Pawin Piemthai - 5931037621

T1: the results obtained are as follow:

initialize mean:

[[ 3. 3.]

[ 2. 2.]

[-3. -3.]]

initialize cov:

[[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

initialize phi: [0.33333333 0.33333333 0.33333333]

Itteration: 0

Using mean: [[ 3. 3.]

[ 2. 2.]

[-3. -3.]]

Using covariant: [[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

Using phi: [0.33333333 0.33333333 0.33333333]

W matrix: [[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]]

Itteration: 1

Using mean: [[ 5.78992692 5.81887265]

[ 1.67718211 2.14523106]

[-4. -4.66666666]]

Using covariant: [[12.31988634, 0. ],

[ 0. , 12.23304914]]

[[0.62066718, 0. ],

[0. , 0.15261824]]

[[5.66666667, 0. ],

[0. , 5.66666668]]

Using phi: [0.45757242 0.20909425 0.33333333]

W matrix: [[1.81294622e-002 9.81582998e-001 2.87540002e-004]

[5.64494061e-001 4.35380622e-001 1.25316584e-004]

[1.92846943e-002 9.80633501e-001 8.18047710e-005]

[1.00000000e+000 4.70826685e-062 5.03915978e-012]

[9.99999990e-001 3.82283898e-027 1.01750382e-008]

[1.00000000e+000 7.32691444e-043 2.49006494e-010]

[1.59795152e-003 7.69753019e-045 9.98402048e-001]

[1.55069753e-003 5.89612118e-058 9.98449302e-001]

[3.59091243e-006 1.89005654e-144 9.99996409e-001]]

Itteration: 2

Using mean: [[ 6.30842698 6.31259558]

[ 1.77218759 2.1815904 ]

[-4.00062813 -4.66675525]]

Using covariant: [[3.24482139, 0. ],

[0. , 3.18737779]]

[[0.54812076, 0. ],

[0. , 0.14993733]]

[[4.67362081, 0. ],

[0. , 2.89766742]]

Using phi: [0.40056227 0.26639968 0.33303805]

W matrix: [[1.81240251e-004 9.99812736e-001 6.02360765e-006]

[1.40770128e-001 8.59229229e-001 6.42663215e-007]

[4.85324724e-004 9.99513546e-001 1.12958651e-006]

[1.00000000e+000 7.30562640e-064 3.40271752e-019]

[1.00000000e+000 4.88274872e-028 5.02342563e-014]

[1.00000000e+000 3.05867029e-044 1.26696752e-016]

[4.85980138e-012 2.29092578e-047 1.00000000e+000]

[3.11260608e-012 1.78986020e-060 1.00000000e+000]

[1.09172834e-023 1.82739807e-151 1.00000000e+000]]

Itteration: 3

Using mean: [[ 6.81963839 6.81969608]

[ 1.95082009 2.30058161]

[-3.9999862 -4.66664913]]

Using covariant: [[1.58836939, 0. ],

[0. , 1.58356938]]

[[0.67983404, 0. ],

[0. , 0.22439122]]

[[4.6667292 , 0. ],

[0. , 2.88899996]]

Using phi: [0.34904852 0.31761728 0.3333342 ]

W matrix: [[9.84227343e-009 9.99991670e-001 8.32048865e-006]

[1.82843704e-004 9.99817007e-001 1.49550814e-007]

[1.44413428e-007 9.99998536e-001 1.31922259e-006]

[1.00000000e+000 6.70321724e-043 1.72563002e-019]

[1.00000000e+000 1.86494636e-018 3.93009098e-014]

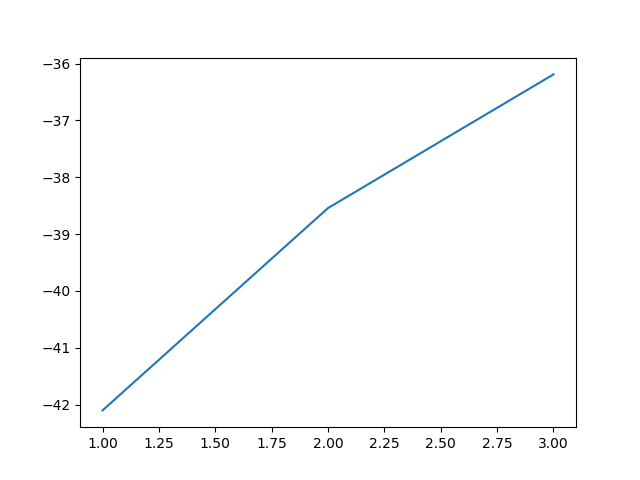
[1.00000000e+000 1.15283773e-029 5.80141001e-017]

[1.71803994e-026 1.54660928e-034 1.00000000e+000]

[8.26524801e-027 5.89169041e-043 1.00000000e+000]

[8.21355496e-052 3.03027464e-108 1.00000000e+000]]

T2: from the plot below, we can see that the likelyhood goes up every itteration.



T3: the results obtained are as follow:

initialize mean: [[ 3. 3.]

[-3. -3.]]

initialize cov: [[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

initialize phi: [0.5 0.5]

Itteration: 0

Using mean: [[ 3. 3.]

[-3. -3.]]

Using covariant: [[1., 0.],

[0., 1.]]

[[1., 0.],

[0., 1.]]

Using phi: [0.5 0.5]

W matrix: [[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]]

Itteration: 1

Using mean: [[ 4.50000001 4.66666667]

[-3.99999997 -4.66666663]]

Using covariant: [[9.16666668, 0. ],

[0. , 8.66666669]]

[[5.66666672, 0. ],

[0. , 5.66666677]]

Using phi: [0.66666666 0.33333334]

W matrix: [[9.94979696e-01 5.02030393e-03]

[9.99922646e-01 7.73541258e-05]

[9.98623856e-01 1.37614400e-03]

[1.00000000e+00 6.27933891e-12]

[9.99999994e-01 6.33185482e-09]

[1.00000000e+00 2.12626993e-10]

[2.77132409e-03 9.97228676e-01]

[2.45908782e-03 9.97540912e-01]

[1.30217751e-06 9.99998698e-01]]

Itteration: 2

Using mean: [[ 4.49739004 4.66243446]

[-3.9912655 -4.65434481]]

Using covariant: [[6.94971903, 0. ],

[0. , 5.94046426]]

[[4.72011919, 0. ],

[0. , 2.98099996]]

Using phi: [0.66652866 0.33347134]

W matrix: [[9.99840572e-01 1.59427561e-04]

[9.99999613e-01 3.86910944e-07]

[9.99967636e-01 3.23639718e-05]

[1.00000000e+00 2.77855078e-18]

[1.00000000e+00 1.61298139e-13]

[1.00000000e+00 7.52961598e-16]

[2.56424026e-04 9.99743576e-01]

[1.65413279e-04 9.99834587e-01]

[6.05550189e-09 9.99999994e-01]]

Itteration: 3

Using mean: [[ 4.49960686 4.66618544]

[-3.99986439 -4.66641867]]

Using covariant: [[6.9196161 , 0. ],

[0. , 5.89303254]]

[[4.66851519, 0. ],

[0. , 2.89184297]]

Using phi: [0.66669218 0.33330782]

W matrix: [[9.99878285e-01 1.21714513e-04]

[9.99999737e-01 2.62696505e-07]

[9.99975707e-01 2.42934485e-05]

[1.00000000e+00 9.72821019e-19]

[1.00000000e+00 7.60047112e-14]

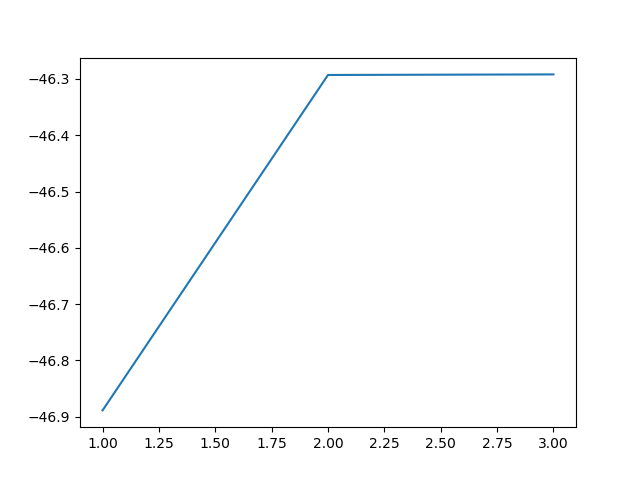
[1.00000000e+00 3.07475059e-16]

[2.42864008e-04 9.99757136e-01]

[1.53905587e-04 9.99846094e-01]

[5.29269940e-09 9.99999995e-01]]

T4: from the plot below, we can see that the 3 gaussians model has better likelyhood (at about -36 compare to -46).



T5:

distance between 0,0 and 0,1: 10.037616294165492

Distance between 0,0 and 1,0: 8.173295099737281

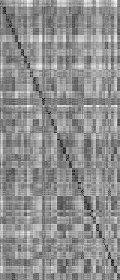
The number does not make sense and it is not useful to our classification.

From the data above, we can see that the distance between the picture of same person (0,0 and 0,1) has a greater distance from the picture of different people (0,0 and 1,0).

If we were to classify the picture 0,0, we would pick the person #2 as the answer since it has lesser distance, which is the wrong answer.

Hence, comparing direct distance is not always useful.

t6: the picture of the similarity matrix is placed below:



T7:

Black square represents the two pictures that have zero distance between them; hence, there are the same pictures

We can observe that the black square is in the diagonal of the matrix; that is when i == j, where i is the row number and j is the column number. (that is the same picture)

The square in the person 1 area has a dimmer shade than the square in the zone of person 2. That means pictures of the person 1 have a closer distance to other pictures in the same class. While the pictures of the person 2 have more distance to the other pictures in the same class.

T8:

Using threshold=10, we get the following result.

false alarm rate = 0.4564102564102564

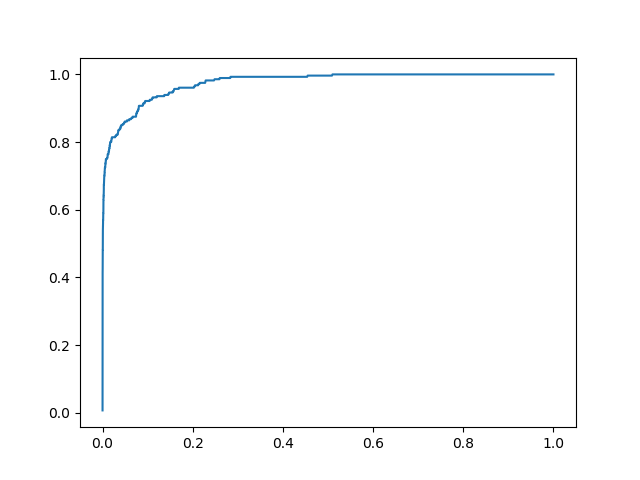
true possitive rate = 0.9964285714285714

T9:

Assume that we know nothing about the data, we can use the min and max of the data as the boundry of our search provided that the range of the data is small.

I use min=2, max=17 (obtained from the min and max of the similarity matrix), and 3000 equally spaced samples from within the boundry.

The ROC is below:



T10: the result is as follow:

ERR is at when FAR= 0.089 and FRR= 0.089 with threshold= 8.087771942985746

When FAR= 0.001 the recall= 0.54643

T11: The meanface is placed below:



T12:

The size of covariance matrix is 2576\*2576

The rank of covariance matrix is 119

T13

The size of Gram matrix is 120\*120

The rank of Gram matrix is 119

If we compute eigenvalues from the matrix, We expect to get 119 non-zero eigenvalues since the number of non zero eigenvalues should equal the rank of the matrix.

T14

the Gram matrix is symmetric since it is generated from the inner product of the same matrix.

Given matrix A, $(A^TA)\_{i,j} = \sum\_{k=1}^n A\_{i,k} \cdot A^T\_{k,j}$

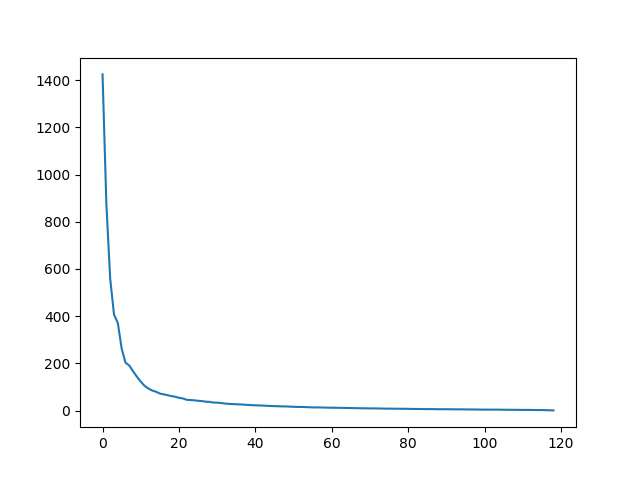
$(A^TA)\_{j,i} = \sum\_{k=1}^n A\_{j,k} \cdot A^T\_{k,i}$

From the definition of matrix transpose, $A\_{i,k} = A^T\_{k,i}$ and $A\_{j,k} = A^T\_{k,j}$

$\therefore (A^TA)\_{i,j} = (A^TA)\_{j,i}$ and hence symmetric.

T15: Using the calculation provided in the python file, there are 119 non zero eigenvalues.

T16: The plot of the eigenvalues is placed below:



From the calculation provided in the python file, We should use 64 eigenvalues to cover 95% variant of the data.

T17: The first 10 eigenfaces are placed below:

|  |  |
| --- | --- |
| Rank | image |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

T18:

The first eigen vector captures the overall structure and characteristic of the face and some part of the hair.

The second eigen vector captures a spesific part of the face, such as eyes, noses and mounth.

We can see that the biggest variant of the human face has been captured by those two eigen vectors. First is the overall face structure, which should be different between two people, and the second is a specific part of the face, which is a defining characteristic that helps us recognize which face it is.

T19: from the calculation in the python file, we get the following result:

PCA with k=10

ERR is at when FAR= 0.079 and FRR= 0.079 with threshold= 4.7869

When FAR= 0.001 the recall= 0.5142857

T20: From my calculation, the EER of k=10 and k=11 is very close:

Using k= 10

ERR is at when FAR= 0.07875 and FRR= 0.07857 with threshold= 4.7869

When FAR= 0.001 the recall= 0.5142857

Using k= 11

ERR is at when FAR= 0.07811 and FRR= 0.07857 with threshold= 4.96899

When FAR= 0.001 the recall= 0.50357

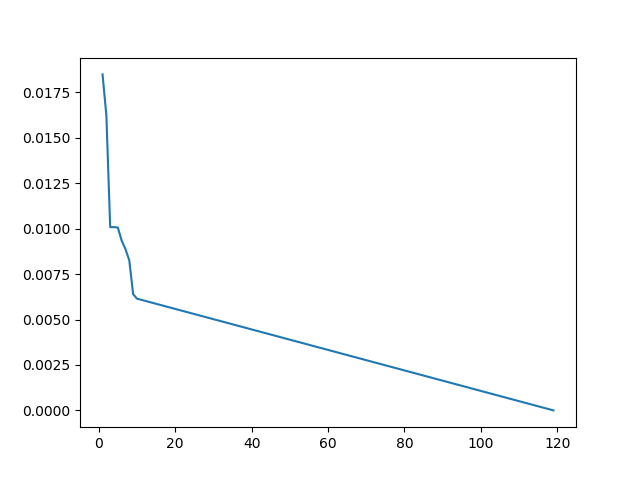
But when k=11, EER is slightly better. So, I would choose k=11 as the k that gives the best EER.

OT1: from the calculation in the python file, MSE for the first image when k= 10 : 0.0061483.

OT2: The reconstructed images and MSE plot is below:

|  |  |
| --- | --- |
| K | image |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 119 |  |

MSE plot



OT3:

Assume every pixel is represented by 32 bit float:

If we want to store 1,000,000 images: 1000000\*46\*56\*32 = 82432000000 bits = 10304000000 bytes = 9.596 Gb

If we compress the pictures with first 10 eigen vectors:

store eigen vectors: 2576\*10\*32

Store projected values: 1000000\*10\*32

Store meanface: 2576\*32

All: 2576\*10\*32 + 1000000\*10\*32 + 2576\*32 = 320906752 bits = 40113344 bytes = 38.255 Mb

T21: We need to keep 80 dimension.

T22:

Placing LDA projection here would be too long, but I have already find the LDA projection as a variable s\_realvec in the code at a later part when I'm doing LDA projection.

$S\_W^{-1}S\_B$ is not symmetric; therefore, we cannot use numpy.linalg.eigh. We have to use numpy.linalg.eig and handle the complex value and numerical instability ourselves.

There are 39 non-zero eigenvalues.

T23

|  |  |
| --- | --- |
| Rank | Image |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

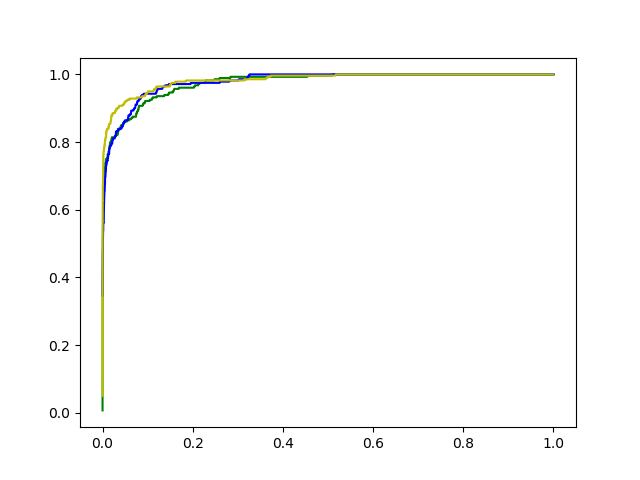
T24: The result is below:

ERR is at when FAR= 0.071 and FRR= 0.071 with threshold= 3.665666

When FAR= 0.001 the recall= 0.67857

T25:

The combine ROC plot is below:

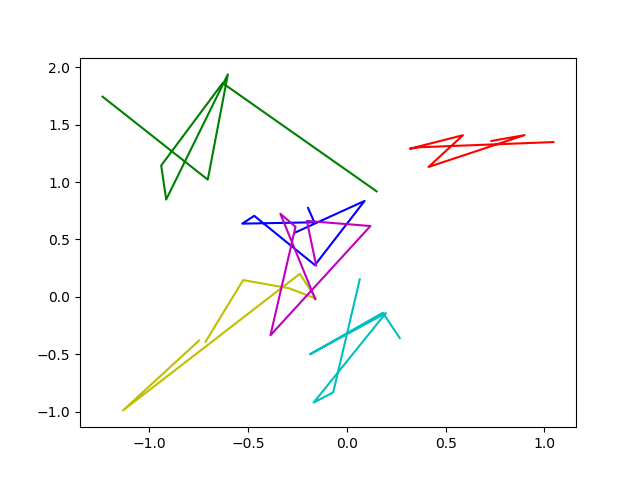


We can observe that the no projection experiment and the PCA experiment gives us the similar-looking curve. At some point, PCA gives more true possitive rate, but overall there is not much different.

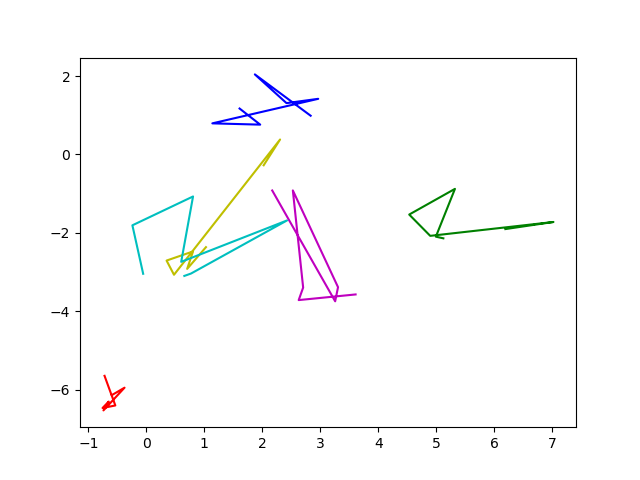
On the other hand, Fisherface gives us a much better curve both in term of EER and area under the curve.

OT4:

The LDA plot



The PCA plot



The plot are as we expected. The point in the LDA plot are more closely packed together in the same class and the distance between the classes are maximized.