Machine learning

-5주차 실습 과제-

chap.1 선형회귀

전자공학부

2022144007

김의진

#과제 2

1) 앞에서 구한 경사하강법을 사용자 지정 함수로 나타내라

경사하강법을 쓰는 이유

->analystic solution은 특징 개수가 많아질 수록 구해야 할 weight 개수가 많아지기 때문에 힘듦

-

$$w_0^* = \frac{\frac{1}{N}\sum_{n=0}^{N-1} y_n \left(x_n - \frac{1}{N}\sum_{i=0}^{N-1} x_i \right)}{\frac{1}{N}\sum_{n=0}^{N-1} x_n^2 - \left(\frac{1}{N}\sum_{n=0}^{N-1} x_n \right)^2}$$

$$w_1^* = \frac{1}{N}\sum_{n=0}^{N-1} \left(y_n - w_0^* x_n \right)$$

이런식으로 특징 개수 +1개 만큼 구해야함

#과제 2

1) 앞에서 구한 경사하강법을 사용자 지정 함수로 나타내라

경사하강법을 이용한 수치적 접근으로 보다 효율적으로 analystic solution의 근사값을 구할 수 있음

Learning Rate

New Old Old

$$w_0[t+1] = w_0[t] - \alpha \frac{\partial}{\partial w_0} \epsilon_{MSE}(w_0, w_1)$$

$$w_1[t+1] = w_1[t] - \alpha \frac{\partial}{\partial w_1} \epsilon_{MSE}(w_1, w_2)$$

Analystic solution 꼴보다 간단함!

MSE를 편미분한 것에 –방향으로 learning rate 값을 곱한 만큼 기존의 weight에서 빼줘서 업데이트

#과제 2

1) 앞에서 구한 경사하강법을 사용자 지정 함수로 나타내라

$$\frac{\partial}{\partial w_0}\epsilon_{MSE}(w_0, w_1) = \frac{1}{N}\sum_{n=0}^{N-1}(\hat{y}_n - y_n)\,x$$

$$\frac{\partial}{\partial w_1}\epsilon_{MSE}(w_0, w_1) = \frac{1}{N}\sum_{n=0}^{N-1}(\hat{y}_n - y_n)$$

Learning Rate

New Old Old

$$w_0[t+1] = w_0[t] - \alpha\frac{\partial}{\partial w_0}\epsilon_{MSE}(w_0, w_1)$$

$$w_1[t+1] = w_1[t] - \alpha\frac{\partial}{\partial w_1}\epsilon_{MSE}(w_1, w_2)$$

경사하강법 구현하기

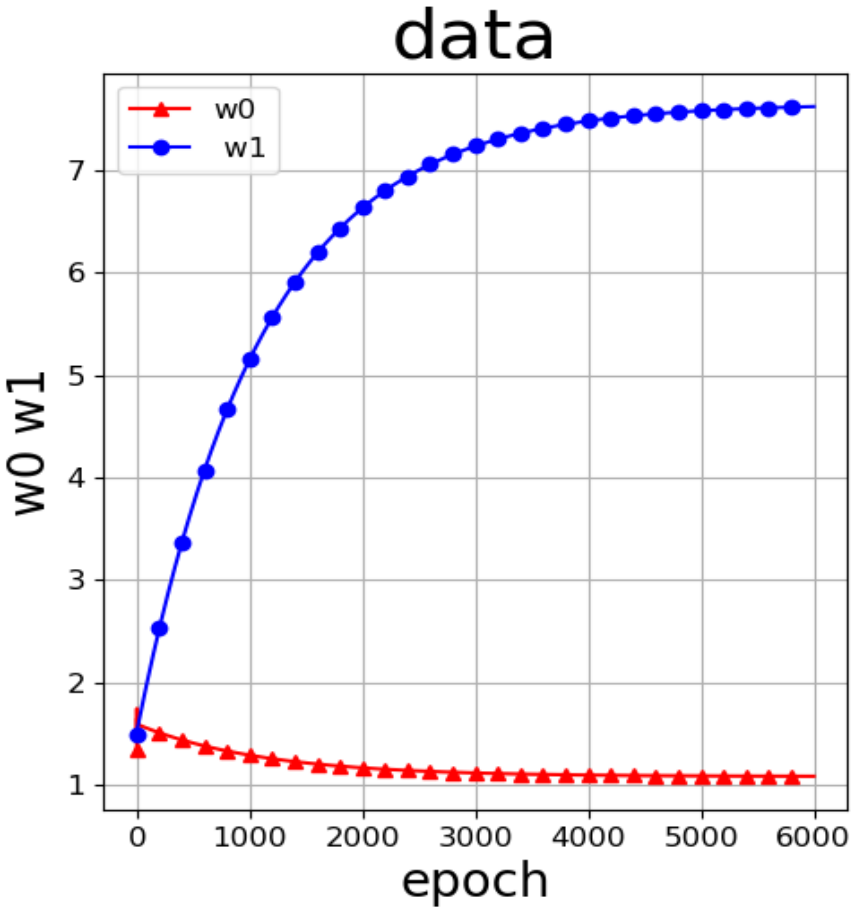
1. 옆의 식을 토대로 weight update

2. Stop 조건 대신 반복횟수 지정
→weight가 거의 변하지 않을 때까지

3. Hyper parameter
→learning rate, epoch, etc…

#과제 2

2) 구현한 경사하강법을 이용해 optimal solution을 구하고 학습 진행에 따른 w, MSE그래프를 그려라



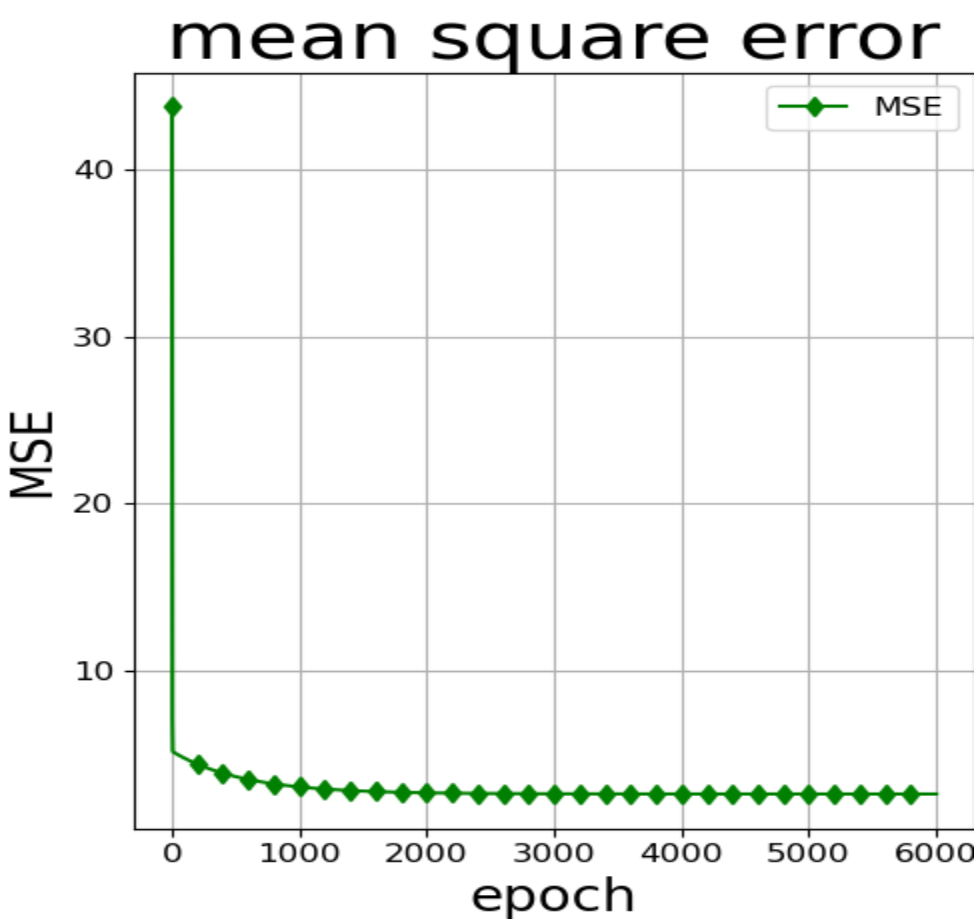
0	1.03114	1.78752
5989	1.08239	7.61985
5990	1.08238	7.61988
5991	1.08238	7.6199
5992	1.08238	7.61993
5993	1.08238	7.61995
5994	1.08238	7.61997
5995	1.08237	7.62
5996	1.08237	7.62002
5997	1.08237	7.62005
5998	1.08237	7.62007
5999	1.08237	7.62009

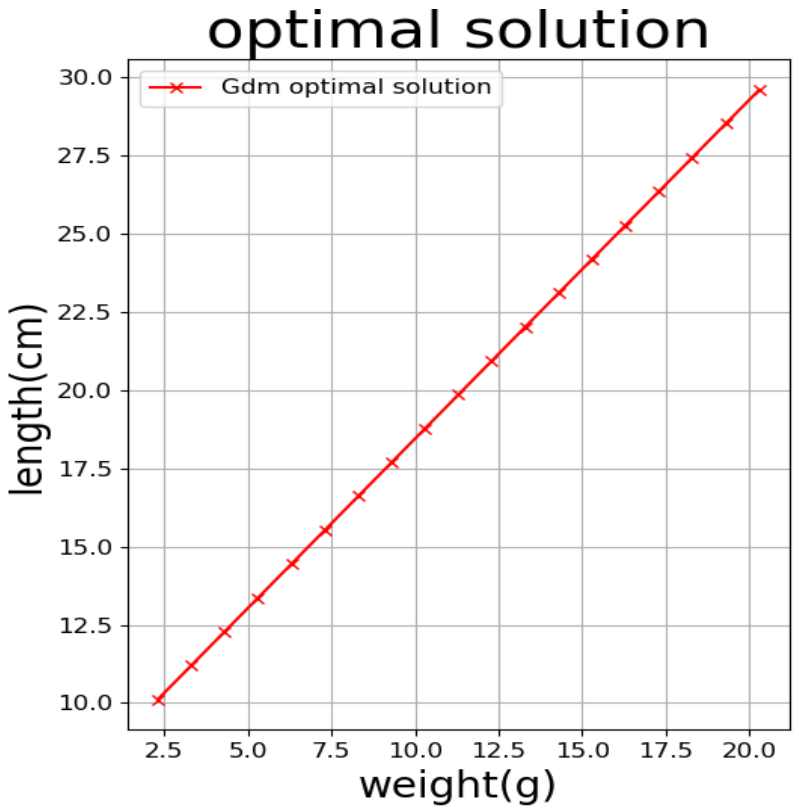
1. 초기값은 w0, w1가 각각 1.08239, 1.78752을 갖는다

2. 6000번의 업데이트를 통해 w0, w1가 1.08237과 7.620으로 수렴함을 알 수 있음

#과제 2

2) 구현한 경사하강법을 이용해 optimal solution을 구하고 학습 진행에 따른 w, MSE그래프를 그려라



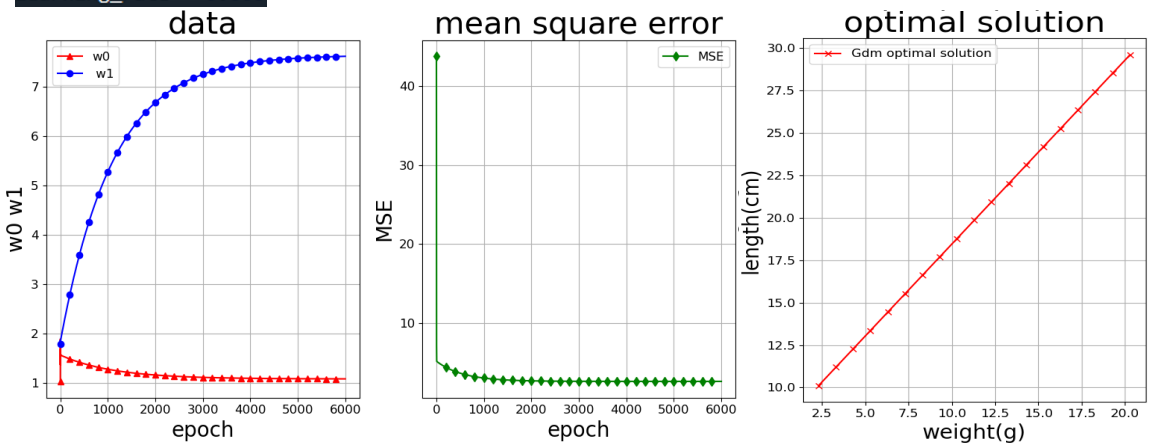



```
In [33]: runfile('C:/Users/kim07/Desktop/
Machinlearning_Workplace/
Machine_learning_practice_homework_5week.py', wdir='C:/
Users/kim07/Desktop/Machinlearning_Workplace')
y_opt = 1.0823631638972955 * x + 7.62011698573688
```

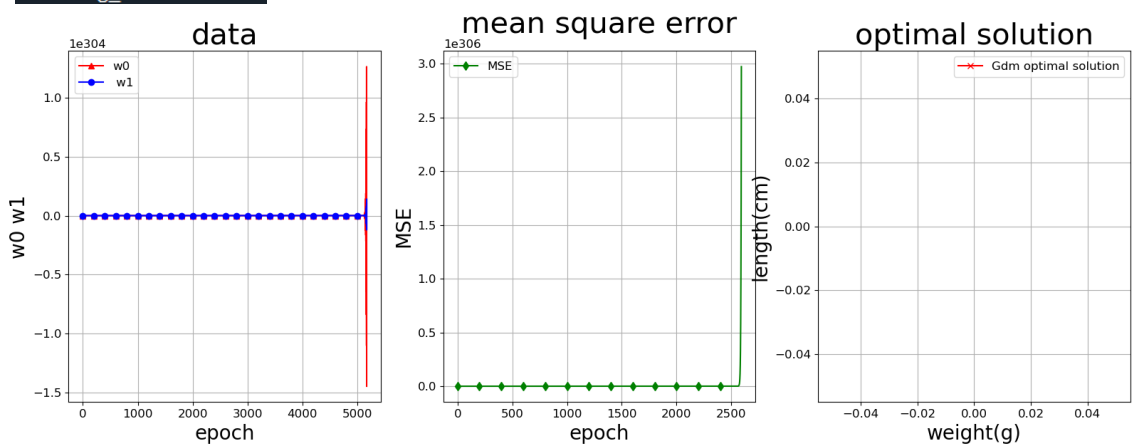
#과제 2

2) 구현한 경사하강법을 이용해 optimal solution을 구하고 학습 진행에 따른 w, MSE그래프를 그려라 (learning rate control)

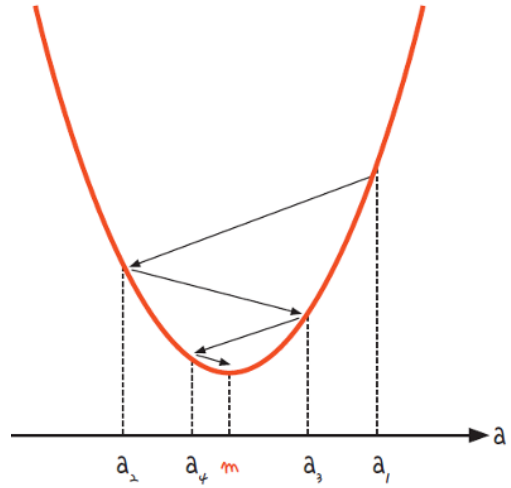
learning_rate = 0.006



learning_rate = 0.008



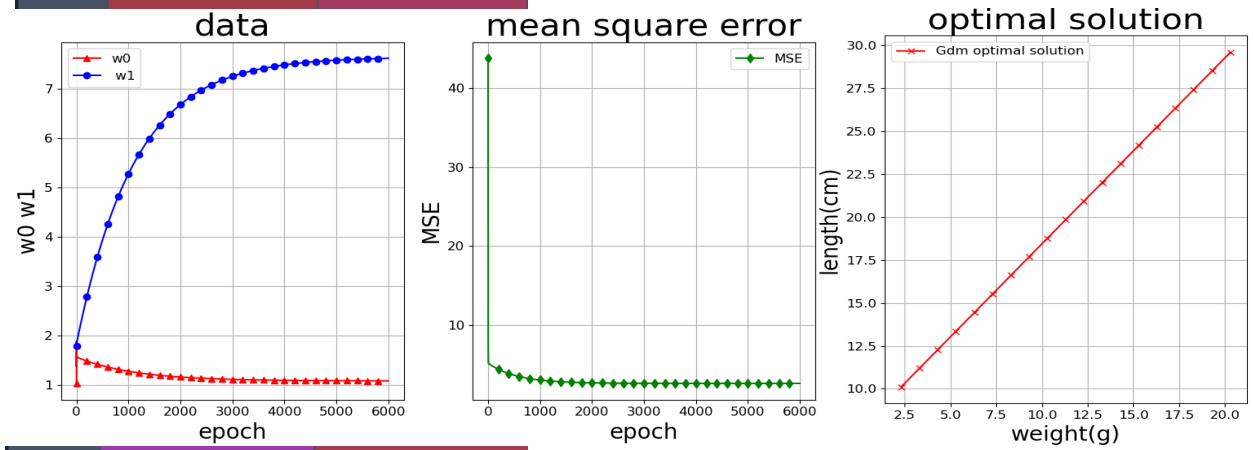
1. Learning rate크기: 0.006 → 0.008
2. w0, w1, MSE: 수렴 → 발산
3. Optimal solution → weight들이 발산하므로 그래프를 구할 수가 없게 됨



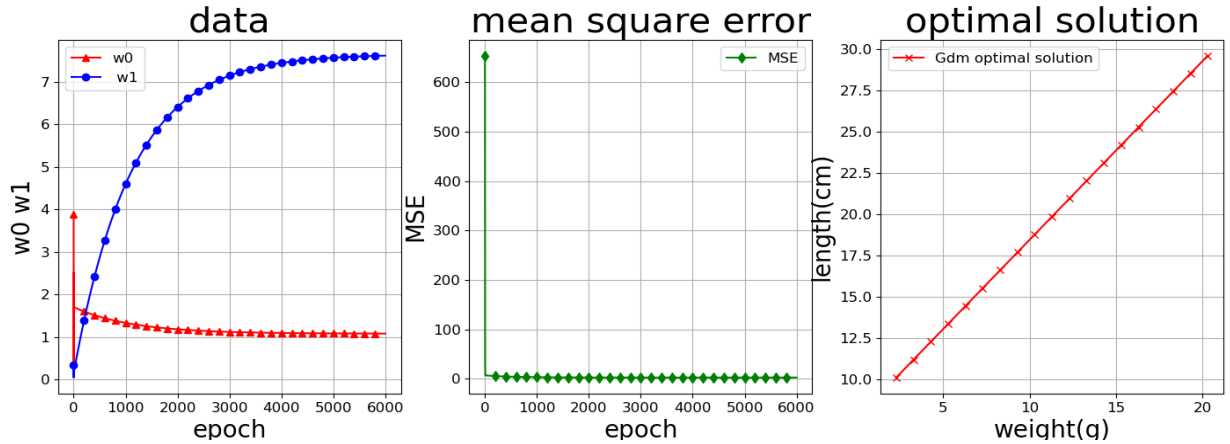
#과제 2

2) 구현한 경사하강법을 이용해 optimal solution을 구하고 학습 진행에 따른 w, MSE그래프를 그려라 (초기값 control)

0	1.03114	1.78752



0	3.89379	0.337461

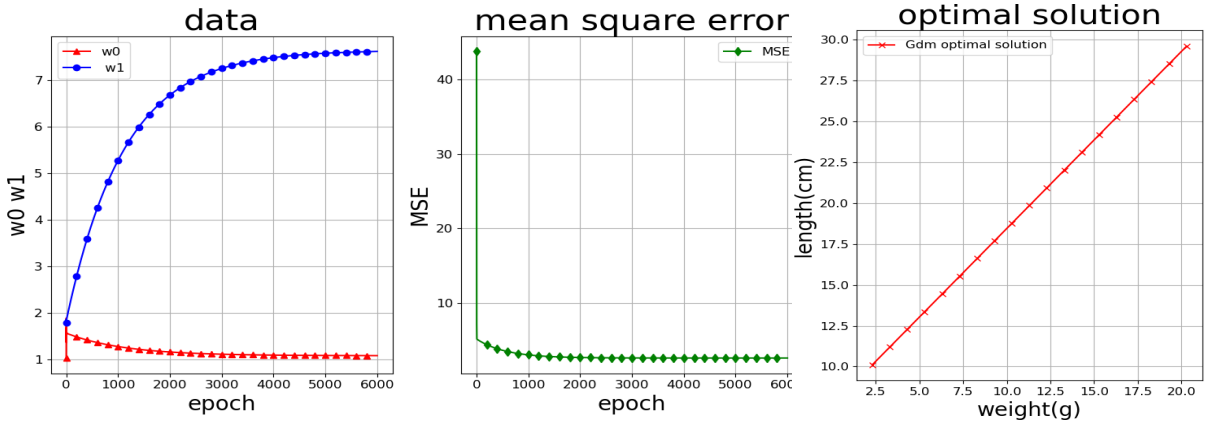


반복횟수에 따라 학습 진행률에 차이가 있을 뿐이지 똑같이 수렴함

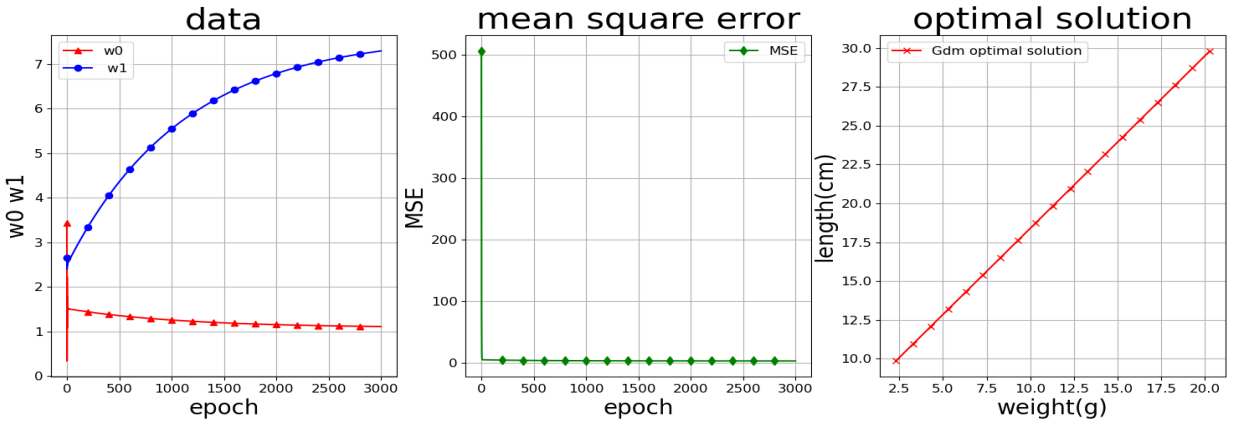
#과제 2

2) 구현한 경사하강법을 이용해 optimal solution을 구하고 학습 진행에 따른 w, MSE그래프를 그려라 (epoch control)

epoch = 6000



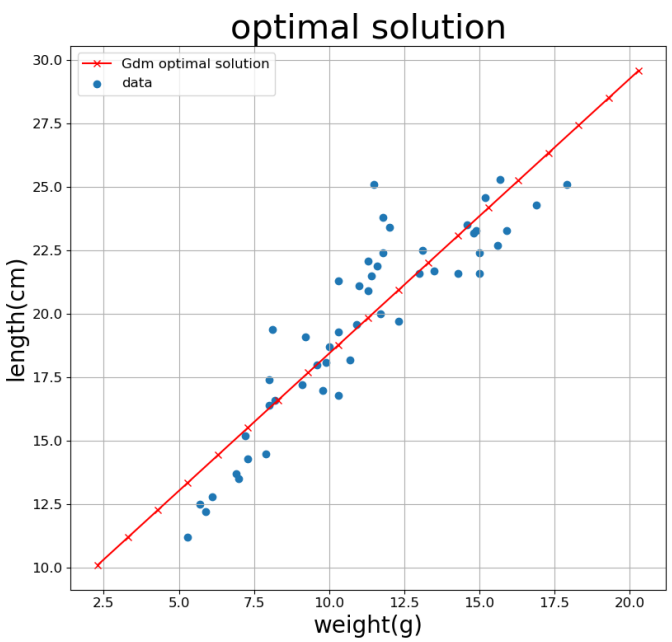
epoch = 3000

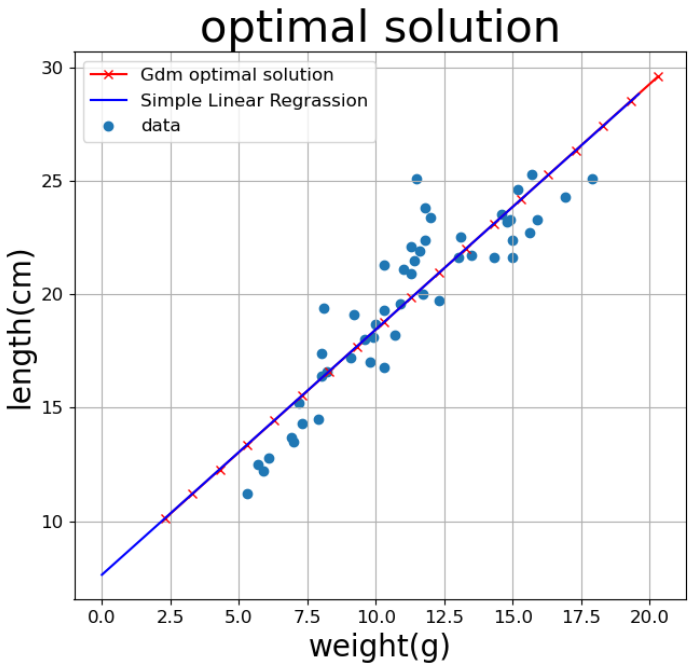


반복횟수가 충분하지 않아 w1이 완벽히 수렴하는 모습을 볼 수 없음

#과제2

3) 선형 회귀 모델로 구한 Optimal Solution과 데이터 set을 하나의 그래프에 표시하고, 실습과제 1-2와 비교하라

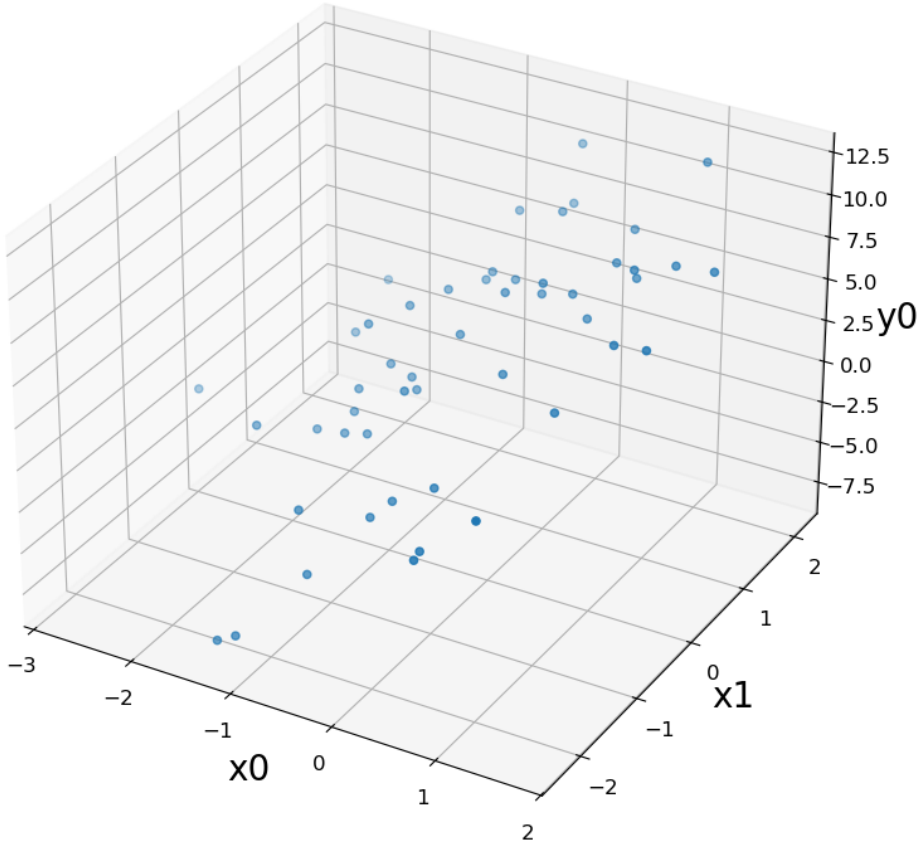




Weight, MSE값들이 거의 같아 solution도 비슷한 그래프를 띈다.

#과제3

1) x축은 x0, y축은 x1, z축은 y를 나타내는 3차원평면에 각 데이터의 위치를 점으로 나타내라



50 by 2의 데이터 set

	0	1
0	-0.121163	0.497399
1	0.223035	-0.154727
2	-0.419295	0.241948
3	0.346946	-1.78832
4	-0.790967	0.686775
5	1.01824	0.699743
6	-0.256221	0.925684
7	-1.46263	-0.438142
8	-0.90219	-0.528037
9	-0.0954984	1.47434
10	-0.236629	-1.2412

이 데이터의 특징이 x0, x1 두개이므로 과제 2와 달리 3차원으로 확장됨

➔ 특징 개수가 늘어나면 차원이 확장됨을 알 수 있음

특징이 많아질 수록 Analystic solution을 쓰기 어려운 이유

#과제3

2) 주어진 x 데이터에 dummy데이터 추가해 행렬을 생성하고, 초기 가중치를 이용하여 예측한 y_hat을 3차원 축에 평면으로 나타내라

dummy data 추가된 x data set

	0	1	2
0	-0.121163	0.497399	1
1	0.223035	-0.154727	1
2	-0.419295	0.241948	1
3	0.346946	-1.78832	1
4	-0.790967	0.686775	1
5	1.01824	0.699743	1
6	-0.256221	0.925684	1
7	-1.46263	-0.438142	1
8	-0.90219	-0.528037	1
9	-0.0954984	1.47434	1
10	-0.236629	-1.2412	1

1로 구성된
dummy data set

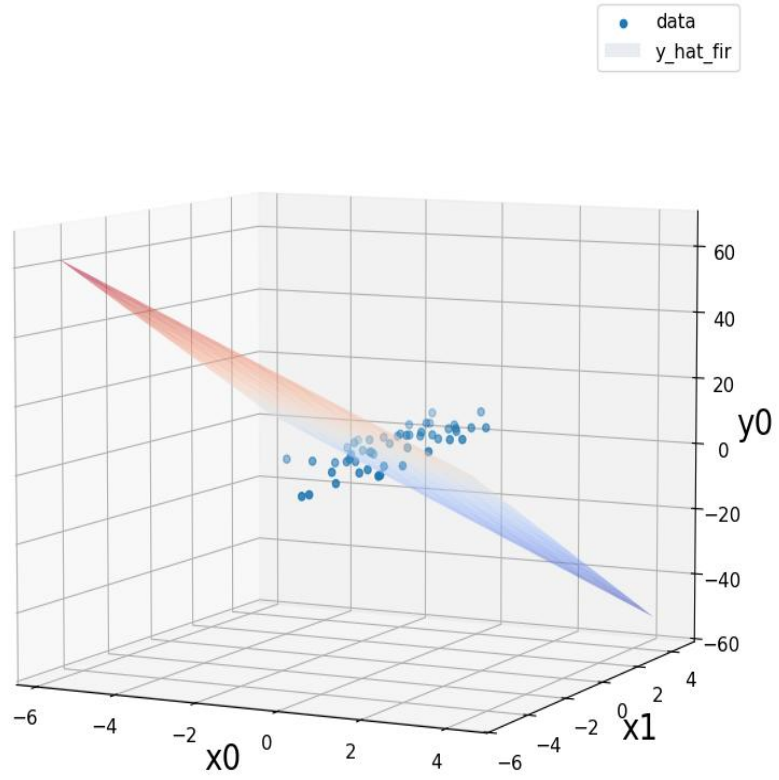
x0, x1데이터에 50size의 1 dummy 행렬이 추가됨

*dummy data 추가한 이유: weight data set은 바이어스가 추가로 있기 때문에 행렬 계산을 위해 사이즈를 맞춰줌

#과제3

2) 주어진 x 데이터에 dummy데이터 추가해 행렬을 생성하고, 초기 가중치를 이용하여 예측한 y_hat을 3차원 축에 평면으로 나타내라

	w0	w1	w2
0	-6.67198	-0.386733	-5.31686



초기 가중치로 만들어진 optimal solution

➔ y = - 6.67198x_0 - 0.386733x_1 - 5.131686

Data와 거의 일치하지 않는 모습을 볼 수 있다.

#과제3

3) 실습#2에서 구현한 경사하강법 함수를 다차원 데이터 입력이 가능하도록 변경해 Optimal Solution (weights, $\boldsymbol{w}$)을 구하고, 학습 진행(epoch)에 따른 w와 MSE에 대한그래프를 그려라. (본인은 과제 2번도 다차원 데이터 입력해 풀었음)

```python
#과제 2 함수
def GDM(epoch, learning_rate):

    w_hist = []
    MSE_hist = []

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

        y_hat = np.dot(x_matrix, w)
        error = y_hat - y_vector
        error = error.reshape(-1, 1)
        MSE = np.mean(error**2)

        w_hist.append(w)
        MSE_hist.append(MSE)

        w_dif = sum(2 * error * x_matrix)/len(y_hat)
        w = w - learning_rate*w_dif

    return w, w_hist, MSE_hist
```

```python
def GDM_work3(epoch_work3, learning_rate_work3):

    w_hist = []
    MSE_hist = []

    #epoch만큼 반복해 weight 업데이트
    for i in range(epoch_work3):
        if i == 0:
            w = np.random.rand(3) * -10

        y_hat = np.dot(x_work3_matrix, w)
        error = y_hat - y0_vector
        error = error.reshape(-1, 1)
        MSE = np.mean(error**2)

        w_hist.append(w)
        MSE_hist.append(MSE)

        w_dif = sum(2 * error * x_work3_matrix)/len(y_hat)
        w = w - learning_rate_work3*w_dif

    return w, w_hist, MSE_hist
```

다차원 데이터 입력가능하도록 할 때 유의할 점

1) 구해야할 값이 스칼라인지 벡터인지 행렬인지 알기
2) 행렬 계산할 때 사이즈 크기 잘 맞춰주기

과제 3번 GDM_work3함수는 과제 2번과 받아오는 데이터의 특징 개수만 다르고 알고리즘은 같음

다차원으로 갈 수록 행렬 계산 이용하는 것이 효율적임!

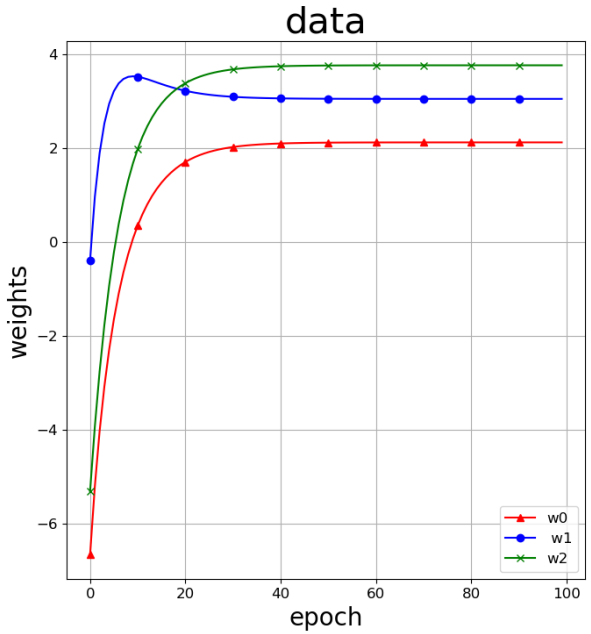
#과제3

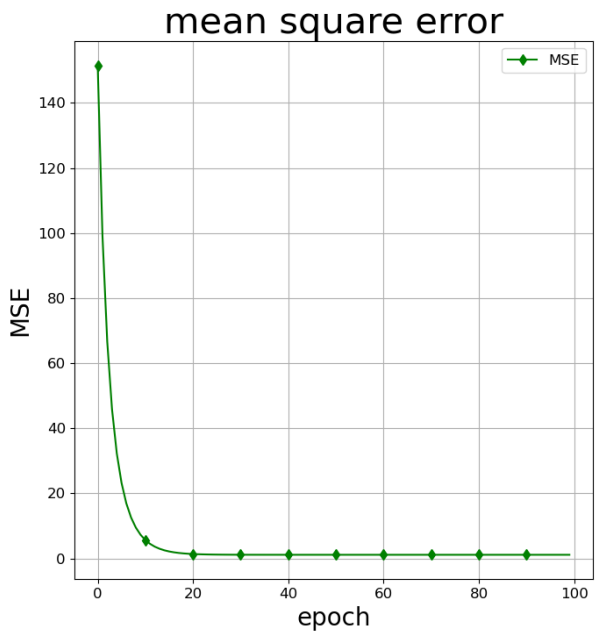
3) 실습#2에서 구현한 경사하강법 함수를 다차원 데이터 입력이 가능하도록 변경해 Optimal Solution (weights, $\boldsymbol{w}$)을 구하고, 학습 진행(epoch)에 따른 w와 MSE에 대한그래프를 그려라.

Optimal Solution

```
In [3]: print("y =", w0_work3_opt,"* x0 +", w1_work3_opt, "* x1 +", w2_work3_opt)
y = 2.128490022755701 * x0 + 3.0568565484370955 * x1 + 3.7735868582712
```

```
epoch_work3 = 100
learning_rate_work3 = 0.1
```





	w0	w1	w2
0	-6.67198	-0.386733	-5.31686
89	2.12847	3.05686	3.77357
90	2.12848	3.05686	3.77358
91	2.12848	3.05686	3.77358
92	2.12848	3.05686	3.77358
93	2.12848	3.05686	3.77358
94	2.12848	3.05686	3.77358
95	2.12849	3.05686	3.77358
96	2.12849	3.05686	3.77358
97	2.12849	3.05686	3.77359
98	2.12849	3.05686	3.77359
99	2.12849	3.05686	3.77359

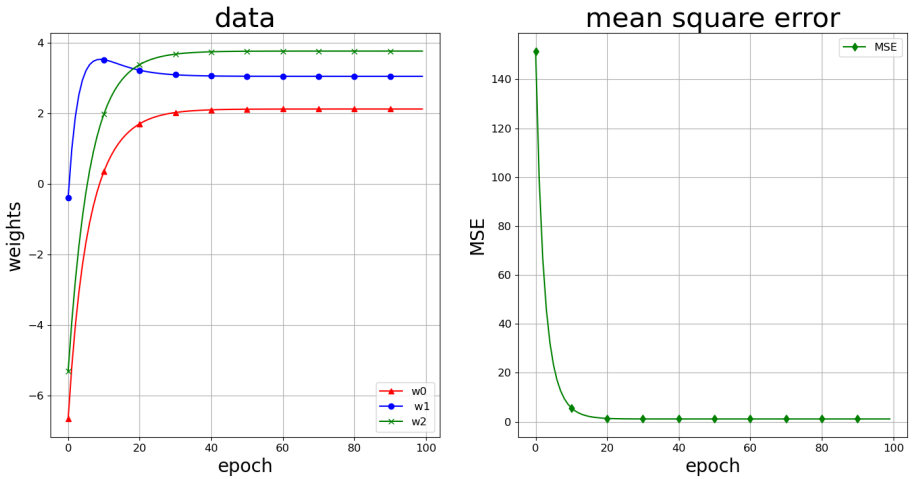
	MSE
0	151.565
89	1.09774
90	1.09774
91	1.09774
92	1.09774
93	1.09774
94	1.09774
95	1.09774
96	1.09774
97	1.09774
98	1.09774
99	1.09774

수렴

#과제3

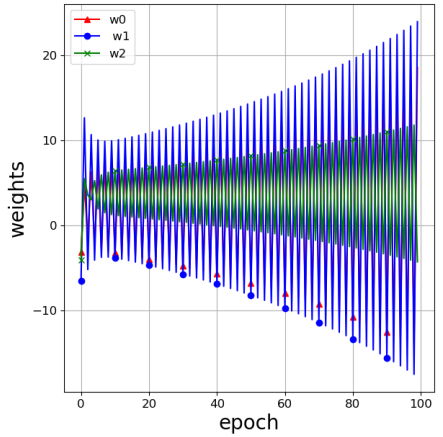
3) 실습#2에서 구현한 경사하강법 함수를 다차원 데이터 입력이 가능하도록 변경해 Optimal Solution (weights, $\boldsymbol{w}$)을 구하고, 학습 진행(epoch)에 따른 w와 MSE에 대한그래프를 그려라. (learning rate control)

```
learning_rate_work3 = 0.1
```

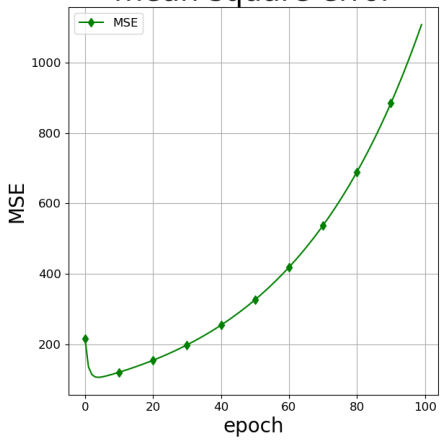



```
learning_rate_work3 = 0.7
```

data

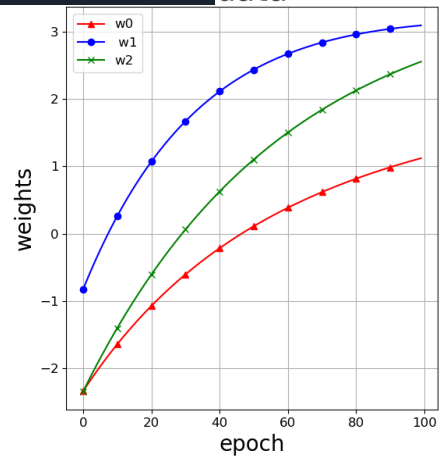


mean square error

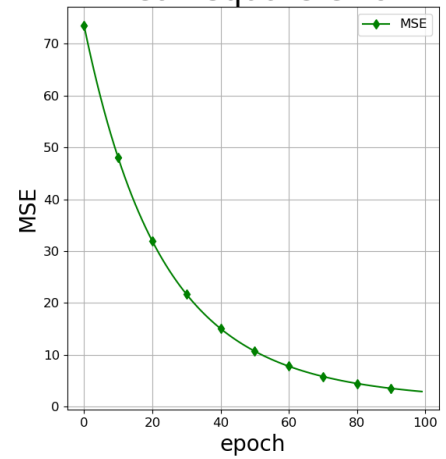



```
learning_rate_work3 = 0.01
```

data



mean square error



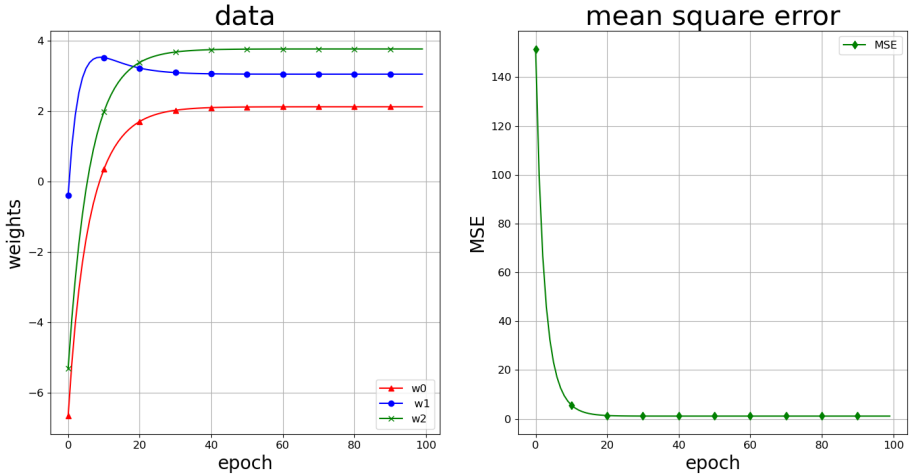
1) 0.7로 커졌을 땐 weight와 MSE가 발산해버림 (이 데이터 기준 0.7 이후로 발산함)

2) 0.01로 작아졌을 땐 학습이 끝나지 않았음

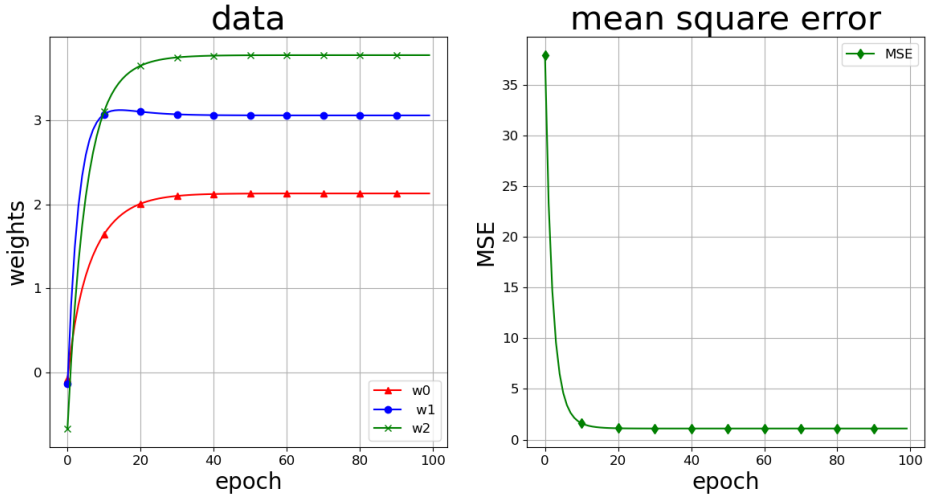
#과제3

3) 실습#2에서 구현한 경사하강법 함수를 다차원 데이터 입력이 가능하도록 변경해 Optimal Solution (weights, $\boldsymbol{w}$)을 구하고, 학습 진행(epoch)에 따른 w와 MSE에 대한그래프를 그려라. (초기값 control)

0	-6.67198	-0.386733	-5.31686



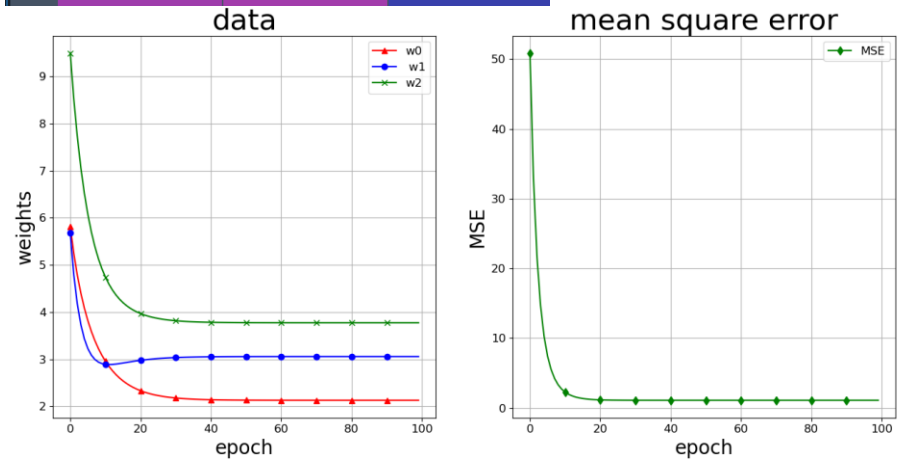
0	-0.0784266	-0.13984	-0.667847



1) 초기값 -1~0사이에 있도록 설정
2) 초기값 0~10사이에 있도록 설정

두 경우 다 반복 횟수가 충분해 시작점은 다르더라도 같은 값으로 수렴함

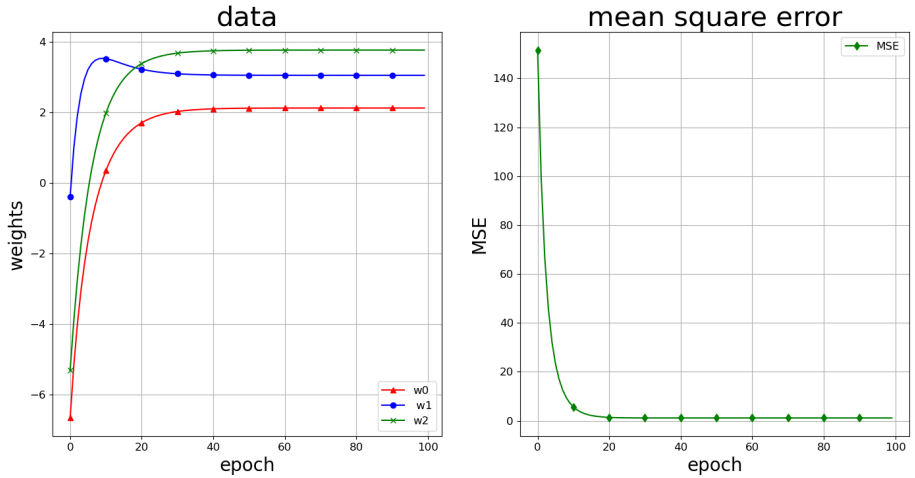
0	5.81854	5.67519	9.49996



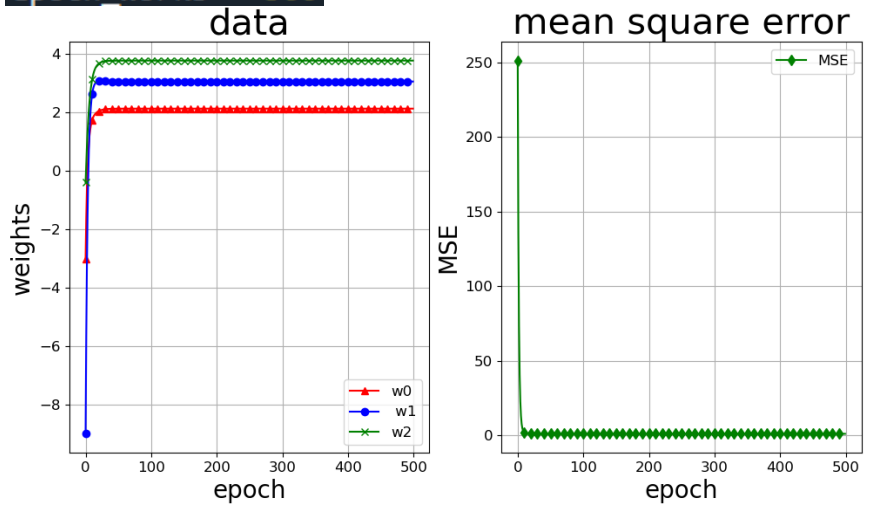
#과제3

3) 실습#2에서 구현한 경사하강법 함수를 다차원 데이터 입력이 가능하도록 변경해 Optimal Solution (weights, $\boldsymbol{w}$)을 구하고, 학습 진행(epoch)에 따른 w와 MSE에 대한그래프를 그려라. (epoch control)

```
epoch_work3 = 100
```



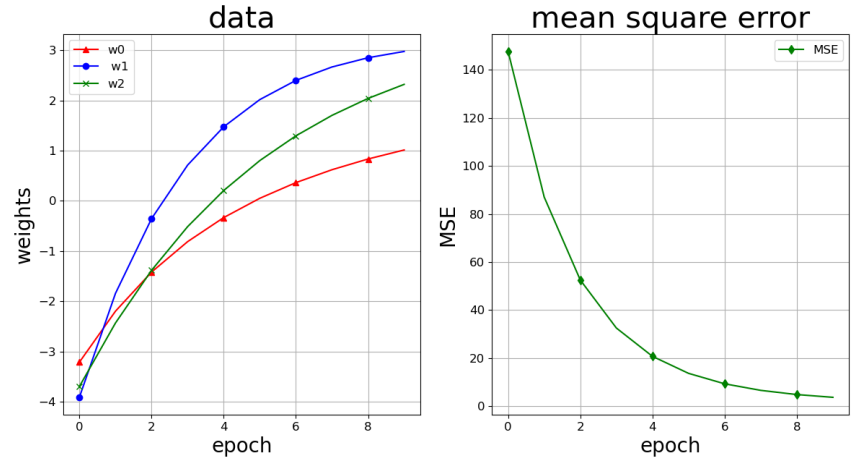

```
epoch_work3 = 500
```



1) 500으로 늘었을 때는 학습이 완료 됐지만 쓸데없는 학습량이 많아 데이터가 커짐

-> 반복 횟수대신 웨이트가 변하지 않을 때를 조건으로 걸면 해결 됨

```
epoch_work3 = 10
```

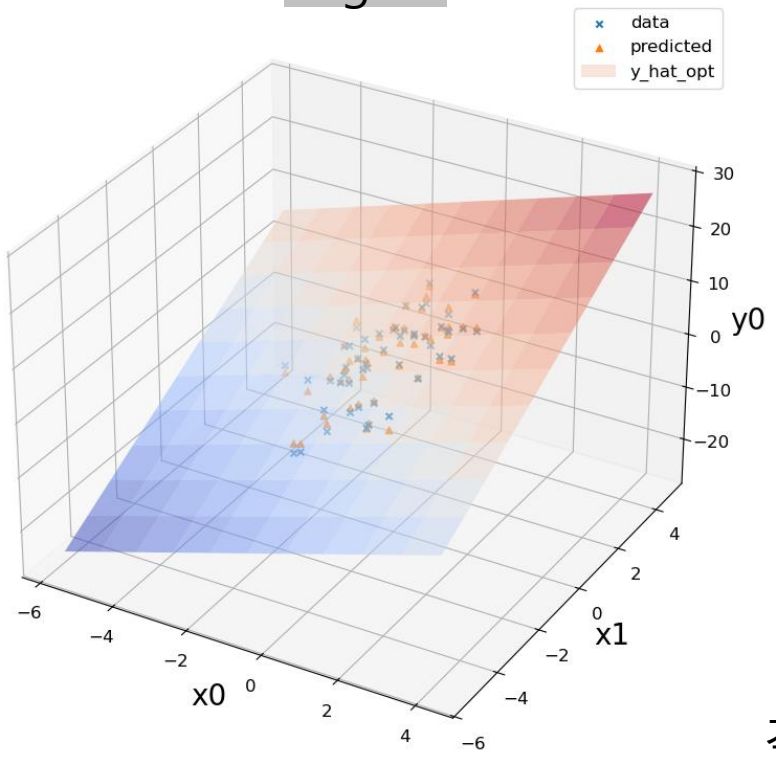


2) 10으로 줄었을 때는 learning rate가 작아졌을 때 처럼 학습이 완료되지 못함

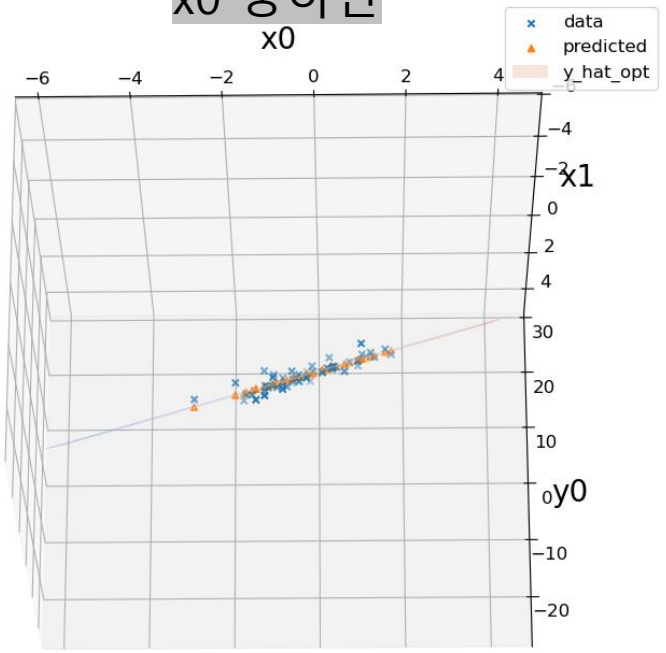
#과제3

4) 3차원 축에 y와 Optimal Solution을 이용한 $\hat{y}$을 점으로 표시하고, $\hat{y}$은 3차원 축에 평면으로 나타내라

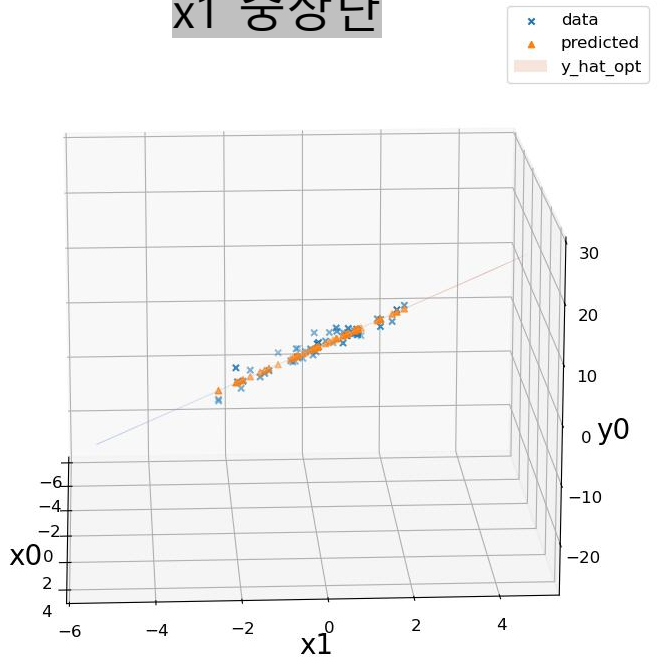
original



x0 중하단



x1 중상단



각 축을 기준으로 보았을 때 특징이 1개였던 과제 2번의 optimal solution과 같이 예측면을 기준으로 비슷한 모양의 그래프를 가진다.