Homework Assignment #6

21600004

Kang Seok-Un

Introduction

This homework uses the famous public dataset, the MNIST dataset. At this time, entire data are used. The algorithm used in this homework is principal component analysis(PCA) and linear discriminant analysis(LDA). When visualize the dataset, dimension will be two.

Experiments

1. PCA

First, calculate the mean of the entire train dataset. Then, calculate the Covariant matrix 'cov'. After that, by using eigendecomposition, find the eigenvalues and eigenvectors. At this time, the important work is that by sorting eigenvalues in descending order, sort eigenvectors corresponding to eigenvalues in descending order.

Multiply the front two vectors of the sorted eigenvector to project the train data into the two-dimensional eigenspace by the train data (Fig 1.). After all these processes, plotting the 'scatter()' is the same as Figure 2.

```
69      train_x, train_y = train_set
70
71      # PCA
72      mean_Train = train_x.mean(0)
73
74      cov = np.cov(train_x.T)
75      eig_val, eigVector = np.linalg.eig(cov)
76
77      sorted_eigVector = eigVector[np.argsort(eig_val)[::-1]]
78
79      pca_dim2 = np.matmul(train_x, sorted_eigVector[:, :2]).real
80
81      visualization_scatter(pca_dim2, train_y, number_of_label=10, save_file_name="PCA")
```

Figure 1. The implementation of PCA

2. LDA

To use LDA, should follow the processes below.

2.1. Calculate Mean of Global and each classes

For Calculate S_b and S_W, should calculate the mean of entire dataset and each classes as figure 2.

```
92      # make classified data list
93      for label in range(10):
94          class_data.append(train_x[train_y==label])
95          class_mean_data.append(np.mean(class_data[label], axis=0).reshape(784, 1))
96
97      sb = np.zeros((784, 784))
```

Figure 2. Calculating the mean of each classes

2.2. Calculate S_b

At section 2.1., we know the mean of entire dataset and each classes. First, calculate the differences between

mean of each classes and global mean (figure 3. Line 102). And then multiply number of dataset in each class, differences and that's transposed matrix. If add all the results of the previous multiplication, will get S_b.

```
100     # Calculate S of b
101     for label in range(10):
102         diff = class_mean_data[label] - global_mean
103         sb += class_data[label].shape[0] * np.matmul(diff, diff.T)
```

Figure 3. Calculating S_b

2.3. Calculate S_w

This section has a similar process to section 2.2. However, this section finds the mean of the class to which the data belongs and the difference between that data.

```
105     # Calculate S of w
106     sw = np.zeros((784, 784))
107     for label in range(10):
108         si = np.zeros((784, 784))
109         for data_x in class_data[label]:
110             diff = data_x.reshape(784, 1) - class_mean_data[label]
111             si += np.matmul(diff, diff.T)
112         sw += si
```

Figure 4. Calculate S_w

2.4. Projection onto 2-dim eigenspace

If multiply the inverse matrix of S_w by S_b and then perform eigenvalue composition, the eigenvalue and eigenvector want can be obtained. Next, the sorted eigenvector can be projected into 2-dim eigenspace by multiplying the dataset as did in PCA.

```
119     invSw = np.linalg.pinv(sw)
120     invSw_mul_Sb = np.matmul(invSw, sb)
121     eig_val, eig_vec = np.linalg.eig(invSw_mul_Sb)
122     sorted_eig_vec = eig_vec[:, np.argsort(eig_val)[::-1]]
123
124     lda_dim2 = np.matmul(train_x, sorted_eig_vec[:, :2]).real
125     visualization_scatter(lda_dim2, train_y, number_of_label=10, save_file_name="LDA")
```

Figure 5. Projection into 2-dim eigenspace

Results

As shown in the Experience section, the following results can be obtained by implementing and operating the code.

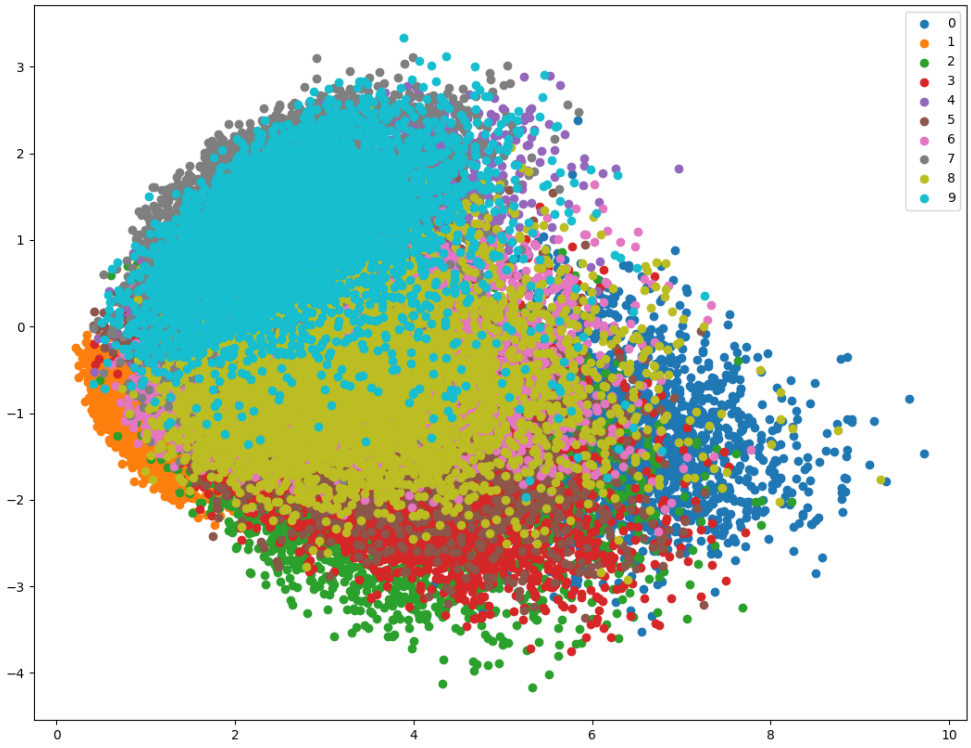

Figure 6. The result of PCA into 2-dim eigenspace

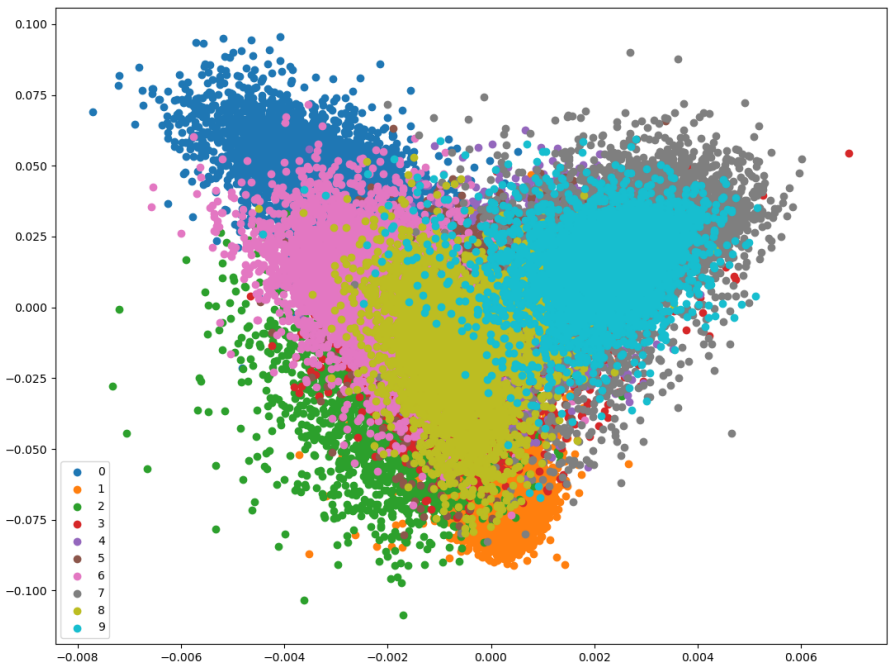

Figure 7. The result of LDA into 2-dim eigenspace

Conclusion

The most difficult thing in carrying out this task was to check and use the dimension of the data. In the case of coding to solve problems by using a specific algorithm, At most of them used a two-dimensional array. However, in this task, which uses a NumPy array and results vary depending on each axis, there were many difficulties in understanding the information on each axis.

LDA says it is easier in aspects of classification than PCA due to using labels. When comparing PCA and LDA, it can be seen that in class 0, LDA is a little denser. However, when comparing Figures 6 and 7, it is still questionable whether an answer can be obtained to the question, "Is it really easy to do classification?". In addition, the results were worse than those using deep learning in the textbook pdf, so I look forward to implementing more advanced technology in the next task.