CS-663 Assignment 4 Q6

Soham Naha (193079003)
Akshay Bajpai (193079002)
Mohit Agarwala (19307R004)

November 6, 2020

6

What will happen if you test your system on images of people which were not part of the training set? (i.e. the last 8 people from the ORL database). What mechanism will you use to report the fact that there is no matching identity? Work this out carefully and explain briefly in your report. Write code to test whatever you propose on all the 32 remaining images (i.e. 8 people times 4 images per person), as also the entire test set containing 6 images each of the first 32 people. How many false positives/negatives did you get? [15 points]
myTestUnknown.py

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import sys

def find_mean(array):
    """Find the mean of train data
    :param array : the train image matrix
    :output mean : the mean image vector
    """
    mean = np.mean(array,axis=0)
    return mean

def im2double(im):
    """Normalizes the image to be in range 0-1
    :param im : input image
    :output out : min-max normalized output
    """
    min_val = np.min(im)
    max_val = np.max(im)
    out = (im.astype('float') - min_val) / (max_val - min_val)

    return out

def read_train_images_ORL(file_path,subjects,images_per_subject):
    """Reads the ORL train image data and returns a stacked array
    :param file_path: the folder path of ORL dataset
    :param subjects: the number of subjects under training consideration
    :param images_per_subject: the number of images to consider for each
                               test subject
    :output image_array: stacked array output
    """
    image_array = []
    for i in range(1,subjects+1):
        folder_path = os.path.join(file_path,"s"+str(i))
        #print(folder_path)
        for j in range(1,images_per_subject+1):
            filepath = os.path.join(folder_path,str(j)+".pgm")
            im = cv2.imread(filepath,0)
```

```
            image_array.append(im2double(im).ravel())

    return np.array(image_array)

def read_test_images_ORL(file_path,subjects,images_per_subject,starting_index):
    """Reads the ORL test image data and returns a stacked array
    :param file_path: the folder path of ORL dataset
    :param subjects: the number of subjects under test
    :param images_per_subject: the number of images to consider for each
                                test subject
    :param starting_index: starting image number for test images
    :output image_array: stacked array output
    """

    test_image_array_last = []
    test_image_labels_last = []


    for i in range(1,subjects+1):
        folder_path = os.path.join(file_path,"s"+str(i))
        for j in range(starting_index,starting_index+images_per_subject):
            filepath = os.path.join(folder_path,str(j)+".pgm")
            im = cv2.imread(filepath,0)
            test_image_array_last.append(im2double(im).ravel())
            test_image_labels_last.append("s"+str(i))

    for i in range(subjects+1,41):
        folder_path = os.path.join(file_path,"s"+str(i))
        for j in range(1,11):
            filepath = os.path.join(folder_path,str(j)+".pgm")
            im = cv2.imread(filepath,0)
            test_image_array_last.append(im2double(im).ravel())
            test_image_labels_last.append("s"+str(i))


    test_image_array_last = np.array(test_image_array_last)
    return test_image_array_last

def subtract_mean(mean,train,test):
    """Subtracts the mean of the training data from both the
    train and the test stacked array
    :param mean: the mean of the training data
    :param train: the train images stack array
    :param test: the test images stack array
    :output train_mean: mean subtracted train stack
    :output test_mean: mean subtracted test stack
    """
    train_mean = train-mean
    test_mean = test-mean

    return train_mean, test_mean

def normalize_vecs(array):
    """Normalizes the Unitary vector matrix
    :param array: the unitary matrix of vectors from svd
    :output array: the normalized array
    """
    _,c = array.shape
    for i in range(c):
        array[:,i] = array[:,i]/(np.sqrt(np.sum(np.square(array[:,i]))))

    return array

def calculate_svd(array):
    """Calculates svd of the train stack
```

```
    :param array: the train stack
    :output U: the left unitary matrix of the svd output
    """
    U,sigma,V_T = np.linalg.svd(array.T,full_matrices=False)
    U = normalize_vecs(U)
    return U


def calculate_alpha(V,train,test):
    """Calculates the reconstruction matrix alpha for train and test images
    :param V: the unitary matrix from decomposition of train stack
    :param train: the train stack
    :param test: the test stack
    """
    alpha_train = np.matmul(V.T,train.T)
    alpha_test = np.matmul(V.T,test.T)

    return alpha_train, alpha_test


def shiftLbyn(arr, n=0):
    return arr[n::] + arr[:n:]

def calculate_mean(array):
    mean = np.mean(array,axis=0)
    return mean

def normalize_vector(U):
    _,c = U.shape
    for i in range(c):
        U[:,i] = U[:,i]/(np.sqrt(np.sum(np.square(U[:,i]))))
    return U

def cross_validate(array,num):
    r,_ = array.shape
    error = np.zeros(r//num)
    for i in range(0,r,num):
        max_error = 0
        squared_error = 0
        for j in range(6):
            indices = shiftLbyn(list(range(6)),j)
            train_indices = indices[:-1]
            test_index = indices[-1]

            train_images = []
            for k in train_indices:
                train_images.append(array[i+k,:])
            train_images = np.array(train_images)
            mean = calculate_mean(train_images)
            train_images = train_images - mean
            test_image = np.array(array[i+test_index,:])
            test_image = test_image - mean

            # SVD
            U,sigma,V_T = np.linalg.svd(train_images.T,full_matrices=False)
            #print(U.shape)
            U = normalize_vector(U)

            alpha_svd =np.matmul(U.T,train_images.T)
            #print("Shape of alpha ",alpha_svd.shape)
            alpha_svd_test = np.matmul(U.T,test_image.T)
            #print("Shape of alpha_test ",alpha_svd_test.shape)


            squared_error += np.sum(np.square(alpha_svd[:,3:]- alpha_svd_test[3:]))
```

```
        error[i//num] = squared_error/6

    return error




def calculate_and_plot_prediction_rates(train, test, alpha_train, alpha_test, ks, \
                                        dataset, method, threshold_error, light=False):
    """Calculates the prediction rates and plots according to the ks supplied
    :param train: the train stack
    :param test: the test stack
    :param alpha_train: the reconstruction coefficient matrix for the train images
    :param alpha_test: the reconstruction coefficient matrix for test images
    :param ks: the list of k (#of coefficients of reconstruction) to consider
    :param dataset: the name of the dataset under consideration(YALE or ORL)
    :param method: the method used to calculate the unitary matrix (eig or svd)
    :param light: whether to discard the lighting effects of the image
                  (default:  False: means not to discard)
    """
    false_positive = 0
    true_positive = 0
    false_negative = 0
    true_negative = 0
    for counter, ele in enumerate(alpha_test.T):
        errors = np.sum(np.square(alpha_train.T[:,3:]
                -alpha_test.T[counter][3:alpha_train.shape[0]]),axis=1)
        index = np.argmin(errors)
        if counter//4<=31:
            if counter//4 != index//6:
                false_negative += 1
            elif counter//4 == index//6:
                true_positive += 1
        else:
            if errors[index] <= threshold_error[index//6]:
                false_positive += 1
            else:
                true_negative += 1

    print("FP : {} :: FN : {} :: TP : {} :: TN : {}".
            format(false_positive,false_negative,true_positive,true_negative))




def ORL_data(path, subjects, train_images_per_subject, test_images_per_subject, ks):
    """The operation pipeline for the training and testing in ORL Dataset
    :param path: path to ORL dataset
    :param subjects: the number of subjects under training consideration
    :param train_images_per_subject: the number of images to consider for each
                                train subject
    :param test_images_per_subject: the number of images to consider for each
                                test subject
    :param ks: the list of k (#of coefficients of reconstruction) to consider
    """
    train_images = read_train_images_ORL(path,subjects,train_images_per_subject)
    threshold_error = cross_validate(train_images,train_images_per_subject)
    test_images = read_test_images_ORL
        (path,subjects,test_images_per_subject
            ,starting_index=train_images_per_subject+1)
    train_mean = find_mean(train_images)
    train_mean_subtracted,
        test_mean_subtracted = subtract_mean(train_mean, train_images, test_images)
    eigen_vectors = calculate_svd(train_mean_subtracted)
    alpha_eig_train, alpha_eig_test, ks,dataset="ORL",method="eig")
```

```
    unitary_vectors = calculate_svd(train_mean_subtracted)
    alpha_svd_train, alpha_svd_test = calculate_alpha(unitary_vectors, train_mean_subtracted,
        test_mean_subtracted)

    calculate_and_plot_prediction_rates(train_images_per_subject,test_images_per_subject, \
                alpha_svd_train, alpha_svd_test, ks,dataset="ORL",
                method="svd",threshold_error=threshold_error)

if __name__=="__main__":
    ORL_PATH = "../../ORL"
    ORL_k = [192]
    print("For ORL Dataset :")
    print("ks :",ORL_k)
    ORL_data(ORL_PATH,subjects=32,train_images_per_subject=6,test_images_per_subject=4,ks=ORL_k)
```

What will happen if you test your system on images of people which were not part of the training set?

1. In such a case, whichever train image has the least "distance" from the image being tested will get matched. And since we know, that the test image is not a part of the database, the output of the system will always be incorrect.

What mechanism will you use to report the fact that there is no matching identity

1. The mechanism which we have used in this assignment uses the concept of cross validation.
2. We take all 6 train images from first 32 people. We calculate the alpha (reconstruction coefficients) values by taking 5 of these as training set and then find the distance of the 6th image from the calculated values.
3. This step is repeated for all the 6 images of a person by taking one of them as the test image at a time.
4. Thus we have 6 different distance values. The mean of all these 6 values is set as the threshold for declaring a test image being identified as this particular subject.
5. This process is repeated to find individual threshold values for all the 32 subjects which are taken in the training dataset.
6. For testing , now a test image will be declared as an image of a particular subject only if the error or the "distance" is less than the threshold value which has been calculated for that particular subject.

How many falsepositives/negatives did you get

```
For ORL Dataset :
False Positive : 58 :: False Negative : 15 :: True Positive : 113 :: True Negative : 22
```