

# 수치해석

## HW #6 (Project 1)

*face recognition*

2018007956 김채아

# face recognition

1. face images 를 모은다 (32x32size, 흑백 사진)
2. svd를 이용해서 basis vector (eigenface)를 구한다
3. eigenvalue가 큰 것부터 순서대로(svd가 해줌), eigenface의 linear combination으로 얼굴을 표현한다
4. 표현된 계수들의 조합으로 face recognition을 한다

## [Collect face images]

- 모든 face images에 대한 정보 :  
총 5749명에 대해 한 장 또는 여러 장의 사진이 있고  
총 13,233개이다

face recognition에 유리하게 변환:  
얼굴 인식이 안되는 것을 제외하고 변환했기 때문에  
총 10,141개이다



원본 파일

흑백변환  
&  
얼굴크롭  
&  
사이즈 32x32



수정 후 파일

## [이미지 수정 코드]

```
import os
import glob
import cv2_# OpenCV, 이미지 처리 라이브러리

face_cascade = cv2.CascadeClassifier(cv2.data.harcascades+'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.harcascades+'haarcascade_eye.xml')

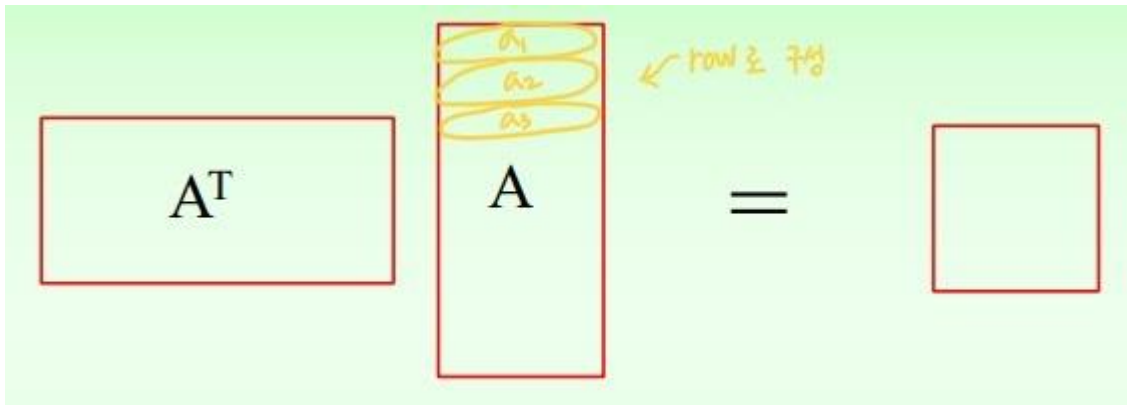
path = 'C:\\Users\\LG\\Desktop\\test\\*.jpg'
img = [cv2.imread(file) for file in glob.glob(path)]
name = [os.path.basename(file) for file in glob.glob(path)]
gray_img=[]
face=[]
cropped_img=[]
resize_img=[]
eyes=[]
for i in range(len(img)):
    gray_img.append(cv2.cvtColor(img[i], cv2.COLOR_BGR2GRAY))
    face.append(face_cascade.detectMultiScale(gray_img[i],scaleFactor=1.1, minNeighbors=1, minSize=(50, 50)))
    if len(face[i])!=1: # 얼굴이 1개로 인식된것만 수행
        for (x,y,w,h) in face[i]:
            cropped_img.append(gray_img[i][y:y+h, x:x+w])
            resize_img.append(cv2.resize(cropped_img[i],(32,32)))
            cv2.imwrite("C:\\Users\\LG\\Desktop\\test2\\%s"%(name[i]),resize_img[i])
    else:
        cropped_img.append(None)
        resize_img.append(None)
        eyes.append(None)
```

## [opencv의 SVD 과정 개념] \_ 알고리즘 구현 과정

0부터 255사이의 값을 갖는 32x32 이미지를 나열한  
1024차원의 벡터공간을 다룬다

얼굴 이미지는 0~255의 양수로 이루어진 벡터이므로,  
벡터 공간을 구성하기 위해 각 face image vector에 평균값을 빼준다  
이 값들로 data matrix A를 구성한다

취득한 데이터가 dimension보다 크므로, data matrix가 row로 구성되고  
covariance matrix =  $A^T @ A$  를 구해준다  
(matrix의 일부분에 basis가 나오니깐, 보다 작은 square matrix로 구해짐)



(row space의 basis vector는 U matrix로 구성된다)

## [python code]

```
def createDataMatrix(images):
    print('creating data matrix', end='.....')
    ...

    Allocate space for all images in one data matrix
    The size of the data matrix is
    ( w * h * 3, numImages )

    where,

    w = width of an image in the dataset
    h = height of an image in the dataset
    3 is for the 3 color channels
    ...

    numImages = len(images)
    sz = images[0].shape
    data = np.zeros((numImages, sz[0] * sz[1]), dtype=np.float32)
    for i in range(numImages):
        image = images[i].flatten()
        data[i,:] = image
    print('DONE')
    return data

path = 'C:\\Users\\LG\\Desktop\\grayfaces\\*.jpg'
img = [cv2.imread(file, cv2.IMREAD_GRAYSCALE) for file in glob.glob(path)]
img = np.array(img)
mean = img.mean(axis=0).astype("int")
A = img - mean
A = createDataMatrix(A)
matA = A.T @ A
U, s, V = np.linalg.svd(matA, full_matrices=True)
S = np.zeros(matA.shape)
for i in range(len(s)):
    S[i][i] = s[i]
appA = U @ S @ V
```

createDataMatrix함수는  
받은 이미지들을 row로 정렬해주는 함수이다  
(10139, 32, 32) → (10139, 1024)  
(한 이미지가 32x32사이즈이므로 1024차원이 된다)

10개 이미지의 img를 직접 표현해보면)

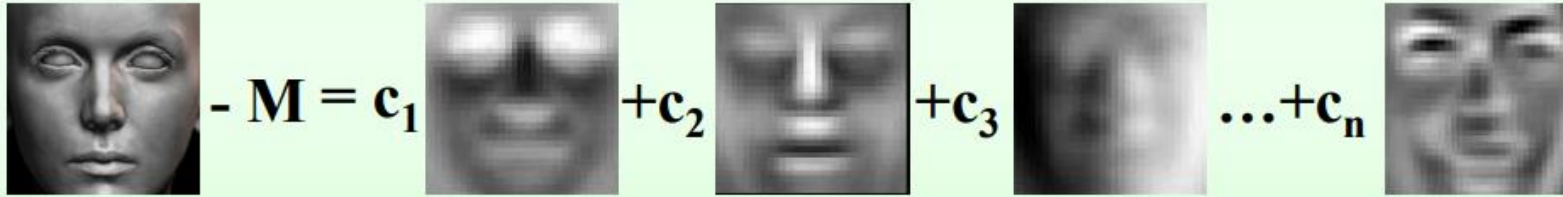
```
[img1벡터값] [img1벡터값] ... [img1벡터값]
[img10벡터값] [img10벡터값] ... [img10벡터값]
[img2벡터값] [img2벡터값] ... [img2벡터값]
.
.
[img9벡터값] [img9벡터값] ... [img9벡터값]
```

print 결과 :

```
[[ 0.47951833 -0.38754643  0.29055709 ...  0.19592439  0.53759052
  0.45575077]
 [-0.38754643 -1.28641351 -0.48882773 ... -0.30383423  0.11253765
 -0.02797979]
 [ 0.29055709 -0.48882773  0.26081932 ... -0.17834389  0.34344585
  0.18580788]
 ...
 [ 0.19592439 -0.30383423 -0.17834389 ...  0.04078646 -0.59109465
 -0.15732563]
 [ 0.53759052  0.11253765  0.34344585 ... -0.59109465 -1.19477596
 -0.7405358 ]
 [ 0.45575077 -0.02797979  0.18580788 ... -0.15732563 -0.7405358
 -0.40443502]]
```

두 행렬의 차이가 영행렬에 가까움을 알 수 있고  
이는 svd가 잘 되었다고 볼 수 있다

## [Find the Coefficients]


$$\text{original face image} - \mathbf{M} = c_1 \text{eigenface}_1 + c_2 \text{eigenface}_2 + c_3 \text{eigenface}_3 + \dots + c_n \text{eigenface}_n$$

$$c_k = \left[ \text{original face image} - \mathbf{M} \right] \cdot \text{eigenface}_k$$

(inner product of  $e_k$  and test face image vector)

( original face image vector – mean vector) @ 특정 eigenface vector  
= 1024차원 벡터  $C_k$ 가 얻어진다

## [코드]

```
# test face recognition
testFace = img[2]
testFaceMS = testFace - mean
testFaceMS = testFaceMS.flatten()
mean = mean.flatten()
r_list = [25, 50, 100, 200, 400, 800, 1600]
for r in r_list:
    # find the coefficients
    coefficients = U[:,r].T @ testFaceMS
    print(coefficients)

# generate face image using eigenfaces
reconFace = mean + U[:,r] @ coefficients
img = plt.imshow(np.resize(reconFace, (32, 32)))
img.set_cmap('gray')
plt.title('r = ' + str(r))
plt.axis('off')
plt.show()
```

=>코드에서는 eigenface vector @ testface(image vector–mean vector)하여 입력된 r만큼의 coefficient벡터를 만든다  
r이상의 coefficient는 다루지 않기때문에 이렇게 만들어도 상관없고 오히려 더 효율적이다



## [coefficients 비교]

〈재구성한 사진의 벡터구조 분석〉 – recognition performance test

● 같은 사람의 디테일(표정, 안경 등)이 다른 사진 분석 1-①

(같은 사람의 다른 사진이 많이 없어서 따로 찾아서 넣었다)

차이 : 안경 유무

★ eigenfaces : 100일 경우, 각각의 계수 벡터 값과 차이

```
1 : [ 504  728  45 163 -223  75  14 172  82 -101 -143 -23 -78  0
    -207 153  11 -274 -125 -27 109  98 -295 -41 -19 -62 -94 -169
      28  39 -75  -3 152  90 153  -1 -11 -42  35 -23  48 -90
    -29  15  33 100  64 -74  42  87 -59 -1 -89 -82 -14 -32
      3 -19 -26 -58  2  89  70  26  1  87 -109  91 -25 -77
    -15 -38 -18 -15 -113 -21  30 -29 -40 -17 -13  83  88  30
      9 -61  30  -6  23 -61 -27  60  24 -23 -23 -37  47 -11
     26 -18]
2 : [ 519 -110 -344  7 -505  19  98 306 -64  15 -254  0  48 -73
    -306 262 -206 -181 157 -89 164  66 -298 -54 -61  17 -60 -66
      42 -190 -260  44  10 -113  98 105  10  40 -51 108 -65  0
      96 -34 -37 127  35 100  61  10 -76 -48  84  81 -47 -68
      96  -4  4  -2 -21 114 -16 -26 -152  89 -35 -106 107  49
     -12 -33 -87 -70 -85  57  -9  27  78  98 -95  76 -26  -7
      0  82 -67  -5 -32  62 -29  73  58  77 -33 -23  18 -44
     16 -92]
1-2 : [ -15  838  389 156 282  56 -84 -134 146 -116 111 -23 -126  73
      99 -109 217 -93 -282  62 -55  32  3  13  42 -79 -34 -103
     -14 229 185 -47 142 203  55 -106 -21 -82  86 -131 113 -90
    -125  49  70 -27  29 -174 -19  77  17  47 -173 -163  33  36
     -93 -15 -30 -56  23 -25  86  52 153  -2 -74 197 -132 -126
      -3  -5  69  55 -28 -78  39 -56 -118 -115  82  7 114  37
      9 -143  97  -1  55 -123  2 -13 -34 -100  10 -14  29  33
     10  74]
```

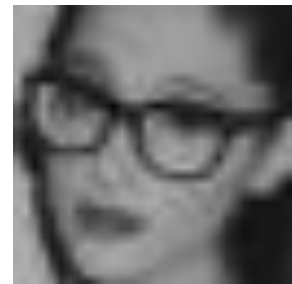
두 벡터의 차이 값이 비교적 작은 걸로 보아  
안경의 유무에 상관없이  
같은 사람을 인식할 수 있다는 것을 알 수 있다

원본 이미지

1



2



→ rank(eigenfaces개수)에 따라  
이미지의 distance(energy)를 구하고  
그 차이 값을 계산  
- distance : 각 벡터 값의 제곱의 합

```
rank : 25
image 1) distance : 1210611
image 2) distance : 1231346
distance difference : 20735
rank : 50
image 1) distance : 1359594
image 2) distance : 1443676
distance difference : 84082
rank : 100
image 1) distance : 1484788
image 2) distance : 1653416
distance difference : 168628
rank : 200
image 1) distance : 1598139
image 2) distance : 1849969
distance difference : 251830
rank : 400
image 1) distance : 1687431
image 2) distance : 1952414
distance difference : 264983
rank : 800
image 1) distance : 1750616
image 2) distance : 2015445
distance difference : 264829
```



## [coefficients 비교]

### 〈재구성한 사진의 벡터구조 분석〉 – recognition performance test

● 같은 사람의 디테일(표정, 안경 등)이 다른 사진 분석표정 변화 1-②

차이 : 표정

★ eigenfaces : 100일 경우, 각각의 계수 벡터 값과 차이

```
1 : [ 504  728   46  163 -223   75   14  172   82 -101 -143  -22  -79   0
    -205  154    8 -275 -124  -26  107   97 -296  -41  -19  -63  -94 -170
      28   39  -76   -6  153   87  154    0  -11  -42   36  -24   44  -91
     -29   15   36   98   63  -76   39   84  -64    0  -89  -84  -14  -32
       5  -21  -30  -54   -1   86   74   24   22   85 -112   85  -27  -80
     -12  -38  -19  -13 -113  -17   30  -28  -40  -19  -13   85   85   28
       7  -61   31   -6   19  -65  -24  -59   27  -22  -26  -36  -46  -13
      26  -20]
2 : [-1392 -187 -202 -117 -738  176 -198   67  355   65  149   38
     109 -131  176  -95  409 -192  -30  212   58   87 -260  -68
       7    47  -82  -20  -92 -136  100  187  109  -61  -62  124
     156 -177   -9  -19  -23  102  112  -27 -118  -76  158  113
     137   65  167  105   73   58 -120  -95   58  -49   76  -66
     111  133  -61  -33   -3  107   61  -98   48  -83  -69  -56
      16   57  -41   42   10   79   60 -110  -60   58  -79   90
      45   34   43 -115   14  -73   24  -87   14  -66  -16   25
      35   -3  -59  127]
1-2 : [1896  915  248  280  515 -101  212  105 -273 -166 -292  -60 -188  131
     -381  249 -401  -83  -94 -238   49   10  -36   27  -26 -110  -12 -150
      120  175 -176 -193   44  148  216 -124 -167  135   45   -5   67 -193
     -141   42  154  174  -95 -189  -98   19 -231 -105 -162 -142  106   63
     -53   28 -106   12 -112  -47  135   57   25  -22 -173  183  -75   3
      57   18  -35  -70  -72  -59   20 -107 -100   91   47   27  164  -62
     -38  -95  -12  109    5    8  -48   28   13  44  -10  -61  -81  -10
      85 -147]
```

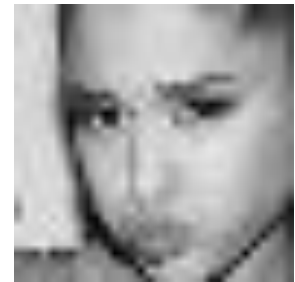
distance difference 결과를 보면  
다른 사람처럼 인식하고 있다  
너무 찡그린 사진은 벡터 차이 값을 가지곤  
판별할 수 없다

원본 이미지

1



2



→ rank(eigenfaces개수)에 따라  
이미지의 distance(energy)를 구하고  
그 차이 값을 계산  
- distance : 각 벡터 값의 제곱의 합

```
rank : 25
image 1) distance : 1210611
image 2) distance : 3204984
distance difference : 1994373
rank : 50
image 1) distance : 1359594
image 2) distance : 3484384
distance difference : 2124790
rank : 100
image 1) distance : 1484788
image 2) distance : 3753446
distance difference : 2268658
rank : 200
image 1) distance : 1598139
image 2) distance : 3954381
distance difference : 2356242
rank : 400
image 1) distance : 1687431
image 2) distance : 4118028
distance difference : 2430597
rank : 800
image 1) distance : 1750616
image 2) distance : 4212912
distance difference : 2462296
```

## [coefficients 비교]

### 〈재구성한 사진의 벡터구조 분석〉 – recognition performance test

● 같은 사람의 디테일(표정, 안경 등)이 다른 사진 분석표정 변화 1-③

차이 : 헤어스타일

★ eigenfaces : 100일 경우, 각각의 계수 벡터 값과 차이

```
1 : [-659 188 563 -707 -253 278 -180 282 155 -315 -78 -1 -101 -112
    39 -193 240 -1 -39 -35 -159 -180 13 -152 73 -83 -74 -90
    75 74 101 -13 94 -95 -107 18 47 68 2 -140 -51 5
    -33 54 -125 -64 64 -23 23 -22 149 85 -2 107 -32 -95
    5 3 57 -29 134 -115 -51 53 10 40 5 15 -45 108
    75 -65 18 -26 -95 9 -2 -42 -16 -42 112 29 -31 4
    29 -18 -37 9 -69 -9 75 38 9 -26 23 -17 -7 4
    -25 57]
2 : [-1150 -246 -121 99 135 -236 251 -90 177 -84 92 -234
    11 -183 -100 -178 255 -37 -188 260 -80 124 40 -202
    79 -71 -111 -144 -95 -29 -108 -7 124 -43 -19 -104
    -48 -26 131 -125 -201 116 61 -19 -74 17 -72 277
    211 -9 92 122 150 51 8 10 -17 56 14 -26
    151 152 -31 62 70 133 -66 -57 -51 6 18 -140
    82 -62 -23 -3 -99 64 86 11 6 70 -54 71
    -59 -69 -41 -120 -64 -53 102 -35 -78 -92 -83 -5
    78 94 24 -3]
1-2 : [ 491 434 684 -806 -388 514 -431 372 -22 -231 -170 233 -112 71
    139 -15 -15 36 149 -295 -79 -304 -27 50 -6 -12 37 54
    170 103 209 -6 -30 -52 -88 122 95 94 -129 -15 150 -111
    -94 73 -51 -81 136 -300 -188 -13 57 -37 -152 56 -40 -105
    22 -53 43 -3 -17 -267 -20 -9 -60 -93 71 72 6 102
    57 75 -64 36 -72 12 97 -106 -102 -53 106 -41 23 -67
    88 51 4 129 -5 44 -27 73 87 66 106 -12 -85 -90
    -49 60]
```

두 벡터의 차이 값이 비교적 작은 걸로 보아  
앞머리의 유무, 헤어스타일의 차이에 상관없이  
같은 사람임을 인식할 수 있다는 것을 알 수 있다

원본 이미지

1



2



→ rank(eigenfaces개수)에 따라  
이미지의 distance(energy)를 구하고  
그 차이 값을 계산  
- distance : 각 벡터 값의 제곱의 합

# [eigenfaces 개수에 따른 이미지 차이]

25

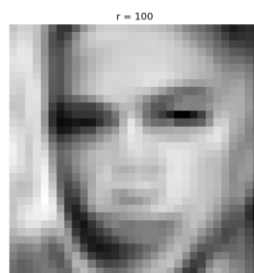
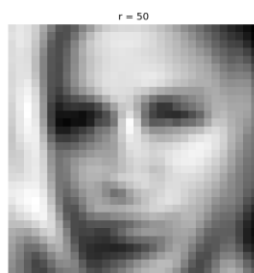
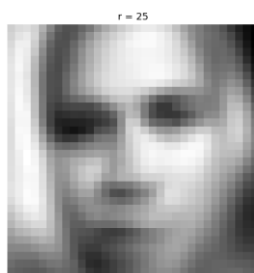
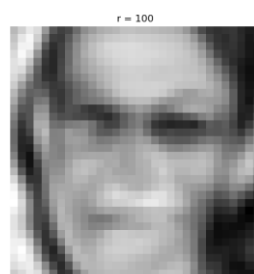
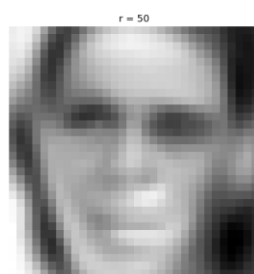
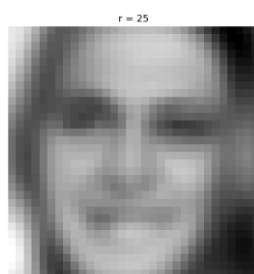
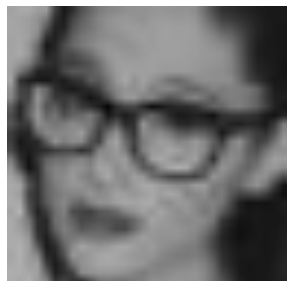
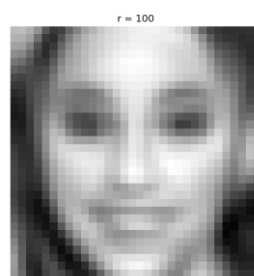
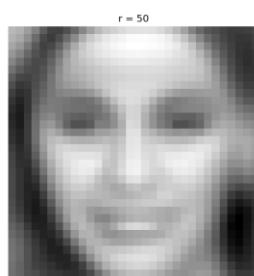
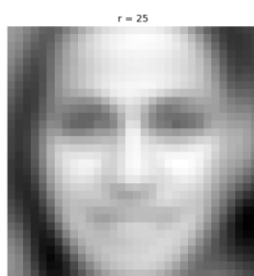
50

100

200

400

800



## [coefficients 비교]

### 〈재구성한 사진의 벡터구조 분석〉 – recognition performance test

#### ● 서로 다른 사람의 벡터 구조 비교 2-①

#### ★ eigenfaces :100일 경우, 각각의 계수 벡터 값과 차이

```
1 : [ 504  728   45  163 -223   75   14  172   82 -100 -143  -22  -78   0
    -205  154   11 -275 -124  -27  107   97 -296  -43  -18  -63  -94 -169
       28   39  -76   -6  151   88  154   0  -11  -42   35  -22   48  -91
     -29   15   37   98   65  -74   41   85  -63   -1  -88  -83  -14  -32
       5  -20  -29  -55   0   88   73   27   39   79 -110   86  -27  -79
      -7  -39  -18  -15 -114  -18   31  -29  -40  -19  -10   84   86   29
       7  -60   32   -5   19  -65  -25   60   25  -25  -23  -36  -47  -12
      27  -20]
2 : [-659  188  563 -707 -253  278 -180  282  155 -315  -78   -1 -101 -112
     39 -193  240   -1  -39  -35 -159 -180   13 -152   73  -83  -74  -90
     75   74  101  -13   94  -95 -107   18   47   68   2 -140  -51   5
    -33   54 -125  -64   64  -23   23  -22  149   85   -2  107  -32  -95
       5    3   57  -29  134 -115  -51   53   10   40   5   15  -45  108
     75  -65   18  -26  -95    9   -2  -42  -16  -42  112   29  -31   4
     29  -18  -37    9  -69   -9   75   38    9  -26   23  -17   -7   4
    -25   57]
1-2 : [1163  540 -518  870   30 -203  194 -110  -73  215  -65  -21  23  112
     -244  347 -229 -274  -85    8  266  277 -309  109  -91   20  -20  -79
     -47  -35 -177    7   57  183  261  -18  -58 -110   33  118   99  -96
        4  -39  162  162    1  -51   18  107 -212  -86  -86 -190   18   63
        0  -23  -86  -26 -134  203  124  -26   29   39 -115   71   18 -187
     -82   26  -36   11  -19  -27   33   13  -24   23 -122   55  117   25
     -22  -42   69  -14   88  -56 -100   22   16    1  -46  -19  -40  -16
       52  -77]
```

같은 사람을 테스트 했던 앞 사례와 비교해보면  
distance difference 값이 더 크게 나오는 것을  
볼 수 있다

```
rank : 25
image 1) distance : 1210611
image 2) distance : 1876569
distance difference : 665958
rank : 50
image 1) distance : 1359594
image 2) distance : 2006540
distance difference : 646946
rank : 100
image 1) distance : 1484788
image 2) distance : 2171840
distance difference : 687052
rank : 200
image 1) distance : 1598139
image 2) distance : 2261247
distance difference : 663108
rank : 400
image 1) distance : 1687431
image 2) distance : 2331837
distance difference : 644406
rank : 800
image 1) distance : 1750616
image 2) distance : 2376424
distance difference : 625808
```

원본 이미지

1



2



→ rank(eigenfaces 개수)에 따라  
이미지의 distance(energy)를 구하고  
그 차이 값을 계산  
- distance : 각 벡터 값의 제곱의 합



## [coefficients 비교]

### <재구성한 사진의 벡터구조 분석> - recognition performance test

#### ● 서로 다른 사람의 벡터 구조 비교 2-②

#### ★ eigenfaces :100일 경우, 각각의 계수 벡터 값과 차이

```
1 : [ 504  728   44  164 -223   76   14  172   81 -101 -143  -23  -78   0
    -208  154   18 -273 -123  -30  108   98 -296  -40  -18  -62  -94 -169
       28   38  -74   -4  152   89  153   -2  -11  -42   35  -23   48  -90
    -29   15   33  100   65  -74   42   89  -56   -2  -87  -83  -14  -32
       4  -19  -24  -57    7   94   68   25  -15   85 -109   88  -26  -78
    -18  -37  -19  -15 -108  -36   32  -29  -39  -17  -11   83   87   32
       6  -62   30   -7   22  -61  -26  -60   25   24  -23  -37   47   11
      25  -18]
2 : [ -720  -666   560 -1860   709   549   247  -112    41  -212   288  -133
      -25     0  -171   148   271   -72  -365  -38   184   23   51  -73
    -338  -25  -120    36   -64   169  -86   93  -83  -31  -36 -130
      123  -29    95   -46  -95  -74   80   49   42    3   10  141
       -1  -93   128   154   87 -115  -24   -1  -71    3 -134   54
    -111 -109  -48   93   31  -54  -28   24  -60  -10   88  -19
      -14   56   216 -106   96   38  -15  179   99  -10  -52   87
       17  -68   97   83   -7    0   37   14   21   11  -35   35
      48   70   45  -44]
1-2 : [1224 1394 -516 2024 -932 -473 -233  284   40  111 -431  110  -53   0
      -37    6 -253 -201  242    8  -76   75 -347   33  320  -37   26 -205
        92 -131   12  -97  235  120  189  128 -134  -13  -60   23  143  -16
    -109  -34   -9   97   55 -215   43  182 -184 -156 -174   32   10  -31
        75  -22  110 -111  118  203  116  -68  -46  139  -81   64   34  -68
    -106  -18   -5  -71 -324   70  -64  -67  -24 -196 -110   93  139  -55
      -11    6  -67  -90   29  -61  -63  -74    4   13   12  -72  -1  -59
      -20   26]
```

인종도 다르고, 성별도 다른 두 사람을 비교해 보았다  
지금까지의 결과 중 벡터 차이 값이  
제일 크게 나오는 것을 볼 수 있다

이렇게 큰 차이가 있는데도  
변함이 크지 않은 계수는 0으로 만든다

```
rank : 25
image 1) distance : 1210611
image 2) distance : 6181672
distance difference : 4971061
rank : 50
image 1) distance : 1359594
image 2) distance : 6353558
distance difference : 4993964
rank : 100
image 1) distance : 1484788
image 2) distance : 6654474
distance difference : 5169686
rank : 200
image 1) distance : 1598139
image 2) distance : 6806226
distance difference : 5208087
rank : 400
image 1) distance : 1687431
image 2) distance : 6902383
distance difference : 5214952
rank : 800
image 1) distance : 1750616
image 2) distance : 6943313
distance difference : 5192697
```

원본 이미지

1



2



→ rank(eigenfaces개수)에 따라  
이미지의 distance(energy)를 구하고  
그 차이 값을 계산  
- distance : 각 벡터 값의 제곱의 합

## [face recognition idea]

idea 1 : 사람이 달라져도 크게 바뀌지 않는 계수는 무시한다

2-② 의 계수 벡터 값 차이가 20이하이면  
그 값들은 0으로 처리한다

```
1-2 : [1224 1394 -516 2024 -932 -473 -233 284 40 111 -431 110 -53 0
-37 6 -253 -201 242 8 -76 75 -347 33 320 -37 26 -205
92 -131 12 -97 235 120 189 128 -134 -13 -60 23 143 -16
-109 -34 -9 97 55 -215 43 182 -184 -156 -174 32 10 -31
75 -22 110 -111 118 203 116 -68 -46 139 -81 64 34 -68
-106 -18 -5 -71 -324 70 -64 -67 -24 -196 -110 93 139 -55
-11 6 -67 -90 29 -61 -63 -74 4 13 12 -72 -1 -59
-20 26]
```

```
zero=[]
for i in range(len(coefficients)):
    if abs(coefficients[i] - coefficients2[i])<20:
        zero.append(i)
```

결과 : 다른 사람의 경우(예시②) 차이가 더 커졌다

```
rank : 25
image 1) distance : 1210611
image 2) distance : 6181672
distance difference : 4971061
rank : 50
image 1) distance : 1359594
image 2) distance : 6353558
distance difference : 4993964
rank : 100
image 1) distance : 1484788
image 2) distance : 6654474
distance difference : 5169686
rank : 200
image 1) distance : 1598139
image 2) distance : 6806226
distance difference : 5208087
rank : 400
image 1) distance : 1687431
image 2) distance : 6902383
distance difference : 5214952
rank : 800
image 1) distance : 1750616
image 2) distance : 6943313
distance difference : 5192697
```



```
rank : 25
image 1) distance : 1185995
image 2) distance : 6158324
distance difference : 4972329
rank : 50
image 1) distance : 1318549
image 2) distance : 6314733
distance difference : 4996184
rank : 100
image 1) distance : 1433998
image 2) distance : 6605512
distance difference : 5171514
rank : 200
image 1) distance : 1524787
image 2) distance : 6737420
distance difference : 5212633
rank : 400
image 1) distance : 1587621
image 2) distance : 6805694
distance difference : 5218073
rank : 800
image 1) distance : 1617677
image 2) distance : 6819183
distance difference : 5201506
```



## [face recognition idea]

idea 1 : 사람이 달라져도 크게 바뀌지 않는 계수는 무시한다

2-② 의 계수 벡터 값 차이가 20이하이면  
그 값들은 0으로 처리한다

```
1-2 : [1224 1394 -516 2024 -932 -473 -233 284 40 111 -431 110 -53 0
-37 6 -253 -201 242 8 -76 75 -347 33 320 -37 26 -205
92 -131 12 -97 235 120 189 128 -134 -13 -60 23 143 -16
-109 -34 -9 97 55 -215 43 182 -184 -156 -174 32 10 -31
75 -22 110 -111 118 203 116 -68 -46 139 -81 64 34 -68
-106 -18 -5 -71 -324 70 -64 -67 -24 -196 -110 93 139 -55
-11 6 -67 -90 29 -61 -63 -74 4 13 12 -72 -1 -59
-20 26]
```

```
zero=[]
for i in range(len(coefficients)):
    if abs(coefficients[i] - coefficients2[i])<=20:
        zero.append(i)
```

결과 : 같은 사람의 경우(예시①) 차이가 더 작아졌다

```
rank : 25
image 1) distance : 1210611
image 2) distance : 1231346
distance difference : 20735
rank : 50
image 1) distance : 1359594
image 2) distance : 1443676
distance difference : 84082
rank : 100
image 1) distance : 1484788
image 2) distance : 1653416
distance difference : 168628
rank : 200
image 1) distance : 1598139
image 2) distance : 1849969
distance difference : 251830
rank : 400
image 1) distance : 1687431
image 2) distance : 1952414
distance difference : 264983
rank : 800
image 1) distance : 1750616
image 2) distance : 2015445
distance difference : 264829
```



```
rank : 25
image 1) distance : 867379
image 2) distance : 870477
distance difference : 3098
rank : 50
image 1) distance : 1013814
image 2) distance : 1077405
distance difference : 63591
rank : 100
image 1) distance : 1102681
image 2) distance : 1250861
distance difference : 148180
rank : 200
image 1) distance : 1202642
image 2) distance : 1432691
distance difference : 230049
rank : 400
image 1) distance : 1270258
image 2) distance : 1505453
distance difference : 235195
rank : 800
image 1) distance : 1301940
image 2) distance : 1539930
distance difference : 237990
```

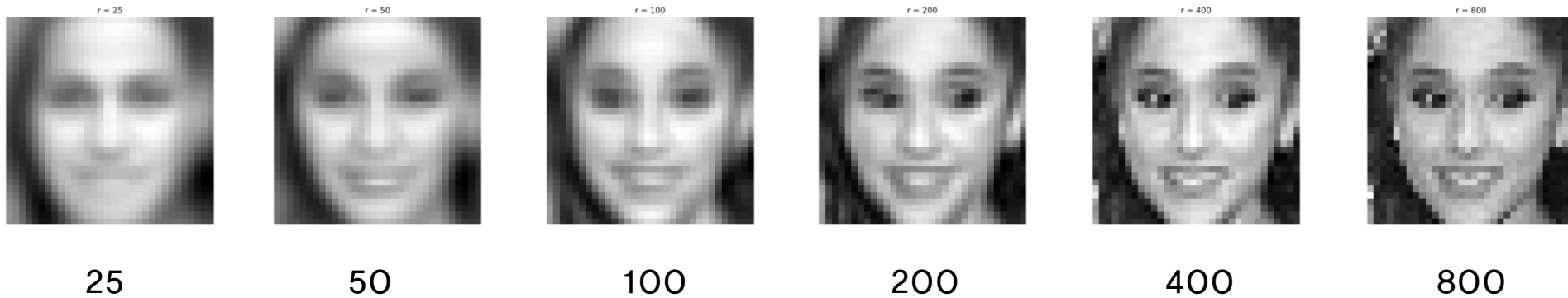
## [face recognition idea]

idea 1 : 사람이 달라져도 크게 바뀌지 않는 계수는 무시한다

이 idea를 통해 이론상으로, 이전에 비해 적은 eigenfaces로도 얼굴을 잘 구현할 수 있다  
하지만 20이하의 차이를 무시하는 것으로 육안으로 드러날 만큼의 차이까진 없었다



크게 바뀌지 않는 계수 무시 전



크게 바뀌지 않는 계수 무시 후



## [face recognition idea]

idea 2 : 두 벡터 간의 distance가 특정범위이면 같은 사람, 아니면 다른 사람이라고 판별한다

### 같은 사람

```
rank : 25
image 1) distance : 1210611
image 2) distance : 1231346
distance difference : 20735
rank : 50
image 1) distance : 1359594
image 2) distance : 1443676
distance difference : 84082
rank : 100
image 1) distance : 1484788
image 2) distance : 1653416
distance difference : 168628
rank : 200
image 1) distance : 1598139
image 2) distance : 1849969
distance difference : 251830
rank : 400
image 1) distance : 1687431
image 2) distance : 1952414
distance difference : 264983
rank : 800
image 1) distance : 1750616
image 2) distance : 2015445
distance difference : 264829
```

```
rank : 25
image 1) distance : 1210611
image 2) distance : 3204984
distance difference : 1994373
rank : 50
image 1) distance : 1359594
image 2) distance : 3484384
distance difference : 2124790
rank : 100
image 1) distance : 1484788
image 2) distance : 3753446
distance difference : 2268658
rank : 200
image 1) distance : 1598139
image 2) distance : 3954381
distance difference : 2356242
rank : 400
image 1) distance : 1687431
image 2) distance : 4118028
distance difference : 2430597
rank : 800
image 1) distance : 1750616
image 2) distance : 4212912
distance difference : 2462296
```

```
rank : 25
image 1) distance : 1876569
image 2) distance : 1968510
distance difference : 91941
rank : 50
image 1) distance : 2006540
image 2) distance : 2282946
distance difference : 276406
rank : 100
image 1) distance : 2171840
image 2) distance : 2564099
distance difference : 392259
rank : 200
image 1) distance : 2261247
image 2) distance : 2730136
distance difference : 468889
rank : 400
image 1) distance : 2331837
image 2) distance : 2873833
distance difference : 541996
rank : 800
image 1) distance : 2376424
image 2) distance : 2973678
distance difference : 597254
```

### 다른 사람

```
rank : 25
image 1) distance : 1210611
image 2) distance : 1876569
distance difference : 665958
rank : 50
image 1) distance : 1359594
image 2) distance : 2006540
distance difference : 646946
rank : 100
image 1) distance : 1484788
image 2) distance : 2171840
distance difference : 687052
rank : 200
image 1) distance : 1598139
image 2) distance : 2261247
distance difference : 663108
rank : 400
image 1) distance : 1687431
image 2) distance : 2331837
distance difference : 644406
rank : 800
image 1) distance : 1750616
image 2) distance : 2376424
distance difference : 625808
```

```
rank : 25
image 1) distance : 1210611
image 2) distance : 6181672
distance difference : 4971061
rank : 50
image 1) distance : 1359594
image 2) distance : 6353558
distance difference : 4993964
rank : 100
image 1) distance : 1484788
image 2) distance : 6654474
distance difference : 5169686
rank : 200
image 1) distance : 1598139
image 2) distance : 6806226
distance difference : 5208087
rank : 400
image 1) distance : 1687431
image 2) distance : 6902383
distance difference : 5214952
rank : 800
image 1) distance : 1750616
image 2) distance : 6943313
distance difference : 5192697
```

=> 위 결과를 통해 인식범위를 600000으로 잡는다 (마지막 rank 800의 distance difference기준)  
너무 핑그린 얼굴은 같은 사람으로 판별하지 못해서 성능은 좀 떨어지지만  
보통의 경우에는 같은 사람인지 다른 사람인지 판별할 수 있다

## [face recognition]

testcase 1 :



결과 : 두 사람은 같은 사람입니다.

testcase 2 :



결과 : 두 사람은 다른 사람입니다.

testcase 3 :



결과 : 두 사람은 같은 사람입니다.

너무 찡그린 사진은 다른 사람으로 인식한다  
→ performance 떨어지지만,  
위 두 사진은 눈으로 봐도 많이 다르다

testcase 4 :



결과 : 두 사람은 다른 사람입니다.

testcase 5 :



결과 : 두 사람은 다른 사람입니다.