CSC320 Assignment3 Winter 2015

Soon Chee Loong

March 20, 2015

Contents

1 Introduction 1

2 Executing Code 2

3 Input Images 2

4 Part 1 2

5 Part 2 3

6 Part 3 4

7 Part 4 7

1 Introduction

This paper is where I present my report for Assignment3 of CSC320 Winter 2015. There is no starter code for this assignment. However, there is a sample Principal Component Analysis code for us to refer to.

Also, the images in this report has been re-scaled to fit the report. Original sizes can be viewed by opening the images themselves directly. The pictures have also been resized to a 32 by 32, converted to grayscale and cropped for their faces according to the annotations of the dataset to enable the code to run faster.

2 Executing Code

I have included 4 different code, one for each part of the assignment. You also have to ensure the directories for storing the output images are created before running the code. The images that I worked with have been submitted as the croppedPngImages.zip file which can be extracted and used to run part2.py, part3.py, and part4.py. This file can also be obtained by running part1.py. The code can be compiled with Python 2.7.5 with Anaconda's Full installation. The files provided for each parts are part1.py, part2.py, part3.py, and part4.py. Note: p3.py is exactly the same as part4.py.

3 Input Images

The input images will be from the FaceScrub dataset. The dataset consists of URLs to images with faces, as well as the bounding boxes of the faces. The dataset that I work with is the $faces\_subset.txt$ file.

4 Part 1

Note: The code for part 1 is in part1.py

The dataset of faces is a subset of all the faces from the FaceScrub dataset. In particular, the dataset of faces contains 8 different people, 3 of whom are male. Figure 1 shows example of images fetch directly from the URL that was given in the dataset.

The quality of the annotation of bounding boxes provided by the dataset is bad for about 25% of the originally downloaded pictures as shown in Figure 2. Figure 2 demonstrates the result of terrible cropped annotation for the faces in Figure 1. I have manually removed these type of images from my croppedPngImages folder as it results in about 5% boost in output correctness for Part 3 and only a 1% boost in performance for Part 4.

Figure 1: Faces Directly Fetched From URL

However, about 75% of the original downloaded pictures have been given high quality bounding boxes as shown in Figure 3. Figure 3 also demonstrates the faces of the 8 different people involved.

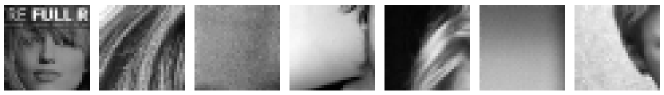

Figure 2: Examples of Terribly Cropped Faces

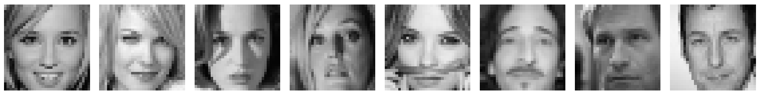

Figure 3: Examples of Well Cropped Faces

The cropped faces cannot be aligned with each other as the faces may be looking towards different directions. This is shown in Figure 4 whereby Sandler (the person) is looking towards different directions. Both these pictures cannot be aligned as he is not looking towards the same direction.

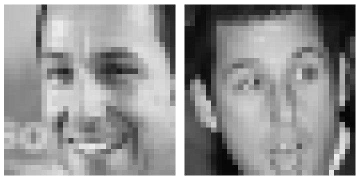

Figure 4: Example of Faces Viewing in Different Direction

5 Part 2

Note: The code for part 2 is in part2.py

The dataset was separated into 3 non-overlapping parts. The training set consists of 800 images whereby there are 100 face images per actor. The validation set consists of 80 images whereby there are 10 face images per actor. The test set consists of 80 images whereby there are 10 face images per actor.

To do this, I only read in 120 images for each faces from the dataset as I only need a maximum of 120 images from each face from the dataset. To illustrate, I need 100 sandler's images for the training set, 10 of his images for the validation set, and 10 of his images for the test set which sums up to 120.

Then, I used the first 100 images for the training set, next 10 images for the test set and finally the last 10 images for the validation set.

These numbers are easy to work with as I can easily identify the person whom the images refer to. For example, the first 100 images in my training set refers to Eckhart, whereas the next 100 images in my training set refers to Sandler. Similarly, the first 10 images in both my validation and test set refers to Eckhart, where as the next 10 images in both my validation and test set refers to Sandler.

The average face and the eigenface was computed from all 800 faces in the training set. The average face is shown in Figure 5. The top 25 eigenfaces are shown in Figure 6.

As the 8 different actors to use as recommended by the professor consists of 500 female pictures and 300 male pictures, the average face appears to look more of a female picture.

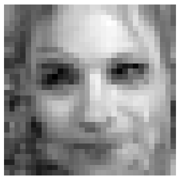

Figure 5: Average Face among the 800 training set faces

6 Part 3

Note: The code for part 3 is in part3.py. You will also need to include images from Part2's croppedPngImages folder into this part to be able to read images from that folder.

Part3.py demonstrates code to recognize every face in a set of faces using the top k eigenfaces. The accuracy on the validation set using k = 2, 5, 10, 20, 50, 80, 100, 150, 200 is shown in Table 1.

Then, the k that produces the best recognition performance on the validation is used for the test set. In this case, the best value of k was k = 150 as shown in Table 1.

The performance that was achieved on the test set is at 61.25% accuracy in which 49 test images were correctly verified for their faces whereas 31 images were incorrectly verified for their faces.

The code output is shown below

```
K:2 numCorrect: 17.0 numWrong: 63.0 accuracy: 21.25
K:5 numCorrect: 28.0 numWrong: 52.0 accuracy: 35.0
```

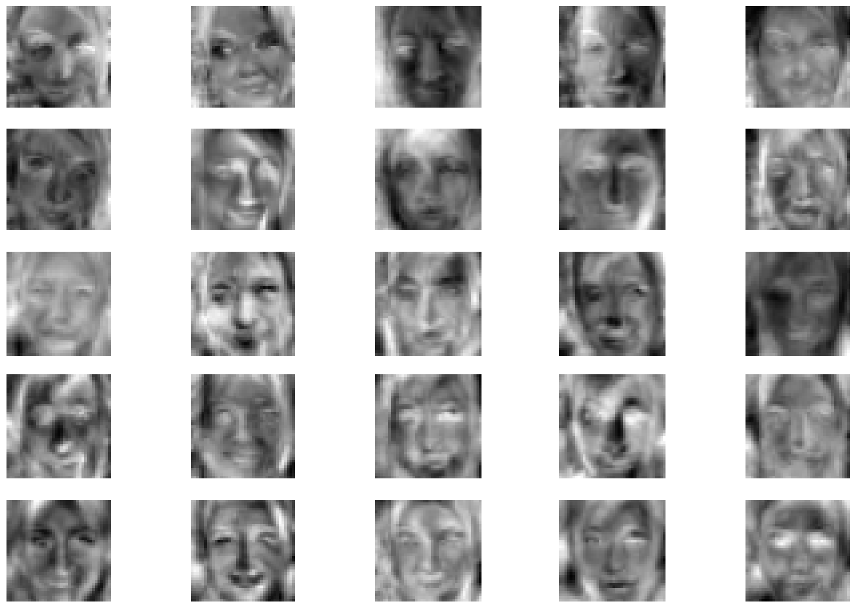
Figure 6: The top 25 eigenfaces from the training set

```
K:10 numCorrect: 29.0 numWrong: 51.0 accuracy: 36.25
K:20 numCorrect: 33.0 numWrong: 47.0 accuracy: 41.25
K:50 numCorrect: 35.0 numWrong: 45.0 accuracy: 43.75
K:80 numCorrect: 38.0 numWrong: 42.0 accuracy: 47.5
K:100 numCorrect: 38.0 numWrong: 42.0 accuracy: 47.5
K:150 numCorrect: 39.0 numWrong: 41.0 accuracy: 48.75
K:200 numCorrect: 39.0 numWrong: 41.0 accuracy: 48.75
Maximum K is : 150
K:150 numCorrect: 49.0 numWrong: 31.0 accuracy: 61.25
```

The 5 failure test cases are shown in Figure 7. The corresponding train faces that are detected to be closest to these failure cases are shown in Figure 8. To explain why, take note of their corresponding eigenfaces. Figure 9 shows the eigenfaces that are generated for the testFaces in Figure 7 whereas Figure 10 shows the eigenfaces that are generated for the trainFaces in Figure

K	NumCorrect	NumWrong	Accuracy %
2	17	63	21.25
5	28	52	35.00
10	29	51	36.25
20	33	47	41.25
50	35	45	43.75
80	38	42	47.50
100	38	42	47.50
150	39	41	48.75
200	39	41	48.75

Table 1: Performance of Face Recognition on Validation Set based on Value of K

8. Although the testFaces and the trainFaces do not look similar, their corresponding eigenFaces looks similar. Also, the testFaces and trainFaces tend to be facing towards the same direction.

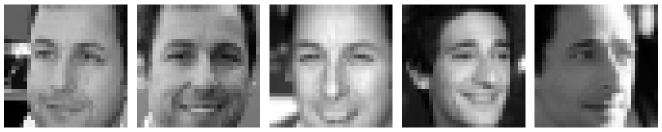

Figure 7: TestFaces

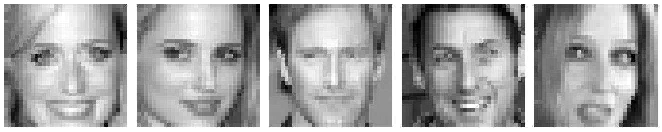

Figure 8: TrainFaces

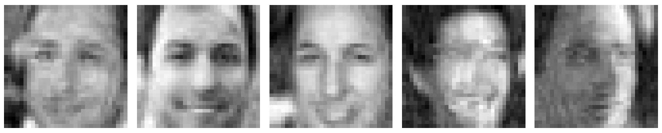

Figure 9: EigenTestFaces

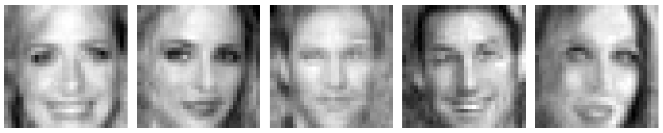

Figure 10: EigenTrainFaces

7 Part 4

Note: The code for part4 is in part4.py. You will also need to include images from Part2's croppedPngImages folder into this part to be able to read images from that folder.

Part4.py demonstrates code to recognize every person's gender in a set of faces using the top k eigenfaces. The accuracy on the validation set using k = 2, 5, 10, 20, 50, 80, 100, 150, 200 is shown in Table 2.

Then, the k that produces the best recognition performance on the validation is used for the test set. In this case, the best value of k was k = 80 as shown in Table 2.

The performance that was achieved on the test set is at 83.75% accuracy in which 67 test images were correctly verified for their faces whereas 13 images were incorrectly verified for their faces.

K	NumCorrect	NumWrong	Accuracy %
2	46	34	5.50
5	56	24	70.00
10	61	19	76.25
20	64	16	80.00
50	64	16	80.00
80	65	15	81.25
100	64	16	80.00
150	64	16	80.00
200	65	15	81.25

Table 2: Performance of Gender Recognition on Validation Set based on Value of K

The code output is shown below

```
K:2 numCorrect: 46.0 numWrong: 34.0 accuracy: 57.5
K:5 numCorrect: 56.0 numWrong: 24.0 accuracy: 70.0
K:10 numCorrect: 61.0 numWrong: 19.0 accuracy: 76.25
```

```
K:20 numCorrect: 64.0 numWrong: 16.0 accuracy: 80.0
K:50 numCorrect: 64.0 numWrong: 16.0 accuracy: 80.0
K:80 numCorrect: 65.0 numWrong: 15.0 accuracy: 81.25
K:100 numCorrect: 64.0 numWrong: 16.0 accuracy: 80.0
K:150 numCorrect: 64.0 numWrong: 16.0 accuracy: 80.0
K:200 numCorrect: 65.0 numWrong: 15.0 accuracy: 81.25
Maximum K is : 80
K:80 numCorrect: 67.0 numWrong: 13.0 accuracy: 83.75
```

The 5 failure test cases are shown in Figure 11. The corresponding train faces that are detected to be closest to these failure cases are shown in Figure 12. To explain why, take note of their corresponding eigenfaces. Figure 13 shows the eigenfaces that are generated for the testFaces in Figure 11 whereas Figure 14 shows the eigenfaces that are generated for the trainFaces in Figure 12. Although the testFaces and the trainFaces do not look similar, their corresponding eigenFaces looks very similar in this case.

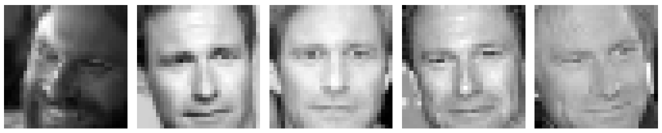
Figure 11: TestFaces

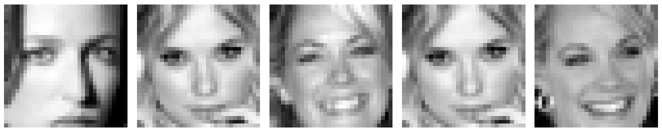
Figure 12: TrainFaces

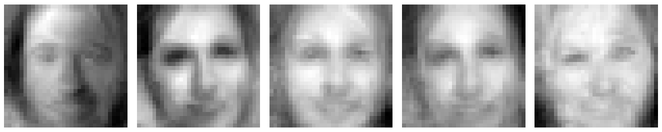
Figure 13: EigenTestFaces

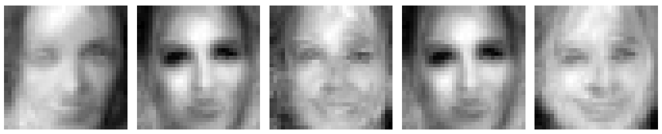

Figure 14: EigenTrainFaces