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:
Xueyu Zhou
Shiyao Sun
Haoxuan Li

Supervisor:
Mingkui Tan

Student ID:
201930384264
201930382376
201930381263

Grade:
2019 Software Engineering Class 1 and 2

December 5, 2021

Face Detection Based on AdaBoost Algorithm

***Abstract*—This experiment requires an in-depth understanding of the principle of AdaBoost and familiarity with the basic methods of face detection Learn to use AdaBoost to solve face detection problems, and integrate theory with practical engineering, so as to experience the whole process of machine learning.**

I. Introduction

THE experiment provides 1000 photos for training face classification. We use AdaBoost method, in which the base classifier is trained by decision tree classifier. At the end of the experiment, we need to experience the results of OpenCV's own face detection method based on Haar feature and AdaBoost.

II. Methods and Theory

Before the AdaBoost algorithm starts, we must first convert the picture data into vector form, so in this section, we will first introduce the NPD features, and then explain the principle of AdaBoost algorithm in detail.

A. NPDfeature

The NPD feature is to detect the difference between two pixels (somewhat similar to Viola Johns), which is defined as the function $f(x, y)$. Where x and y are the pixel values of any two pixels. And specify $f(0, 0) = 0$.

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

B. Adaboost Algorithm

Suppose our training set sample is:

$$T = \{(x,y_1), (x_2, y_2), ...(x_m, y_m)\}$$

The output weight of the k-th weak learner of the training set is:

$$D(k) = (w_{k1}, w_{k2}, ...w_{km}); \;\; w_{1i} = \frac{1}{m}; \;\; i = 1, 2...m$$

Since multivariate classification is a generalization of binary classification, here we assume that we are a binary classification problem, and the output is - 1,1, then the weighted error rate of the k-th weak classifier $G_k(x)$ on the training set is:

$$e_k = P(G_k(x_i) \neq y_i) = \sum_{i=1}^{m} w_{ki} I(G_k(x_i) \neq y_i)$$

For the binary classification problem, the weight coefficient of the k-th weak classifier GK (x) is:

$$w_{k+1,i} = \frac{w_{ki}}{Z_K} exp(-\alpha_k y_i G_k(x_i))$$

Where Z_k is the normalization factor:

$$Z_k = \sum_{i=1}^{m} w_{ki} exp(-\alpha_k y_i G_k(x_i))$$

Therefore, it can be seen from the calculation formula of $w_{k+1,i}$ that if the ith sample is classified incorrectly, $y_i G_k(x_i) < 0$, resulting in the increase of the weight of the sample in the k + 1 weak classifier. If the classification is correct, the weight decreases in the k + 1 weak classifier.

AdaBoost classification adopts the weighted voting method, and the final strong classifier is:

$$f(x) = sign(\sum_{k=1}^{K} \alpha_k G_k(x))$$

Then we can give the pseudo code of AdaBoost according to the analysis just now:

Algorithm 1: Adaboost Algorithm

Input: Training set D=$\{(x,y_1), (x_2, y_2), ...(x_n, y_n)\}$

Output: A strong classifier

Initialize the weight vector:$w_1(i) = 1/n, i = 1, 2, ..., n$

for $t = 1, ..., T$ **do**

fit base learner$h_t(x) \in -1, +1$ Calculate the classification error rate e_t of $h_t(x)$

$$e_t = p(h_t(x) \neq y_i) = \sum_{i=1}^{N} w_t(i)(h_t) I(x_i \neq y_i)$$

where$I(.)$is the indicator function

$$I(X = x_i) = \begin{cases} 1, & h_t(x_i) \neq y_i \\ 0, & h_t(x_i) = y_i \end{cases}$$

;

Calculate the weight α_t of $h_t(x)$

$$\alpha_t = \frac{1}{2} \ln \frac{1 - e_t}{e_t}$$

Update the weights of each data point

$w_{t+1}(i) = \frac{w_t(i)}{z_t} e^{-\alpha_t y_i h_t(x_i)}$, Where$z_t = \sum_{i=1}^{n} w_t(i) e^{-\alpha_t y_i h_t(x_i)}$

end

Output the final hypothesis:$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x_i))$

III. Experiments

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.And we also need to divide it into training set and validation set.

B. Implementation

1) Initialize Adaboost classifier class:

```
def __init__(self, weak_classifier, n_weakers_limit)


    '''Initialize AdaBoostClassifier
    self.weak_classifier_list = []
    self.classifier_weight = []
    self.n_weakers_limit = n_weakers_limit
    self.Hx = []
    self.pred_list = []
```

2) Training function:

```
def fit(self, X, y):
    self.Hx = np.zeros((len(y), self.n_weakers_limit



    weight = np.full(shape=(X.shape[0]\
    ), fill_value=1/X.shape[0])
    print(weight.shape)
    for iters in range(self.n_weakers_limit):
        weak_classifier = DecisionTree\
        Classifier(
        criterion='entropy',
        splitter='random',
        max_features='log2',
        max_depth=10,
        max_leaf_nodes=10,
        min_samples_split=10,
        min_samples_leaf=3,
        class_weight='balanced'
        )
        weak_classifier.fit(X, y, \
        sample_weight=weight)
        hx = weak_classifier.predict(X)
        score = weak_classifier.score(X, y)
        print('score {}: '.format(iters+1), score)
        error = 1 - score
        alpha = 0.5 * np.log((1-error)/error)
        exp = np.exp(-alpha * np.multiply\
        (hx, y.flatten()))
        exp = np.array(exp).flatten()
        weight = np.multiply(weight, exp)
        zm = np.sum(weight)
        weight = weight / zm
```

The base classifier selects the decision tree as the initialization hx. The default weight is $\frac{1}{n}$ at the beginning. Next, the base classifier trains the training set and uses the prediction score to calculate the error rate and the next update of the weight (that is, the next weight with high score is small, and the next weight with low score is significant).

```
if(score>0.5):
            self.classifier_weight.append(alpha)
            self.weak_classifier_list.append\
            (weak_classifier)
            self.Hx[:,iters] = alpha*np.array(hx)
            self.pred_list.append(np.array(hx))
```

3) Prediction function (predicted value and score): predict function and predict_scores function uses the weak classifier filtered from the fit function to predict the Hx value. Finally, the sum of each column of Hx is the prediction result of AdaBoost. If the sum value is greater than 1, it will be regarded as 1 (with face), and if it is less than 1, it will be regarded as 0 (without face).The predict_scores function uses the average value obtained by comparing with the Y label column as the prediction score.

```
def predict_scores(self, X, y):
    Hx = np.zeros((len(y), self.n_\
    weakers_limit))
    for i in range(len(self.weak_\
    classifier_list)):
        hx = self.weak_classifier\
        _list[i].predict(X)
        Hx[:,i] = self.classifier\
        _weight[i]*np.array(hx)
    pred = np.sum(Hx, axis=1)
    pred = (pred>=1)
    accuracy = (pred==y.flatten())
    return np.mean(accuracy)

def predict(self, X, threshold=0):
    Hx = np.zeros((X.shape[0], self.\
    n_weakers_limit))
    for i in range(len(self.weak_\
    classifier_list)):
        hx = self.weak_classifier_list[i].\
        predict(X)
        Hx[:,i] = self.classifier_weight[i]*\
        np.array(hx)
    pred = np.sum(Hx, axis=1)
    pred[pred>=1] = 1
    pred[pred<1] = 0
    return pred
```

4) Processing image data: Read the picture data and classify whether there is a face. If there is a face, it is 1 and if there is no face, it is 0.

```
def dumpFeatures(path, label = 1):
    img_names = os.listdir(path)
    file = ''
    if label == 1:
        file = face_data_file
    else:
        file = nonface_data_file
    write = open(file, 'wb')
    feat = []
    for name in img_names:
        img_gray = cv.imread(path+name,\
        cv.IMREAD_GRAYSCALE)
        img_resize = cv.resize(img_gray, (24,24))
        npd = NPDFeature(img_resize)
        feat.append(npd.extract())
    pickle.dump(feat, write)

def loadFeatures(filename):
    read = open(filename, 'rb')
    data = pickle.load(file=read)
    return np.matrix(data)

def addLabel(data, label):
    data = np.matrix(data)
    return np.concatenate((np.full\
    (shape=(data.shape[0],1),fill_value=label)\
    , data), axis=1)
```

5) Executing main function:

```
if __name__ == "__main__":
    face_data = loadFeatures(face_data_file)
    nonface_data = loadFeatures(nonface_data_file)

    face_data = addLabel(face_data, 1)
    nonface_data = addLabel(nonface_data, 0)

    data = np.concatenate((face_data,\
    nonface_data), axis=0)

    fraction = 0.9
```

```
    X_train, X_valid, y_train, y_valid\
    = sk.model_selection.train_test_split\
    (data[:, 1:data.shape[1]], data[:, 0],\
    train_size=fraction,test_size=1 - fraction)

    X_train = np.matrix(X_train)
    X_valid = np.matrix(X_valid)
    y_train = np.matrix(y_train)
    y_valid = np.matrix(y_valid)

    weight = np.full(shape=(X_train.shape[0])\
    , fill_value=1 / X_train.shape[0])
    weak_classifier = DecisionTreeClassifier()

    classifier = AdaBoostClassifier\
    (weak_classifier, 10)
    classifier.fit(X_train, y_train)

    hx = classifier.predict(X_valid)

    file_write = open(report_file, 'wb')
    print(classification_report(y_valid, hx))

    reportContent = 'Accuracy = ' + '\n'
    reportContent += classification_report\
    (y_valid, hx)
    with open(report_file, 'w') as report:
        report.write(reportContent)
```

C. Result

1) Base Classifier Score: Finally, the detailed scores of each base classifier are output according to the training process.

```
score 1:  0.7811111111111111
score 2:  0.74
score 3:  0.7733333333333333
score 4:  0.7422222222222222
score 5:  0.7077777777777777
score 6:  0.7177777777777777
score 7:  0.7655555555555555
score 8:  0.61
score 9:  0.7422222222222222
score 10:  0.5188888888888888
0.8744444444444445, 0.81
```

Fig. 1. Base Classifier Score

2) Report: And generate the final report.

```
Accuracy =
              precision    recall  f1-score   support

         0.0       0.78      0.80      0.79        45
         1.0       0.83      0.82      0.83        55

    accuracy                           0.81       100
   macro avg       0.81      0.81      0.81       100
weighted avg       0.81      0.81      0.81       100
```

Fig. 2. Final Report

3) Haar Feature: This is the face detection from Haar feature.

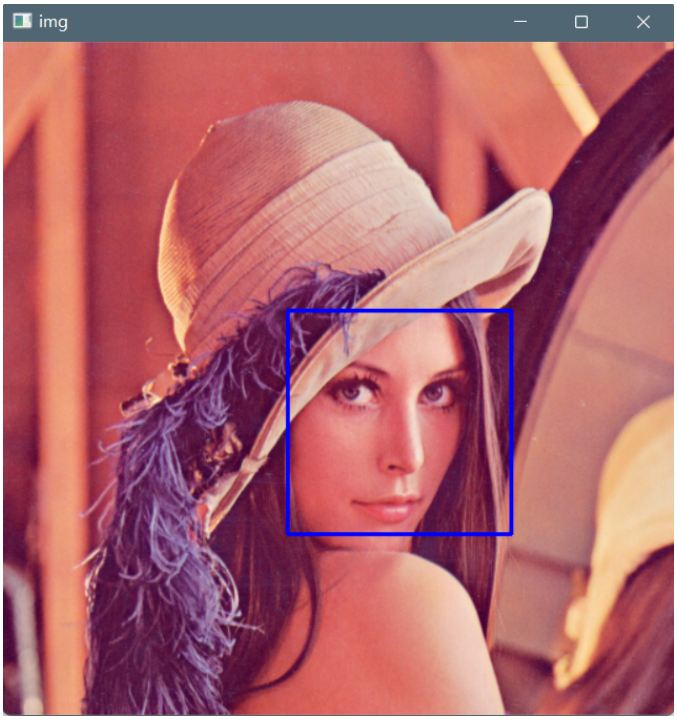

Fig. 3. Face Detection from Haar 1

4) Adaboost: This is the face detection from Adaboost.

Fig. 4. Face Detection from Adaboost 1

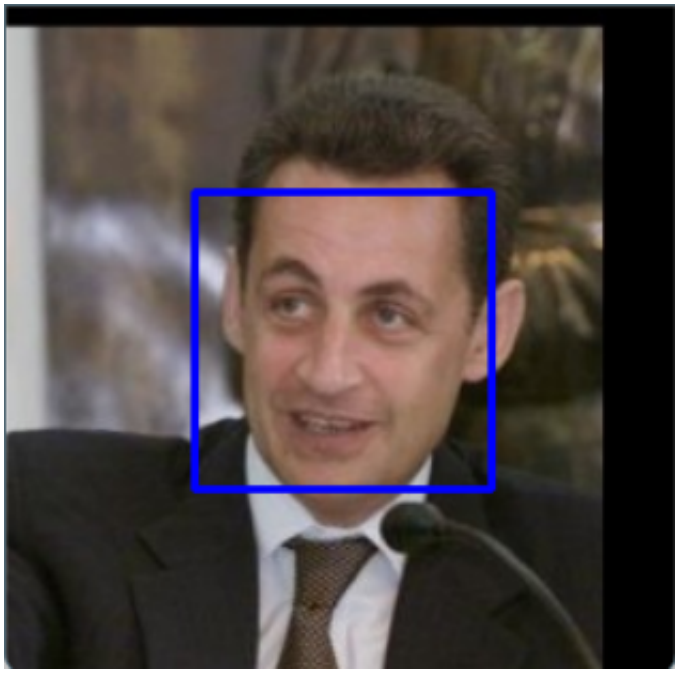

Fig. 5. Face Detection from Adaboost 2

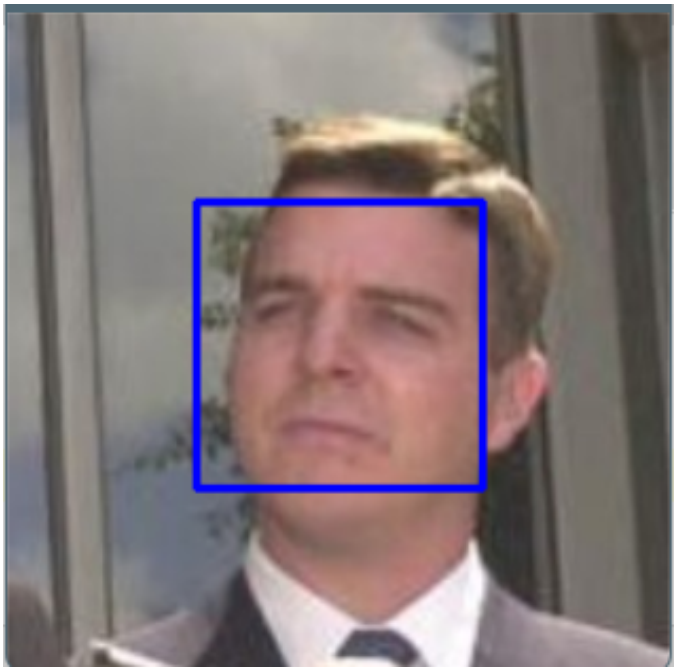

Fig. 6. Face Detection from Adaboost 3

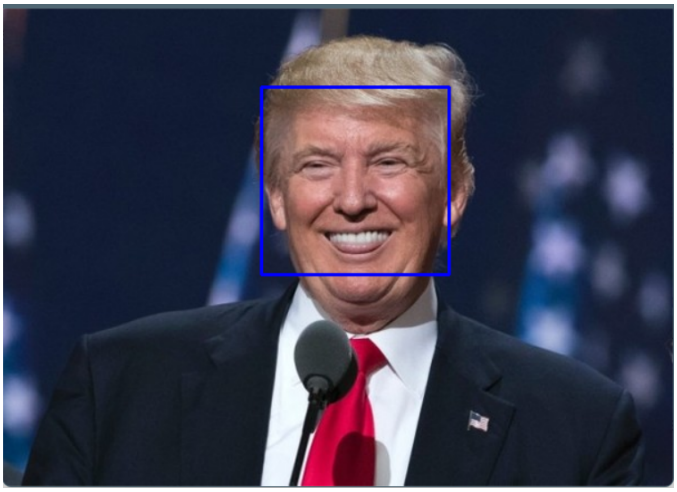

Fig. 7. Face Detection from Adaboost 4

IV. Conclusion

In this experiment, a simple AdaBoost algorithm is implemented, which has a deeper understanding of the implementation of the algorithm, and the idea of learning from a weak learner to a strong learner of boost has been integrated. The advantages of AdaBoost include

1) when used as a classifier, the classification accuracy is very high

2) Under the framework of AdaBoost, various regression classification models can be used to build weak learners, which is very flexible.

3) As a simple binary classifier, the structure is simple and the results are understandable.

4) It is not easy to fit

The main disadvantages of AdaBoost are:

1) It is sensitive to abnormal samples. Abnormal samples may obtain high weight in the iteration, which affects the prediction accuracy of the final strong learner.

To sum up, this experiment uses the characteristics of AdaBoost to realize face recognition, and can also achieve approximate face recognition for pictures on other networks, so this experiment is successful.

In addition, the experiment made me more familiar with the principle of AdaBoost, realized the whole process of face recognition and the fun of machine learning.