The Experiment Report of Machine Learning



Student ID：

201720144894|201721045763|201710106598

Author:

ChangxiZhu|YuzhongYe| YangyangZhao

December 21, 2017

Grade:

 Graduate

Supervisor:

 Qingyao Wu

**SUBJECT:**SOFTWARE ENGINEERING

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

# Face Classification Based on AdaBoost Algorithm

Abstract—This paper describes the classification of humans based on Adabooost algorithm. Adabooost have some confusion about is why each iteration to wrong point weight greaten, perhaps by mistake can make the back of the classifier weights are higher, and finish the experiment found our guess is very close to but not entirely, by error of full-time, the next time a classifier after points wrong these points, will enhance the overall error rate, thus led to a smaller, eventually led to the classifier in the hybrid classifier weights of the land.That is to say, this algorithm makes good classifiers account for a higher weight, while the wrong classifier weights are lower.

# INTRODUCTION

In order for us to understand Adaboost further and get damiliar with the basic method of face detection,more importantly,It is useful for us to use Adaboost to solve the face classification problem,and conbine the theory with the actual project.We experience the project “Face Classification Based on AdaBoost Algorithm”to feel the complete process of machine learning.Futhermore,we use 1000pictures as the dataset ,of which 500 are human face RGF images,stored in datasets/original/face,the other 500 is a non-face RGB images,stored in datasets/original/nonface,which included in the example repository.We experience the poject in PyCharm Community ,which has built- in python package above.

# METHODS AND THEORY

According to the methods provided by teacher Wu (Steps 1-5 below )and we do it.

1. Read data set data. The images are supposed to converted into a size of 24 \* 24 grayscale, the number and the proportion of the positive and negative samples is not limited, the data set label is not limited.

def import\_data(name, convert\_type='L', resize=None):

Data import, judge for file or directory

|  |  |  |
| --- | --- | --- |
| Param | Name | the file or directory name entered |
| Convert type | the type that needs to be converted, which defaults to grayscale |
| Resize | the size that you want to give back is None |
| Return | return an Image object if it is a file, otherwise it returns an Image list, if both are not | |

1. Processing data set data to extract NPD features. Extract features using the NPDFeature class in feature.py. (Tip: Because the time of the pretreatment is relatively long, it can be pretreated with pickle function library dump () save the data in the cache, then may be used load () function reads the characteristic data from cache.)

def prepare\_data() and class PDFeature()

It is a tool class to extract the NPD features.

|  |  |
| --- | --- |
| Attributes: | |
| image | A two-dimension ndarray indicating grayscale image |
| n\_pixels | An integer indicating the number of image total pixels |
| features | A one-dimension ndarray to store the extracted NPD features |

1. The data set is divisded into training set and calidation set, this experiment does not divide the test set.

def split\_data(data, label, train\_size=0.8):

The shard data sets are training sets and test sets

|  |  |  |
| --- | --- | --- |
| Param | Data | data set that needs to be syncopated, list type |
| Label | The corresponding label |
| Train\_size | the size of the training set is divided, the default is 0.8, and the default test set is 0.2 |
| Return | the data set after segmentation | |

4.Write all AdaboostClassifier functions based on the reserved interface in ensemble.py. The following is the guide of fit function in the AdaboostClassifier class:

4.1 Initialize training set weights , each training sample is given the same weight.

4.2Training a base classifier , which can be sklearn.tree library DecisionTreeClassifier (note that the training time you need to pass the weight  as a parameter).

4.3 Calculate the classification error rate of the base classifier on the training set.

4.4 Calculate the parameter according to the classification error rate .

4.5 Update training set weights .

4.6 Repeat steps 4.2-4.6 above for iteration, the number of iterations is based on the number of classifiers.

5.Predict and verify the accuracy on the validation set using the method in AdaboostClassifier.

if \_\_name\_\_ == '\_\_main\_\_':

This is the preparation of data, including reading images and extracting features

**Adaboost:**

AdaBoost is an iterative algorithm, and its core idea is different for the same training set training classifier, namely the weak classifier, then put these weak classifier together, to build a stronger final classifier.(many bloggers say that the three cobblers are better than zhuge liang.) The algorithm itself is the realization of data distribution, change it according to the classification of each sample of each training set is correct, the overall classification accuracy as well as the last time, to determine the weights of each sample.The new data for the right value is given to the lower-class classifier for training, and then the classifiers that are received each time are integrated as the final decision classifier.

# Experiment

We did the experiment with classifier 2, 5, 6, 7, 10, and listed the results of 2, 5, 10 below.

Number of classifiers：1

precision recall f1-score support  
  
 -1 0.82 0.81 0.82 96  
 1 0.83 0.84 0.83 104  
  
avg / total 0.82 0.82 0.82 200

Number of classifiers：2

predict and result...

base classifier: precision recall f1-score support

-1 0.82 0.81 0.82 96

1 0.83 0.84 0.83 104

avg / total 0.82 0.82 0.82 200

final classifier: precision recall f1-score support

-1 0.82 0.81 0.82 96

1 0.83 0.84 0.83 104

avg / total 0.82 0.82 0.82 200

Number of classifiers：5

read dump data...

split image data...

train adaboost classifier...

x shape is: (800, 165600) y shape is: (800,)

iteration times: 0 precision recall f1-score support

-1 0.91 0.89 0.90 390

1 0.90 0.92 0.91 410

avg / total 0.91 0.91 0.90 800

iteration times: 1 precision recall f1-score support

-1 0.85 0.92 0.88 390

1 0.92 0.85 0.88 410

avg / total 0.89 0.88 0.88 800

iteration times: 2 precision recall f1-score support

-1 0.83 0.76 0.80 390

1 0.79 0.86 0.82 410

avg / total 0.81 0.81 0.81 800

iteration times: 3 precision recall f1-score support

-1 0.86 0.83 0.85 390

1 0.84 0.88 0.86 410

avg / total 0.85 0.85 0.85 800

iteration times: 4 precision recall f1-score support

-1 0.74 0.85 0.79 390

1 0.84 0.72 0.77 410

avg / total 0.79 0.78 0.78 800

predict and result...

each base classifier: precision recall f1-score support

-1 0.84 0.79 0.82 110

1 0.76 0.82 0.79 90

avg / total 0.81 0.81 0.81 200

each base classifier: precision recall f1-score support

-1 0.84 0.89 0.87 110

1 0.86 0.80 0.83 90

avg / total 0.85 0.85 0.85 200

each base classifier: precision recall f1-score support

-1 0.83 0.68 0.75 110

1 0.68 0.83 0.75 90

avg / total 0.77 0.75 0.75 200

each base classifier: precision recall f1-score support

-1 0.87 0.84 0.85 110

1 0.81 0.84 0.83 90

avg / total 0.84 0.84 0.84 200

each base classifier: precision recall f1-score support

-1 0.76 0.87 0.81 110

1 0.81 0.66 0.72 90

avg / total 0.78 0.78 0.77 200

final classifier: precision recall f1-score support

-1 0.93 0.93 0.93 110

1 0.91 0.91 0.91 90

avg / total 0.92 0.92 0.92 200

Number of classifiers：10

C:\Users\chauncy\Anaconda3\python.exe D:/Coding/pycharm-professional/pycharm-file/deep\_learning\_classthree/train.py

read dump data...

split image data...

train adaboost classifier...

x shape is: (800, 165600) y shape is: (800,)

iteration times: 0 precision recall f1-score support

-1 0.89 0.90 0.90 403

1 0.90 0.89 0.90 397

avg / total 0.90 0.90 0.90 800

iteration times: 1 precision recall f1-score support

-1 0.84 0.90 0.87 403

1 0.89 0.83 0.86 397

avg / total 0.87 0.87 0.87 800

iteration times: 2 precision recall f1-score support

-1 0.78 0.90 0.83 403

1 0.88 0.74 0.80 397

avg / total 0.83 0.82 0.82 800

iteration times: 3 precision recall f1-score support

-1 0.88 0.89 0.88 403

1 0.88 0.87 0.88 397

avg / total 0.88 0.88 0.88 800

iteration times: 4 precision recall f1-score support

-1 0.88 0.74 0.80 403

1 0.77 0.89 0.83 397

avg / total 0.82 0.82 0.82 800

iteration times: 5 precision recall f1-score support

-1 0.80 0.74 0.77 403

1 0.76 0.81 0.78 397

avg / total 0.78 0.78 0.78 800

iteration times: 6 precision recall f1-score support

-1 0.85 0.86 0.86 403

1 0.86 0.85 0.85 397

avg / total 0.86 0.85 0.85 800

iteration times: 7 precision recall f1-score support

-1 0.83 0.81 0.82 403

1 0.81 0.83 0.82 397

avg / total 0.82 0.82 0.82 800

iteration times: 8 precision recall f1-score support

-1 0.81 0.94 0.87 403

1 0.93 0.78 0.85 397

avg / total 0.87 0.86 0.86 800

iteration times: 9 precision recall f1-score support

-1 0.79 0.81 0.80 403

1 0.80 0.79 0.79 397

avg / total 0.80 0.80 0.80 800

predict and result...

each base classifier: precision recall f1-score support

-1 0.85 0.84 0.84 97

1 0.85 0.86 0.86 103

avg / total 0.85 0.85 0.85 200

each base classifier: precision recall f1-score support

-1 0.82 0.87 0.84 97

1 0.87 0.83 0.85 103

avg / total 0.85 0.84 0.85 200

each base classifier: precision recall f1-score support

-1 0.72 0.92 0.81 97

1 0.89 0.66 0.76 103

avg / total 0.81 0.79 0.78 200

each base classifier: precision recall f1-score support

-1 0.82 0.82 0.82 97

1 0.83 0.83 0.83 103

avg / total 0.83 0.82 0.83 200

each base classifier: precision recall f1-score support

-1 0.74 0.69 0.71 97

1 0.72 0.77 0.75 103

avg / total 0.73 0.73 0.73 200

each base classifier: precision recall f1-score support

-1 0.81 0.62 0.70 97

1 0.71 0.86 0.78 103

avg / total 0.76 0.74 0.74 200

each base classifier: precision recall f1-score support

-1 0.78 0.77 0.78 97

1 0.79 0.80 0.79 103

avg / total 0.78 0.79 0.78 200

each base classifier: precision recall f1-score support

-1 0.79 0.74 0.77 97

1 0.77 0.82 0.79 103

avg / total 0.78 0.78 0.78 200

each base classifier: precision recall f1-score support

-1 0.77 0.91 0.83 97

1 0.90 0.75 0.81 103

avg / total 0.84 0.82 0.82 200

each base classifier: precision recall f1-score support

-1 0.76 0.76 0.76 97

1 0.78 0.78 0.78 103

avg / total 0.77 0.77 0.77 200

final classifier: precision recall f1-score support

-1 0.94 0.95 0.94 97

1 0.95 0.94 0.95 103

avg / total 0.95 0.94 0.95 200

Process finished with exit code 0

# conclusion

Adabooost have some confusion about is why each iteration to wrong point weight greaten, perhaps by mistake can make the back of the classifier weights are higher, and finish the experiment found our guess is very close to but not entirely, by error of full-time, the next time a classifier after points wrong these points, will enhance the overall error rate, thus led to a smaller, eventually led to the classifier in the hybrid classifier weights of the land.That is to say, this algorithm makes good classifiers account for a higher weight, while the wrong classifier weights are lower.Finally, we can summarize some of the actual scenarios that can be used by the adaboost algorithm:1) application scenarios for binary classification or multi-classification2) baseline for classification tasksNo brainification, simple, no overfitting, no classifier3) feature selection4) framework used for Boosting badcase correctionYou only need to add new classifiers, you don't need to change the original classifierBecause the adaboost algorithm is a simple and easy to implement algorithm.Adaboost algorithm is obtained by combination of weak classifier and strong classifier, both upper and lower bounds of the classification error rate with the increase of training and steady decline, does not have the nature of the fitting and so on, should say is a kind of very suitable for scenarios used in a variety of classification algorithm.