
# The Experiment Report of Machine Learning

## SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

## SUBJECT: SOFTWARE ENGINEERING

Author:
鲍淞琳，肖纾予 and 赵一鸣

Supervisor:
Mingkui Tan

Student ID：
201530611029，201530613160
and 201530661741

Grade:
Undergraduate

December 21, 2017

# Face Classification Based on AdaBoost Algorithm

**Abstract—Face recognition is an important image technology with extensive application. This paper describes the basic principles that using adaboost arithmetic to achieve face classification. Finally, we verify the accuracy on the validation set using the method in AdaboostClassifier.**

## I. INTRODUCTION

Face Detection, generally refers to a given image to determine whether the image information contain faces. The face detection algorithm which Paul Viola and Michael Jones made in 2001 based on the method of Adaboost is the current hotspot, the method is simple, fast testing, implementation of good.

And the motivation of this experiment is:

(1) Understand Adaboost further

(2) Get familiar with the basic method of face detection

(3) Learn to use Adaboost to solve the face classification problem, and combine the theory with the actual project

(4) Experience the complete process of machine learning

## II. METHODS AND THEORY

**AdaBoost algorithm**

AdaBoost is a machine learning meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Adaboost determines the weight w of sample according to whether it is correctly classified or not. It makes the wrong predictive samples more important, and handles it in next round.

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

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

and $z_m$ is normalization term, makes $w_m(i)$ become probability distributions.

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

So in the next round, $\frac{w_{wrong}(i)}{w_{right}(i)} = e^{2\alpha_m} = \frac{1-\epsilon_m}{\epsilon_m}$ and $\epsilon_m < 0.5$, wrong sample would be more important.

Every iteration generates a new base learner and its impotance score.

- Base learner : $h_m(x): x \mapsto \{-1,1\}$
- Error rate : $\epsilon_m = p(h_m(x_i) \neq y_i) = \sum_{i=1}^{n} w_m(i) \mathbb{I}(h_m(x_i) \neq y_i)$

$\epsilon_m < 0.5$, or the performance of Adaboost is weaker than random classification.

To makes the base learner with lower $\epsilon_m$ more important.

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

Final learner is $H(x) = sign(\sum_{i=1}^{M} \alpha_m h_m(x))$. Note that $h_m(x) = sign(w^T x)$ is a nonlinear function, so the Adaboost can deal with nonlinear problem.

**Adaboost:**

**Input:** D = {($x_1$; $y_1$),..., ($x_n$; $y_n$)}, **where** $x_i \in X, y_i \in \{-1,1\}$

**Initialize:** Sample distribution $w_m$

**Base learner:** $\mathcal{L}$

1 $w_1(i) = \frac{1}{n}$

2 **for** m=1,2,...,M **do**

3 $h_m(x) = \mathcal{L}(D, w_m)$

4 $\epsilon_m = \sum_{i=1}^{n} w_m(i) \mathbb{I}(h_m(x_i) \neq y_i)$

5 **if** $\epsilon_m$> 0.5 **then**

6 **break**

7 **end**

8 $\alpha_m = \frac{1}{2} log \frac{1-\epsilon_m}{\epsilon_m}$

9 $w_{m+1}(i) = \frac{w_m(i)}{z_m} e^{-\alpha_m y_i h_m(x_i)}$, **where** *i*= 1, 2, …, n **and** $z_m = \sum_{i=1}^{n} w_m(i) e^{-\alpha_m y_i h_m(x_i)}$

10 **end**

**Output:** $H(x) = \sum_{i=1}^{M} \alpha_m h_m(x)$

**CART decision tree:**

Specifically, in the classification problem, if there are K categories and the probability of the K category is pk, the expression of the Gini coefficient is:

$$Gini(p) = \sum_{k-1}^{K} p_k(1-p_k) = 1 - \sum_{k-1}^{K} p_k^2$$

For the two class of classification problems, the calculation is more simple, if the probability of the first sample output is p, then the expression of the Gene coefficient is:

$$Gini(p) = 2p(1-p)$$

For a given sample D, if there are K categories and the number of category K is Ck, the expression of the Gini coefficient of the sample D is:

$$Gini(D) = 1 - \sum_{k-1}^{K}(\frac{|C_k|}{|D|})^2$$

In particular, for the sample D, if the D is divided into two parts of D1 and D2 according to a certain value a of the characteristic A, the expression of the Gini coefficient of D is in the condition of the characteristic A, the expression of the Gini coefficient is:

$$Gini(D,A) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

The input of the algorithm is the threshold of the training set D, the Gini coefficient and the number of the sample.

The output is the decision tree T.

Our algorithm starts from the root node, and uses the training set to build the CART tree recursively.

1) For the data set of the current node is D, if the number of samples is less than the threshold or no feature, then the decision subtree is returned, and the current node stops recursion.

2) The Gini coefficient of the sample set D is calculated. If the Gini coefficient is less than the threshold, then the decision tree subtree is returned, and the current node stops recursion.

3) Calculate the Gene coefficients of the data of each node of the current node's characteristic values to the data set D. There are second sections for the calculation of discrete values and continuous values and the calculation of Gene coefficients. The processing method of the missing value is the same as that described in the C4.5 algorithm in the last one.

4) in the Gini coefficient of the data set D, each characteristic value of the calculated features selects the feature A with the smallest Gini coefficient and the corresponding eigenvalue a. According to this optimal feature and the best eigenvalue, the data set is partitioned into two parts, D1 and D2, and the left and right nodes of the current node are established at the same time. The data set of the node is D, D1, and the data set of the right node of D is D2..

5) 1-4 step is called for the recursion of the left and right subnodes to generate the decision tree.

## III. EXPERIMENT

A. Experiment dataset

This experiment provides 1000 pictures, of which 500 are human face RGB images, the other 500 is a non-face RGB images

B. Experimental step

（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.

（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.)

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

（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 w, each training sample is given the same weight.

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

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 .

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

Parameters :

| | |
|---|---|
| Image size | 24*24 |
| Max depth of weak classifier | 4 |
| Delta of weak classifier | 1e-6 |
| Training size | 800 |

Report.txt:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1.0 | 0.98 | 0.95 | 0.96 | 99 |
| 1.0 | 0.95 | 0.98 | 0.97 | 101 |
| avg / total | 0.97 | 0.96 | 0.96 | 200 |

## IV. CONCLUSION

In this experiment, to achieve the purpose of face classification and testing, we implement the Adaboost based on the decisionTree Classifier. Now adaboost algorithm is a very good algorithm in all the methods of face detection algorithm, fast, good real-time, and the high rate of detection.