The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
CaiZhe ,Jianxiong Li , YaoYing

Supervisor:
Qingyao Wu

Student ID: 201721045701 ,
201721045640 , 201721045695

Grade:
Graduate

December 21, 2017

Face Classification Based on AdaBoost Algorithm

Abstract—

The recognition and classification of face images by computer plays a great role in many aspects. Image recognition is an important application direction of machine learning technology. In this experiment, we are going to complete a Face classification System Based on AdaBoost Algorithm. This system can classify face images and non face images.

I. INTRODUCTION

In this experiment we Process data set data to extract NPD features .And then we used AdaBoost Algorithm to train the model.At last we use classification_report() of the sklearn.metrics library function writes predicted result to report.txt.

The motivation for this experiment is to Learn to use Adaboost to solve the face classification problem,and combine the theory with the actual project. Understand Adaboost further, get familiar with the basic method of face detection and Experience the complete process of machine learning.

II. METHODS AND THEORY

A image feature called Normalized Pixel Difference (NPD) 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 forAdaptive Boosting, is a machine learning meta-algorithm 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.

III. EXPERIMENT

1.Data sets and data analysis:

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. The dataset is included in the example repository. Please download it and divide it into training set and validation set.

2.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.

2.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.)

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

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.

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

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

4.4 Calculate the parameter according to the classification error rate .

4.5 Update training set weights .

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

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 .

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

3.Algorithm method

3.1 first step:

Initialize training data (per sample) weight distribution. Each training sample is given the same weight w = 1 / N at initialization. N is the total number of samples.D1 represents the weight of each sample for the first iteration. w11 represents the weight of the first sample at the first iteration. N is the total number of samples.

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\cdots,N$$

3.2 The second step:

multiple iterations, m = 1,2 .... m represents the number of iterations

a) Learning using a training sample set with weight distribution Dm (m = 1, 2, 3 ... N) gives a weak classifier.This formula shows that the weak classifier at the mth iteration classifies the sample x either into -1 or into 1.

$$G_m(x): \quad \chi \to \{-1,+1\}$$

Criterion: The error function of the weak classifier is the smallest, that is, the sum of the weights corresponding to the mis-divided samples is the smallest.

$$\epsilon_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

b) Calculate the discourse weight of the weak classifier Gm (x). The speech right am represents the importance of Gm (x) in the final classifier. Where em, is the εm in the step (the value of the error function).This formula increases as em decreases. That is, the error rate of small classifier, in the final classifier is more important.

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$$

c) Update the weight distribution of the training sample set. For the next iteration. Among them, the misclassified samples will increase the weight, the weight is correctly reduced.Dm + 1 is the weight of the sample for the next iteration, Wm + 1, where i is the weight of the ith sample at the next iteration.

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N}),$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

Where yi represents the category (1 or -1) corresponding to the ith sample and Gm (xi) represents the classification (1 or -1) of the sample xi by the weak classifier. If the points, yi * Gm (xi) the value of 1, otherwise -1. Where Zm is the normalization factor, so that the sum of the weights corresponding to all the samples is 1.

$$Z_m = \sum_{i=1}^{N} w_{mi}\exp(-\alpha_m y_i G_m(x_i))$$

3.3 The third step:

After the iteration is completed, combined weak classifier .Add a sign function on f (x), which finds the sign of the value. Value is greater than 0, 1. Is less than 0, is -1, equal to 0, is 0. The resulting strong classifier G (x)

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

4.Core code

train.py:

```
X = np.load("X.npy")

y = np.load("y.npy")

mode = DecisionTreeClassifier(criterion='gini')

adaBoost = ensemble.AdaBoostClassifier(mode, 10)

xTrain, xValidation, yTrain, yValidation = 
train_test_split(X, y, test_size=0.5, random_state=42)

m = adaBoost.n_weakers_limit

for i in range(m):

    mode = adaBoost.fit(xTrain, yTrain)

    xTest = adaBoost.predict(xValidation)

    errorRate=0;

    for j in range(xValidation.shape[0]):

        if xTest[j] != yValidation[j]:

            errorRate =errorRate+adaBoost.weight[j]

    if errorRate>0.5:

        break
```

```
        alpha = math.log((1-errorRate)/errorRate)/2
        z=0
        for k in range(adaBoost.weight.shape[0]):
            z=z+adaBoost.weight[k]*math.exp(-alpha*yTrain[k]*xTest[k])

        for k in range(adaBoost.weight.shape[0]):
            adaBoost.weight[k]=adaBoost.weight[k]*math.exp(-alpha*yTrain[k]*xTest[k])/z
        adaBoost.weak_classifier_list.append(mode)
        adaBoost.alphas.append(alpha)
    h = adaBoost.predict_scores(xValidation)

    labels=[-1,1]
    target_names = ['face','nonface']

print(classification_report(yValidation,h,labels,target_names))
```

ensemble.py:

```
def __init__(self, weak_classifier, n_weakers_limit):
        self.weak_classifier = weak_classifier
        self.n_weakers_limit = n_weakers_limit
        self.alphas = []
        self.weak_classifier_list = []
        self.weight = None
def fit(self,X,y):
        if self.weight is None:
            self.weight = np.zeros(X.shape[0])
            for i in range(self.weight.shape[0]):
                self.weight[i] = 1/self.weight.shape[0]
        self.weak_classifier = DecisionTreeClassifier(criterion='gini')
        self.weak_classifier.fit(X, y,sample_weight=self.weight)
        return self.weak_classifier
def predict_scores(self, X):
        h= np.zeros(X.shape[0])
        for i in range(len(self.weak_classifier_list)):
            h = h+self.alphas[i]*self.weak_classifier_list[i].predict(X)
        for i in range(h.shape[0]):
            if h[i]<=0:
                h[i]=-1
            else:
                h[i]=1
        return h
def predict(self, X, threshold=0):
        xTest = self.weak_classifier.predict(X)
        return xTest
```

5. Experimental results

This experiment uses the Adaboos method to solve the face classification problem, and obtains the optimal model by continuously training the model on a large number of photos containing human faces and non-human faces. Verify that this optimal model yields the test results shown in the image below. The accuracy of identifying whether a photo contains a face is 91% and the recall is 91%.

	precision	recall	f1-score	support
face	0.90	0.93	0.91	253
nonface	0.92	0.89	0.91	247
avg / total	0.91	0.91	0.91	500

IV. Conclusion

After this experiment, I further understand the principle of AdaBoost Algorithm. AdaBoost is often referred to as the best out-of-the-box classifier.At the same time we know that we should work hard if we want to leran something.