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:
Muyi Li, Weizhao Li, Dong Liang

Supervisor:
Mingkui Tan

Student ID:
201530612019,
201530611890,
201530651292

Grade:
Undergraduate

December 21, 2017

Face Classification Based on AdaBoost Algorithm

Abstract— Face classification, an important part of human face analysis, deals with whether the image is a human face. There are tons of methods to solve the face classification problem. In this experiment, we used Adaboost to solve the problem, and tested the performance of this method. From the experiment, this algorithm adaptively adjusted the error rate of the hypothesis, based on the feedback of weak learning, and it produced higher detection accuracy with stable efficiency.

I. Introduction

People used to use boosting method to make weak learning promoted to the strong learning, and made the machine learning more convenient. In 1995, an algorithm called Adaptive Boosting was purposed by Freund and Schapire. Instead of requiring to know the lower limit of hypothesis error in advance, Adaboost can solve problems efficiently without prior knowledge of weak learner's performance. And it has received much attention in machine learning since it was purposed. Therefore, in this experiment, we aim to use Adaboost to solve the face classification problem.

II. Methods and Theory

AdaBoost

As its name, the self-adaptation of this algorithm reflects in its adaption to weak classifier's training error rate. More specifically, given a certain training dataset D with x_i and y_i.

All samples are given the same weight as $\frac{1}{n}$. Then, after iterations, we get a base classifier

$$h_m(x): x \longrightarrow \{-1,1\}$$

and we can calculate the error rate:

$$\epsilon_m = p(h_m(x) \neq y_i) = \sum_{i=1}^{n} \omega_m(i)\, I(h_m(x) \neq y_i)$$

A new weak classifier is added to each round until it reaches a predetermined error rate or the maximum of iterations specified.

Then, we calculate the coefficient of h, and α_m :

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

represents the importance of $h_m(x)$ in the final classifier. So that we can get the weight of the basic classifier in the final classifier. As we can that, when

$$\epsilon_m \leq \frac{1}{2}, \alpha_m \geq 0$$

and α_m decrease with the increase of ϵ_m, which means the smaller the classification error rate is, the greater role basic classifier plays in the final classifier.

What's more, when we update weights of training set,

$$\omega_{m+1}(i) = \frac{\omega_m(i)}{z_m} e^{-\alpha_m y_i h_m(x_i)}$$

where i=1,2,3…n and the normalization term z_m, which makes $\omega_m(i)$ become probability distributions:

$$z_m = \sum_{i=1}^{n} \omega_m(i) e^{-\alpha_m y_i h_m(x_i)}$$

So, we get new weight distribution:

$$\omega_{m+1}(i) = \begin{cases} \frac{\omega_m(i)}{z_m} e^{-\alpha_m} & \text{For right predictive sample} \\ \frac{\omega_m(i)}{z_m} e^{\alpha_m} & \text{For wrong predictive sample} \end{cases}$$

and in next round,

$$\frac{\omega_{wrong}(i)}{\omega_{right}(i)} = e^{2\alpha_m} = \frac{1-\epsilon_m}{\epsilon_m} \quad and \quad \epsilon_m < 0.5$$

the samples of the previous basic classifier(learner) are strengthened, and the weighted samples are used to train the next based learner.

In the end, the final learner to be output is:

$$H(x) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x) \right)$$

III. Experiment

A. *Dataset*

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

B. *Steps*

1. Read dataset. The images are supposed to be converted into grayscales with size of 24 * 24. Besides, the number and the proportion of the positive and negative samples are not limited, and the data set label is not limited either.
2. Process data set data to extract NPD features. Extract features using the "NPDFeature" class in feature.py.
3. Divide the preprocessed dataset into training set and validation set.
4. Complete the "AdaboostClassifier" class.
    1) Initialize training set weights ω, each training sample is given the same weight.

2) Train a base classifier. And pass the weight as a parameter.
3) Calculate the classification error rate ϵ of the base classifier on the training set.
4) Calculate the parameter α according to the classification error rate.
5) Update training set weights ω.
6) Repeat step 2 to 6 above, for iteration.

5. Predict and verify the accuracy on the validation set using the methods in the AdaboostClassifier and write the predicted result to “report.txt” with the function “classification_report”.

C. *Result*

The results of this experiment is as below:

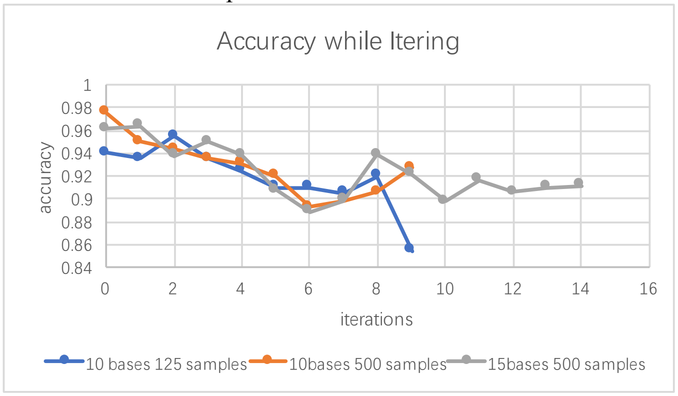
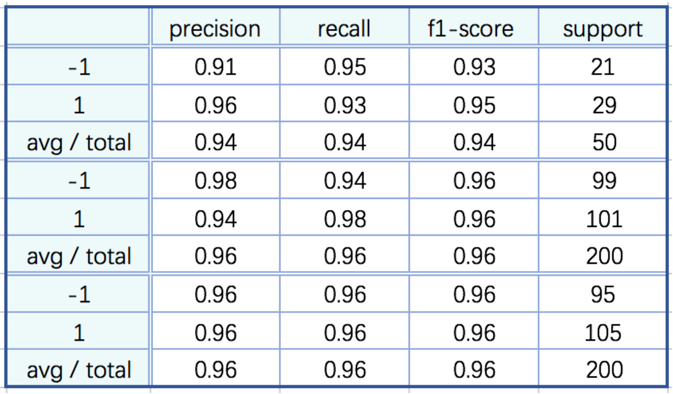


Fig 1. Accuracy of each iteration



	precision	recall	f1-score	support
-1	0.91	0.95	0.93	21
1	0.96	0.93	0.95	29
avg / total	0.94	0.94	0.94	50
-1	0.98	0.94	0.96	99
1	0.94	0.98	0.96	101
avg / total	0.96	0.96	0.96	200
-1	0.96	0.96	0.96	95
1	0.96	0.96	0.96	105
avg / total	0.96	0.96	0.96	200

Fig 2. The report of each time

From the results shown above, we can see that the accuracy of predictions decreases at first and becomes almost stable with more iterations. Moreover, the average accuracy of Adaboost classification is over 0.94. There is a slight increase when we use the more samples, and more iterations.

IV. CONCLUSION

Adaboost uses a number of weak classifier by means of certain methods to boost them up so as to construct a strong classifier with stronger classification ability. In this experiment, we realized the Adaboost algorithm for face detection and tested the performance of this method. Also, we have found that Adaboost operates effectively with high accuracy, and maybe when the number of simple classifier tends to infinity, the error rate will be 0.