BDA x 이지스 퍼블리싱 머신러닝 스터디 2주차 과제
조 이름 : 런닝머신 (9조)

1. 다음의 데이터는 전등의 수명을 나타낸 자료이다.

   3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487

   위의 시간이 지수분포 $Exp(\lambda)$를 따른다고 가정해보자.

   1.1 p45를 참고해 기울기가 0이 지점의 최적의 λ를 구하시오.

   1.2 뉴턴법을 이용하여 최적의 λ를 구하는 알고리즘을 작성해보시오.

1.1

```
import numpy as np

def find_optimal_lambda(data):
    n = len(data)
    optimal_lambda = n / np.sum(data)
    return optimal_lambda

# 주어진 데이터
data = np.array([3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487])

#최적의 람다 계산
optimal_lambda = find_optimal_lamba(data)
print(optimal_lambda)
```

계산된 최적의 람다 **: 0.009252120277563608**

1.2

뉴턴법은 함수의 극소점을 찾는 방법 중 하나로, 다음과 같이 반복하여 근사적으로 해를 찾는다.

$$\lambda_{\text{new}} = \lambda_{\text{old}} - \frac{f'(\lambda_{\text{old}})}{f''(\lambda_{\text{old}})}$$

```
import numpy as np

# 주어진 데이터
data = np.array([3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487])

# 로그-우도 함수의 미분과 이계도 함수 정의
def log_likelihood_lambda(lmbda, data):
    n = len(data)
    return -(n / lmbda) + np.sum(data)

def log_likelihood_lambda_prime(lmbda, data):
    n = len(data)
    return n / (lmbda**2) - np.sum(data)
```

```
def log_likelihood_lambda_double_prime(lmbda, data):
    n = len(data)
    return -2 * n / (lmbda**3)

# 뉴턴법 알고리즘
def newton_method(data, initial_guess, tol=1e-6, max_iter=100):
    lmbda_old = initial_guess
    for i in range(max_iter):
        lmbda_new = lmbda_old - log_likelihood_lambda_prime(lmbda_old, data) /
log_likelihood_lambda_double_prime(lmbda_old, data)
        if np.abs(lmbda_new - lmbda_old) < tol:
            return lmbda_new
        lmbda_old = lmbda_new
    return lmbda_old

# 초기 추정값 설정
initial_guess = 0.1

# 최적의 lambda 계산
optimal_lambda = newton_method(data, initial_guess)

print("Optimal lambda:", optimal_lambda)
```

2. $A = \begin{pmatrix} 3 & 0 \\ 4 & 5 \end{pmatrix}$ 일때, U, Σ, V^T를 구해보시오.

특이값 분해를 사용하면 $A = U\Sigma V^T$ 이다.

이 때, U, V^T는 직교행렬이고, Σ는 A^TA의 고유값의 제곱근을 대각원소로 가지는 대각행렬이다.

또한, $A_{2\times2}$ 이므로 U, Σ, V도 2x2 행렬이다. 따라서 $U = \begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}$, $V = \begin{pmatrix} v_1 & v_2 \\ v_3 & v_4 \end{pmatrix}$ 로 가정한다.

i) 먼저 대각행렬 Σ를 구하기 위해 A^TA의 고유값을 찾는다.

$A^TA = \begin{pmatrix} 3 & 4 \\ 0 & 5 \end{pmatrix}\begin{pmatrix} 3 & 0 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} 25 & 20 \\ 20 & 25 \end{pmatrix}$

$\det(A^TA - \lambda I) = \begin{pmatrix} 25-\lambda & 20 \\ 20 & 25-\lambda \end{pmatrix} = (25-\lambda)^2 - 400 = \lambda^2 - 50\lambda - 225 = (\lambda-5)(\lambda-45)$

$\det(A^TA - \lambda I) = 0$ 이 되게하는 λ가 고유값이므로 $\lambda = 5, 45$ 이다.

$\therefore$ 따라서 $\Sigma = \begin{pmatrix} 3\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{pmatrix} = \sqrt{5}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}$ 이다.

ii) $A^TA = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^2 V^T$ ($\because \Sigma$: 대각행렬, $\Sigma = \Sigma^T$, U: 직교행렬, $U^TU = I$)

$\begin{pmatrix} 25 & 20 \\ 20 & 25 \end{pmatrix} = \begin{pmatrix} v_1 & v_2 \\ v_3 & v_4 \end{pmatrix} 5 \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 & v_3 \\ v_2 & v_4 \end{pmatrix}$

$\begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} 9v_1 & v_2 \\ 9v_3 & v_4 \end{pmatrix}\begin{pmatrix} v_1 & v_3 \\ v_2 & v_4 \end{pmatrix}$

$= \begin{pmatrix} 9v_1^2 + v_2^2 & 9v_1v_3 + v_2v_4 \\ 9v_1v_3 + v_2v_4 & 9v_3^2 + v_4^2 \end{pmatrix}$

* 직교행렬의 경우 $A^TA = AA^T = I$ 이므로

$K = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$ 일 때, $\begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}\begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} = \begin{pmatrix} x_1^2+x_2^2 & x_1x_3+x_2x_4 \\ x_1x_3+x_2x_4 & x_3^2+x_4^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

$\therefore \ x_1^2+x_2^2 = x_3^2+x_4^2 = 1, \ x_1x_3 + x_2x_4 = 0$ 이다.

1) $9v_1^2 + v_2^2 = 8v_1^2 + (v_1^2 + v_2^2) = 5$

$8v_1^2 + 1 = 5$ (by 직교행렬의 성질) $\therefore v_1^2 = \frac{1}{2}$

$\frac{9}{2} + v_2^2 = 5$ $\therefore v_2^2 = \frac{1}{2}$

같은 방식으로 v_3^2, v_4^2도 계산하면 $v_1^2 = v_2^2 = v_3^2 = v_4^2 = \frac{1}{2}$ 이다.

2) $9v_1v_3 + v_2v_4 = 8v_1v_3 + (v_1v_3 + v_2v_4) = 4$

$\therefore v_1v_3 = \frac{1}{2}$, $v_2v_4 = -\frac{1}{2}$ ($\because v_1v_3 + v_2v_4 = 0$)

1), 2)를 종합하면 $V = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix}$ or $\begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix}$ 이다.

a) $V = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix}$

$A = U\Sigma V^T$

$\begin{pmatrix} 3 & 0 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}\sqrt{5}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}\frac{\sqrt{2}}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$

$= \frac{\sqrt{10}}{2}\begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}\begin{pmatrix} 3 & -3 \\ 1 & 1 \end{pmatrix}$

$\frac{1}{\sqrt{10}}\begin{pmatrix} 6 & 0 \\ 8 & 10 \end{pmatrix} = \begin{pmatrix} 3u_1 + u_2 & -3u_1 + u_2 \\ 3u_3 + u_4 & -3u_3 + u_4 \end{pmatrix}$

3) $3u_1 + u_2 - 3u_1 + u_2 = \frac{6}{\sqrt{10}}$ $\therefore u_2 = \frac{3}{\sqrt{10}}$

$3u_1 = u_2$ $\therefore u_1 = \frac{1}{\sqrt{10}}$

4) $3u_3 + u_4 - 3u_3 + u_4 = \frac{10}{\sqrt{10}}$ $\therefore u_4 = \frac{9}{\sqrt{10}}$

$u_4 = 3u_3 + \frac{10}{\sqrt{10}}$ $\therefore u_3 = -\frac{1}{3\sqrt{10}}$

$\therefore U = \frac{1}{\sqrt{10}}\begin{pmatrix} 1 & 3 \\ 9 & -\frac{1}{3} \end{pmatrix}$

b) $V = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix}$

$A = U\Sigma V^T$

$\begin{pmatrix} 3 & 0 \\ 4 & 5 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}\sqrt{5}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}\frac{\sqrt{2}}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

$= \frac{\sqrt{10}}{2}\begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}\begin{pmatrix} 3 & 3 \\ 1 & -1 \end{pmatrix}$

$\frac{1}{\sqrt{10}}\begin{pmatrix} 6 & 0 \\ 8 & 10 \end{pmatrix} = \begin{pmatrix} 3u_1 + u_2 & 3u_1 - u_2 \\ 3u_3 + u_4 & 3u_3 - u_4 \end{pmatrix}$

3) $3u_1 + u_2 + 3u_1 - u_2 = \frac{6}{\sqrt{10}}$ $\therefore u_1 = \frac{1}{\sqrt{10}}$

$3u_1 = u_2$ $\therefore u_2 = \frac{3}{\sqrt{10}}$

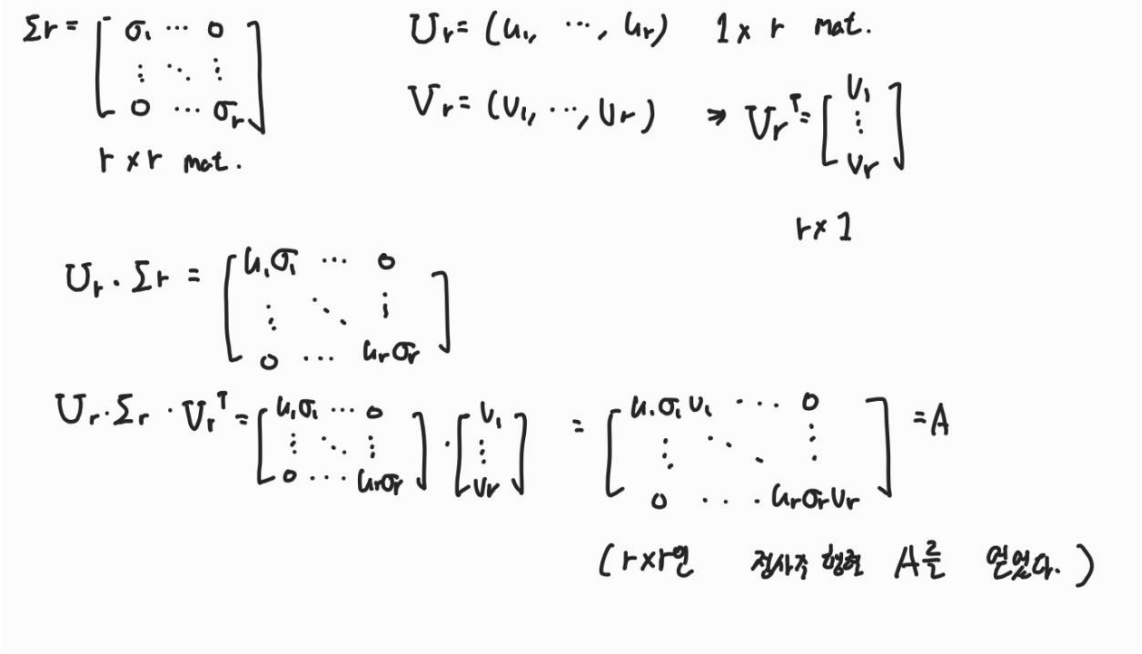
4) $3u_3 + u_4 + 3u_3 - u_4 = \frac{18}{\sqrt{10}}$ $\therefore u_3 = \frac{3}{\sqrt{10}}$

$u_4 = 3u_3 - \frac{10}{\sqrt{10}}$ $\therefore u_4 = -\frac{1}{\sqrt{10}}$

$\therefore U = \frac{1}{\sqrt{10}}\begin{pmatrix} 1 & 3 \\ 3 & -1 \end{pmatrix}$

최종적으로, $A = \begin{pmatrix} 3 & 0 \\ 4 & 5 \end{pmatrix}$ 일 때, $\begin{cases} U = \frac{1}{\sqrt{10}}\begin{pmatrix} 1 & 3 \\ 9 & -\frac{1}{3} \end{pmatrix}, \Sigma = \sqrt{5}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, V = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \\ U = \frac{1}{\sqrt{10}}\begin{pmatrix} 1 & 3 \\ 3 & -1 \end{pmatrix}, \Sigma = \sqrt{5}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, V = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{cases}$ 이다.

3. p50의 표기를 참고해서, $\sigma_{r+1} = \ldots = \sigma_p = 0$이라고 가정해보자$(r < p)$. Σ_r을 대각성분이 $\sigma_1, \ldots, \sigma_r$ 인 대각행렬, U_r을 $u_1, .. u_r$의 행벡터를 가지는 행렬, V_r을 $v_1, .. v_r$의 행벡터를 가지는 행렬이라고 할 때, $A = U_r \Sigma_r V_r^T$임을 보이시오.



4. L2 규제를 포함한 로지스틱 모델을 구현해 p90에 예시와 비교해보시오.

L2

```
# import modules
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')

X, y = load_breast_cancer(return_X_y=True, as_frame=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1234)

X_train = X_train.iloc[:, :3]
X_test = X_test.iloc[:, :3]
```

```
clf = LogisticRegression(random_state=1234, penalty="l2")
clf.fit(X_train, y_train)

y_train_pred = clf.predict(X_train)
y_pred = clf.predict(X_test)

print(f"train set accuracy: {(y_train == y_train_pred).sum() / len(y_train) * 100: .2f}%")
print(f"test set accuracy: {(y_test == y_pred).sum() / len(y_test) * 100: .2f}%")
```

```
train set accuracy:  91.60%
test set accuracy:  86.70%
```

- train set은 84.51%에 비해 7.09% 올랐다.
- test set은 76.06%에 비해 10.06% 올랐다.

5. 3장 되새김 문제 2번을 같이 풀어보시오.

```
ys = []
intercept = -7
beta = np.array([[-4, -12, 8, 9, 6]]).reshape(-1, 1)

np.random.seed(1111)
for i in range(n):
    xb = np.exp((intercept + (X.iloc[i].values.reshape(1, -1) @ beta))[0] + np.random.normal(0, 7.5))
    pi = xb / (1 + xb)
    if pi >= 0.5:
        y = 1
    else:
        y = 0
    ys.append(y)
y = pd.Series(ys)
```

```
[13]  from sklearn.linear_model import LogisticRegression
      clf=LogisticRegression(random_state=1234)
      clf=clf.fit(X,y)

      print(f'절편: {clf.intercept_[0]}')
      print(f'계수: {clf.coef_[0]}')

      y_pred =clf.predict(X)

      print(f'정확도:{(y==y_pred).mean() * 100: 2f}%')
```

```
절편: -1.3873690423718181
계수: [-0.68220025 -2.27202536  1.63149555  1.93208239  0.70733439]
정확도: 89.000000%
```

6. 선형 회귀모델 $Y = X\beta + \epsilon$ ($X \in \mathbb{R}^{n\times(p+1)}$는 disigned matrix)을 생각해보자. 부분집합 $M \in \{1, \ldots, p\}$에 대해, $\beta_M = (\beta_0, (\beta_j)_{j\in M}), X_M = (1, (x_j)_{j\in M})$($x_j$는 X의 j번째 열벡터)라 하고, 다음과 같은 선형 모델 $Y \sim N(X_M\beta_M, \sigma^2 I)$을 모델 M이라고 하자. 충분히 작은 x에 대해 $log(1+x)$를 x로 근사할 수 있다는 사실을 이용하여, 모든 $j \in M^C$에 대해 $\beta_j = 0$일때, 모델 M의 AIC를 Mallows's C_p로 근사해 보이시오.

(참고. $C_p = \frac{SSE_p}{S^2} + 2(|M|+1) - n$, SSE_p는 모델 M의 SSE,
$AIC_M = -2log\hat{L}(M) + 2(|M|+1)$

$AIC_M = -2\log\underbrace{\hat{L}(M)}_{\text{최대가능도}} + 2(|M|+1) \approx C_p = \frac{SSE_p}{S^2} + 2(|M|+1) - n$ (P: 파라메터 수)

$Y = X\beta + \varepsilon \sim N(X_M\beta_M, \sigma^2 I)$

* 노이즈의 분산에 대한 추정값을 σ^2, 목표값과 예측값을 각각 $y_i, \hat{y}_i$이라 할 때, 오차항이 정규분포를 따르는 OLS 모델의 최대로그가능도 $\log\hat{L} = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma^2 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{2\sigma^2}$ 이다. (교재 P.129)

위 식을 활용하여 AIC에 대입하면

$$AIC_M = -2\left(-\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma^2 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_{pi})^2}{2\sigma^2}\right) + 2p$$
$$= n\log 2\pi\sigma^2 + \frac{\sum_{i=1}^{n}(y_i-\hat{y}_{pi})^2}{\sigma^2} + 2p$$

* $SSE_p = \sum_{i=1}^{n}(y_i - y_{pi})^2$ & $\sigma^2 = \frac{SSE_p}{n-p}$

$$\approx n\log 2\pi\frac{SSE_p}{n-p} + n - p + 2p$$
$$= n\log 2\pi + n\log\frac{SSE_p}{n-p}$$

7. 교재 4장 되새김 문제 2번을 풀어보시오.

```
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso
X, y= make_regression(n_samples=300,
                      n_features=400,
                      n_informative=50,
                      n_targets=1,
                      bias=0.0,
                      noise=10.0,
                      random_state=1234)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=1234)
```

문제: (최소제곱법)OLS 모델로 (X_train, y_train) 쌍을 학습하고 이를 (X_test, y_test) 쌍에 적용하여 MSE를 계산하라.

라쏘 모델로도 동일한 분석을 한 후 이를 OLS의 결과와 비교하라.

1. OLS 모델로 MSE 계산

```
reg1= LinearRegression()
reg1= reg1.fit(X_train, y_train)

y_train_pred= reg1.predict(X_train)
print(f'train data 기준 OLS 모델의 MSE:{((y_train -
y_train_pred)**2).mean(): .2f}')

y_test_pred=reg1.predict(X_test)
print(f'test data 기준 OLS 모델의 MSE:{((y_test-y_test_pred)**2).mean():
.2f}')
```

결과:

```
train data 기준 OLS 모델의 MSE: 0.00
test data 기준 OLS 모델의 MSE: 67809.12
```

2. 라쏘 모델로 MSE 계산

```
reg2=Lasso()
reg2=reg2.fit(X_train, y_train)

y_train_pred=reg2.predict(X_train)
print(f'train data 기준 LASSO 모델의 MSE:{((y_train -
y_train_pred)**2).mean(): .2f}')

y_test_pred=reg2.predict(X_test)
print(f'test data 기준 LASSO 모델의 MSE:{((y_test -
y_test_pred)**2).mean(): .2f}')
```

결과:
train data 기준 LASSO 모델의 MSE: 130.41
test data 기준 LASSO 모델의 MSE: 457.40