Homework 1: Face Detection

Spring 2022 Introduction to Artificial Intelligence

Part I. Implementation

Part 1 : Load and prepare your dataset

```
    # Begin your code (Part 1)
    dataset=[]

    Face=dataPath+'/face'
    for file in os.listdir(Face):
        print(file)
        img=cv2.imread(Face + '/' + file,cv2.IMREAD_GRAYSCALE)
        dataset.append((img,1))

    Face2=dataPath+'/non-face'
    for file in os.listdir(Face2):
        print(file)
        img=cv2.imread(Face2+'/'+file,cv2.IMREAD_GRAYSCALE)
        dataset.append((img,0))
    # raise NotImplementedError("To be implemented")
    # End your code (Part 1)
    return dataset

```

For files in the folder “face” and “non-face”, first make a path to the folder which these files in. Then, print each file out and load the image by the imread function in OpenCV, turn it into a grayscale image by so at the same time.
Last, label each image (1 for face and 0 for non-face) and add images to the list by the append function.

Part 2 : Implement Adaboost algorithm

```
        # Begin your code (Part 2)
        bestError = (len(iis))+1
        bestClf = []
        error = np.zeros(len(features))

        F = np.zeros((len(features),len(iis)))


        for i in range(len(features)):
            for j in range(len(iis)):
                if(featureVals[i][j] < 0):
                    F[i][j] = 1
                else:
                    F[i][j] = 0

                error[i] = error[i] + weights[j] * abs(F[i][j] - labels[j])

        fid = -1
        for i in range(len(error)):
            if error[i] <= bestError:
                bestError = error[i]
                fid = i

        bestClf = WeakClassifier(features[fid])

        # raise NotImplementedError("To be implemented")
        # End your code (Part 2)
        return bestClf, bestError
```

The classifier was applied in line 158-163 :

$$h_j(x_i) = \begin{cases} 1 & f_j(x_i) < 0 \\ 0 & else \end{cases}$$

The error of each feature is calculated in line 165 :

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

Find the least error in 167-171 :

(The initial value of best error has been set as (len(iis))+1,which is the maximum value of error)

Last, choose a best classifier in line 173, where the variable “fid” indicated the the serial number of the best feature.

Part 4 : Detect face

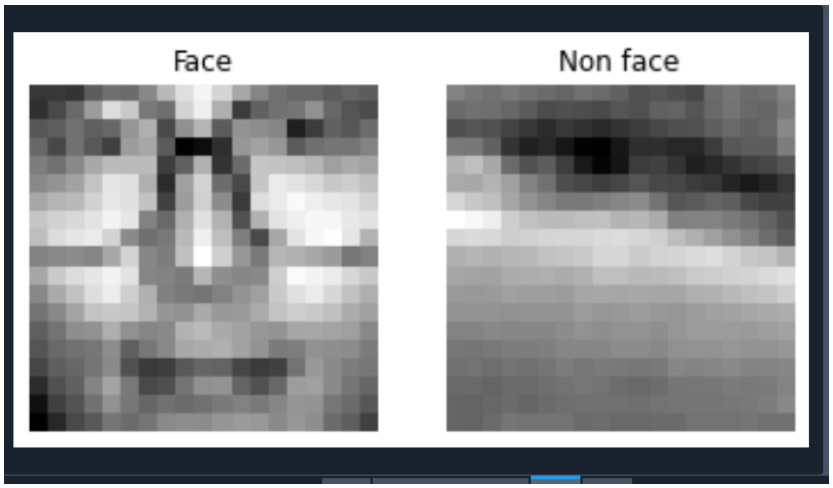
The explanation is in the picture of the annotation part.

```
    """
    # Begin your code (Part 4)

    x_coordinate = []
    y_coordinate = []
    beatles = []
    obama = []
    size = []

    ## each line in txt is an element in the list 'str1'
    ## str1 = [the-beatles.jpg 4 , 242 78 55 55 , 368 175 71 71, ...]
    F = open(dataPath)
    str1 = F.readlines()
    for line in str1:
        print(line)

    for i in range(len(str1)):

        if(i==0):
            beatles = str1[0].split(' ') ## beatles = [the-beatles.jpg , 4]

        elif(i==5):
            obama = str1[5].split(' ') ## obama = [p110912sh-0083.jpg , 15]
        else:
            str2 = []
            str2 = str1[i].split(' ') ## i.e. str2 = [242,78,55,55]
            x_coordinate.append(int(str2[0]))
            y_coordinate.append(int(str2[1]))
            size.append(int(str2[2]))
            str2 = []  ## initialize the list 'str2' and run next line in str1


image1 = 'data/detect/' + beatles[0] ## The path to the image of the beatles
image2 = 'data/detect/' + obama[0] ## The path to the image of Obama
img1 = cv2.imread(image1)
img2 = cv2.imread(image2)

for i in range(0,4): ## face0 to face3 in the beatles' image

## Turn the image to grayscale and find out the position of the face then resize it
    imagegray = cv2.imread(image1,cv2.IMREAD_GRAYSCALE)
    imagechop = imagegray[x_coordinate[i]:size[i],y_coordinate[i]:size[i]]
    imagesmall = np.resize(imagechop,(19,19))

    ## Run the classifier and draw a red or green rectangle on the faces of the original images
    if(clf.classify(imagesmall)):
        cv2.rectangle(img1,(x_coordinate[i],y_coordinate[i]),(x_coordinate[i] + size[i],y_coordinate[i] + size[i]),(0,255,0),2)
    else:
        cv2.rectangle(img1,(x_coordinate[i],y_coordinate[i]),(x_coordinate[i] + size[i],y_coordinate[i] + size[i]),(0,0,255),2)

for i in range(4,19): ## face4 to face18 in the Obama's image

     ## Turn the image to grayscale and find out the position of the face then resize it
     imagegray = cv2.imread(image2,cv2.IMREAD_GRAYSCALE)
     imagechop = imagegray[x_coordinate[i]:x_coordinate[i] + size[i],y_coordinate[i]:y_coordinate[i] + size[i]]
     imagesmall = np.resize(imagechop,(19,19))

     ## Run the classifier and draw a red or green rectangle on the faces of the original images
     if(clf.classify(imagesmall)):
        cv2.rectangle(img2,(x_coordinate[i],y_coordinate[i]),(x_coordinate[i] + size[i],y_coordinate[i] + size[i]),(0,255,0),2)
```

```
        else:
            cv2.rectangle(img2,(x_coordinate[i],y_coordinate[i]),(x_coordinate[i] + size[i],y_coordinate[i] + size[i]),(0,0,255),2)

    cv2.namedWindow("obama",0)
    cv2.resizeWindow("obama",20000,20000)
    cv2.imshow("obama",img2)
    cv2.waitKey(0)
    cv2.namedWindow("beatles",0)
    cv2.resizeWindow("beatles",20000,20000)
    cv2.imshow("beatles",img1)
    cv2.waitKey(0)
    # raise NotImplementedError("To be implemented")
    # End your code (Part 4)

```

Part II. Results & Analysis

The result of Part 1 :



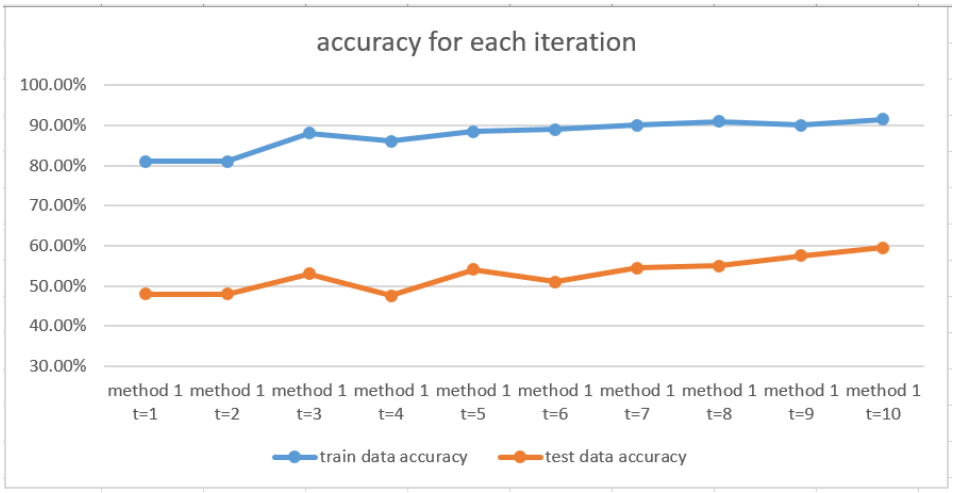

```
cmu_7102.pgm
cmu_7103.pgm
cmu_7104.pgm
cmu_7105.pgm
cmu_7106.pgm
cmu_7107.pgm
cmu_7108.pgm
cmu_7109.pgm
cmu_7110.pgm
The number of test samples loaded: 200
Show the first and last images of training dataset
```

The result of Part 2 (with T = 10) :

```
Help Variable Explorer Plots Files
Console 1/A
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(5, 2, 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

In [18]:
IPython console History
LSP Python: ready  conda: base (Python 3.9.7)  Line 168, Col 13  ASCII  LF  RW  Mem 90%
```

The result of part 3 : Chart of train data accuracy and test data accuracy with T = 1 ~ T = 10

200張	train data accuracy	test data accuracy
method 1 t=1	81.00%	48.00%
method 1 t=2	81.00%	48.00%
method 1 t=3	88.00%	53.00%
method 1 t=4	86.00%	47.50%
method 1 t=5	88.50%	54.00%
method 1 t=6	89.00%	51.00%
method 1 t=7	90.00%	54.50%
method 1 t=8	91.00%	55.00%
method 1 t=9	90.00%	57.50%
method 1 t=10	91.50%	59.50%



Both Train data accuracy and test data accuracy ascend through the iteration of T . It can be observed that both of them improve about ten percent after several times of iteration.

The result of part 4 (detection) :

The result of part 5 (My own picture) :

The result of part 6 (Bonus part):

```
150         # Begin your code (Part 2)
151         bestError = (len(iis))+1
152         bestClf = []
153         error = np.zeros(len(features))
154
155         F = np.zeros((len(features),len(iis)))
156
157
158         for i in range(len(features)):
159             for j in range(len(iis)):
160
161                 if(featureVals[i][j] < -10):
162                     F[i][j] = 1
163                 elif(featureVals[i][j] < 50):
164                     F[i][j] = 0.2
165                 else:
166                     F[i][j] = 0
167
168                 error[i] = error[i] + weights[j] * abs(F[i][j] - labels[j])
169
170         fid = -1
171         for i in range(len(error)):
172             if error[i] <= bestError:
173                 bestError = error[i]
174                 fid = i
175
176         bestClf = WeakClassifier(features[fid])
177
178         # raise NotImplementedError("To be implemented")
179         # End your code (Part 2)
180         return bestClf, bestError
```

I modified the way to choose the classifier by making the boundary more precise and giving different weights to each range of feature values. (line 158-166)

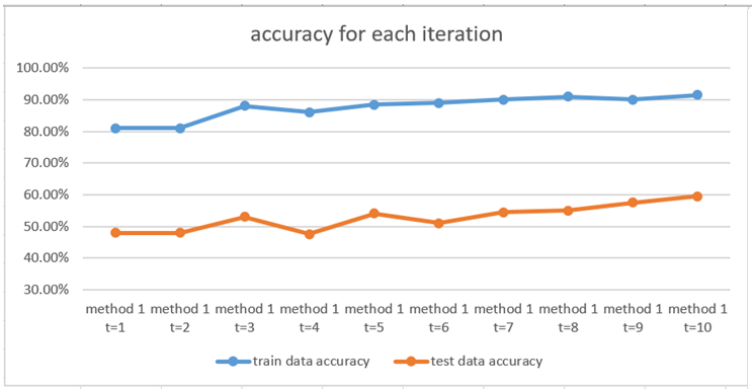
Original :

200張	train data accuracy	test data accuracy
method 1 t=1	81.00%	48.00%
method 1 t=2	81.00%	48.00%
method 1 t=3	88.00%	53.00%
method 1 t=4	86.00%	47.50%
method 1 t=5	88.50%	54.00%
method 1 t=6	89.00%	51.00%
method 1 t=7	90.00%	54.50%
method 1 t=8	91.00%	55.00%
method 1 t=9	90.00%	57.50%
method 1 t=10	91.50%	59.50%

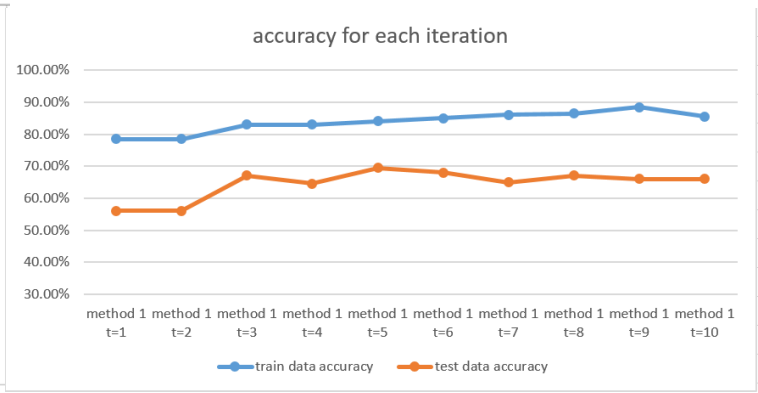
Modified :

200張	train data accuracy	test data accuracy
method 1 t=1	78.50%	56.00%
method 1 t=2	78.50%	56.00%
method 1 t=3	83.00%	67.00%
method 1 t=4	83.00%	64.50%
method 1 t=5	84.00%	69.50%
method 1 t=6	85.00%	68.00%
method 1 t=7	86.00%	65.00%
method 1 t=8	86.50%	67.00%
method 1 t=9	88.50%	66.00%
method 1 t=10	85.50%	66.00%

Original :

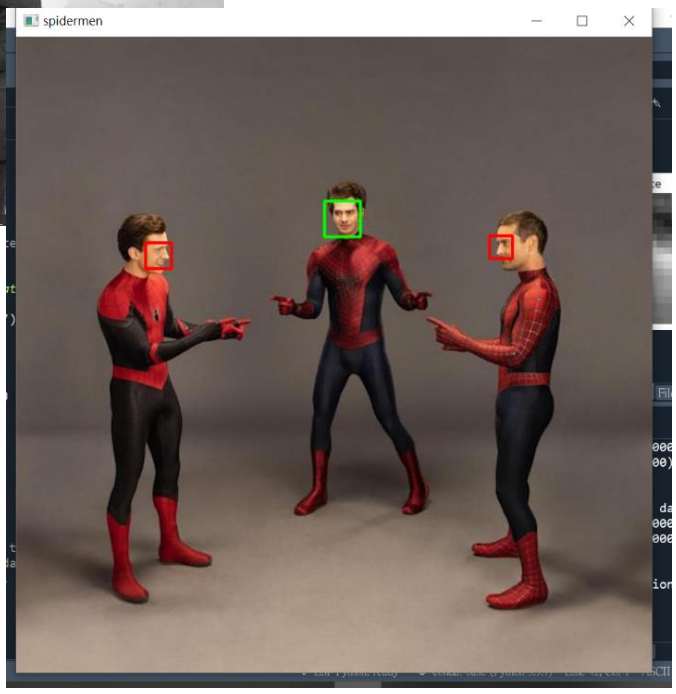


Modified :



Both of the two ways have ascending train data accuracy and test data accuracy overall. In fact, the modified one has the higher test data accuracy rate because of the more precisely way of choosing classifliер. The result of the picture detection also shows this by more green rectangles.

The result of detection (part 6)



Part III. Answer the questions

1. **Please describe a problem you encountered and how you solved it.**
   Due to the unfamiliarity of the Python's syntax, I spent much time looking for tutorial on google. One of the problem I met is I can't compile my program because of unknown syntax error that can't be solve even I check it on the net. Finally, I solve the problem with my brilliant friends' help. Much obliged for them !

2. **What are the limitations of the Viola-Jones' algorithm ?**
   Viola-Jones' algorithm are not good at detecting the non-frontal or tilted faces in the Image. According to my experiment of program, both of the spiderman on the edge of the image whose face is tilted cannot be detected, while the middle one can be detected. Thus, it's one of the limitation of the algorithm.

3. **Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T ?**
   To improve the accuracy of this algorithm, one can design a more specific way to train a program to detect faces. That is, we can modify the classifier in our algorithm, making it more conform to the actual condition. Besides checking it positive or not in the haar-like features of the integral images, we can formulate a more precise condition to do so.

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.**

   One way I came up with is to detect the color of the face in the image. For example, our eyes are half white and half black, our lips are red and our eyebrows are black. Hence, we can detect the composition and the position of the color of the image to check if it matches a human's face. The pros is that this way is more possible to detect tilted faces because the colors won't change even we see someone's face in other angle. The cons is that it may be invalid if one is in the dark, one's face is going to be single-color and hard to be detected.