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:
Dongming Sheng, Haokun Li and Yu Guo

Supervisor:
Mingkui Tan

Student ID:
201530612699 , 201530611883 and 201530611500

Grade:
Undergraduate

December 20, 2017

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—In this experiment, we conduct Adaboost to detect whether there is a human face in a picture. We use 1000 pictures of which 500 are human face images to train and validate the Adaboost model. The result shows that Adaboost performs well in such classification problem.**

I. Introduction

HUMAN face detection is a widely-used technique in daily life. This area is related to computer vision and machine learning. In this experiment, we simplify this problem to detect whether a picture has a human face in it. It comes down to a classification problem. In order to solve it, we use Adaboost and try to validate its accuracy. We choose python to implement algorithm since it has convenient libraries to use for featuring engineering and modeling. Our hope is the best accuracy can reach over 80%.

II. Methods and Theory

A. AdaBoost Algorithm

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other weak learners 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. 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.

Suppose we have a data set $(x_1, y_1), \cdots, (x_m, y_m)$ where each item x_i has an associated class $y_i \in -1, 1$, and a set of weak classifiers$G_1, \cdots, G_T$each of which outputs a classification $h_t(x_i) \in -1, 1$for each item. After the $T-1$-th iteration our boosted classifier is a linear combination of the weak classifiers of the form:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{1}$$

The weighted error function is:

$$\epsilon = \sum_{i=1}^{N} \omega_i I(h(x_i) \neq f(x_i)) \tag{2}$$

The weight of weak classification α can be defined as:

$$\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon}) \tag{3}$$

We use normalization factor to normalize ω:

$$Z = \sum_{i=1}^{N} \omega(x) e^{-\alpha f(x) h(x)} \tag{4}$$

The procedure of AbaBoost Algorithm is shown here:

Algorithm 1 AdaBoost

Require: Dataset: $D = (x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$;
Basis learning algorithm: G;
Number of weak classifier: T.

Ensure: $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

1: $\omega_1(x) = 1/m$.
2: **for** $t = 1, 2, \cdots, T$ **do**
3: $h_t = G(D, \omega_t)$;
4: $\epsilon_t = P_{x \sim \omega_t}(h_t(x) \neq f(x))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$;
7: $Z_t = \sum_{i=1}^{N} \omega_t(x) e^{-\alpha_t f(x) h_t(x)}$;
8: $\omega_{t+1}(x) = \frac{\omega_t(x)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if } h_t(x) \doteq f(x) \\ e^{\alpha_t}, & \text{if } h_t(x) \neq f(x) \end{cases}$
$= \frac{w_t(x) exp(-\alpha_t f(x) h_t(x))}{Z_t}$
9: **end for**

III. Experiments

A. Dataset

This experiment provides 1000 pictures, of which 500 are human face RGB images and the rests are non-face RGB images. In the experiment, we label the face images as 1 and the non-face as -1 respectively.

B. Implementation

We implement the experiment in the following order:

1) Main Frame:

- Load data. The RGB images are supposed to be converted into gray-scaled images with a size of 24 * 24.
- Extract NPD features. Extract features using the *NPDFeature* class in *feature.py*.
- Divide data set into training set and validation set.
- Fill all functions in *AdaboostClassifier* based on the interfaces provided in *ensemble.py*.
- Predict and verify the accuracy on validation set using the function provided in *AdaboostClassifier* and use *classification_report* in *sklearn.metrics* to output your predicted result to *report.txt*.
- Organize the experiment results and finish the report.

2) Fit Function in AdaboostClassifier:

- Initialize training set weights. Each training sample is given the same weight.
- Train a base classifier(*sklearn.tree.DecisionTreeClassifier*). Remember to pass the sample weights as parameters when training.
- Calculate the classification error rate of the base classifier on the training set.
- Calculate the weight of the current classifier according to its classification error rate.
- Update training sample weights.
- Repeat the steps above. Use *n_weakers_limit* parameter to control the number of iterations.

In the experiment, we use hold-out method and spare 25% of the data as validation data when evaluating our model. We choose $threshold = 0$ to judge when predicting labels. At the end of each iteration, we use our validation data to evaluate the current boosting result so as to avoid over-fitting.

We choose *DecisionTreeClassifier* in *sklearn.tree* as our base classifier. The detailed parameters are listed in Table I. When selecting base classifiers, we remove base classifiers with error rate ϵ_m greater than 0.5. Meanwhile, if the training error rate reaches 0, we will end the iteration.

TABLE I
HYPER-PARAMETER SELECTED FOR DECISIONTREECLASSIFIER

max_depth	3
criterion	gini
splitter	best
min_samples_split	2
min_samples_leaf	1
max_features	None

As for *AdaBoostClassifier*, we set the maximum iteration *n_weakers_limit* to be 10 and plot a curve(Fig 1) according to the validation accuracies after each parameter update.

Although AdaBoost is said not to be prone to over-fitting due to its weak classifiers and the boosting method, in our experiment, the best validation score is reached at the second iteration. What's more, during the following iterations, the results are worse than just using a single base classifier to predict, which the latter scores about 86%. As a conclusion, AdaBoost still has the problem of over-fitting after all.

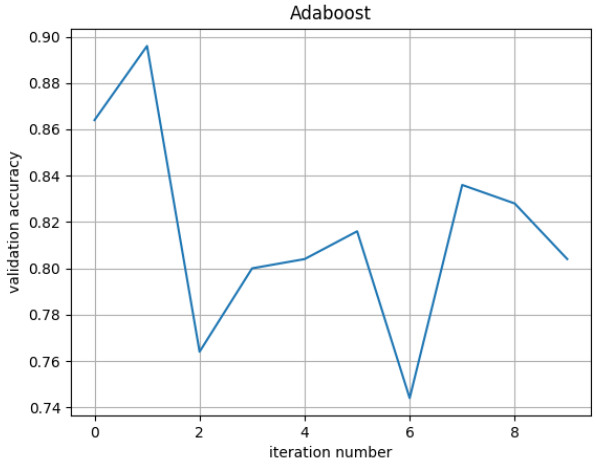

Fig. 1. The accuracy curve of Adaboost classifier.

IV. CONCLUSION

In this experiment, we implement Adaboost in python and try to solve the face classification problem. We combine the machine learning theory with the actual project, understand Adaboost further and get familiar with the basic method of face detection. The result shows that Adaboost do well in this classification problem. However, it still has the risk of over-fitting.