HW1

Machine Learning

21600004
Kang Seok-Un

This homework #1 is requiring us to calculate mean, variance, eigenvectors, and eigenvalues about MNIST dataset. And it is also requiring us to print the mean and the first 10 eigenvector images and plot the first 100 eigenvalues of MNIST dataset.

The MNIST Dataset has 50,000 training datasets, 10,000 validation datasets and 10,000 test datasets. Each training data exists as a 784-dimensional vector, and each vector has a label that is hand-written digit of 0 to 9.

I used this training dataset for this homework.

This homework requires a total of three tasks, each of which is as follows and will be explained in order.

1. calculate mean, variance, eigenvectors, and eigenvalues.
2. Print the mean and the first 10 eigenvector images.
3. Plot the first 100 eigenvalues.

Calculate mean, variance, eigenvectors, and eigenvalues

Need to calculate the mean and variation of the training data. And need to calculate the eigenvector and eigen value using eigenvalue decomposition.

For this, I used functions: ‘numpy.mean()’, ‘numpy.var()’. At this time, since each data consists of 784 features, set the parameter of function as axis=0 to calculate mean and variation for each feature. This content can be seen by looking at line 58 to line 68 in Figure 1.

Next, eigenvector and eigenvalue should be calculated. To calculate Eigenvector and eigenvalue, must first obtain the covariance of the data. So, I used the function: 'numpy.cov()'. At this time, the parameter of the 'numpy.cov()' function must be a one-dimensional or two-dimensional array. In the case of a two-dimensional array, since the column of the array represents observations and each row represents variables, train dataset should be transposed (figure 2. Line 79).

After calculating covariance, finally can obtain eigenvalue and eigenvector by performing eigenvalue decomposition using function: ‘numpy.linalg.eig()’ (figure 2, line 80).

Print the mean and the first 10 eigenvector images.

To print the previously calculated mean and eigenvector, it must be transformed from a one-dimensional array to a two-dimensional array. To change the mean and eigenvector consisting of 784 values to [28x28] matrix, I used the function: 'reshape()' (figure 1, line 70-71; figure 2. line 88). In the case of Eigenvector, since the value is a floating type rather than unit8, it should be converted its type as unit8 from floating type(figure 2. Line 89).

As the results, image of Mean is at figure 3., and images of first 10 eigenvectors are at figure 4. to figure. 13.

At figure 4 . to figure. 13, start point of eigenvector is (0, 0) and end point is location of white dot. But I am not sure what eigen vector of certain data means.

Plot the first 100 eigenvalues.

For plotting first 100 eigenvalues, I used function ‘plot’ and set the range of 0 to 100 (figure 2. Line 95). As the results, the image of plotted first 100 eigenvalues are figure 14. Loot at the figure 14, only first 10 to 20 eigenvalue is over 1, others are almost close to 0.

Conclusion

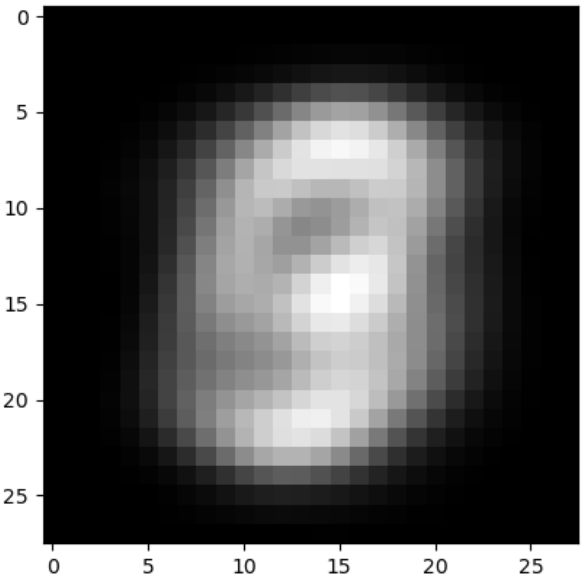
While performing this task, the eigenvalue deposition learned in linear algebra class was performed using a numpy library. Whenever theoretical knowledge is implemented through computer programming, the need for learning is felt. And I realized the need to study statistics while performing this task.

```
58    train_set, val_set, test_set = load_data('mnist.pkl.gz')
59
60    train_x, train_y = train_set
61    val_x, val_y = val_set
62    test_x, test_y = test_set
63
64    print(train_x.shape)
65    print(train_y.shape)
66
67    meanTrain = np.mean(train_x, axis=0)
68    varTrain = np.var(train_x, axis=0)
69
70    mean_img = meanTrain.reshape((28,28))
71    var_img = varTrain.reshape((28,28))
72
73    plt.figure("mean")
74    plt.imshow(mean_img, cmap='gray')
75
76    plt.figure("var")
77    plt.imshow(var_img, cmap='gray')
```

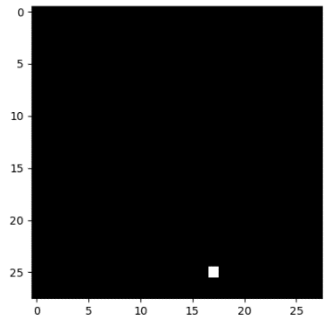
[Figure 1. Calculating Mean and Variance]

```
79      cov = np.cov(train_x.T)
80      eigValue, eigVector = np.linalg.eig(cov)
81
82
83      print("cov::", cov.shape)
84      print("eigeVaule::", eigValue.shape)
85      print("eigeVector::", eigVector.shape)
86
87      for i in range(10):
88          vec_img = eigVector[i].reshape((28,28))*255.9
89          vec_img = Image.fromarray(vec_img.astype(np.uint8))
90
91          plt.figure("eigen vector image"+str(i))
92          plt.imshow(vec_img, cmap='gray')
93
94      plt.figure("eigen value 1 to 100")
95      plt.plot(eigValue[:100])
96      plt.show()
```

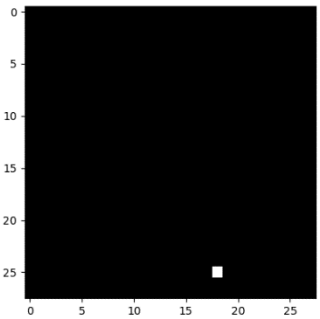
[Figure 2. Calculating Eigenvalue and Eigenvector]



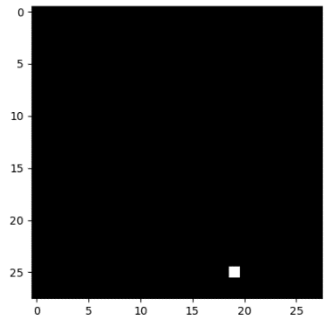
[figure 3. Mean of dataset]



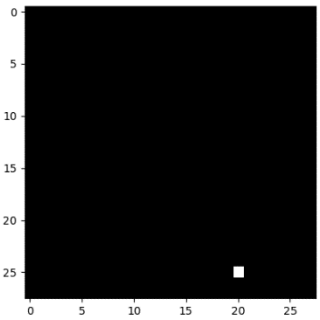
[Figure 3. 1st eigenvector]



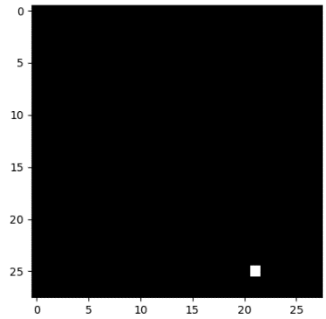
[Figure 4. 2nd eigenvector]



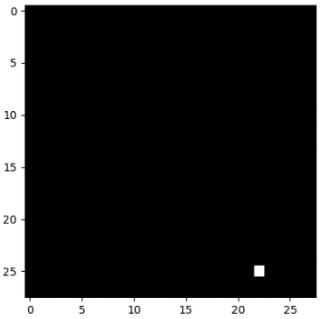
[Figure 6. 3rd eigenvector]



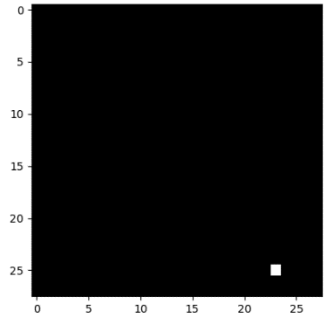
[Figure 7. 4th eigenvector]



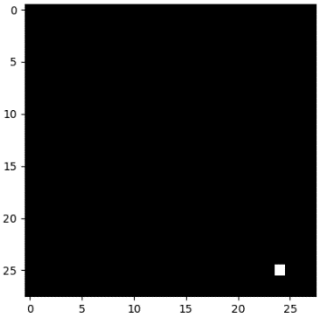
[Figure 8. 5th eigenvector]



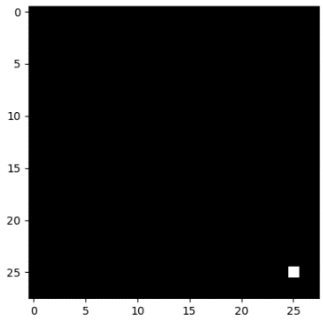
[Figure 9. 6th eigenvector]



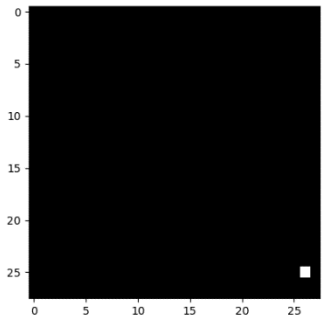
[Figure 10. 7th eigenvector]



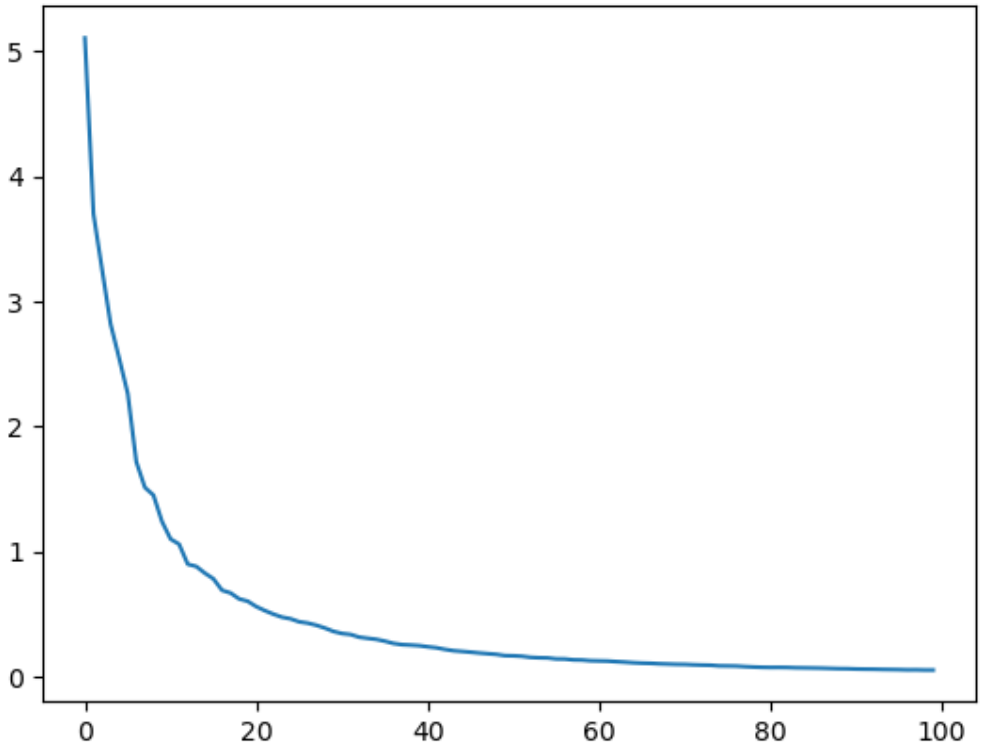
[Figure 11. 8th eigenvector]



[Figure 12. 9th eigenvector]



[Figure 13. 10th eigenvector]



[figure 14. First 100 eigenvalues]