HW1 Report

109550206 陳品劭 **Self link** (https://hackmd.io/@pinchen/IntroAIHW1report)

Part I. Implementation

Part 1

1. Regest the folder (face, non-face) location.
2. List all files in these floders by os.
3. Read them by cv2 in grayscale.
4. Append the info of image and its label into dataset
5. Return dataset

```
# Begin your code (Part 1)
#raise NotImplementedError("To be implemented")
dataset = []

root = str(dataPath) + "/face/"
for image in os.listdir( root ):
    dataset.append( (cv2.imread(root+image, cv2.IMREAD_GRAYSCALE), 1) )

root = str(dataPath) + "/non-face/"
for image in os.listdir( root ):
    dataset.append( (cv2.imread(root+image, cv2.IMREAD_GRAYSCALE), 0) )
    # End your code (Part 1)
```

Part 2

1. Regest the size of features and dataset.
2. Create a list with 0's (lenght = the number of features).
3. Evaluate ϵ_j for each feature.
4. Select the classifier with lowest ϵ_t.
5. Return ϵ_t and classifier.

```
# Begin your code (Part 2)
# raise NotImplementedError("To be implemented")
features_num = featureVals.shape[0]
dataset_num = featureVals.shape[1]
epsilon = np.zeros(features_num)

for j in range(features_num):
    for i in range(dataset_num):
        epsilon[j] += weights[i] *
                    abs( (1 if featureVals[j][i] < 0 else 0) - labels[i] )

bestError = epsilon[0]
bestClf = WeakClassifier(features[0])

for i in range(1, features_num):
    if epsilon[i] < bestError:
        bestClf = WeakClassifier(features[i])
        bestError = epsilon[i]
# End your code (Part 2)
```

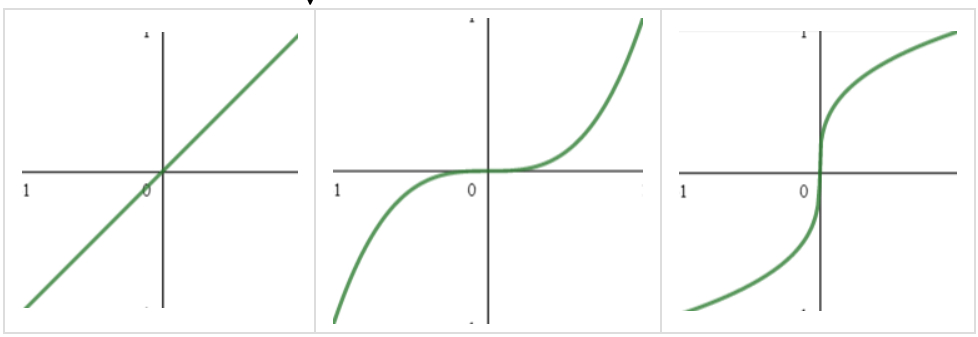
Part 4

1. Read the file (dataPath) and convert to lines in list
2. Simulate input to load data.
    - Get the file name and the number of squares.
    - Get every squares' parameter convert to tuple and append to a list.
3. Read that image with grayscale and orignal by cv2.
4. Use the grayscale's image and the parameter of squares to convert to a 19 * 19 grayscale image.
5. Use clf.classfy to check this image if face or not
6. Draw a square with red or green color with the paremeter.
7. Output this image which is draw.

```
# Begin your code (Part 4)
# raise NotImplementedError("To be implemented")
detectData = list(open(dataPath, "r"))
line = 0
while line < len(detectData):
    image_name, people_num = map(str, detectData[line].split())
    line += 1
    people = []
    for i in range( int(people_num) ):
        people.append( tuple( map( int, detectData[line].split() ) ) )
        line += 1
    image = cv2.imread("data/detect/" + image_name)
    image_cmp = cv2.imread("data/detect/" + image_name, cv2.IMREAD_GRAYSCALE)
    for face in people:
        face_image = cv2.resize(
            image_cmp[ face[1]:face[1]+face[3], face[0]:face[0]+face[2] ],
            (19, 19), interpolation=cv2.INTER_LINEAR )
        if clf.classify(face_image) == 1:
            cv2.rectangle(image, ( face[0], face[1] ),
                          ( face[0] + face[2], face[1] + face[3] ),
                          (0, 255, 0), thickness=2 )
        else:
            cv2.rectangle(image, ( face[0], face[1] ),
                          ( face[0] + face[2], face[1] + face[3] ),
                          (0, 0, 255), thickness=2 )
    cv2.imwrite("result/test/test_"+image_name, image)
# End your code (Part 4)
```

Part 6 (Bonus)

1. Let the value h be continuous in [0, 1] instead of 1 or 0.
2. Convert it into x^k (k is odd).
3. Convert it into $\sqrt[k]{x}$ (k is odd).

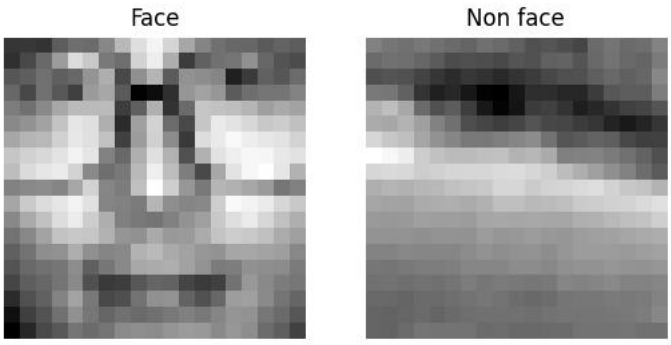



```
# Begin your code (Part 6)
# raise NotImplementedError("To be implemented")
...
    fv = sorted(featureVals[j])
    for i in range(dataset_num):
        h = abs( 1 - ( featureVals[j][i] +
                    abs(fv[0]) ) / ( fv[dataset_num - 1] - fv[0] ) )
        #h = 2 * (h - 0.5)
        #h = (h * h * h * h * h + 1) / 2
        #h = (h/abs(h)*pow(abs(h), 1/5) + 1 ) / 2
        epsilon[j] += weights[i] * abs( h - labels[i] )
...
# End your code (Part 6)
```

Part II. Results & Analysis

Part 1

We will get the following image after part 1 done.



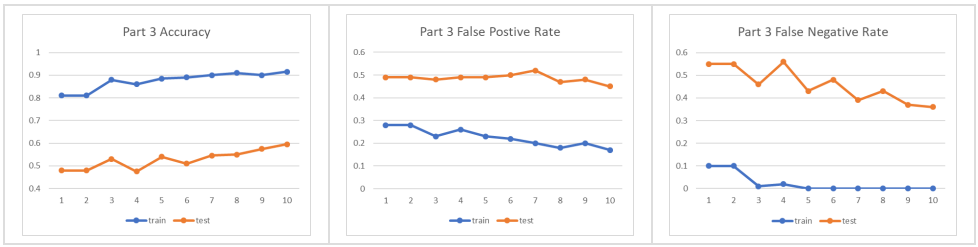
Part 2

We will get the folloing data after part 2 done.

```
[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=[Recta
ngleRegion(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

Detect faces on your own images
```

Part 3

Change t = 1 ~ 20, we will get the folloing data.



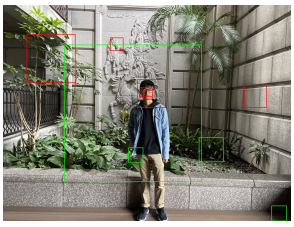
As the number of trains increases, the accuracy gradually increases, and the false negative rate decreases significantly, but the false positive rate decreases slowly relatively. We can get the better result of accuracy in train data, that is the obvious result. Because the classifier of this machine is trained from this data. However the test data can tell us whether the machine is good or bad, it can recognize faces in general images, not just limited to training data. These results of test data tell us this machine need to train more times or use a better way to choose classfier. In addition, we can also find one thing in these data this machine has a high true detection rate, but the false positive rate is also high. That means this machine has learned the relevant features of the face, but the conditions are too loose, so that many images which are not faces will also be identified as faces, but those which are faces are not easy to judge wrong.

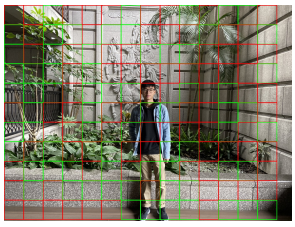
Part 4

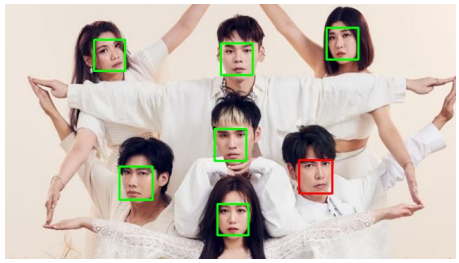
We will get the folloing images after part 4 done.

Part 5

Select the pixal point by Microsoft Paint and test them, we will get the folloing results.



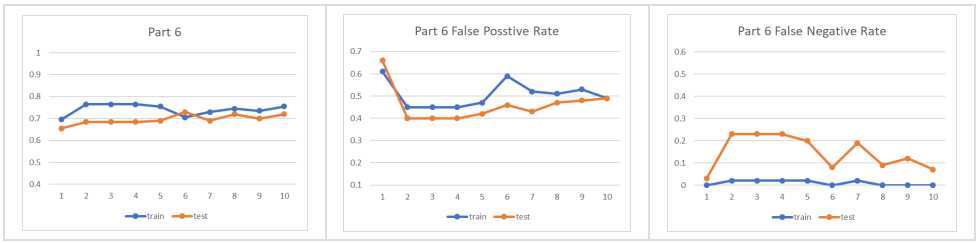




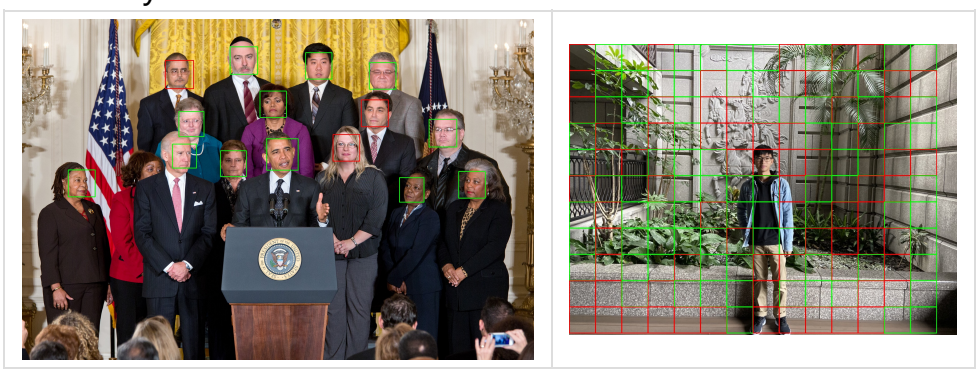
The results of these images tell us this machine can not use into general images yet, and the squares need to be near with face instead a big square contains one face and other things as the imgae in part 4. However it can be found that this machine will judge many things that are not faces as faces.

Part 6

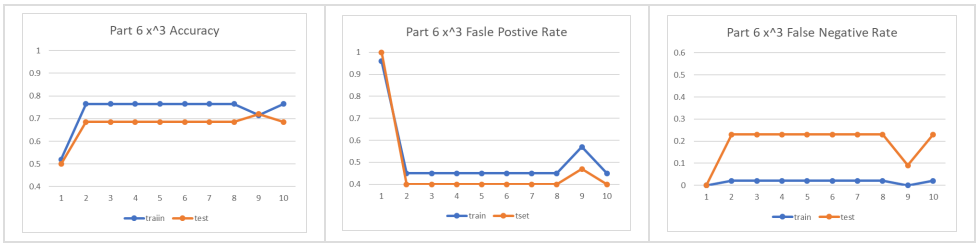
We will get the folloing data after part 6 done.



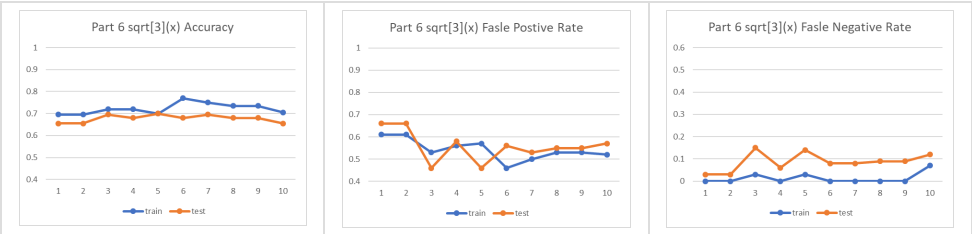
After let h be a continuous function whose domain and range in [0, 1], the accuracy of train data will be lower but the accuracy of test data be higher and close to train data. And the false Positive rate still be high. However the great thing is the false negative rate is more and more lower obviously.



Compare these photos with part 4, this machine can recognize more faces. But it still will recognize many images which is not faces as faces.



After convert it into x^k (k is odd), the classifier will quickly lock into some features and difficult to change. But that may not be best. I think this is not a good way.



After convert it into $\sqrt[k]{x}$ (k is odd), it has same problem with x^k (k is odd). But it is relatively easy to change. I think this may better than x^k, but still not a good way.
Let h be a contnious fuctions to make classifications weighted may be a good idea. That may the false negative rate be lower. But convert it into some functions may not a good idea. It will affect learning.

Part III. Answer the questions

1. Please describe a problem you encountered and how you solved it.
   - I do not use grayscale.
     Reference discussion board.
   - Part2 can not run because
     `np.linalg.norm(weights)` .
     Reference discussion board to change to
     `np.sum(weights)` .
   - I do not know how to get the paremeters of square with my image.
     Use Microsoft Paint to get the pixel point.

2. What are the limitations of the Viola-Jones' algorithm?
   - Only can be used for binary classification.
   - May be limited by the state of the image (light high or low, frontal face or not).

3. Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?
   - Construct a good way to determine the classifier to let the mathine can take more essential classifier.

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.
    - Random algorithm
      Input an image then compute an random number in range (-1, 1), if it is positive then this image is face, otherwise not face.

	Adaboost algorithm	**Random algorithm**
Accuracy	60%~80%✔️	50%
False postive rate	HiGH	HIGH
False negative rate	LOW ✔️	HIGH
Used time	SLOW	QUICK ✔️