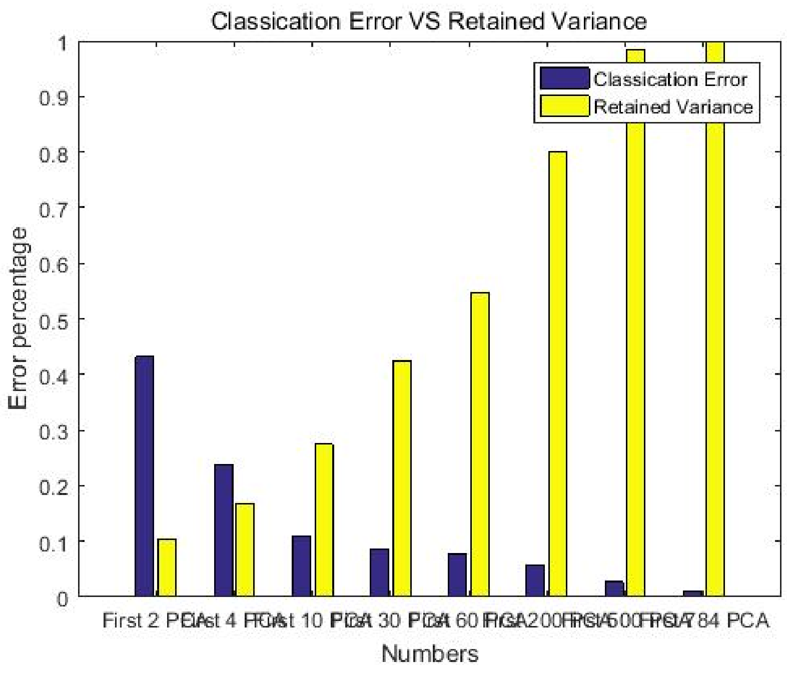

Figure 9: Classification error and retained variance

2.5 The first 2 LDA components

Next, we used LDA method to reduce dataset to two dimensions. So that we could plot data points as below. Comparing fig.10 to fig.7 we could see that data points from LDA could be easily distinguished as different classes. So for the same data set, LDA method had a much higher accuracy than the PCA method.

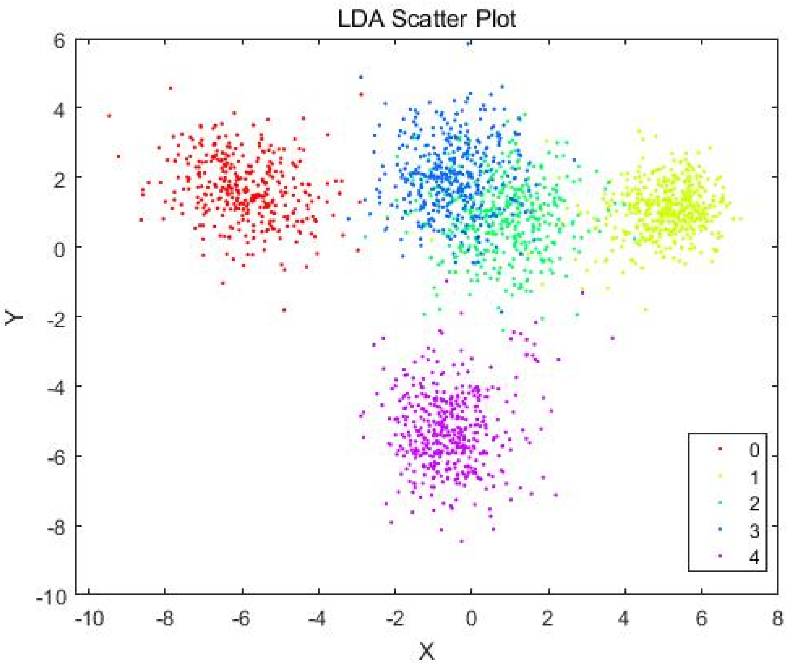

Figure 10: Data points reduced using LDA method @data set B

3 Part III

3.1 Plot LLE of image digit 3

After applying LLE function to all the images of digit 3, we generated a 4-dimension dataset. Using first and second components of the datasets as x and y-axis, we randomly got one of these two plots:

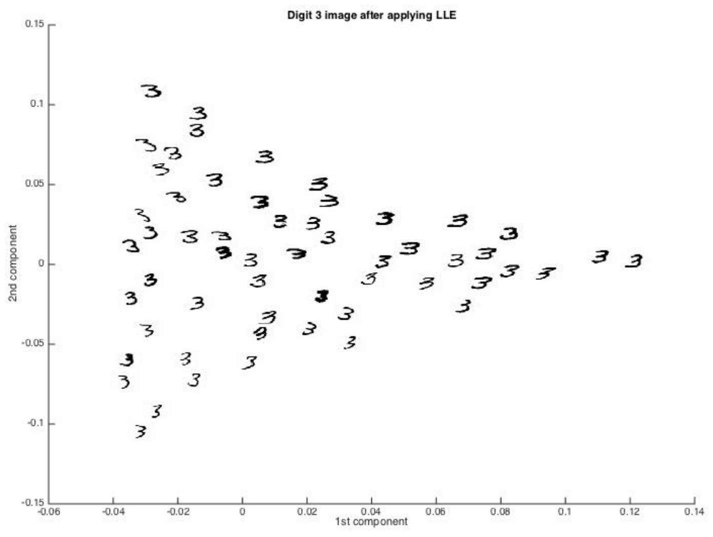

Figure 11: Plot a

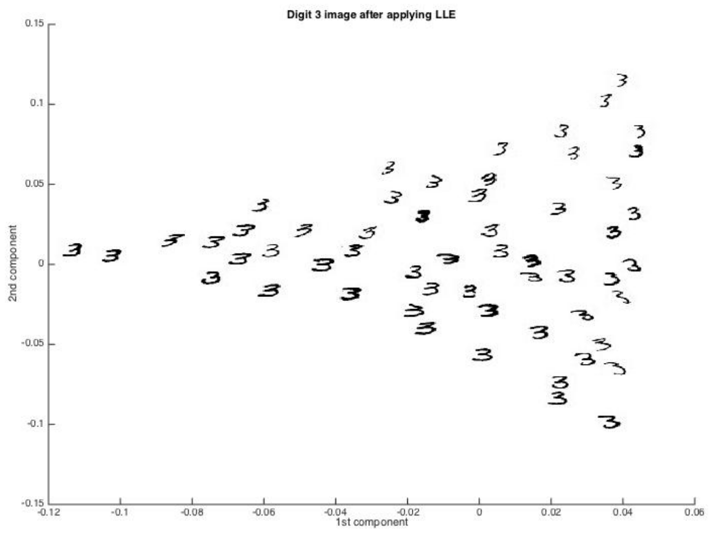

Figure 12: Plot b

Both of them were triangle shaped and symmetric. For Plot a, when x gets larger, the absolute value of y intended to be smaller towards zero; for Plot b, when x gets larger, the absolute value of y intended to be larger. The pattern spread out evenly. Since the LLE method began by finding 5 nearest neighbors of each point and then computed weights for them. With these weights, we could describe points as a linear combination of its neighbors before finding the low-dimensional embeddings of them. The weights were fixed, every point in LLE plot is locally based, that could explain why we could see a pattern in the final plot.

3.2 Plot ISOMAP of image digit 3

Using first and second components of ISOMAP dataset as x and y-axis, we generated Plot c look like this:

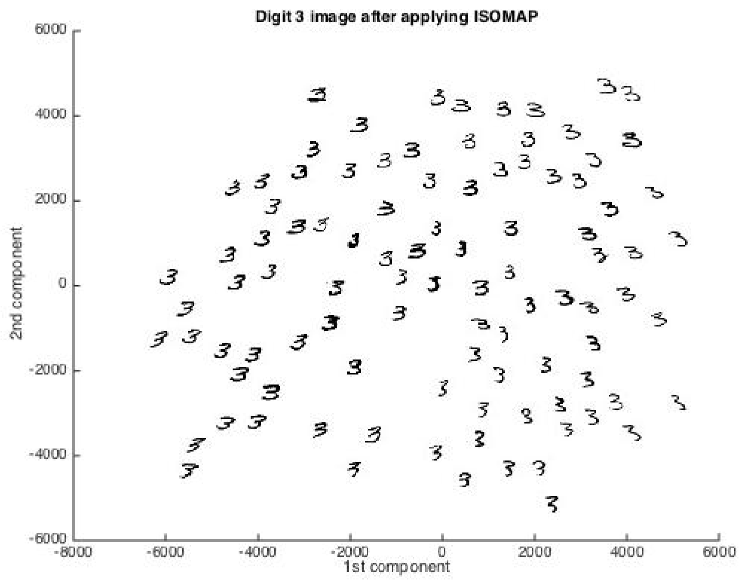

Figure 13: Plot c

Its evenly spread out between -6000 and 6000 and theres no specific pattern in the plot. In ISOMAP methods, we had to find 5 nearest points and then construct a neighborhood graph using Euclidean distances between points. Next, we needed to compute shortest paths between every two nodes. And finally, we applied Multi-dimensional Scaling method to compute the lower-dimensional embedding. The advantage is that ISOMAP could capture spaces geometric properties along manifold. However, the calculation could be slow. The patterns are globally based, while LLE is locally based.

3.3 Use Naive Bayes classifier base on LLE and ISOMAP

We randomly selected 70% of the dataset as a training set and the rest 30% as a testing set. We iterated 7 times because $3^7 \times 2066 < 1$, so that all the points would have a chance to be used in the training set. The average accuracy of LLE is 0.8635 and the average accuracy of ISOMAP is 0.8478. While LLE and ISOMAP made nonlinear transformation on the datasets, PCA and LDA are linear transformations. When we applied PCA and LDA on the data and used first 4 components to train the Naive Bayes classifier, the accuracy for PCA is 0.7608 and the accuracy for LDA is 0.9933.

Method	Accuracy
LLE	0.8635
ISOMAP	0.8478
PCA	0.7608
LDA	0.9933

Table 1: Accuracy Comparison

PCA was obtained by finding high variances for more discriminating features with more information by computing eigenvalues and eigenvectors, and then we set them as principle components. Because we only used first 4 components, there were limited features used and many other components eliminated, so the accuracy is not as high.

LDA was a supervised approach, it focused on finding projections that best identify differences in low dimensions while minimizing distances between points within the same class. Its final goal was to separate points into different classes, so LDA has the highest accuracy. In short, PCA is better to use according to variances and errors while LDA does better in discrimination based on class labels.

4 Part IV

In this part, we used different feature selection method to select features from the original dataset(DataC.mat) and measure the runtime and average accuracy of different cases. First, we use min-max normalization to preprocess the original data.

$$x_i' = \frac{x_i - minX}{maxX - minX} \tag{1}$$

This makes the values invariant to the rigid displacement of coordinates.

4.1 Filter feature selection

This is a filter feature selection question and asked us to use squared Euclidean distance as a distance base objective function together with the SFS strategy to select 8 features from the total 21 ones. Squared Euclidean distance:

$$d^2(p,q) = (p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + ... + (p_n - q_n)^2 \tag{2}$$

The distance between classes:

$$class - dist(p,q) = \frac{1}{n_p n_q} \sum_{i=1}^{n_p} \sum_{j=1}^{n_q} d^2(p_i, q_j) \tag{3}$$

In the selection processes, as the Sequential Forward Selection strategy asked, we started with an empty set, calculated the distance between classes and selected the feature which maximized the distance, and then removed the selected feature, calculated the distance again to select the next best feature. Continue this loop 8 times to select the feature subset. In the end, save the selected subset and reported the runtime of the algorithm.

Case1 runtime = 2.0568

4.2 Wrapper-based feature selection

This is a wrapper-based feature selection which asked us to use the Naive Bayes classifier as the objective function together with an SFS strategy to select the required feature subset.

Same as the process in question 1, every time we selected the feature with the highest accuracy. And within each loop, we use the same strategy as PART III.3, train the classifier by randomly selected 70% of data, and test with the remained 30%.

Within each loop, we iterated 7 times to calculate an average accuracy. The reason we choose to iterate 7 times is because$3^7 \times 2100 < 1$, and all samples can be used.

This 7 times loop also iterated 21 times to have all the average accuracy for 21 features. In the end, save the selected subset and reported the runtime of the algorithm.

Case2 runtime = 2.1960

4.3 Sequential Backward Selection (SBS) strategy

In this question, we implemented the SBS strategy with the same objective function we used in question 2.

Different from SFS we started with a full set of features, and each time delete the one with the worst accuracy.

In order to get 8 features, the loop took 13 times.

In the end, save the selected subset and reported the runtime of the algorithm.

Case3 runtime = 3.4994

Since in the SBS strategy we used in this question, the algorithm actually did more iteration than the former one in question2. The running time in this algorithm is bigger than the one in question2.

4.4 The average accuracy and run time

We used the Naive Bayes classifier to classify the 3 different feature subset we got from the questions before, and the whole 21 features.
The classification methods are basically the same as we used in PART III and the former 3 questions.

Case#	Accuracy	Runtime
1	0.7611	0.1659
2	0.7611	0.0191
3	0.3918	0.0200
4	0.7757	0.0207

Table 2: Accuracy and Runtime for different cases

As can be seen from the figure before, the running time of case1 is much bigger than those in the other cases.

Case3 has the lowest accuracy while the others are in the basically same level. We thought this is in case3 the dataset we chose was by SBS which took more iterations and the result during each loop was more changeable.

References

[1] drtoolbox. Matlab Toolbox for Dimensionality Reduction. Available from World Wide Web: (http://lvdmaaten.github.io/drtoolbox/).

[2] Hampel identifier function. Copyright 2015 The MathWorks, Inc.

[3] ECON toolbox. Copyright The MathWorks, Inc.

[4] SIGNAL toolbox. Copyright The MathWorks, Inc.

[5] STATS toolbox. Copyright The MathWorks, Inc.