고급소프트웨어실습I
3주차 보고서

학번 : 20171663
이름 : 이도훈

```
# TODO -->
pca_student_score = student_pca(X, n_components = 2)

mx = np.mean(X, axis = 0)
nX = X - mx
digits_cov = np.cov(nX, rowvar = False)
eig_vals,eig_vecs = np.linalg.eigh(digits_cov)
sort = eig_vals[::-1]
index = np.argsort(sort)
sorted_eigvecs = eig_vecs[:,index]
eigvecs_s = sorted_eigvecs[:,0:2]

pca_results_t = np.transpose(pca_student_score)
tmp = np.transpose((eigvecs_s.dot(pca_results_t)))
new = tmp + mx

n_samples = new.shape[0]
images = new.reshape((n_samples,-1))
```

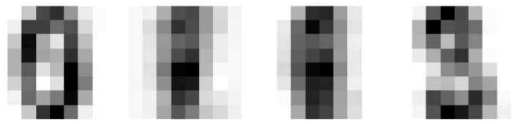
```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image in zip(axes, images):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
result = np.square(np.subtract(X, new)).mean()
print(result)
```

```
13.421012200761453
```

```
# TODO -->
pca_student_score = student_pca(X, n_components = 3)

mx = np.mean(X, axis = 0)
nX = X - mx
digits_cov = np.cov(nX, rowvar = False)
eig_vals,eig_vecs = np.linalg.eigh(digits_cov)
sort = eig_vals[::-1]
index = np.argsort(sort)
sorted_eigvecs = eig_vecs[:,index]
eigvecs_s = sorted_eigvecs[:,0:3]

pca_results_t = np.transpose(pca_student_score)
tmp = np.transpose((eigvecs_s.dot(pca_results_t)))
new = tmp + mx

n_samples = new.shape[0]
images = new.reshape((n_samples,-1))
```

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image in zip(axes, images):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
result = np.square(np.subtract(X, new)).mean()
print(result)
```

```
11.206800697129166
```



```
# TODO -->
pca_student_score = student_pca(X, n_components=4)

mx = np.mean(X, axis = 0)
nX = X - mx
digits_cov = np.cov(nX, rowvar = False)
eig_vals,eig_vecs = np.linalg.eigh(digits_cov)
sort = eig_vals[::-1]
index = np.argsort(sort)
sorted_eigvecs = eig_vecs[:,index]
eigvecs_s = sorted_eigvecs[:,0:4]

pca_results_t = np.transpose(pca_student_score)
tmp = np.transpose((eigvecs_s.dot(pca_results_t)))
new = tmp + mx

n_samples = new.shape[0]
images = new.reshape((n_samples,-1))
```

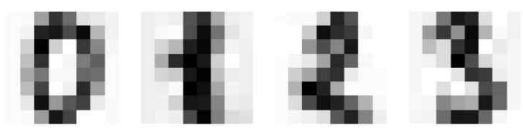
```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image in zip(axes, images):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
result = np.square(np.subtract(X, new)).mean()
print(result)
```

```
9.627986407129212
```

```
# TODO -->
pca_student_score = student_pca(X, n_components = 32)

mx = np.mean(X, axis = 0)
nX = X - mx
digits_cov = np.cov(nX, rowvar = False)
eig_vals,eig_vecs = np.linalg.eigh(digits_cov)
sort = eig_vals[::-1]
index = np.argsort(sort)
sorted_eigvecs = eig_vecs[:,index]
eigvecs_s = sorted_eigvecs[:,0:32]

pca_results_t = np.transpose(pca_student_score)
tmp = np.transpose((eigvecs_s.dot(pca_results_t)))
new = tmp + mx

n_samples = new.shape[0]
images = new.reshape((n_samples,-1))
```

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image in zip(axes, images):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
result = np.square(np.subtract(X, new)).mean()
print(result)
```

```
0.6316360146108382
```



왼쪽 위부터 시계방향으로 차례대로 2차원, 3차원, 4차원, 32차원으로 pca를 통해 차원축소한 뒤 복원시킨 결과와 해당하는 mse(mean square error) 값이다.
고차원으로 pca를 통해 차원 축소하면 할수록 mse의 값이 더 적은 것을 알 수 있다.

$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$으로 나타낼 수 있다. 코드에서는 np.square(np.subtract(X, new)).mean()으로 구현을 했는데 X는 모든 차원에 대한 코드에서 동일하다. 하지만 new의 값은 차원에 대한 코드마다 다르다. 그 이유로는 new는 pca_student_score라는 변수를 사용하여 계산이 되었는데 이 pca_student_score 변수는 student_pca라는 함수의 연산 결과이고, student_pca는 각각의 차원 값이 parameter로 입력된다. 이 차원 값이
eigvecs_s = sorted_eigvecs[:,0:n_components]의 연산 과정을 거치게 되는데 sorted_eigvecs에서 열의 인덱스가 0부터 n_components - 1까지를 eigvecs_s에 저장하게 된다. 이때 n_components가 입력받은 차원값이고 이로 인해 eigvecs_s에 저장되는 정보의

양이 달라진다. 고차원으로 축소할수록 유실 되는 데이터의 양이 적어지게 되므로 mse라는 평균 제곱 오차의 값이 더 작아지게 되는 것이다.