The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:

Zhenyi Tang

Student ID:

201630503026

Supervisor:

Qingyao Wu

Grade:

Undergraduate

November 17, 2018

Face Detection Based on AdaBoost Algorithm

Abstract—The motivation of this experiment is to understand Adaboost further. Also it will help student get familiar with the basic method of face detection and Learn to use Adaboost to solve the face detection problem, and combine the theory with the actual project. Finally experience the complete process of machine learning

I. INTRODUCTION

The first part of this experiment will imply face detection based on Adaboost Algorithm. First extract features using the NPDFeature class in feature.py, then divide the dataset into training set and validation set, and write all AdaBoostClassifier functions based on the reserved interface in ensemble.py. Finally 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 classifier_report.txt.

the second part of this experiment will run the face_detection.py file, and experience the OpenCV's built-in method of face detection using Haar Feature-based Cascade Classifiers. The result will be save as detect_result.jpg.

II. METHODS AND THEORY

Lab3.1 will using the additive model as follow and try to minimize the exponential loss on training data. And the loss function is showing follow.

$$\mathrm{L}\big(\mathrm{y}, F_m(\mathrm{x})\big) = e^{-yF_m(x)} \quad (1.1)$$

Setting the derivation of this loss as zero will give:

$$\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m} \quad (1.2)$$

Recall the distribution weight of samples and based on the additive model.

$$F_m(x) = F_{m-1}(x) + a_m h_m(x) \quad (1.3)$$

Then we can find the final update equation.

$$\omega_{(m+1,i)} = \frac{\omega_{(m,i)} e^{-y_i a_m h_m(x_i)}}{z_m} \quad (1.4)$$

However, Adaboost have some limitations it treats samples from different classes equally, leading to sub-optimal solutions for imbalanced data

III. EXPERIMENT

A. Dataset

This experiment provides 1000 pictures, of which 500 are human face RGB images, stored in datasets /original /face; the other 500 are non-face RGB images, stored in datasets/original/nonface.

B. Implementation

In the lab3.1 we first load data set data. The images are supposed to converted into grayscale images with size of 24 * 24, the number. Then processing data set data to extract NPD features. Extract features using the NPDFeature class in feature.py. The data set is divided into training set and validation set. Then write all AdaBoostClassifier functions based on the reserved interface in ensemble.py. The following is the fit function in the AdaBoostClassifier class:

Initialize training set weights, each training sample is given th e same weight. Training a base classifier, which is sklearn.tree library DecisionTreeClassifier. Calculate the classification error rate of the base classifier on the training set. Calculate the parameter according to the classification error rate. Update training set weights. Repeat above for iteration, the number of iterations is based on the number of classifiers.

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 classifier_report.txt . The following table shows the parameter value.

TABLE I. The parameter list of lab3.1

Parameter name	value
sample_num	500
weak_clf_num	6
DecisionTreeClf(max_depth)	5
test_size	0.2
learning_rate	1

TABLE II. The classification_report of lab3.1

	precision	recall	f1-score	support
face	0.91	0.95	0.92	91
nonface	0.95	0.92	0.93	109
avg / total	0.93	0.93	0.93	200

Fig. 1. Face detection result of lab3.2

IV. CONCLUSION

From this experiment I learn that Adaboost 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.

Also every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.