Introduction to Artificial Intelligence

HW1-Report

0716325-曾正豪

Part 1.

```
16      # Begin your code (Part 1)
17
18      dataset = []
19
20
21      filelist = glob.glob(str(dataPath)+"/face/*.pgm")
22      for file_name in filelist:
23          new_element = (cv2.imread(file_name, cv2.IMREAD_GRAYSCALE), 1)
24          dataset.append(new_element)
25
26
27      filelist = glob.glob(str(dataPath)+"/non-face/*.pgm")
28      for file_name in filelist:
29          new_element = (cv2.imread(file_name, cv2.IMREAD_GRAYSCALE), 0)
30          dataset.append(new_element)
31
32
33      #raise NotImplementedError("To be implemented")
34
35      # End your code (Part 1)
```

First, I initialize the dataset as an empty list. Then, I use the “glob” function to get all the file names in the face and non-face directory. To read the images, I used OpenCV’s “imread” function, and I set the parameter to “IMREAD_GRAYSCALE” to read the image as a gray scale image. It will return a NumPy 2D-array, its values are the gray scale values of each pixels. After that, I packed the NumPy 2D-array and the label into a tuple, and append it to the dataset list.

Part 2.

```
 9 def classifier(x):
10     return 1 if x < 0 else 0
11     #return 1 - .5 * (1 + np.tanh(.5 * x))
```

```
155         # Begin your code (Part 2)
156         data_num = featureVals.shape[1]
157         feature_num = featureVals.shape[0]
158         errors = np.zeros(feature_num)
159 
160         for i in range(feature_num):
161             for j in range(data_num):
162                 h = classifier(featureVals[i][j])
163                 errors[i] += weights[j] * abs(h - labels[j])
164 
165         bestError = errors[0]
166         lowest_error_index = 0
167         for i in range(feature_num):
168             if errors[i] < bestError:
169                 lowest_error_index = i
170                 bestError = errors[i]
171         bestClf = WeakClassifier(features[lowest_error_index])
172 
173         #raise NotImplementedError("To be implemented")
174         # End your code (Part 2)
```

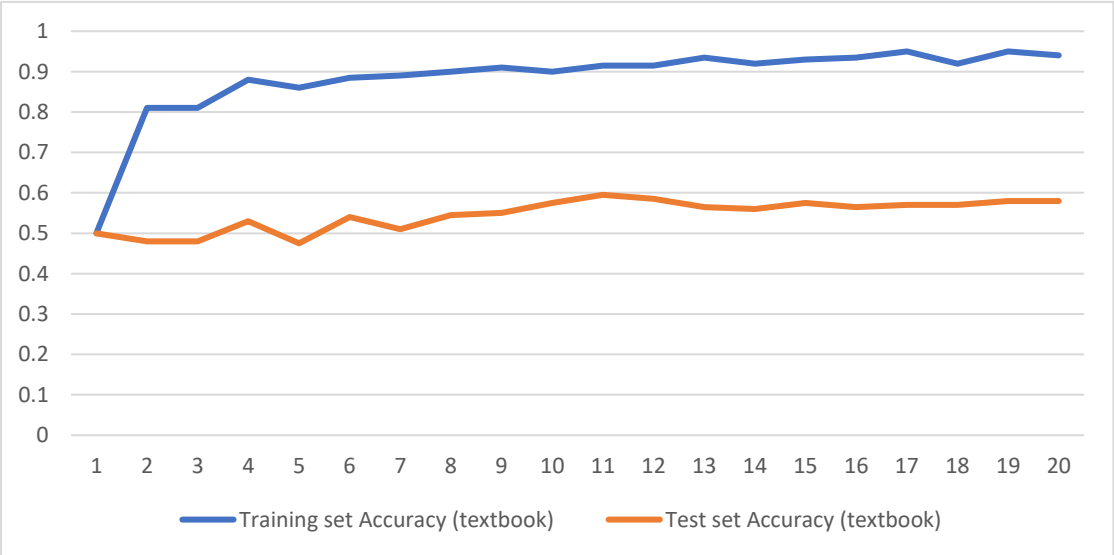
In the “selectBest” function, I get the numbers of data and feature first. Then, I create a 1D-array called “errors” to store the error values of each features. In the first loops, I iterate all the “featureVals” to calculate the error values of the features base on each images. Here, I use the same way as the paper done. That is, first calculate the “h”, if featureVals[i][j] is less than 0, h would be 1, otherwise 0. Then it will calculate the distance to labels and times the weight. Finally, added it to the error array.

In the second round of the loop, I will iterate all the elements in “errors” to find the lowest error value. After I get the index of the best classifier, I will create its “WeakClassifier Class”.

Part 3.

Here, I test the hyperparameter “T” from 1 to 20, and I got the following results.

Training set Accuracy	Test set Accuracy
0.5	0.5
0.81	0.48
0.81	0.48
0.88	0.53
0.86	0.475
0.885	0.54
0.89	0.51
0.9	0.545
0.91	0.55
0.9	0.575
0.915	0.595
0.915	0.585
0.935	0.565
0.92	0.56
0.93	0.575
0.935	0.565
0.95	0.57
0.92	0.57
0.95	0.58
0.94	0.58



Part 4.

```
    # Begin your code (Part 4)
    f = open(dataPath, "r")
    file_list = []

    for line in f:
        spilt_str = line.split(' ')
        if len(spilt_str) == 2:
            file_list.append((spilt_str[0], []))
        elif len(spilt_str) == 4:
            file_list[-1][1].append((int(spilt_str[0]), int(spilt_str[1]), int(spilt_str[2]), int(spilt_str[3])))

    for file in file_list:
        img_original = cv2.imread("data/detect/" + file[0])
        img = cv2.imread("data/detect/" + file[0], cv2.IMREAD_GRAYSCALE)
        for face_area in file[1]:
            face = img[face_area[1]:face_area[1]+face_area[3], face_area[0]:face_area[0]+face_area[2]]
            face = cv2.resize(face, (19, 19), interpolation=cv2.INTER_LINEAR)
            result = clf.classify(face)
            if result == 1:
                cv2.rectangle(img_original, (face_area[0], face_area[1]), (face_area[0] + face_area[2], face_area[1]
+ face_area[3]), (0, 255, 0), thickness=4)
            else:
                cv2.rectangle(img_original, (face_area[0], face_area[1]), (face_area[0] + face_area[2], face_area[1]
+ face_area[3]), (0, 0, 255), thickness=4)
        img_original = cv2.cvtColor(img_original, cv2.COLOR_BGR2RGB)
        plt.axis('off')
        plt.imshow(img_original)
        plt.show()
    # End your code (Part 4)
```

First, I open the file in “dataPath” and create a list called “file_list”. It is a list containing tuples. A tuple consists of the image file name and a list of face locations. Then, I would iterate all the lines in the file. I use “split” to split the space. Then, if the length is 2, means that it is a file name. So I append the file name to the file_list. If the length of that line is 4, I will append the location to the list in last tuple.

After read the file names, I will iterate all the file in the ”file_list”. In each iteration, I would read the image twice. The first is in RGB, to draw the result rectangle and display it. The second is in gray scale, to feed it into the classifier. After that I would iterate all the face location. In each iteration, I would take all the pixels in that area and resize it to 19x19. After that, I would feed the image into the trained classifier. If the result is 1, I will draw a green rectangle on the face (on the RGB image), otherwise, a red rectangle. After iterate all the face location, I would use “matplotlib” to draw the image on the Jupyter_notebook.

Part 5.

```
In [16]: print('\nDetect faces on your own images')
         detection.detect('data/detect/my_image.txt', clf)
```

In my_image.txt:

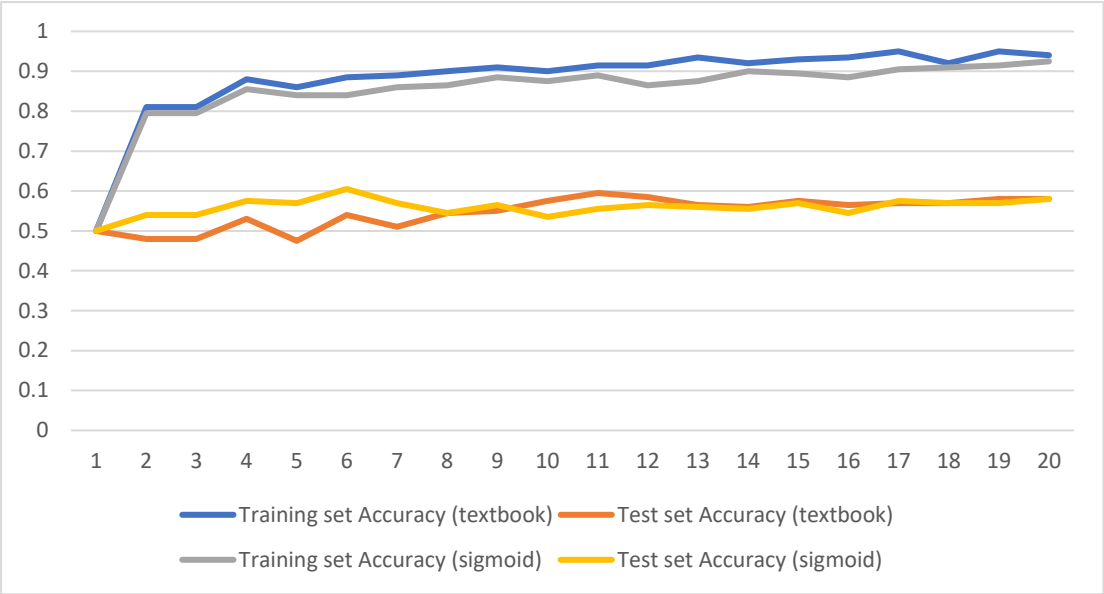
IMAG1063.jpg 5
130 700 300 300
1300 700 400 400
150 300 500 500
1500 700 500 500
50 50 1000 1000

Part 6.

```
9  def classifier(x):
10     return 1 if x < 0 else 0
11     #return 1 - .5 * (1 + np.tanh(.5 * x))
```

In the “adaboost.py”, I implement different classifier functions, sigmoid function. Originally, I use this form:$\sigma(x) = \frac{1}{1+e^{-x}}$, however, it occurred the overflow warnings, so I change the form to $\sigma(x) = \frac{1}{2} + \frac{1}{2}\tanh\frac{x}{2}$.

Training set Accuracy (textbook)	Test set Accuracy (textbook)	Training set Accuracy (sigmoid)	Test set Accuracy (sigmoid)
0.5	0.5	0.5	0.5
0.81	0.48	0.795	0.54
0.81	0.48	0.795	0.54
0.88	0.53	0.855	0.575
0.86	0.475	0.84	0.57
0.885	0.54	0.84	0.605
0.89	0.51	0.86	0.57
0.9	0.545	0.865	0.545
0.91	0.55	0.885	0.565
0.9	0.575	0.875	0.535
0.915	0.595	0.89	0.555
0.915	0.585	0.865	0.565
0.935	0.565	0.875	0.56
0.92	0.56	0.9	0.555
0.93	0.575	0.895	0.57
0.935	0.565	0.885	0.545
0.95	0.57	0.905	0.575
0.92	0.57	0.91	0.57
0.95	0.58	0.915	0.57
0.94	0.58	0.925	0.58


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Training set Accuracy (textbook)
Test set Accuracy (textbook)
Training set Accuracy (sigmoid)
Test set Accuracy (sigmoid)