숙제 1

1. [선형회귀] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (17점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
# 관련 라이브러리
import numpy as np
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
plt.show()
# (1) 화면 출력 확인

### 정규 방정식을 사용한 선형회귀 접근 ###
X_b = np.c_[np.ones((100, 1)), X]
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
# (2) theta_best 출력 확인

X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]
y_predict = X_new_b.dot(theta_best)
# (3) y_predict 출력 확인

plt.plot(X_new, y_predict, "r-", linewidth=2, label="prediction")
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([0, 2, 0, 15])
plt.show()
# (4) 화면 출력 확인
```

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
# (5) lin_reg.intercept_, lin_reg.coef_ 출력 확인

# (6) lin_reg.predict(X_new) 출력 확인

theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-6)
# (7) theta_best_svd 출력 확인

# (8) np.linalg.pinv(X_b).dot(y) 출력 확인

### 경사 하강법을 사용한 선형회귀 접근 ###
eta = 0.1
n_iterations = 1000
m = 100
theta = np.random.randn(2,1)
for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients
# (9) theta 출력 확인

# (10) X_new_b.dot(theta) 출력 확인

theta_path_bgd = []
def plot_gradient_descent(theta, eta, theta_path=None):
    m = len(X_b)
    plt.plot(X, y, "b.")
    n_iterations = 1000
    for iteration in range(n_iterations):
        if iteration < 10:
            y_predict = X_new_b.dot(theta)
            style = "b-" if iteration > 0 else "r--"
            plt.plot(X_new, y_predict, style)
        gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
        theta = theta - eta * gradients
        if theta_path is not None:
            theta_path.append(theta)
    plt.xlabel("$x_1$", fontsize=18)
```

```
    plt.axis([0, 2, 0, 15])
    plt.title(r"$₩eta = {}$".format(eta), fontsize=16)
np.random.seed(42)
theta = np.random.randn(2,1)
plt.figure(figsize=(10,4))
plt.subplot(131); plot_gradient_descent(theta, eta=0.02)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.subplot(132); plot_gradient_descent(theta, eta=0.1, theta_path=theta_path_bgd)
plt.subplot(133); plot_gradient_descent(theta, eta=0.5)
plt.show()
# (11) 화면 출력 확인

### 스토캐스틱 경사 하강법을 사용한 선형회귀 접근 ###
theta_path_sgd = []
m = len(X_b)
np.random.seed(42)
n_epochs = 50
t0, t1 = 5, 50
def learning_schedule(t):
    return t0 / (t + t1)
theta = np.random.randn(2,1)
for epoch in range(n_epochs):
    for i in range(m):
        if epoch == 0 and i < 20:
            y_predict = X_new_b.dot(theta)
            style = "b-" if i > 0 else "r--"
            plt.plot(X_new, y_predict, style)
        random_index = np.random.randint(m)
        xi = X_b[random_index:random_index+1]
        yi = y[random_index:random_index+1]
        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(epoch * m + i)
        theta = theta - eta * gradients
        theta_path_sgd.append(theta)
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
plt.show()
```

```
# (12) 화면 출력 확인

# (13) theta 출력 확인

from sklearn.linear_model import SGDRegressor
sgd_reg = SGDRegressor(max_iter=50, penalty=None, eta0=0.1, random_state=42)
# (14) sgd_reg.fit(X, y.ravel()) 출력 확인

# (15) sgd_reg.intercept_, sgd_reg.coef_ 출력 확인

### 미니배치 경사 하강법을 사용한 선형회귀 접근 ###
theta_path_mgd = []
n_iterations = 50
minibatch_size = 20
np.random.seed(42)
theta = np.random.randn(2,1)
t0, t1 = 200, 1000
def learning_schedule(t):
    return t0 / (t + t1)
t = 0
for epoch in range(n_iterations):
    shuffled_indices = np.random.permutation(m)
    X_b_shuffled = X_b[shuffled_indices]
    y_shuffled = y[shuffled_indices]
    for i in range(0, m, minibatch_size):
        t += 1
        xi = X_b_shuffled[i:i+minibatch_size]
        yi = y_shuffled[i:i+minibatch_size]
        gradients = 2/minibatch_size * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(t)
        theta = theta - eta * gradients
        theta_path_mgd.append(theta)
# (16) theta 출력 확인

theta_path_bgd = np.array(theta_path_bgd)
theta_path_sgd = np.array(theta_path_sgd)
theta_path_mgd = np.array(theta_path_mgd)

plt.figure(figsize=(7,4))
```

```
plt.plot(theta_path_sgd[:, 0], theta_path_sgd[:, 1], "r-s", linewidth=1, label="SGD")
plt.plot(theta_path_mgd[:, 0], theta_path_mgd[:, 1], "g-+", linewidth=2, label="MINI_BATCH")
plt.plot(theta_path_bgd[:, 0], theta_path_bgd[:, 1], "b-o", linewidth=3, label="BATCH")
plt.legend(loc="upper left", fontsize=16)
plt.xlabel(r"$₩theta_0$", fontsize=20)
plt.ylabel(r"$₩theta_1$     ", fontsize=20, rotation=0)
plt.axis([2.5, 4.5, 2.3, 3.9])
plt.show()
# (17) 화면 출력 확인
```

2. [다차항회귀] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (6점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
# 관련 라이브러리
import numpy as np
import numpy.random as rnd

np.random.seed(42)
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([-3, 3, 0, 10])
plt.show()
# (1) 화면 출력 확인

from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
# (2) X[0] 출력 확인

# (3) X_poly[0] 출력 확인

lin_reg = LinearRegression()
```

```
lin_reg.fit(X_poly, y)
# (4) lin_reg.intercept_, lin_reg.coef_ 출력 확인

X_new=np.linspace(-3, 3, 100).reshape(100, 1)
X_new_poly = poly_features.transform(X_new)
y_new = lin_reg.predict(X_new_poly)
plt.plot(X, y, "b.")
plt.plot(X_new, y_new, "r-", linewidth=2, label="prediction")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([-3, 3, 0, 10])
plt.show()
# (5) 화면 출력 확인

from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

for style, width, degree in (("g-", 1, 300), ("b--", 2, 2), ("r-+", 2, 1)):
    polybig_features = PolynomialFeatures(degree=degree, include_bias=False)
    std_scaler = StandardScaler()
    lin_reg = LinearRegression()
    polynomial_regression = Pipeline([
            ("poly_features", polybig_features),
            ("std_scaler", std_scaler),
            ("lin_reg", lin_reg),
        ])
    polynomial_regression.fit(X, y)
    y_newbig = polynomial_regression.predict(X_new)
    plt.plot(X_new, y_newbig, style, label=str(degree), linewidth=width)
plt.plot(X, y, "b.", linewidth=3)
plt.legend(loc="upper left")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([-3, 3, 0, 10])
plt.show()
# (6) 화면 출력 확인
```

3. [규제] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (2점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
# 관련 라이브러리
from sklearn.linear_model import Ridge

np.random.seed(42)
m = 20
X = 3 * np.random.rand(m, 1)
y = 1 + 0.5 * X + np.random.randn(m, 1) / 1.5
X_new = np.linspace(0, 3, 100).reshape(100, 1)
def plot_model(model_class, polynomial, alphas, **model_kargs):
    for alpha, style in zip(alphas, ("b-", "g--", "r:")):
        model = model_class(alpha, **model_kargs) if alpha > 0 else LinearRegression()
        if polynomial:
            model = Pipeline([
                    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
                    ("std_scaler", StandardScaler()),
                    ("regul_reg", model),
                ])
        model.fit(X, y)
        y_new_regul = model.predict(X_new)
        lw = 2 if alpha > 0 else 1
        plt.plot(X_new, y_new_regul, style, linewidth=lw, label=r"$₩alpha = {}$".format(alpha))
    plt.plot(X, y, "b.", linewidth=3)
    plt.legend(loc="upper left", fontsize=15)
    plt.xlabel("$x_1$", fontsize=18)
    plt.axis([0, 3, 0, 4])
plt.figure(figsize=(8,4))
plt.subplot(121)
plot_model(Ridge, polynomial=False, alphas=(0, 10, 100), random_state=42)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.subplot(122)
plot_model(Ridge, polynomial=True, alphas=(0, 10**-5, 1), random_state=42)
plt.show()
# (1) 화면 출력 확인
```

4. 다음 훈련집합을 3차원 공간에 그리고, 선형분리 가능 여부와 그 이유를 제시하시오.

$$\mathbf{x}_1 = \begin{pmatrix}0\\0\\0\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}0\\1\\1\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}1\\1\\1\end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix}1\\0\\1\end{pmatrix}, \mathbf{x}_5 = \begin{pmatrix}0\\0\\1\end{pmatrix}, \mathbf{x}_6 = \begin{pmatrix}0\\1\\0\end{pmatrix}$$

$$y_1 = 1, y_2 = 1, y_3 = 1, y_4 = -1, y_5 = -1, y_6 = -1$$

5. $\mathbf{A} = \begin{pmatrix}1 & -2 & 3 & 5\\ 2 & 2 & -1 & 0\\ 3 & 0 & 1 & 2\\ 1 & 0 & 2 & 0\end{pmatrix}$의 $2\mathbf{A}, \mathbf{A}^{\mathrm{T}}, \mathbf{A}^{-1}$을 쓰시오. 또한 $\mathbf{A}$의 계수를 쓰시오.

더불어 행렬식, 고유 분해, 특이값 분해도 쓰시오. (python 사용)

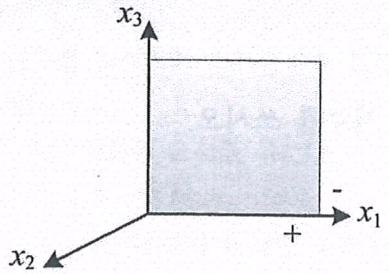
6. 놈을 계산하시오.

(1) $\mathbf{x} = (3 \ \ -4 \ \ -1.2 \ \ 0 \ \ 2.3)^{\mathrm{T}}$의 1차, 2차, 3차 놈과 최대 놈

(2) $\begin{pmatrix}2 & 1\\ 1 & 5\\ 4 & 1\end{pmatrix}$의 프로베니우스 놈

7. [예제 2-3]에서 $T = 2.0$으로 바뀌었다고 가정하고 [그림 2-4(b)]를 새로 그리시오. T가 변함에 따라 어떤 변화가 나타나는지 설명하시오.

8. 아래 그림은 x_1과 x_3 축이 이루는 평면이 결정평면인 상황이다. 이에 해당하는 퍼셉트론을 [그림 2-4(a)]처럼 제시하시오.



9. 다음 합성함수에 대해 답하시오.

$$f(x) = 2\left(\frac{1}{4}(1-2x)^2 - 1\right)^3 - 3\left(\frac{1}{4}(1-2x)^2 - 1\right)^2 - 3$$

(1) 식 (2.53)에 따라 $i(x)$와 $h(x)$를 쓰시오.

(2) 연쇄법칙을 이용하여 $f'(x)$를 구하시오.

(3) $f'(0)$과 $f'(2.1)$을 계산하시오.

10. 다음 함수에 대해 답하시오.

$$f(x_1, x_2) = 2x_1^2 + 3x_1x_2 + 2x_2^2 - 4x_1 + 2x_2 - 24$$

(1) 최소점과 최솟값을 분석적으로 구하시오.

(2) 난수를 생성하여 초깃값 $\mathbf{x}_0 = (1.0, 0.9)^T$를 얻었다고 가정하고, 식 (2.58)을 연속으로 적용하여 얻는 점 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$을 구하시오. 이때 학습률 $\rho = 0.1$을 사용하시오. (1)에서 구한 최소점을 향해 이동하는지 확인하시오.

11. (10점) 몬티홀Monty Hall 게임 쇼에서 참가자들은 다음과 같은 결과를 말하고 있다.

1, 2, 3번이 붙어있는, 세 개의 문이 있다. 한 개의 상품이 세 개의 문 뒤에 있다. 하나의 문을 선택할 수 있으며, 처음에 선택된 문은 열리지 않을 것이다. 대신에 게임 진행자는 다른 두개의 문 중 하나를 열며, 그는 상품이 드러나지 않게 문을 열 것이다. 예를 들어 당신이 1번 문을 처음에 선택한다면 그는 2번과 3번 중 하나를 열고, 그가 선택해서 열리는 문에는 상품이 나타나지 않을 것이다.

여기서 진행자는 당신은 새롭게 문을 선택할 수 있도록 할 것이다. 당신은 첫 번째 선택을 그대로 유지할 수 있거나, 다른 닫힌 문으로 선택을 바꿀 수 있다. 모든 문이 열리고 당신이 마지막으로 선택한 문 뒤에 무엇이 있는지 확인한다.

- **참가자는 처음에 1번 문을 선택한다고 가정하며, 게임 진행자는 3번 문을 열고 문 뒤에 아무것도 없음을 보장한다. 참가자는 (a) 1번 문에 머물거나, (b) 2번 문으로 변경, 또는 (c) 무엇이든 상관이 없다. 어느 전략이 좋은지 제시하시오.**

처음 상품이 3개의 문 뒤에 동일하게 있다고 가정하며, 베이즈 규칙을 사용하라

표시된 점수 외의 문제의 점수는 모두 5점