South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

December 18, 2017

Author:

Shengjie Wang and Rui Lv and Zhixiang Xue

王圣杰　吕睿　薛志翔

Supervisor:

Qingyao Wu

Student ID：

201530612873 and 20153061247
7 and 201530612873

Grade:

Undergraduate

Face Classification Based on AdaBoost Algorithm

Abstract—

I. Introduction

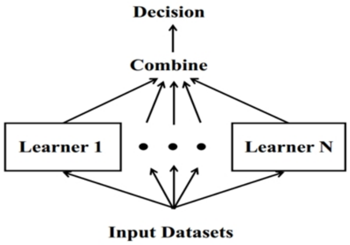
This experiment is for:

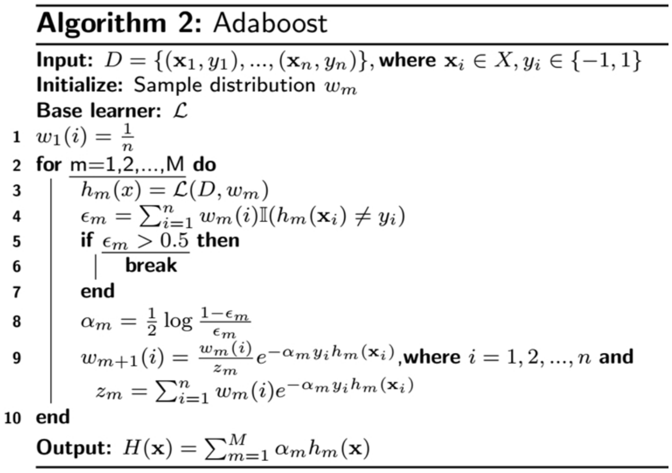
1. Understand Adaboost further
2. Get familiar with the basic method of face detection
3. Learn to use Adaboost to solve the face classification problem,and combine the theory with the actual project
4. Experience the complete process of machine learning

II. Methods and Theory

Ensemble learning: Combine numerous weak learners to a strong learner

Main methods: Boosting, Bagging





```
Algorithm 2: Adaboost
Input: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in X, y_i \in \{-1, 1\}$
Initialize: Sample distribution $w_m$
Base learner: $\mathcal{L}$
$w_1(i) = \frac{1}{n}$
for m=1,2,...,M do
    $h_m(x) = \mathcal{L}(D, w_m)$
    $\epsilon_m = \sum_{i=1}^{n} w_m(i)\mathbb{I}(h_m(\mathbf{x}_i) \neq y_i)$
    if $\epsilon_m > 0.5$ then
        break
    end
    $\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}$
    $w_{m+1}(i) = \frac{w_m(i)}{z_m}e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$,where $i = 1, 2, ..., n$ and
      $z_m = \sum_{i=1}^{n} w_m(i)e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$
end
Output: $H(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$
```

III. Experiment

A. Dataset

1. 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*.
2. The dataset is included in the example repository. Please download it and divide it into training set and validation set.

B. 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).

C. *Implementation*

Source code:

train.py

```
import numpy


class NPDFeature():
    """It is a tool class to extract the NPD features.

    Attributes:

        image: A two-dimension ndarray indicating grayscale image.

        n_pixels: An integer indicating the number of image total pixels.

        features: A one-dimension ndarray to store the extracted NPD features.

    """

    __NPD_table__ = None


    def __init__(self, image):

        '''Initialize NPDFeature class with an image.'''

        if NPDFeature.__NPD_table__ is None:

            NPDFeature.__NPD_table__ = NPDFeature.__calculate_NPD_table()

        assert isinstance(image, numpy.ndarray)

        self.image = image.ravel()

        self.n_pixels = image.size

        self.features = numpy.empty(shape=self.n_pixels * (self.n_pixels - 1) // 2, dtype=float)


    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


    @staticmethod
    def __calculate_NPD_table():
        '''Calculate all situations table to accelerate feature extracting.'''
        print("Calculating the NPD table...")
        table = numpy.empty(shape=(1 << 8, 1 << 8), dtype=float)
        for i in range(1 << 8):
            for j in range(1 << 8):
                if i == 0 and j == 0:
                    table[i][j] = 0
                else:
                    table[i][j] = (i - j) / (i + j)
        return table

import pickle


class AdaBoostClassifier:


    @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)

import os
```

```
#定义返回指定路径 path 的所有文件的地址的数组
def get_path(path):
    return [os.path.join(path,f) for f in os.listdir(path)]


from PIL import Image
from pylab import *
import numpy as np
#将 face 文件夹里的 500 张图的路径存到 face_pth
face_pth = get_path('C:/Users/Administrator/Desktop/MLLab/ML2017-lab-03-master/datasets/original/face')
#定义一个链表通过对 face_pth 的遍历将每一张图转化为 24*24 的灰度图并存到 face 链表里
face=[]
for i in face_pth:
    im = Image.open(i).convert('L')
    im=np.array(im.resize((24,24)))
    face.append(im)


from PIL import Image
from pylab import *
import numpy as np
#将 nonface 文件夹里的 500 张图的路径存到 non_pth
non_pth = get_path('C:/Users/Administrator/Desktop/MLLab/ML2017-lab-03-master/datasets/original/nonface')
#定义一个链表通过对 non_pth 的遍历将每一张图转化为 24*24 的灰度图并存到 non_face 链表里
non_face=[]
for i in non_pth:
    im = Image.open(i).convert('L')
    im=np.array(im.resize((24,24)))
    non_face.append(im)
```

```
#定义一个face_f链表并对face进行遍历存储利用NPDFeature类的extract方法提取到的每个灰度图的特征

face_f=[]

for i in face:

  NPDf=NPDFeature(i)


face_f.append((np.array(NPDf.extract())).reshape((165600,1)))

#定义一个non_face_f链表并对non_face进行遍历存储利用NPDFeature类的extract方法提取到的每个灰度图的特征

non_face_f=[]

for i in non_face:

  NPDf=NPDFeature(i)


non_face_f.append((np.array(NPDf.extract())).reshape((165600,1)))



#调用AdaBoostClassifier类的静态方法save将上面提取到的人脸图的特征存到文件feature.txt中

AdaBoostClassifier.save(face_f,'feature.txt')



#调用AdaBoostClassifier类的静态方法save将上面提取到的非人脸图的特征存到文件feature.txt中

AdaBoostClassifier.save(non_face_f,'non_f.txt')



#调用AdaBoostClassifier类的静态方法load方法从上面缓存的数据中提取特征

fff=AdaBoostClassifier.load('feature.txt')

nnn=AdaBoostClassifier.load('non_f.txt')



#构造一个由300个人脸图和300个非人脸图组成的训练集

X_train=np.zeros((600,165600))

y_train=np.vstack((ones((300,1)),-ones((300,1))))

for i in range(300):

  X_train[i,:]=fff[i].reshape(165600,)

for i in range(300):

  X_train[i+300,:]=nnn[i].reshape(165600,)

#把样本和label合起来，方便下面进行随机打乱

Xy_train=np.hstack((X_train,y_train))



#将样本进行随机打乱

np.random.shuffle(Xy_train)



#将打乱的样本进行分割得到特征集和label集

X_train=Xy_train[:,0:165600]

y_train=Xy_train[:,165600].reshape(600,1)



#构造一个由200个人脸图和200个非人脸图组成的测试集

X_test=np.zeros((400,165600))

y_test=np.vstack((ones((200,1)),-ones((200,1))))

for i in range(200):

  X_test[i,:]=fff[i+300].reshape(165600,)

for i in range(200):

  X_test[i+200,:]=nnn[i+300].reshape(165600,)



#训练

from sklearn import tree

T = 20 #设置基分类器的个数

h_list=np.zeros((T,400))  #创建数组以储存各个基分类器的结果

alpha=np.zeros((T,1))  #创建数组以储存各个基分类器对应的Alpha

w=np.ones((600,1))/600 #初始化所有训练样本的权值

mode=tree.DecisionTreeClassifier(max_depth=2)  #生成一个
```

```
初始的分类方法
for i in range(T):
   mode.fit(X_train,y_train,sample_weight=w.reshape(600,)) #生成单个基分类器
   epsilon=1-mode.score(X_train,y_train)   #计算分类的错误率 epsilon
   if epsilon>0.5:
      break
   alpha[i]=0.5*np.log((1-epsilon)/epsilon) #计算 Alpha
   #更新权重
   w=w*np.exp(-alpha[i]*y_train*((mode.predict(X_train)).reshape(600,1)))
   Zm = np.sum(w)
   w = w/Zm

   h_list[i]=mode.predict(X_test)#储存基分类器在测试集上的结果
   #计算并储存各基分类器 Boost 的结果
y_label=(h_list*alpha).sum(axis = 0)
y_label[y_label>=0]=1
y_label[y_label<0]=-1
#计算在测试集上的准确率
right=0
for i in range(400):
   if y_label[i]==y_test[i]:
      right=right+1
print(right/400.0)

import codecs
from sklearn import metrics
#用 sklearn.metrics 库的 classification_report()函数将预测结果写入 report.txt 中
f=codecs.open('C:/Users/user/Desktop/ML2017-lab-03-master/report.txt','w')
f.write(metrics.classification_report(y_test, y_label))
```

IV.CONCLUSION

We took 600 samples as training set, and 400 samples as test set, each of them consists of half face data, and half non-face data.

Then we selected DecisionTree as our model of base classifier.

After a iteration of 20 times, we got a AdaBoost classifier that can reach 97% on the currect rate.

First, we had learnt how to deal with images as the data of Machine Learning: to transfer colorful images into matrixes of pixels, than exact features from them. Second, through this experiment we had a futher understanding of Adaboost and the theory of Ensemble. Third, it’s the first for us to experience a whole process of Machine Learning as an actural project.