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:
胡永浩 袁振宇

Supervisor:
Qingyao Wu

Student ID：
201530611630 and 201530613542

Grade:
Undergraduate

December 12, 2017

Face Classification Based on AdaBoost Algorithm

Abstract—

I. INTRODUCTION

The Adaboost algorithm is used to analyze the processed image features, and the classification model is trained according to the features for face classification

II. METHODS AND THEORY

Adaboost：Adaboost is an iterative algorithm. Its core idea is to train different classifiers (weak classifiers) on the same training set, and then combine these weak classifiers to form a stronger final classifier (strong classifier)

III. EXPERIMENT

Dataset：This experiment provides 1000 pictures, of which 500 is a human face containing RGB pictures， the other 500 are not included face RGB images

Experimental steps：

1. 读取数据集数据。读取图片，将全部图片转成大小为 24*24 的灰度图，数据集正负类样本的个数和比例不限，数据集标签形式不限。
2. 处理数据集数据，提取 NPD 特征。使用 *feature.py* 中 *NPDFeature* 类的方法提取特征。（提示：因为预处理数据集的时间比较长，可以用 *pickle* 库中的 *dump()* 函数将预处理后的特征数据保存到缓存中，之后可以使用 *load()*函数读取特征数据）
3. 将数据集切分为训练集和验证集，本次实验不切分测试集。
4. 根据 *ensemble.py* 中的预留的接口编写 *AdaboostClassifier* 所有函数。以下为 *AdaboostClassifier* 类中的 *fit()*方法的思路：

    4.1 初始化训练集的权值 ,每一个训练样本被赋予相同的权值。

    4.2 训练一个基分类器，基分类器可以使用 *sklearn.tree* 库中 *DecisionTreeClassifier*(注意训练的时候需要将权重 作为参数传入)。

    4.3 计算基分类器在训练集上的分类误差率 。

    4.4 根据分类误差率 ，计算参数 。

    4.5 更新训练集的权值 。

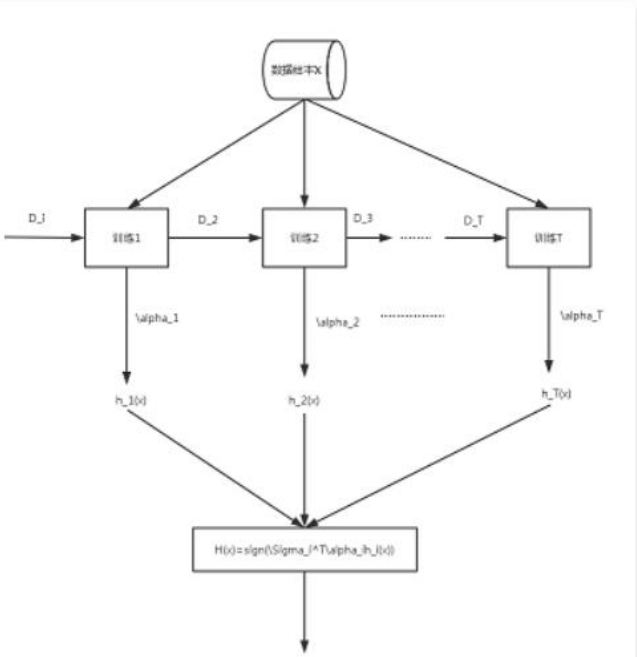
    4.6 重复以上 4.2-4.6 的步骤进行迭代，迭代次数为基分类器的个数。
5. 用 *AdaboostClassifier* 中的方法在验证集上进行预测并计算精确率,并用 *sklearn.metrics* 库的 *classification_report()*函数将预测结果写入 *report.txt* 中。

IV. CONCLUSION

The idea of Adaboost algorithm is to integrate a strong classifier with high accuracy through a few simple weak classifiers.

Adaboost Analysis：

Although there are no restrictions on weak classifiers in AdaBoost, they can be decision trees and SVM based on multidimensional features. However, each weak classifier is usually constructed based on a certain dimension of all features and the output is only +1, 1 two kinds (two classification problem), so each round of iteration in training corresponds to the selection of the best features.



Code（fit（））：

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

    Returns:
        X: An ndarray indicating the samples to be trained, which shape should be (n_samples,n_features).
        y: An ndarray indicating the ground-truth labels correspond to X, which shape should be (n_samples,1).
    '''
    w = 1/X.shape[0] * np.ones((X.shape[0]))
    alpha_array = np.empty((self.n_weakers_limit))

    for m in range(self.n_weakers_limit):
        # 训练弱分类器
        self.weak_classifiers[m].fit(X, y, sample_weight=w)
        # 用弱分类器预测训练集
        y_predict = np.sign(self.weak_classifiers[m].predict(X))

        error_rate = 0
        for i in range(y_predict.shape[0]):
            if (y_predict[i] != y[i][0]):
                error_rate += w[i]
    if (error_rate > 0.5):
        print('There may be some error!')
        print('Error rate is bigger than 0.5')
        print('Training has stopped')
        break

    alpha = math.log((1 - error_rate) / error_rate) / 2
    alpha_array[m] = alpha

    Zm = 0
    exp_array = np.empty(w.shape)
    for i in range(w.shape[0]):
        e = math.exp(-alpha * y[i][0] * y_predict[i])  # y为二维数组，y_predict为一维数组
        exp_array[i] = e
        Zm += w[i] * e

    for i in range(w.shape[0]):
        w[i] = w[i] * exp_array[i] / Zm

self.alpha_array = alpha_array

    print(self.alpha_array)
    for m in range(self.n_weakers_limit):
        print(id(self.weak_classifiers[m]))
```