COMP5318 - Machine Learning and Data Mining

Assignment 1

Team Members:
Ruochen Pi

Student Id
500055496

Group Number
Group 174

April 7, 2022

目录

1 Introduction

The objective of this task is to select and establish four classifiers to process a group of 28*28 images. There are ten categories of images, thirty thousand training sets and two thousand test sets. Group174 uses 3 classifiers, which is Nearest Neighbor, Logistic Regression, SVM, and one ensemble method, which is Random forest.

The purpose of this study is to find the best classifiers and parameters for this image classification task. In the process of using different classifiers, understand the influence of different parameters on classifiers.

2 Methods

The main content of this chapter is the method we used in the process of experiment, including pretreatment method, classifier selection, classifier parameter adjustment method, etc.

2.1 Data pre-processing

Group174 intends to use two methods of normalization and PCA in data preprocessing. After rigorous study and comparison of control variables, it was found that although the speed of classifier code was faster after dimension reduction with PCA, the accuracy was not significantly improved, so it was deleted.

2.1.1 Normalization

The specific function of normalization is to induce the statistical distribution of uniform samples. Normalization between 0 and 1 is a statistical probability distribution. After data normalization, the optimization process of the optimal solution will obviously become gentle, and it is easier to correctly converge to the optimal solution.

```
In [8]: data_train_feature[0][0]

Out[8]: array([  0,   0,   0,   0,   1,   0,   0, 131, 184, 199, 229, 234, 217,
                212, 204, 208, 226, 227, 203, 185, 173,  44,   0,   4,   0,   0,
                  0,   0], dtype=int64)

In [9]: train_norm[0][0]

Out[9]: array([0.        , 0.        , 0.        , 0.        , 0.00505051,
               0.        , 0.        , 0.59817352, 0.75720165, 0.78039216,
               0.89803922, 0.91764706, 0.85098039, 0.83137255, 0.8       ,
               0.81568627, 0.88627451, 0.89019608, 0.79607843, 0.7254902 ,
               0.68924303, 0.17254902, 0.        , 0.01818182, 0.        ,
               0.        , 0.        , 0.        ])
```

Figure 2.1.1 Output of Normalization

The classifiers' accuracy changes before and after Normalization were compared. For example, when we use KNN without Normalization, the processing time is no changes, and the accuracy is higher. So, we kept it.

2.1.2 PCA

The singular value decomposition method is used to linearly reduce the dimension of the data and project it into a low dimensional space. This method may cause the curse of dimensionality. In this task, group 174 Reduce the dimension of 784 to 200. Fit the model with train_norm and apply the dimensionality reduction on train_norm. Set n_components=200, which means the number of components to keep is 200.

```
In [10]:  # PCA (after normalization)
          train_norm = train_norm.reshape((train_norm.shape[0],-1))

          from sklearn.decomposition import PCA
          pca = PCA(n_components = 200)
          train_norm_pca = pca.fit_transform(train_norm)

In [11]:  train_norm_pca.shape

Out[11]: (30000, 200)
```

Figure 2.1.2 Output of PCA

The accuracy changes before and after PCA were compared. For example, when we use KNN without PCA, the processing time is slow, but the accuracy is lower. Group 174 does not want to sacrifice accuracy, so, commented out the PCA code.

2.2 Classification algorithms

In this task, group 174 choose three classifiers, which is Nearest Neighbor, Logistic Regression, SVM, and one ensemble method, which is Random forest.

2.3 Parameter Tuning

Key word: grid search, KFold, Visualization, graph display.

In order to obtain a more accurate classification model, group 174 needs to adjust parameters. The main content of this section is what parameters are adjusted for the four classifiers, how to adjust parameters, and what methods are used to adjust parameters.

In the parameter tuning process, group174 uses the method of for loop nesting and enumerating parameters, which is a form of grid search method. The Kfold method is also used, which is cross-validation. Cross-validation is often used for data verification. The principle is to divide data into N groups, and each group of data should be used as a validation set for once, while the other N-1 groups of data are used as training sets. So you do this n times, you validate n times, you get n models, and you average the n errors of these n models, and you get the cross validation errors.

2.3.1 KNeighborsClassifier

Parameters were adjusted to try the influence of different values of K on the accuracy. It was found that inflection point appeared when K was 8 and the peak value was reached. Figure as showed as below.

```
In [73]:
# parameter tunning
k2acc = {}
for k in range(3,11):
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train_f, y_train)
    y_pred = knn.predict(X_test_f)
    knn_accuracy = np.mean(y_pred == y_test)
    k2acc[k] = knn_accuracy

In [74]:
k2acc

Out[74]: {3: 0.8396666666666667,
 4: 0.8471666666666666,
 5: 0.8475,
 6: 0.8491666666666666,
 7: 0.8453333333333334,
 8: 0.849,
 9: 0.8428333333333333,
 10: 0.844}
```

Figure 2.3.1-1 using default metric(minkowski)

```
In [76]:
# parameter tunning
k2acc = {}
for k in range(3,11):
    knn = KNeighborsClassifier(n_neighbors = k, metric = "manhattan")
    knn.fit(X_train_f, y_train)
    y_pred = knn.predict(X_test_f)
    knn_accuracy = np.mean(y_pred == y_test)
    k2acc[k] = knn_accuracy

In [77]:
k2acc

Out[77]: {3: 0.8488333333333333,
 4: 0.85,
 5: 0.8523333333333334,
 6: 0.8536666666666667,
 7: 0.8511666666666666,
 8: 0.8526666666666667,
 9: 0.8493333333333334,
 10: 0.85}
```

Figure 2.3.1-2 using Manhattan distance method

The optimal parameters of N_neighbor, P corresponding to KNN were found by using the method of grid search for for cycle. The trend charts of accuracy and running time were drawn according to different parameters, and the accuracy and running time were written into different corresponding N_neighbor, P by matrix. The parameters of p represent different ways of calculating distance. When p = 1, this is equivalent to using Manhatton distances (L1) and Euclidean distances (L2) for p = 2. For any p, use minkowski_distance (l_p).

```
In [14]:
# use KNN classifier atfer paramenter tunning
import numpy as np
from sklearn.neighbors import KNeighborsClassifier



#set parameter
para_grid_knn = {'n_neighbors':[3,4,5,6,7,8],'p':[1,2]}
#10-fold method
#KFold_knn = StratifiedKFold(n_splits = 10, shuffle = True

#store data
time_knn = []
acc_knn = []

#grid search
for p in para_grid_knn["p"]:
    acc_mean = []
    time_list = []
    for n_neighbors in para_grid_knn["n_neighbors"]:
            acc= []
            start = time()
            knn = KNeighborsClassifier(n_neighbors = n_neighbors, p = p)
            knn.fit(X_train, y_train)
            y_pred = knn.predict(X_test)
            acc.append(accuracy_score(y_pred,y_test))
            acc_mean.append(np.around(np.mean(acc)*100,2))
            time_list.append(time()-start)
            print("n_neighbors = %d, p = %d, Accuracy = %.2f%%, running time = %.2fs"%(n_neighbors,p,acc_mean[-1],time_list[-1]))
    acc_knn.append(acc_mean)
    time_knn.append(time_list)
```

```
n_neighbors = 3, p = 1, Accuracy = 84.48%, running time = 37.88s
n_neighbors = 4, p = 1, Accuracy = 85.28%, running time = 56.28s
n_neighbors = 5, p = 1, Accuracy = 85.42%, running time = 42.92s
n_neighbors = 6, p = 1, Accuracy = 85.62%, running time = 41.07s
n_neighbors = 7, p = 1, Accuracy = 85.32%, running time = 38.66s
n_neighbors = 8, p = 1, Accuracy = 85.55%, running time = 41.16s
n_neighbors = 3, p = 2, Accuracy = 84.88%, running time = 3.12s
n_neighbors = 4, p = 2, Accuracy = 85.43%, running time = 3.70s
n_neighbors = 5, p = 2, Accuracy = 85.30%, running time = 3.70s
n_neighbors = 6, p = 2, Accuracy = 85.87%, running time = 3.77s
n_neighbors = 7, p = 2, Accuracy = 85.45%, running time = 4.02s
n_neighbors = 8, p = 2, Accuracy = 85.63%, running time = 3.83s
```

Figure 2.3.1-3 using parameter grid to find the best accuracy for KNN

According to the accuracy chart drawn according to the data, it can be found that when the corresponding N_neighbor, P equals 6 and 2 respectively, the accuracy is the highest. We can see that when P =2 and N_NEIGHBORS =6, the classifier is most accurate. Figure as showed as below.

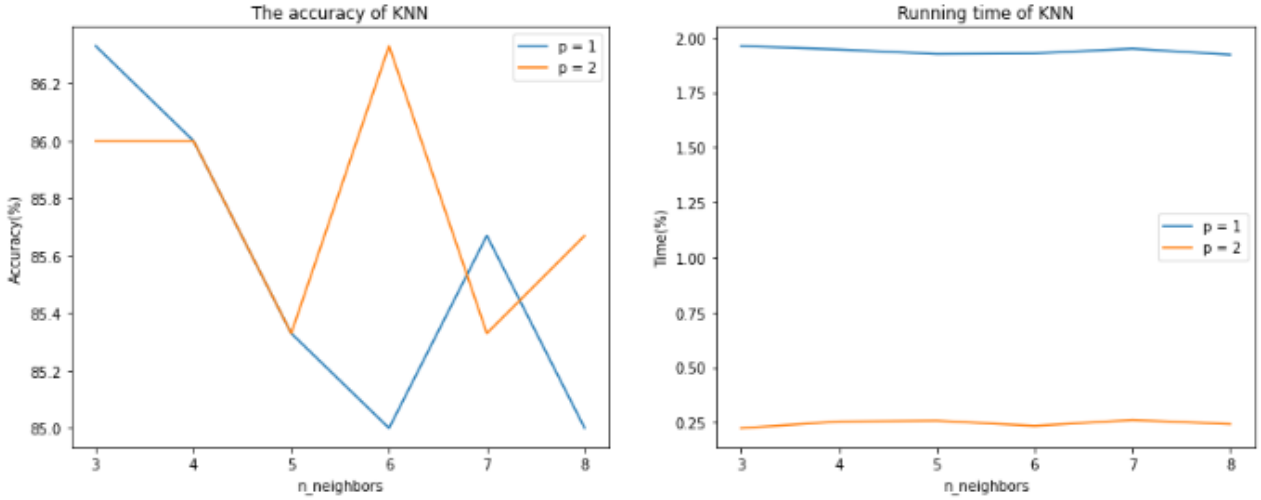


Figure 2.3.1-4 The accuracy and running time of KNN

It is worth mentioning that group174 found that PCA had a great influence on the running time of KNN during the experiment. It takes more than two minutes to train KNN model once before PCA pretreatment, while the running time of PCA dimension reduction 784 to 200 pretreatment only takes more than one second. The following figure shows the results of pca pretreatment.

```
n_neighbors = 3, p = 1, Accuracy = 86.33%, running time = 1.96s
n_neighbors = 4, p = 1, Accuracy = 86.00%, running time = 1.94s
n_neighbors = 5, p = 1, Accuracy = 85.33%, running time = 1.92s
n_neighbors = 6, p = 1, Accuracy = 85.00%, running time = 1.93s
n_neighbors = 7, p = 1, Accuracy = 85.67%, running time = 1.95s
n_neighbors = 8, p = 1, Accuracy = 85.00%, running time = 1.92s
n_neighbors = 3, p = 2, Accuracy = 86.00%, running time = 0.22s
n_neighbors = 4, p = 2, Accuracy = 86.00%, running time = 0.25s
n_neighbors = 5, p = 2, Accuracy = 85.33%, running time = 0.26s
n_neighbors = 6, p = 2, Accuracy = 86.33%, running time = 0.24s
n_neighbors = 7, p = 2, Accuracy = 85.33%, running time = 0.26s
n_neighbors = 8, p = 2, Accuracy = 85.67%, running time = 0.24s
```

2.3.2 SVM

SVM is a traditional machine learning image recognition classifier. In skLearn, it has fewer tunable parameters, so we choose parameters max_iter=5000, max_iter =False. Use Kfold, k=5, to find the accuracy of the cross-validation, and use time function to output the running time.

```
0.8388888888888889
Finish classifying, it takes 0.009974241256713867s
0.8294612794612795
Finish classifying, it takes 0.009477853775024414s
0.8420875420875421
Finish classifying, it takes 0.00803828239440918s
0.8373737373737373
Finish classifying, it takes 0.007039546966552734s
0.8292929292929293
Finish classifying, it takes 0.009000062942504883s
```

Figure2.3.2-1 result of svm without PCA

2.3.3 Logistic regression

Logistic Regression (LR) is a generalized linear Regression analysis model, which belongs to supervised learning algorithm and requires marking data. It can be used in Regression, dichotomy and multiple classification, and dichotomy is the most commonly used. Logistic regression requires that each set of data be numerical, because it needs to be calculated to get linear coefficients, and the marked data are generally 0 and 1.

In this task, we choose solver, max_iter, penalty and tol parameters of logistic regression to adjust. The control variables max_iter and penalty were displayed using visual graph.
It was found that the accuracy was higher when penalty= L1, while max_iter had no significant effect on the accuracy, only on the running time. Figure as showed as below.

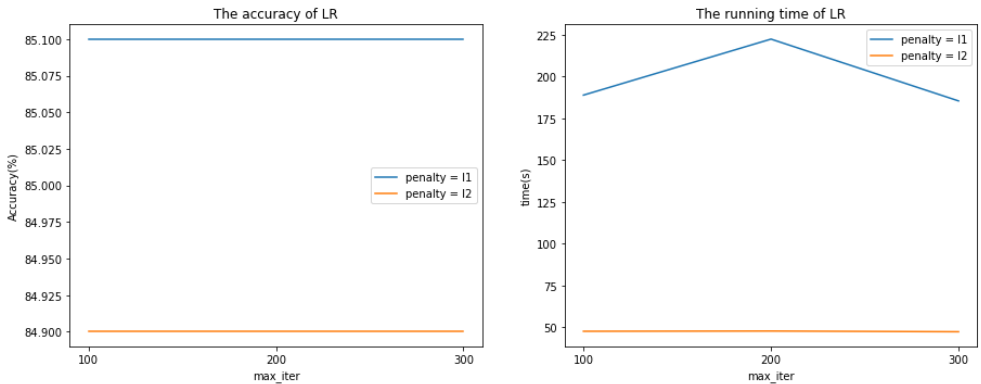


Figure 2.3.3-1 The accuracy and running time of KNN

For preprocessing, it takes about 40s to 200 seconds for logistic regression model to run once using normalization. If both normalization and PCA are used, the runnning time is 0.01 seconds, depending on different parameters. The following figure shows the model accuracy and running time results of PCA treatment and no PCA treatment respectively

```
0.8486531986531987
Finish classifying, it takes 0.007008552551269531s
0.8393939393939394
Finish classifying, it takes 0.011997222900390625s
0.8501683501683501
Finish classifying, it takes 0.0070073604583740234s
0.8459595959595959
Finish classifying, it takes 0.007952690124511719s
0.8412457912457912
Finish classifying, it takes 0.009002447128295898s
```

Figure2.3.3-1 result of lr with PCA

```
max_iter = 100, penalty = l1, Accuracy = 85.10%, running time = 188.89s
max_iter = 200, penalty = l1, Accuracy = 85.10%, running time = 222.37s
max_iter = 300, penalty = l1, Accuracy = 85.10%, running time = 185.43s
max_iter = 100, penalty = l2, Accuracy = 84.90%, running time = 47.73s
max_iter = 200, penalty = l2, Accuracy = 84.90%, running time = 47.87s
max_iter = 300, penalty = l2, Accuracy = 84.90%, running time = 47.45s
```

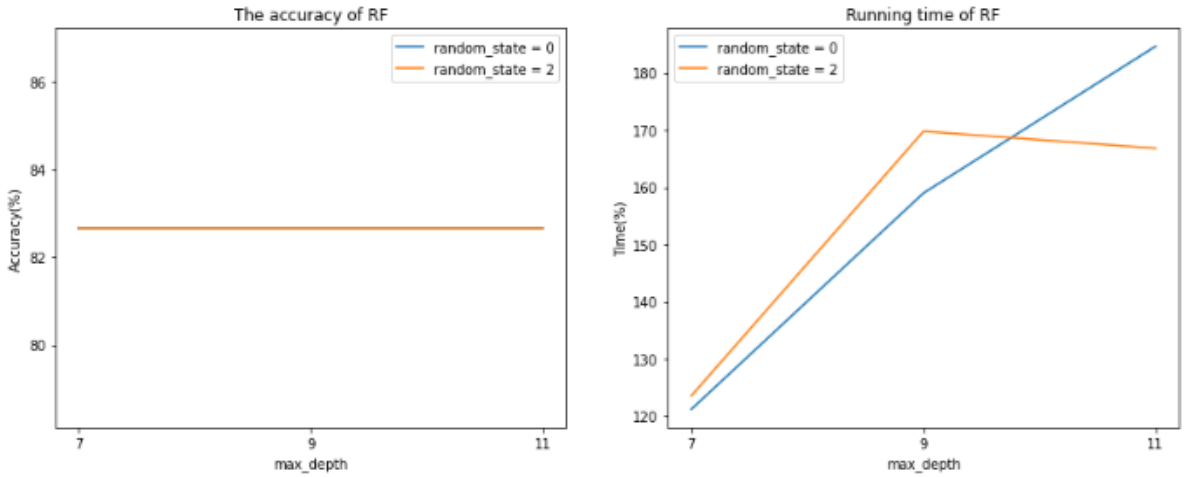
Figure2.3.3-1 result of lr without PCA

2.3.4 Random forest

Random Forest is a classifier composed of multiple decision trees and a supervised learning algorithm, which is mostly trained by bagging method.

In this method, we continue to call skLearn's classifier package.In this task, we choose max_depth, min_samples_leaf _iter, n_estimators, min_samples_split and random_state parameters of random forest to adjust. The control variables max_depth and random_state were displayed using visual graph.

Control variables found that these two parameters had no significant impact on the accuracy, but only on the running time. At max_depth between 7 and 9, max_depth is almost proportional to the running time



For preprocessing, it takes about 120s to 160 seconds for logistic regression model to run once using normalization. If both normalization and PCA are used, the running time is about 0.8 seconds, depending on different parameters. The following figure shows the model accuracy and running time results of PCA treatment and no PCA treatment respectively

```
0.8606060606060606
Finish classifying, it takes 0.8252971172332764s
0.8488215488215488
Finish classifying, it takes 0.5999958515167236s
0.8653198653198653
Finish classifying, it takes 0.6478431224822998s
0.858922558922559
Finish classifying, it takes 0.5890023708343506s
0.8553872053872054
Finish classifying, it takes 0.6040041446685791s
```

Figure2.3.3-1 result of rf with PCA

```
max_depth = 7, random_state = 0, Accuracy = 82.67%, running time = 121.19s
max_depth = 9, random_state = 0, Accuracy = 82.67%, running time = 158.98s
max_depth = 11, random_state = 0, Accuracy = 82.67%, running time = 184.65s
max_depth = 7, random_state = 2, Accuracy = 82.67%, running time = 123.55s
max_depth = 9, random_state = 2, Accuracy = 82.67%, running time = 169.79s
max_depth = 11, random_state = 2, Accuracy = 82.67%, running time = 166.82s
```

Figure2.3.3-1 result of rf without PCA

2.4 Classifier comparisons

After using grid search mentioned before, group 174 get the best parameter for all of the classifier. Using the KFold method, when K =5, we train the classifier and get a group of data with 5 precision data, and then take the average value to get the accuracy of the final classifier. Compare each classifier.

Use the max() function to find the most accuracy classifier. Compare and get the best accuracy classifier and output the test set data.

3 Experiments result and discussion

Experimental results show that the performance of Random forest accuracy is good, SVM is poor, KNN and logistic regression are normal. There was little difference between the various classifiers, ranging from 84 to 85%. However, for running time, the performance of each classifier is very different. KNN training model takes the most time, while logistic regression only takes about 0.007 seconds, which is the highest efficiency.

Classifier	preprocessing	Accuracy (%)	Running time	Parameter tunning for best performing
KNeighbors	Normalization	85.003%	159s	n_neighbors = 6, p = 1
	Normalization+PCA	84.83%	1.96s	n_neighbors = 3, p = 1
SVM	Normalization	83.542%	0.009s	max_iter=5000,dual=False
Logistic regression	Normalization	84.508%	179.74s	solver='liblinear', max_iter=300, penalty='l2', tol=0.001
	Normalization+PCA	83.00%	0.007s	

Random forest	Normalization	85.781%	121.19s	max_depth=11, min_samples_leaf=2, n_estimators=500, min_samples_split=10, random_state=0
	Normalization+PCA	82.67%	0.825s	

After sorting out the data, it was found that random forest had the highest accuracy, and logistic regression had the shortest running time after normalization and PCA pretreatment. In addition, Group 175 found a trend that for the data processed by PCA, the accuracy would be reduced a little and the running time would be greatly reduced.

The reason for this phenomenon may be that after dimensionality reduction, less data needs to be processed during operation, so the running time is shorter. In addition, the trained data reduces many attributes. Although the efficiency can be improved, some important attributes may be neglected, so the classification accuracy after training becomes low.

4 Conclusion

Normalization and PCA are both good methods for preprocessing, but it should be noted that PCA may reduce accuracy while reducing running time. Grid search function allows us to find the best parameter, which is a good helper for parameter tuning, but the disadvantage is that it takes too long to run. Kfold allows us to make full use of training data when training models.

For KNN, he performed well in the image classification task. His accuracy is stable with or without two pretreatments. For the SVM. In this case, we had to tweak a few parameters, and it performed just fine, but it was fast. For logistic regression in experiments, it is found that logistic regression is faster after dimensionality reduction, and the accuracy of high-dimensional features is lower. For random forest, the accuracy is high, but a little slow. One of the advantages of stochastic forest algorithm is to make the model more robust by using randomness.

5. reference

Zhou, & Liu, S. (2021). Machine Learning. Springer Singapore Pte. Limited.

Machine learning (Online). (1986). Kluwer Academic Publishers.

API reference. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/classes.html

Sklearn.ensemble.randomforestclassifier. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sklearn.linear_model.logisticregression. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Sklearn.metrics.accuracy_score. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

Sklearn.model_selection.Kfold. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Sklearn.neighbors.kneighborsclassifier. scikit. (n.d.). Retrieved April 8, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Appendix 1 System Information

System Information

Time of this report: 4/5/2022, 20:57:04
Machine name: DESKTOP-0H97T1N
Machine Id: {2A87F8D7-8B63-4B50-9A30-36FDF38E82AC}
Operating System: Windows 11 Family 64-bit (10.0, Build 22000) (22000.co_release.210604-1628)
Language: Chinese (Simplified) (Regional Setting: Chinese (Simplified))
System Manufacturer: LENOVO
System Model: 81C4
BIOS: 8GCN32WW (type: UEFI)
Processor: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (8 CPUs), ~2.0GHz
Memory: 16384MB RAM
Available OS Memory: 16226MB RAM
Page File: 19792MB used, 4626MB available
Windows Dir: C:\WINDOWS
DirectX Version: DirectX 12
DX Setup Parameters: Not found
User DPI Setting: 288 DPI (300 percent)
System DPI Setting: 288 DPI (300 percent)
DWM DPI Scaling: UnKnown
Miracast: Available, with HDCP
Microsoft Graphics Hybrid: Not Supported
DirectX Database Version: 1.2.2
DxDiag Version: 10.00.22000.0001 64bit Unicode

DxDiag Notes

Appendix 2 Instructions on how to run the code

The code in the submitted file consists of four parts. The first part is input data, which is completed according to tutorial content, and the second part is pre-processing, which is completed by normalization and PCA. The third part is the use of four classifiers, which is divided into parameters that have been adjusted. The fourth part uses the label derived from the test set and outputs it in the form of CVS file.