华南理工大学

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Haipeng Deng

Supervisor:
Mingkui Tan

Student ID:
201730686193

Grade:
Undergraduate

November 7, 2019

Face Detection Based on AdaBoost Algorithm

***Abstract*—This report introduces my work in Lab3 : Face Detection Based on AdaBoost Algorithm in which I use some picture processing methods to make the raw material (training pictures data) suitable for fitting and first try Adaboost to implement a face classification model.**

I. Introduction

THE feature of the boosting algorithm is that there are strong dependencies between individual base learners and those learners are generated by the cascaded method. Adaboost (adaptive boosting) is one of the most famous boosting algorithms in ensemble learning. In this lab, I was required to implement a face classifying model using Adaboost. It was interesting to get familiar with the basic strategy of face detection and combine the theory I learned in class with this actual project practically for the first time. Besides, I was looking forward to experiencing the complete process of machine learning through this experiment.

II. Methods and Theory

In the practical classification problem, we will encounter some situation that one model can not separate the data of different labels well. To address this problem, a method called boosting is created.

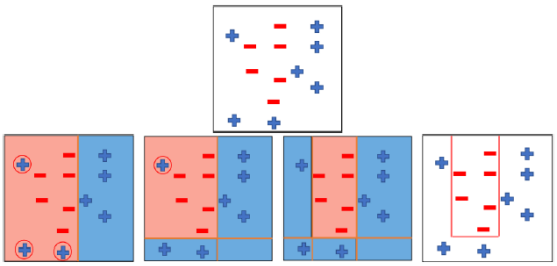

Fig. 1. Example using boosting algorithm

The Adaboost (adaptive boosting) model can be seen as a set of other models. In the training process, the boosting algorithm requires a base learner which trains on the original training set. Then it updates the weights of every single training datum according to the performance of this base learner i.e. enlarge the weight of those which is mistakenly classified to force the next learner focus it more and make the weight of correctly classified data smaller. In this lab it is implemented by the formula below:

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

$$z_m = \sum_{i=1}^{n} w_m(i) e^{-\alpha_m y_i h_m(x_i)} \tag{2}$$

In order to evaluate the performance of every base learner we then introduce a term called error rate, which is computed as below:

$$\epsilon_m = p(h_m(x_i) \neq y_i) + \sum_{i=1}^{n} w_m(i) I(h_m(x_i) \neq y_i) \tag{3}$$

In those base learner, we require that the performance of every one of them has the property $\epsilon_m < 0.5$.
So there comes another question, how should we determine the importance or weight for each base learner? To make the base learner with lower ϵ_m more important, an formula below gives us a solution:

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

By using the α_m(s) computed above as the weights for base learners, we can finally assemble a final learner

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

Note: $h_m(x) = sign(w^T x)$ is a nonlinear function, so the Adaboost can deal with nonlinear problem

III. Experiments

A. Dataset

The experiment document downloaded at Github provides 1000 pictures, of which 500 human face RGB images, stored in datasets/original/face; and 500 non-face RGB images, stored in datasets/original/nonface. In this lab, I do picture process and build my Adaboost model on the given dataset.

B. Implementation

Face Classification

To start with, I load data set data and convert them into grayscale images with size of 24 * 24 and set the label of face images as $+1$ while the label of nonface images are -1.

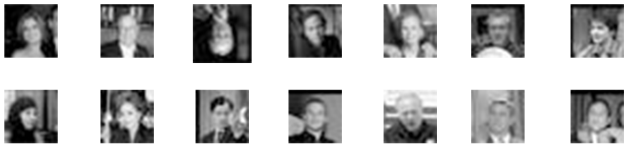

Fig. 2. Example of grey pictures

Secondly I use some already picture processing data code to extract NPD features of all pictures and store them in a

numpy array. The following step I do is shuffling the data and splitting the dataset into a training set of 900 images and a validation set of 100 images. Then I implement all functions in AdaboostClassifier class which constructs an Adaboost model. In this part, I use 'DecisionTreeClassifier' in sklearn.tree library as my base learner class. Three base learners is assembled in my final classifier in total and their weight α is calculated using the formula mentioned above. Finally, the performance of my Adaboost model is shown below:

TABLE I
CLASSIFIER REPORT

	precision	recall	f1-score	support
-1.0	0.96	0.96	0.96	51
1.0	0.96	0.96	0.96	49
accuracy			0.96	100
macro avg	0.96	0.96	0.96	100
weighted avg	0.96	0.96	0.96	100

Face Detection

In this part, I run $facedetection.py$ and experience the OpenCV's built-in method of face detection using Haar Feature-based Cascade Classifiers.

Fig. 3. Result of OpenCV Face Detection

IV. CONCLUSION

To summarize, I feel this lab very inspiring and beneficial for me. Instead of reading and deriving the theory and formulas in books and slices, it offers us a practical way to practise machine learning.