109550069 林宇柔

HW1 Report

I. Implementation

Part 1

```
    """
    1) In the directory of parameter dataPath, check further directories, there should be
    one storing images with faces and the other with non-faces.
    2) For every file in the face folder, load it with gray-scale mode, and store its
    information with classification 1 (which stands for being a face).
    3) Do the same thing as the previous step for the non-face folder.
    4) Return a tuple list "dataset" whose elements are a numpy array representing the image and
    and the classification of it, indicating if it is a face or not.
    """
    # load all images in folder and make list of tuples
    dataset = []
    for file in os.listdir(dataPath):
        if file == 'face':
            face_path = os.path.join(dataPath, file)
            for face in os.listdir(face_path):
                img = cv2.imread(os.path.join(face_path, face), cv2.IMREAD_GRAYSCALE)
                t = img, 1
                dataset.append(t)
        if file == 'non-face':
            n_face_path = os.path.join(dataPath, file)
            for n_face in os.listdir(n_face_path):
                img = cv2.imread(os.path.join(n_face_path, n_face), cv2.IMREAD_GRAYSCALE)
                t = img, 0
                dataset.append(t)

    return dataset
```

Part 2

```
    # Begin your code (Part 2)
    """
    1) Create a 2D list "faceOrNot" with shape (len(features), len(dataset)), set the value
    to 1 if the corresponding "featureVal" value is smaller than 0, otherwise set the value
    to 0.
    2) Create a list "errors" to record the errors of each feature (for each, use the weight times
    the difference between "featureVal" value and the label of them, then sum them up).
    3) Find the smallest error, record its value and the index to know which feature we want.
    4) If the value of "faceOrNot" of the feature is equal to the original label of the image,
    update the weight (the original weight times the error and divide by 1 minus the error).
    5) Return the error and the best WeakClassifier class with parameter being the feature with
    minimum error.
    """
    # identify face or not
    faceOrNot = np.zeros((len(features), featureVals.shape[1]))
    for val in range(featureVals.shape[0]):
        for data in range(featureVals.shape[1]):
            if featureVals[val][data] < 0:
                faceOrNot[val][data] = 1
            else:
                faceOrNot[val][data] = 0
```

```
171     # record all errors
172     errors = []
173     for val in range(featureVals.shape[0]):
174         e = 0
175         for data in range(featureVals.shape[1]):
176             e += (weights[data] * abs(faceOrNot[val][data] - labels[data]))
177         errors.append(e)
178     # find the smallest error
179     error = min(errors)
180     for i in range(len(errors)):
181         if errors[i] == error:
182             index = i
183             break
184     # update the weights
185     for i in range(len(weights)):
186         if faceOrNot[index][i] == labels[i]:
187             weights[i] = weights[i] * (error / (1 - error))
188     # the final strong classifier
189     bestClf = WeakClassifier(features[index])
190     bestError = error
191     # raise NotImplementedError("To be implemented")
192     # End your code (Part 2)
193     return bestClf, bestError
```

Part 4

```
17      # Begin your code (Part 4)
18      """
19      1) Read the txt file, and load the image
20      2) By the given number, store the left-up coordinate and the width and the height
21      3) Crop the image by the coordinates and convert it into a gray-scale image
22      4) Use the "classify" function to test the cropped image, if the function returns 1,
23      it shall be a face therefore box it with a green rectangle, otherwise red.
24      5) Check the image with imshow
25      """
26      # read the txt file
27      file = open(dataPath, 'r')
28      for line in file:
29          img = cv2.imread(os.path.join('data/detect', line.split()[0]))
30          for i in range(int(line.split()[1])):
31              data = file.readline().split()
32              x = int(data[0])
33              y = int(data[1])
34              w = int(data[2])
35              h = int(data[3])
36              # crop the faces
37              crop = img[y:y+h, x:x+w]
38              crop = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
39              crop = cv2.resize(crop, (19, 19), interpolation=cv2.INTER_AREA)
40              if clf.classify(crop) == 1:
41                  cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
42              else:
43                  cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
44          cv2.imshow('With Box', img)
45          cv2.waitKey(0)
46      # raise NotImplementedError("To be implemented")
47      # End your code (Part 4)
```

II. Results and Analysis

```
The number of training samples loaded: 200
The number of test samples loaded: 200
Show the first and last images of training dataset
Computing integral images
Building features
Applying features to dataset
Selecting best features
Selected 5171 potential features
Initialize weights
Run No. of Iteration: 1
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), RectangleRegion(7, 3, 1, 3)], negative regions=[RectangleRegio
with accuracy: 162.000000 and alpha: 1.450010
Run No. of Iteration: 2
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 8, 2, 9)], negative regions=[RectangleRegion(2, 8, 2, 9)])
with accuracy: 156.000000 and alpha: 1.308437
Run No. of Iteration: 3
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 1, 2, 13)], negative regions=[RectangleRegion(8, 1, 2, 13)])
with accuracy: 150.000000 and alpha: 1.305683
Run No. of Iteration: 4
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(5, 1, 4, 5), RectangleRegion(1, 6, 4, 5)], negative regions=[RectangleRegio
with accuracy: 65.000000 and alpha: 1.524321
Run No. of Iteration: 5
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 0, 4, 6), RectangleRegion(6, 6, 4, 6)], negative regions=[RectangleRegi
with accuracy: 149.000000 and alpha: 1.802505
Run No. of Iteration: 6
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 2, 2, 5)], negative regions=[RectangleRegion(5, 2, 2, 5)])
with accuracy: 64.000000 and alpha: 1.246900
Run No. of Iteration: 7
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 3, 6, 2)], negative regions=[RectangleRegion(4, 5, 6, 2)])
with accuracy: 153.000000 and alpha: 1.917649
Run No. of Iteration: 8
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(16, 17, 1, 1)], negative regions=[RectangleRegion(15, 17, 1, 1)])
with accuracy: 156.000000 and alpha: 1.680572
Run No. of Iteration: 9
Chose classifier:
Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9, 3, 2)], negative regions=[RectangleRegion(1, 9, 3, 2)])
with accuracy: 149.000000 and alpha: 2.086934
Run No. of Iteration: 10
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: 1.389306


Evaluate your classifier with training dataset
False Positive Rate: 27/100 (0.270000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 173/200 (0.865000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 19/100 (0.190000)
Accuracy: 132/200 (0.660000)

Detect faces at the assigned location using your classifier

Process finished with exit code 0
```

(Photos in Part 4)

(Photo in Part 5, path = data/detect2)

Performance difference between the training and data set：
The training data always performs better than the test data.

200 images	train data accuracy	test data accuracy
T = 1	(162/200) 81%	(96/200) 48%
T = 2	(162/200) 81%	(96/200) 48%
T = 3	(166/200) 83%	(121/200) 60.5%
T = 4	(153/200) 76.5%	(85/200) 42.5%
T = 5	(165/200) 82.5%	(124/200) 62%
T = 6	(160/200) 80%	(125/200) 62.5%
T = 7	(172/200) 86%	(117/200) 58.5%
T = 8	(172/200) 86%	(134/200) 67%
T = 9	(175/200) 87.5%	(130/200) 65%
T = 10	(173/200) 86.5%	(132/200) 66%

III. Questions

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

There are more than just one problem I encountered:

1) I've never coded in Python before this homework, so I'm not familiar with the syntax of it. While I was doing this assignment, I often used lists to store my data. In Python, you don't have to declare the length of the list before you use it, I kept forget that and when I wanted to add an element to the list, I wrote something like "arr[i] = value", so the debugger kept showing warnings saying the index is out of range. I changed that into "arr.append(value)" and the problem was solved. (However I figured out it was just like vector.push_back() in C++)
2) While running detection.py, the images didn't show up, I kept checking if there was anything wrong with my code and couldn't figure it out. Then I found that it the window was minimized on the dock of my computer. Problem solved.

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

1) It is very slow to train with all the images.
2) It only works best when the face is frontal–view.

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

Decrease the weight for a classifier that has high probability of false–positive errors.

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.

–> Neural Network–Based Face Detection

Pros compared to Adaboost:

A. More fault tolerant
B. More suited for real time operation due to high computational rates
C. Learns by example

Cons compared to Adaboost:

A. Also only detects upright faces looking at the camera