The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Ruiyong Li
Xin Shu
Xingyang Cai

Supervisor:
Qingyao Wu

Grade:
Undergraduate or Graduate

Student ID：
201530612033
201530612705
201530611074

December 9, 2017

Face Classification Based on AdaBoost Algorithm

Abstract—

I. Introduction

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

Motivation of Experiment:

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

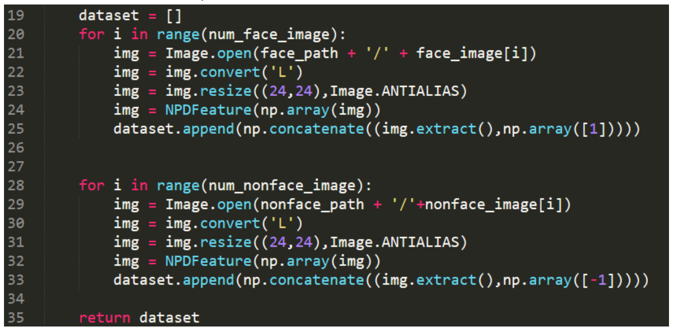
II. Methods and Theory

AdaBoost; Ensemble learning; Normalized Pixel Difference

III. Experiment

Experiment Step

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.

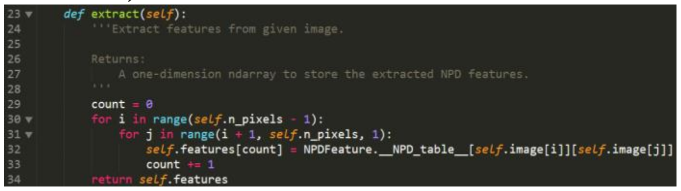


```
    dataset = []
    for i in range(num_face_image):
        img = Image.open(face_path + '/' + face_image[i])
        img = img.convert('L')
        img = img.resize((24,24),Image.ANTIALIAS)
        img = NPDFeature(np.array(img))
        dataset.append(np.concatenate((img.extract(),np.array([1]))))


    for i in range(num_nonface_image):
        img = Image.open(nonface_path + '/'+nonface_image[i])
        img = img.convert('L')
        img = img.resize((24,24),Image.ANTIALIAS)
        img = NPDFeature(np.array(img))
        dataset.append(np.concatenate((img.extract(),np.array([-1]))))

    return dataset
```

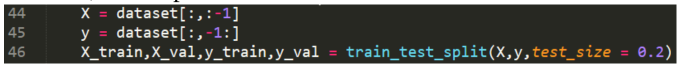
2. Processing data set data to extract NPD features. Extract features using the NPD Feature class in feature.py. (Tip: Because the time of the pretreatment is relatively long, it can be pretreated with pickle function library dump () save the data in the cache, then may be used load () function reads the characteristic data from cache.)



```
    def extract(self):
        '''Extract features from given image.

        Returns:
            A one-dimension ndarray to store the extracted NPD features.
        '''
        count = 0
        for i in range(self.n_pixels - 1):
            for j in range(i + 1, self.n_pixels, 1):
                self.features[count] = NPDFeature.__NPD_table__[self.image[i]][self.image[j]]
                count += 1
        return self.features
```

3. The data set is divided into training set and validation set, this experiment does not divide the test set.



```
    X = dataset[:,:-1]
    y = dataset[:,-1:]
    X_train,X_val,y_train,y_val = train_test_split(X,y,test_size = 0.2)
```

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, each training sample is given the same weight.

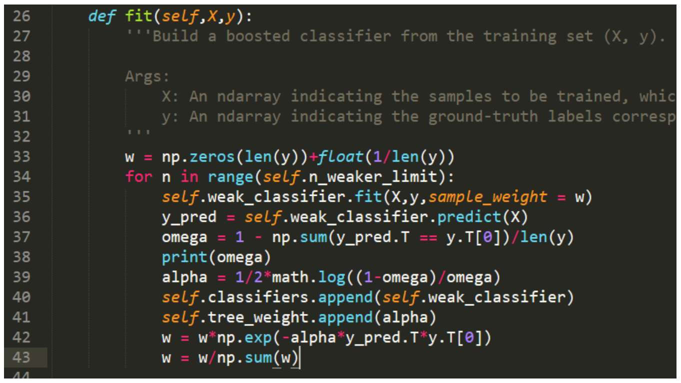
4.2 Training a base classifier, which can be sklearn.tree library DecisionTreeClassifier (note that the training time you need to pass the weight ω as a parameter).

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 ε.

4.5 Update training set weights ω.

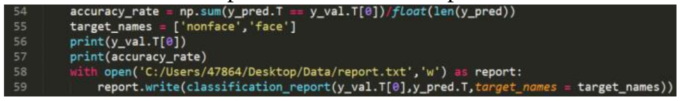
4.6 Repeat steps 4.2-4.6 above for iteration, the number of iterations is based on the number of classifiers.



```
    def fit(self,X,y):
        '''Build a boosted classifier from the training set (X, y).

        Args:
            X: An ndarray indicating the samples to be trained, whic
            y: An ndarray indicating the ground-truth labels corresp
        '''
        w = np.zeros(len(y))+float(1/len(y))
        for n in range(self.n_weaker_limit):
            self.weak_classifier.fit(X,y,sample_weight = w)
            y_pred = self.weak_classifier.predict(X)
            omega = 1 - np.sum(y_pred.T == y.T[0])/len(y)
            print(omega)
            alpha = 1/2*math.log((1-omega)/omega)
            self.classifiers.append(self.weak_classifier)
            self.tree_weight.append(alpha)
            w = w*np.exp(-alpha*y_pred.T*y.T[0])
            w = w/np.sum(w)
```

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 .

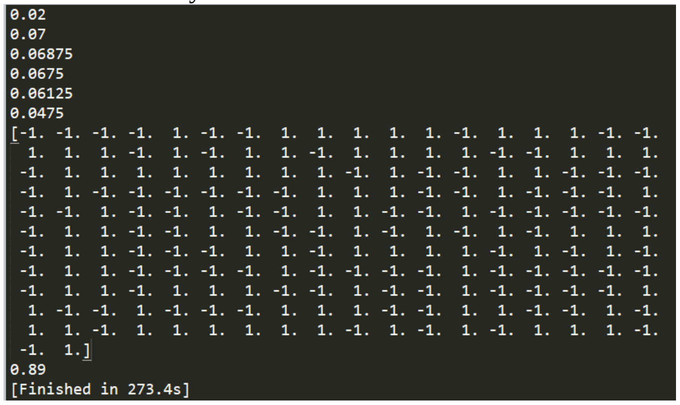


```
    accuracy_rate = np.sum(y_pred.T == y_val.T[0])/float(len(y_pred))
    target_names = ['nonface','face']
    print(y_val.T[0])
    print(accuracy_rate)
    with open('C:/Users/47864/Desktop/Data/report.txt','w') as report:
        report.write(classification_report(y_val.T[0],y_pred.T,target_names = target_names))
```

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

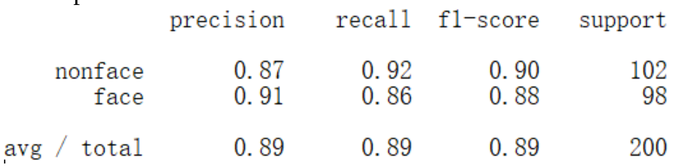
IV. CONCLUSION

The result of train.py is displayed as the follow picture:
The first six rows shows the error of the six Decision Tree Classifier;
The follow -1&1 matrix shows the validation set of y;
The final accuracy rate is 0.89.



```
0.02
0.07
0.06875
0.0675
0.06125
0.0475
[-1. -1. -1. -1.  1. -1. -1.  1.  1.  1.  1.  1. -1.  1.  1.  1. -1. -1.
  1.  1.  1. -1.  1. -1.  1.  1. -1.  1.  1.  1.  1. -1. -1.  1.  1.  1.
 -1.  1.  1.  1.  1.  1.  1.  1.  1. -1.  1. -1. -1.  1.  1. -1. -1. -1.
 -1.  1. -1. -1. -1. -1. -1. -1.  1.  1.  1. -1. -1.  1. -1. -1. -1.  1.
 -1. -1.  1. -1. -1.  1. -1. -1.  1.  1. -1. -1.  1. -1. -1. -1. -1. -1.
 -1.  1.  1. -1. -1.  1.  1. -1. -1.  1. -1.  1. -1.  1. -1.  1.  1.  1.
 -1.  1.  1. -1. -1. -1.  1.  1. -1.  1.  1.  1.  1. -1.  1. -1.  1. -1.
 -1.  1.  1. -1. -1. -1. -1.  1. -1. -1. -1. -1.  1. -1. -1. -1. -1. -1.
 -1.  1.  1. -1.  1.  1.  1. -1. -1.  1. -1. -1.  1. -1. -1. -1. -1.  1.
  1. -1. -1.  1. -1. -1. -1.  1.  1.  1. -1. -1.  1.  1. -1. -1.  1. -1.
  1.  1. -1.  1.  1.  1.  1.  1.  1. -1.  1. -1.  1. -1.  1.  1.  1. -1.
 -1.  1.]
0.89
[Finished in 273.4s]
```

The report.txt:



```
             precision    recall  f1-score   support

    nonface       0.87      0.92      0.90       102
       face       0.91      0.86      0.88        98

avg / total       0.89      0.89      0.89       200
```