전자공학과 2020142001 곽종근

4주차 선형회귀 실습과제1,2

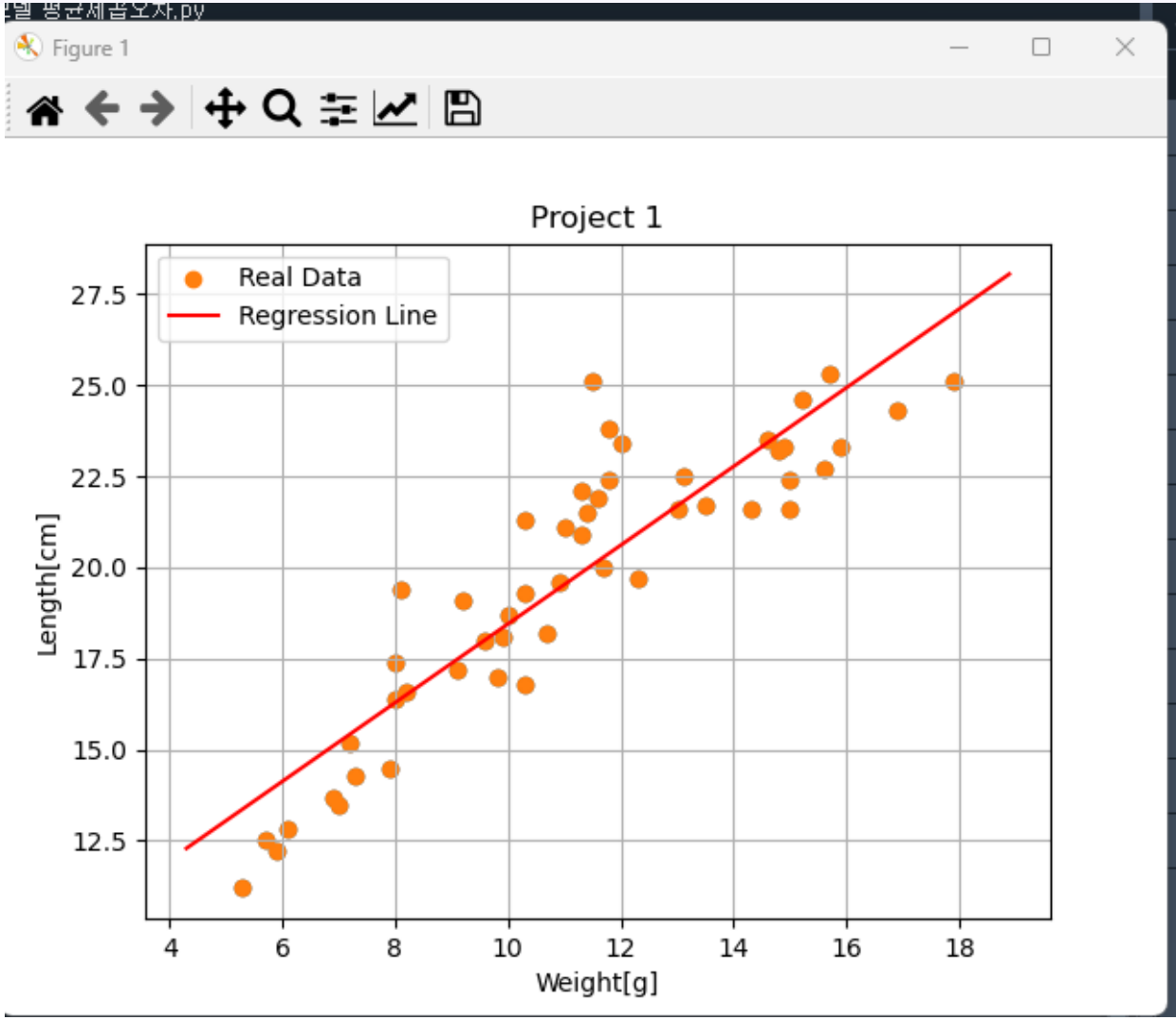
제출일: 2024.04.10.

실습 과제 1. 코드

temp.py × 실습과제1. 선형회귀모델 평균제곱오차.py ×

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

fold_dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\Machine-Learning\\5주차\\lin_regression_data_01.csv"
temp_data = pd.read_csv(fold_dir, header=None)
temp_data = temp_data.to_numpy()  #위에서 읽은 temp_data는 dataframe형태이므로 이를 numpy배열로 변환하여 slice이용한다.

Wei = np.zeros([50])  # x축 값인 Wei(무게)를 zeros 50으로 선언해준다.
Len = np.zeros([50])  # y축 값인 Len(길이)를 zeros 50으로 선언해준다.


for j in range(50):  # 파일을 열어보면 총 개수가 50개이므로 50번 반복하도록 반복문 실행
    Wei[j] = temp_data[j, 0]  # 위에서 선언한 데이터는 50by2가 되는데 여기서 0번째 열의 데이터만 Wei에 저장
    Len[j] = temp_data[j, 1]  # 1번째 열의 데이터만 Len에 저장해준다.

Wei_Len = sum(Wei * Len)   #Wo과 Wi를 구하기 위해 Wei_Len이라는 것을 따로 선언하여 이를 위에 선언한 Wei와 Len배열의 곱한 값의 합으로 선언
Wei_squared = sum(Wei**2)   #Wo과 Wi를 구하기 위해 Wei_squared를 따로 선언해 이를 Wei의 제곱의 합으로 선언

plt.scatter(Wei, Len)  # 점으로 Wei와 Len을 선언한다.
plt.legend(['RealData']) #위에서 선언한 점의 이름을 화면에 띄운다.


Wei_Sum=sum(Wei)  #Wei의 총 합을 Wei_Sum으로 선언
Len_Sum=sum(Len)    #Len의 총 합을 Len_Sum으로 선언


Wo=((Wei_Len/50)-((Wei_Sum/50)*(Len_Sum/50)))/((Wei_squared/50)-(Wei_Sum/50)**2)  #위에서 선언해준 것들을 이용해 Wo의 식 정리
Wi=(Len_Sum-Wo*Wei_Sum)/50  #Wi의 식을 정리한다.



# 그래프 그리기
plt.scatter(Wei, Len, label='Real Data')  # 실제 데이터 산점도
plt.xlabel('Weight[g]')  # x축을 무게 Weight[g]으로 설정
plt.ylabel('Length[cm]')  # y축을 길이 Length[cm]로 설정
plt.title('Project 1')  # 그래프 제목 설정
plt.grid(True)  # 격자를 표시해준다.


x_values = np.arange(Wei.min() - 1, Wei.max() + 1, (Wei.max() + 1 - (Wei.min() - 1)) / 1000)  # x 값 범위 설정
#x의 범위 설정에서 일단 min을 이용해 Wei의 최소값에서 -1을 한 값을 시작으로 하고, max를 이용해 Wei의 최대값에 +1을 이용해 마지막 값을 선언한다
#그 후 간격을 설정하는데 그 간격을 임의로 1000개로 선언하기 위해 처음값과 끝값을 더해 1000으로 나누면 그 간격이 된다.
y_values = Wo * x_values + Wi  # y^(예측값)의 식을 써주면 y의 범위가 정해진다.
plt.plot(x_values, y_values, color='red', label='Regression Line')  # 회귀선 그리기

fore_value = Wo * Wei + Wi  #위에서 예측값을 위한 Wo과 Wi를 구했으므로 y=Wo x+ Wi 식을 이용해 식을 작성해준다.
mse = (sum((fore_value-Len)**2))/50  #mse의 식을 위에서 정해준 값들로 식을 작성해준다.
print("MSE:", mse) #mse의 값을 표현

plt.legend()  # 범례 표시
plt.show  # 그래프 출력
```

실습 과제 1-2. 출력 결과



Name	Type	Size	Value
fold_dir	str	101	C:\Users\user\OneDrive - 한국공…
fore_value	Array of float64	(50,)	[20.60846134 23.09282093 13.37… 18.7 ...
j	int	1	49
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_Sum	float64	1	981.6
mse	float64	1	2.684614692248344
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_Len	float64	1	11433.92
Wei_squared	float64	1	6657.919999999999
Wei_Sum	float64	1	554.8
Wi	float64	1	7.646585177675327
Wo	float64	1	1.080156346640652
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y_values	Array of float64	(1000,)	[12.29125747 12.30702775 12.32… 28.0 ...

X축에는 무게 Weight[g], y축은 길이 Length[cm]으로 나타냈고 그래프에 점으로 각 데이터를 표시하였고 같은 그래프에 선형회귀의 Analytic solution을 구해서 그래프로 표시하였다.

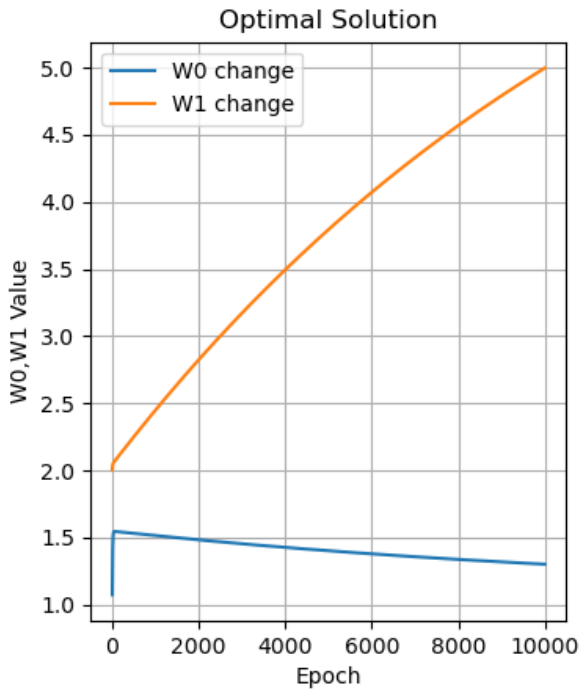
실습 과제 2 . 코드

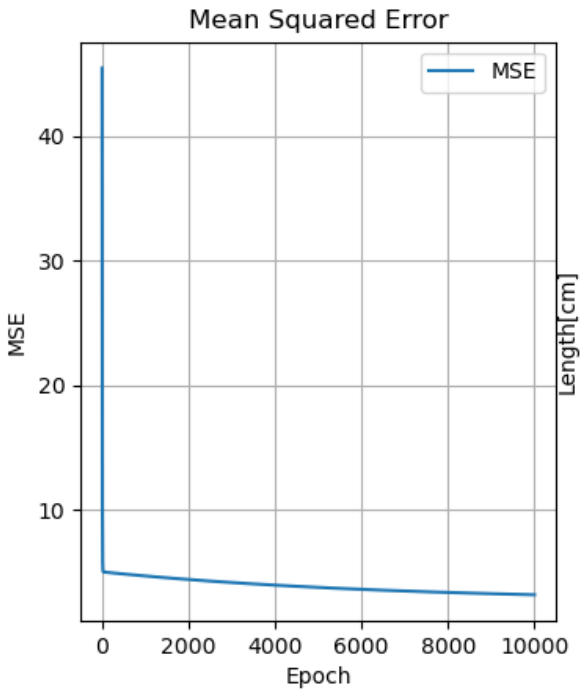
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 데이터 불러오기
fold_dir = "C:\\Users\\user\\OneDrive - 한국공학대학교\\바탕 화면\\3학년 1학기\\머신러닝실습\\Machine-Learning\\5주차\\lin_regression_data_01.csv"
temp_data = pd.read_csv(fold_dir, header=None)
temp_data = temp_data.to_numpy()    #data를 numpy로 불러와서 슬라이싱등 할 수 있도록 선언한다.

# 데이터 분리
Wei = temp_data[:, 0]  # 무게 데이터를 Wei저장
Len = temp_data[:, 1]  # 길이 데이터를 Len에 자ㅣ장

# 초기화와 함께 값을 설정해주고, 값의 변화를 주고자 하면 여기서 변화를 주면 된다
w0 = 1
w1 = 2
Learn_R = 0.001
rp = 10000

# 경사하강법 사용자 지정 함수를 만들어준다.
def GDM(x, y, Learn_R, rp):   # x는 w0, y는 w1, Learn_R은 알파, rp는 반복횟수

    Wei_new = []    #미분하여 변화하는 Wei값들은 저장하는 배열을 만들어준다.
    Len_new = []    #미분하여 변화하는 Len값들은 저장하는 배열을 만들어준다.
    MSE_new = []    #변화하는 MSE값들은 저장하는 배열을 만들어준다.

    for i in range(rp):          #설정한 반복횟수 rp만큼
        new_y = x * Wei + y      #new_y는 예측값y^을 의미한다.
        error = new_y - Len      #아래의 미분식과 mse를 편하게 계산하기 위해 error로 예측값 - 실제값을 선언
        dif_W0 = np.mean(error * Wei) #W0미분식
        dif_W1 = np.mean(error)      #W1미분식

        x = x - (Learn_R * dif_W0) # x는 계속해서 새로 변화하는 w0값을 저장해주고 최종적으로 마지막 x에는 최종w0값이 저장됨
        y = y - (Learn_R * dif_W1)   # y는 계속해서 새로 변화하는 w1값을 저장해주고 최종적으로 마지막 x에는 최종w0값이 저장됨
        mse = np.mean(error ** 2)  # 평균 제곱 오차 계산식

        Wei_new.append(x)   #위에서 구해지는 변화하는 w0,w1,mse값들을 위에서 만들어준 빈 배열에 계속해서 넣어준다.
        Len_new.append(y)

        MSE_new.append(mse)

    return x, y, Wei_new, Len_new, MSE_new

# 경사 하강법 수행하는데 x,y는 여기서 결과적으로 최종w0,w1값을 의미하게 된다.
x, y, Wei_new, Len_new, MSE_new = GDM(w0, w1, Learn_R, rp)
```

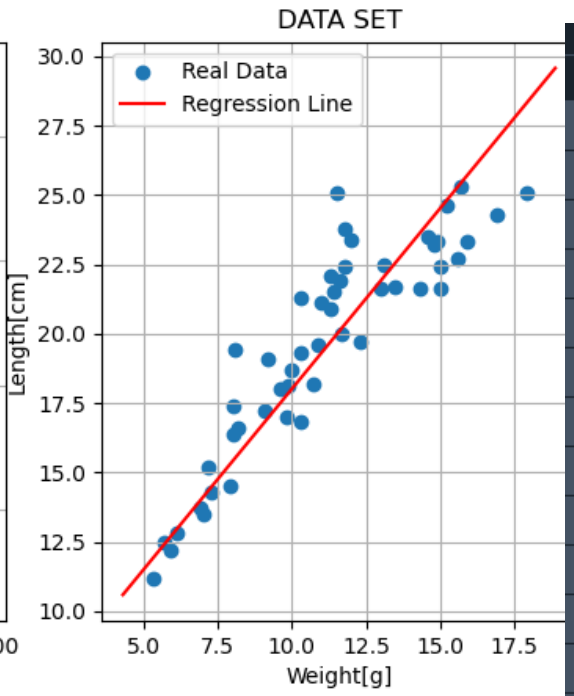
실습 과제 2 . 코드

```
# w0, w1 변화 그래프
plt.subplot(1, 3, 1)   #1행 3열 1번에 선언한다는 의미이다
plt.plot(Wei_new, label='W0 change')  # w0의 변화 그래프
plt.plot(Len_new, label='W1 change')  # w1의 변화 그래프
plt.xlabel('Epoch') #x축 이름을 Epoch로 설정
plt.ylabel('W0,W1 Value') #y축 이름을 w0,w1 Value로 설정
plt.title('Optimal Solution') #이 그래프의 제목을 Optimal Solution으로 설정
plt.grid()
plt.legend()

# MSE 변화 그래프
plt.subplot(1, 3, 2)   #1행 3열 2번에 선언
plt.plot(MSE_new, label='MSE')
plt.xlabel('Epoch') #x축 이름을 Epoch로 설정
plt.ylabel('MSE') #y축 이름을 mse로 설정
plt.title('Mean Squared Error') #이 그래프의 제목을 Mean Squared Error으로 설정
plt.grid()
plt.legend()

# 예측 결과 그래프
plt.subplot(1, 3, 3)
plt.scatter(Wei, Len, label='Real Data')
x_values = np.arange(Wei.min() - 1, Wei.max() + 1, (Wei.max() + 1 - (Wei.min() - 1)) / 1000)
#x의 범위 설정에서 일단 min을 이용해 Wei의 최소값에서 -1을 한 값을 시작으로 하고, max를 이용해 Wei의 최대값에 +1을 이용해 마지막 값을 선언한다
#그 후 간격을 설정하는데 그 간격을 임의로 1000개로 선언하기 위해 처음값과 끝값을 더해 1000으로 나누면 그 간격이 된다.
yy = x * x_values + y  # 회귀선
# y^(예측값)의 식을 써주면 y의 범위가 정해진다.
plt.plot(x_values, yy, c='r', label='Regression Line')  # 회귀선 그래프
plt.xlabel('Weight[g]')
plt.ylabel('Length[cm]')
plt.title('DATA SET')
plt.grid()
plt.legend(loc='upper left')


plt.show()
```

실습 과제 2 . 출력 초기값 설정





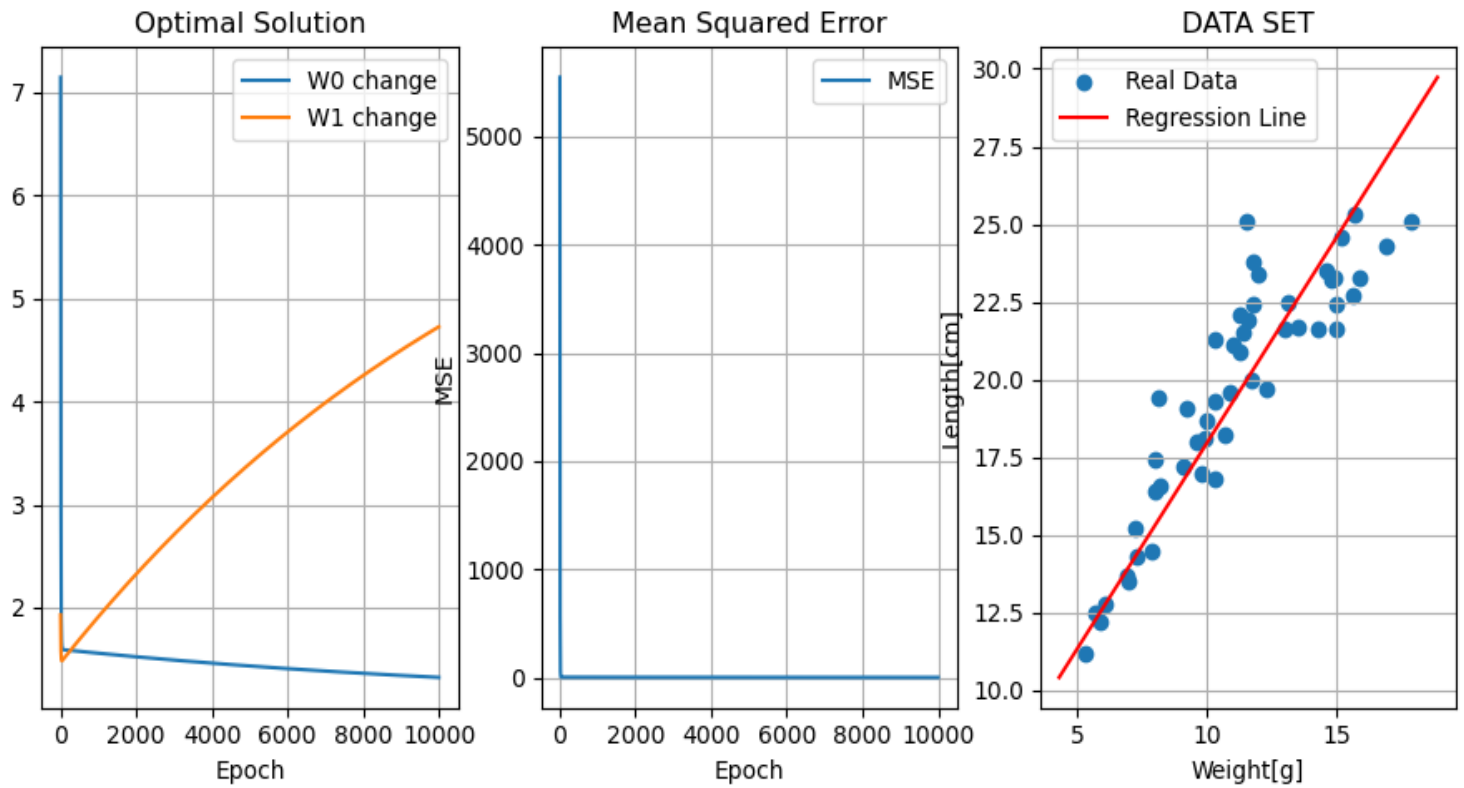


Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[2.006536, 2.0122518165120002, 2…
MSE_new	list	10000	[45.468399999999995, 35.35564108…
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[1.073328, 1.1368192373888, 1.19…
x	float64	1	1.3010534836206042
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	4.9971823770415735
yy	Array of float64	(1000,)	[10.59171236 10.61070774 10.6297… 29.5 ...

W0, w1, 반복횟수, Learning Rate를 설정해 주는데 이 값을 기준으로 변화를 주기로 하고 각 값은 순서대로 1, 2, 0.001, 10000이 된다.

결과를 살펴보니 mse는 수렵을 하긴 하지만 그 경사가 가파른 것 같고, 최종 그래프는 과제 1에서 구한 그래프보다 조금 더 위로 올라간 것을 알 수 있다.

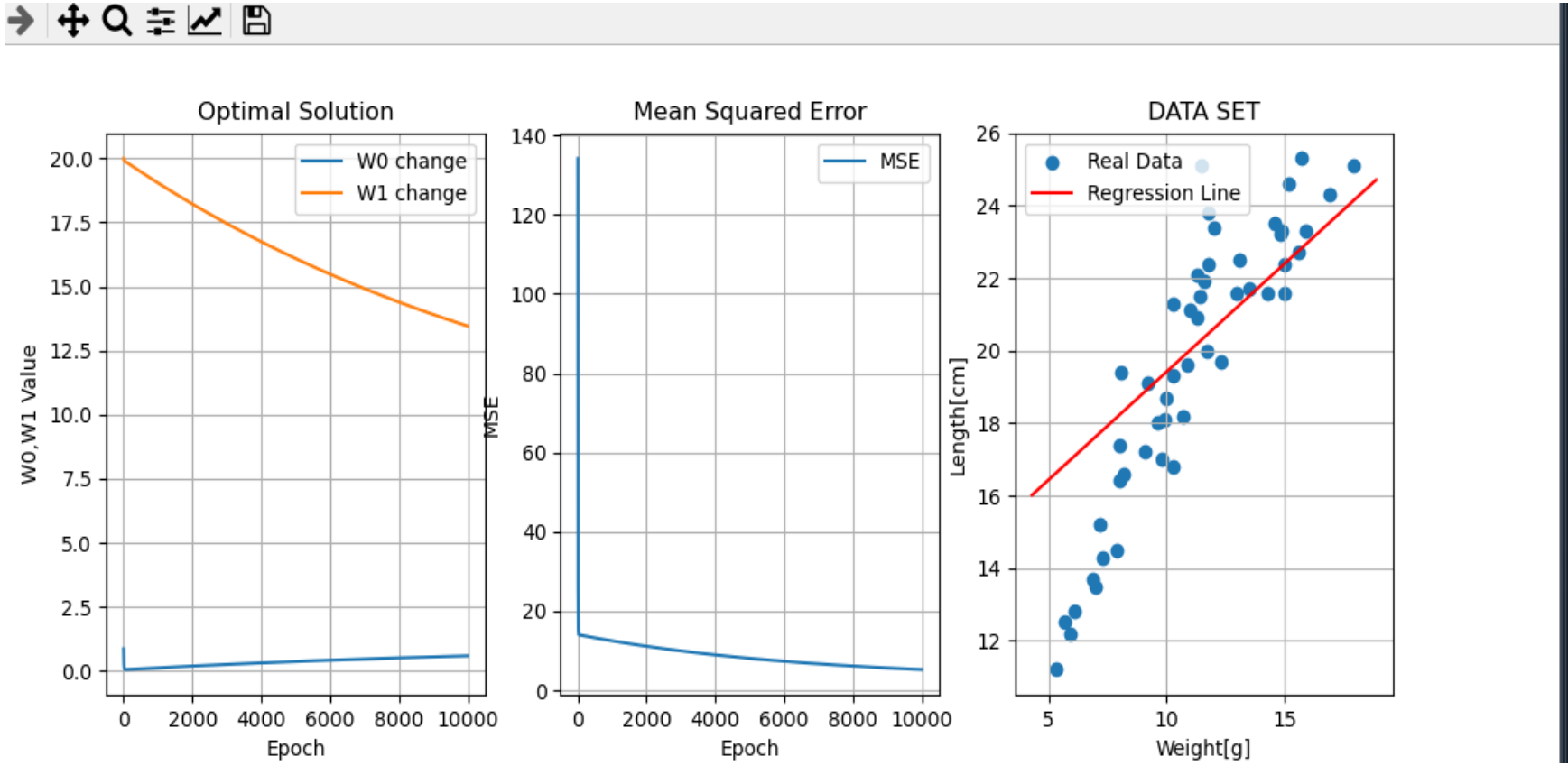
실습 과제 2 . W0변화



Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[1.928864, 1.8673281677567999, 1…
MSE_new	list	10000	[5543.638000000001, 4158.0741030…
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	8
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[7.1412192, 6.39758160233472, 5.…
x	float64	1	1.3239125591118577
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	4.723014502100208
yy	Array of float64	(1000,)	[10.41583851 10.43516763 10.4544… 29.7 ...

W0값을 8로 변화시켰더니 optimal solution 그래프에서 w1의 그래프까지 같이 변화했음을 알 수 있고 Mse값은 완전히 0으로 붙어버린 것을 알 수 있다.

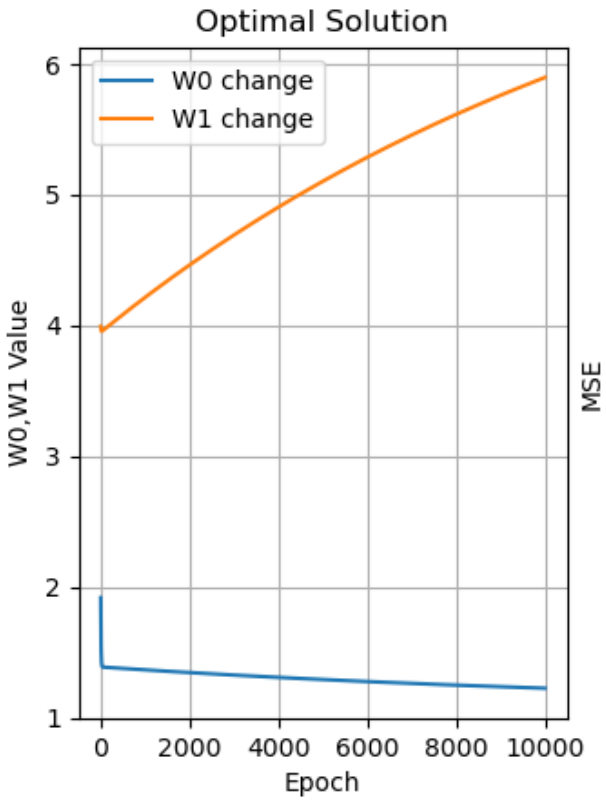
실습 과제 2 . W1변화

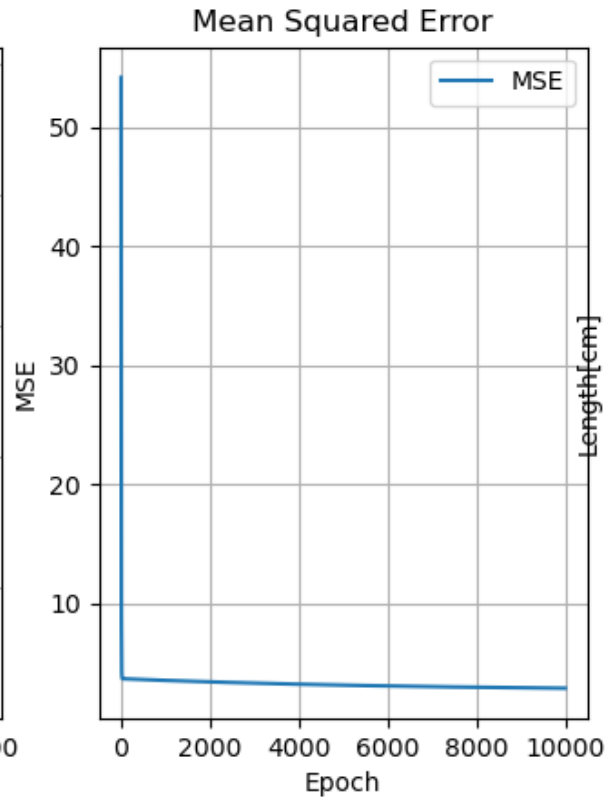


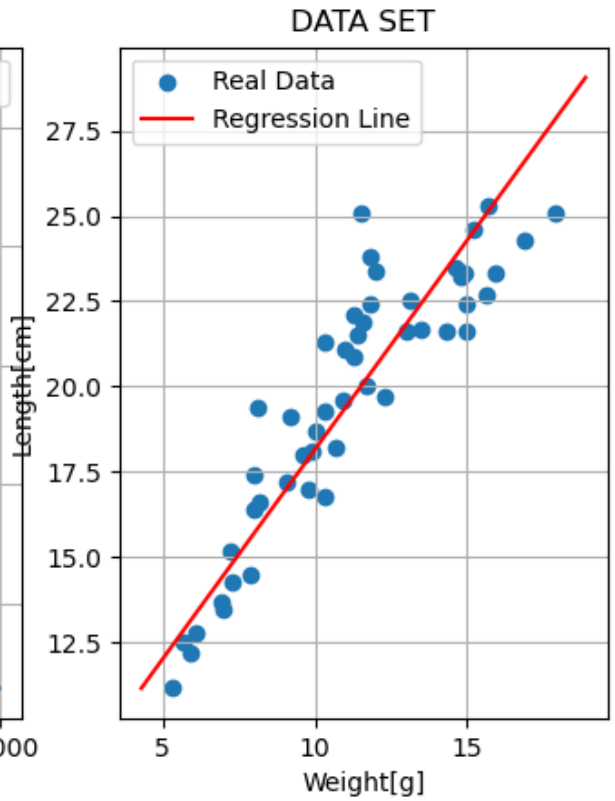
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[19.988536, 19.9784859984, 19.96
MSE_new	list	10000	[134.1724, 104.11538857048316, 8
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	20
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[0.8735999999999999, 0.764158426
x	float64	1	0.5960503766285369
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	13.452870086066962
yy	Array of float64	(1000,)	[16.01588671 16.02458904 16.0332 24.7 ...

W1값을 20으로 변화시켰더니 optimal solution 그래프에서 w0, w1의 그래프가 같이 변화했음을 알 수 있고 Mse값은 초기값보다 곡률이 더 커졌다. 또한 최종적인 그래프에서 값이 오차가 많은 그래프가 생기게 되었다.

실습 과제 2 . W0, W1변화



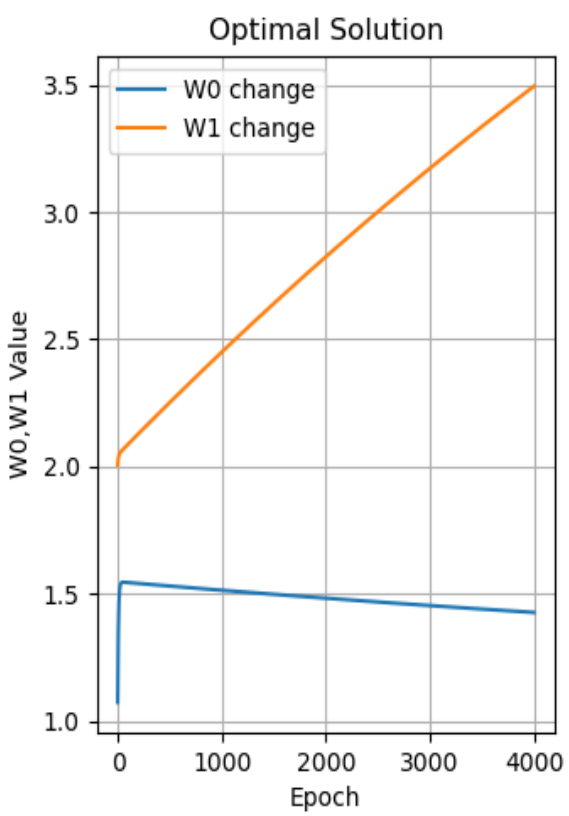


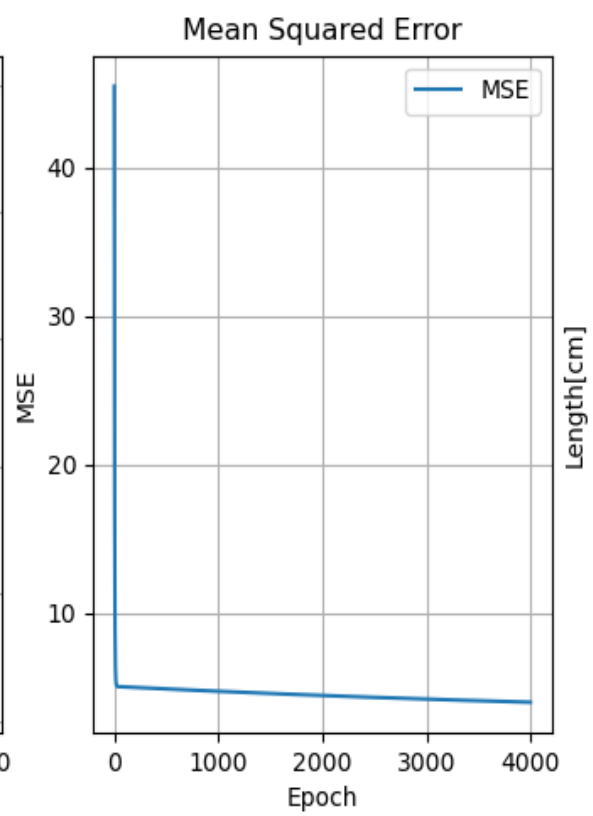


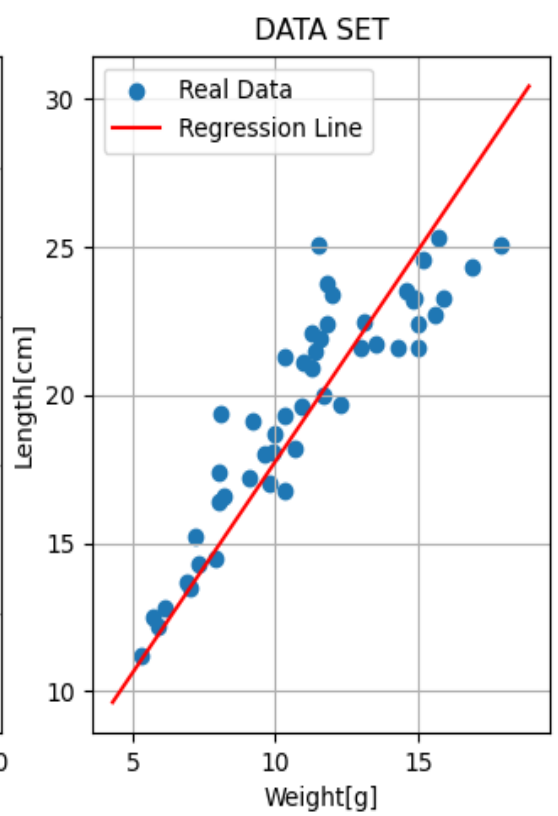
fold_dir	str	101	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[3.99344, 3.9877966805504, 3.982…
MSE_new	list	10000	[54.2108, 41.57721473081304, 32.…
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	2
w1	int	1	4
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[1.9179776, 1.84694996130816, 1.…
x	float64	1	1.2259853872789674
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	5.897536394322597
yy	Array of float64	(1000,)	[11.16927356 11.18717295 11.2050… 29.0 ...

W0, W1값을 2배로 변화시켰더니 optimal solution 그래프에서 w0, w1의 그래프가 같이 변화했음을 알 수 있고
W0은 그래프가 뒤집어진 모양으로 변화하였다 Mse값은 초기값과 비슷함을 알 수 있고,
 또한 최종적인 그래프도 비슷하게 나왔다.

실습 과제 2 . RP변화 작은 값



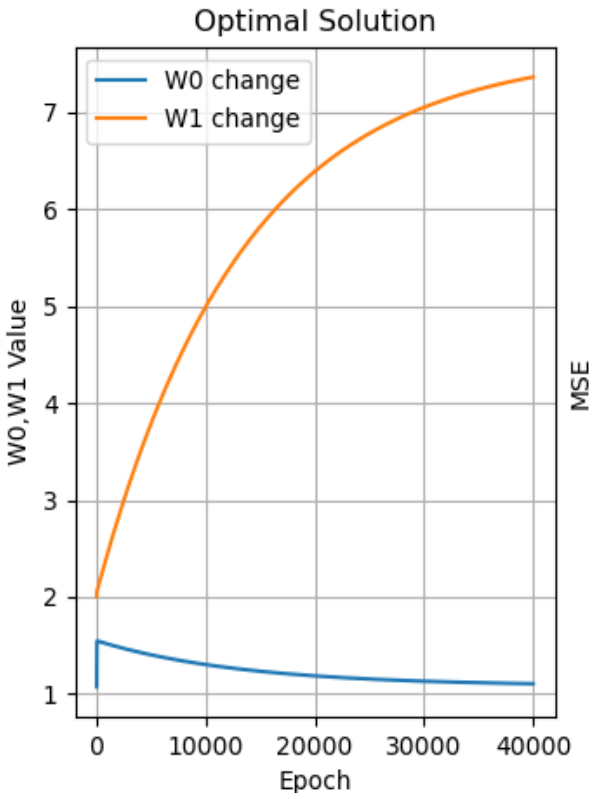


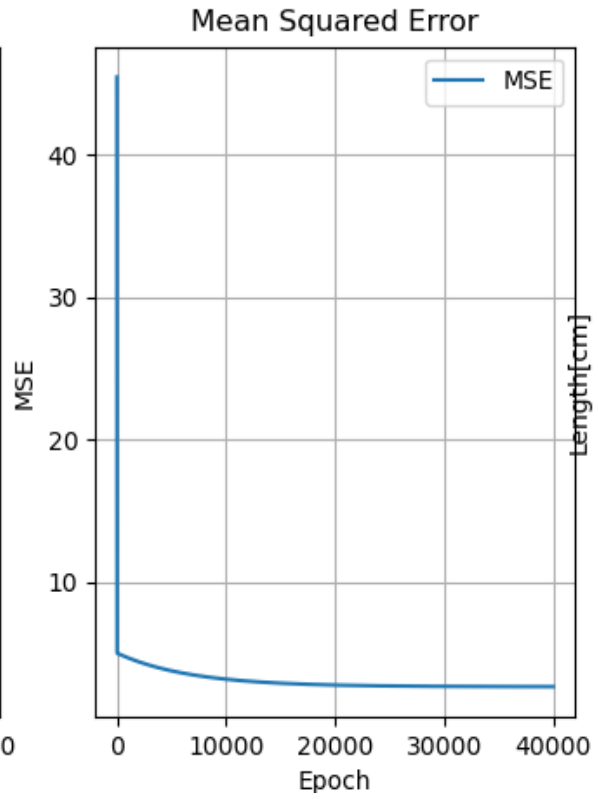


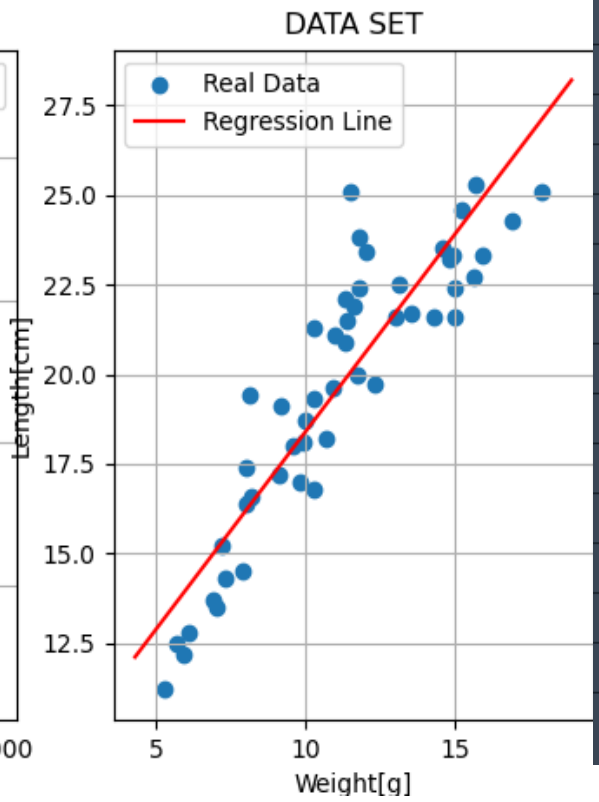
Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	4000	[2.006536, 2.0122518165120002, 2…
MSE_new	list	4000	[45.468399999999995, 35.35564108…
rp	int	1	4000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	4000	[1.073328, 1.1368192373888, 1.19…
x	float64	1	1.4263021525157051
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	3.4949710823046654
yy	Array of float64	(1000,)	[9.62807034 9.64889435 9.6697… 30.4 ...

RP값을 기존보다 작은 4000으로 반복했더니 W1그래프는 곡률이 거의 사라진 직선으로, W0그래프는 증가치가 더욱 높다가 꺾이는 그래프가 되었지만 mse값은 큰 변화가 없었고, 최종 그래프는 올라간 것을 알 수 있다.

실습 과제 2 . RP변화 큰 값



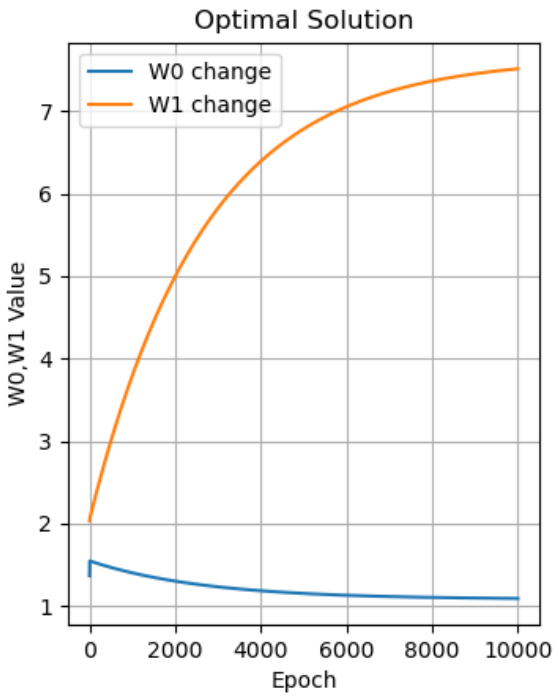


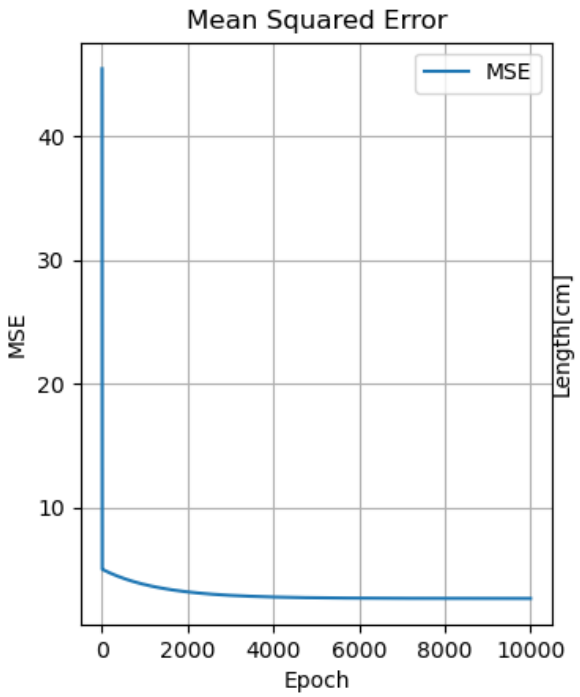


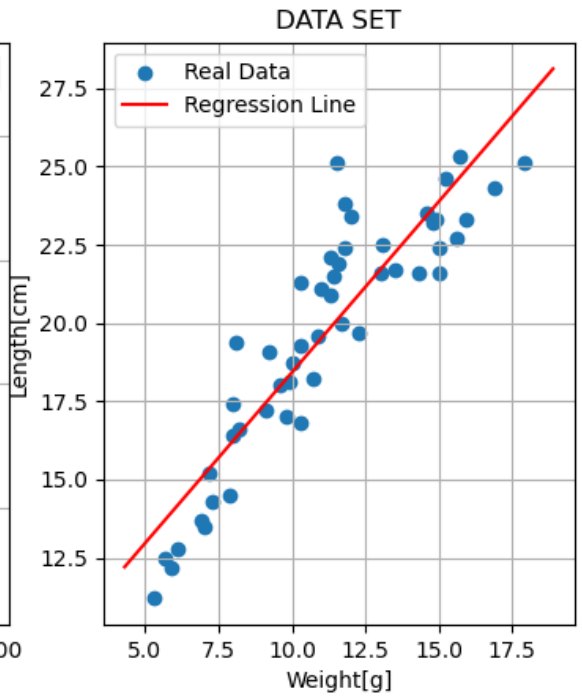
Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학
Learn_R	float	1	0.001
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	40000	[2.006536, 2.0122518165120002, 2
MSE_new	list	40000	[45.468399999999995, 35.35564108
rp	int	1	40000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	40000	[1.073328, 1.1368192373888, 1.19
x	float64	1	1.1035363814533754
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	7.366169004977868
yy	Array of float64	(1000,)	[12.11137545 12.12748708 12.1435 28.2 ...

반복수를 40000으로 변화했더니 w0,w1그래프의 곡률이 매우 커짐을 알 수 있고, mse의 그래프 또한 더욱 큰 곡률을 갖게 되었고, 최종 그래프는 보다 정확한 값이 나왔음을 알 수 있다.

실습 과제 2 . LR변화 작은 값



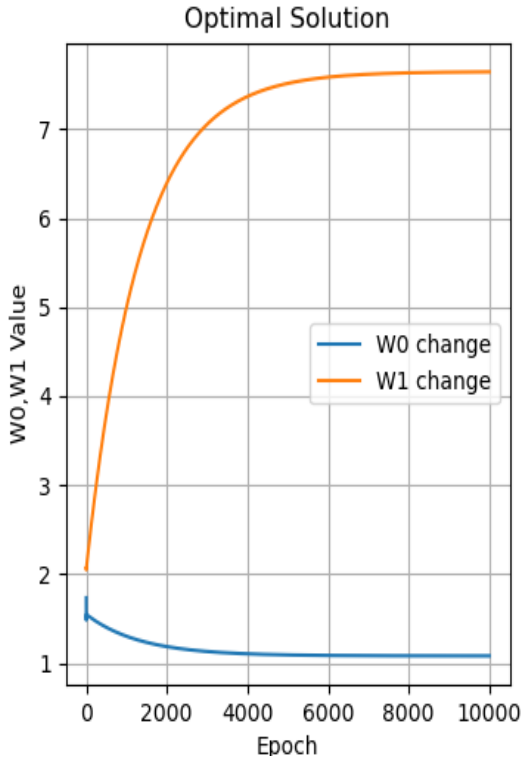


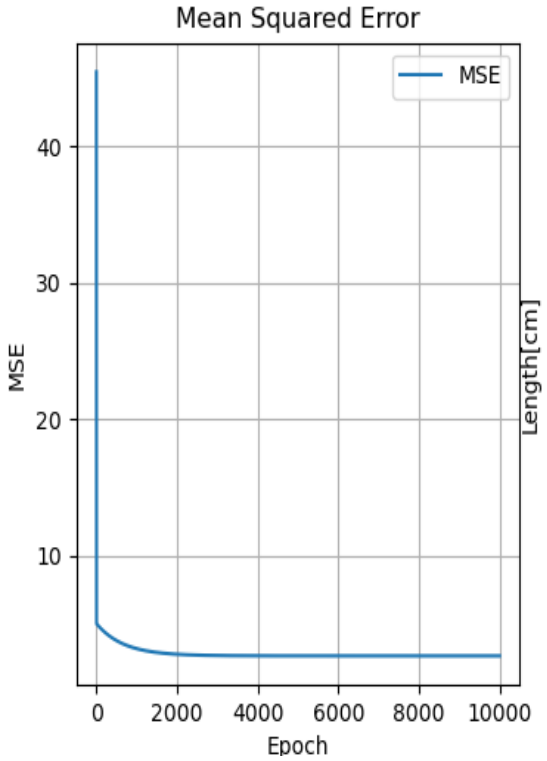


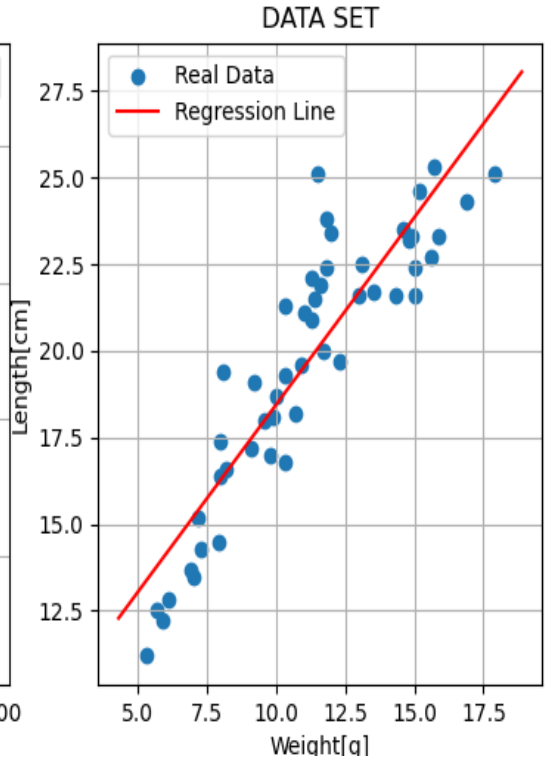
Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공
Learn_R	float	1	0.005
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[2.03268, 2.0448554128, 2.0502
MSE_new	list	10000	[45.468399999999995, 9.4380224
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[1.36664, 1.48736093472, 1.527
x	float64	1	1.0912095138073745
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	7.51401536510213
yy	Array of float64	(1000,)	[12.20621627 12.22214793 12.23 28.1 ...

LR값을 0.001의 5배를 한 0.005로 하였더니 w0,w1,mse의 곡률은 크게 나오고 최종 그래프는 기준의 그래프보다는 조금 더 정확함을 알 수 있었다.

실습 과제 2 . LR변화 큰 값



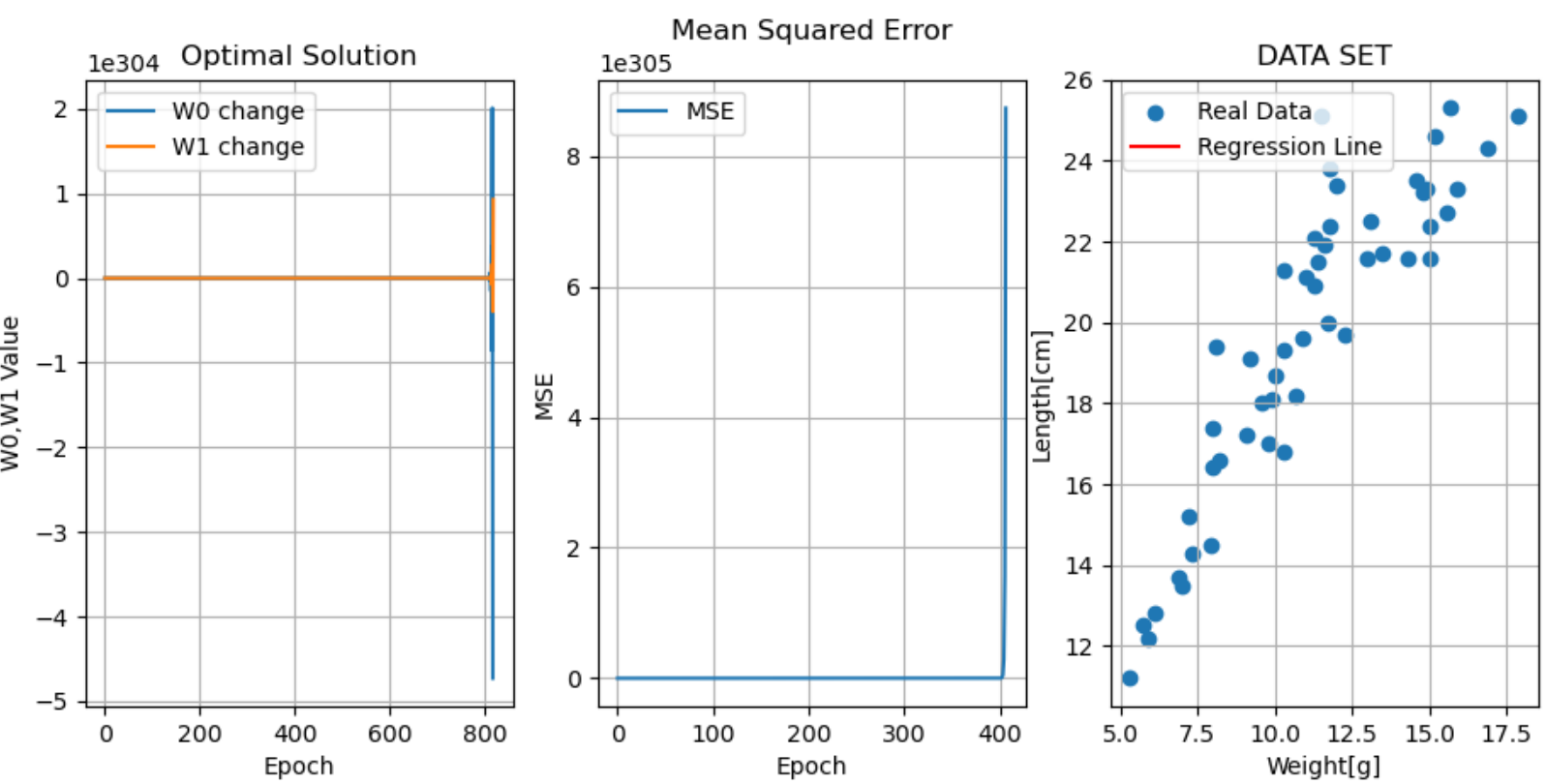




Name	Type	Size	Value
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.01
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	10000	[2.06536, 2.0487016512, 2.059993…
MSE_new	list	10000	[45.468399999999995, 9.741187203…
rp	int	1	10000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	1
w1	int	1	2
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	10000	[1.7332800000000002, 1.482883738…
x	float64	1	1.080417598593089
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	7.643451766061573
yy	Array of float64	(1000,)	[12.28924744 12.30502154 12.3207… 28.0 ...

LR을 크게 0.01로 변화했더니 W0은 완전히 모양이 바뀌었고, W1은 곡률이 커졌고, mse그래프는 모서리 부분의 곡률이 생겼지만 결과는 같다, 최종 그래프는 더욱 정확한 그래프가 나왔다.

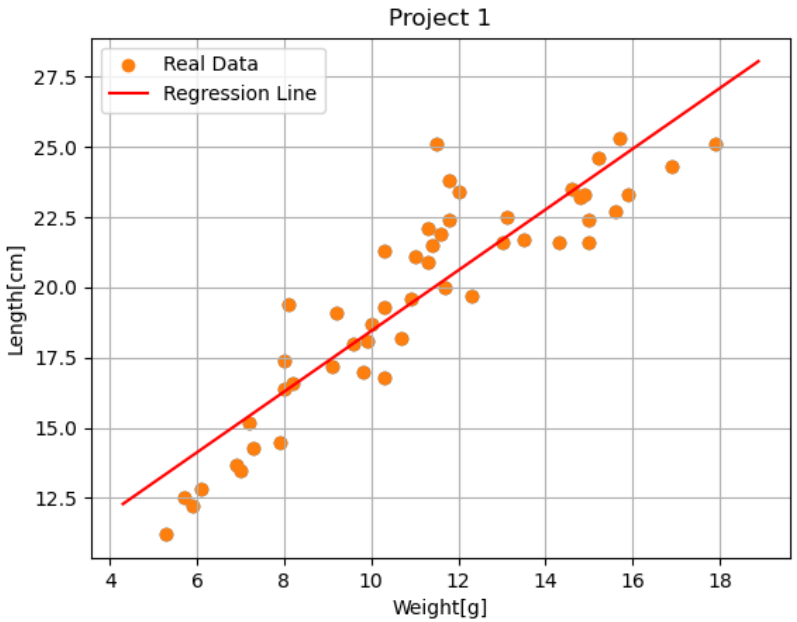
실습 과제 2 . 모두 변화

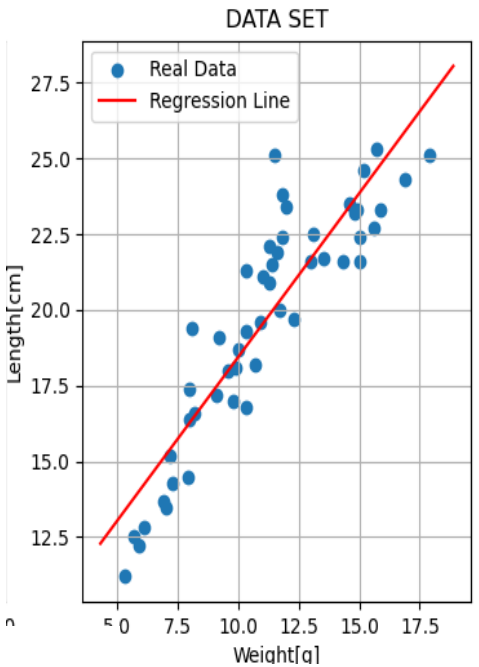


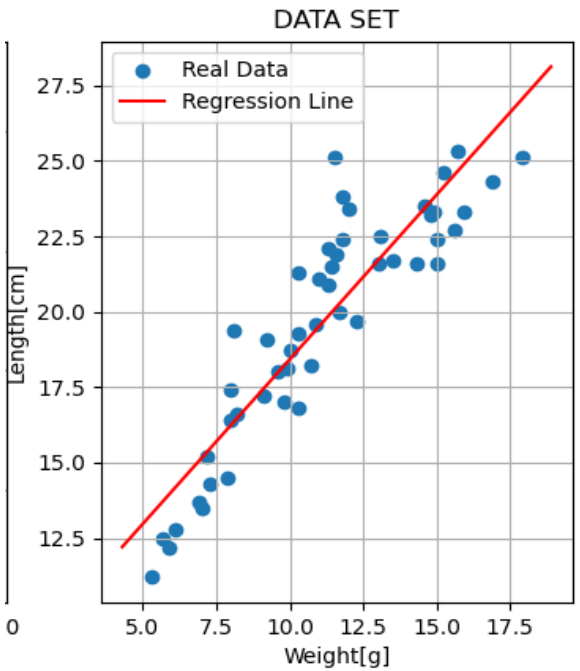
fold_dir	str	84	C:\Users\user\OneDrive - 한국공학…
Learn_R	float	1	0.02501
Len	Array of float64	(50,)	[23.4 21.6 11.2 ... 24.3 25.1 16.8]
Len_new	list	30000	[17.21568672, 23.696026253588087…
MSE_new	list	30000	[13195.202799999997, 73025.64553…
rp	int	1	30000
temp_data	Array of float64	(50, 2)	[[12. 23.4] [14.3 21.6]
w0	int	1	10
w1	int	1	20
Wei	Array of float64	(50,)	[12. 14.3 5.3 ... 16.9 11.5 10.3]
Wei_new	list	30000	[-23.13388825599999, 54.85041014…
x	float64	1	nan
x_values	Array of float64	(1000,)	[4.3 4.3146 4.3292 ... 18.8562 18.8708 18.8854]
y	float64	1	nan
yy	Array of float64	(1000,)	[nan nan nan ... nan nan nan]

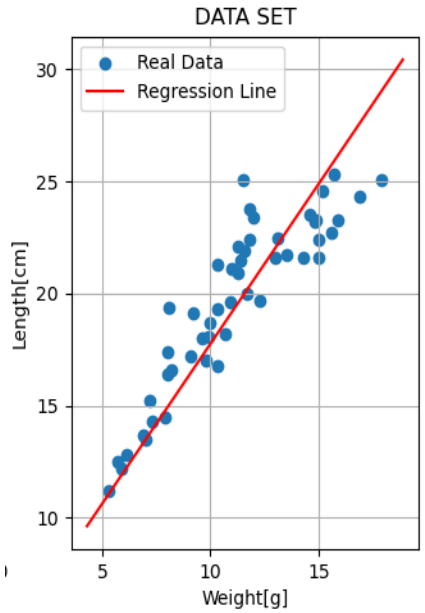
모든 값을 변화시켰는데 순서대로 10,20,0.02502,30000로 변화시켰고,
모든 그래프가 이상한 값을 나타내는 그래프로 변화하였다.

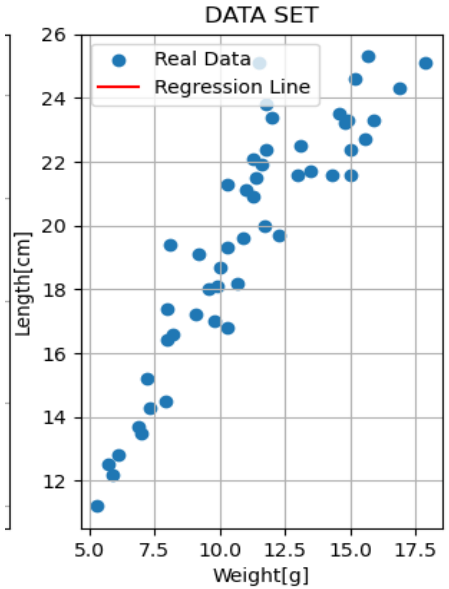
실습 과제 2. 실습과제 1과 비교











가장 비슷하게 나온 2개와 가장 다르게 나온 2개를 추려봤을 때 위에 있는 2개의 그래프는 LR값을 크고, 작게 변화시킨 그래프이고 왼쪽의 그래프 2개는 RP의 값을 아주 작게 한 것과, 모든 값을 무작위로 바꿨을 때의 그래프로 비교할 수 있다.