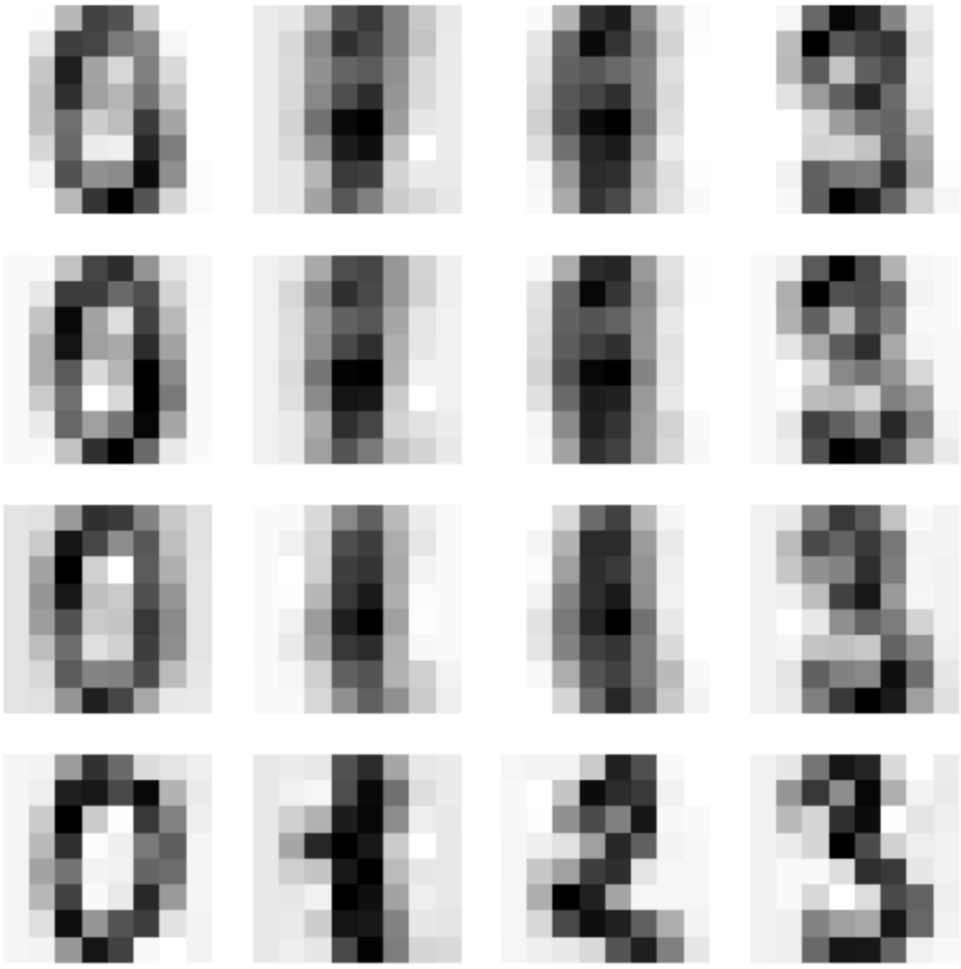
고급소프트웨어실습1

3주차 과제

20161663 허재성

실험 결과 및 코드

```
2-dim mse : 13.42101220076145
3-dim mse : 11.206800697129168
4-dim mse : 9.62798640712921
32-dim mse : 0.631636014610838
```



위에서부터 차례대로 2차원, 3차원, 4차원, 32차원으로 PCA한 결과를 reconstruction한 후 복원 데이터를 시각화한 것이다. PCA의 차원이 커질수록 원본 데이터에 가까워지고 오차율도 작아지는 것을 확인할 수 있다. 코드는 다음과 같다.

```
# TODO -->
def draw(images):
    _, 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')

def reconstruct(origin, pca, n):
    X = np.array(origin)
    meanX = X.mean(axis=0)
    length = len(X[0])
    pca = pca.T
    norm_X = X - X.mean(axis=0)
    cov_X = np.cov(norm_X.T)
    _, eigenvector = np.linalg.eig(cov_X)
    eigenvector = eigenvector[:,:n]

    result = eigenvector@pca
    result = result.T
    result = result + meanX

#     print(n, "차원 복원 결과")
#     print(result)

    return result

def mse(y, t):
    return np.square(np.subtract(y, t)).mean()

pca2 = student_pca(digits["data"])
pca3 = student_pca(digits["data"], 3)
pca4 = student_pca(digits["data"], 4)
pca32 = student_pca(digits["data"], 32)

# 2 차원 pca 복원
rev2 = reconstruct(digits.data, pca2, 2)
# 3 차원 pca 복원
rev3 = reconstruct(digits.data, pca3, 3)
# 4 차원 pca 복원
rev4 = reconstruct(digits.data, pca4, 4)
# 3 차원 pca 복원
rev32 = reconstruct(digits.data, pca32, 32)

# 복원 데이터 그리기
# draw(digits.data)
draw(rev2)
draw(rev3)
```

```
draw(rev4)
draw(rev32)

ms2 = mse(digits.data, rev2)
print("2-dim mse :", mse2)
ms3 = mse(digits.data, rev3)
print("3-dim mse :", mse3)
ms4 = mse(digits.data, rev4)
print("4-dim mse :", mse4)
ms32 = mse(digits.data, rev32)
print("32-dim mse :", mse32)
```

오차율 분석

차원 축소가 클수록 더 적은 공간을 사용해서 원본 데이터 특성을 나타낼 수 있는 대신 원본 데이터와 차이가 커지게 된다. 또한 차원을 축소한 PCA로 추출한 데이터를 복원할 경우에도 차원이 작을수록 원본 데이터의 eigenvector가 덜 반영되기 때문에 복원된 데이터도 원본 데이터와의 오차가 커지게 된다.