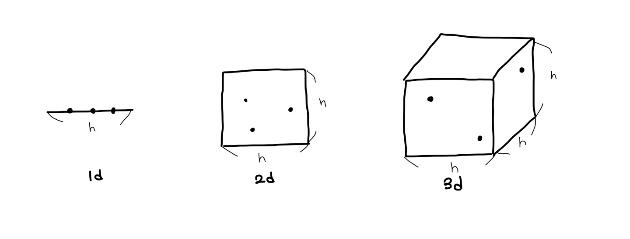
**PCA와 LDA를 이용한 이미지 차원 축소**

바이오인공지능융합학과 2021192861 김가연

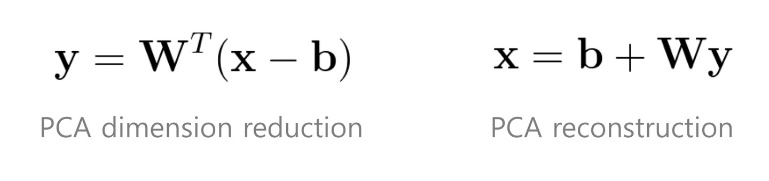
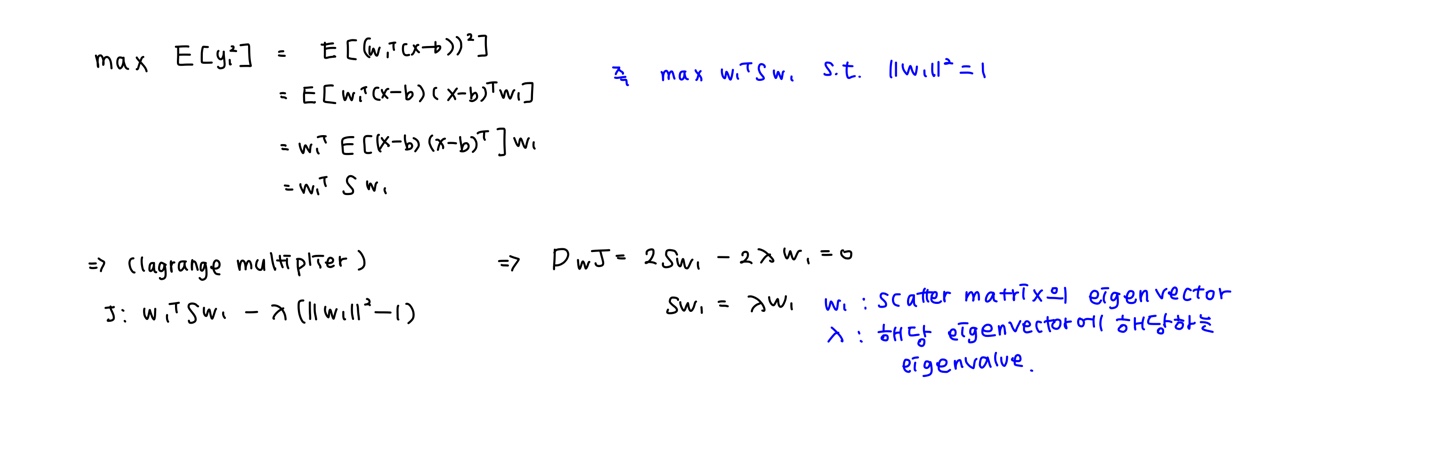
**1. Introduction**

**1.1 Curse of Dimensionality**

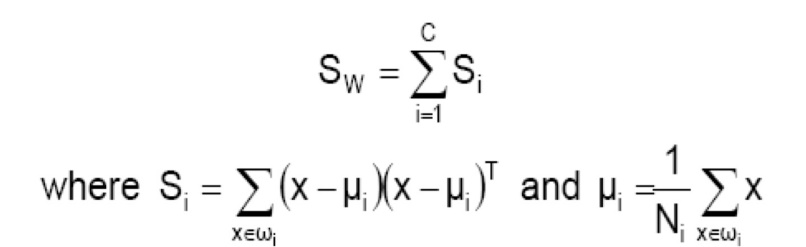
**** Data의 개수가 고정되어 있을 때, 차원이 늘어남에 따라 data들 사이 간격은 늘어나게 되고 그 사이사이는 0으로 채워지게 된다. 0으로 채워진다는 것은 정보가 없다는 것으로 model의 성능저하를 초래한다.

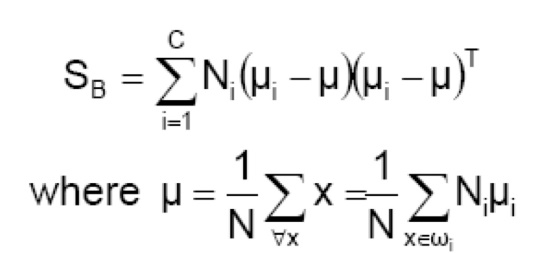
이것을 curse of dimensionality, 차원의 저주라고 한다. Curse of dimensionality를 피하는 방법에는 여러가지가 있는데 그 중 차원을 축소하여 해결하는 방법이 있다. 본 실험에서는 현재 데이터의 특징을 조합하여 차원을 축소하는 방법인 ‘feature extraction’ 중에서 PCA(Principal Components Analysis) 와 LDA(Linear Discriminant Analysis)를 이용해 실험을 하였다.

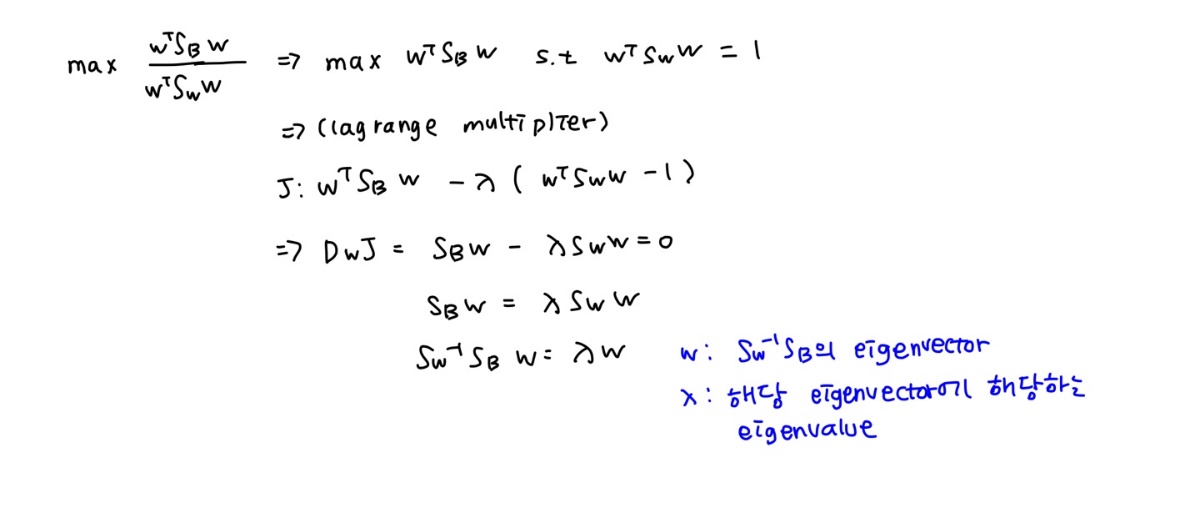
**1.2 PCA(Principal Components Analysis)**

PCA는 차원 축소된 y의 분산을 최대로 유지하는 것이 목적이다. 데이터(x)에 평균(b)를 빼서 평균을 원점으로 옮긴 후 weight(W)를 곱하여 차원 축소를 한다.

즉 PCA를 통해 차원축소를 하려면 데이터의 scatter matrix의 eigenvector들로 W를 구성하여 사용한다.

**1.3 LDA (Linear Discriminant Analysis)**



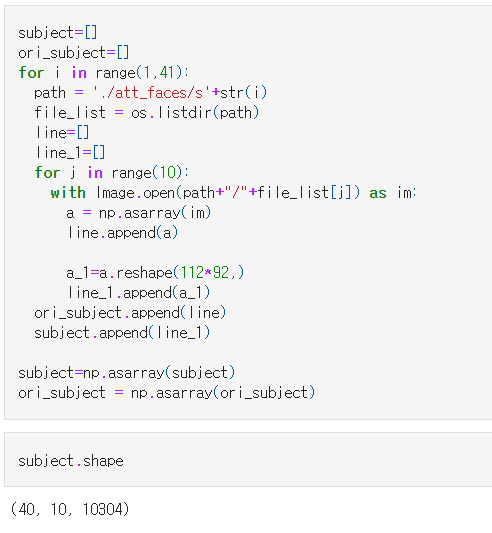
LDA는 class간 거리는 멀게(, class 안의 분산은 작게, 유지시키는 것이 목적이다. 의 는 class 데이터 수에 따른 가중치인데 본 실험에서는 class 별 데이터의 개수가 모두 동일하므로 1로 간주한다.

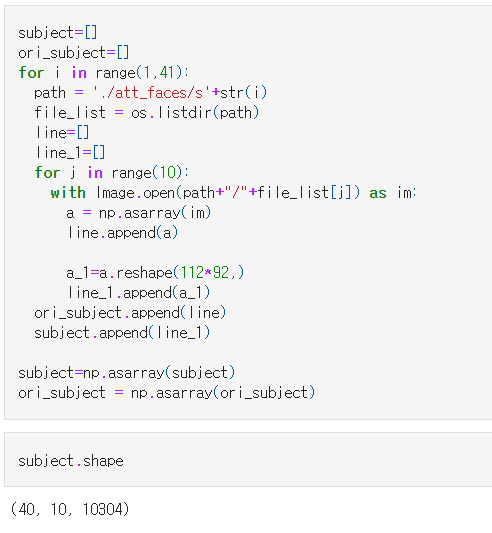
즉 LDA를 통해 차원축소를 하려면 의 eigenvector들로 W를 구성하여 사용한다.

본 실험에서는 각 이미지들을 차원축소 하여 eigenvector들로 표현하였을 때 얼마나 잘 표현되는지에 대해 실험을 해 본다. PCA는 차원축소 후 ‘eigen face’를 출력해 보고 reconstruct 후 ‘reconstructed image’도 비교해본다. LDA는 차원축소 후 ‘fisher face’를 출력해 본다.

후에 두 방법 모두 다양한 dimension으로 차원축소를 하여 정확도를 비교해보고, 10-Fold Cross Validation방법을 통해 평균적으로 얼만큼의 정확도를 나타내는지 실험해 본다.

**2. Method**

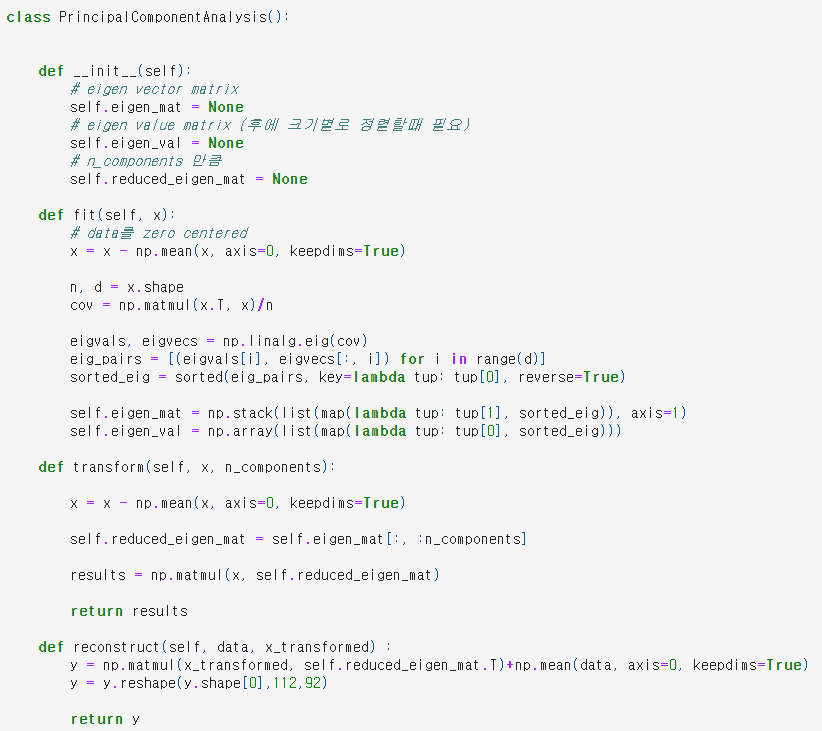
**2.1 Data**

****

총 40명의 사진이 10장씩 400장의 사진을 array로 변환하여 (40, 10, 112\*92[[1]](#footnote-1)) shape로 만들어 데이터(subject)로 사용한다.

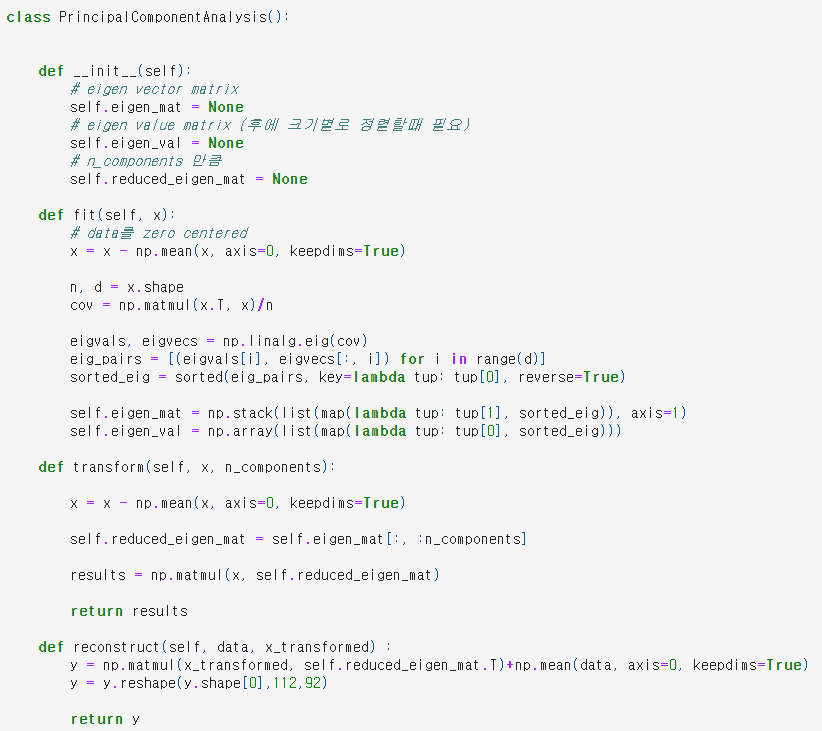
**2.2 PCA function**

2.2.1 def fit(data)

****

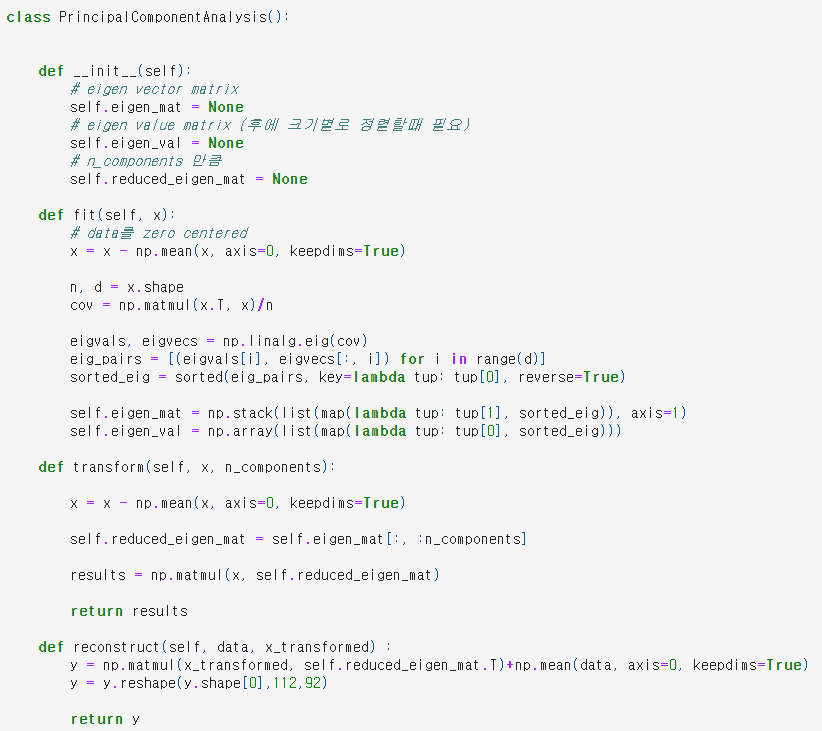
Data를 변수로 받아서 평균을 원점으로 옮기고 np.matmul 함수를 통해 covariance matrix를 구한다. Covariance의 eigvals(eigen values)와 eigvecs(eigen vectors)를 구하고 eigvals가 큰 것부터 해당되는 eigvecs를 정렬하여 eigen\_mat를 구성한다.

2.2.2 def transform(data, n\_components)

****

Data와 목표로 하는 차원 수(n\_componets)를 변수로 받아 데이터의 평균을 원점으로 옮기고 eigen\_mat에서 n\_components 만큼에 해당하는 열을 잘라 reduced\_eigen\_mat로 칭하고 데이터와 reduced\_eigen\_mat를 matmul을 통해 차원 축소를 시킨다.

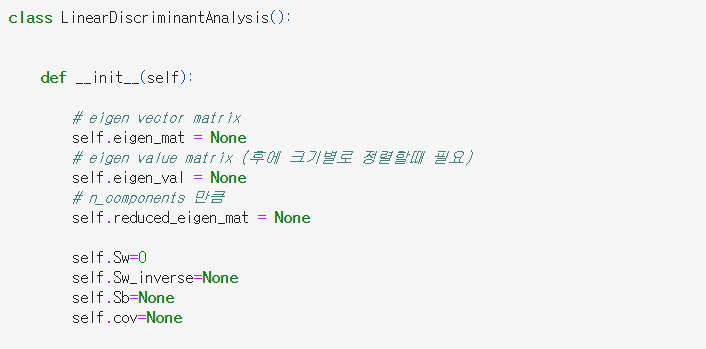
2.2.3 def reconstruct(data, x\_transformed)

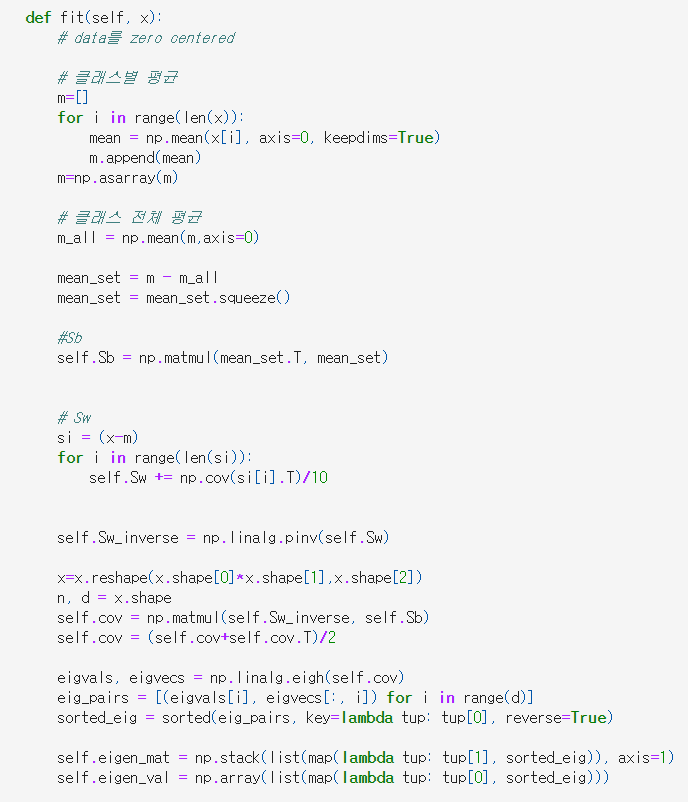
****

차원 축소된 데이터(x\_transformed)와 원래 데이터를 변수로 받아 x\_transformed와 reduced\_eigen\_mat를 곱하고 원래 데이터의 평균을 더해준다. 후에 다시 112, 92 shape으로 만들어준다.

**2.3 LDA function**

2.3.1 def fit(data)

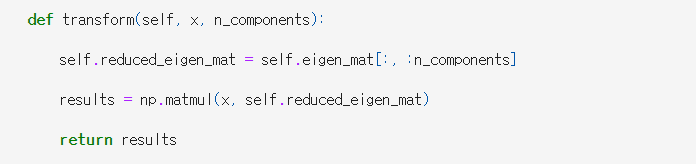
****

****

데이터를 변수로 받아 와 를 구한다. 는 각 class별 평균에 전체 class의 평균을 빼서 사용하고 는 데이터에 class 별 평균을 뺀 들의 합으로 이루어진다.

가 singular일 수도 있으니 pseudo inverse를 통해 를 구하고 와 를 matmul하여 cov matrix로 사용한다. 이때 cov matrix의 eigen value와 eigen vector를 구하는데 복소수가 나오는 경우가 발생해 eigh를 통해 eigvals와 eigvecs를 구하였다. np.linalg.eigh 함수를 이용하기 위해서 cov matrix를 (cov + cov.T)/2를 통해 임의로 symmetric하게 만들어 주었다. 마지막으로 eigvals가 큰 것부터 해당되는 eigvecs를 정렬하여 eigen\_mat를 구성한다.

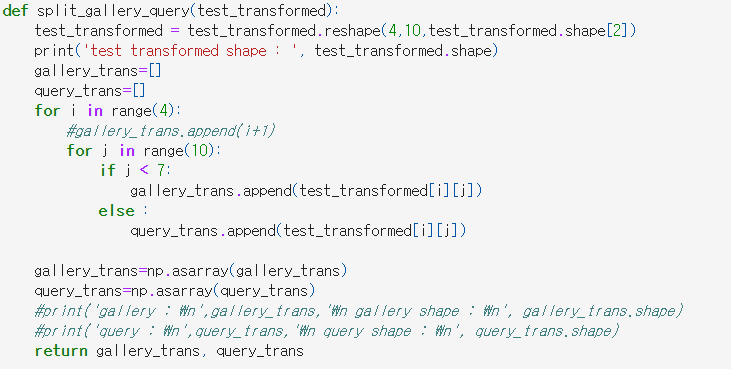
2.3.2 def transform(data, n\_components)

****

Data와 목표로 하는 차원 수(n\_componets)를 변수로 받아 데이터의 평균을 원점으로 옮기고 eigen\_mat에서 n\_components 만큼에 해당하는 열을 잘라 reduced\_eigen\_mat로 칭하고 데이터와 reduced\_eigen\_mat를 matmul을 통해 차원 축소를 시킨다.

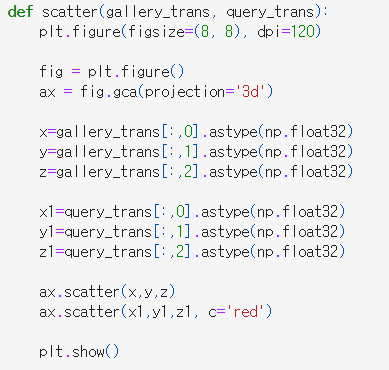
**2.4 공통된 function**

2.4.1 def split\_gallery\_query(test\_transformed)

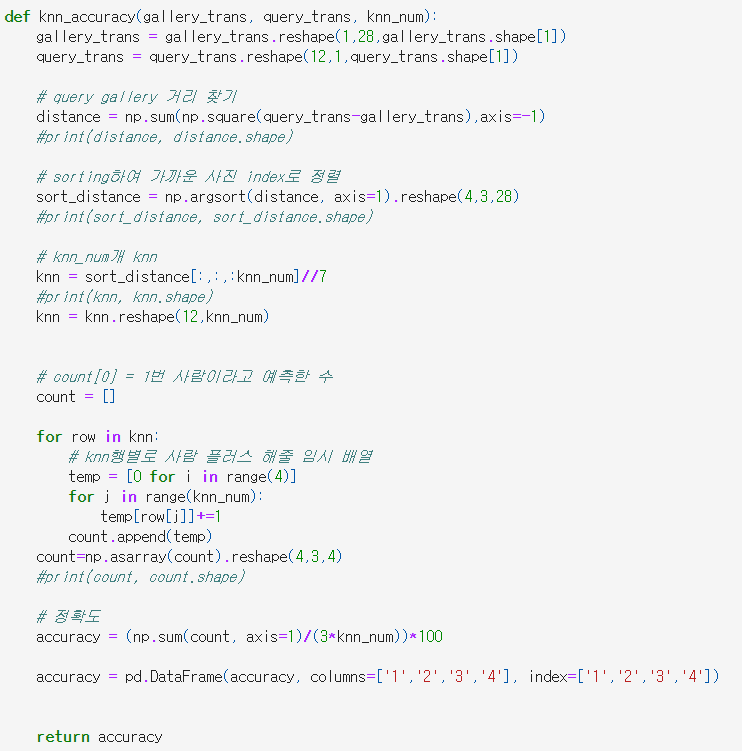
****

차원 축소된 데이터(test\_transformed)를 변수로 받아 7:3의 비율로 gallery, query로 나누어 return해준다.

2.4.2 def scatter(gallery, trans)

 Gallery와 query의 3차원까지만 3-d scatter로 뿌려 확인한다.

2.4.3 def knn\_accuracy(gallery\_trans, query\_trans, knn\_num)



gallery\_trans와 query\_trans, knn을 수행할 개수를 변수로 받아 정확도를 측정하는 함수이다.

Query들과 gallery 데이터들 모두 거리를 측정하여 argsort를 이용해 query들이 gallery 몇 번째 사진과 가장 가까운지를 sort\_distance에 저장한다. 이때 gallery에는 한 사람의 사진이 7장씩 존재하므로 0~6번은 1번 사람, 7~13 번은 2번 사람, … , 21~27 번은 4번 사람에 해당되는 사진이다. 각 query사진마다 몇 번 사진이 몇 번 나왔는지를 count에 저장한다. 예를 들어 1번 사람 query 사진이 1번 사람 gallery 사진이 5번, 3번 사람 gallery 사진이 2번 나왔다면 [5,0,2,0]이 count에 저장된다.

후에 count를 모두 더해 3\*knn\_num으로 나누고 100을 곱하여 정확도 matrix를 구한다. 실제 정확도는 정확도 matrix의 trace를 모두 더해 4로 나눈 값이 된다.

본 실험에서 한 사람 당 gallery의 개수를 7개로 정하였으므로 knn은 최대 7을 사용해야 의미 있는 결과를 도출해 낼 수 있다. knn\_num을 7개로 설정을 하면 내 gallery의 사진 7개 모두가 들어왔는지 확인할 수 있기 때문이다. 그러므로 본 실험에서는 knn\_num을 7으로 설정해 실험해 본다.

**3. Result**

**3.1 다양한 dimension으로 차원 축소했을 때 비교**

Subject에서 random하게 4명의 사진[[2]](#footnote-2)(총 40장)은 test로 나머지 36명의 사진(총 360장)은 train으로 사용한다. 각 데이터의 shape는 (n\*10,10304)이다.

3.1.1 PCA

- eigen face

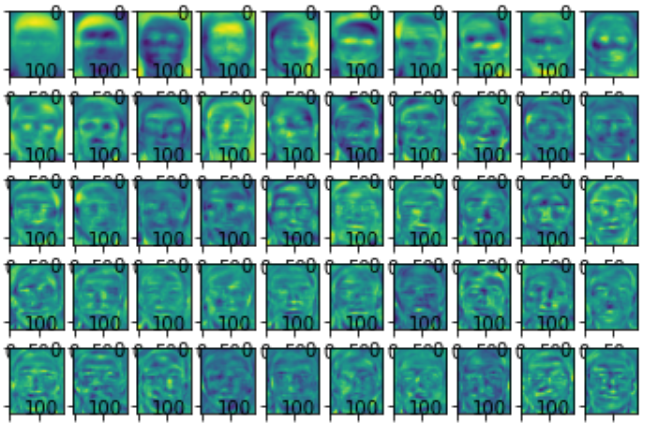
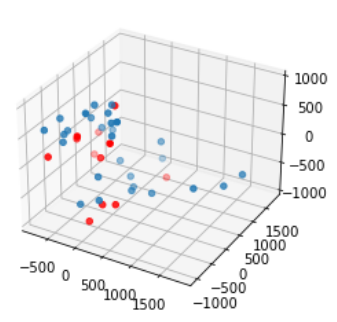


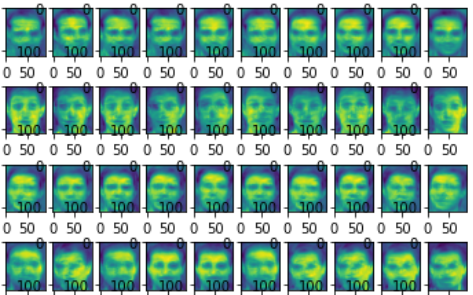
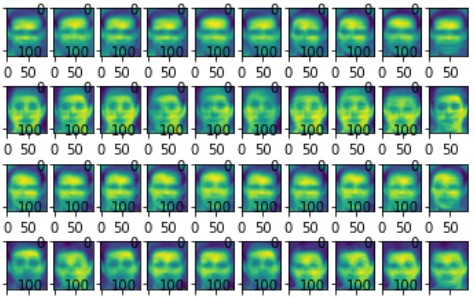
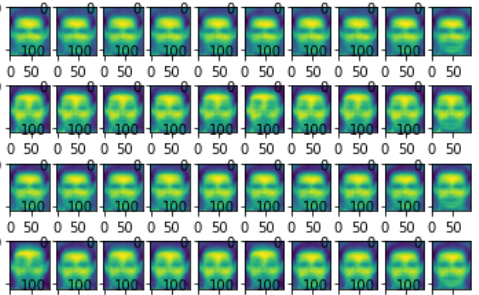
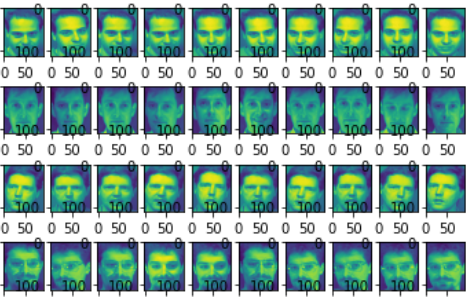
Figure eigen face

- scatter

scatter 함수를 통해 이미지들의 위치를 찍어 본다.

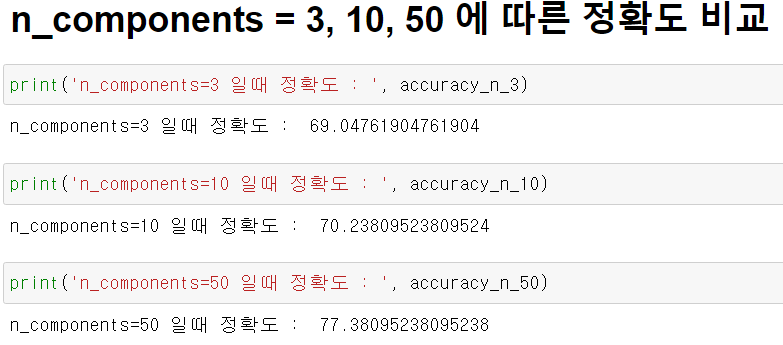
빨간 점이 query, 파란 점이 query이다.

- reconstruct 후 image



왼쪽 위부터 시계방향으로 각각 original image, n\_components=3, n\_components=50, n\_components=10으로 차원축소 후 다시 reconstruct한 이미지이다. 축소한 차원이 클수록 원래 이미지와 좀 더 비슷하게 reconstruct 되는 것을 볼 수 있다.

- 차원 수에 따른 정확도

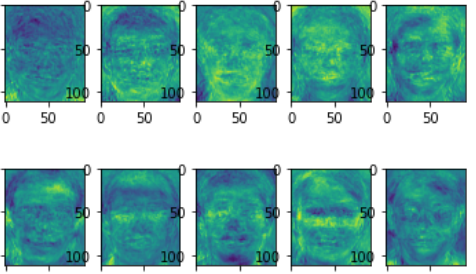


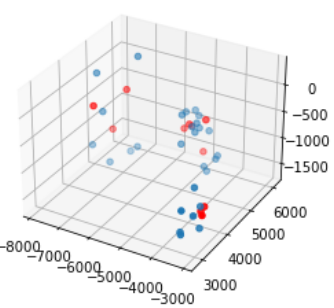
위의 reconstruct image 결과와 비슷하게 n\_components의 수가 높아질수록 정확도가 높아짐을 확일 할 수 있다.

3.1.2 LDA

LDA는 PCA와 다르게 축소할 수 있는 최대 차원은 의 최대 rank인 C-1까지이다. 본 실험에서는 36명을 train data로 사용하였으니 축소 가능한 최대 n\_components는 35이다.

- fisher face



- scatter

scatter 함수를 통해 이미지들의 위치를 찍어 본다.

빨간 점이 query, 파란 점이 query이다.

- 차원 수에 따른 정확도 비교



PCA와 마찬가지로 차원의 수가 커질수록 정확도가 높아진다.

차원에 따른 비교는 각 방법별로 비교해 보았다. 먼저 test data로 사용한 4명의 사람을 random으로 뽑아 비교해 보았다. 원래의 차원은 10304 였지만 PCA, LDA 모두 20배가 넘는 차원으로 줄여도 높은 정확도를 나타냈다.

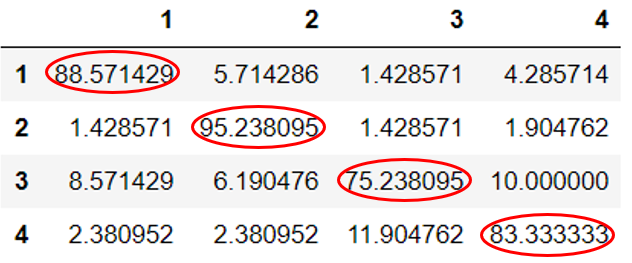
PCA에서는 3차원에서는 69.05%, 10차원에서는 70.34%, 50차원에서는 77.38%로 70%이상의 정확도를 볼 수 있었고, LDA에서는 3차원에서는 82.14%, 10차원에서는 84.53%, 최대 차원인 35차원에서는 86.90%로 82% 이상의 결과를 도출해 낸다. 특히 LDA에서는 최대 차원인 35차원으로 축소하였을 때 86.90%로 아주 높은 정확도를 나타냈다.

**3.2 10-Fold Cross Validation 방법을 이용한 정확도 비교**

원래의 데이터는 (n,10304) shape, 즉 10304 차원이다. 그중 1% 정도에 해당되는 10차원으로줄였으때 얼만큼의 정확도를 나타내는지 확인하고 싶어 n\_components를 10으로 설정하고 10-Fold Cross Validation 방법을 이용한 실험 결과이다.

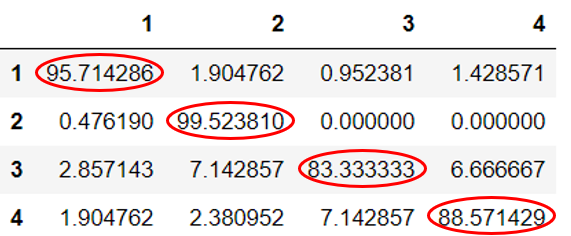
PCA, LDA 모두 40명의 이미지를 차례로 4명씩 잘라 test data로, 나머지 36명의 이미지는 train data로 설정하여 총 10번의 실험을 한 결과이다.

3.2.1 PCA



**PCA**에서는 한사람당 query 사진 3장은 평균적으로 1번 사람은 88.57%, 2번 사람 사진은 95.24%, 3번 사람은 75.24%, 4번 사람은 83.33%의 정확도를 보여주었다. 평균적으로 총 **85.60%의 정확도**를 도출해냈다.

3.2.2 LDA



**LDA**에서는 한사람당 query 사진 3장은 평균적으로 1번 사람은 95.71%, 2번 사람 사진은 99.52%, 3번 사람은 83.33%, 4번 사람은 88.57%의 정확도를 보여주었다. 평균적으로 총 **91.79%의 정확도**를 도출해냈다.

본 실험에서는 LDA의 성능이 더 좋게 나오는 것을 확인할 수 있다. 전체적으로 두 방법의 개념은 유사하지만 PCA는 class label 없이 데이터 셋 전체에 적용하는 ‘unsupervised’ 알고리즘이고, LDA는 class label 정보를 이용하여 각 class를 최대한 분리해 내는 ‘supervised’ 알고리즘이다. 이를 보면 LDA가 PCA보다 성능이 좋다는 것을 알 수 있다. 하지만 image 인식에서 class의 sample 개수가 적으면 PCA의 성능이 더 좋을 경우도 존재한다.[[3]](#footnote-3) 따라서 데이터의 수가 더 적어졌을 때 본 실험을 수행하면 PCA의 성능이 더 높게 나올 수도 있을 것이다.

1. 이미지당 (112,92) pixel로 이루어져 있는데 이를 112\*92 로 펼쳐서 사용 [↑](#footnote-ref-1)
2. 차원에 따른 비교는 각 방법마다(PCA, LDA) 비교를 하여 test 데이터를 랜덤으로 추출하였다. [↑](#footnote-ref-2)
3. Martinez et al., 2001, https://ieeexplore.ieee.org/document/908974?arnumber=908974 [↑](#footnote-ref-3)