South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERIN
G

SUBJECT: SOFTWARE ENGINEERIN
G

Author:
Shoukai Xu and Yaofu Chen
Yujia Zhang and Caili Li

Supervisor:
Qingyao Wu

Student ID：
201530611111 and 20153060000
201530611821 and
201530613566

Grade:
Undergraduate

December 9, 2017

Face Classification Based on AdaBoost Algorithm

Abstract—

I. Introduction

AdaBoost is in English, "the Adaptive Boosting" (Adaptive enhancement), by Yoav Freund and Robert Schapire put forward in 1995. Its self-adaptation is that the sample of the previous basic classifier is strengthened and the weighted group of samples is used again to train the next basic classifier. At the same time, a new weakly classifier is added to each round until it reaches a predetermined error rate or the maximum number of iterations specified. Specifically, the entire Adaboost iteration algorithm is 3 steps:
The weight distribution of the initial training data. If there are N samples, each of the training samples will initially be given the same weight: 1/ N.
Train the weak classifier. In the specific training process, if a sample point has been accurately classified, then the weight of the next training set will be reduced. Conversely, if a sample point is not classified correctly, its weight is improved. The updated sample set is then used to train the next classifier, and the whole training process goes on so iteratively.
Combine the weak classifier of various training into strong classifier. Each weak classifier after the training process, increase the weight of weak classifier classification error rate is small, in the final classification function plays a decisive role, and reduce classification error rate of the weak classifier weights, in the final classification function plays a smaller decision role. In other words, the weak classifier with low error rate occupies a large weight in the final classifier, otherwise it is small.

II. Methods and Theory

$$D_1=(w_{11},w_{12}\cdots w_{1i}\cdots,w_{1N}),\ w_{1i}=\frac{1}{N},\ i=1,2,\cdots,N$$

$$G_m(x):\ \chi\rightarrow\{-1,+1\}$$

$$e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$$

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$$

$$D_{m+1}=(w_{m+1,1},w_{m+1,2}\cdots w_{m+1,i}\cdots,w_{m+1,N}),$$

$$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\ i=1,2,\cdots,N$$

$$Z_m=\sum_{i=1}^{N}w_{mi}\exp(-\alpha_m y_i G_m(x_i))$$

代码对应如下：

```
self.wclassifier.fit(X, y, sample_weight=self.sample_weight)#训练m个分类器
pre_y = self.wclassifier.predict(X).reshape(-1,1)
y=y.reshape(-1,1)
print(y)
print(pre_y)
    #如果不正确加入误差率计算
for i in range(self.n):
    if y[self.n:]!=pre_y[self.n:]:
        self.e[m] += samble_weight[self.n] #误差率为错误的权重之和
if self.e[m]==0:
    self.a.append(self.alpha)
    self.pre.append(self.wclassifier)
else:
    alpha=0.5*math.log((1-self.e[m])/self.e[m]) #这是更新参数的参数
    self.a.append(self.alpha)
    Z=self.sample_weight.sum(dtype=np.float64)*exp(-alpha*self.pre_y*y)


    self.sample_weight=self.samble_weight.reshape(-1,1)*exp(-alpha*self.pre_y*y)
    self.pre.append(self.wclassifier)
```

III. Expriments

A.

100data

training data:80*2(80-face,80-nonface)

validation data:20*2

B.

```
def __init__(self, weak_classifier, n_weakers_limit):


    self.n_weakers_limit=10
    self.depth=3
    #弱分类器暂设为10，可在后期调整。
    self.n= 80*2 #400个样本 即x.shape[0]=80
    self.sample_weight= [1./self.n]*self.n#400个样本最开始

    self.a=[]#初始化参数
    self.alpha=1#初始化参数
    self.e=[0]* self.n_weakers_limit#初始化误差率e[m]
    self.z=[0]* self.n_weakers_limit#初始化规范化因子z[m]
    self.pre=[]
    self.f=[]
    self.wclassifier=weak_classifier
```

proocess:
Read dataset data. Read the picture, all the pictures into a size of 24 * 24 grayscale, positive and negative samples of the data set the number of samples and the proportion is not limited, the data set label is not limited.
Processing dataset data and extracting NPD features. Extract features using the NPDFeature class in feature.py. (Hint: You can use the dump () function in the pickle library to save the preprocessed signature data to the cache because the data set can be preprocessed for a long time. You can then use the load () function to read the signature data)
The data set is divided into training set and verification set, this experiment does not divide the test set.
Write all AdaboostClassifier functions based on the reserved interface in ensemble.py. The following is the idea of the fit () method in the AdaboostClassifier class:
4.1 Initialize the training set weights, each training sample is given the same weight.
4.2 Training a base classifier, the base classifier can use the sklearn.tree library DecisionTreeClassifier (note that when training 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 According to the classification error rate, calculate the parameters.
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.
Predict and calculate the accuracy on the validation set using the method in AdaboostClassifier and write the prediction result to report.txt using the classification_report () function of the sklearn.metrics library.
Sort out the experimental results and complete the experimental report.

Result:

```
             precision    recall  f1-score   support

       face      0.941     0.889     0.914        18
    nonface      0.913     0.955     0.933        22

avg / total      0.926     0.925     0.925        40
```

Most parameters are all zero initialization
e is the error rate is the weight of misprediction and alpha as a parameter
z is a normalization factor such that d (m + 1) becomes a probability distribution
n is the training sample
w is the weight, which is initialized to 1 / n, each sample has the same weight initially

IV. CONCLUSION

In this experiment I have learned the basic principle of adaboost algorithm, and realized the coding.Anymore,I also looked at the paper in the process of learning, enhancing the ability of reading papers.And I knew how to training the classifier and deepened the understanding of the decision tree. And I I found in the experimental process can get a lot of knowledge from papers, often after reading the paper process can get a lot of knowledge from papers, which make me often rea
d the papers.