CV Assignment1 Report

406410035 秦紫頤

Method Description

Seeing the whole problem as 3 subproblems.

hw1to1.m

Getting the eigenfaces when d=1,5,9. Before running the code you need to specify your train image folder in variable "**traindir**". After this, you can execute this code file. The mechanism of my code is as follow:

1. Open train image folder
    a. locate the directory of your train image data
    b. check if the directory is a correct path
    c. open folder
    d. How many images are in the train folder
2. Read train images and generate image matrix (19800x1300)
    a. Get the filename (complete path) of each image in the train folder
    b. Read the images
    c. Convert the color image to grayscale
    d. scale the value of each pixel in the image to double datatype
    e. turn every image into 1d vector (165x120 -> 19800x1)
    f. concatenate all the vector into an image matrix (19800x1300)
3. Find mean face (19800x1)
4. Subtract the mean face
5. Calculate the covariance matrix
6. Find eigenvalues and eigenvector
7. Reshape the vector back to image size
    a. Split 19800x9 into 9 vectors each size of 19800x1
    b. Reshape the vectors to 9 matrices of size 165x120 (image size)
8. Convert matrix back to grayscale images (eigenfaces)
    a. matrix value -> grayscale image value
    b. Save the 1,5 and 9 as an image (**Save in the folder "eigfaces"**)
9. Save the eigenfaces and mean face to csv file (**eigenface.csv, original_mean.csv)**

hw1to2.m

Reduce the dimension of the test set using the eigenfaces.

Note that before executing this program you need to specify the test dataset folder path in variable "**testdatapath**". After this, you can execute this code file. The mechanism of my code is as follow:

1. Get eigenface and original mean face from the csv file (**eigface.csv,original_mean.csv)**
2. Get 1 random photo from each person in test images
    a. open the folder of test images and read the images
    b. I use random in my code to choose the images. Also, the chosen images will be saved in the folder **"selected_test_images"**. There will be 100 images inside.
3. Preprocess the selected test images (100 images)
    a. turn color image to grayscale image
    b. use 1 vector to represent each image: 165x120 -> 19800x1
    c. concatenate all the vector to 1 matrix -> 19800x100
4. Compress and decompress the selected images using d=1,5,9
    a. compress the image -> compress=original_image*eigenface(d=?)
    b. decompress the image -> decompres=compress*inverseofeigenface(d=?)
    c. reshape the image matrix (19800x1 -> 165x120) and turn into grayscale image and save the image
    d. The decompressed image will all be saved in the folder "after_reduce_dimension". And if using 1 dimension to compress the image the decompressed image will be saved in d1 folder. And if using 5 dimensions to compress the image the decompressed image will be saved in d5. The same method works for d9. And d1, d5, d9 each contains 100 images. From the first person to the one-hundredth person.

hw2.m

Compute the error rate of images after doing PCA between the original images. Note that before executing this code file you need to set your test dataset folder path in variable **"testpath"**. After doing so you can execute this program. The mechanism of my code is as follow:

1. Open the folder of the test images and form the image matrix
    a. Open the folder and read the image inside
    b. convert rgb to grayscale
    c. convert to double precision
    d. concatenate to image matrix (19800x1300)
2. Open the folder of train images and form the image matrix
3. Calculate d=1 image error rate
    a. Compress the train images (d=1)
    b. Decompress the image
    c. Calculate the total error
    d. Error rate=total error/total image
4. Repeat **Step3** with d=5,9
5. Write error rate to csv file
    a. Transform matrix to table
    b. Add header for csv file

c. Write to csv **(error_rate.csv)**

Experimental Result

hw1to1.m

The result of this problem is 9 images (**d1.bmp,d2.bmp...d9.bmp**). They are all stored in folder **"./eigfaces"**.

hw1to2.m

The result of this problem will all be saved in the folder **"after_reduce_dimension"**. And inside this folder, there are 3 subfolders

1. d1: This folder has 100 images which have done PCA using only **1** dimension. Each image is a person's random selected from test images
2. d5: This folder has 100 images which have done PCA using **5** dimensions
3. d9: This folder has 100 images which have done PCA using **9** dimensions

hw2.m

The result of this problem is in the csv file **"error_rate.csv".** Inside the file will have 3 rows. The first row is the error rate of d=1 and the second is d=5 and the third is d=9.

Discussion

hw1to1.m

While doing this first problem I found out that there are two methods to find eigenvalues and eigenvectors. The first one is using **eig()** and the second one is using **svds()**. At first, I try eig() but after a few seconds, my computer told me that it is out of memory. I think my RAM isn't enough (8GB). So I try svds() instead. I have searched online that svds()'s precision isn't as good as eig() but it will still work. And as for this assignment, I think it is ok to use svds().

hw1to2.m

I think it is quite amazing to look at the picture after doing PCA. At first, when using d=1 every output decompresses image looks the same. I was a bit shock cause I think this shouldn't be like this. But from the theory we learned in class, we only use one dimension to represent 19800 dimensions so we can't blame the result. And when using 5 dimension and 9 dimensions I can see the difference between the images and they look more similar to the original images.

hw2.m

I still think the step of doing this problem is quite weird. The error rate is to calculate the error between the original image and the image applying PCA. So I think if change the steps to

1. Applying PCA to all the test set images (d=1,5,9)

2. Calculate the error rate between the original test set image and image after applying PCA. From this, we will know whether this compress dimension is ok or the error rate is too high, not suitable to represent the images.

Problem and Difficulties

There definitely have some difficulties while doing this assignment. I thought my computer is powerful enough. But after doing this assignment I think I should buy one more 8GB of RAM. Although I can execute all the code but sometimes it runs too slow even got stuck in the middle of the process. Also, I couldn't use eig() to calculate eigenvalues and eigenvectors. So there will be some error between svds() and real eig(). The other diffiluty for me is the clue on the SPEC isn't enough. I wish to have more specific instruction. How many output and output to where? What is the output format? I think is there are more clues I can finish this assignment more smoothly.