UE19CS333 - IPCV Assignment 6

Pattern Classification – Face recognition using Eigenfaces

Team No: 5
Name and SRN of Members of the Team:

Abhishek Aditya BS - PES1UG19CS019
T Vijay Prashant - PES1UG19CS536
Vishal R - PES1UG19CS571
Yashas KS - PES1UG19CS589

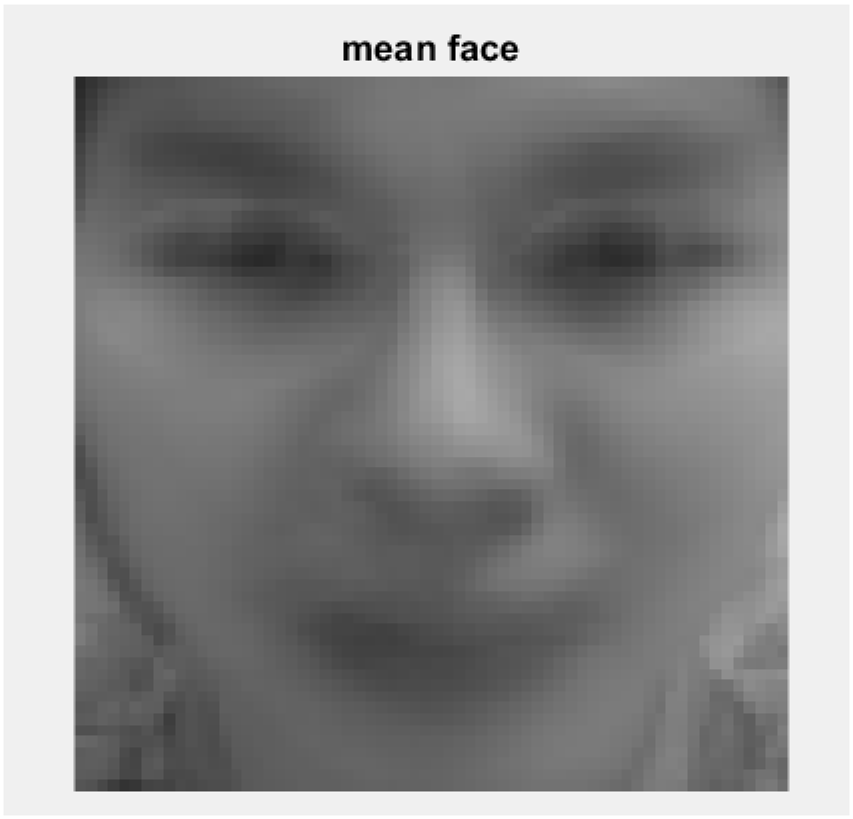
1. Read ‘faceData.mat’ that contains 68 face images, each of size 64x64 for 10 people. Split this into train (64 face images per person) and test data (4 face images per person)

```
load faceData.mat;
train = [];
train_label = [];
for i=1:10;
   for j=1:64;
      temp = faceData{i}{j};
      temp = temp(:);
      train = horzcat(train,temp);
      train_label = cat(1,train_label,i);
   end;
end;
test = [];
test_label = [] ;
for i=1:10;
for j=65:68;
   temp = faceData{i}{j};
   temp = temp(:);
   test = horzcat(test,temp);
   test_label = cat(1, test_label,i);
end
end
```

2. Use the training data to compute the covariance matrix and find the eigenvalues and eigenvectors for this. Plot the ‘mean face’ and first five eigenvectors (Eigenfaces).

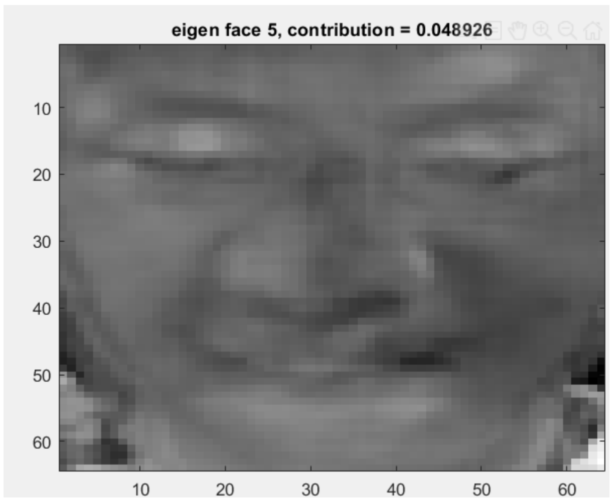
Mean face:

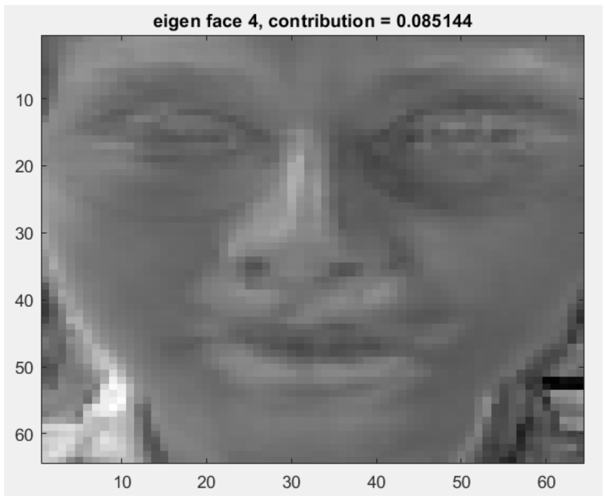
```
meanImg = mean(train,2);
figure; imshow(uint8(reshape(meanImg,64,64)));
hold on; title('mean face');
```

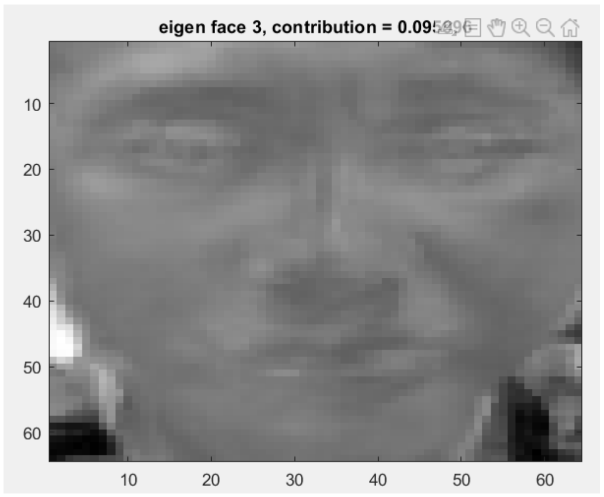


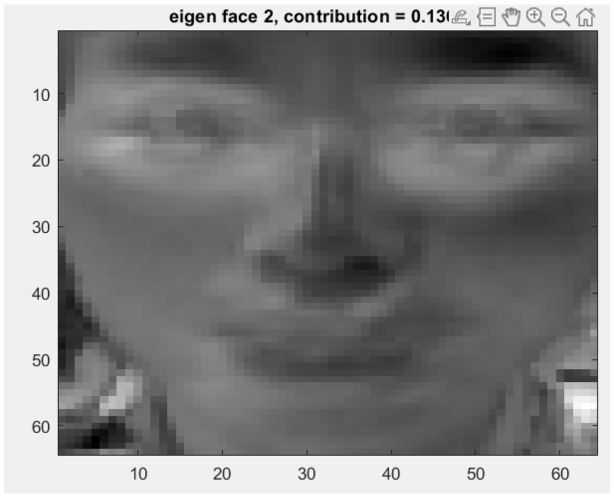
Covariance matrix and Eigen faces:

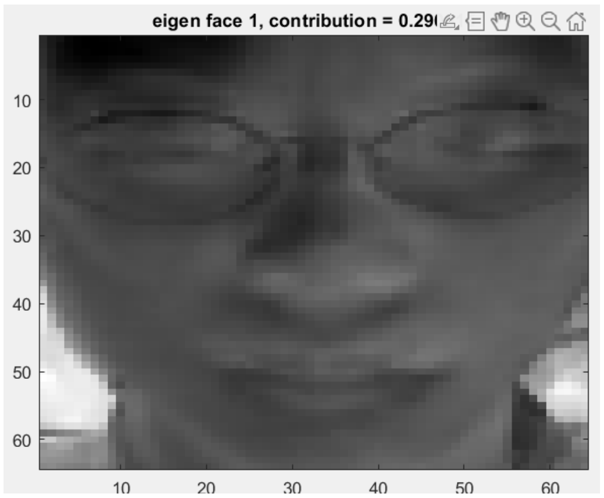
```
covMatrix = (train-meanImg)*((train-meanImg)');
[eigVec, eigVal] = eig(covMatrix);
for i=1:5
   figure; imagesc(reshape(redEig(:,i),64,64)); colormap gray;
   hold on;
   str1=sprintf('eigen face %d, contribution = %f',i,(eigVal(i,i))/trace(eigVal));
   title(str1);
end
```


eigen face 5, contribution = 0.048926
10
20
30
40
50
60


eigen face 4, contribution = 0.085144
10
20
30
40
50
60


eigen face 3, contribution = 0.09
10
20
30
40
50
60


eigen face 2, contribution = 0.13
10
20
30
40
50
60

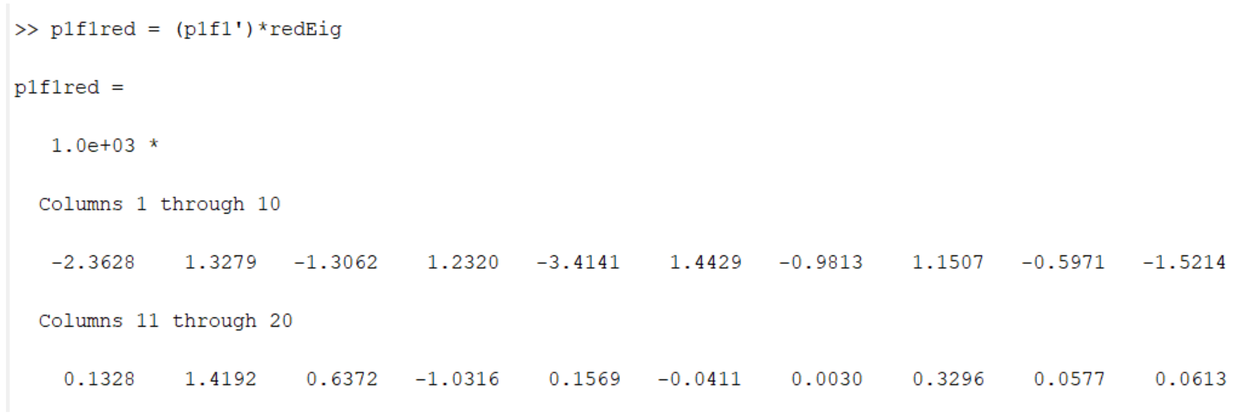

eigen face 1, contribution = 0.29
10
20
30
40
50
60

3. How many eigenvectors are required to achieve a contribution of >= 90%? Project the first face image of (person 1 – meanface) to the reduced eigenspace to obtain a 20x1 vector, [a1 a2 ... a20]T.

A minimum of 20 eigen vectors are required to achieve a contribution of 90%.

```
>> contributn = (trace(eigVal(1:20,1:20)))/(trace(eigVal))
contributn = (trace(eigVal(1:19,1:19)))/(trace(eigVal))
contributn = (trace(eigVal(1:21,1:21)))/(trace(eigVal))

contributn =

   0.9041 - 0.0000i



contributn =

   0.8996 - 0.0000i



contributn =

   0.9085 - 0.0000i
```

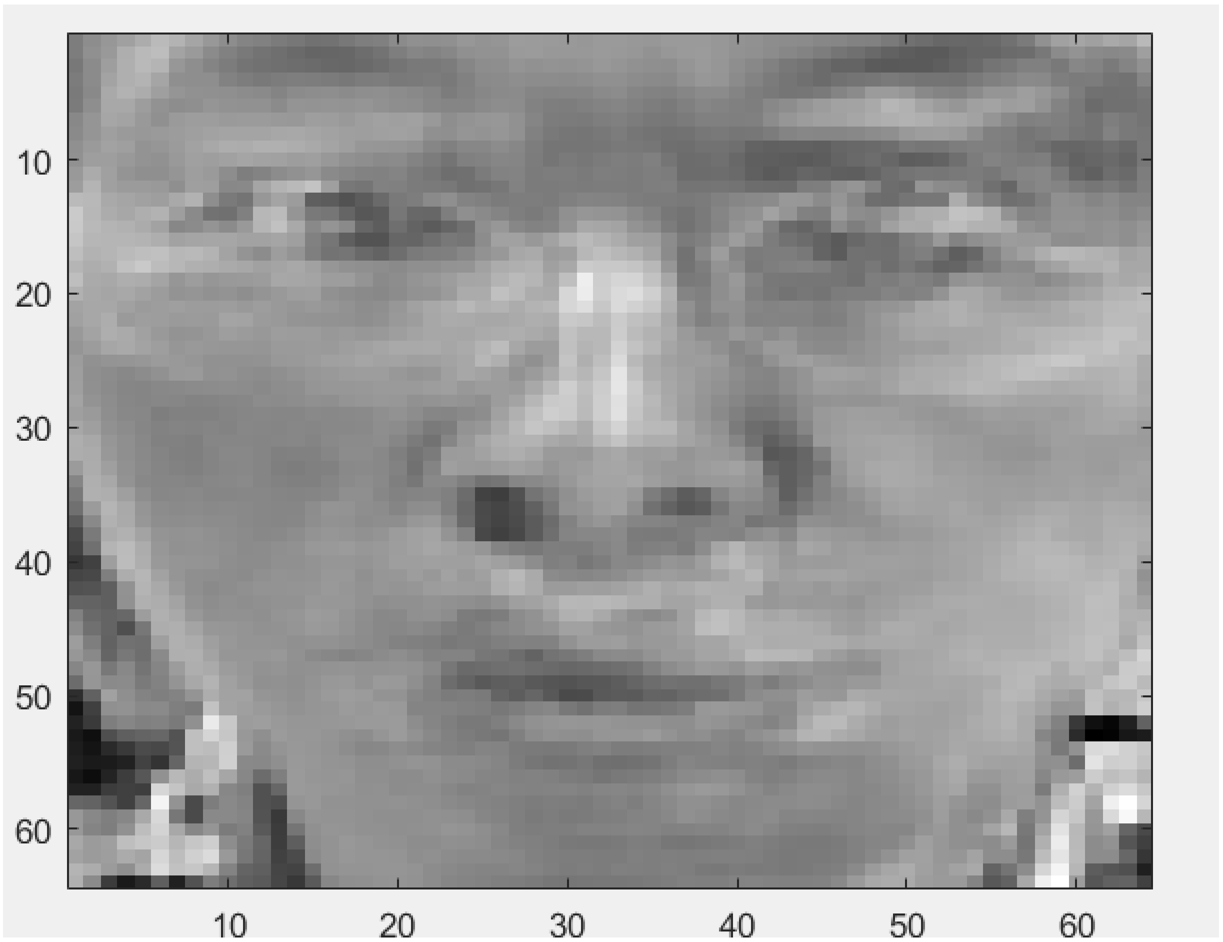
covMatrix = (train-meanImg)*((train-meanImg)');
[eigVec, eigVal] = eig(covMatrix);
redEig = eigVec(:,1:20);



```
>> p1f1red = (p1f1')*redEig

p1f1red =

   1.0e+03 *

  Columns 1 through 10

   -2.3628    1.3279   -1.3062    1.2320   -3.4141    1.4429   -0.9813    1.1507   -0.5971   -1.5214

  Columns 11 through 20

    0.1328    1.4192    0.6372   -1.0316    0.1569   -0.0411    0.0030    0.3296    0.0577    0.0613
```

4. Reconstruct the face for person 1 as (reconstructedFace = meanface + a1*eigenvector1 + a2*eigenvector2 +..a20*eigenvector20)

```
plf1recons = meanImg + redEig*(plf1red');
figure;
imagesc(reshape((plf1recons(:,1)),64,64)); colormap gray;
```



What is the MSE between the original face of person1 and reconstructed face?

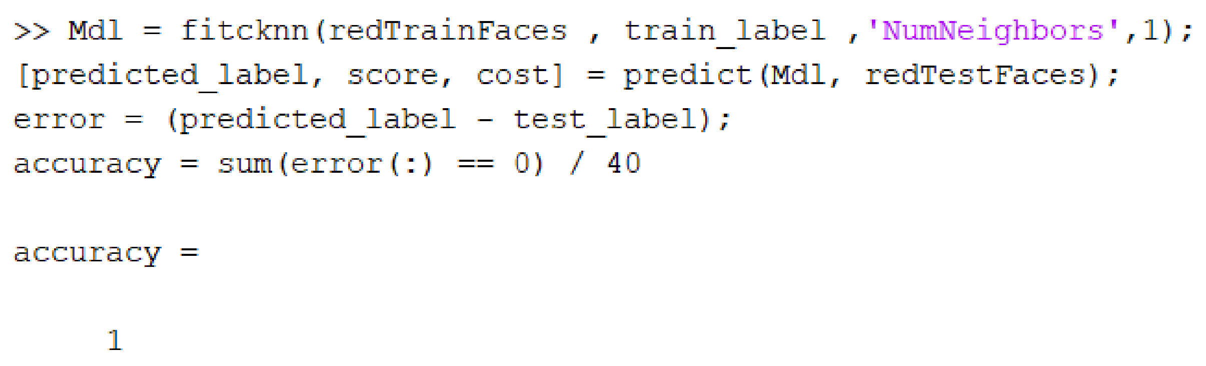
%SqError = (p1f1 – reconstructedFace).*(p1f1 – reconstructedFace)

%MSE = sum(sum(SqError))/ (size(SqError,1)*size(SqError,2))

```
>> reconsMSE = sum((p1f1-p1f1recons).*((p1f1-p1f1recons)))/(size(p1f1,1)*size(p1f1,2))

reconsMSE =

   5.3826e+03
```

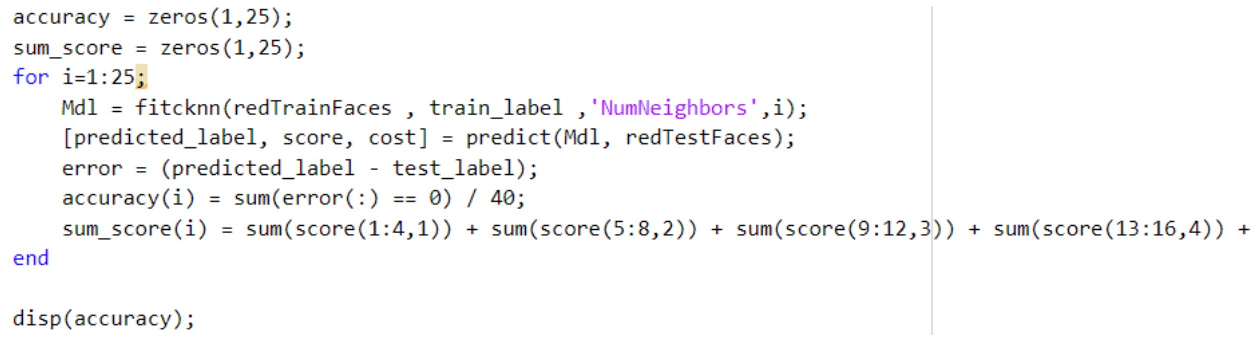
5. Find the 20x1 representations for all the training data (640x20) and for all the test data (40x20). Use the k-nearest neighbor classification (k=1) to label the test images and compute the average accuracy. (Note: target labels would look like [1 1 1 1 2 2 2 2 .... 20 20 20 20] T since there are 4 face images per person in the test data.

redTestFaces = (test')*redEig;
redTrainFaces = (train')*redEig;



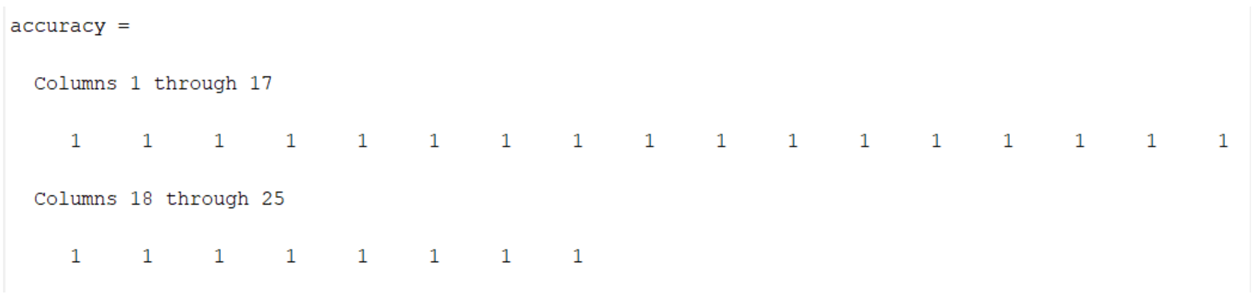
```
>> Mdl = fitcknn(redTrainFaces , train_label ,'NumNeighbors',1);
[predicted_label, score, cost] = predict(Mdl, redTestFaces);
error = (predicted_label - test_label);
accuracy = sum(error(:) == 0) / 40

accuracy =

     1
```

6. Find the most appropriate value for k (0<k<=25) that yields the maximum classification accuracy. Paste the code and report the value of k.



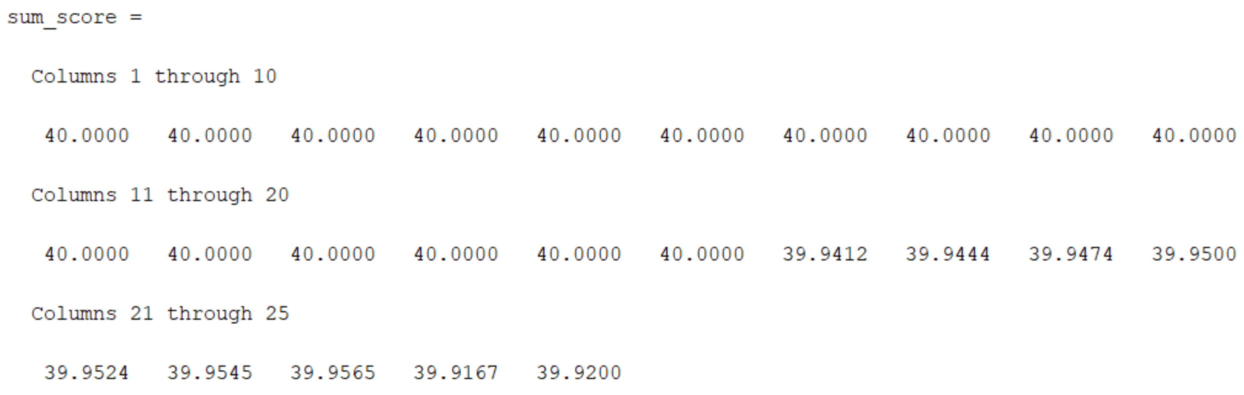
```
accuracy = zeros(1,25);
sum_score = zeros(1,25);
for i=1:25;
    Mdl = fitcknn(redTrainFaces , train_label ,'NumNeighbors',i);
    [predicted_label, score, cost] = predict(Mdl, redTestFaces);
    error = (predicted_label - test_label);
    accuracy(i) = sum(error(:) == 0) / 40;
    sum_score(i) = sum(score(1:4,1)) + sum(score(5:8,2)) + sum(score(9:12,3)) + sum(score(13:16,4)) +
end

disp(accuracy);
```

Classifiaction accuracy for all k between 0 and 25 is 1



```
accuracy =

  Columns 1 through 17

     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1     1

  Columns 18 through 25

     1     1     1     1     1     1     1     1
```

Sum_score store the sum of confidence of the classification



```
sum_score =

  Columns 1 through 10

   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000

  Columns 11 through 20

   40.0000   40.0000   40.0000   40.0000   40.0000   40.0000   39.9412   39.9444   39.9474   39.9500

  Columns 21 through 25

   39.9524   39.9545   39.9565   39.9167   39.9200
```

It can be seen that 1 to 16 the confidence for prediction of each class is 40 which equates to one for every prediction

So choosing an optimal value of k based on validation accuracy is not possible

Based on research optimal k is sqrt(no of labels)

sqrt(10) = 3.16
So optimal can be chosen as 3.