# Machine Learning

8주차 실습

-chap.3 로지스틱 회귀

전자공학과

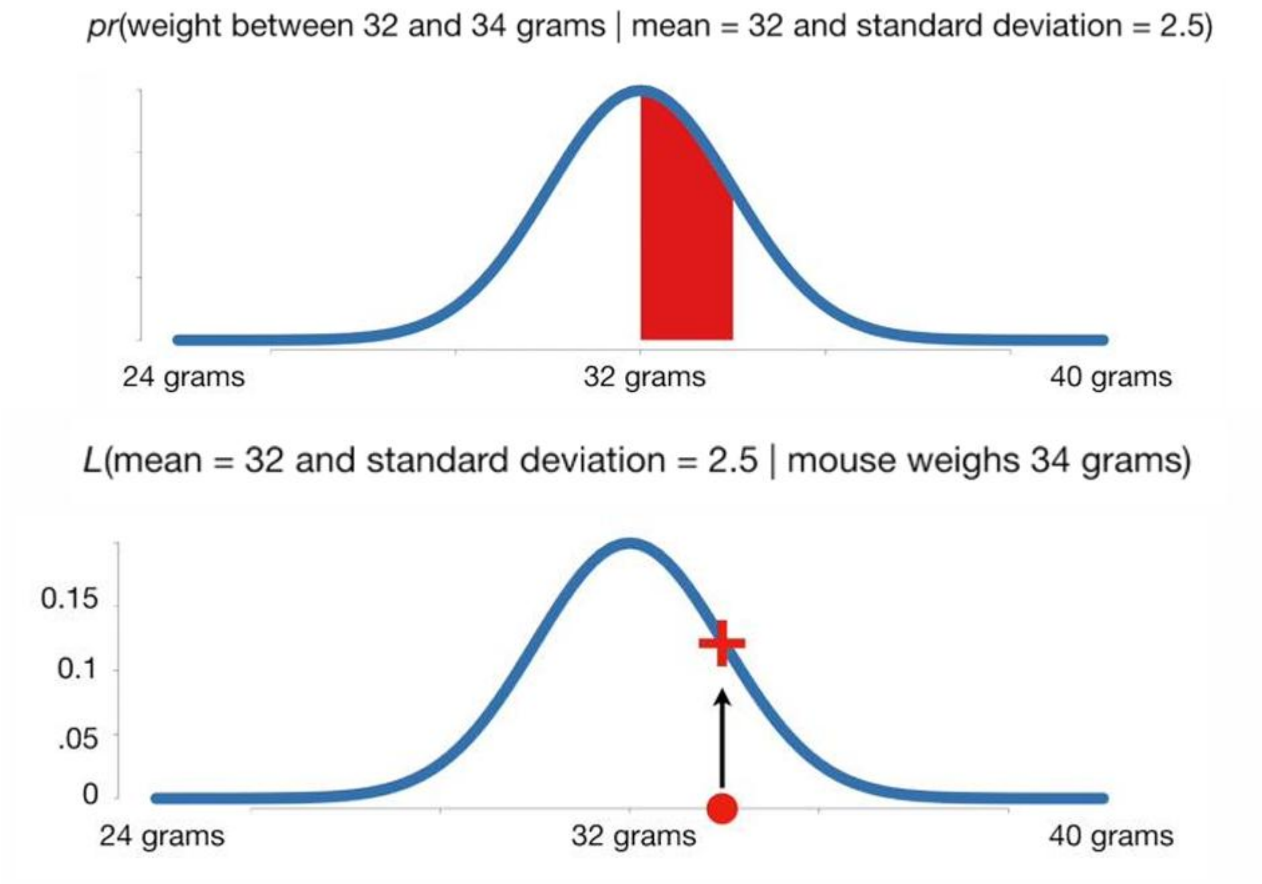
2022144007

김의진

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

- Probablity VS Likelihood




Probability → 주어진 파라미터(weight)로 새로운 입력에 대한 확률
→ 예측의 개념

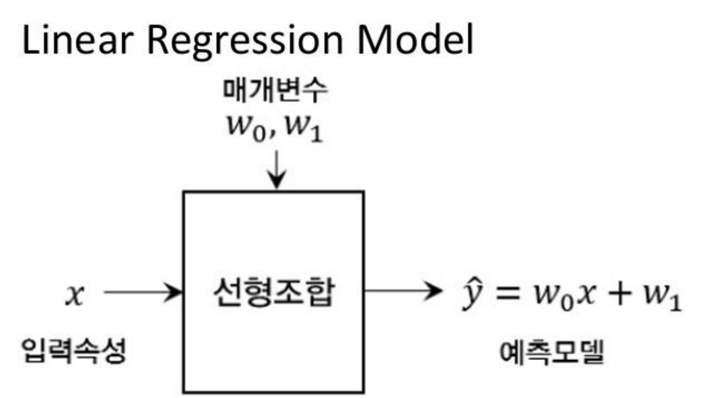
Likelihood → 주어진 데이터를 이용해 파라미터(weight) 찾는 과정
→ 학습의 개념

→ Maximum Likelihood
:주어진 데이터 이용해 가장 합리적인 확률 분포(weight)찾는 과정

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

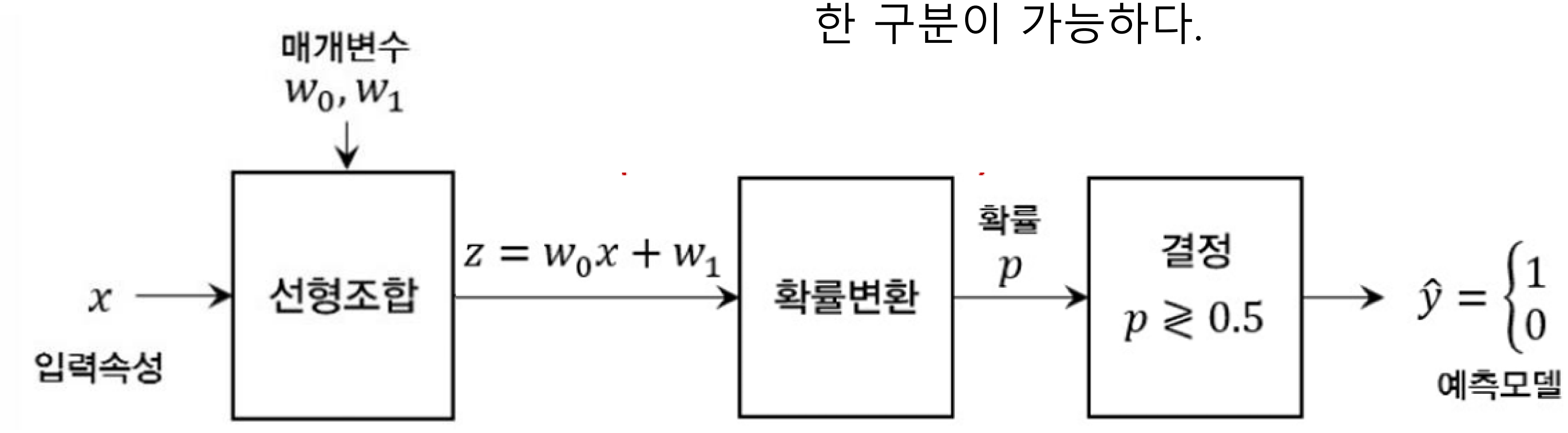
## Regression과 classification




1)Classification은 regression 한 모델에서 0~1사이의 함수를 이용해 확률로 변환 후 0과 1로 구분해 class를 나눈다.

2-1 ) Regression: class를 예측하는 것이 아닌 숫자를 예측
→ 경향성을 예측하기 좋음

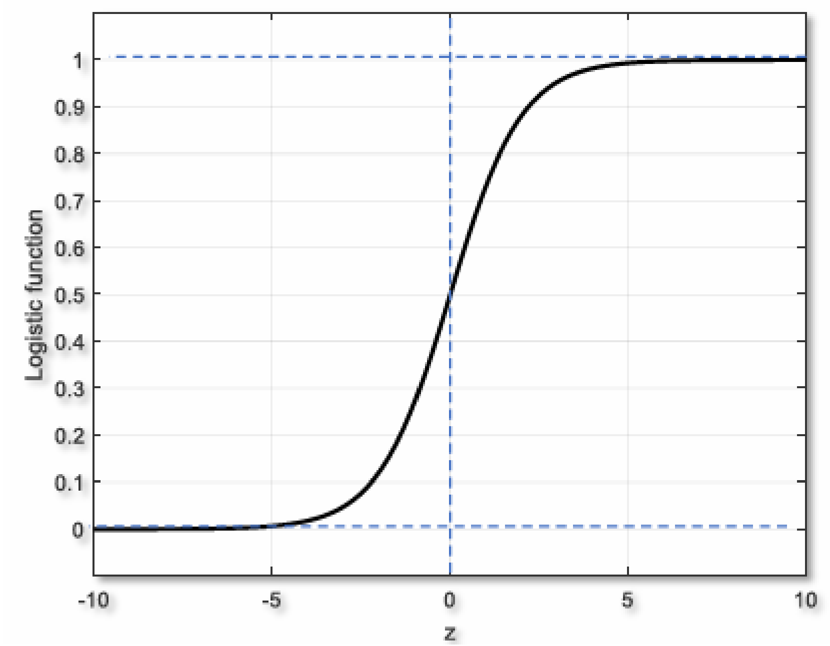
2-2) Classification: class를 나눠 “맞다”, “아니다” 같은 명확한 구분이 가능하다.



# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

## 확률변환 함수로 sigmoid function을 쓰는 이유




$$f(z) = \frac{1}{1+e^{-z}}$$

$$f(z) = \begin{cases} 0, & z \rightarrow -\infty \\ 0.5, & z = 0 \\ 1, & z \rightarrow \infty \end{cases}$$

$$\frac{df(z)}{dz} = f(z)(1 - f(z))$$

1) 0~1 사이로 제한된 bounded 함수여서 확률적 해석이 가능하다.

2) Step형태라 이진분류에서 class 경계를 명확히 나누는데 유리하다.

3) 미분이 가능한 간단한 수식을 가지고 있어서 경사하강법 기반의 regression 학습에 효율적이다.

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

weight 식의 변화

$$w^* = \arg\max \prod_{n=0}^{N-1} p_n^{y_n}(1 - p_n)^{1-y_n}$$

$$w^* = \arg min - \prod_{n=0}^{N-1} p_n^{y_n}(1 - p_n)^{1-y_n}$$

$$w^* = \arg min - \sum_{n=0}^{N-1} \{ y_n lnp_n + (1 - y_n)\ln(1 - p_n) \}$$

1) $p_n^{y_n}(1 - p_n)^{1-y_n}$ : 1에 가까울수록 예측이 잘 된 것

Ex) y = 1일 때 $p^1 (1 - p)^0 = p$

y = 0일 때 $p^0 (1 - p)^1 = 1 - p$

→ 우리가 학습시키는 머신은 최소가 되게하는 동작을 해야하는데 이 식은 최대의 개념.

2) – 를 붙여서 최소의 개념을 만들어 줌

3) 단조 증가 함수인 자연로그를 붙여서 합 형태로 바꿔줌

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

**자연로그를 취하는 이유**

**자연로그 $ln(x)$의 성질**

**1)** 단조증가함수라 값이 커지면 로그도 커진다.

**2)** 원래 함수 $f(x)$가 최대인 지점 $x = a$라면, $\ln(f(x))$도 동일한 지점에서 최대를 갖는다.

→자연로그를 취해도 극댓값 또는 극솟값의 위치는 바뀌지 않는다.

**3)** 곱셈식을 덧셈식으로 바꿔줌

$$w^* = \arg min - \prod_{n=0}^{N-1} p_n^{y_n}(1 - p_n)^{1-y_n}$$

⬇

**→ 미분 과정 계산이 쉬워짐**

$$w^* = \arg min - \sum_{n=0}^{N-1} \{ y_n ln p_n + (1 - y_n) \ln(1 - p_n) \}$$

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

최종 Cross Entropy Error와 미분

$$\epsilon_{CEE} = -\frac{1}{N}\sum_{n=0}^{N-1}\{y_n ln p_n + (1 - y_n)\, ln(1 - p_n)\}$$

$$\frac{\partial}{\partial w}\epsilon_{CEE}(w) = \frac{1}{N}\sum_{n=0}^{N-1}(p_n - y_n)x_n$$

이 때 $x_n = [x_{0,n}, x_{1,n}, x_{2,n}, x_{3,n} \cdots x_{M+1,n}\ 1]^T$

바뀐 weight update 식

$$w_{new} = w_{old} - \alpha \frac{\partial}{\partial w_{old}}\epsilon_{CEE}(w_{old})$$

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

CEE 적용

```python
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 logistic_regression(epoch, IR, x, y):


    w_hist = []
    CEE_hist = []
    ACC_hist = []

    #epoch만큼 반복해 weight 업데이트
    for i in range(epoch):

        if i == 0:
            w = np.random.rand(len(x[0,:])) * 5

        w = np.reshape(w, [-1, 1])
        z = np.dot(x, w)
        p = sigmoid_funtion(z)
        y_h = classification_data(p)
        y_h.reshape(-1, 1)

        CEE = -np.mean(y * np.log(p) + (1 - y) * np.log(1 - p))

        accuracy = data_accuracy(y_h, y)

        w_hist.append(w)
        CEE_hist.append(CEE)
        ACC_hist.append(accuracy)

        x_t = np.transpose(x)
        dif = np.dot(x_t, (p - y)) / len(y)

        w = w - IR * dif

    return w, w_hist, CEE_hist, ACC_hist
```

1) Linear regressio에서 data와 weight의 선형 조합을 z로 받아줌
2) Sigmoid 함수에 z를 넣어 0~1에서 bounde된 값 p에 받음
3) MSE 대신에  CEE를 사용하여 Error 확인
4) 기울기 구하기 전에 입력 데이터를 전치 시켜줌

# #과제 1

1) Chap.1 에서 구현한 linear regression에 대한 경사하강법 사용자 지정함수를 응용 및 수정하여, logistic regression을 위한 경사하강법 사용자 지정함수를 구현하라.

MSE 적용

```
def logistic_regression_with_MSE(epoch, IR, x, y):


    w_hist = []
    MSE_hist = []
    ACC_hist = []

    #epoch만큼 반복해 weight 업데이트
    for i in range(epoch):

        if i == 0:
            w = np.random.rand(len(x[0,:])) * 5

        w = np.reshape(w, [-1, 1])
        z = np.dot(x, w)
        p = sigmoid_funtion(z)
        y_h = classification_data(p)
        y_h.reshape(-1, 1)

        error = p - y


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

        accuracy = data_accuracy(y_h, y)

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

        x_t = np.transpose(x)
        w_dif = 2* np.dot(x_t, error)/len(y)

        w = w - IR * w_dif

    return w, w_hist, MSE_hist, ACC_hist
```

1) CEE를 이용한 logistic regression과 비슷하게 구성

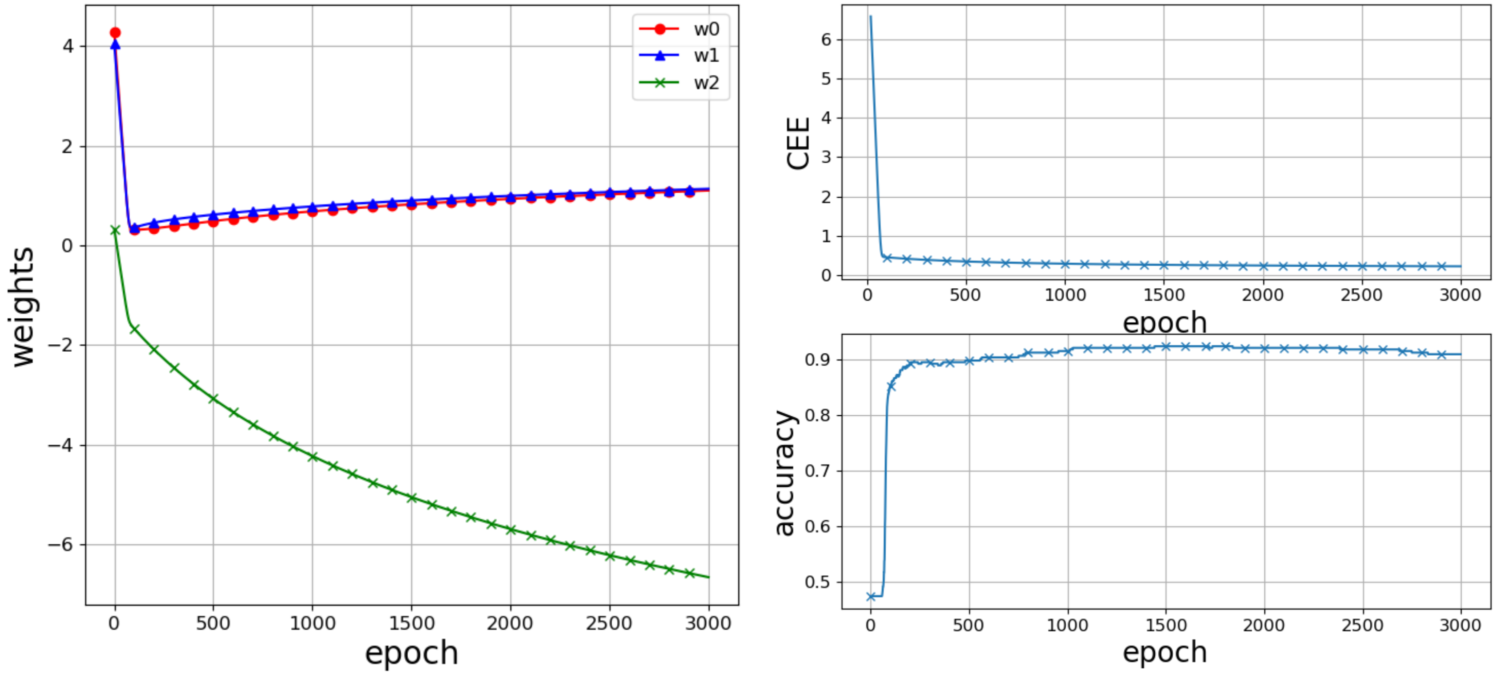
MSE와 CEE의 차이

| 항목 | MSE | CEE |
|---|---|---|
| 주용도 | 회귀문제 | 분류문제 |
| 확률 해석 | 단순 오차 | 확률적으로 log likelihood 해석 가능 |
| 수렴속도 | 상대적으로 느림 | 일반적으로 빠르게 수렴 |
| 분류 성능 | 경계 불확실해 성능 떨어질 수 있음 | 명확한 경계 지녀 높은 정확도 기대 가능 |
| 예측 확률 해석 | 예측값 0~1이지만 확률이라 보기 어려움 | 예측값을 정확한 확률로 해석 가능함 |

# #과제1

3) 앞에서 구현한 경사하강법 함수를 이용해 Training set을 활용하여 logistic regression 모델학습을진행하고, 학습진행(epoch)에 따른$\boldsymbol{w}$/ CEE / training set에 대한 분류 정확도를 각각 그래프로 나타내라

Training set에 대한 weigh와 CEE, accuracy 그래프




```
epoch = 3000
IR = 0.05
```

1) w2 빼고 w1, w2는 수렴하는 모습이 보임

2) CEE와 accuracy도 잘 수렴함.

Weight가 완전히 수렴하지 않았는데도 CEE와 accuracy가 수렴되는 안정된 값을 갖는 이유

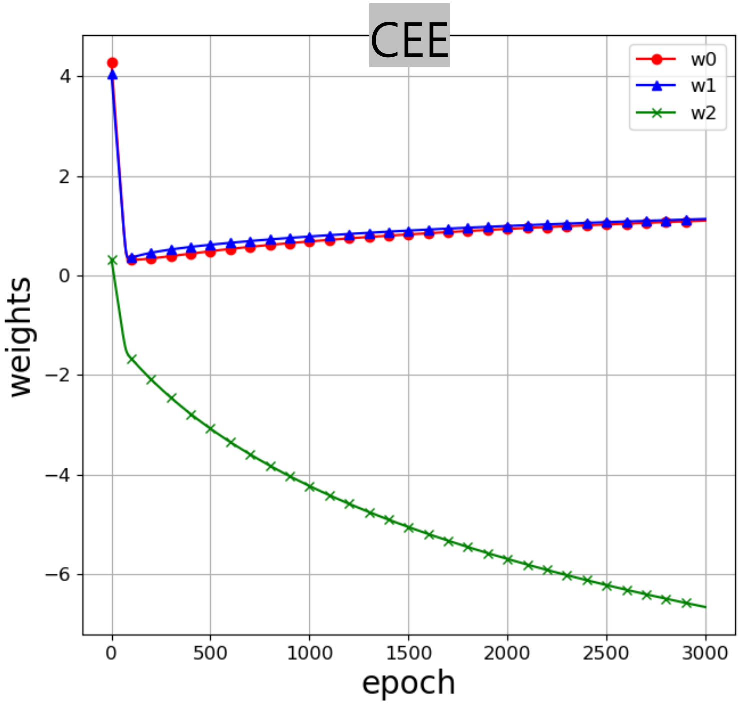
-1 weight는 다른 값으로 같은 decision boundary를 만들 수 있음(비율 일정하기 때문)

-2 weight가 이미 적절한 위치에 있어서 예측값이 0 또는 1로 수렴 가능함

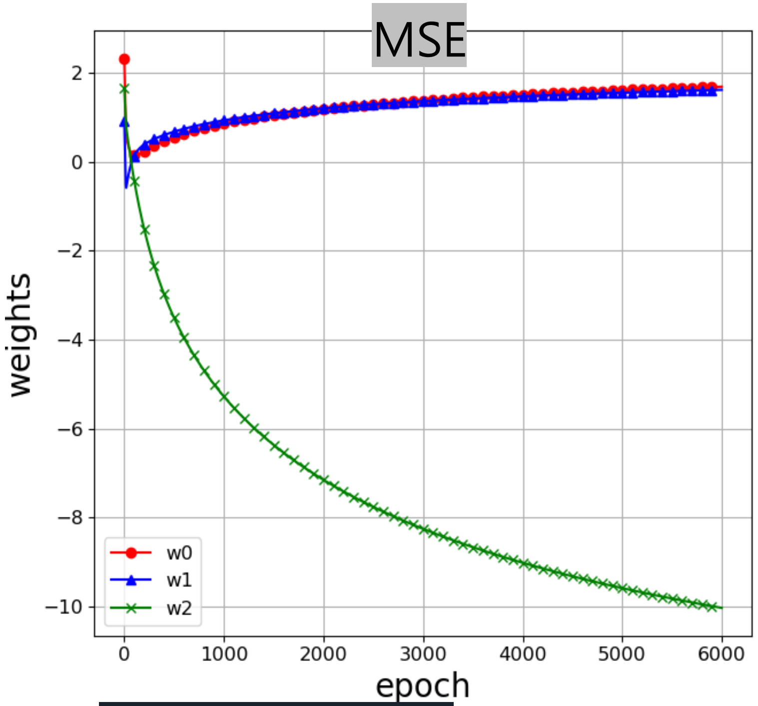
# #과제1

3) 앞에서 구현한 경사하강법 함수를 이용해 Training set을 활용하여 logistic regression 모델학습을진행하고, 학습진행(epoch)에 따른 $\boldsymbol{w}$/ CEE / training set에 대한 분류 정확도를 각각 그래프로 나타내라

Training set에 대한 CEE, MSE의 weigh변화




```
epoch = 3000
IR = 0.05
```




```
M_epoch = 6000
M_IR = 0.05
```

1) CEE와 같이 MSE로 구할 때도 weight가 안정적으로 update 되는 모습을 보임

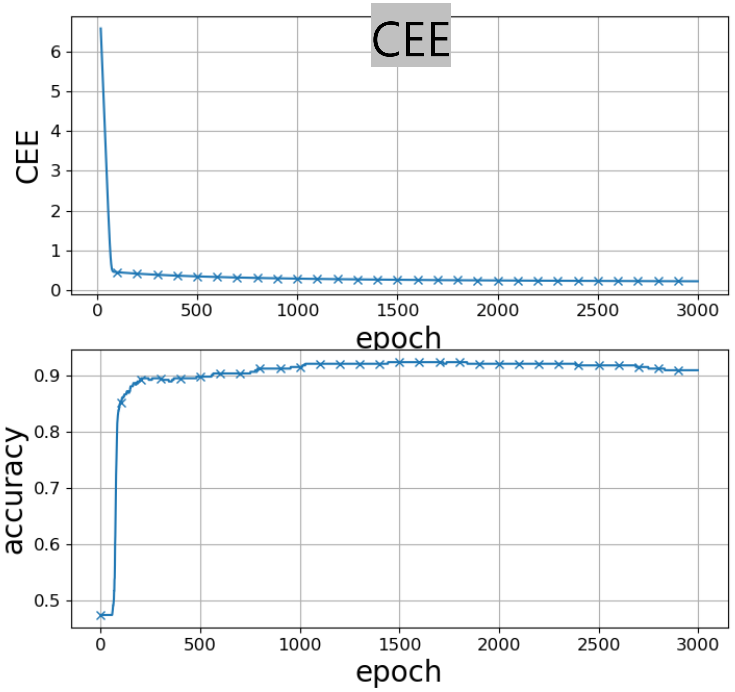
2) CEE의 weight 수렴 구간이 MSE의 weight 수렴 구간보다 빠름

3) MSE의 학습이 좀 더 느림

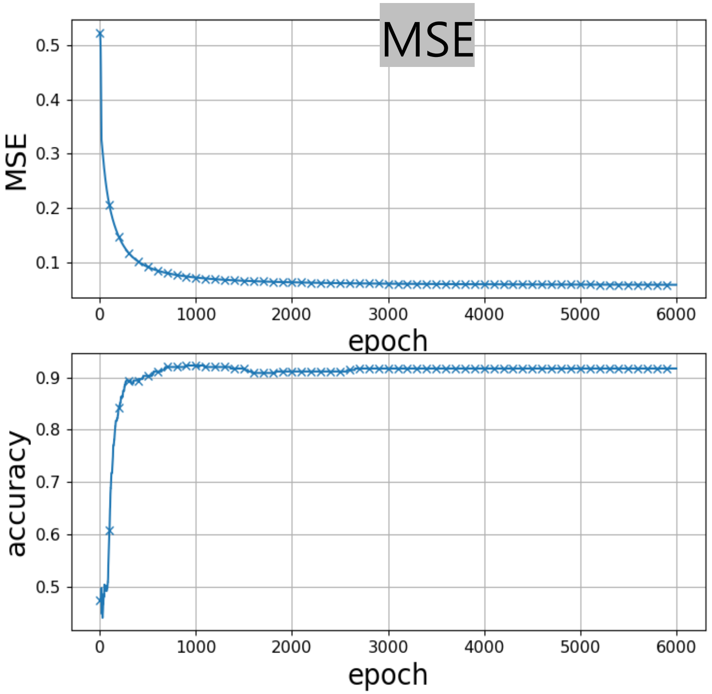
# #과제1

3) 앞에서 구현한 경사하강법 함수를 이용해 Training set을 활용하여 logistic regression 모델학습을진행하고, 학습진행(epoch)에 따른$\boldsymbol{w}$/ CEE / training set에 대한 분류 정확도를 각각 그래프로 나타내라

Training set에 대한 CEE, MSE의 accuracy변화




```
epoch = 3000
IR = 0.05
```




```
M_epoch = 6000
M_IR = 0.05
```

1) CEE와 MSE, 각 accuracy가 잘 수렴함

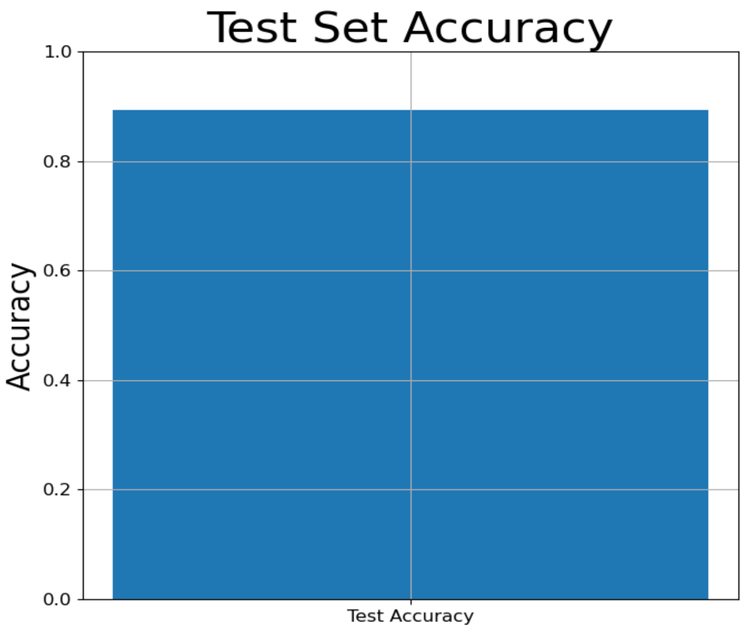
2) weight 업데이트 시킬 때와 같이 CEE의 수렴구간이 MSE의 수렴구간보다 빠름

3) CEE의 학습 속도가 MSE보다 더 빠름

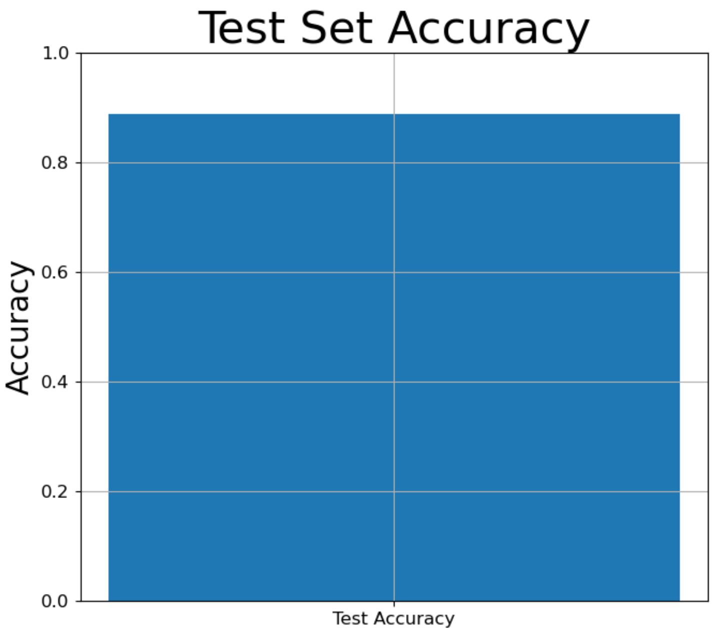
# #과제1

4) 학습이 완료된 모델을 활용하여, Test set을 분류하고, 분류 정확도를 나타내라.

CEE로




MSE로




CEE, MSE로 학습한 머신의 분류 정확도가 각각 0.893, 0.887이 나옴

→ 분류 정확도가 큰 차이 있지는 않지만 CEE가 더 정확한 예측을 함

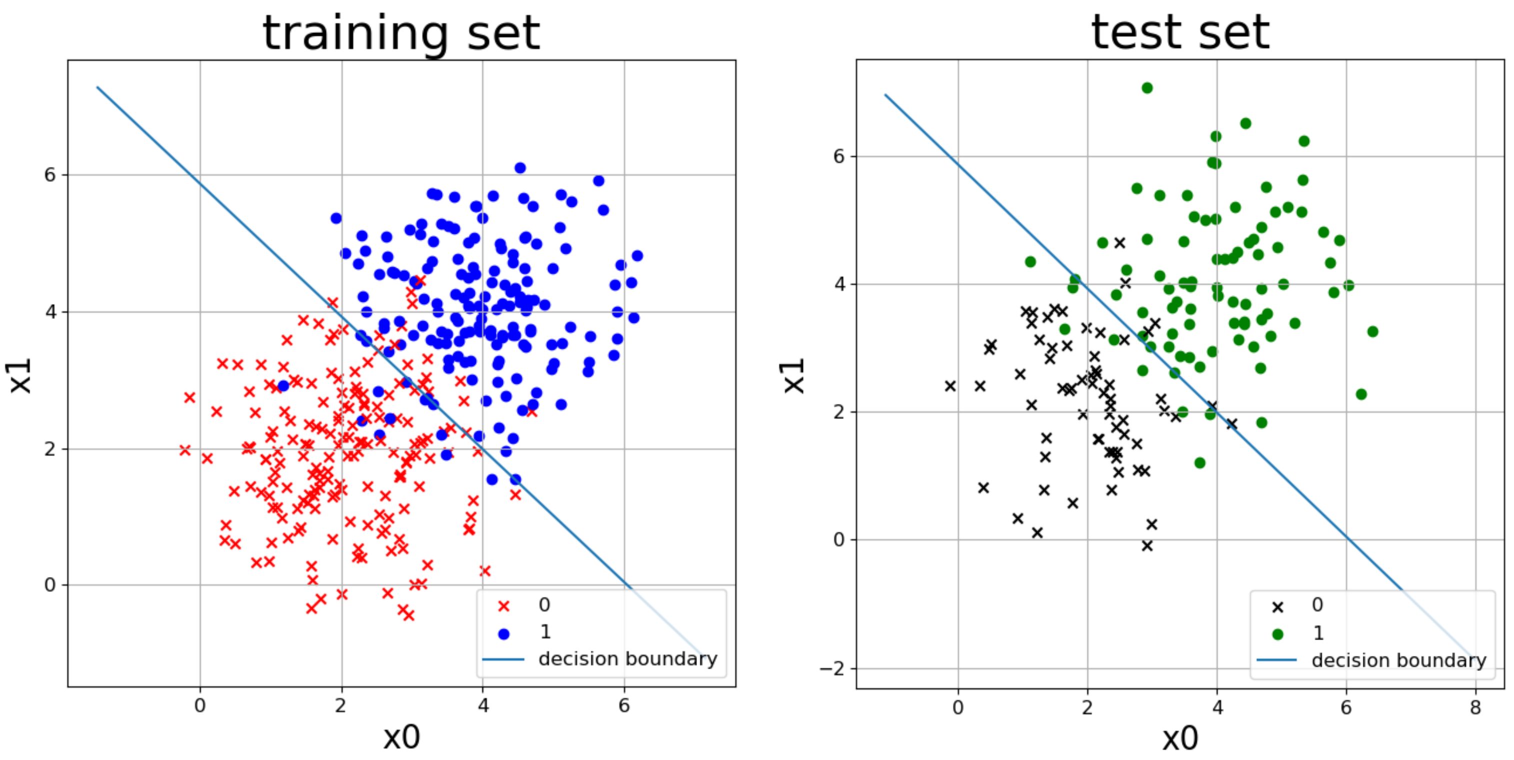
```
CEE로 학습한 머신의 test set Accuracy:  0.8933333333333333
MSE로 학습한 머신의 test set Accuracy:  0.8866666666666667
```

분류 정확도가 1이 될 순 없을까?

# #과제1

5) 학습이 끝난 모델을 활용하여, Training set 및 Test set에 대한 Decision Boundary 그래프를 각각 그려라.

CEE




데이터 분포를 보면 겹치는 class가 있음
→ 1)선형 분리 불가능함
2)분류 했다간 오히려 overfitting될 가능성 있음

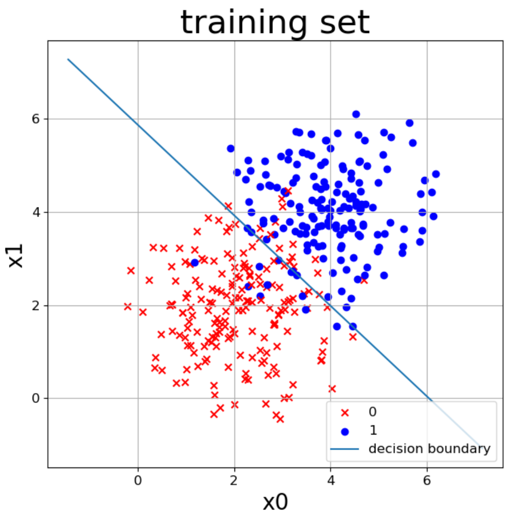
→ 1. 분류 정확도가 1이 되려면 두 class가 확실히 분리 되어 있어야 함
2. training, test set이 매우 유사해야 함

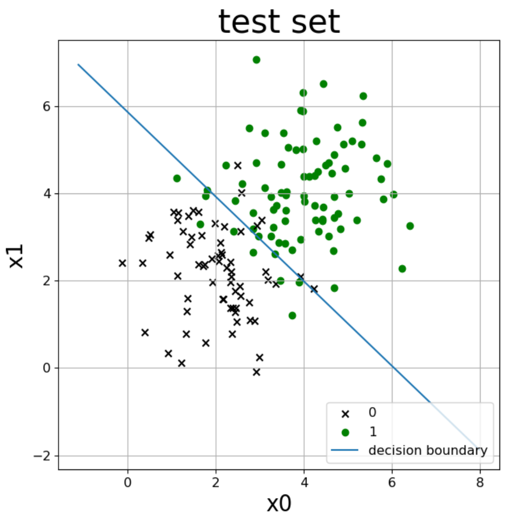
위 내용 고려하면 분류를 잘 해냄

# #과제1

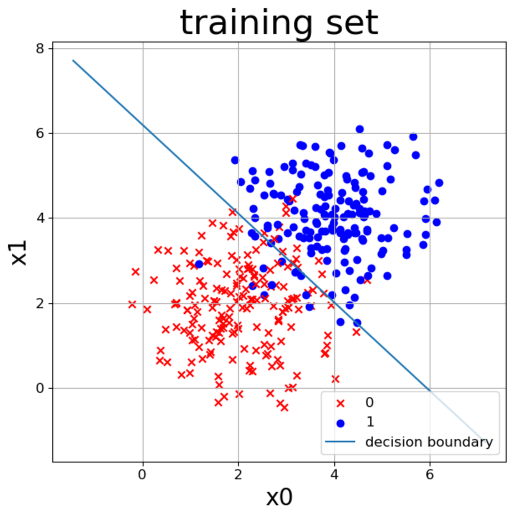
5) 학습이 끝난모델을 활용하여, Training set 및 Test set에 대한 Decision Boundary 그래프를 각각 그려라.

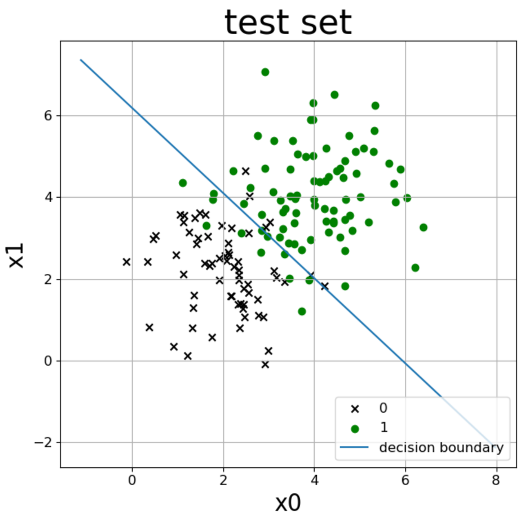
CEE




MSE




CEE로 구한 Decision boundary와 MSE로 구한 Decision boundary는 살짝 다르지만 둘 모두 데이터를 잘 분류 해냄

→ 이번 과제 데이터에 대해서 CEE와 MSE를 이용한 logistic regression은 MSE가 CEE보다 학습 속도는 느리지만 성능은 비슷함.

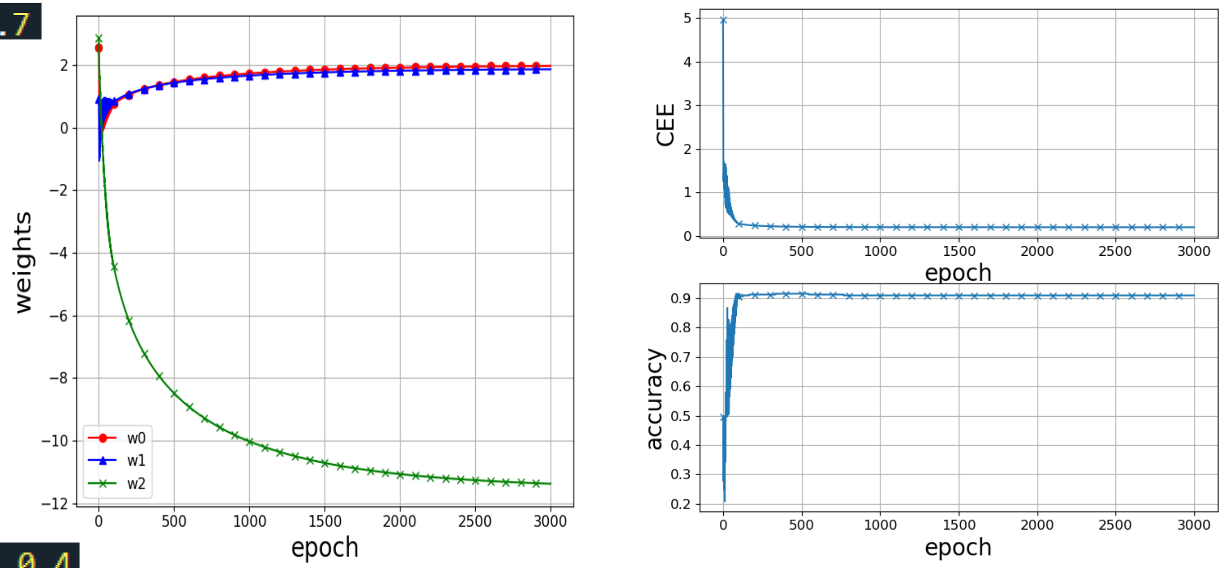
→ 일반적인 상황(데이터 복잡, 클래스 잘 분리 x)을 생각하면 CEE가 logistic regression에서 더 적합한 loss function임

# #과제 1

## 6)추가실습 – learning rate가 몇일 때 불안정할까?

IR = 0.7

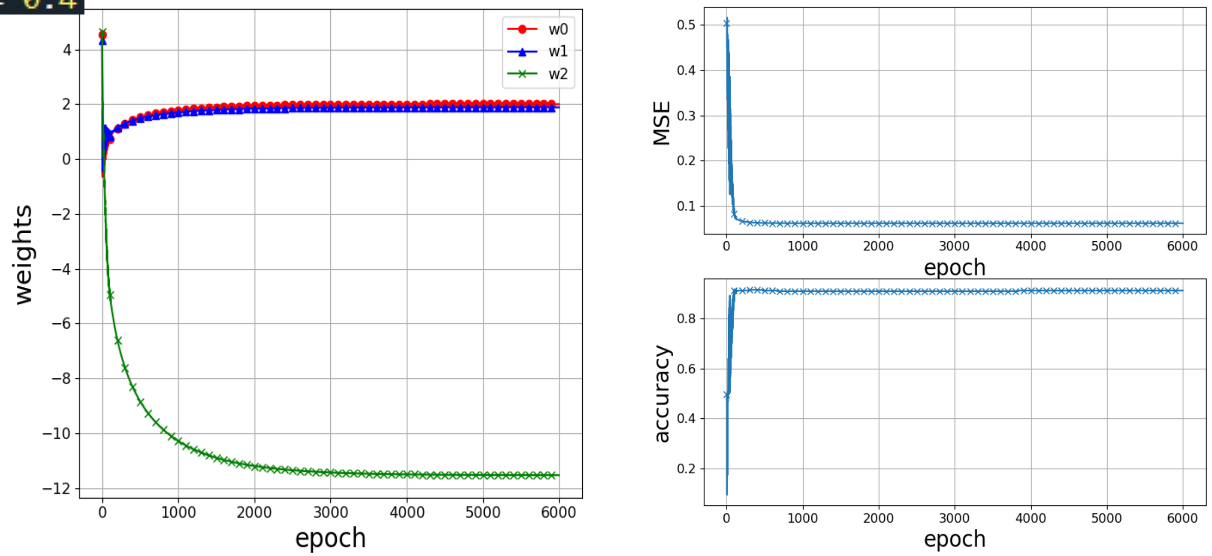
CEE




Loss function이 CEE일 때 learning rate가 0.7이면 초기값이 툭툭 튀기 시작함

M_IR = 0.4

MSE



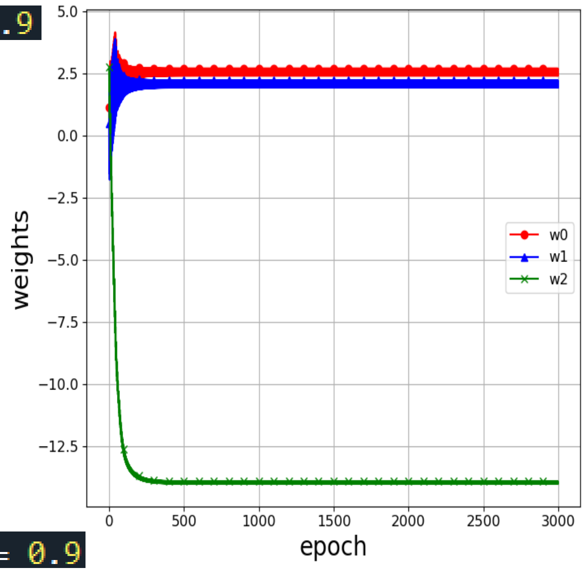

Loss function이 MSE일 때 learning rate가 0.4면 초기값이 툭툭 튀기 시작함.

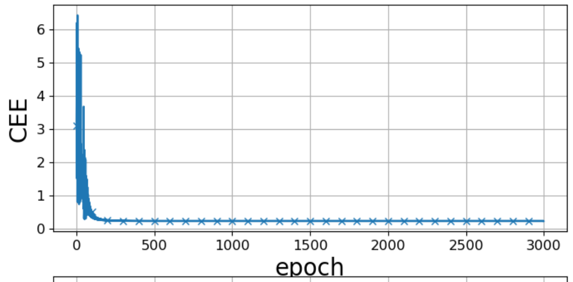
# #과제 1

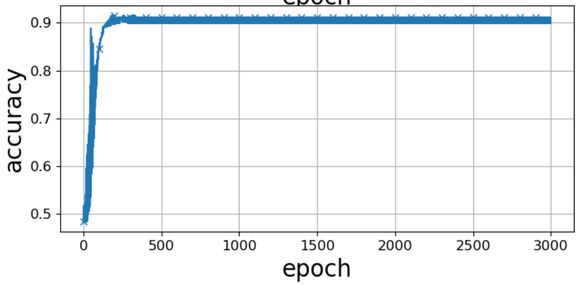
## 6)추가실습 – learning rate가 몇일 때 불안정할까?

lR = 1.9

CEE

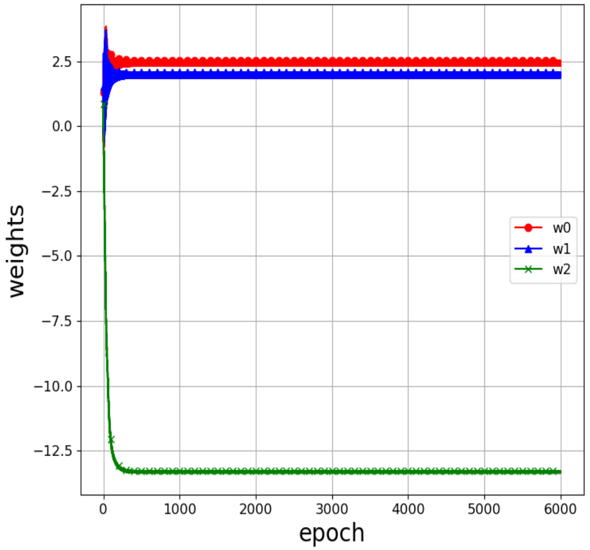


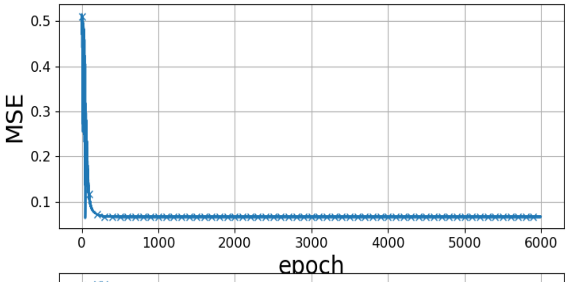



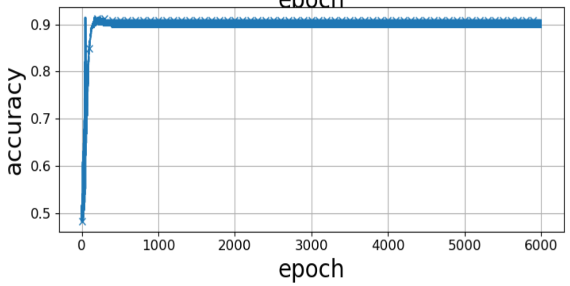




M lR = 0.9

MSE




Loss function이 CEE일 때
learning rate가 1.9면 그래프가
진동하기 시작함

→ 1.9* 이상일 때 불안정

Loss function이 MSE일 때
learning rate가 0.9면 그래프가
진동하기 시작함

→ 0.9 이상일 때 불안정

*데이터가 잘 정규화 돼있고
loss function의 경사가 작으면
1보다 큰 learning rate 가능