**3 Complete the lab2.m file**

A face image is considered as a weighted combination of some "basis" faces – eigenfaces. In order to learn eigenfaces, we use Principle Components Analysis (PCA) to get principle components which best preserve more information.

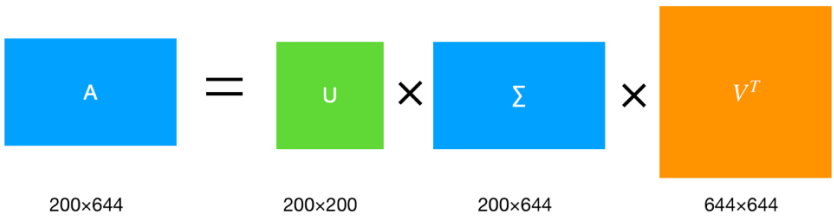
The original face is considered as a vector in a high-dimensional space, in our case each face can be represented by a vector in a 644-dimeansional space ($23 \times 28$), where these dimensions may be correlated. Different from the previous high-dimensional space, eigenface can be represented in a new high-dimensional space (which possibly has lower dimensions) and these dimensions are linearly uncorrelated i.e. they are orthogonal. In fact, 20 eigenfaces are selected as principle components.

**3.2 Construct the mean image and the covariance matrix of the training image set.**

In our training set, there are 200 records (images) and each record has 644 attributes (pixels). The procedures to construct the covariance: First, calculate the mean vector for 200 records. Then, using train data minus mean data to get 'CenteredVectors'. Finally, using the centered vectors to calculate the covariance matrix.

**3.3 Compute the eigenfaces of the training set.**

$$A = U\Sigma V^T \tag{1}$$




$$A_{m\times n} \approx U_{m\times r}\Sigma_{r\times r}V^T{}_{r\times n} \tag{2}$$

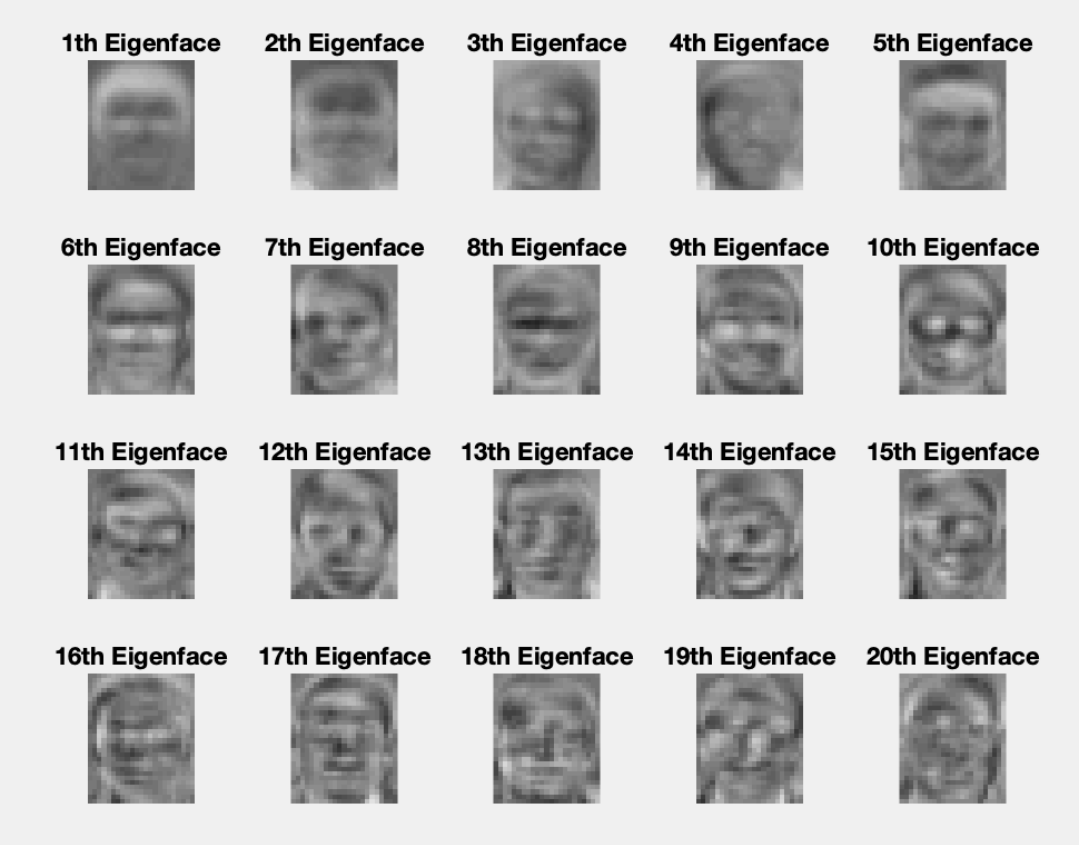
In order to get eigenfaces (eigenvectors), we should use singular value decomposition. The size of each matrix is shown above. Actually, the vectors in $V^T$ are the eigenvectors we are finding. In addition, we could use the first some vectors in $V^T$ to represent the whole space, which still preserves most of information. In the next step, we choose first 200 eigenfaces as our principle components space ('Space').

**3.4 Display the mean image.**


The mean image is calculated by averaging each corresponding pixel of all the images.

### 3.5 Display the first 20 eigenfaces.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display of the 20 first eigenfaces : Write your code here
figure;
x=4;
y=5;
Eigenface = uint8 (zeros(28, 23));
for i = 1:20
    for k = 0:643
        Eigenface( mod (k,28)+1, floor(k/28)+1 ) = (Space (i,k+1)+0.2)*(255/0.4);
    end
 subplot (x,y,i);
 imshow(Eigenface);
 title([ num2str(i),'th Eigenface']);
end
```




### 3.6 Project both training images and testing images onto the first 20 eigenfaces.

Using the given function to project all the train and test images onto the space ("Space") with 200 dimensions, which is lower than the original 644 dimensions. Threshold is set as 20, which means we only use the first 20 eigenfaces.

### 3.7 Compute the distance from the projected test images to the projected training images.

The projection value of the train image subtracts that of the test image at each corresponding position, then the result is squared. The distance of a test image of a train image is the sum by aggregating all the result values from the previous step for all the 20 eigenfaces. Compute all the distance between all the train images and all the test images. At last, we get a distance matrix – 'Distance'.

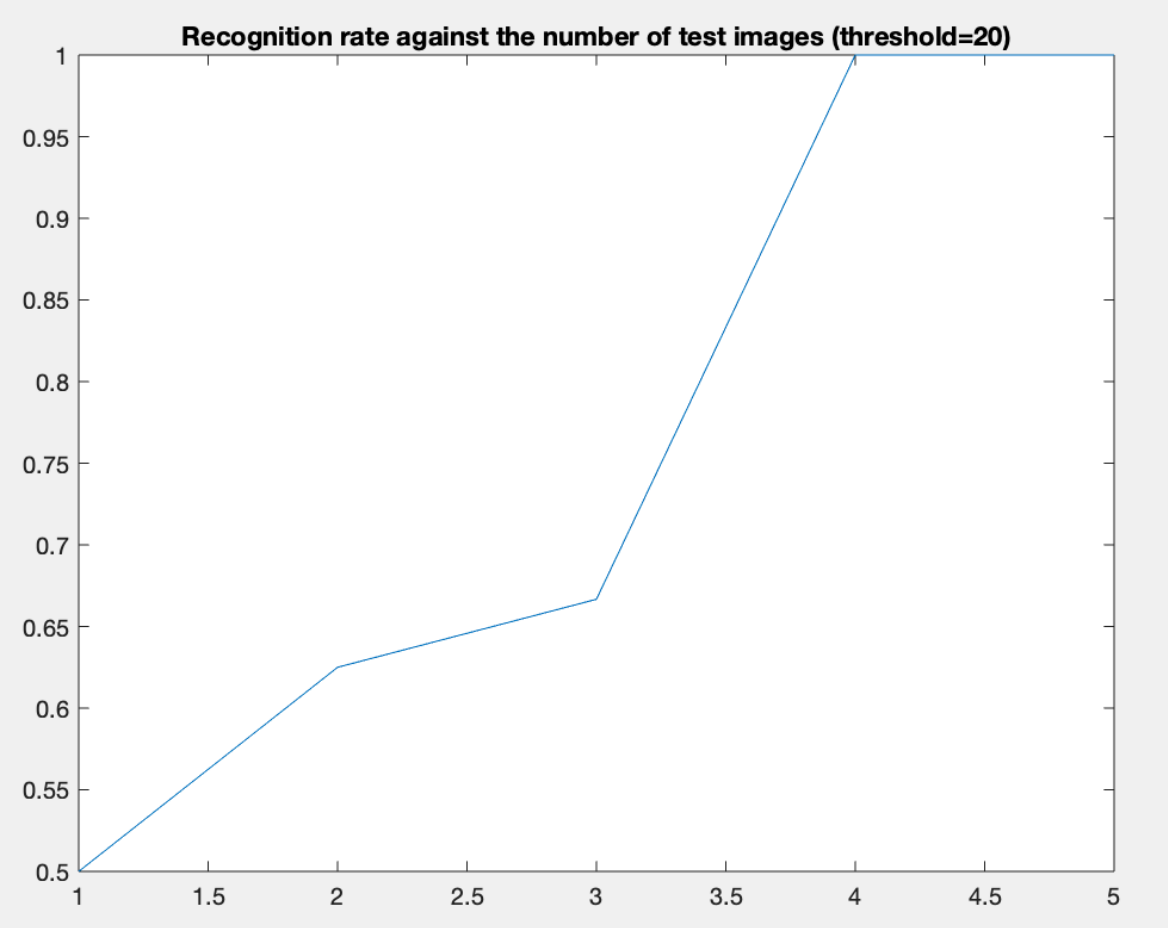
### 3.8 Display the top 6 best matched training images for each test image.


From the image above, we can find that almost all the matched results are right, expect for the two images in first row.

### 3.9 Compute the recognition rate using 20 eigenfaces.

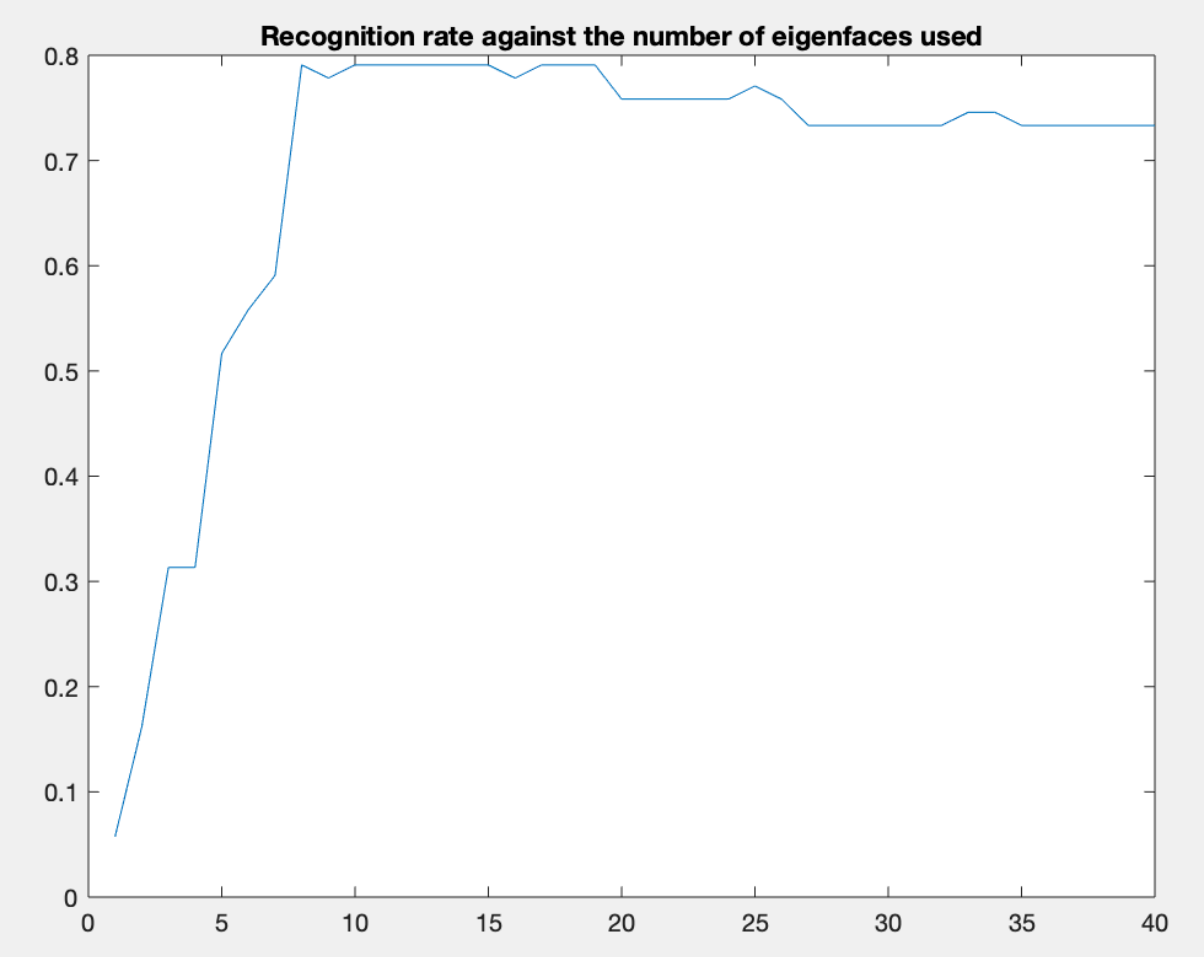
```
%recognition rate compared to the number of test images: Write your code here to co
mpute the recognition rate using top 20 eigenfaces.
number_of_test_images=zeros(1,40);% Number of test images of one given person.
for  i=1:70
    number_of_test_images(1,Identity(1,i))= number_of_test_images(1,Identity(1,i))+
1;
end

recognised_person=zeros(1,40);% Number of the recognised person for each tested per
son.
recognitionrate=zeros(1,5);
number_per_number=zeros(1,5);% Number of persons who have 1, 2,..,5 images in the t
esting set.


i=1;
while (i<70)
    id=Identity(1,i);
    distmin=Values(id,1);
    indicemin=Indices(id,1);
```

```
17.
18.     while (i<70)&&(Identity(1,i)==id)
19.         if (Values(i,1)<distmin)
20.             distmin=Values(i,1);
21.             indicemin=Indices(i,1);
22.         end
23.         i=i+1;
24.     end
25.
26.     recognised_person(1,id)=indicemin;
27.     number_per_number(number_of_test_images(1,id))=number_per_number(number_of_test
    _images(1,id))+1;
28.
29.     if (id==floor((indicemin-1)/5)+1) %the good person was recognised
30.         recognitionrate(number_of_test_images(1,id))=recognitionrate(number_of_test
    _images(1,id))+1;
31.     end
32. end
33.
34. for  i=1:5
35.     recognitionrate(1,i)=recognitionrate(1,i)/number_per_number(1,i);
36. end
37.
38. figure;
39. plot (recognitionrate(1,:));
40. title('Recognition rate against the number of test images (threshold=20)');
```



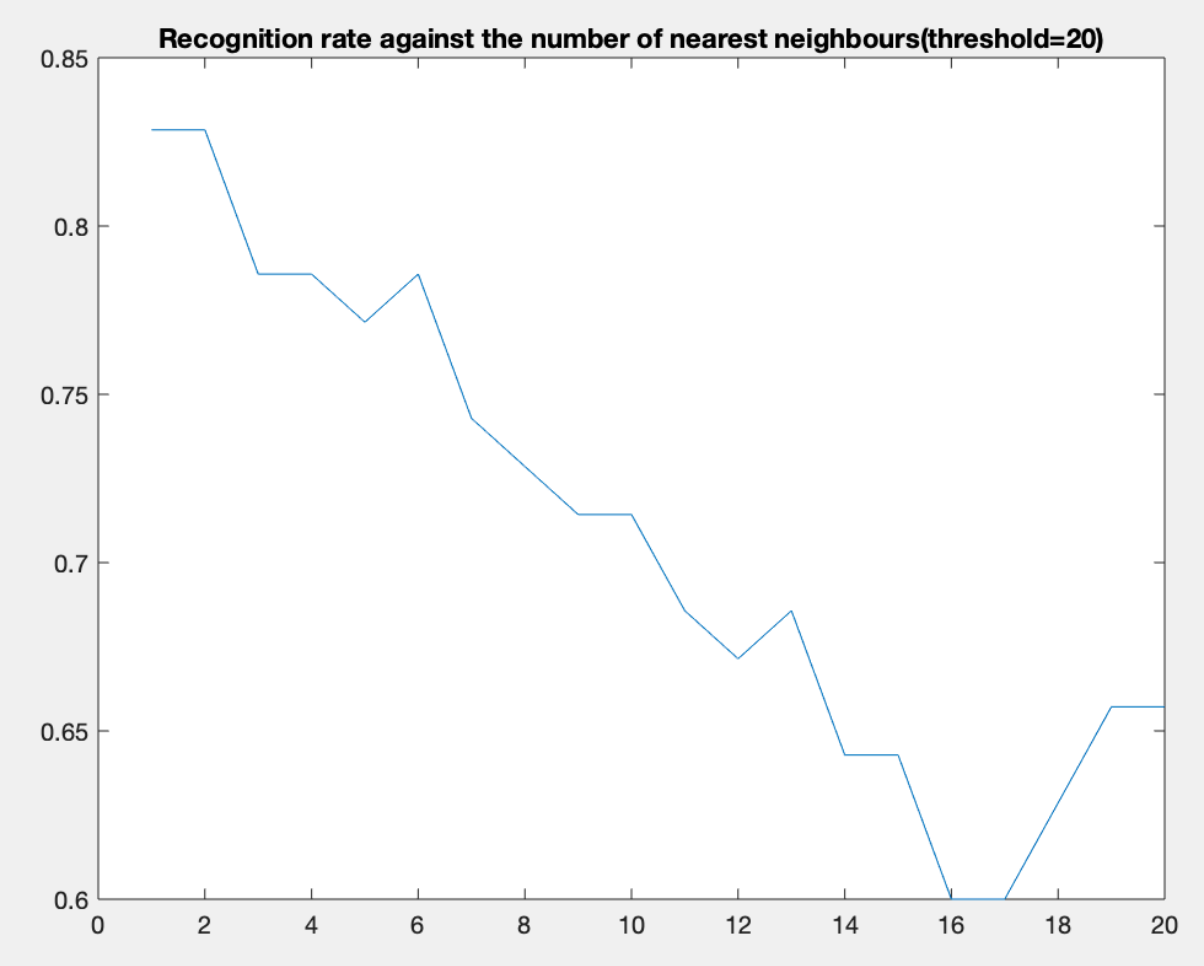

From the image above, we can find that persons who has 4 or 5 test images in test dataset are 100% recognized, whereas the percentages for persons who just has 1, 2, 3 test images are dramatically low (all of them are lower than 70%). In brief, with the increase of the test image number a person has, the person becomes easier to be recognized.

**3.10 Investigate the effect of using different number of eigenfaces for recognition (e.g. plot the recognition rate against the number of eigenfaces).**




From the plot above, it is clear that the recognition rate rockets to nearly 80% at the first stage (the number of eigenfaces used increase from 1 to around 7 or 8), and then the recognition rate keeps steady with the increase of the number of eigenfaces used. After the number of eigenfaces used surpasses 20, there is a slightly drop in recognition rate (which is still above 70%). This is possibly because more eigenfaces benefit recognition rate at the early stage with the increase of the number of eigenfaces used, but too many eigenfaces would make the recognition rate decrease slightly. Finally, we can draw a conclusion that the best bracket of the number of eigenfaces used for recognition is between around 7 and 20.

**3.11 Investigate the effect of K in K-Nearest-Neighbour (KNN) classifier.**
(Using 20 eigenfaces i.e. 'Threshold' is set as 20.)




From the image above, we can clearly find that the recognition rate goes down dramatically with the increase of K in KNN, though there is a slight fluctuation. When the K is set 1 or 2, the recognition rate is the peak, up to roughly 83%, and when K is set as 16 or 17, the recognition rate is the lowest (60%).

```
%effect of K: You need to evaluate the effect of K in KNN and plot the recognition
rate against K. Use 20 eigenfaces here.

averageRR=zeros(1,20);
Threshold =20;
Distances=zeros(TestSizes(1),TrainSizes(1));

for i=1:TestSizes(1)
    for j=1: TrainSizes(1)
        Sum=0;
        for k=1: Threshold
            Sum=Sum+((Locationstrain(j,k)-Locationstest(i,k)).^2);
        end
        Distances(i,j)=Sum;
    end
end

Values=zeros(TestSizes(1),TrainSizes(1));
Indices=zeros(TestSizes(1),TrainSizes(1));
for i=1:70
```

```
    [Values(i,:), Indices(i,:)] = sort(Distances(i,:));
end


person=zeros(70,200);
person(:,:)=floor((Indices(:,:)-1)/5)+1;

for K=1:20
recognised_person_=zeros(1,70);
recognitionrate=0;
number_per_number=zeros(1,5);
number_of_occurance=zeros(70,K);

for i=1:70

    max=0;
    for j=1:K
        for k=j:K
            if (person(i,k)==person(i,j))
                number_of_occurance(i,j)=number_of_occurance(i,j)+1;
            end
        end

        if (number_of_occurance(i,j)>max)
            max=number_of_occurance(i,j);
            jmax=j;
        end
    end
    recognised_person(1,i)=person(i,jmax);

    if (Identity(1,i)==recognised_person(1,i))
        recognitionrate=recognitionrate+1;
    end

    averageRR(1,K)=recognitionrate/70;

end

end

figure;
plot(averageRR(1,:));
title('Recognition rate against the number of nearest neighbours(threshold=20)');
```