Istanbul Technical University, Computer Engineering

BLG527E Machine Learning Homework 2

Anıl Öztürk

504181504

November 14, 2018

1 Multivariate Analysis

We are needed to create a discriminant function for the classes (i=0..9) in optdigits dataset. For that we need to do these;

- Read the training data, split the features and the labels
- Eliminate the unnecessary features
- Create a discriminate function for each class
  - Do parametric classification
  - Parameter estimation with common covariance matrix
- Run the discriminant estimator, test and score it

In order to read the data, we used **csv** and **NumPy** packages. We don't want unnecessary features, so we deleted the features which has 'zero' variance (no change). After that, we can start creating our discriminant function.

First, we calculate the prior possibilities for each class;

$$P(C_i) = \frac{\sum_{i=1}^{n} r_i^t}{N} \tag{1.1}$$

Then, we create mean vectors for each class;

$$m_i = \frac{\sum_{i=1}^{n} r_i^t x^t}{\sum_{i=1}^{n} r_i^t} \tag{1.2}$$

Then, we create covariance matrices for each class;

$$S_i = \frac{\sum_{i=1}^{n} r_i^t (x^t - m_i)(x^t - m_i)^T}{\sum_{i=1}^{n} r_i^t} \tag{1.3}$$

Then, we derive a common (shared) covariance matrix from classes' covariance matrices;

$$S = \sum_{i=1}^{n} P(C_i) S_i \tag{1.4}$$

We had a discriminant function like this;

$$g_i(x) = -\frac{1}{2} log|S_i| - \frac{1}{2}(x - m_i)^T S_i^{-1}(x - m_i) + logP(C_i) \tag{1.5}$$

After setting the common covariance matrix up, now we have it as;

$$g_i(x) = -\frac{1}{2}(x - m_i)^T S^{-1}(x - m_i) + logP(C_i) \tag{1.6}$$

$TrainingError = 0.037405179178655504$

	Predicted Class										
True Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
0	374	0	0	0	1	0	1	0	0	0	376
1	0	364	6	0	0	0	0	1	13	5	389
2	0	1	364	3	0	0	1	1	8	2	380
3	0	1	1	376	0	2	0	0	2	7	389
4	0	4	0	0	373	0	5	0	4	1	387
5	1	0	0	1	0	356	0	0	0	18	376
6	0	1	1	0	1	0	374	0	0	0	377
7	0	1	0	4	0	0	0	379	0	3	387
8	1	13	0	0	2	2	1	0	359	2	380
9	1	4	0	5	3	2	0	1	5	361	382
Total	377	389	372	389	380	362	382	382	391	409	3823

$TestError = 0.06121313299944352$

	Predicted Class										
True Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
0	174	0	0	0	0	2	0	0	1	1	178
1	0	166	5	0	0	0	0	0	5	6	182
2	0	0	168	7	0	0	0	0	1	1	177
3	0	0	1	171	0	3	0	0	5	3	183
4	0	2	0	0	175	0	0	1	2	1	181
5	0	0	0	0	0	179	0	0	0	3	182
6	0	2	0	0	1	0	178	0	0	0	181
7	0	0	0	0	1	3	0	163	2	10	179
8	0	12	0	0	0	5	0	1	142	14	174
9	0	3	0	1	1	1	0	0	3	171	180
Total	174	185	174	179	178	193	178	165	161	210	1797

2 Dimensional Reduction

This time we need to apply PCA and LDA to the training and test set, separately.

2.1 Applying PCA

To apply PCA and illustrate it on 2D space, we must execute some operations as follows;

- Derive the covariance matrix for the given input
- Find this matrice's eigenvector and eigenvalues
- Select the largest two eigenvalues' eigenvectors
- Project the features on these eigenvectors with their labels

After getting these two eigenvectors, if we dot product them with the feature matrix; we get features projected on these vectors, without losing their IDs on the feature space. This way we can color and separate them to the classes, properly.

$$[Features]_{3823x62}.[Eigens]_{62x2} = [Projected]_{3823x2} \tag{2.1}$$

This way, we have 3823 instance with 2 parameters (X and Y locations on the graph). You can see the projected outputs for training and test datasets below;

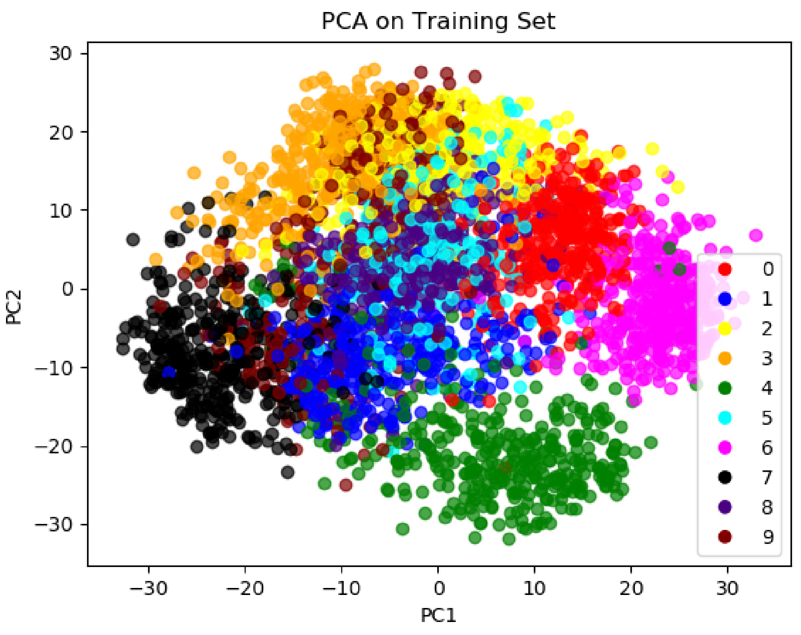


Figure 2.1: PCA on Training Set

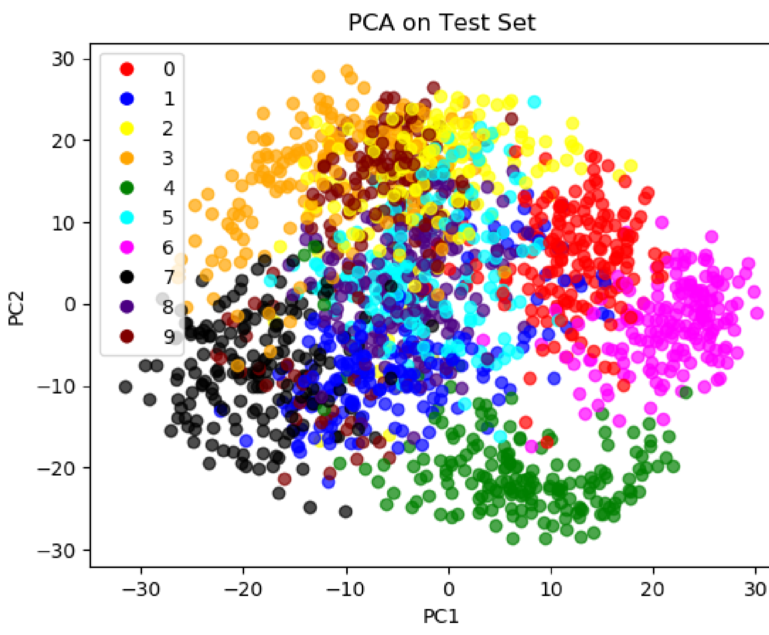


Figure 2.2: PCA on Test Set

Comparing each test-set instance's label with the nearest training instance neighbour's label can give us the error measurement. If we compare the test-set with test-set but not training-set, in circumstances like having few test subject, even if the test instances have the right label it can be counted as a failure because of the distance between them. So, the test error is as below;

$$Error = 0.4713411240957151$$

2.2 Applying LDA

To apply LDA and illustrate it on 2D space, we must execute some operations as follows;

- Find S_B and S_W matrices
- Find eigenvector and eigenvalues
- Select the largest two eigenvalues' eigenvectors
- Project the features on these eigenvectors with their labels

For this operation we need to matrices called S_B, S_W and general mean vector m. Their definitions are below;

$$S_W = \sum_{i=1}^{K} S_i \tag{2.2}$$

$$m = \frac{1}{K}\sum_{i=1}^{K} m_i \tag{2.3}$$

$$S_B = \sum_{i=1}^{K} N_i(m_i - m)(m_i - m)^T \tag{2.4}$$

After calculating these matrices, we must find the eigenvalues and eigenvectors for the term below;

$$S_W^{-1} S_B \tag{2.5}$$

Actually by doing that, we selected the axis combination that has maximum distance between class means and minimum in-class variance as a projection space. Since we will project our features on 2D space, we must pick the largest 2 eigenvalues' corresponding eigenvectors as projection axes. After dot producting the eigenvectors and the inputs, following graphs will be created.

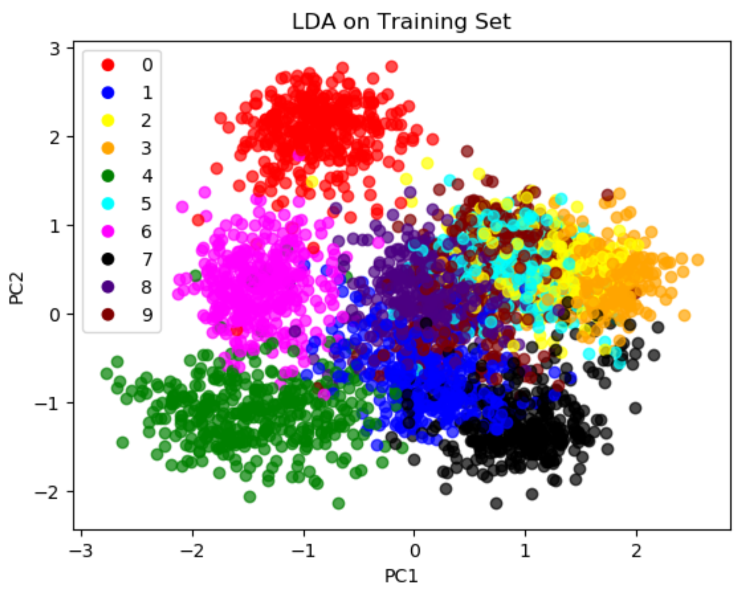


Figure 2.3: LDA on Training Set

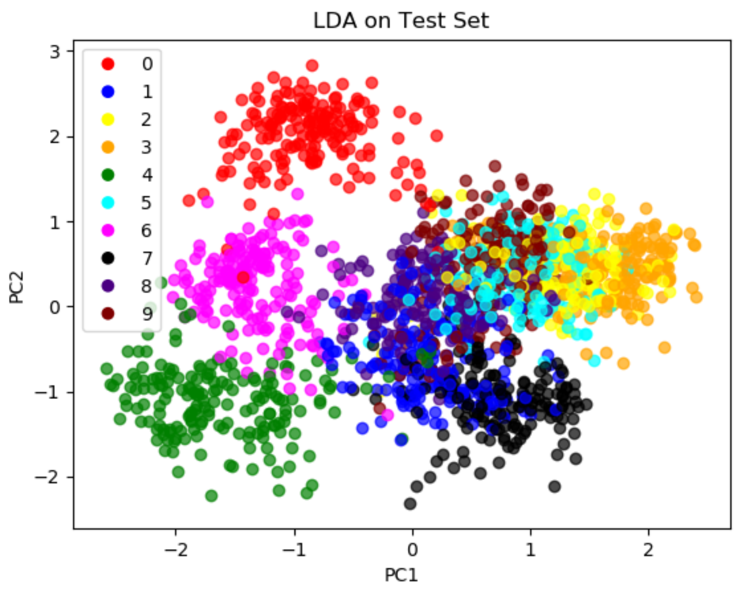


Figure 2.4: LDA on Test Set

$$Error = 0.40734557595993 32$$

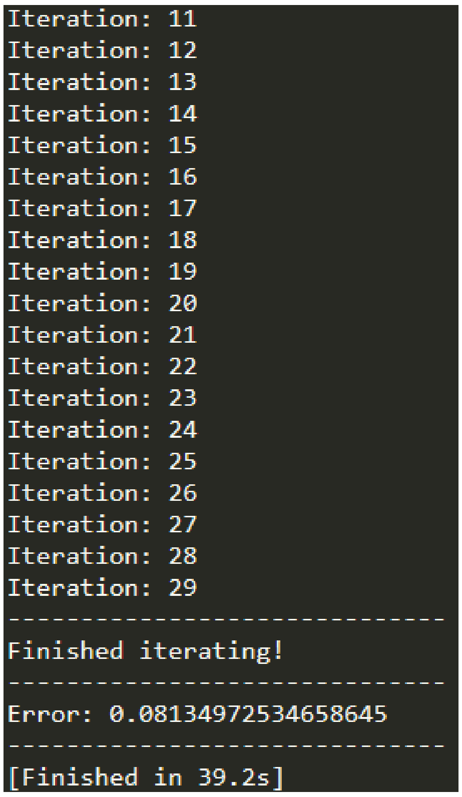
3 Clustering

To do k-means clustering we need to do following:

- Create k random cluster centers
- Find nearest cluster center for each instance
- Set each instance's cluster ID as nearest cluster's ID
- Find new center of the clusters based on their member instances' average
- Iterate until no change at cluster centers

Since the operation must be started manually, at the each run the program outputs different cluster combinations and error values depends on them.

In the picture, you can see one of the best results from the program.
Generally the error value with L1 measure scale varies between (0.11 – 0.15).



```
Iteration: 11
Iteration: 12
Iteration: 13
Iteration: 14
Iteration: 15
Iteration: 16
Iteration: 17
Iteration: 18
Iteration: 19
Iteration: 20
Iteration: 21
Iteration: 22
Iteration: 23
Iteration: 24
Iteration: 25
Iteration: 26
Iteration: 27
Iteration: 28
Iteration: 29
------------------------------
Finished iterating!
------------------------------
Error: 0.08134972534658645
------------------------------
[Finished in 39.2s]
```

Figure 3.1: k=20 Cluster runtime example

4 RUNNING THE CODE

The assignment has been written in **Python 3.6.2** with **Sublime Text 3.1.1** IDE. In order to run the assignment code, the following Python libraries are required;

- math
- numpy
- matplotlib
- scipy
- csv
- random

4.1 TO RUN Q1

Simply run **question1.py**. You will see accuracy, error and confusion matrixes as a result.

4.2 TO RUN Q2

Simply run **question2.py**. You can change the first two line to change PCA to LDA and training data input to test data input, you will get error values depending on this choices. It can take a while when plotting the eigen-projection and measuring the error.

4.3 TO RUN Q3

Simply run **question3.py**. You must wait the k-means iterations to finish if you want to see the result.