ธนนท์ ต้องลาภา 6231374221

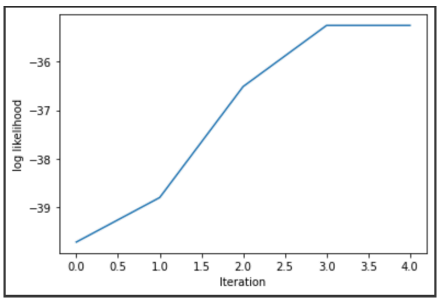
T1. Using 3 mixtures, initialize your Gaussian with means (3,3), (2,2), and (-3,-3), and standard Covariance, **I**, the identity matrix. Use equal mixture weights as the initial weights. Repeat three iterations of EM. Write down $w_{n,j}, m_j, \vec{\mu_j}, \mathbf{\Sigma}_j$ for each EM iteration. (You may do the calculations by hand or write code to do so)

```
Iteration:  0
Theta:
[0.45757242 0.20909425 0.33333333]
W:
[1.19202922e-01 8.80797076e-01 1.81545808e-09]
[7.31058579e-01 2.68941421e-01 1.69570706e-16]
[2.68941421e-01 7.31058579e-01 1.01529005e-11]
[9.99983299e-01 1.67014218e-05 2.03105874e-42]
[9.99088949e-01 9.11051194e-04 5.37528453e-32]
[9.99876605e-01 1.23394576e-04 3.30529272e-37]
[2.31952283e-16 1.38879439e-11 1.00000000e+00]
[2.31952283e-16 1.38879439e-11 1.00000000e+00]
[3.30570063e-37 5.90009054e-29 1.00000000e+00]
Means:
[5.78992692 5.81887265]
[1.67718211 2.14523106]
[-4.         -4.66666666]
Cov:
[[4.53619412 0.        ]
 [0.         4.28700611]]
[[0.51645579 0.        ]
 [0.         0.13152618]]
[[4.66666668 0.        ]
 [0.         2.88888891]]
```

```
Iteration:  1
Theta:
[0.40711618 0.25954961 0.33333421]
W:
[3.16932821e-03 9.96824702e-01 5.96935641e-06]
[6.55101207e-01 3.44898109e-01 6.84250685e-07]
[5.77503537e-03 9.94223665e-01 1.30002282e-06]
[1.00000000e+00 9.14501760e-73 4.53098372e-19]
[1.00000000e+00 3.18241075e-32 5.49960655e-14]
[1.00000000e+00 1.60282391e-50 1.66508966e-16]
[4.73612484e-08 1.97981704e-52 9.99999953e-01]
[3.08502694e-08 1.35874716e-67 9.99999969e-01]
[5.39509443e-016 1.08758623e-168 1.00000000e+000]
Means:
[6.27176215 6.27262711]
[1.72091544 2.14764812]
[-3.99998589 -4.6666488 ]
Cov:
[[2.94672736 0.        ]
 [0.         2.93847196]]
[[0.49649261 0.        ]
 [0.         0.12584815]]
[[4.66673088 0.        ]
 [0.         2.88900236]]
```

```
Iteration:  2
Theta:
[0.36070909 0.30595677 0.33333414]
W:
[9.82897443e-05 9.99896667e-01 5.04302423e-06]
[2.45965474e-01 7.54033295e-01 1.23094211e-06]
[3.18013803e-04 9.99680992e-01 9.94302212e-07]
[1.00000000e+00 9.43430508e-76 3.14527994e-19]
[1.00000000e+00 1.86592722e-33 4.19732750e-14]
[1.00000000e+00 1.37418755e-52 1.08248377e-16]
[5.61748867e-13 6.98460925e-55 1.00000000e+00]
[3.64921259e-13 1.02517456e-70 1.00000000e+00]
[1.03044593e-025 1.72424695e-176 1.00000000e+000]
Means:
[6.6962644  6.69629468]
[1.91071238 2.27383436]
[-3.99998673 -4.6666501 ]
Cov:
[[1.73961067 0.        ]
 [0.         1.73929602]]
[[0.62898406 0.        ]
 [0.         0.1988491 ]]
[[4.66672942 0.        ]
 [0.         2.88899545]]
```

T2. Plot the log likelihood of the model given the data after each EM step. In other words, plot $log \prod_n p(\vec{x_n}|\phi, \vec{\mu}, \mathbf{\Sigma})$. Does it goes up every iteration just as we learned in class?



จะเห็นได้ว่าในทุกๆ Iteration log likelihood จะเพิ่มขึ้น

โดยที่ $p(\vec{x}_n|\phi,\vec{\mu},\Sigma) = \sum p(\vec{x}_n,k|\phi,\vec{\mu},\Sigma)p(k)$

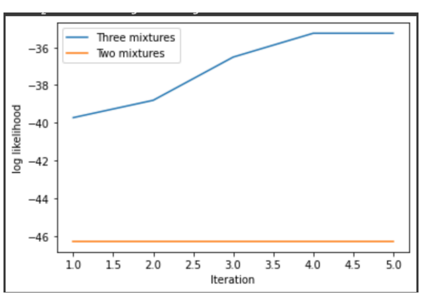
T3. Using 2 mixtures, initialize your Gaussian with means (3,3) and (-3,-3), and standard Covariance, **I**, the identity matrix. Use equal mixture weights as the initial weights. Repeat three iterations of EM. Write down $w_{n,j}, m_j, \vec{\mu_j}, \mathbf{\Sigma}_j$ for each EM iteration.

```
Iteration:  0
Theta:
[0.66666666 0.33333334]
W:
[9.99999985e-01 1.52299795e-08]
[1.00000000e+00 2.31952283e-16]
[1.00000000e+00 3.77513454e-11]
[1.00000000e+00 2.03109266e-42]
[1.00000000e+00 5.38018616e-32]
[1.00000000e+00 3.30570063e-37]
[2.31952283e-16 1.00000000e+00]
[2.31952283e-16 1.00000000e+00]
[3.30570063e-37 1.00000000e+00]
Means:
[4.50000001 4.66666667]
[-3.99999997 -4.66666663]
Cov:
[[6.91666665 0.        ]
 [0.         5.88888889]]
[[4.66666677 0.        ]
 [0.         2.8888891 ]]
```

```
Iteration:  1
Theta:
[0.66669436 0.33330564]
W:
[9.99879274e-01 1.20725832e-04]
[9.99999741e-01 2.59403362e-07]
[9.99975922e-01 2.40783341e-05]
[1.00000000e+00 9.39286607e-19]
[1.00000000e+00 7.41043154e-14]
[1.0000000e+00 2.9836637e-16]
[2.41448223e-04 9.99758552e-01]
[1.52869075e-04 9.99847131e-01]
[5.22429300e-09 9.99999995e-01]
Means:
[4.49961311 4.66620178]
[-3.99993241 -4.66651231]
Cov:
[[6.91944755 0.        ]
 [0.         5.89275124]]
[[4.66806942 0.        ]
 [0.         2.89103318]]
```

```
Iteration:  2
Theta:
[0.66669453 0.33330547]
W:
[9.99878589e-01 1.21411383e-04]
[9.99999738e-01 2.61769575e-07]
[9.9997577e-01 2.4229583e-05]
[1.00000000e+00 9.63496021e-19]
[1.00000000e+00 7.54839047e-14]
[1.00000000e+00 3.04963776e-16]
[2.42792697e-04 9.99757207e-01]
[1.53838319e-04 9.99846162e-01]
[5.28882554e-09 9.99999995e-01]
Means:
[4.49961084 4.66619903]
[-3.99993206 -4.66651141]
Cov:
[[6.91946372 0.        ]
 [0.         5.8927741 ]]
[[4.66807754 0.        ]
 [0.         2.89104566]]
```

T4. Plot the log likelihood of the model given the data after each EM step. Compare the log likelihood between using two mixtures and three mixtures. Which one has the better likelihood?

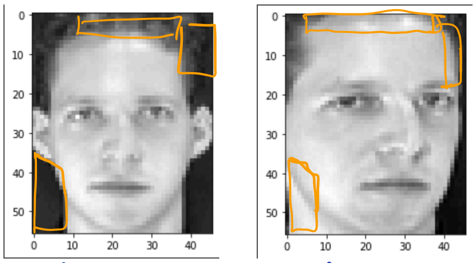


แบบ Three mixture จะดีกว่า เนื่องจากข้อมูลมีลักษณะ

เป็นสามกลุ่มแยกกันอยู่แล้ว ดังนั้นการที่มี Three mixture

จึงมี likelihood ที่มากกว่า แบบ Two mixture

T5. What is the Euclidean distance between `xf[0,0]` and `xf[0,1]`? What is the Euclidean distance between `xf[0,0]` and `xf[1,0]`? Does the numbers make sense? Do you think these numbers will be useful for face verification?



xf[0,0]   xf[0,1]

Euclidean distance = 10.04

ซึ่งตัวเลขที่ได้ แสดงถึงว่าทั้งสองรูปมีความแตกต่างกันในระดับหนึ่ง ซึ่งประมาณส่วนที่ถูก Mark โดยสีส้มและมีปริมาณไม่มากเมื่อเทียบทั้งรูป

โดย Euclidien distance นี้คำนวณจาก xf ที่ถูก Scale แล้ว ตัวเลขเหล่านี้แสดงถึงความต่างของแต่ละ Pixel ซึ่งตัวเลขนี้ไม่สามารถบอกอะไรได้เลยถึงการทำ Face verification ที่จะใช้นำไปแยกรายบุคคลจากบุคคลที่เป็นต้นแบบ โดยเนื่องจากรูปภาพแต่ละรูปเป็นเฉพาะใบหน้าที่ถ่ายในมุมที่ใกล้เคียงกัน แสงที่ใกล้เคียงกัน จึงทำให้ยากที่จะแยกระหว่างกันได้ ดังนั้นอาจต้องการเทียบหลายๆภาพที่จะให้มุมที่ใกล้.

```
1 np.linalg.norm(pic00-pic01, ord = 2)
10.037616294165492
```
คนเดียวกัน ต่างมุม.

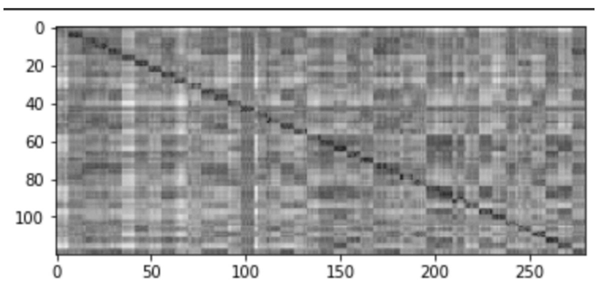
```
1 np.linalg.norm(pic10-pic00, ord = 2)
8.173295099737283
```
ต่างคน

```
1 np.linalg.norm(pic10-pic11, ord = 2)
5.824297830071833
```
คนเดียวกัน ต่างมุม.

T6. Write a function that takes in a set of feature vectors T and a set of feature vectors D, and then output the similarity matrix A. Show the matrix as an image. Use the feature vectors from the first 3 images from all 40 people for list T (in order $x[0,0], x[0,1], x[0,2], x[1,0], x[1,1], \ldots x[39,2]$). Use the feature vectors from the remaining 7 images from all 40 people for list D (in order $x[0,3], x[0,4], x[0,5], x[1,6], x[0,7], x[0,8], x[0,9], x[1,3], x[1,4] \ldots x[39,9]$). We will treat T as our training images and D as our testing images

The picture below shows an example similarity matrix calculated by the first 5 images from the first 5 people (for both T and D).

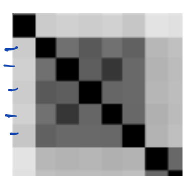
T



D

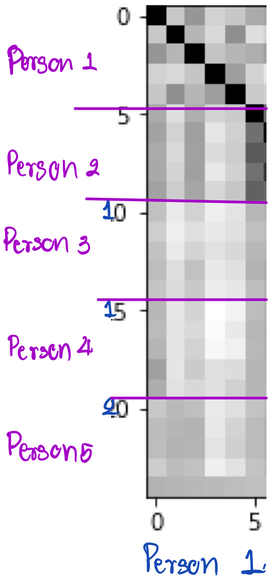
T7. From the example similarity matrix above, what does the black square between [5:10,5:10] suggest about the pictures from person number 2? What do the patterns from person number 1 say about the images from person 1?

5
6
7
8
9



5 6 7 8 9
Person 2

แสดงให้เห็นว่ารูปที่มีลักษณะคล้ายกันมาก จะค่า Distance ต่ำๆ แสดงสีดำๆ. โดยถ้าเป็นรูปเดียวกันจะแสดงสีดำทึบเลย แต่ถ้ามีความต่างกันน้อยมากๆ สีที่แสดงสีเทาเข้ม ดังนั้นรูปที่ คนที่สองในแต่ละมุมจึงมีความคล้ายกันมาก



Pattern หนึ่ง ที่เราจะเห็นได้จากคนที่หนึ่งเลยว่า

ถ้ารูปนั้นเกิดมาจากคนละคนกัน จะไม่มีทางเลยที่ค่า Distance จะสูง จนปรากฏสีในรูปภาพเป็นสีขาว

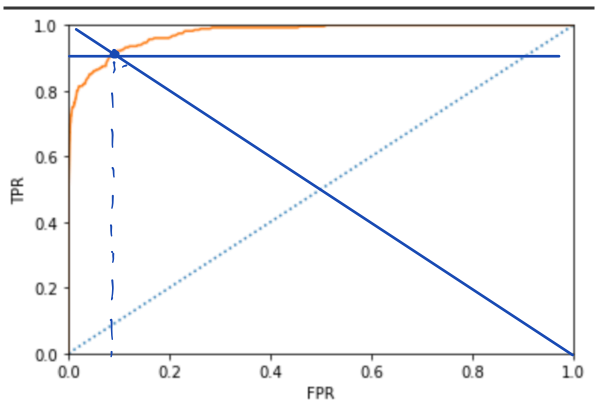
แต่ถ้ามารูปนั้นมาจากคนๆ เดียว ก็จะมีค่า Distance ที่ต่ำ ๆ แต่ในบางครั้งถ้ารูปของอีกมีลักษณะที่คล้ายกันกับรูปที่นำไปทดสอบ อาจจำเป็นที่จะต้องทดสอบ โดยหลายๆ รูป พร้อมกัน

T8. Write a function that takes in the similarity matrix created from the previous part, and a threshold t as inputs. The outputs of the function are the true positive rate and the false alarm rate of the face verification task (280 Test images, tested on 40 people, a total of 11200 testing per threshold). What is the true positive rate and the false alarm rate for $t = 10$?

```
simple_face_verification(T,D, t = 10, verbose = True)
```

```
279 4984 5936 1
{'FAR': 0.4564102564102564, 'TPR': 0.9964285714285714}
```

T9. Plot the RoC curve for this simple verification system. What should be the minimum threshold to generate the RoC curve? What should be the maximum threshold? Your RoC should be generated from at least 1000 threshold levels equally spaced between the minimum and the maximum. (You should write a function for this).



Minimum : 0 (Distance ≥ 0)

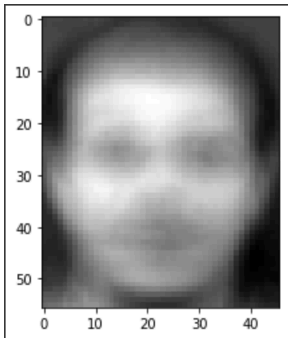
Maximum : 20 (ทดลองแล้วว่าใกล้เคียงจุดที่เริ่ม TPR = 1 และ FPR = 1)

T10. What is the EER (Equal Error Rate)? What is the recall rate at 0.1% false alarm rate? (Write this in the same function as the previous question)

- EER ที่ t = 7.8%

- Recall = 0.575 at FAR = 0.1%

T11. Compute the mean vector from the training images. Show the vector as an image (use `numpy.reshape()`). This is typically called the meanface (or meanvoice for speech signals). You answer should look exactly like the image shown below.



T12. What is the size of the covariance matrix? What is the rank of the covariance matrix?

Size of covariance matrix คือ 2576 * 2576

และ: Rank of covariance matrix คือ 119

```
1 np.linalg.matrix_rank(cov_matrix)
119
```

T13. What is the size of the Gram matrix? What is the rank of Gram matrix? If we compute the eigenvalues from the Gram matrix, how many non-zero eigenvalues do we expect to get?

Gram matrix ($\hat{X}^T\hat{X}$) มีขนาด 120 × 120

Rank ของ Gram matrix = 119

∴ จะมี Non-zero eigenvalue 119 ตัว.

T14. Is the Gram matrix also symmetric? Why?
Using the gram matrix, we instead solve for the eigenvector, $\vec{v'}$.

$$\hat{X}^T \hat{X} \vec{v'} = \lambda \vec{v'} \tag{7}$$

where the desired eigenvector (eigenvector of the covariance matrix) can be computed from $\vec{v'}$ (eigenvector of the gram matrix) using the following relationship

$$\vec{v} = \hat{X}\vec{v'} \tag{8}$$

Gram matrix เป็น Symmetric โดย

ให้ $G = \hat{X}^T\hat{X}$ และ $G^T = (\hat{X}^T\hat{X})^T = \hat{X}^T(\hat{X}^T)^T = \hat{X}^T\hat{X} = \hat{X}^T(\hat{X})$

$\therefore G = G^T \therefore G$ เป็น Symmetric

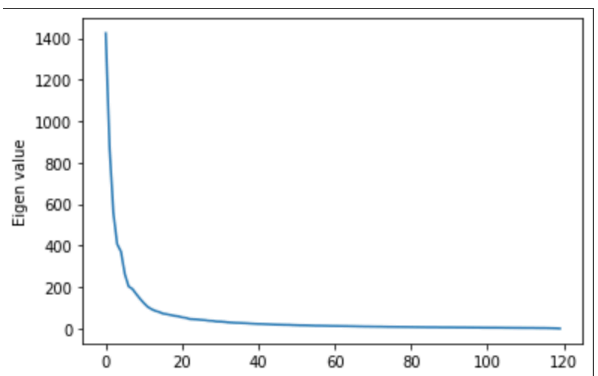
การแก้หา $\lambda, \vec{v}'$ อยู่ใน Code

T15. Compute the eigenvectors and eigenvalues of the Gram matrix, v' and λ. Sort the eigenvalues and eigenvectors in descending order so that the first eigenvalue is the highest, and the first eigenvector corresponds to the best direction. How many non-zero eigenvalues are there? If you see a very small value, it is just numerical error and should be treated as zero.

Non-zero eigenvalues มี 119 ตัว

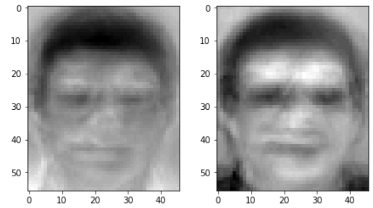
ค่า Eigenvalue ที่มากที่สุดคือ 1.42×10^3

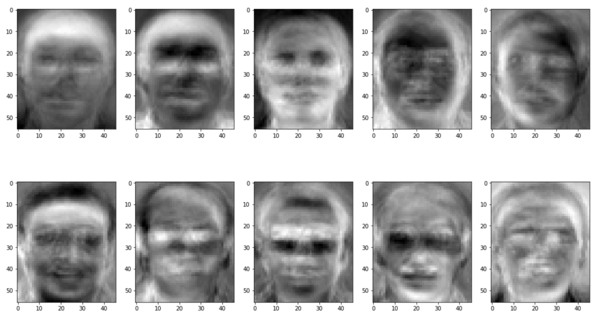
T16. Plot the eigenvalues. Observe how fast the eigenvalues decrease. In class, we learned that the eigenvalues is the size of the variance for each eigenvector direction. If I want to keep 95% of the variance in the data, how many eigenvectors should I use?



ถ้าหากต้องการ 95% จะต้องคง Eigenvector ไว้ 64 ตัว โดยดูจากจาก Cummulative of ratio-variance.

T17. Compute $\vec{v}$. Don't forget to renormalize so that the norm of each vector is 1 (you can use `numpy.linalg.norm`). Show the first 10 eigenvectors as images. Two example eigenvectors are shown below. We call these images eigenfaces (or eigenvoice for speech signals).

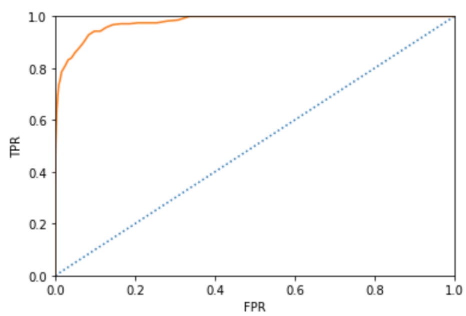




T18. From the image, what do you think the first eigenvector captures? What about the second eigenvector? Look at the original images, do you think biggest variance are capture in these two eigenvectors?

รูปแรกและรูปที่สองเก็บเกี่ยวกับลักษณะของใบหน้า ที่บ่งบอกว่าเป็นหน้าคนที่มีหน้า มีศรีษะ และ บริเวณตา แต่ไม่ได้บอกถึงลักษณะใบหน้า ลักษณะทรงผม

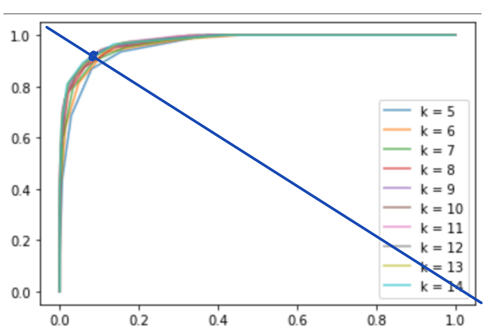
T19. Find the projection values of all images. Keep the first $k = 10$ projection values. Repeat the simple face verification system we did earlier using these projected values. What is the EER and the recall rate at 0.1% FAR?



EER ที่ 4.68

และที่ FAR = 0.1% Recall 0.56

T20. What is the k that gives the best EER? Try $k = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14$.



ที่ K=14 ให้ EER ที่ดีที่สุด

T21. In order to assure that S_W is invertible we need to make sure that S_W is full rank. How many PCA dimensions do we need to keep in order for S_W to be full rank? (Hint: How many dimensions does S_W have? In order to be of full rank, you need to have the same number of linearly independent factors)

ถ้า S_W จะเป็น Full rank ได้นั้น จำนวน Column และ Row จะต้องเท่ากับจำนวน Rank ที่เป็นไปได้

หา Rank ของ S_W

เนื่องจาก $S_W = \sum_{i=1}^{N_c} \sum_{j=1}^{N_i} (\vec{x}_j - \vec{\mu}_i)(\vec{x}_j - \mu_i)^T$

ถ้าพิจารณา Matric $(\vec{x}_j - \vec{\mu}_i)(\vec{x}_j - \mu_i)^T$ อย่างเดียวจะเป็น Matrix rank 1.

$\therefore$ S_W จะมี $N_i - 1$ linearly independent vector ในแต่ละ Class.

$\therefore$ พิจารณา S_W ทั้งหมด จะ มี rank คือ N-C ถ้า N คือจำนวนจุด และ C คือจำนวน Class.

ดังนั้น PCA dimension ที่เราต้องการคือ N-C หรือ 120-40 = 80

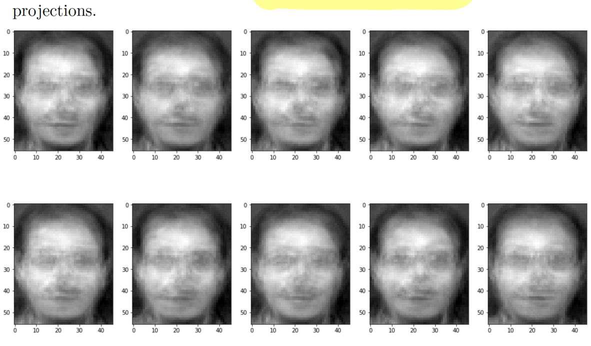
T22. Using the answer to the previous question, project the original input to the PCA subspace. Find the LDA projections. To find the inverse, use `numpy.linalg.inv`. Is $S_W^{-1}S_B$ symmetric? Can we still use `numpy.linalg.eigh`? How many non-zero eigenvalues are there?

Code implement ใน LDA class ซึ่งเราไม่สามารถสรุปได้ว่า $S_W^{-1}S_B$

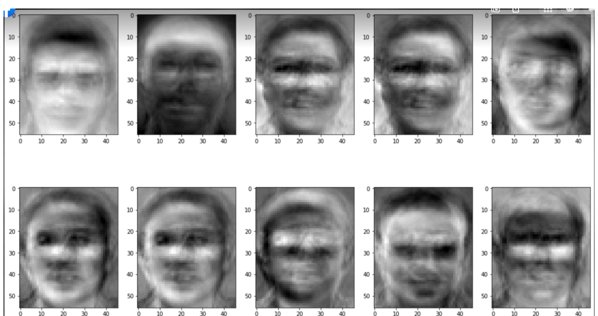
และมี Eigenvalues ที่ไม่เท่ากับ 0 ทั้งหมด 39 ตัว (ละทิ้งตัวที่น้อยมากๆ).

```
array([ 1.36917494e+07+0.00000000e+00j,  4.93597810e+03+0.00000000e+00j,
        1.44384617e+03+0.00000000e+00j,  7.48187352e+02+0.00000000e+00j,
        5.74469071e+02+0.00000000e+00j,  3.87183690e+02+0.00000000e+00j,
        2.48051664e+02+0.00000000e+00j,  1.71906841e+02+0.00000000e+00j,
        1.26585061e+02+0.00000000e+00j,  1.06607491e+02+0.00000000e+00j,
        7.67088363e+01+0.00000000e+00j,  7.22738617e+01+0.00000000e+00j,
        3.78927725e+01+0.00000000e+00j,  3.26640504e+01+0.00000000e+00j,
        3.18331255e+01+0.00000000e+00j,  2.49291868e+01+0.00000000e+00j,
        2.32673579e+01+0.00000000e+00j,  1.72583144e+01+0.00000000e+00j,
        1.43056638e+01+0.00000000e+00j,  1.25046335e+01+0.00000000e+00j,
        1.07063353e+01+0.00000000e+00j,  9.50022145e+00+0.00000000e+00j,
        8.02710020e+00+0.00000000e+00j,  6.57192475e+00+0.00000000e+00j,
        5.64011574e+00+0.00000000e+00j,  4.79227241e+00+0.00000000e+00j,
        4.47543662e+00+0.00000000e+00j,  4.09103860e+00+0.00000000e+00j,
        3.55918880e+00+0.00000000e+00j,  2.53766624e+00+0.00000000e+00j,
        2.22727717e+00+0.00000000e+00j,  1.87035204e+00+0.00000000e+00j,
        1.61566721e+00+0.00000000e+00j,  1.50186379e+00+0.00000000e+00j,
        1.08633260e+00+0.00000000e+00j,  7.93598556e-01+0.00000000e+00j,
        7.12275306e-01+0.00000000e+00j,  5.01212459e-01+0.00000000e+00j,
        3.50101376e-01+0.00000000e+00j,  2.93855811e-10+0.00000000e+00j,
       -2.51148700e-10+1.05527048e-10j, -2.51148700e-10-1.05527048e-10j,
        1.96670915e-10+0.00000000e+00j, -1.24147050e-10+1.27336761e-10j,
       -1.24147050e-10-1.27336761e-10j,  1.45510153e-10+9.26279018e-11j,
        1.45510153e-10-9.26279018e-11j,  2.97022228e-11+1.29151502e-10j,
        2.97022228e-11-1.29151502e-10j, -1.02834009e-10+6.60906942e-11j,
       -1.02834009e-10-6.60906942e-11j,  2.16935028e-11+1.08876276e-10j,
        2.16935028e-11-1.08876276e-10j,  1.08858735e-10+0.00000000e+00j,
       -9.15812869e-11+0.00000000e+00j,  8.22410948e-11+0.00000000e+00j,
       -2.32923437e-11+7.75880201e-11j, -2.32923437e-11-7.75880201e-11j,
        6.08072909e-11-3.46718998e-11j,  6.08072909e-11+3.46718998e-11j,
       -5.52178219e-11+0.00000000e+00j,  4.04537861e-11+9.58779991e-12j,
        4.04537861e-11-9.58779991e-12j, -2.70147807e-11+2.95760460e-11j,
       -2.70147807e-11-2.95760460e-11j, -7.33841905e-12+2.91620478e-11j,
       -7.33841905e-12-2.91620478e-11j,  2.47924605e-11+0.00000000e+00j,
       -1.82077785e-11+1.57404987e-11j, -1.82077785e-11-1.57404987e-11j,
       -2.34264824e-11+0.00000000e+00j,  1.63661797e-11+9.87984967e-12j,
        1.63661797e-11-9.87984967e-12j, -1.81494494e-11+0.00000000e+00j,
        5.27017126e-12+1.52842248e-11j,  5.27017126e-12-1.52842248e-11j,
```

T23. Plot the first 10 LDA eigenvectors as images (the 10 best projections). Note that in this setup, you need to convert back to the original image space by using the PCA projection. The LDA eigenvectors can be considered as a linear combination of eigenfaces. Compare the LDA projections with the PCA projections.



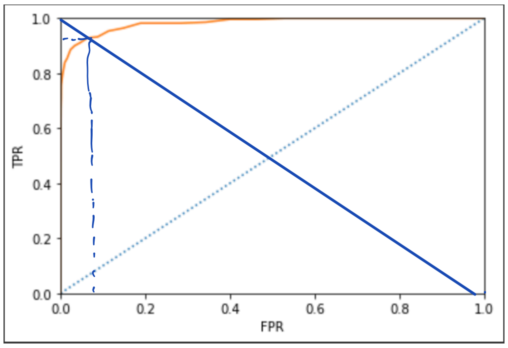
มี Mean face



ไม่มี Mean Face

ถ้าหากดูจากรูปที่ไม่มี Mean face คาดว่า LDA Component สื่อถึง ลักษณะของใบหน้า เช่น รูปหน้า ตา การส่องแว่นตา หรือลักษณะหน้าผาก.

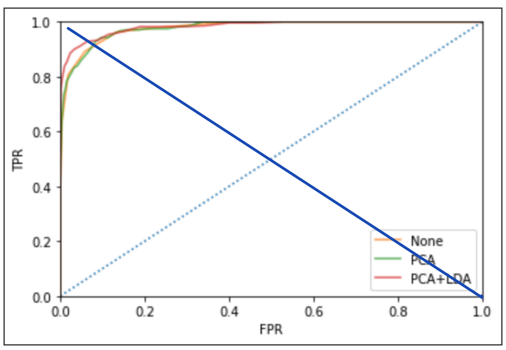
T24. The combined PCA+LDA projection procedure is called fisherface. Calculate the fisherfaces projection of all images. Do the simple face verification experiment using fisherfaces. What is the EER and recall rate at 0.1% FAR?



EER ที่

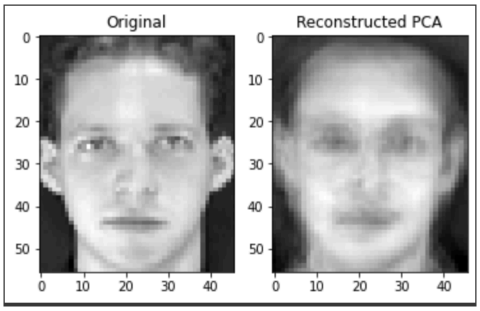
- EER ที่ t = 3.63
- Recall 0.71 at FAR = 0.1%

T25. Plot the RoC of all three experiments (No projection, PCA, and Fisher) on the same axes. Compare and contrast the three results. Submit your writeup and code on MyCourseVille.



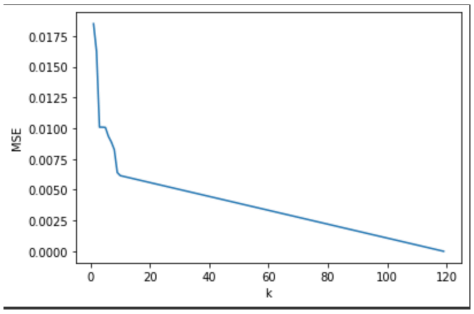
หากพิจารณาแล้วจะพบว่าผลของ PCA กับแบบปกติ จะให้ประสิทธิภาพที่ไม่ต่างกันโดยสังเกตจากการแนบชิดของ ROC curve แต่ถ้าเป็นการลดมิติแบบ PCA+LDA หรือ Fisher จะมีประสิทธิภาพดีกว่า โดยดูได้จาก ROC curve ที่จะพุ่งเข้ามุมซ้ายบน หรือ ดูจากที่ FPR ระดับเดียวกันให้ TPR ที่สูงกว่า

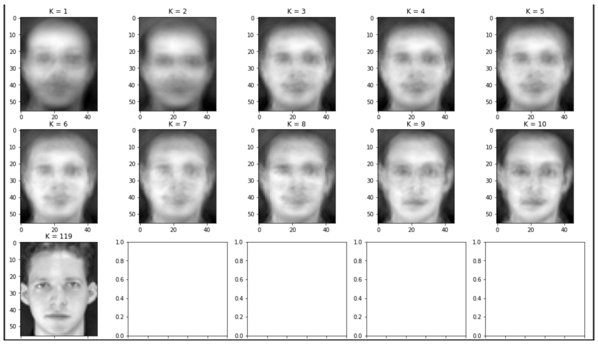
OT1. Reconstruct the first image using this procedure. Use $k = 10$, what is the MSE?



MSE = 0.006

OT2. For k values of 1,2,3,...,10,119, show the reconstructed images. Plot the MSE values.





OT3. Consider if we want to store 1,000,000 images of this type. How much space do we need? If we would like to compress the database by using the first 10 eigenvalues, how much space do we need? (Assume we keep the projection values, the eigenfaces, and the meanface as 32bit floats)

เดิม 1 รูป มี Float = 56*46 = 2,576 ตัว ~ 82.432 Kbit ~ 10.304 KB

ถ้ามี 1 M รูป ~ 10 GB

PCA 1 รูป มี Float 80 ตัว ~ 2 Kbit ~ 0.32 KB

ถ้ามี 1 M รูป ~ 0.32 GB.

OT4. Plot the first two LDA dimensions of the test images from different people (6 people 7 images each). Use a different color for each person. Observe the clustering of between each person. Repeat the same steps for the PCA projections. Does it come out as expected?