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:
Junwen Mo, Huantong Li and Zhiwei Lin

Supervisor:
Mingkui Tan

Student ID:
201530612545, 201530611944 and
201530612316

Grade:
Undergraduate

December 22, 2017

Face Classification Based on AdaBoost Algorithm

***Abstract*—This report is about the experiment of Face Classification Based on AdaBoost Algorithm. In this report, by introducing the AdaBoost algorithm, the face classification problem is converted to a classification problem using ensemble method.**

I. Introduction

THIS experiment is focus on the face classification problem. The motivations of experiment as below: 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

A. AdaBoost

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gdel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('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. 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.
Here is the algorithm below:

Algorithm 1 Adaboost

Input: training data set D = $\{(x_1, y_1),(x_2, y_2),...,(x_m, y_m)\}$
base classifier algorithm ξ
training times T
Output: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

$\mathcal{D}_1(x)$ = 1/m.
for t = 1,2,...,T **do**
  h_t = ξ(D,$\mathcal{D}_t$)
  ϵ_t= $P_{x\ D_t}(h_t(x) \neq f(x))$;
  if $\epsilon > 0.5$ **then break**
  α_t = $\frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
  $\mathcal{D}_{t+1}(x)$ = $\frac{\mathcal{D}_t(x)}{Z_t} \times exp(\alpha_t)$
  =$\frac{\mathcal{D}_t(x)exp(-\alpha_t f(x)h_t(x))}{Z_t}$
end for

B. Base Classifier

III. Experiments

A. Dataset

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

2.The dataset is included in the example repository. Divide it into training set and validation set.

B. Implementation

In this experiment, we use different maximum depth and different numbers of decision trees to implement the Adaboost algorithm. We will compare the accuracy of them. Within the 1000 samples, we use 33 percent of them as the testing set while the left as the training set.

1) Experiment Steps:

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 ω , 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 ω as a parameter).

4.3 Calculate the classification error rate ϵ of the base classifier on the training set.

4.4 Calculate the parameter α according to the classification error rate .

4.5 Update training set weights ω.

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).

2) Experiment Results and Analysis:

With different maximum depth and different numbers of decision trees, we can get the different accuracy as show in Table I and Fig. 1. We can see that the accuracy increase along with increase of the maximum depth of the decision tree in the beginning. However, it will reach a peak and then begin to decrease. So as the numbers of the decision tree. In our analysis, we think that it may be caused by the over-fitting because we found that the accuracy can reach 100 percent when we used the classifier to classify the training set. When the maximum depth increase, every classifier will try to fit every training samples. Also, when the number of classifiers increase, the whole classifier will try to fit all the samples.

TABLE I
ACCURACY

	Tree mount					
Tree Depth	1	2	3	4	5	6
2	0.842	0.833	0.891	0.885	0.927	0.933
3	0.860	0.824	0.912	0.915	0.936	0.954
4	0.833	0.839	0.912	0.918	0.939	0.939
5	0.848	0.879	0.942	0.945	0.897	0.927

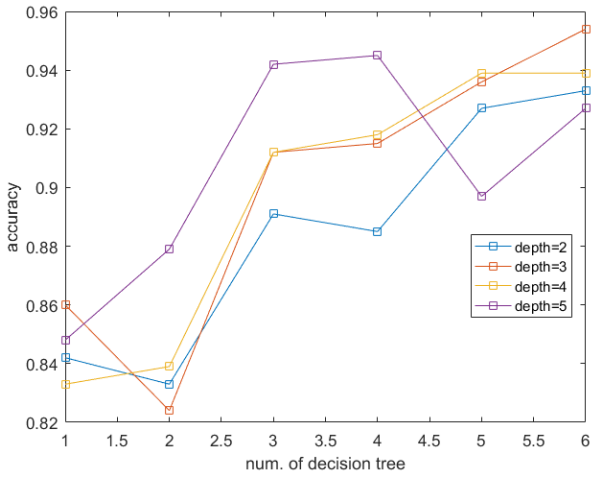

Fig. 1. The accuracy figure with different maximum depth and different numbers of decision trees.

IV. CONCLUSION

In this experiment, we try using the adaboost to implement a simple face detection. Adaboost is a algorithm about the combination of the weak classifiers. The reason why it does not use the strong classifiers is because it's very easy to overfit with the strong classifiers, which we can see from the result with the depth of the trees increasing.