```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


'''파일 받아오기'''


M0 = pd.read_csv('C:₩₩Users₩₩kim07₩₩Downloads₩₩lin_regression_data_01.csv',
header = None)    #데이터파일 불러오기

M0 = M0.to_numpy(dtype = float)
#numpy array 형식으로 변환


x_vector = M0[:,0]
#weight data set

y_vector = M0[:,1]
#length data set

x_dummy = np.array([1]*50)
#x_dummy data

x_matrix = np.column_stack([x_vector, x_dummy])
#dummy 포함된 x matrix




'''GDM 함수'''

#과제 2 함수

def GDM(epoch, learning_rate):
#변화 시킬 variable


    w_hist = []
#w값 저장할 list
```

```
    MSE_hist = []
#MSE값 저장할 list


    #epoch만큼 반복해 weight 업데이트
    for i in range(epoch):
        if i == 0:
            w = np.random.rand(2) * 5
#w0, w1값을 0과 5사이에 생성


        y_hat = np.dot(x_matrix, w)
#y_hat 행렬곱 => 결과값이 vector
        error = y_hat - y_vector
#eror 값 구하기 => 결과값 vector
        error = error.reshape(-1, 1)
#단순 곱 위해 사이즈 (50,)에서 (50,1)로 만듦
        MSE = np.mean(error**2)
#MSE구하기


        w_hist.append(w)
#w_hist에 w값 저장
        MSE_hist.append(MSE)
#MSE_hist에 MSE값 저장


        w_dif = sum(2 * error * x_matrix)/len(y_hat)
#w의 기울기 구하기 => size가 2인 vector(w0, w1)
        w = w - learning_rate*w_dif
#weight update
```

```
    return w, w_hist, MSE_hist
#이 함수에서 w, w_hist, MSE_hist값을 받아옴




'''함수의 사용'''


epoch = 6000
# epoch 설정

learning_rate = 0.006
# learning_rate 설정(0.0075부터는 발산 해버림)



w_opt, w_list, MSE_list = GDM(epoch, learning_rate)
#w_opt, w_list, MSE_list에 GDM함수 return값 받기

w_list = np.array(w_list)
#w_list는 list이기 때문에 numpy array로 바꿔줌

MSE_graph = np.array(MSE_list)
#list numpy array로 바꾸기


w0_graph = w_list[:, 0]
#그래프용 w0

w1_graph = w_list[:, 1]
#그래프용 w1


w0_opt = w_opt[0]
#optimal solution w0

w1_opt = w_opt[1]
```

```
#optimal solution w1


epoch_graph = epoch
#그래프용 epoch

x_fir = np.min(x_vector) - 3
#x 데이터 최솟값보다 3 적은 범위 설정

x_las = np.max(x_vector) + 3
#x 데이터 최댓값보다 3 많은 범위 설정

x_opt = np.arange(x_fir, x_las, 1)
#위의 범위에서 1씩 증가하는 x

y_opt = w0_opt * x_opt + w1_opt
#그 때의 y


x_point_graph = M0[:,0]
#그래프에 그릴 x를 x_point_graph에 받는다.

y_point_graph = M0[:,1]
#그래프에 그릴 x를 x_point_graph에 받는다.



'''과제 2-4위한 simple linear regrassion'''

w0 = np.mean((y_vector)*(x_vector - np.mean(x_vector)))/(np.mean(x_vector**2)-
(np.mean(x_vector))**2)    #np.mean을 이용해서 1/n*시그마연산을 계산함

w1 = np.mean(y_vector - w0*x_vector)


x_start = 0
# x_vector의 최솟값이 5.3이므로 앞으로의 예측을 위해 더 넓게 0으로 설정

x_end = 20
# x_vector의 최댓값이 17.9 이므로 앞으로의 예측을 위해 더 넓게 20으로 설정
```

```
x_step = (x_end - x_start) / 50
#우리가 가진 data vector의 크기가 50이기에 x범위를 50으로 나눠주어 크기를 맞춰줌

x = np.arange(x_start, x_end , x_step)  #x의 값이 일정하게 증가하는 size 50의 벡터 생성

y_hat = w0*x + w1    #단순 선형 조합으로 예측값 y_hat 작성

y_hat_graph = y_hat


'''그래프 그리기'''


#rcParams의 기본값을 수정함 (x축, y축 label 크기, 제목크기, 눈금크기, 범례크기)

parameters = {"axes.labelsize": 20, "axes.titlesize": 30, 'xtick.labelsize': 12, "ytick.labelsize": 12, "legend.fontsize": 12}

plt.rcParams.update(parameters)



f, axes = plt.subplots(1, 3)


#weight graph

axes[0].plot(w0_graph, 'r-^', markevery = 200, label = 'w0')
#w0그래프 200마다 세모로 표시함

axes[0].plot(w1_graph, 'b-o', markevery = 200, label = 'w1')
#w1그래프 200마다 동그라미로 표시함

axes[0].set_xlabel("epoch")
#x축을 epoch이라고 이름 붙임

axes[0].set_ylabel("w0 w1")
#y축을 w0, w1로 이름 붙임

axes[0].set_title('data')
#제목 data라 설정
```

```
axes[0].legend(['w0',' w1'])
#범례 설정

axes[0].grid(True)
#눈금 표시


#MSE graph

axes[1].plot(MSE_graph, 'g-d', markevery = 200, label = 'MSE')

axes[1].set_xlabel("epoch")

axes[1].set_ylabel("MSE")

axes[1].set_title('mean square error')

axes[1].legend(['MSE'])

axes[1].grid(True)


#optimal solution graph with data

axes[2].plot(x_opt, y_opt, 'r-x')

axes[2].set_xlabel("weight(g)")

axes[2].set_ylabel("length(cm)")

axes[2].set_title('optimal solution')

axes[2].legend(['Gdm optimal solution'])

axes[2].grid(True)


fig = plt.figure()

plt.plot(x_opt, y_opt, 'r-x')

plt.plot(x, y_hat_graph, 'b')

plt.scatter(x_point_graph, y_point_graph)
#점 찍기
```

```
plt.xlabel("weight(g)")

plt.ylabel("length(cm)")

plt.title('optimal solution')

plt.legend(['Gdm optimal solution','Simple Linear Regrassion', 'data'])

plt.grid(True)


print("y_opt =",w0_opt,"* x +",w1_opt )
#과제 2 optimal solution


'''과제3'''


M1 = pd.read_csv('C:₩₩Users₩₩kim07₩₩Downloads₩₩lin_regression_data_02.csv')
#데이터파일 불러오기

M1 = M1.to_numpy(dtype = float)
#numpy array float형으로 변환


x_work3_matrix = M1[:,0:2]
#데이터의 0, 1번 column 성분들 저장(50,2)

y0_vector = M1[:, 2]
#데이터의 2번 column 성분들 저장(50,1)

x_work3_dummy = np.array([1]*50)
#더미데이터 생성

x_work3_matrix = np.column_stack([x_work3_matrix, x_work3_dummy])
#np.column_stack을 이용해 (50,3) 행렬 만들기



def GDM_work3(epoch_work3, learning_rate_work3):
#변화 시킬 variable
```

```
    w_hist = []
#w값 저장할 list

    MSE_hist = []
#MSE값 저장할 list


    #epoch만큼 반복해 weight 업데이트

    for i in range(epoch_work3):

        if i == 0:

            w = np.random.rand(3) * -10
#w0, w1값을 0과 -10사이에 생성


        y_hat = np.dot(x_work3_matrix, w)
#y_hat 행렬곱 => 결과값이 vector

        error = y_hat - y0_vector
#eror 값 구하기 => 결과값 vector

        error = error.reshape(-1, 1)
#단순 곱 위해 사이즈 (50,)에서 (50,1)로 만듦

        MSE = np.mean(error**2)
#MSE구하기


        w_hist.append(w)
#w_hist에 w값 저장

        MSE_hist.append(MSE)
#MSE_hist에 MSE값 저장


        w_dif = sum(2 * error * x_work3_matrix)/len(y_hat)
#w의 기울기 구하기 => size가 2인 vector(w0, w1)
```

```
            w = w - learning_rate_work3*w_dif
#weight update


      return w, w_hist, MSE_hist
#반환시킬 값



'''함수의 사용'''

epoch_work3 = 100
#100번 반복

learning_rate_work3 = 0.1
#learning_rate 0.1 설정 (0.7부터 발산)


x0_point = M1[:, 0]

x1_point = M1[:, 1]

y0_point = M1[:, 2]


w_work3_opt, w_work3_list, MSE_work3_list = GDM_work3(epoch_work3,
learning_rate_work3) #GDM_work3 함수 사용

w_work3_list = np.array(w_work3_list)
#w, MSE numpy array로 받기

MSE_work3_list = np.array(MSE_work3_list)


w0_work3_opt = w_work3_opt[0]
#optimal solution의 weight 받기

w1_work3_opt = w_work3_opt[1]

w2_work3_opt = w_work3_opt[2]
```

```
w0_work3_graph = w_work3_list[:, 0]
#w값 50개 하나씩 받기

w1_work3_graph = w_work3_list[:, 1]

w2_work3_graph = w_work3_list[:, 2]


x0_fir_work3 = np.min(x0_point) - 3
#optimal solution의 x값 범위 지정하기

x0_las_work3 = np.max(x0_point) + 3

x1_fir_work3 = np.min(x1_point) - 3

x1_las_work3 = np.max(x1_point) + 3

x0_opt_work3 = np.arange(x0_fir_work3, x0_las_work3, 1)

x1_opt_work3 = np.arange(x1_fir_work3, x1_las_work3, 1)


X0, X1 = np.meshgrid(x0_opt_work3, x1_opt_work3)
#2개의 1차원 배열로 2차원 행렬 만들기


y_hat_fir = w0_work3_graph[0] * X0 + w1_work3_graph[0] * X1 + w2_work3_graph[0] #초기값 weight로 만든 y_hat


y_hat_opt = w0_work3_opt * X0 + w1_work3_opt * X1 + w2_work3_opt
#optimal solution y_hat


y_hat_pred = w0_work3_opt * x0_point + w1_work3_opt * x1_point + w2_work3_opt


#Optimal solution

print("y =", w0_work3_opt,"* x0 +", w1_work3_opt, "* x1 +", w2_work3_opt)
```

#과제 3 optimal solution

'''gpt가 알려준 3d 그래프 그리는 법'''

```
# 과제 3-1: 받아온 데이터 3d그래프로 점 표시하기
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x0_point, x1_point, y0_point)
ax.set_xlabel("x0")
ax.set_ylabel("x1")
ax.set_zlabel("y0")

# 과제 3-2: 초기 가중치에 대한 y_hat을 3차원축에 평면으로 나타내기
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(x0_point, x1_point, y0_point)
ax.plot_surface(X0, X1, y_hat_fir, cmap = 'coolwarm', alpha = 0.5)
ax.set_xlabel("x0")
ax.set_ylabel("x1")
ax.set_zlabel("y0")
ax.legend(['data', 'y_hat_fir'])

# 과제 3-3: optimal solution에 대해 w와 mse의 변화값 그래프에 나타내기
f, axes = plt.subplots(1, 2)
```

```
axes[0].plot(w0_work3_graph, 'r-^', markevery = 10, label = 'w0')
axes[0].plot(w1_work3_graph, 'b-o', markevery = 10, label = 'w1')
axes[0].plot(w2_work3_graph, 'g-x', markevery = 10, label = 'w2')
axes[0].set_xlabel("epoch")
axes[0].set_ylabel("weights")
axes[0].set_title('data')
axes[0].legend(['w0',' w1', 'w2'])
axes[0].grid(True)



axes[1].plot(MSE_work3_list, 'g-d', markevery = 10, label = 'MSE')
axes[1].set_xlabel("epoch")
axes[1].set_ylabel("MSE")
axes[1].set_title('mean square error')
axes[1].legend(['MSE'])
axes[1].grid(True)

# 과제 3-4: y와 optimal solution의 y_hat 점찍고 y_hat 3d그래프로 나타내기
fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(x0_point, x1_point, y0_point, marker = 'x')
ax.scatter(x0_point, x1_point, y_hat_pred, marker = '^')
ax.plot_surface(X0, X1, y_hat_opt, cmap = 'coolwarm', alpha = 0.5)
#alpa로 투명도 조절
ax.set_xlabel("x0")
```

```
ax.set_ylabel("x1")

ax.set_zlabel("y0")

ax.legend(['data', 'predicted', 'y_hat_opt'])
```