HW 1- Report

Part I. implementation:

- Part 1

```python
# Begin your code (Part 1)
"""
define a list to contain all the (image, label) tuple.
"""
dataset = list()
for file in os.listdir(dataPath + "/face"):
    # read all face images and label 1
    img = cv2.imread(f"{dataPath}/face/{file}", cv2.IMREAD_GRAYSCALE)
    dataset.append((img, 1))
for file in os.listdir(dataPath + "/non-face"):
    # read all non-face images and label 0
    img = cv2.imread(f"{dataPath}/non-face/{file}", cv2.IMREAD_GRAYSCALE)
    dataset.append((img, 0))
# End your code (Part 1)
```

- Part 2

```python
# Begin your code (Part 2)
"""
for each feature (weak classifier), compute its error and choose the best weak classifier.
because error always <= 1.0, so best error is assigned 1.0 by default.
the feature of the best weak classifier will be modified when best error changed.
"""
bestError = 1.0
bestClf = WeakClassifier(feature=None)

for i in range(len(features)):
    # the weak classifier to be tested, and compute its error
    clf = WeakClassifier(feature=features[i])
    err = sum([weights[j] * abs(clf.classify(iis[j]) - labels[j]) for j in range(len(labels))])
    if err < bestError:
        # if this error is lower than the best one, then modify the result.
        bestClf = clf
        bestError = err
# End your code (Part 2)
```

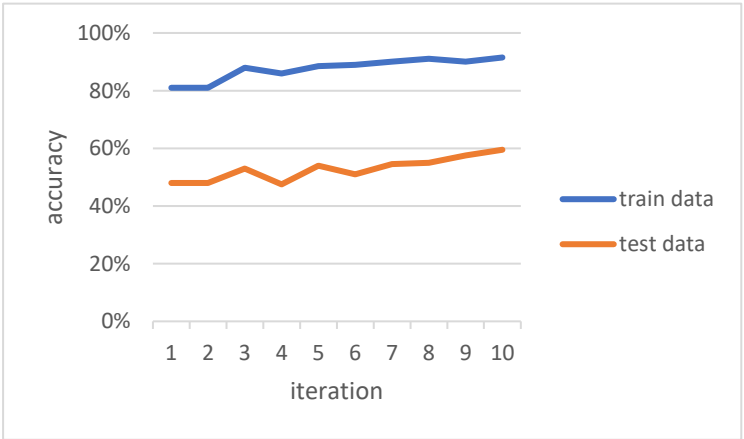
- Part 4

```
    # Begin your code (Part 4)
    """
    read in all images and the position to be detected, resize it , and classify it.
    """
    # get all data in a list, for the convenience when processing it
    data = []
    with open(dataPath) as f:
        t = f.readline()
        while t != '':
            file, count = t.split()
            # record all rectangles to be classified
            rect = []
            for i in range(int(count)):
                x, y, w, h = [int(k) for k in f.readline().split()]
                rect.append((x, y, w, h))
            # append (filename, rectangles) tuple to data list
            data.append((file, rect))
            t = f.readline()

    # show all images in one plot
    fig, axs = plt.subplots(len(data), 1)
```

```

    for i, file in enumerate(data):
        # read images and convert to gray (to classify) and BGR(for matplotlib)
        img = cv2.imread("data/detect/" + file[0])
        imgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        axs[i].axis('off')
        axs[i].imshow(imgb)
        axs[i].set_title(file[0])
        for rect in file[1]:
            x, y, w, h = rect
            # for each rectangle, get the sub image and resize it to 19x19
            sub = gray[y:y+h, x:x+w]
            sub = cv2.resize(sub, (19, 19), cv2.INTER_AREA)
            # use a patch to:
            # draw a green box if the rectangle is positively classified,
            # red box if it is negatively classified.
            if clf.classify(sub):
                patch = patches.Rectangle((x, y), w, h, edgecolor='green', facecolor='none')
            else:
                patch = patches.Rectangle((x, y), w, h, edgecolor='red', facecolor='none')
            axs[i].add_patch(patch)

    # show the results
    plt.show()
    # End your code (Part 4)
```

Part II. Results & Analysis

```
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)
```



- Because we train the classifier by the train data set, so the accuracy of train data is growing overall.
- But when this classifier overfits the train data, the accuracy of other data might not be higher.
- According to the figure above, the accuracy of train data grows with iteration count, but the accuracy of test data is not, even dropping sometimes. Thus, training a classifier that fits the train data perfectly might not be a good choice.

Part III. Answer the questions:

1. When I started working on this project, I didn't even know what a classifier is and what to do with the features. After watching the lecture replays and searching for information on the Internet, I figured out the relation between a classifier and a feature, and realized how to derive a strong classifier from some weak classifiers. Thanks to the preparation, I can finish this project without encountering any big difficulties.
2. Viola-Jones' algorithm needs a lot of pre-defined features to ensure its performance, it can't detect a part of a face, and it can't detect faces which are too close to each other.
3. Rotate the haar feature in some degree, so it can detect tilted faces more correctly.
4. Identify the edge of an item, and try to find eyes, nose, and mouth in the area. If can be found, then classify as a face.
    a. Pros: can easily detect tilted faces.
    b. Cons: may cause more complexity, can't detect when part of the face is covered, and even facial expression can cause its function failure.