3주차 보고서

Created	@October 6, 2021 12:32 PM
Created by	**20190258 김혜린**

PCA reconstruction

$$\hat{P} = V \times Q + \mu$$

위와 같은 식을 만족하는 PCA reconstruction을 아래와 같이 `reconstruct_pca`함수로 구현하였다.

```
def reconstruct_pca(Q, V, M):
    return Q@V + M
```

`reconstruct_pca`함수의 파라미터 Q, V, M은 아래와 같다.

- Q : PCA 결과, $N \times d'$ 의 차원을 가진다.
- V : eigen vector, $d' \times d$ 의 차원을 가진다.
- M : 원래 데이터 X의 각 차원당 평균, $1 \times d$ 차원을 가진다.

`reconstruct_pca`함수에 대해 2, 3, 4, 32차원의 PCA 결과를 복원하고 아래 MSE 함수를 이용해 원래 데이터 X와의 MSE error를 구하였다.

```
def MSE(origin, predict):
    return np.square(np.subtract(origin, predict)).mean()
```

2차원 PCA reconstruction

```
# 2차원 PCA
pca_digits, V, X_bar = student_pca_2(X, n_components=2)
# PCA reconstruction
```

```
re_X = reconstruct_pca(pca_digits, V, X_bar)
# MSE with X
mse = MSE(X, re_X)

# visualize re_X
print("2차원: MSE Error: ", mse)
images = re_X.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')
```

MSE error

```
2차원: MSE Error:  13.421012200761451
```

3차원 PCA reconstruction

```
# 3차원 PCA
pca_digits, V, X_bar = student_pca_2(X, n_components=3)
# PCA reconstruction
re_X = reconstruct_pca(pca_digits, V, X_bar)
# MSE with X
mse = MSE(X, re_X)

# visualize re_X
print("3차원: MSE Error: ", mse)
images = re_X.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')
```

MSE error

```
3차원: MSE Error:  11.206800697129161
```

4차원 PCA reconstruction

```
# 4차원 PCA
pca_digits, V, X_bar = student_pca_2(X, n_components=4)
# PCA reconstruction
re_X = reconstruct_pca(pca_digits, V, X_bar)
# MSE with X
mse = MSE(X, re_X)

# visualize re_X
print("4차원: MSE Error: ", mse)
images = re_X.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')
```

MSE error

```
4차원: MSE Error:  9.62798640712921
```

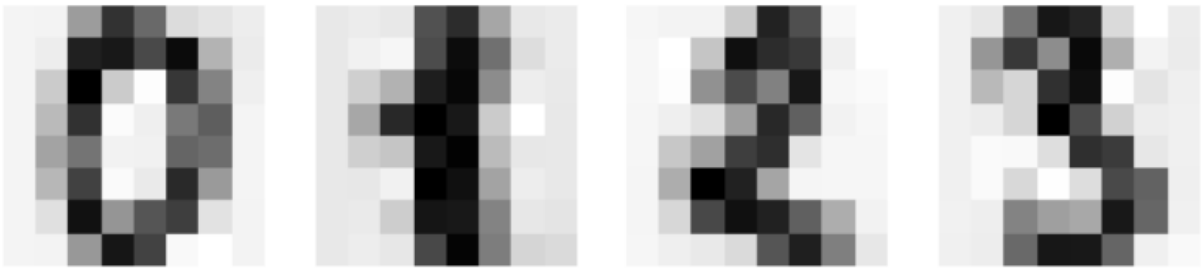
32차원 PCA reconstruction

```
# 32차원 PCA
pca_digits, V, X_bar = student_pca_2(X, n_components=32)
# PCA reconstruction
re_X = reconstruct_pca(pca_digits, V, X_bar)
# MSE with X
mse = MSE(X, re_X)

# visualize re_X
print("32차원: MSE Error: ", mse)
images = re_X.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')
```

MSE error

```
32차원: MSE Error:  0.6316360146108382
```



결과분석

PCA 실행시, 축소되는 차원의 크기 d' 의 크기가 클수록 PCA reconstruction과 원래 데이터 X와의 MSE error가 작은 것을 확인할 수 있다.

PCA의 주목적은 차원 축소로 PCA에서 원본 데이터를 projection할 d' 차원의 초평면을 선택할 때 분산(eigen value)이 최대가 되는 축(eigen vector)을 초평면으로 선택한다. 분산이 가장 큰 축을 고른다는 뜻은 해당 축을 원본 데이터를 projection 했을 때 projection된 데이터와 원래 데이터 사이의 평균 제곱 거리가 최소가 된다는 뜻이다.

따라서 원본 데이터의 분산을 최대한 보존하는 방향으로 projection을 진행하는 것을 알 수 있다.

평균을 0으로 맞춘 원본 데이터 $X(=P-\mu)$를 단위벡터 $\overrightarrow{e}$ 인 임의의 축에 projection 한다고 했을 때 결과는 $X\overrightarrow{e}$ 로 표현할 수 있고 이때의 분산은 다음과 같이 나타낼 수 있다.

$$
\begin{aligned}
Var[X\overrightarrow{e}] &= \frac{1}{n-1}\sum_{i=1}^{n}[X\overrightarrow{e} - E(X\overrightarrow{e})]^2 \\
&= \frac{1}{n-1}\sum_{i=1}^{n}[X\overrightarrow{e} - E(X)\overrightarrow{e}]^2, (E(X)=0) \\
&= \frac{1}{n-1}\sum_{i=1}^{n}(X\overrightarrow{e})^2 \\
&= \frac{1}{n-1}(X\overrightarrow{e})^T(X\overrightarrow{e}) \\
&= \frac{1}{n-1}\overrightarrow{e}^T X^T X\overrightarrow{e} \\
&= \overrightarrow{e}^T\left(\frac{X^TX}{n-1}\right)\overrightarrow{e} \\
&= \overrightarrow{e}^T\Sigma\overrightarrow{e}
\end{aligned}
$$

이때 $X^TX = (P-\mu)^T(P-\mu)$ 를 만족하고 따라서 $\frac{X^TX}{n-1}$ 는 P 의 공분산 (Σ) 이 된다.

따라서 PCA 결과 $Var[X\overrightarrow{e}] = \overrightarrow{e}^T\Sigma\overrightarrow{e}$ 를 최대화하는 $\overrightarrow{e}$를 찾는 문제이며 이때 제약 조건은 $\|e\|^2 = 1$ 이다.

$\|e\|^2 = \overrightarrow{e}^T \cdot \overrightarrow{e}$ 으로 이를 라그랑지안 함수 L 은 다음과 같다.

$$
L(\overrightarrow{e}, \lambda) = \overrightarrow{e}^T\Sigma\overrightarrow{e} - \lambda(\overrightarrow{e}^T\overrightarrow{e}^-1)
$$

L 을 $\overrightarrow{e}$에 대해 편미분 하면 다음을 만족한다.

$$
\begin{aligned}
\frac{\partial L}{\partial \overrightarrow{e}} &= (\Sigma + \Sigma^T)\overrightarrow{e} - 2\lambda\overrightarrow{e} \\
&= 2\Sigma\overrightarrow{e} - 2\lambda\overrightarrow{e} = 0 \\
&\therefore \Sigma\overrightarrow{e} = \lambda\overrightarrow{e}
\end{aligned}
$$

따라서 Σ 의 eigen vector가 분산을 최대로 하는 값임을 알 수 있고 이때의 eigen value λ가 projection한 결과의 분산 값임을 알 수 있다.

d 차원을 가진 원래 데이터 X를 d' 차원으로 원본 데이터를 축소할 경우, eigen vectors $V = [\overrightarrow{e_1}, \overrightarrow{e_2}, ..., \overrightarrow{e_{d'}}]$에 projection하는 것이 되므로 PCA 결과 d'개의 eigen vectors에 해당하는 분산의 합을 가져가게 되고 $d - d'$ 개의 eigen vector가 가지는 분산을 잃게 된다. 따라서 축소되는 차원이 클수록 잃는 분산이 적어지기 때문에 분포 양상이 비슷해지게 되고 MSE 계산 결과 원본데이터의 각 원소들과 거리 차를 계산 평균이 작아지게 된다.