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:
Zhishang Zhou, Fankai Xu, Li Kang

Supervisor:
Mingkui Tan and Qingyao Wu

Student ID:
201720144931, 201721046012,
201721046005

Grade:
Graduate

December 22, 2017

Face Classification Based on AdaBoost Algorithm

***Abstract*—In this experiment report, I will show you how AdaBoost algorithm works in face classification problem. Face classification is a typical classification problem, in which we should distinguish whether the current photo the system seeing is a face or not. Although we can use outstanding single model such as SVM, Decision Tree to solve the problem, we can use an ensemble method, combining a class of single model to get a more poweful model. Adaboost is one famous ensemble method, which we will illustrate below. In part 1, I will briefly talk about whole procedure we implemented. In part2, I will show Adaboost algorithm in detail. Finally, in part 3, I will show you complete experimental setup and experimental results to comfirm you the power of Adaboost algorithm.**

I. Introduction

IN the field of machine learning, the goal of statistical classification is to use an objects characteristics to identify which class (or group) it belongs to. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics. An objects chaacteristics are also known as feature values and are typically presented to the machine in a vector called a feature vector. Such classifiers work well for practical problems such as document classification, and more generally for problems with many variables (features), reaching accuracy levels comparable to non-linear classifiers while taking less time to train and use.

Face Classification is a typical classification problem, in which we should distinguish whether the current photo the system seeing is a face or not. By saying this, I mean the innate difference between face classification and other classification problem lie in the feature we used to classify. The whole classifying procedure is just as same as the fundamention classification process.

There had been a lot of method to extract the feature of human face. Some well-knwon features such as SIFT, HoG are widely used, and those features already achieved good generalization effect. In this experiment we will use Normalized Pixel Difference (NPD) feature for face classification problem. NPD feature is computed as the difference to sum ratio between two pixel values, inspired by the Weber Fraction in experimental psychology. The new feature is scale invariant, bounded, and is able to reconstruct the original image.

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.

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.

II. Methods and Theory

Face Classification: Given training data (x_i, y_i) for $i = 1...n$, with $x_i \in R_m$ and $y_i \in 1, 1$, learn a classfier $f(x)$ such that $y_i f(x_i) > 0$ for a correct classification. And $f(x)$ often takes the form just like linear combination of x_i $(i = 1...n)$.

As the previous experiments have shown, if the input feature vector to the classifier is a real vector $\boldsymbol{x}$, then the linear classification output score is $\boldsymbol{y} = f(\boldsymbol{w} \cdot \boldsymbol{x}) = \sum_j w_j x_j$, where $\boldsymbol{w}$ is a real vector of weights and f is a function that converts the dot product of the two vectors into the desired output. (In other words, $\boldsymbol{w}$ is a one-form or linear functional mapping $\boldsymbol{x}$ onto $\boldsymbol{R}$.) The weight vector $\boldsymbol{w}$ is learned from a set of labeled training samples. Often f is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex f might give the probability that an item belongs to a certain class.

For a two-class classification problem, one can visualize the operation of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side of the hyperplane are classified as "yes", while the others are classified as "no".

A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when $\boldsymbol{x}$ is sparse. Also, linear classifiers often work very well when the number of dimensions in $\boldsymbol{x}$ is large, as in document classification, where each element in $\boldsymbol{x}$ is typically the number of occurrences of a word in a document (see document-term matrix). In such cases, the classifier should be well-regularized.

Now, by replacing the features with face or non-face detected features, taking or not taking the form of linear classifivation model, we can simple get the problem of face classification. In this experiment, we will implement a new kind of classification method, called Dicision Tree Classification, which is different with Linear Classification. We will illustrate it below.

NPD Features: As we mentioned that by replacing the features with face or non-face detected features, taking or not taking the form of linear classifivation model, we can simple get the problem of face classification. It is straight that the different feature extraction methods will determine the quality of face classification system. In this experiment, we will use Normalized Pixel Difference (NPD) feature for face classification problem.

NPD feature is computed as the difference to sum ratio between two pixel values, inspired by the Weber Fraction in experimental psychology. The new feature is scale invariant, bounded, and is able to reconstruct the original image. NPD feature is computed as the difference to sum ratio between two pixel values, inspired by the Weber Fraction in experimental psychology.

$$f(x, y) = \frac{x - y}{x + y}$$

We choose the NPD extraction window size as d, so we can get $d * (d - 1)/2$ dimension feature vector for each window. The new feature is scale invariant, bounded, and is able to reconstruct the original image.

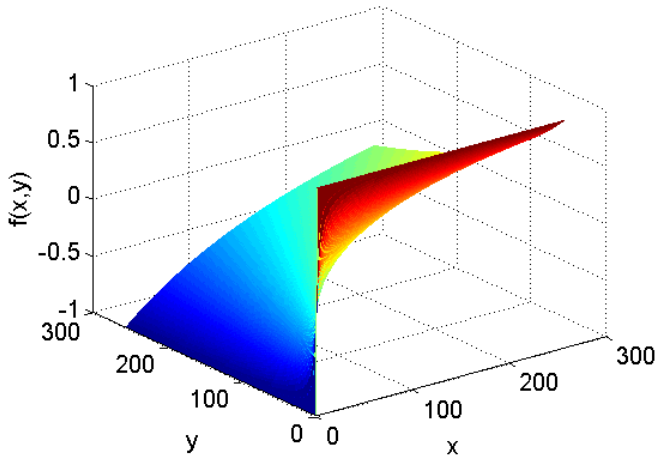

Fig. 1. A plot of the NPD function $f(x, y)$.

Decision Tree: A decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision. It is a type of supervised learning algorithm (with a predefined target variable) that is mostly used in classification problems and works for both categorical and continuous input and output variables. It is one of the most widely used and practical methods for inductive inference. (Inductive inference is the process of reaching a general conclusion from specific examples.) Decision trees learn and train themselves from given examples and predict for unseen circumstances.

Generally, we use ID3 algorithm to builds a decision tree from a fixed set of examples and the resulting tree is used to classify future samples. The basic idea is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node. Sounds simple , but which node should we select to build the correct and most precise decision tree? How would we decide that? Well, we have some measures that can help us in selecting the best choice!

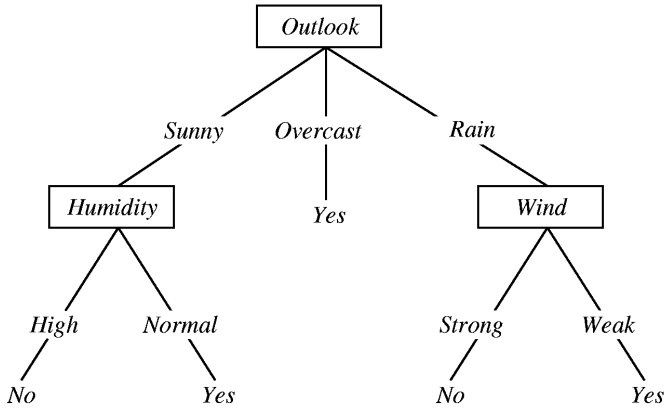

Fig. 2. A graphical representation of a sample decision tree.

- **Entropy**
  In information theory, entropy is a measure of the uncertainty about a source of messages. It gives us the degree of disorganization in our data. Given a collection S containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is:

  $$Entropy(S) = -p_+ \cdot ln\ p_+ - -p_- \cdot ln\ p_-,$$

  Here, p_+ and p_- are the proportion of positive and negative examples in S.
- **Information Gain**
  It measures the expected reduction in entropy. It decides which attribute goes into a decision node. To minimize the decision tree depth, the attribute with the most entropy reduction is the best choice!
  More precisely, the information gain $Gain(S, A)$ of an attribute A relative to a collection of examples S is defined as:

  $$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v),$$

  where S = Each value v of all possible values of attribute A, S_v = Subset of S for which attribute A has value v.

Adaboost: AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire, who won the 2003 Godel 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.[1][2] 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.

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:

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

Fig. 3. The boosting algorithm AdaBoost.

Pseudocode for AdaBoost is shown as before. Here we are given m labeled training examples $(x_1, y_1), ..., (x_m, y_m)$ where the x_is are in some domain X, and the labels $y_i \in \{1, +1\}$. On each round $t = 1, ..., T$, 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

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

This is equivalent to saying that H is computed as a weighted majority vote of the weak hypotheses h_t where each is assigned weight α_t. (In this chapter, we use the terms hypothesis and classifier interchangeably.)

III. Experiments

A. Dataset

In this work, we used 1000 pictures for training and validation. 500 face pictures and 500 non-face pictures respectively. The original pictures have resolution of 250*250, then we resieze these pictures into 25*25 pixels. We split 1000 pictures into two parts, one is for traning and another one is for validation, at a ratio of 8:2.

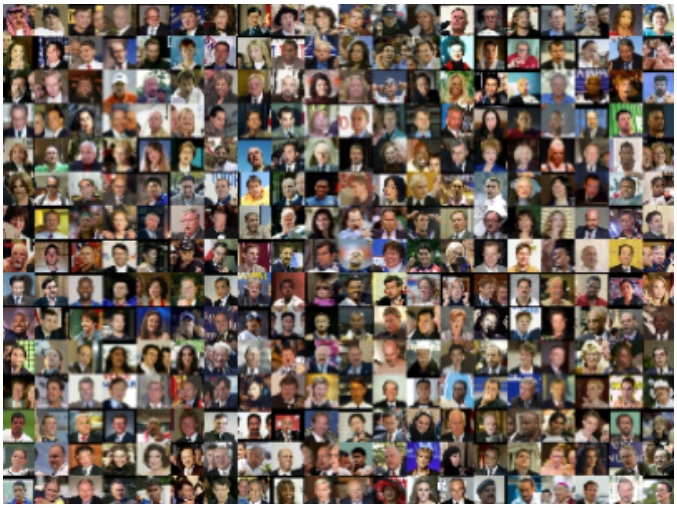

Fig. 4. Face pictures.

Fig. 5. Non-face pictures.

B. Implementation

We used AdaBoost as the model in the experiment, and Decisiontree as the weak classifier of AdaBoost. The experiment steps are as follow:

1. Read data set data, and converted images into a size of 24 * 24 grayscale.
2. Processing data set data to extract NPD features. Extract features using the NPDFeature class in feature.py. It took about 2 mins in my computer. We then dumped these features into local file with pickle. (The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.).
3. We then reloaded the dumped pickle file, and converted it into numpy.array in a shape of (1000,16500) . We then split the 1000 samples dataset randomly into two parts at a ration of 8:2, the large part is training set, the small part is validation set. The label is -1 and 1 for non-face pictures and face pictures respectively.
4. After the processing of datas was done, we started to

tranned the model.

4.1 We initialized training set weight w for training, the weight value for each sample is $\frac{1}{m}$ (m is the number of samples), and set the DecisionTree as the weak classifier. We set the number of weak classifier is 5, and set the max_depth as 5 for each Decisiontree.

4.2 After all initializations were done, we start training a base classifier, which is a DecisionTreeClassifier as we mentioned before.

4.3 Calculated the classification error rateof the base classifier on the training set.

4.4 Calculated the parameteraccording to the classification error rate .

4.5 Updated training set weights .

4.6 Repeated steps 4.2-4.6 above for iteration, the number of iterations is based on the number of classifiers.

5. After fit, we then did the prediction part. We predicted and verified the accuracy on the validation set using the method in AdaboostClassifier and use classification_report() of the sklearn.metrics library function writing predicted result to report.txt.We also visualized each trained Decisiontree and output to png file.

C. Results

	Precision	Recall	F1-score	Support
non-face	0.96	0.99	0.98	110
face	0.99	0.96	0.97	90
avg/total	0.98	0.97	0.97	200

Fig. 6. Experiment results.

As Fig. 6 shows the precision, recall and f1-score of the model. The results are quite well as you can see. For each trained Decisiontree model and Adaboost model, we had the accracy of 0.91, 0.88, 0.89, 0.89, 0.84, 0.975. The accuracy of Adaboost model is higher than any of these single Decisiontree model.

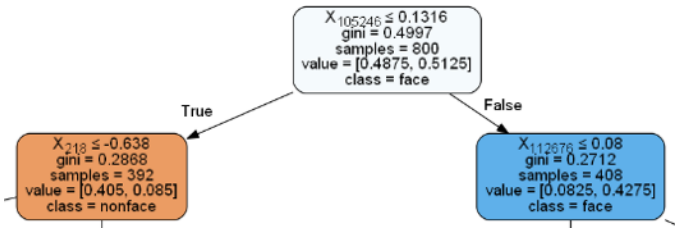

Fig. 7. Top 2 level of Decisiontree I.

In Fig. 7, you can see three nodes, each nodes tell you how it make decision to distinguish face and non-face. In First node, it choosed feature 105248 $\leq$ 0.1316 to make a decision.If feature 105248 $\leq$ 0.1316, then it should be belong face class.The value array shows proportion of non-face and face class. 0.4875 means the proportion of non-face class in 800 samples is 0.4875. 0.5125 means the proportion of non-face class in 800 samples is 0.5125. The gini value is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. In Fig. 8, you may ask why the sum of value array is not 1, because we change the weight for each sample.

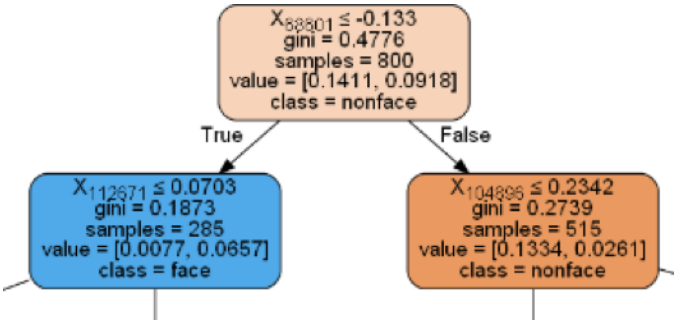

Fig. 8. Top 2 level of Decisiontree II.

IV. Conclusion

In this experiment, we use 5 Decisiontrees as the base weak classifier of Adaboost classifier. As a productive machine learning method, Adaboost has been widely used in may kinds of real appplication. Not only feature extracion but also feature selection, AdaBoost shows promising and satisfied performance. Only one Decisiontree may not perform well, but assemble Decisiontree into Adaboost model doing a great job.