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:
袁澍民 and 刘志杰

Supervisor:
Qingyao Wu

Student ID：
201530613528 and 201530612392

Grade:
Undergraduate

December 21, 2017

基于 AdaBoost 算法的人脸分类

Abstract—

I. INTRODUCTION

Today,I try to solve a question which named Face Classification Based on AdaBoost Algorithm, There are the terminals of this question:
1.In-depth understanding of Adaboost's principle
2.Familiar with the basic method of face detection
3.Learn to use Adaboost to solve the face classification problem, the theory and the actual project
4.Experience the complete process of machine learning

This experiment provides 1000 images, of which 500 are face-containing RGB images stored in ./datasets/original/face; the other 500 are non-faceted RGB images stored in ./datasets/original/nonface Inside.
Dataset included in the sample warehouse, please download and cut it into a training set, validation set.

II. METHODS AND THEORY

1.Read dataset data. Read the picture, all the pictures into a size of 24 * 24 grayscale, the number of positive and negative samples of the data set the number and proportion of any data set label is not limited.
2.Processing dataset data and extracting NPD features. Extract features using the NPDFeature class in feature.py. (Hint: You can use the dump () function in the pickle library to save the preprocessed signature data to the cache because the data set can be preprocessed for a long time. You can then use the load () function to read the signature data)
3.The data set is divided into training set and verification 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 idea of the fit () method in the AdaboostClassifier class:
4.1 Initialize the training set weights, each training sample is given the same weight.
4.2 Training a base classifier, the base classifier can use the sklearn.tree library DecisionTreeClassifier (note that when training 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 According to the classification error rate, calculate the parameters.
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.
Predict and calculate the accuracy on the validation set using the method in AdaboostClassifier and write the prediction result to report.txt using the classification_report () function of the sklearn.metrics library.
Sort out the experimental results and complete the experimental report.

III. EXPERIMENT

```
#train.py
import numpy
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.externals.joblib import Memory
from sklearn.datasets import load_svmlight_file
import numpy as np
import pickle
from numpy import *
import matplotlib.image as mpimg
from skimage import io
import os
from feature import *
from PIL import Image
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from ensemble import AdaBoostClassifier
#这里是直接读取灰度图，灰度图在 original 文件夹里面
path1=[os.path.join('G:\\Users\\qqqqqq1997520\\Desktop\\original\\face',f) for f in os.listdir('G:\\Users\\qqqqqq1997520\\Desktop\\original\\face')]
path2 = [os.path.join('G:\\Users\\qqqqqq1997520\\Desktop\\original\\nonface',f) for f in os.listdir('G:\\Users\\qqqqqq1997520\\Desktop\\original\\nonface')]
ABC=AdaBoostClassifier(DecisionTreeClassifier(), 1)
im=[0 for i in range(1000)]
for i in range(500):
    im[i]=plt.imread(path1[i])
for i in arange(500,1000):
    im[i]=plt.imread(path2[(i%500)])
_feature=[0 for i in range(1000)]
for i in range(1000):
    feature=NPDFeature(im[i])
    _feature[i]=feature.extract()
```

```
    feature_data=array(_feature)
    y=[1 for i in range(1000)]
    for i in range(500,1000):
        y[i]=-1;
    y=array(y)
    X_train, X_vali, y_train, y_vali = train_test_split(feature_data, y, test_size=0.3, random_state=37)

    ABC=AdaBoostClassifier(DecisionTreeClassifier(),20)
    ABC.fit(X_train,y_train)
    predict=ABC.predict(X_vali,0)
    classification_name=["Y","N"]
    with open('report.txt', 'w') as f:
        f.write(classification_report(y_vali, predict, target_names=classification_name))

    #ensemble

    import numpy
    import matplotlib.pyplot as plt
    %matplotlib inline
    from sklearn.externals.joblib import Memory
    from sklearn.datasets import load_svmlight_file
    import numpy as np
    import pickle
    from numpy import *
    import matplotlib.image as mpimg
    from skimage import io
    import os
    from PIL import Image
    from sklearn.metrics import classification_report
    from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeClassifier

    import pickle

    class AdaBoostClassifier:
        def __init__(self, weak_classifier, n_weakers_limit):
            self.weak_classifier = weak_classifier
            self.n_weakers_limit = n_weakers_limit
            self.classifier_list = []
            self.alpha = zeros(n_weakers_limit)

        def fit(self,X,y):

            w=np.ones(X.shape[0])/(X.shape[0])
            for i in range(self.n_weakers_limit):

classification=DecisionTreeClassifier(max_depth=1)
                classification.fit(X,y,sample_weight=w)
                self.classifier_list.append(classification)
                predict=classification.predict(X)
                x=np.sum(w[y!=predict])
                self.alpha[i]=0.5*np.log((1-x)/x)
                sumW=w*np.exp((-1)*y*predict)
                w=sumW/np.sum(sumW)

        def predict_scores(self, X):
            s=[]
            score=zeros(X.shape[0])
            for i in range(self.n_weakers_limit):
                s.append(self.classifier_list[i].predict(X))
                score += self.alpha[i] * s[i]
            return score

        def predict(self, X, threshold=0):
            score = self.predict_scores(X)
            score[score > threshold] = 1
            score[score < threshold] = -1
            return score

        @staticmethod
        def save(model, filename):
            with open(filename, "wb") as f:
                pickle.dump(model, f)

        @staticmethod
        def load(filename):
            with open(filename, "rb") as f:
                return pickle.load(f)
```

forecast result:

	precision	recall	f1-score	support
Y	0.92	0.94	0.93	149
N	0.94	0.91	0.93	151
avg / total	0.93	0.93	0.93	300

IV. CONCLUSION

The classifier can distinguish human face with high accuracy, but the feature extraction function has more features extracted. Only 24 * 24 grayscale has extracted more than 160,000 features, which will take a lot of time if large-scale extraction is performed. The increase of maximum depth should improve classification accuracy, but there is not much in-depth testing due to the machine's performance.