신경망

JSY

순서

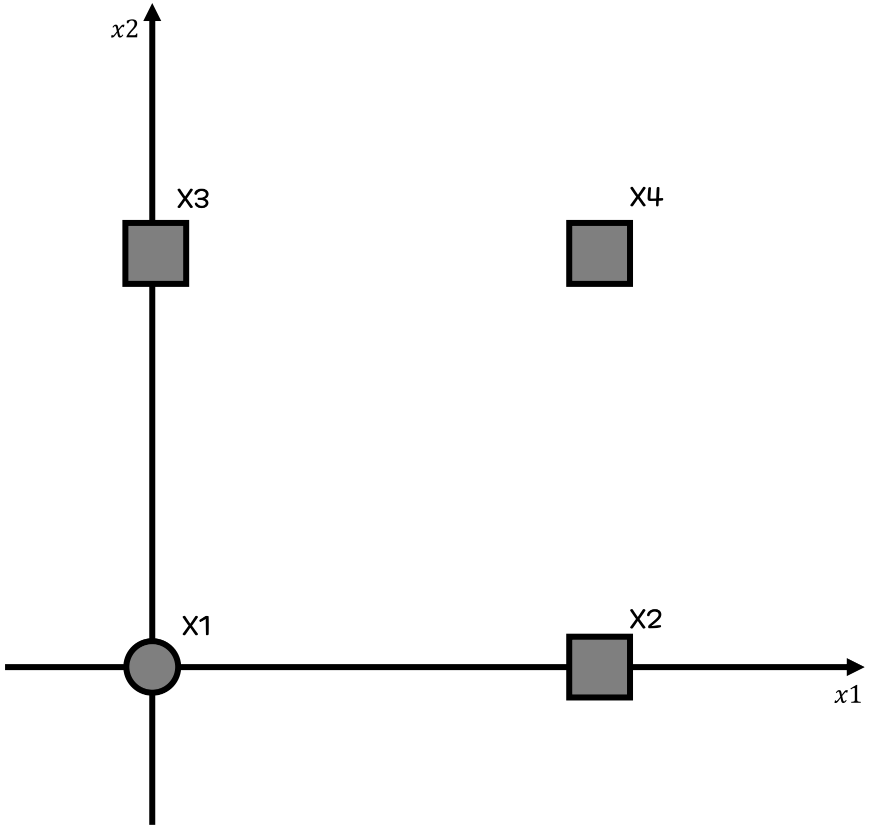
- 퍼셉트론
  - 단층 퍼셉트론
  - 다층 퍼셉트론
  - 한계
- 오차 역전파
- 딥러닝의 시작

퍼셉트론

- 학습이 가능한 초창기 신경망 모델(아주 원시적인 신경망)
- 노드, 가중치, 층(Layer)에 대한 개념이 새로이 도입됨
- 데이터가 선형적으로 분리될 수 있다면, 미분을 통한 학습은 반드시 100%의 정확도에 수렴한다.

퍼셉트론 - 구조

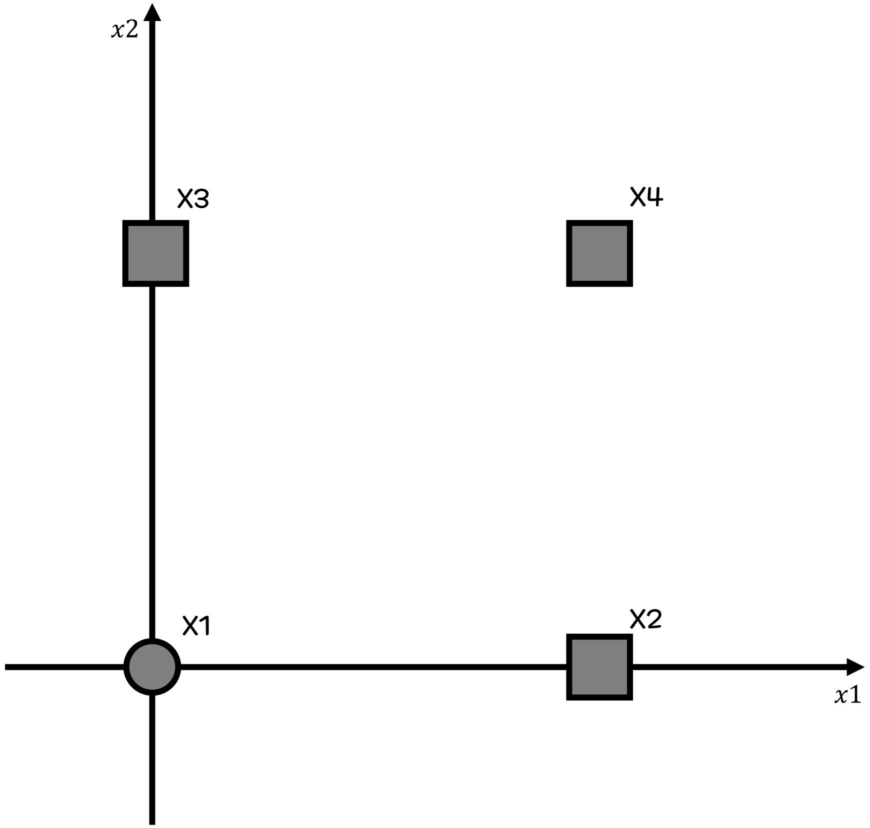
- 동그라미와 네모를 구분하는 선형 분리기를 설계 한다고 가정

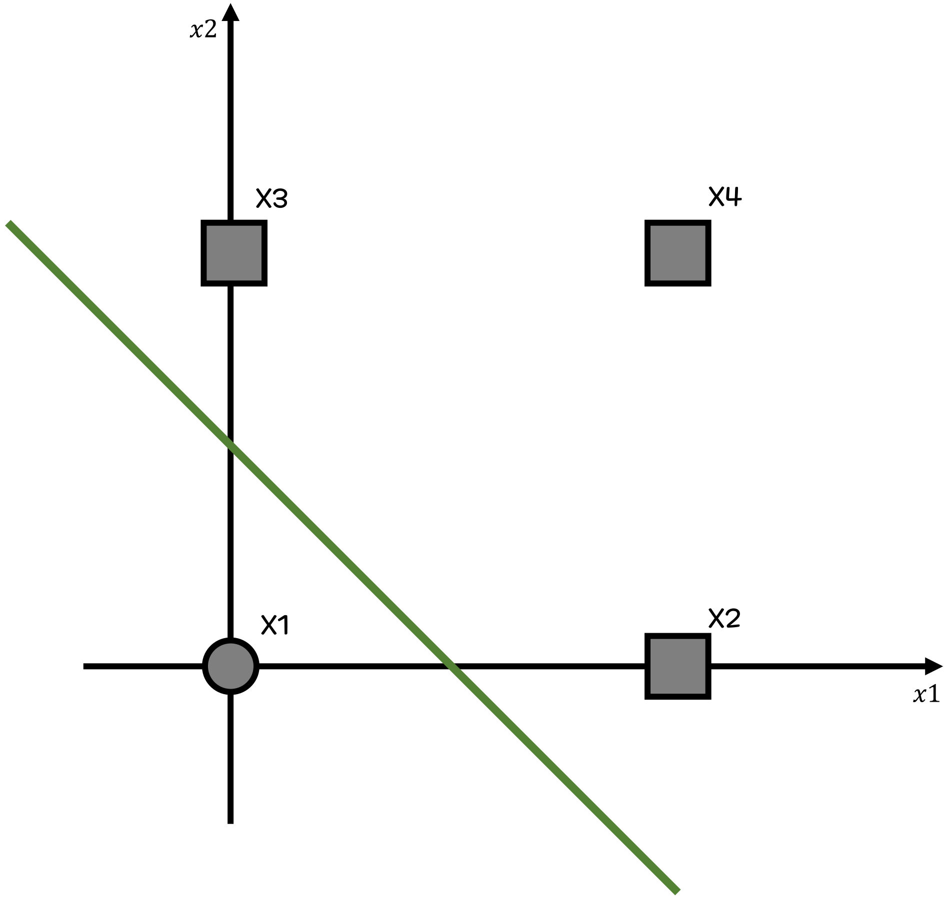


- X = [[0, 0] [1, 0], [0,1], [1, 1]]

퍼셉트론 - 구조

- 동그라미와 네모를 구분하는 선형 분리기를 설계 한다고 가정

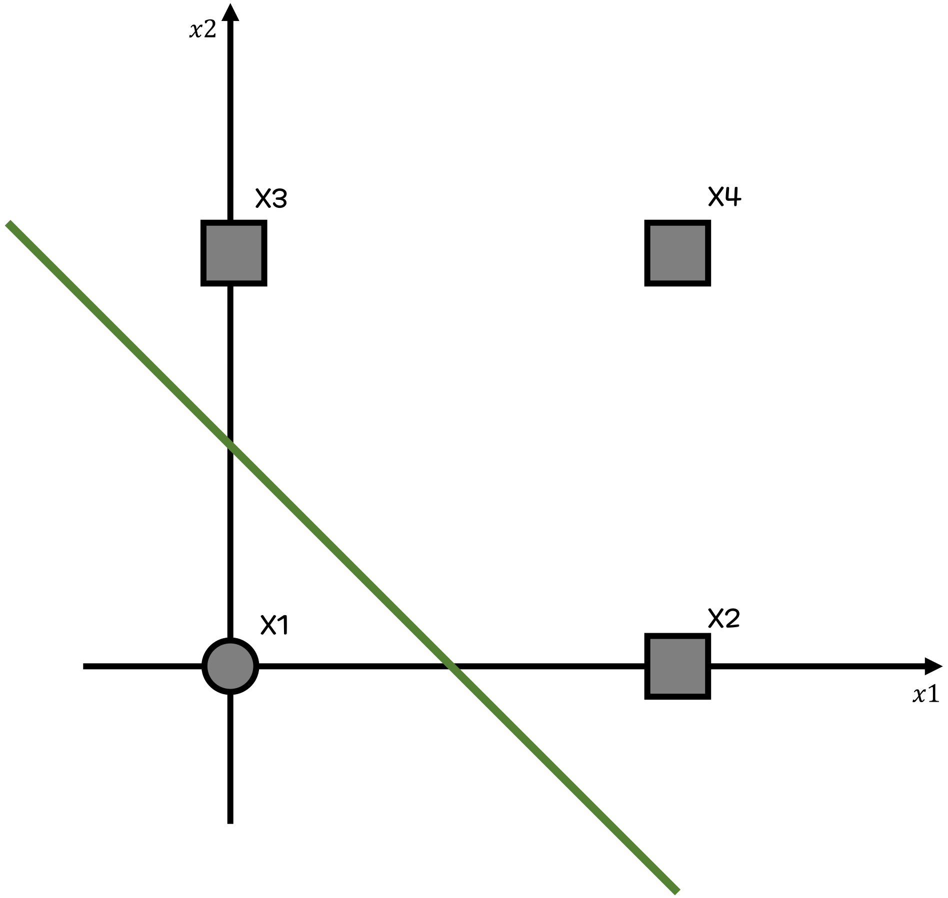


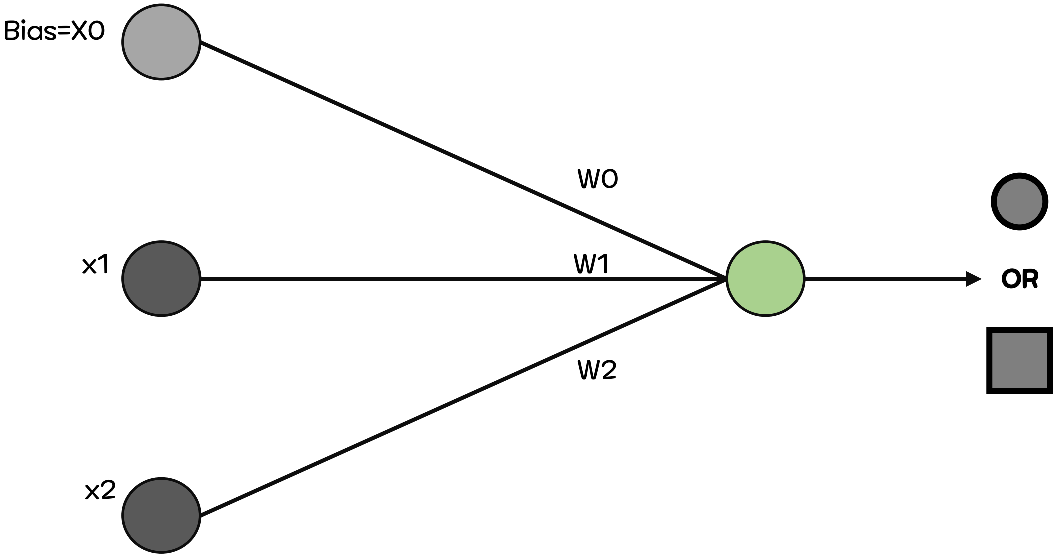


- X = [[0, 0] [1, 0], [0,1], [1, 1]]

퍼셉트론 - 구조

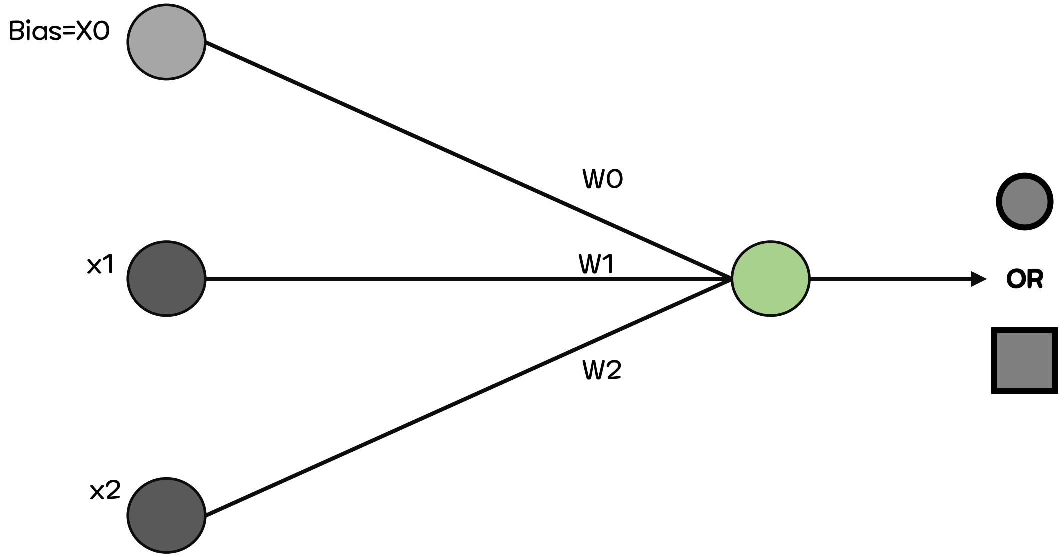
- 동그라미와 네모를 구분하는 선형 분리기를 설계한다고 가정





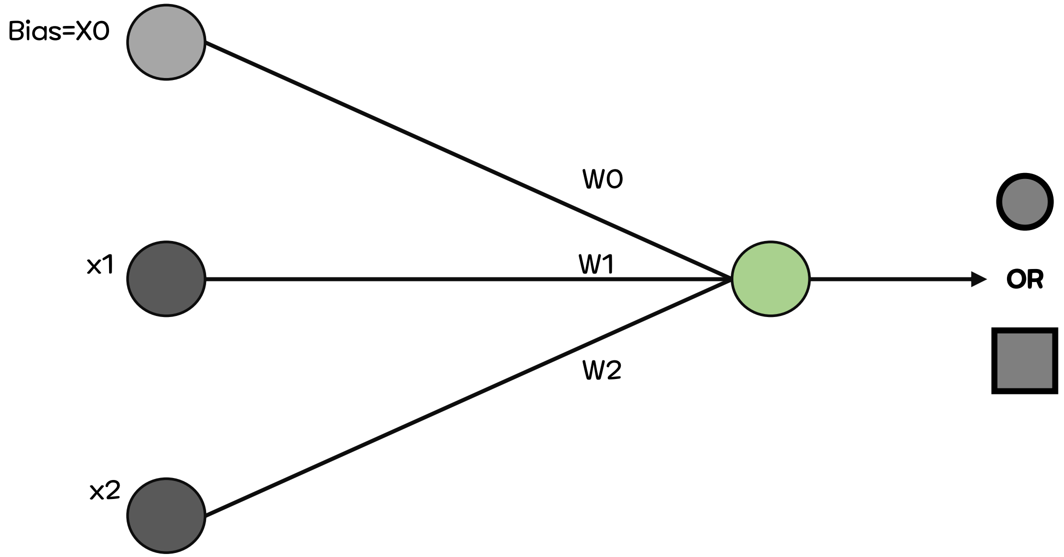
퍼셉트론 - 구조

- 동그라미와 네모를 구분하는 선형 분리기를 설계한다고 가정



- $Output = X_0 * w_0 + X_1 * w_1 + X_2 * w_2$

퍼셉트론 - 구조

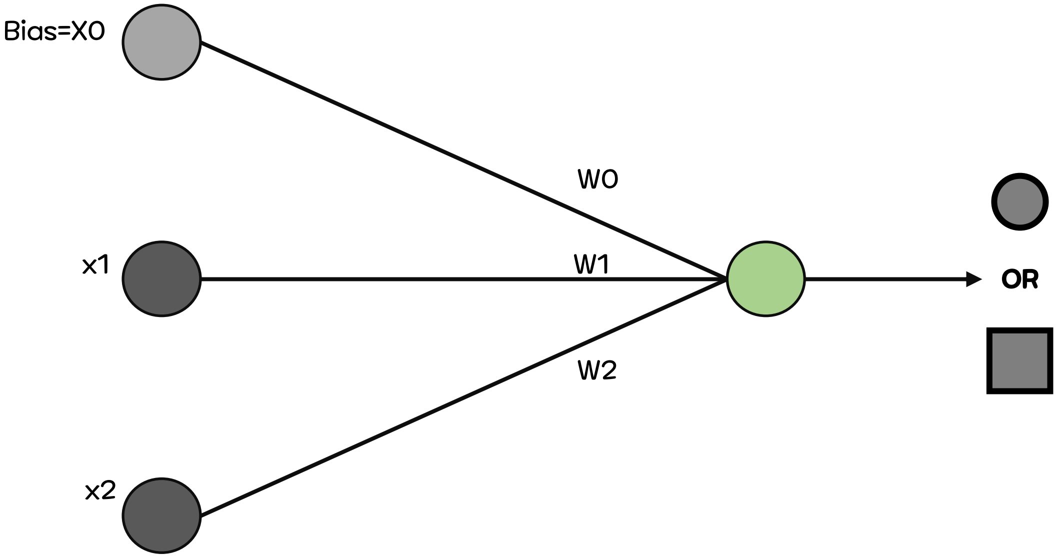


- 동그라미와 네모를 구분하는 선형 분리기를 설계한다고 가정

- $Output = x_0 * w_0 + x_1 * w_1 + x_2 * w_2$
- $Output = X * W^T$

여기서 $X = [X_0, X_1, X_2], W = [w_0, w_1, w_2]$

퍼셉트론 - 구조



- 동그라미와 네모를 구분하는 선형 분리기를 설계한다고 가정

- $Output = x_0 * w_0 + x_1 * w_1 + x_2 * w_2$
- $Output = X * W^T$

여기서 $X = [X_0,\ X_1,\ X_2], W = [w_0,\ w_1, w_2]$

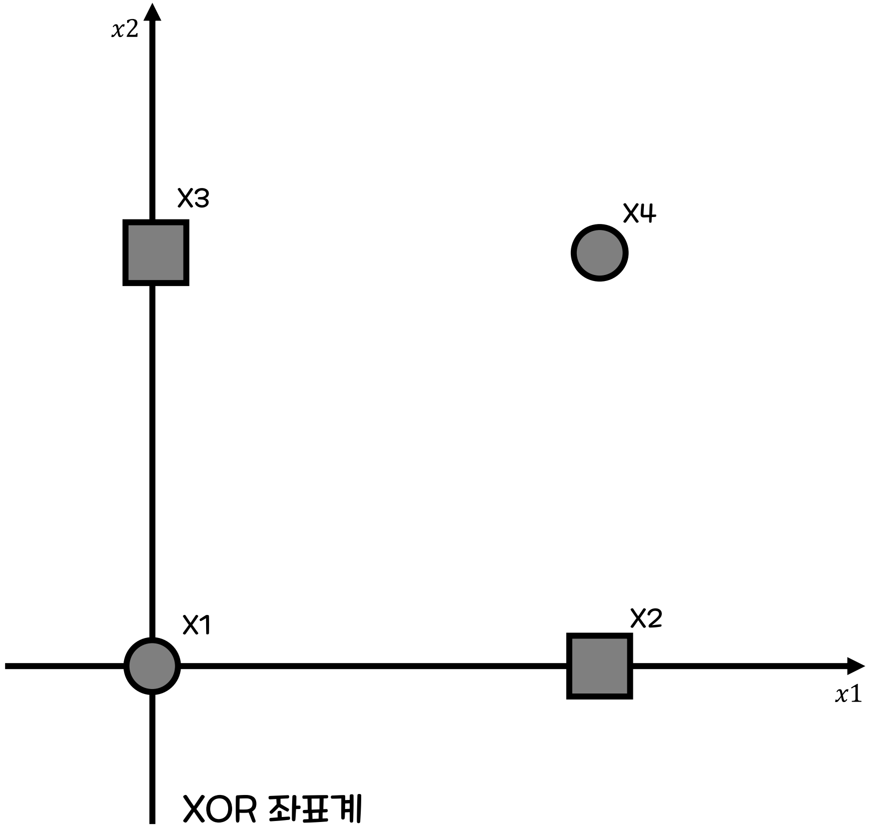
- $\tau(x) =$ if x 〉= 0 else
- $\tau(x) =$ 1 if x 〉= 0 else -1

퍼셉트론 - 학습

- Loss_fn(목적함수)를 설정하고 목적함수의 미분값을 가중치 갱신 규칙에 대입
  - $\theta_{new} = \theta_{old} - \alpha \frac{\Delta J(\theta)}{\Delta \theta}$
  - α는 학습률

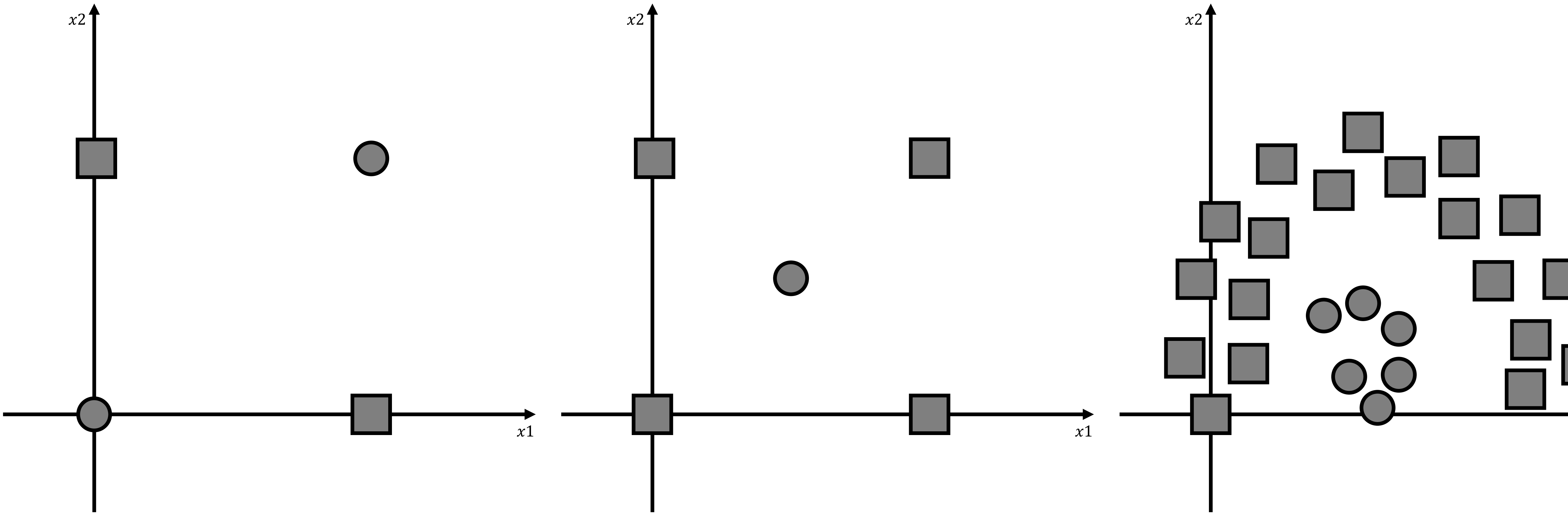
다층 퍼셉트론

- 퍼셉트론 1개로는 선형분리가 불가능한 상황에 치명적인 약점이 발생
  - ex) XOR
- 퍼셉트론을 여러 개 사용하여 약점을 극복
- 퍼셉트론 1개로는 XOR문제 최대 75%의 정확도



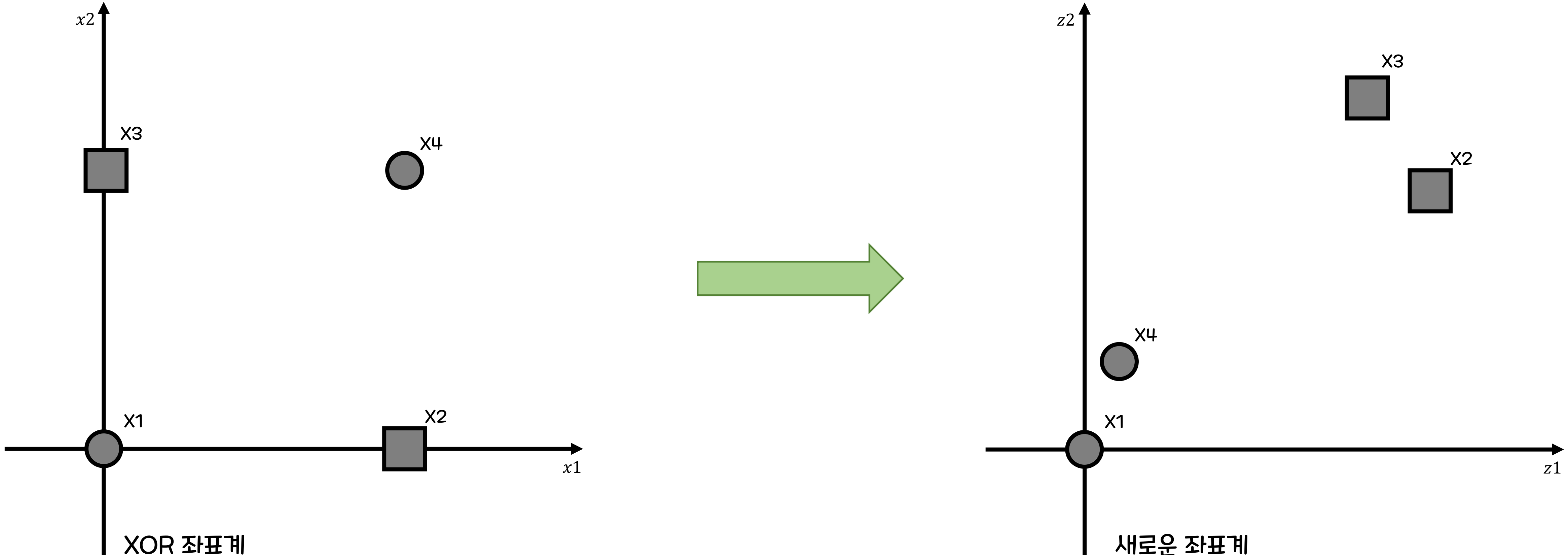
XOR 좌표계

다층 퍼셉트론

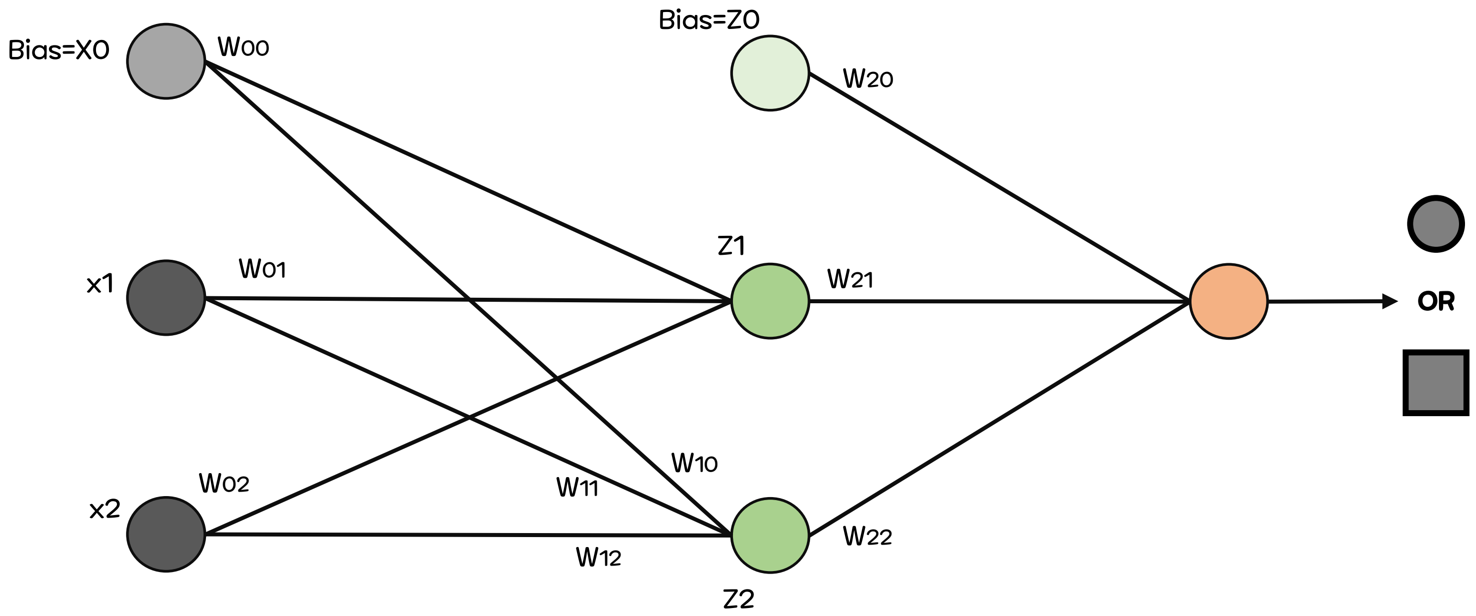


다층 퍼셉트론

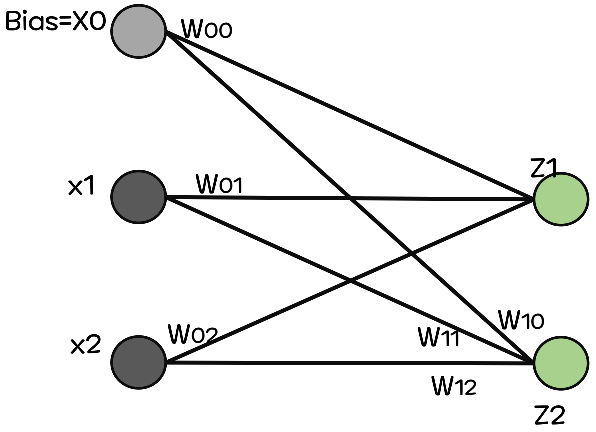
- 여러 개의 퍼셉트론을 가정
- 원래의 특징공간을 분류하는데 더욱 유리한 특징 곤간으로 변환
- 새로운 특징공간은 선형분리에 더 적합함



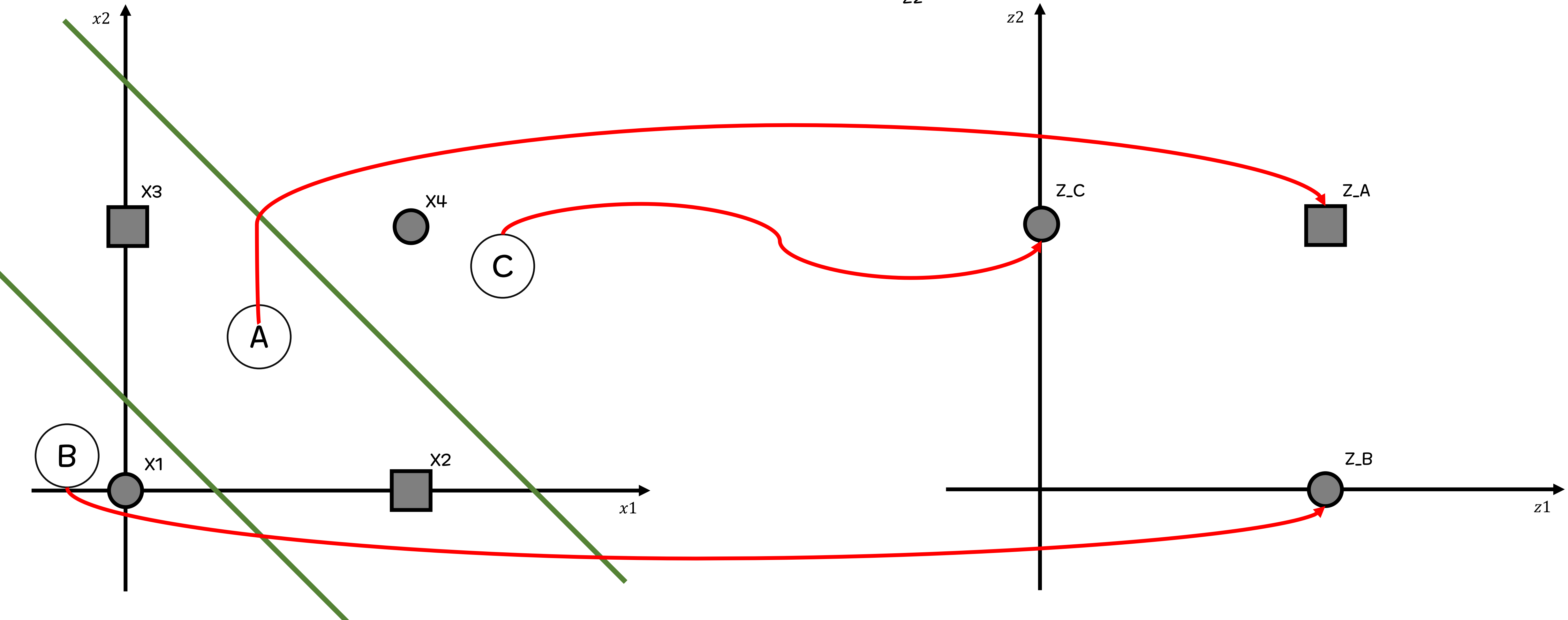
다층 퍼셉트론



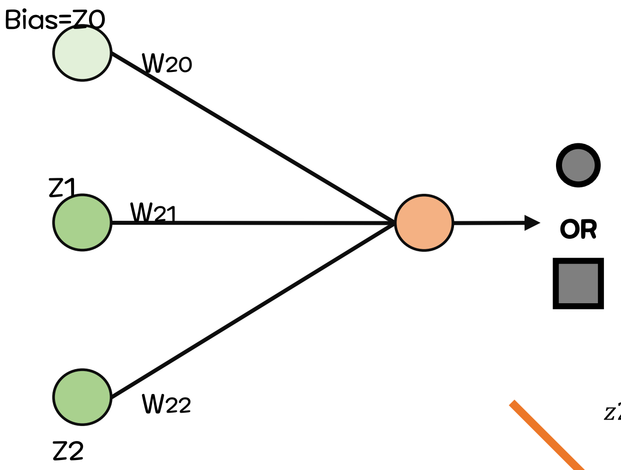
다층 퍼셉트론



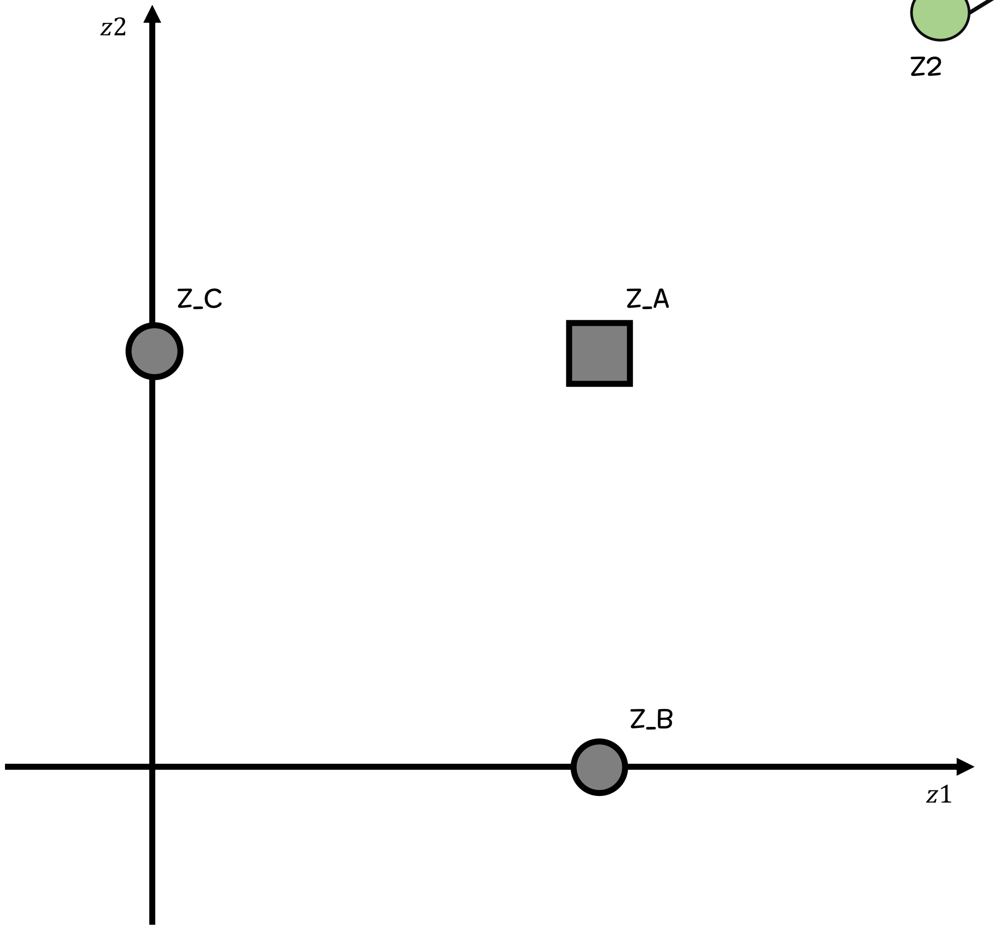
- 원래의 특징공간을 선형 분리하는데 더욱 유리한 공간으로 변환

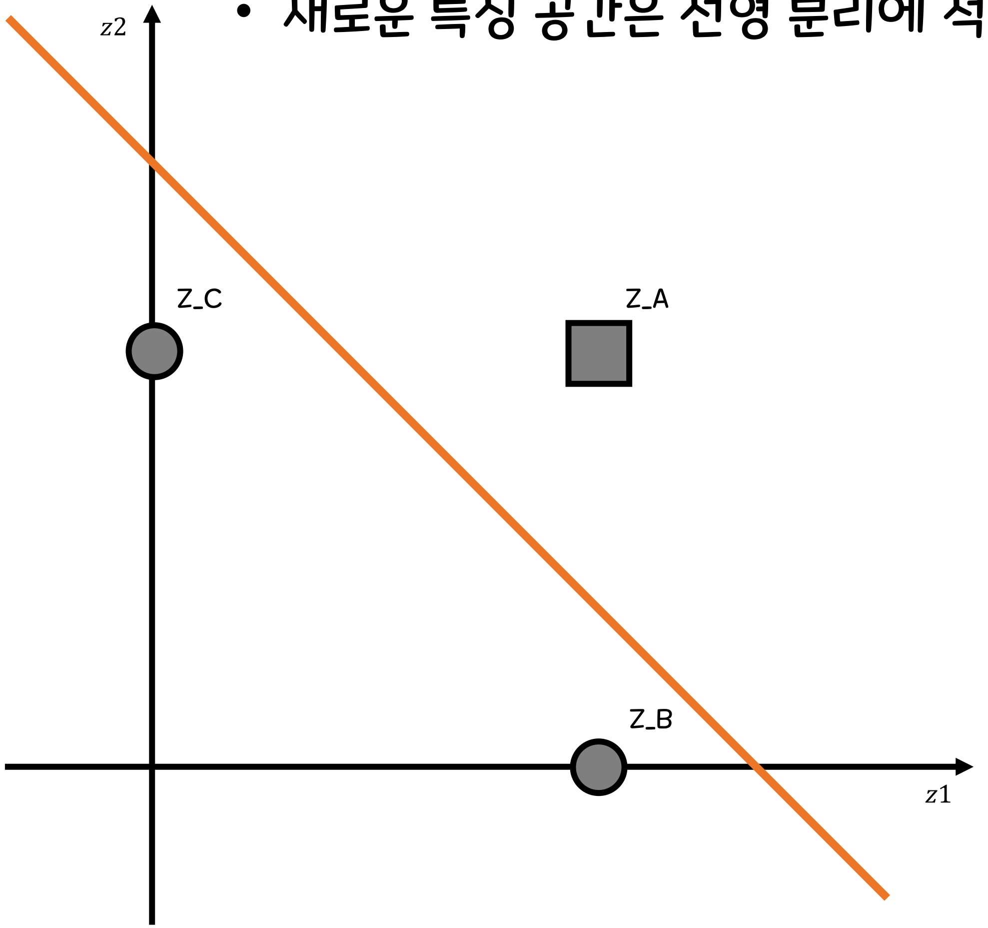


다층 퍼셉트론

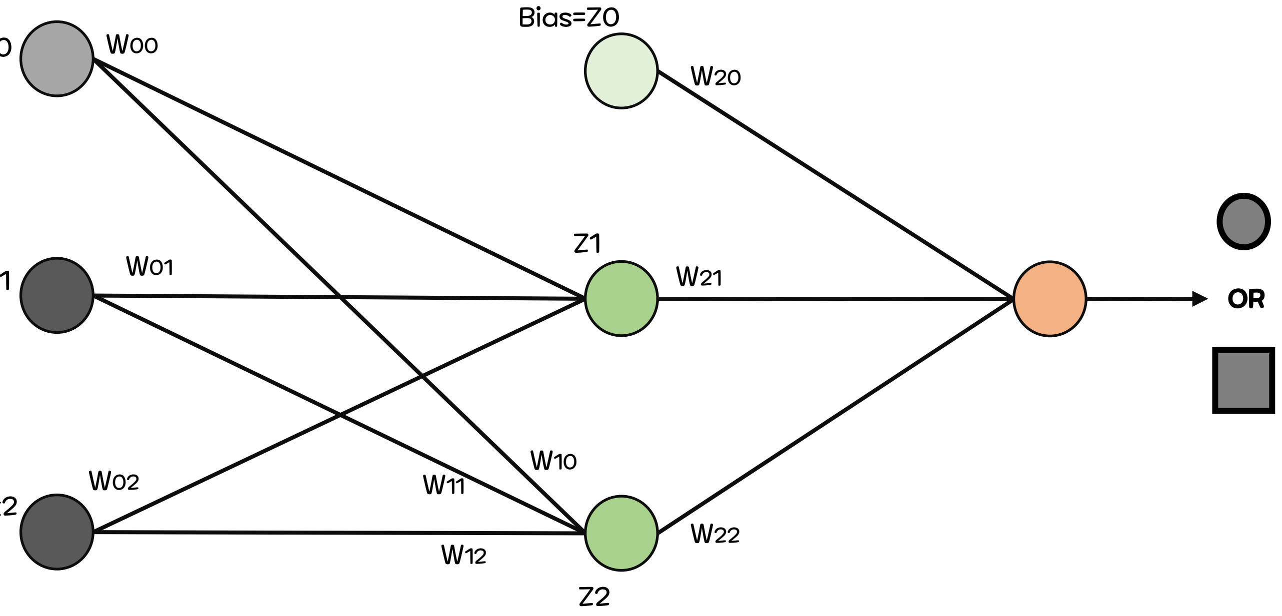


- 원래의 특징공간을 선형 분리하는데 더욱 유리한 공간으로 변환
- 새로운 특징 공간은 선형 분리에 적합





다층 퍼셉트론



- $Hidden_Z = x_0 * w_{00} + x_1 * w_{01} + x_2 * w_{02} + x_0 * w_{10} + x_1 * w_{11} + x_2 * w_{12}$
- $Hidden_Z = X * W_1^T$

여기서 $X = [X_0, X_1, X_2], W_1 = \begin{matrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \end{matrix}$

- $Output = z_0 * w_{20} + z_1 * w_{21} + z_2 * w_{22}$
- $Output = Z * W_2^T$

여기서 $Z = [z_0, z_1, z_2], W_2 = [w_{20}, w_{21}, w_{22}]$

- $\tau(x) =$ ■ if x 〉= 0 else ●
- $\tau(h) =$ 1 if h 〉= 0 else −1

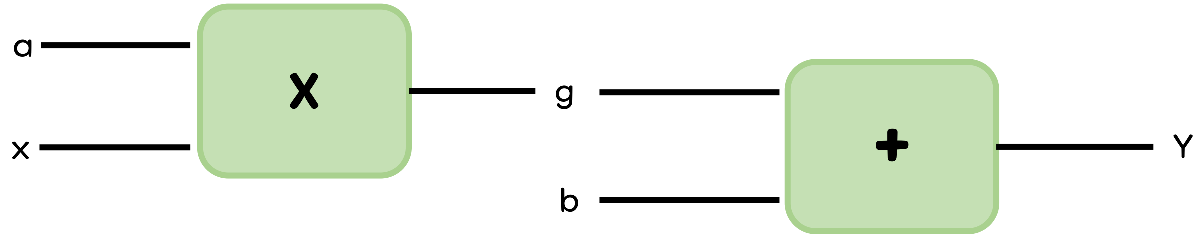
(다층)퍼셉트론 한계

- 층이 거듭될 수록(깊어질 수록) 학습효율이 떨어짐
- 이러한 한계를 극복한 학습 알고리즘인 “오차(오류) 역전파” 알고리즘이 탄생
- 오차 역전파(Error BackPropagation)

(다층)퍼셉트론 한계

- 층이 거듭될 수록(깊어질 수록) 학습이 어려워짐
  - 미분값을 구해야 하는데, 계산도 힘들뿐만 아니라 당시 컴퓨팅파워도 부족했음
  - 또한 깊은 층에 대한 학습 알고리즘이 없었다.
- 이러한 한계를 극복한 학습 알고리즘인 “오차(오류) 역전파” 알고리즘이 탄생
- 오차 역전파(Error BackPropagation)
  - 가중치를 갱신하기 위해선 오차값에 대한 가중치의 미분값(기울기)가 필요하다.
    - $w_{new} = w_{old} - \alpha \frac{\Delta Error}{\Delta w}$
  - 오차 값을 출력층으로 부터 입력층까지 흘려보내어 가중치에 대한 미분값($\frac{\Delta Error}{\Delta w}$)을 계산함

오차 역전파 알고리즘



먼저 Y = a*x + b를 예시로

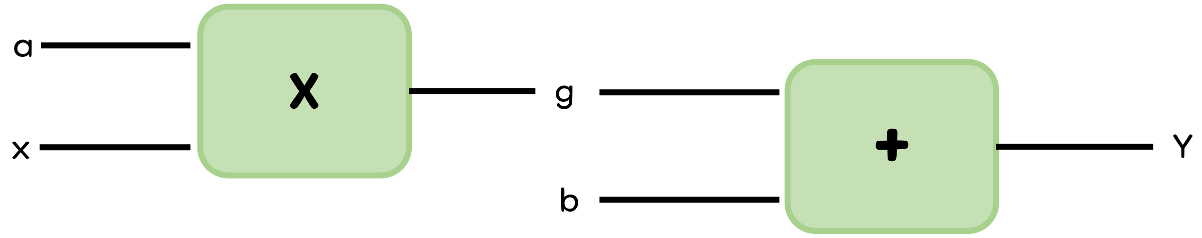
$$y = ax + b$$
$$g = ax$$
$$y = g + b$$

a, x, b가 Y에 미치는 영향력은?

미분을 통해 알 수 있을 것

오차 역전파 알고리즘

$$y = ax + b$$
$$g = ax$$
$$y = g + b$$

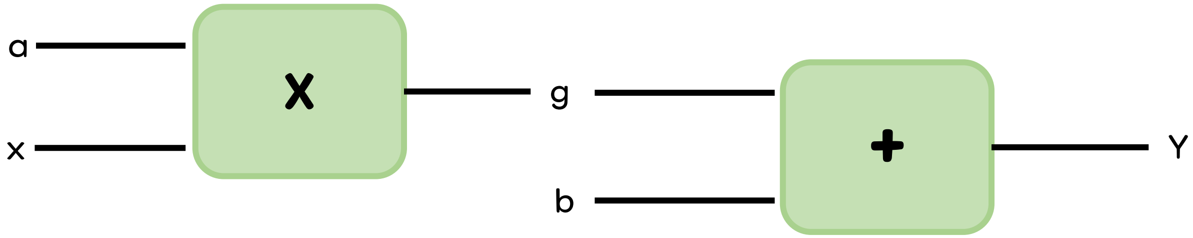


a가 Y에 미치는 영향력은? $^{\Delta Y}/_{\Delta a}$
x가 Y에 미치는 영향력은? $^{\Delta Y}/_{\Delta x}$
b가 Y에 미치는 영향력은? $^{\Delta Y}/_{\Delta b}$

오차 역전파 알고리즘

$y = ax + b$
$g = ax$
$y = g + b$

b가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta b}$



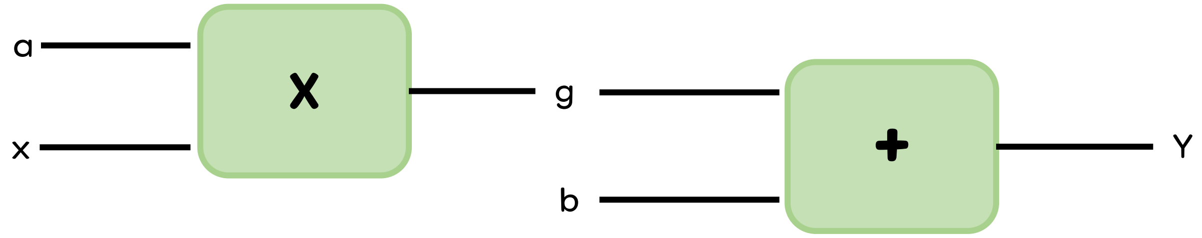
${}^{\Delta Y}/_{\Delta b}$는 y를 b에 대해 미분한 값

$${}^{\Delta Y}/_{\Delta b} = {}^{\Delta(g+b)}/_{\Delta b} = 1$$

오차 역전파 알고리즘

$$y = ax + b$$
$$g = ax$$
$$y = g + b$$

a가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta a}$



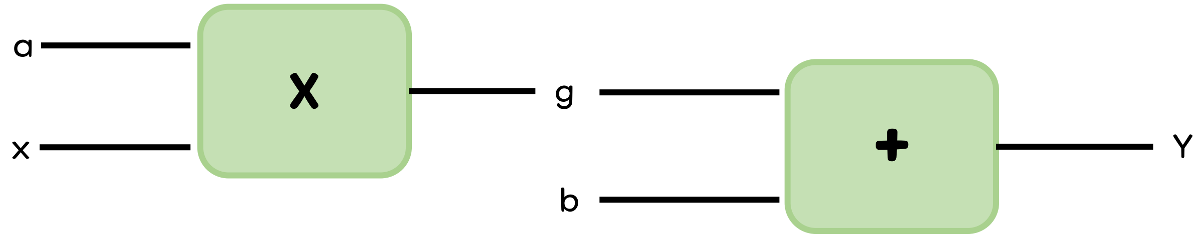
${}^{\Delta Y}/_{\Delta a}$는 y를 a에 대해 미분한 값

$${}^{\Delta Y}/_{\Delta a} = \frac{\Delta g}{\Delta a} * \frac{\Delta Y}{\Delta g} = \frac{\Delta(ax)}{\Delta a} * \frac{\Delta(g+b)}{\Delta g} = x * 1 = x$$

오차 역전파 알고리즘

$y = ax + b$
$g = ax$
$y = g + b$

x가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta x}$

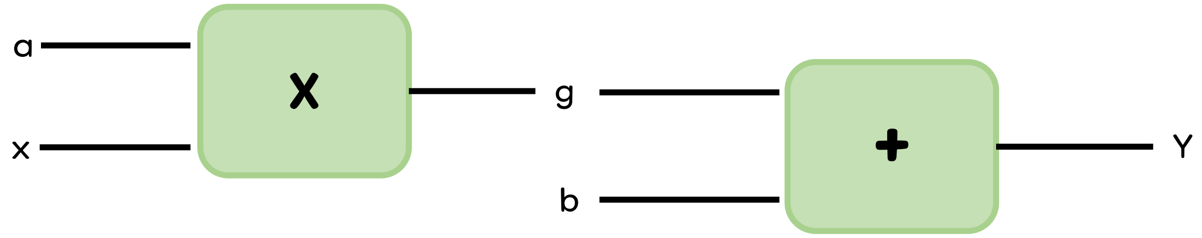


${}^{\Delta Y}/_{\Delta x}$는 y를 x에 대해 미분한 값

$$ {}^{\Delta Y}/_{\Delta x} = \frac{\Delta g}{\Delta x} * \frac{\Delta Y}{\Delta g} = \frac{\Delta(ax)}{\Delta x} * \frac{\Delta(g+b)}{\Delta g} = a * 1 = a $$

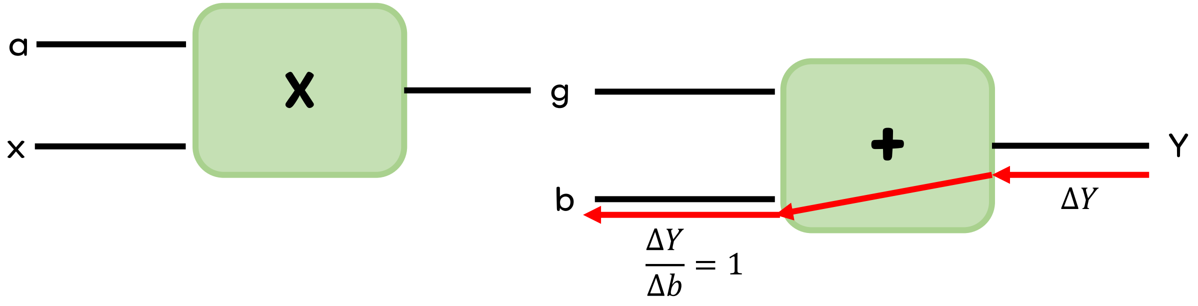
오차 역전파 알고리즘

$$y = ax + b$$
$$g = ax$$
$$y = g + b$$



a가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta a} = x$
x가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta x} = a$
b가 Y에 미치는 영향력은? ${}^{\Delta Y}/_{\Delta b} = 1$

오차 역전파 알고리즘

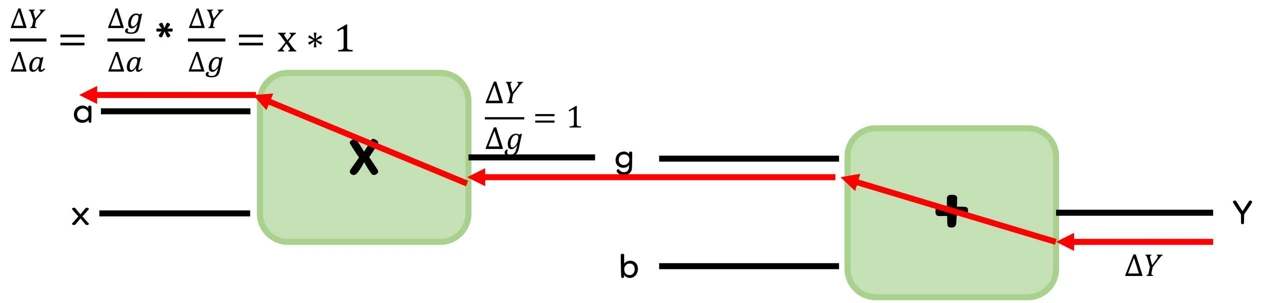


$$y = ax + b$$
$$g = ax$$
$$y = g + b$$

선형 회귀 문제에서 구해야 하는 것은 **a와 b**이다.

실제값(Label)과 예측값(Predict)의 오차(Error)를 역방향으로 흘려보내서 미분값을 구하여 경사하강에 필요한 값을 구함

오차 역전파 알고리즘



$y = ax + b$
$g = ax$
$y = g + b$

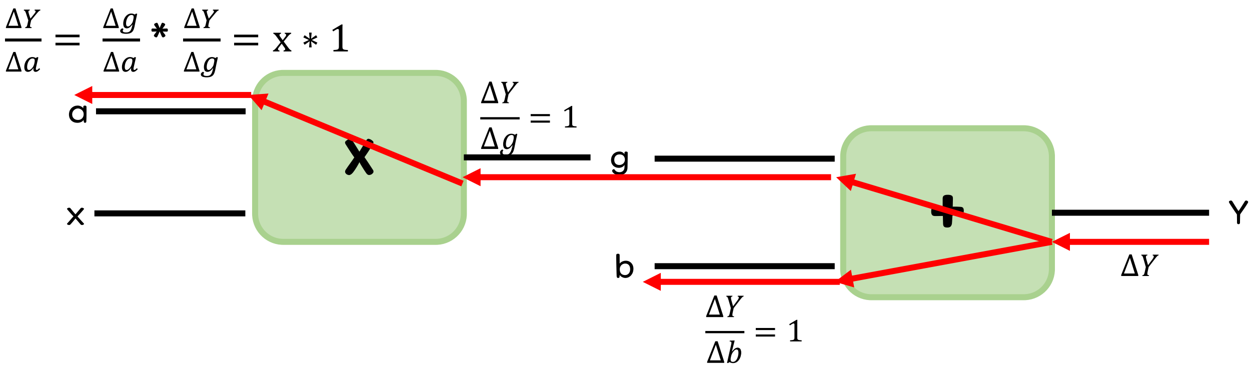
선형 회귀 문제에서 구해야 하는 것은 **a와 b**이다.

실제값(Label)과 예측값(Predict)의 오차(Error)를 역방향으로 흘려보내서 미분값을 구하여 경사하강에 필요한 값을 구함

a는 g로부터 들어온 값에 대한 영향력을 구하는 것으로 볼 수 있음

즉, y에대한 g의 영향력에 대한 a의 영향력(?)

오차 역전파 알고리즘



$$a_{new} = a_{old} - \alpha \frac{\Delta Y}{\Delta a} = a_{old} - \alpha * x$$

$$b_{new} = b_{old} - \alpha \frac{\Delta Y}{\Delta b} = b_{old} - \alpha * 1$$

$y = ax + b$
$g = ax$
$y = g + b$

선형 회귀 문제에서 구해야 하는 것은 **a와 b**이다.

실제값(Label)과 예측값(Predict)의 오차(Error)를 역방향으로 흘려보내서 미분값을 구하여 경사하강에 필요한 값을 구함

a는 g로부터 들어온 값에 대한 영향력을 구하는 것으로 볼 수 있음

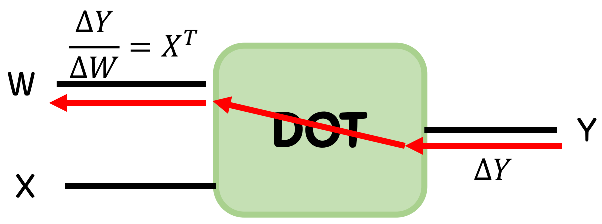
즉, y에대한 g의 영향력에 대한 a의 영향력(?)

오차 역전파 알고리즘

쉽게 이야기 하자면,
곱셈은 자신과 곱해지는 대상의 값이 미분값이고
덧셈은 단순히 1이 미분값이다.

반면, 행렬곱(내적 = Dot)은 어떤 값을 가지냐면,
내적이 되는 상대행렬의 전치값이다.

$$W_{new} = W_{old} - \alpha \frac{\Delta Y}{\Delta W} = W_{old} - \alpha * X^T$$



딥러닝의 부활

- 오차 역전파로인해 깊은-다층 퍼셉트론(DMLP)가 주목받기 시작하였다.