The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Hu Lin, Jiaqi Zhong and
Fengchao Wang

Supervisor:
Qingyao Wu

Student ID：
201721045497,201720144948
and 201720145013

Grade:
Graduate

December 19, 2017

Face Classification Based on AdaBoost Algorithm

Abstract—This experiment is about face classification based on AdaBoost algorithm, which aims to classify the images into two categories, one is the face and the other is non-face. This method has been successful conducted on a small face dataset. Furthermore, we train serval AdaBoost model by fine-tuning its parameters to achieve the better model with higher accuracy.

I. INTRODUCTION

In this experiment, face classification based on AdaBoost algorithm has been done on a small dataset. The dataset contains 1000 images, of which 500 are human face RGB images and the other 500 is non-face RGB images. What can be shown in this experiment is that giving an image, the model with AdaBoost algorithm trained can recognize whether it is a face or not. And in our experiment, we try serval times to tune the two parameters in order to get better results. In the last, we all understand Adaboost further and learn to use Adaboost to solve the face classification problem and combine the theory with the actual project.

II. METHODS AND THEORY

A. Background

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules. The AdaBoost algorithm of Freund and Schapire, also named as discrete AdaBoost Algorithm, was the first practical boosting algorithm, and remains one of the most widely used and studied, with applications in numerous fields. In a word, AdaBoost is an algorithm for constructing a "strong" classifier as linear combination. And it can be represented by the formula:

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Where $h_t(x)$ is called as "weak" or "basic" classifier, the $F(x)$ is called as "strong" classifier, and each is assigned weight α_t.

B. Algorithm

Pseudocode for AdaBoost is shown Figure 1. Here we are given m labeled training examples $(x_1, y_1), \ldots, (x_m, y_m)$ where x_i is in some domain X, and the labels $y_j \in \{-1, +1\}$. On each round $t = 1, \ldots, T$,a distribution D_t is computed as in the figure over the m training examples, and a given weak learner or weak learning algorithm is applied to find a weak hypothesis $h_t: X \to \{-1, +1\}$, where the aim of the weak learner is to find a weak hypothesis with low weighted error ϵ_t relative to D_t. The *final or combined* hypothesis H computes the sign of a weighted combination of weak hypotheses.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Fig. 1: The AdaBoost algorithm

C. Evaluation metrics

In the experiment, we utilize three evaluation metrics to measure the binary classification model performance, including precision, recall and F1-measure. At first, confusion matrix should be introduced. As is shown in Figure 2, TP represents the number of positive class predicted to be positive. TN means the number of negative class predicted as the negative. FP is the number of negative class predicted to be positive, which means type I error. And FN is the number of positive class predicted to be negative, which means type II error.

	Positive	Negative
True	True Positive (TP)	True Negative (TN)
False	False Positive (FP)	False Negative (FN)

Fig. 2: The confusion matrix

(1) Precision

Precision is abbreviated as P and can be written as the formula based on the confusion matrix.

$$P = \frac{TP}{TP+FP}$$

(2) Recall

Recall is abbreviated as R and can be written as the formula based on the confusion matrix.

$$R = \frac{TP}{TP+FN}$$

(3) F1-measure

F1-measure is abbreviated as F1 and can be written as the formula based on the confusion matrix.

$$F_1 = \frac{2PR}{P+R} = \frac{2TP}{2TP+FP+FN}$$

Especially, higher values of precision and recall mean the better model performance, while higher values of F1-measure mean the method more effective.

III. EXPERIMENT

A. Dataset

The experiment uses the small dataset that contains 1000 images, of which 500 are human face RGB images and the other 500 is non-face RGB images. The dataset is divided into as training set with 50 percent and validation set with 50 percent. And it is downloaded from the Internet.

B. Implementation

The whole experiment is implemented in accordance with the experimental steps.

(a)Download data set and divide it into training set and validation set, which accounting for 80% and 20% respectively.

(b)Read data set data. The images are supposed to converted into a size of 24 * 24 grayscale. Then set positive samples label into 1 and negative samples into 1.

(c)Processing data set data to extract NPD features. Extract features using the NPD Feature class in feature.py.

(d)Write all *AdaboostClassifier* functions based on the reserved interface in *ensemble.py*:

(i) Initialize training set weights, each training sample is given the same weight.

(ii)Training a base classifier, which can be sklearn.tree library *DecisionTreeClassifier* .

(iii)Calculate the classification error rate of the base classifier on the training set.

(iv)Calculate the parameter according to the classification error rate.

(v)Update training set weights.

(vi)Repeat steps (ii)-(vi) above for iteration, the number of iterations is based on the number of classifiers.

(e)Predict and verify the accuracy on the validation set using the method in *AdaboostClassifier*, and writes predicted result to report.txt.

(f)Organize the experiment results and complete the lab report.

The experiment result is shown as follow.

C. Results and analysis

(1) Face classification result

For this experiment, we try to tune parameters to obtain the better result of model. The main two parameters influencing the model is the maximum depth of decision tree (depth) and the number of basic classifiers(N). The parameters setting and its result in the Table 1. There are 12 models in Table 1 and each model is validated once. The precision, recall and f1-score is reflected the average value among validation dataset.

(2) Analysis of face classification result

Through fine-tuning the two parameters, we find some interesting things. Firstly, increasing the number of basic classifiers with fixed depth of decision tree, the value of precision, recall and F1-measure basically show a rising trend. It reflects the performance of model more precise and effective. Secondly, deeper depth of decision tree can also get better result generally. In particular, model 5 shows the best result when P is 2 and N is 10, whose value of three evaluation metrics equals 0.95. At last, as the depth of decision tree become 4, the whole result of model 10-12 is inferior to these models with the value 2 or 3 of depth. Therefore, we choose the two parameters of best result as the face classification model.

Table 1: model parameters and its result

model	depth	N	P	R	F1
1	1	5	0.83	0.81	0.81
2	1	10	0.82	0.81	0.80
3	1	20	0.84	0.83	0.83
4	2	5	0.92	0.92	0.91
5	**2**	**10**	**0.95**	**0.95**	**0.95**
6	2	20	0.94	0.94	0.94
7	3	5	0.91	0.91	0.90
8	3	10	0.94	0.94	0.94
9	3	20	0.94	0.94	0.94
10	4	5	0.86	0.86	0.86
11	4	10	0.92	0.92	0.91
12	4	20	0.93	0.93	0.93

IV. CONCLUSION

In the experiment, we have achieved the face classification based on AdaBoost algorithm. Through fine-tuning the two parameters, we get 12 results of models and analyze it in detail. In term of this experiment, model 5 in Table 1 shows the best result than other models. And it is very important that the parameters should be tuned reasonably based on dataset, because too many classifiers can easily lead to over-fitting while too few classifiers do not work well. In the future, we can try to improve the algorithm to get better result.