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:
Junteng Pang Zhaohui Huang Lilong Huang

Supervisor:
Qingyao Wu

Student ID：201720145174 201720145112
201720144962

Grade:
Graduate

December 22, 2017

# Face Classification Based on AdaBoost Algorithm

Abstract—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.

## I. Introduction

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Gödel 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.

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.

This experiment aims at understanding Adaboost further, getting familiar with the basic method of face detection, learning to use Adaboost to solve the face classification problem, combining the theory with the actual project and experiencing the complete process of machine learning

## II. Methods and Theory

Problems in machine learning often suffer from the curse of dimensionality — each sample may consist of a huge number of potential features (for instance, there can be 162,336 Haar features, as used by the Viola–Jones object detection framework, in a 24×24 pixel image window), and evaluating every feature can reduce not only the speed of classifier training and execution, but in fact reduce predictive power, per the Hughes Effect. Unlike neural networks and SVMs, the AdaBoost training process selects only those features known to improve the predictive power of the model, reducing dimensionality and potentially improving execution time as irrelevant features need not be computed.

**Training**

AdaBoost refers to a particular method of training a boosted classifier. A boost classifier is a classifier in the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

where each $f_t$ is a weak learner that takes an object x as input and returns a value indicating the class of the object. For example, in the two class problem, the sign of the weak learner output identifies the predicted object class and the absolute value gives the confidence in that classification. Similarly, the Tth classifier is positive if the sample is in the positive class and negative otherwise.

Each weak learner produces an output hypothesis, $h_{(x_i)}$, for each sample in the training set. At each iteration t, a weak learner is selected and assigned a coefficient $\propto_t$ such that the sum training error $E_t$ of the resulting t-stage boost classifier is minimized.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)]$$

Here $F_{t-1}(x)$ is the boosted classifier that has been built up to the previous stage of training, E(F) is some error function and $f_t(x) = \propto_t h_{(x)}$is the weak learner that is being considered for addition to the final classifier.

**Weighting**

At each iteration of the training process, a weight {\displaystyle w_{t}} w_{t} is assigned to each sample in the training set equal to the current error E($F_{t-1}(x_i)$) on that sample. These weights can be used to inform the training of the weak learner, for instance, decision trees can be grown that favor splitting sets of samples with high weights.

## III. Experiment

a. 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.

b. 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.)

c. The data set is divisded into training set and calidation set, this experiment does not divide the test set.

d. Write all AdaboostClassifier functions based on the reserved interface in ensemble.py. The following is the guide of fit function in the AdaboostClassifier class:

1. Initialize training set weights, each training sample is given the same weight.

2. Training a base classifier, which can be sklearn.tree library DecisionTreeClassifier (note that the training time you need to pass the weight as a parameter).

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

4. Calculate the parameter $\propto$ according to the classification error rate.

5. Update training set weight.

6. Repeat steps 4.2-4.6 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 report.txt.

e. Organize the experiment results and complete the lab report (the lab report template will be included in the example repository).

**Key Code**

fit function in ensemble.py file

```
num_sample = X.shape[0]
sample_W = np.zeros((num_sample,), dtype=np.float32)
sample_W.fill(1.0 / num_sample)
for i in range(self.max_number_classifier):
    clf = tree.DecisionTreeClassifier(max_depth=4)
    clf = clf.fit(X, y, sample_weight=sample_W)
    y_predict = clf.predict(X)
    error = np.sum(sample_W[np.where(y_predict != y)])
    print('Error: {}'.format(error))
    self.alpha[i] = np.log((1 - error) / error) / 2.0
    sample_W = sample_W * np.exp(-self.alpha[i] * y *
y_predict) / np.sum(
        sample_W * np.exp(-self.alpha[i] * y * y_predict))
     self.weaker_classifier.append(clf)
```

in train.py file

```
# Train the model
adaboost_classifier.fit(X_train, y_train)
y_predict = adaboost_classifier.predict(X_validation)
target_names = ['non_face', 'face']
accuracy = np.mean(y_predict == y_validation)
```

**Result**

CLASSIFICATION RESULTS OF ADABOOST

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| non_face | 0.84 | 0.90 | 0.87 | 211 |
| face | 0.90 | 0.85 | 0.87 | 229 |
| Avg/total | 0.87 | 0.87 | 0.87 | 440 |

## IV. CONCLUSION

Through this experiment we have further understanding in the Adaptive Boosting algorithm. We learned how to train the model and use the model recognize the human face well. We found that 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.