Homework 1 : Face detection

Report

Part I. Implementation (6%):

● Please screenshot your code snippets of Part 1, Part 2, Part 4, and explain your implementation. For example,

Part1:

```
15      # Begin your code (Part 1)
16      dataset=[]
17      Path=os.path.join(dataPath,'face')
18      #set up the path to reach the folder
19      for filename in os.listdir(Path):
20      #traverse all images in the folder.
21          img_path=os.path.join(Path,filename)
22          #combine the path of the image
23          img=cv2.imread(img_path , cv2.IMREAD_GRAYSCALE)
24          #read the image
25          datatuple = (img, 1)
26          dataset.append(datatuple)
27          #store the data in the array dataset[] as the format datatuple
28
29      Path=os.path.join(dataPath,'non-face')
30      for filename in os.listdir(Path):
31          img_path=os.path.join(Path,filename)
32          img=cv2.imread(img_path , cv2.IMREAD_GRAYSCALE)
33          datatuple = (img, 0)
34          dataset.append(datatuple)
35      # raise NotImplementedError("To be implemented")
36      # End your code (Part 1)
```

Step1:

First set up the path to reach the folder (’face’ and ‘non-face’), then use for loop to traverse all images in the folder.

Step2:

However, os.listdir() return only the filename, so I use os.path.join() to combine the path of the image, then use cv2.imread() to read the image file.

Step3:

For face, label should be 1; for non-face,label should be 0. Finally, store the data in the array dataset[] as the format datatuple, first element is a numpy array, and the second element is the label.

Part2:

```
150         # Begin your code (Part 2)
151         bestError=float("inf")
152         #Set up the variable bestError equal to infinite
153         for i in range(len(featureVals)):
154             weightError=0.0
155             for j in range(len(featureVals[i])):
156                 if featureVals[i][j]<0:
157                     weightError+=weights[j]*abs(1-labels[j])
158                     #Formula:Error=sum(wight*|hj(Xi)-Yi|)
159                 else:
160                     weightError+=weights[j]*abs(0-labels[j])
161             if weightError<bestError:
162             #Compare the error of each classifier
163                 bestError=weightError
164                 #choose the minimum error one recorded in variable bestError
165                 bestClf=WeakClassifier(features[i])
166                 #use bestClf to recorded the best minimum error classifier
167         # raise NotImplementedError("To be implemented")
168         # End your code (Part 2)
```

Step1:

Set up the variable bestError equal to infinite, so that it can be the first standard to compare.

Step2:

Use two layer of for loop to traverse the table in order to do the formula.

Formula:Error=sum(wight*|hj(Xi)-Yi|)

Step3:

Compare the error of each classifier and choose the minimum error one recorded in variable bestError, then use bestClf to recorded the best minimum error classifier.

Part4:

```python
    # Begin your code (Part 4)
    green_color=(0,255,0)#BGR
    red_color=(0,0,255)#BGR

    f=open(dataPath,'r')
    #Open detectData.txt file
    for line in f:
    #read each line then stored the data
        filename=line.split()[0]
        path=os.path.join('data/detect',filename)
        #use os.path.join() to combine that it can be a file path
        img=cv2.imread(path)
        faceNum=int(line.split()[1])
        #change the type into int
        for i in range(faceNum):
            data=f.readline()
            x=int(data.split()[0])
            y=int(data.split()[1])
            width=int(data.split()[2])
            height=int(data.split()[3])
            crop_img=img[y:y+height,x:x+width]
            #divide the face image in each image
            crop_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
            Img=cv2.resize(crop_img,(19,19),interpolation=cv2.INTER_AREA)
            #save the cutting image be grayscale and the size be (19*19)
            if(clf.classify(Img)==1):
            #Use clf.classify to determine whether the cutting image
                cv2.rectangle(img,(x,y),(x+width,y+height),green_color,2)
                #the rectangle line should be green
            else:
                cv2.rectangle(img,(x,y),(x+width,y+height),red_color,2)
                #the rectangle line should be red
        cv2.imshow('image',img)
        #show the original image
        cv2.waitKey(0)

    # raise NotImplementedError("To be implemented")
    # End your code (Part 4)
```

Step1:

Open detectData.txt file, read each line then stored the data. Notice that the type we read in each line split is string, for the use to be file path, we should use os.path.join() to combine that it can be a file path; for some use to do operation or be the coordinate, we should change the type into int.

Step2:

Use the data to divide the face image in each image, then save the cutting image be grayscale and the size be (19*19).

Step3:

Use clf.classify to determine whether the cutting image is face (return 1) or non-face (return 0), if is face, the rectangle line should be green; else , the rectangle line should be red. Finally, show the original image.

Part II. Results & Analysis (12%):

- Please screenshot the results. For instance,

T=10

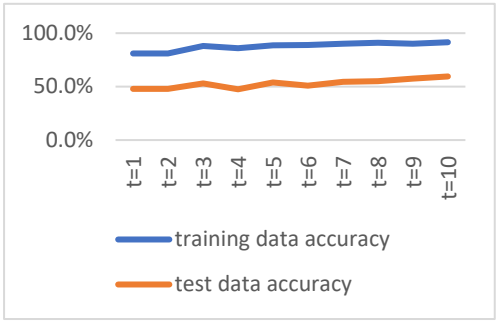
```
Console 1/A
10, 2)], negative regions=[RectangleRegion(5, 4, 10, 2)]) with accuracy: 145.000000 and alpha: 0.719869
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12,
11, 5, 1)], negative regions=[RectangleRegion(12, 12, 5, 1)]) with accuracy: 72.000000 and alpha: 0.685227
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10,
4, 1, 1)], negative regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and alpha: 0.707795
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9,
2, 2), RectangleRegion(2, 11, 2, 2)], negative regions=[RectangleRegion(2, 9, 2, 2), RectangleRegion(4,
11, 2, 2)]) with accuracy: 137.000000 and alpha: 0.811201

Evaluate your classifier with training dataset
False Positive Rate: 17/100 (0.170000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 183/200 (0.915000)

Evaluate your classifier with test dataset
False Positive Rate: 45/100 (0.450000)
False Negative Rate: 36/100 (0.360000)
Accuracy: 119/200 (0.595000)

Detect faces at the assigned location using your classifier
```

● Your analysis or observation. Please discuss the performance difference between the training and testing dataset, and present the results using a table or chart as follows.



200張	training data accuracy	test data accuracy
t=1	81.0%	48.0%
t=2	81.0%	48.0%
t=3	88.0%	53.0%
t=4	86.0%	47.5%
t=5	88.5%	54.0%
t=6	89.0%	51.0%
t=7	90.0%	54.5%
t=8	91.0%	55.0%
t=9	90.0%	57.5%
t=10	91.5%	59.5%

According to the table and chart, in microscopic view, we can see both training data and test data accuracy are not always decrease as T decrease, but in macroscopic view, the accuracy when T=1 is actually smaller than the accuracy when T=10. Compare the difference between training and test dataset, obviously, the accuracy of training data is larger than test data, because that for the classifier, data set is a totally unknown dataset, and that the classifier is use training dataset to train, that why the accuracy is higher.

T=1

T=5

T=8 T=10


From the image, we can see the accuracy perform om the image, the different number of classifier would have different performance.

Part III. Answer the questions (12%):

1. **Please describe a problem you encountered and how you solved it.**

   There was too much variable in the previous code, before I start to doing homework, I take a lot of time to understand them.

   When doing part1, at the beginning, because I do the job that putting the file path (C:\Users\jenny\OneDrive\桌面\AI_HW1\data\test) like this, then the result would only read two folder (face and non-face) in the test folder, but never read any image in the folder.

   When doing part4, at first I used for line in f.readline() to read the file line by line, but there was always ran error, I try to print what I read in each loop, then I found that using this method, it will divide a word to many char, that is why always error.

2. **What are the limitations of the Viola-Jones' algorithm?**

   From the perspective of result, Viola-Jones' may detect exact same face to different result, this may cause by what the rectangle window cover. Another limitation is Viola-Jones can only detect front face, even if wear glass would cause to error detect, and the light of the face may influence the result.

   From the perspective of program running time, Viola-Jones need to take time training classifier, and the training process is slow.

3. **Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?**

   There are two methods to improve the accuracy. First, turn the image into grayscale before use imread to read the image file in. Second, rotate the window of face 45 degree.

4. **Please propose another possible face detection method (no matter how good or bad, please come up with an idea). Please discuss the pros and cons of the idea you proposed, compared to the Adaboost algorithm.**

To detect faces, maybe we can design an algorithm that detect the main face features, for example, eyes, nose, mouth and so on. If the algorithm can get the position of the feature, and check that the features' distance is in an acceptable range, then it can be classifier as a face.

Talking about the pros, for a clear front face, it we be very easy to detect whether it is face or not without complex computation. However, talking about the cons, because the rules is too general, it is hard to build an appropriate set of rules.