華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Zhuoman Liu and Feng Chen and
Congcong Zhan

Supervisor:
Mingkui Tan

Student ID:
201530612415 and 201530613573 and
201530611159

Grade:
Undergraduate

December 22, 2017

Face Classification Based on AdaBoost Algorithm

***Abstract*—This article describes the adaboost to solve the face classification problem.**

I. Introduction

ADABOOST is an iterative algorithm. Its main idea is to train different weak classifiers for the same training set, and then combine these weak classifiers to form a stronger final classifier. The algorithm itself is achieved by changing the data distribution, which determines the weight of each sample based on the correct classification of each sample in each training set and the accuracy of the last overall classification. The new data set with modified weights is sent to the next classifier for training. Finally, the classifier obtained by each training is finally merged as the final decision classifier.

a) : This experiment provides 1000 pictures, of which 500 are human face RGB images, stored in datasets/original/-face; the other 500 is a non-face RGB images.Then we should implement a model based on the principle of Adaboost for face classification.

b) : In terms of evaluation, we will use classification_report() of the sklearn.metrics library function.

II. Methods and Theory

Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

Adaboost is one of boosting methods which works well on boosting the performance of decision trees for binary classification. Some weak classifiers are prepared on the training data using the weighted samples in Adaboost. Then each decision stump makes one decision on one input variable and outputs a $+1.0$ or -1.0 value for the first or second class value. After a round of decision, the training weights are updated giving more weight to incorrectly predicted samples, and less weight to correctly predicted samples. So the weight of a training sample will be updated using

$$w_{,m+1}(i) = \frac{w_m(i)}{z_m} e^{-\alpha_m y_i h_m(\mathbf{x}_i)} \tag{1}$$

where $z_m = \sum_{i=1}^n w_m(i) e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$ is normalization term, making $w_m(i)$ become probability distributions. That is,

$$w_{m+1}(i) = \begin{cases} \dfrac{w_m(i)}{z_m} e^{-\alpha_m} & for\ right\ predictive\ sample \\ \dfrac{w_m(i)}{z_m} e^{\alpha_m} & for\ wrong\ predictive\ sample \end{cases} \tag{2}$$

Algorithm 1 Adaboost

```
Input: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$,where $\mathbf{x}_i \in X, y_i \in \{-1, 1\}$
Initialize: Sample distribution $w_m$
Base classifier: $\mathcal{L}$
1: $w_1(i) = \frac{1}{n}$
2: for m=1,2,...,M do
3:     $h_m(x) = \mathcal{L}(D, w_m)$
4:     $\epsilon_m = \sum_{i=1}^n w_m(i)\mathbb{I}(h_m(\mathbf{x}_i) \neq y_i)$
5:     if $\epsilon_m > 0.5$ then
6:         break
7:     $\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}$
8:     $w_{m+1}(i) = \frac{w_m(i)}{z_m} e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$,where $i = 1, 2, ..., n$
   and $z_m = \sum_{i=1}^n w_m(i) e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$
Output: $H(\mathbf{x}) = \sum_{m=1}^M \alpha_m h_m(\mathbf{x})$
```

then we have $\frac{w_{wrong}(i)}{w_{right}(i)} = e^{2\alpha_m} = \frac{1-\epsilon_m}{\epsilon}$ and $\epsilon < 0.5$, therefore wrong sapmles will be more important. And performance of the base classifier is evaluated by its error rate,

$$\epsilon_m = p(h_m(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^n w_m(i)\mathbb{I}(h_m(\mathbf{x}_i) \neq y_i) \tag{3}$$

Afterwards, Adaboost generates a new base classifier $h_m(\mathbf{x})$ and its importance score α_m of every iteration. And making the base classifier with lower ϵ_m more important using Eq.4.

$$\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m} \tag{4}$$

$$H(\mathbf{x}) = sign(\sum_{m=1}^M \alpha_m h_m(\mathbf{x})) \tag{5}$$

Finally, we can obtain the final classifier(Eq.5) and the algorithm of Adaboost is shown as Alg.1.

III. Experiment

A. Dataset

This experiment provides 1000 pictures, of which 500 are human face RGB images, stored in datasets/original/face; the other 500 is a non-face RGB images, stored in datasets/original/nonface.

B. Implementation

Initialization:

Y(1000,): The label 1 of the first 500 is face, The label -1 of the last 500 is nonface

classifyNum=10: the number of the base classify is 10

Process and MainCodes:

1. 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.

```
im = Image.open(imgDir+imgFoldName+"/"+imgs[i])
im = im.resize((24, 24),Image.ANTIALIAS)
im = im.convert("L")
```

2. 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.)

```
feature = F.NPDFeature(X[i])
f = feature.extract()
```

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

```
X_train,X_test,y_train,y_test=train_test_split
(features, y, test_size=0.33, random_state=42)
```

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 ω , each training sample is given the same weight.

```
 w = numpy.ones((X.shape[0],))/X.shape[0]
```

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

```
weak = self.weak_classifier(max_depth=2)
weakClassify=weak.fit(X,y,sample_weight=w)
```

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

```
# sum of the weights of the misclassified instances
minErr+=w[j]
```

4.4. Calcualte the parameter α according to the classification error rate ε .

```
alpha = float(numpy.log((1.0 - minErr)/minErr)/2.0)
```

4.5. Update training set weights ω .

```
w = w*(numpy.e**(-1*alpha*y*y_pre))
w = w/w.sum()
```

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 and use classification_report () of the sklearn.metrics library function writes predicted result to report.txt(TableI) .

6. Organize the experiment results and complete the lab report (the lab report template will be included in the example repository).

TABLE I: Report Result

	precison	recall	f1-score	support
label_1	0.969	0.952	0.960	165
label_-1	0.952	0.970	0.961	165
avg / total	0.961	0.961	0.961	330

IV. Conclusion

In this experiments, we understand adaboost further and get familiar with the basic method of face detection. Here's a summary of Adaboost's advantages and disadvantages.

Advantages:

1.as a classifier, the classification accuracy is high

2.Under the framework of Adaboost, various regression classification models can be used to build weak learning machines, which are very flexible.

3.As a simple binary classifier, the structure is simple and the result is understandable.

4.It's not easy to overfit.

Disadvantages:

In the case of abnormal sample sensitivity, the abnormal samples may obtain higher weights in the iteration, which will affect the accuracy of the prediction of the final strong learner.