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:
Zhaoyu Zhou, Hongjing Li and Liqin zan

Supervisor:
Qingyao Wu

Student ID：201530613900,
201530081266 and 201530651780

Grade:
Undergraduate

December 21, 2017

Face Classification Based on AdaBoost Algorithgm

Abstract—We use Adaboost to solve the face classification problem and combine theory with the actual project for learning.

I. INTRODUCTION:

In this experiment, we'd like to finish the task as follows:

（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 theory with the actual project

(4) Experience the complete process of machine learning

II. *METHODS AND THEORY :*

The define of 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).

Adaboost structure: the final classifier YM is composed of several weak classifiers (weak classifiers), which is equivalent to the last m weak classifiers to vote for the classification results, and each weak classifier "speaking right" Factor α is not the same size.

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.

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).

The experiment code:

train.py:

```
#图片处理
def process_image(path, purpose_path):
    files = os.listdir(path)
    for i in files:
        #将图片转化为灰度图
        im = Image.open(path + i).convert('L')
        #将图片转化为 24*24 的大小
        im.resize((24, 24),Image.ANTIALIAS)
        #保存图片
        im.save(purpose_path + i)

#获取特征
def get_feature(path):
    features = numpy.array([])
    files = os.listdir(path)
```

```
        for k in range(len(files)):
            im = Image.open(path + files[k])
            image = numpy.ones(shape=(24, 24), dtype=int)
            for i in range(24):
                for j in range(24):
                    image[i][j] = im.getpixel((i, j))
            NPDFeature1 = NPDFeature(image)
            feature = NPDFeature1.extract()
            features = numpy.concatenate((features, feature))
        return features

#预处理过程
def pre_process():
        face_features = numpy.array([])
        nonface_features = numpy.array([])
        features = numpy.array([])

        #转化图片
        process_image(path_face_origin, path_face_purpose)
        process_image(path_nonface_origin, path_nonface_purpose)

        num_face = len(os.listdir('datasets/original/face'))
        num_nonface = len(os.listdir('datasets/original/nonface'))

        #获取特征
        face_features = get_feature(path_face_purpose)
        nonface_features = get_feature(path_nonface_purpose)

        print(face_features.shape,nonface_features.shape)
        print(num_face,num_nonface)

        #改变特征形状
        face_features.shape = num_face, 165600
        nonface_features.shape = num_nonface, 165600

        #准备加入 label
        face_y = numpy.ones(shape=(num_face, 1))
        nonface_y = numpy.zeros(shape=(num_nonface, 1))
        nonface_y -= numpy.ones(shape=(num_nonface, 1))

        #加入 label
        face_features = numpy.concatenate([face_features, face_y], axis=1)
```

```
        nonface_features = numpy.concatenate([nonface_features, nonface_y], axis=1)

        #将所有数据合并
        features = numpy.row_stack((face_features,nonface_features))

        #写入缓存
        AdaBoostClassifier.save(features,"feature.data")
        return features

if __name__ == "__main__":
        #Data 格式为[X:y]
        if os.path.exists('feature.data'): #如果预处理过，直接用 load()读取数据
                Data = AdaBoostClassifier.load('feature.data')
        else :
                Data = pre_process()

        #将 X_data 与 y_data 分开
        X_data,y_data = Data[:,:-1],Data[:,-1]

        #切分训练集与验证集
      X_train,X_test,y_train,y_test=train_test_split(X_data,y_data,test_size=0.3,random_state=10)

        print(len(y_train),len(y_test))

        #进行 AdaBoost 训练
        mode = tree.DecisionTreeClassifier(max_depth=1)
        adaboost=AdaBoostClassifier(mode,20)
        adaboost.fit(X_train,y_train)

        #得到预测结果
        y_predict=adaboost.predict(X_test)

        #输出正确率
        count=0
        for i in range(len(y_test)):
                if y_test[i]==y_predict[i]:
                        count=count+1
        target_names = ['1', '-1']
        print(count/len(y_test))

        #调用 classification_report 获得预测结果
        report=classification_report(y_test, y_predict, target_names=target_names)
```

```
    #写入 report.txt
    with open("report.txt", 'w') as f:
        f.write(report)
    print(report)
```

ensemble.py:

```
def fit(self,X,y):
        num=len(y)
        w = np.empty(shape=num, dtype=float)
        for i in range(num):
            w[i] = 1.0 /num
        for t in range(self.n_weakers_limit):
            clf = DecisionTreeClassifier(max_depth=1)
            clf.fit(X, y,w)
            epsilon = 0
            y_test=clf.predict(X)
            print(y_test)
            print(y)
            count=0
            for i in range(len(y)):
                if y_test[i]!= y[i]:
                    epsilon =epsilon+ w[i]
                else:
                    count=count+1
            print(count)
            if epsilon > 0.5: break
            self.alpha_list.append(0.5 * math.log((1-epsilon)/epsilon))
            z_sum = 0
            for i in range(len(y)):
                w[i] = w[i] * math.exp(-self.alpha_list[t]*y_test[i]*y[i])
                z_sum +=w[i]
            for i in range(len(y)):
                w[i] /= z_sum
            self.h_list.append(clf)
        pass
def predict(self, X, threshold=0):
        for i in range(X.shape[0]):
            s = 0
            for j in range(len(self.h_list)):
                s += self.h_list[j].predict(X[i].reshape(1,-1)) * self.alpha_list[j]
            if s >= threshold:
```

```
                self.res.append(1)
            else:
                self.res.append(-1)
        return self.res
        Pass
```

Experiment result :

We have 10 weak classifiers,and the max-depth is 1.
The accuracy is about 0.95.

Report.txt:

	precision	recall	f1-score	support
1	0.96	0.95	0.95	75
-1	0.95	0.96	0.95	75
avg / total	0.95	0.95	0.95	150

IV. ***CONCLUSION:***

The more weak classifiers, the higher the accuracy.
The more max-depth, the higher the accuracy.

If we change the pictures at first, then the pictures' size would be changed when we change them to Grayscale.

If the decision tree's depth is deep, the algorithm we performed would be invalid because epsilon would be zero when we do so.

Summary:

In the course of this experiment, the Internet access to a lot of information on face recognition, excavation although completed and can not touch the face recognition, but far from the real life needs. The pixels of the faces and nonface libraries we experiment with are not only very low, but also the position and size of each face is very close, which greatly reduces the difficulty. There are many things that can be learned in this direction.