숙제 1

제출 주의 사항:

- 하나의 PDF문서만 제출 (반드시 코드 포함. 없으면 풀이 불인정)

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()
# 화면 출력 확인 및 결과 해석
```

4. [활성함수] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
# 파이썬 2, 파이썬 3 지원
from __future__ import division, print_function, unicode_literals

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

def logit(z):
    return 1 / (1 + np.exp(-z))

def relu(z):
    return np.maximum(0, z)

def derivative(f, z, eps=0.000001):
    return (f(z + eps) - f(z - eps))/(2 * eps)

z = np.linspace(-5, 5, 200)

plt.figure(figsize=(11,4))

plt.subplot(121)
plt.plot(z, np.sign(z), "r-", linewidth=2, label="step")
plt.plot(z, logit(z), "g--", linewidth=2, label="sigmoid")
```

```
plt.plot(z, np.tanh(z), "b-", linewidth=2, label="tanh")
plt.plot(z, relu(z), "m-.", linewidth=2, label="ReLU")
plt.grid(True)
plt.legend(loc="center right", fontsize=14)
plt.title("activation function: g(z)", fontsize=14)
plt.axis([-5, 5, -1.2, 1.2])

plt.subplot(122)
plt.plot(z, derivative(np.sign, z), "r-", linewidth=2, label="step ")
plt.plot(0, 0, "ro", markersize=5)
plt.plot(0, 0, "rx", markersize=10)
plt.plot(z, derivative(logit, z), "g--", linewidth=2, label="sigmoid")
plt.plot(z, derivative(np.tanh, z), "b-", linewidth=2, label="tanh")
plt.plot(z, derivative(relu, z), "m-.", linewidth=2, label="ReLU")
plt.grid(True)
plt.title("gradient: g'(z)", fontsize=14)
plt.axis([-5, 5, -0.2, 1.2])

plt.show()
# 화면 출력 확인 및 각 활성함수의 특징을 비교 서술
```

5. [오류 역전파] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (4점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)
```

```
# (1) 해당 연산망의 그래프 연산을 손으로 작성

grad_c = 1.0
grad_b = grad_c * np.ones ((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a*y
grad_y = grad_a*x
# (2) 위의 연산을 통한 grad_c, grad_b, grad_a, grad_z, grad_x, grad_y 출력 확인

import torch
x = torch.randn(N, D, requires_grad=True)
y = torch.randn(N, D, requires_grad=True)
z = torch.randn(N, D)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()
# (3) 역전파 함수 backward()를 이용한 x의 미분, y의 미분 출력 확인

# (4) (2)와 (3)의 방법의 차이를 설명
```

6. [신경망 학습] 다음 코드를 무엇을 의미하는지 이해하고 실행하여 결과를 확인하세요. (3점)

(코드의 해석과 결과의 의미를 작성하세요.)

```
import torch
N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 10e-6
```

```
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()

    with torch.no_grad():
        w1 -= learning_rate * w1.grad
        w2 -= learning_rate * w2.grad
        w1.grad.zero_()
        w2.grad.zero_()
# 매 t마다 y_pred에 따른 loss 변화를 화면 출력 확인 (plot)하고, 결과 해석
```

7. [probability; 10p] 직원이 A제조사로부터 1000개의 직접회로 (IC)를, B제조사로부터 2000개의 IC를, C제조사로부터 3000개의 IC를 구매했다. IC의 불량 검사 결과, A사로부터 구매한 IC의 불량 확률은 0.05, B사로부터 구매한 IC의 불량 확률은 0.10, C사로부터 구매한 IC의 불량 확률은 0.10이었다. (6점)

(1) 만약 3개의 제조사로부터 구매한 IC가 섞여 있는 경우, 임의로 선택한 IC가 불량일 확률은 얼마인가?

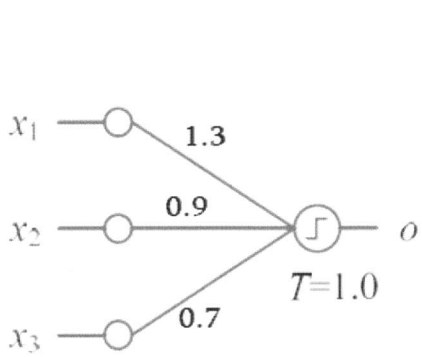
(2) 임의로 선택한 IC가 불량인 경우, 그것이 제조사 A로부터 만들어질 확률은 얼마인가?

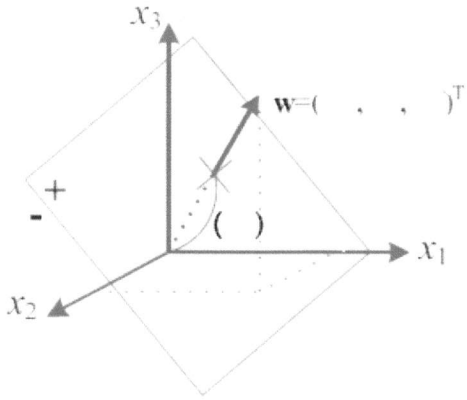
8. [probability; 10p] K대학은 대학원생보다 2배의 학부생이 재학중이다. 대학원생의 25%가 기숙사에 살고 있고, 학부생의 10%가 기숙사에 살고 있다. (6점)

(1) 한 학생을 임의로 선정한 경우, 그 학생이 기숙사에 살고 있는 학부생일 확률은 얼마인가?

(2) 기숙사에 살고 있는 한 학생을 임의로 선정한 경우, 그 학생이 대학원생일 확률은 얼마인가?

9. [퍼셉트론] 다음 그림의 퍼셉트론 (perceptron) 입니다. (6점)





(1) 해당 퍼셉트론에 의해 결정되는 결정평면의 방향과 원점에서의 거리를 구하세요.

(2) T=2.0, T=0.0으로 바꾸기 위해 퍼셉트론을 수정하고, 결정평면의 변화를 설명하세요.

10. [PLA] PLA 가중치 갱신 법칙 $w(t+1) = w(t) + y(t)x(t)$를 보고 다음 문제의 답을 보이세요. (20점)

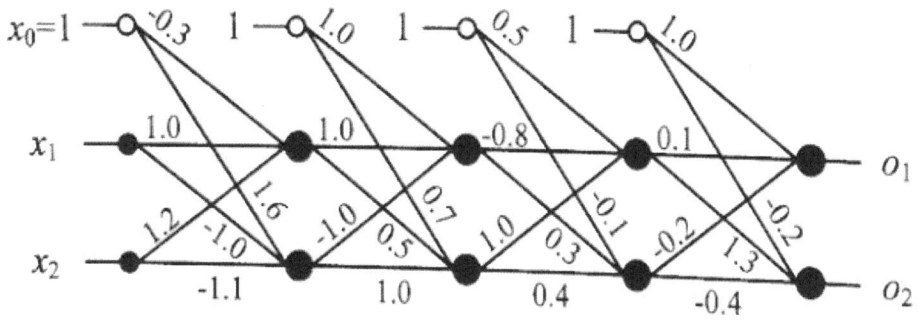
(1) $y(t)w^T(t)x(t) < 0$임을 보이세요. (Hint: $x(t)$는 $w(t)$에 의해 오분류 됨)

(2) $y(t)w^T(t+1)x(t) > y(t)w^T(t)\,x(t)$ 임을 보이세요. (Hint: $w(t+1) = w(t) + y(t)x(t)$ 이용)

(3) $w(t)$에서 $w(t+1)$로 이동하는 것이 $x(t)$를 분류하는데 올바른 방향으로 이동함을 설명하세요.

(4) $\mathbf{w} = [\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2]^{\mathrm{T}}$이고, $\mathbf{x} = [1, \mathrm{x}_1, \mathrm{x}_2]^{\mathrm{T}}$인 $h(x) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ 일 때, $h(\mathbf{x}) = -1$와 $h(\mathbf{x}) = 1$는 결정 직선 $\mathrm{x}_2 = \mathrm{ax}_1 + b$ 에 의해 구분된다. 결정 직선의 기울기 a와 절편 b를 가중치 $\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2$에 의해 설명하세요.

11. [MLP] 다음은 은닉층이 3개인 MLP입니다. (16점)



(1) 가중치 행렬 $\mathbf{U}^1$, $\mathbf{U}^2$, $\mathbf{U}^3$, $\mathbf{U}^4$를 식 (4.1)처럼 쓰세요.

(2) $\mathbf{x}=(1,0)^T$가 입력되었을 때 출력 $\mathbf{o}=(o_1,o_2)^T$를 구하세요. 활성함수는 로지스틱 시그모이드를 사용하세요.

(3) $\mathbf{x}=(1,0)^T$가 입력되었을 때 출력 $\mathbf{o}=(o_1,o_2)^T$를 구하세요. 활성함수는 ReLU를 사용하세요.

(4) $\mathbf{x}=(1,0)^T$의 기대 출력이 $\mathbf{o}=(0,1)$일 때, 현재 1.0인 u^3_{12} 가중치를 0.9로 줄이면 오류에 어떤 영향을 미치는지 설명하세요.

12. 아래 그림의 연산 그래프 예처럼 f(x, y, z)= (x+y)z 연산에 대한 연산 그래프를 새롭게 생성하고, x=-2, y=5, z=-4인 경우에 전방 전파와 이에 대응되는 오류 역전파를 각 가중치마다 계산하세요. (10점)

[아래 예에 표시된 것처럼 전방 전파 연산 결과는 검은색 빈칸, 오류 역전파 연산 결과는 빨간색 빈칸으로 표시하여 구분하세요]