COMP5318 - Machine Learning and Data Mining
Assignment 1

Yue Zhang(500414752)

1. Introduction

1.1 Aim

The purpose of this study is twofold. The first goal is to understand the whole process of machine learning. The first step is to preprocess the data, then to train the fitting model of the data, then to select the appropriate parameters, and finally to predict. The second goal is to learn the principles of classifiers. By writing code, we understand the algorithm principle of classifier.

1.2 The import of study

The importance of this learning is to strengthen our understanding of the whole process of machine learning, and be able to learn some models more autonomously to solve the problems encountered. The knowledge learned is not only algorithm, but also transformed into code. The visual implementation of code is also a very important ability.

2. Methods

2.1 Pre-processing

In generally, before executing the classification algorithm, inputting and handling the selected dataset is the first step.
We need to handle the original data which include training data(30000,784), training labels(30000,), testing data(2000,784) and testing labels(2000,) to improve the performance. Singular Value Decomposition (SVD) and Principal Components Analysis(PCA) are great methods to handle this data.

2.1.1 Singular Value Decomposition(SVD)

Singular value decomposition(SVD) is used to reduce the dimension of data. In this report, the parameter here is 10, because when I output this image, there is no visible noise, which is similar to the original data image.

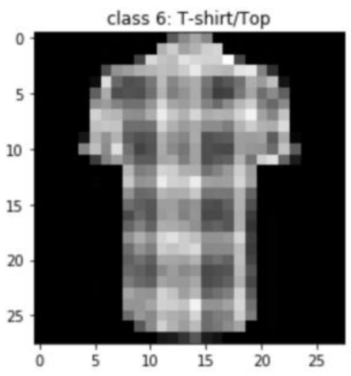


Figure 1: image

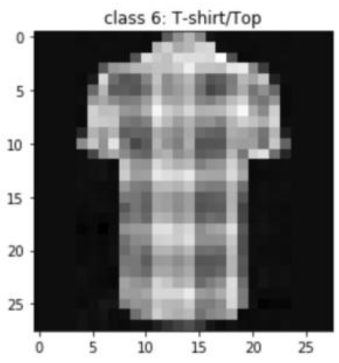
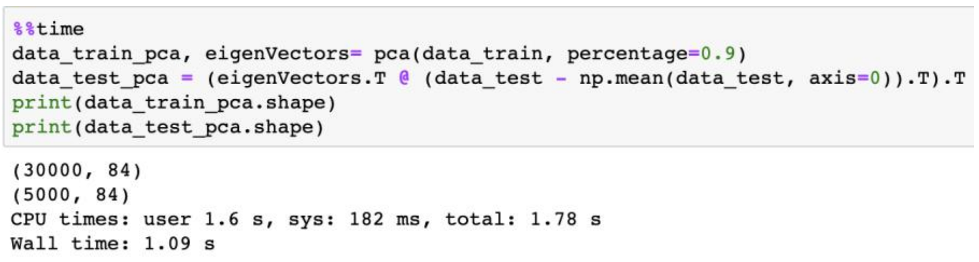


Figure 2: image after SVD

2.1.2 Principal Components Analysis(PCA)

Principal Components Analysis(PCA) is used to reduce the dimension of data to reduce running time and improve the performance. In this report, we use Principal Components Analysis (PCA) to transform the original sample data (784 dimension) into the new data (84 dimensions).



```
%%time
data_train_pca, eigenVectors= pca(data_train, percentage=0.9)
data_test_pca = (eigenVectors.T @ (data_test - np.mean(data_test, axis=0)).T).T
print(data_train_pca.shape)
print(data_test_pca.shape)
```

```
(30000, 84)
(5000, 84)
CPU times: user 1.6 s, sys: 182 ms, total: 1.78 s
Wall time: 1.09 s
```

Figure 3: data shape after PCA

2.2 Classification design

2.2.1 K-Nearest Neighbor

In order to realize KNN, we need to find k nearest neighbors to make the prediction, so we only need to calculate the distance between the prediction samples and all samples in the training set, and then calculate the minimum k distance. In this report, we find the best value of parameter k through training model.

2.2.2 Naive Bayesian Model

In order to implement the naive bayesian algorithm, we need to estimate the prior probability P(C) of the class based on the training set, and estimate the conditional probability P(xi | c) for each attribute. Here maximum likelihood estimation (MLE) is used to estimate the corresponding probability. And since some of the parameters that we calculated using the maximum likelihood estimate were zero, then when we calculated using Bayesian formulas, the numerator becomes zero because you multiply a value of zero by a value of zero, and the denominator becomes zero, so it becomes 0/0 unpredictable. In this report, we use Laplace smoothing to solve this problem. And there is no need to adjust the parameters

2.2.3 Softmax Regression

In order to realize the softmax logistic regression mode, we want to estimate the probability of occurrence of each classification result for X. Therefore, our hypothesis function will output a k-dimensional vector to represent the probability value of the K estimates. We train the model parameters to minimize the cost function. We use cross entropy as the loss function. Then, we take the derivative of the loss function. The values are mapped to 0-1, and then derivative is used to determine the weight according to the derivative and learning rate. Then the prediction value is determined according to the weight. We can adjust the iteration and learning rate to determine the weight, and then get different prediction values.

2.3 K-folds Cross Validation

In k-fold cross-validation, the training set data is divided into k parts. In this k part, one of the K parts is selected as the test data, and the remaining k-1 is used as the training data. Repeat K times to ensure that the data of K parts have been tested respectively. Finally, the K experimental results are averaged. This can better find the most suitable KNN value. In this report, we will use the 5-fold cross-validation to evaluate the model.

3.Experiments and Results

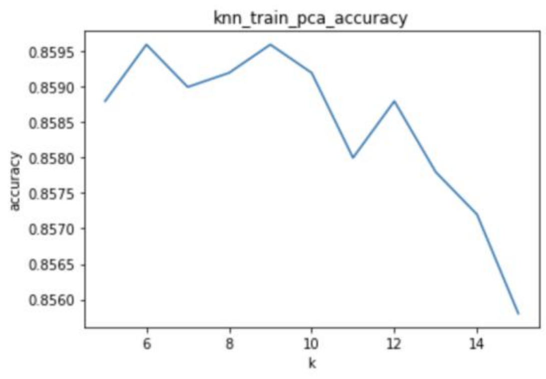
3.1 Experiments

All the experiment details of these three classifications would be described and analyzed in this section. This project would run in ***python 3.7*** on ***Macbook Pro*** with 2.3 GHz ***Intel Core i5*** CPU 8GB.

3.2.1 K-Nearest Neighbor

(1) If we don’t use the K-folds Cross Validation, we divided all train_data into 25000 and 5000 as training set and test set respectively,the best accuracy can be get on K=6. But I don't think it is fair.

```
[0.8588, 0.8596, 0.859, 0.8592, 0.8596, 0.8592, 0.858, 0.8588, 0.8578, 0.8572, 0.8558]
```

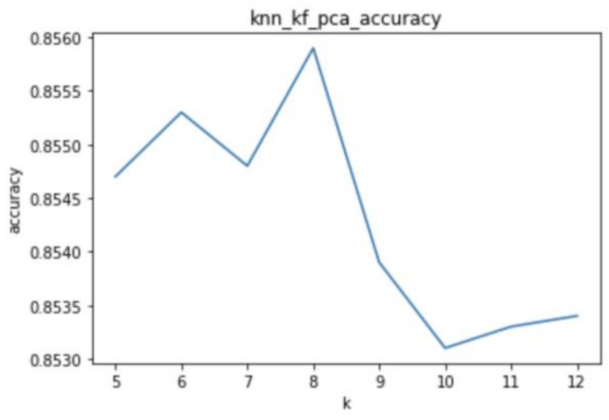


```
CPU times: user 14min 45s, sys: 1min 50s, total: 16min 35s
Wall time: 8min 31s
```

Figure 4: Accuracy~k after KNN

(2) So we use PCA and 5-fold Cross Validation(reduce dimension to 84). Obviously, when k = 8, the accuracy reaches the peak. So k = 8 may be the best training model.

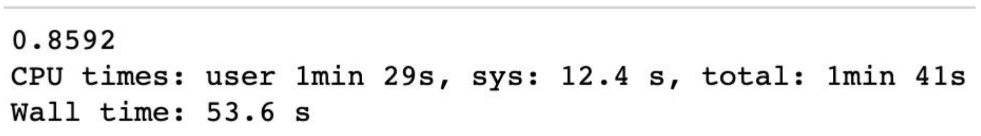
```
[0.8547, 0.8553, 0.8548, 0.8559, 0.8539, 0.8531, 0.8533, 0.8534]
```



```
CPU times: user 1h 6min 39s, sys: 7min 34s, total: 1h 14min 14s
Wall time: 39min 3s
```

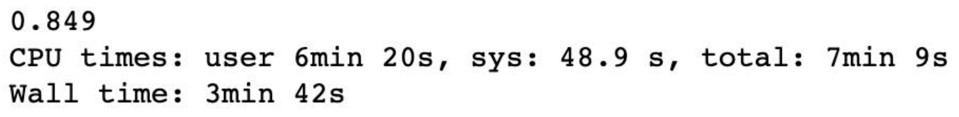
Figure 5: Accuracy~k after KNN + 5 folds Cross Validation

(3) We split the data into 25000 and 5000 training models. When k = 8, compare the accuracy and speed of PCA, SVD and PCA + SVD respectively.



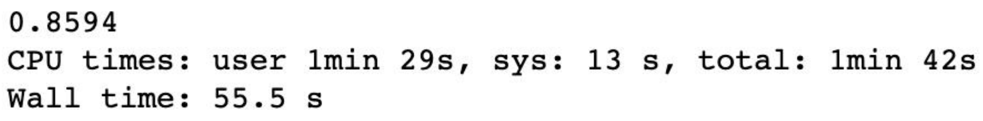
```
0.8592
CPU times: user 1min 29s, sys: 12.4 s, total: 1min 41s
Wall time: 53.6 s
```

Figure 6: Accuracy + Time after PCA



```
0.849
CPU times: user 6min 20s, sys: 48.9 s, total: 7min 9s
Wall time: 3min 42s
```

Figure 7: Accuracy + Time after SVD



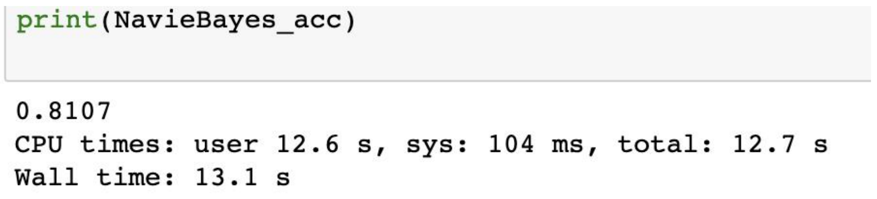
```
0.8594
CPU times: user 1min 29s, sys: 13 s, total: 1min 42s
Wall time: 55.5 s
```

Figure 8: Accuracy + Time after PCA

We found that PCA + SVD has the highest accuracy, and the speed is faster than SVD.

3.2.2 Naive Bayesian Model

According to the 5-fold Cross Validation, we can see the accuracy of the training model.



```
print(NavieBayes_acc)
```

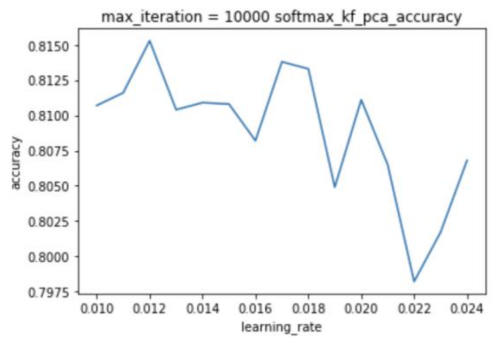
```
0.8107
CPU times: user 12.6 s, sys: 104 ms, total: 12.7 s
Wall time: 13.1 s
```

Figure 9: Accuracy + Time after NBM

3.2.3 Softmax Regression

Find the best model using PCA 5-fold Cross Validation(reduce dimension to 84), we can see when the max_iteration = 20000, the best learning_rate is 0.014

```
[0.8107, 0.8116, 0.8153, 0.8104, 0.8109, 0.8108, 0.8082, 0.8138, 0.8133, 0.8049, 0.8111, 0.8065, 0.7982, 0.8017, 0.80
68]
```

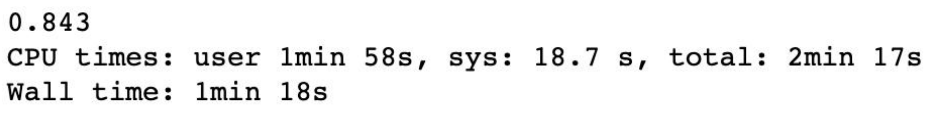


```
CPU times: user 3min 27s, sys: 2.58 s, total: 3min 30s
Wall time: 3min 48s
```

Figure 10: Accuracy~learning_rate after Softmax

3.2 Results

For three different classifiers in this report, we can see that using KNN model have the best accuracy, Naive Bayesian Model have the lowest accuracy. On the contrary, KNN have the slowest speed and Naive Bayes have the fastest speed.
So choose the model with the highest accuracy, which is KNN of k=8 . We use the Testing data (2000) for verification.

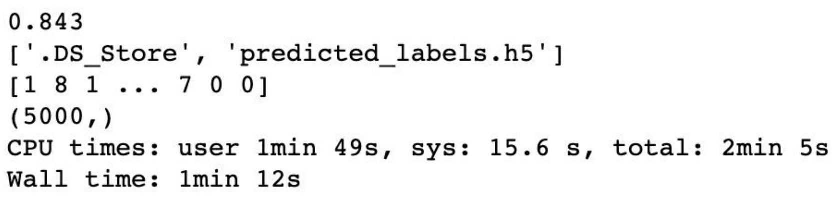


```
0.843
CPU times: user 1min 58s, sys: 18.7 s, total: 2min 17s
Wall time: 1min 18s
```

Figure 11: Accuracy after KNN(k=8)

4. Conclusion

Overall, we guess KNN is the greatest algorithm in these three algorithm to predict dataset. Although KNN has large calculated amount which would cause longer running time, it has the best accuracy. We can speed it up by reducing the dimension of the data through PCA.



```
0.843
['.DS_Store', 'predicted_labels.h5']
[1 8 1 ... 7 0 0]
(5000,)
CPU times: user 1min 49s, sys: 15.6 s, total: 2min 5s
Wall time: 1min 12s
```

Figure 12: Output predict labels

5. Appendix

STEP 1:   running   Dataset (Step: 1 Running this block to output predict_labels)
STEP 2:   running   Pre-processing   (Step: 2 Running this block to output predict_labels)
STEP 3:   running   Output (Step: 3 Running this block to output predict_labels)