Ch02.

간단한 분류 알고리즘 구현

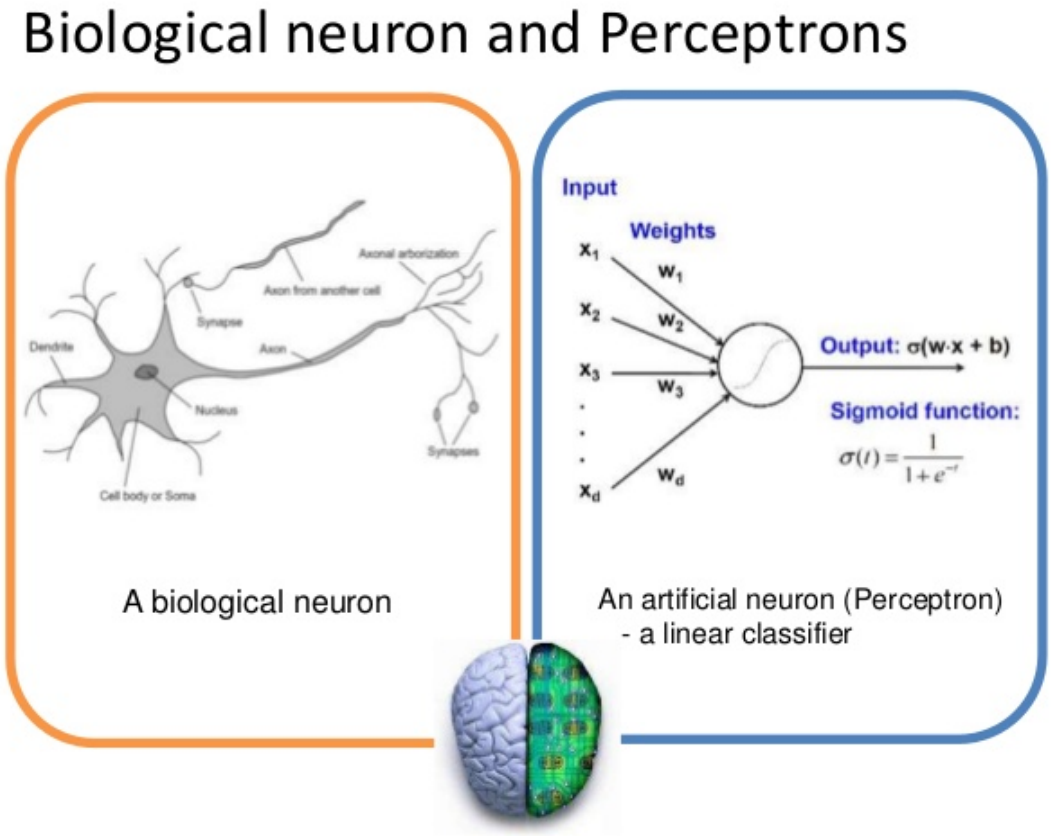
202STG18 이재빈

CONTENTS

01 ,
인공 뉴런 : 초기 머신 러닝의 간단한 역사

1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- Perceptron

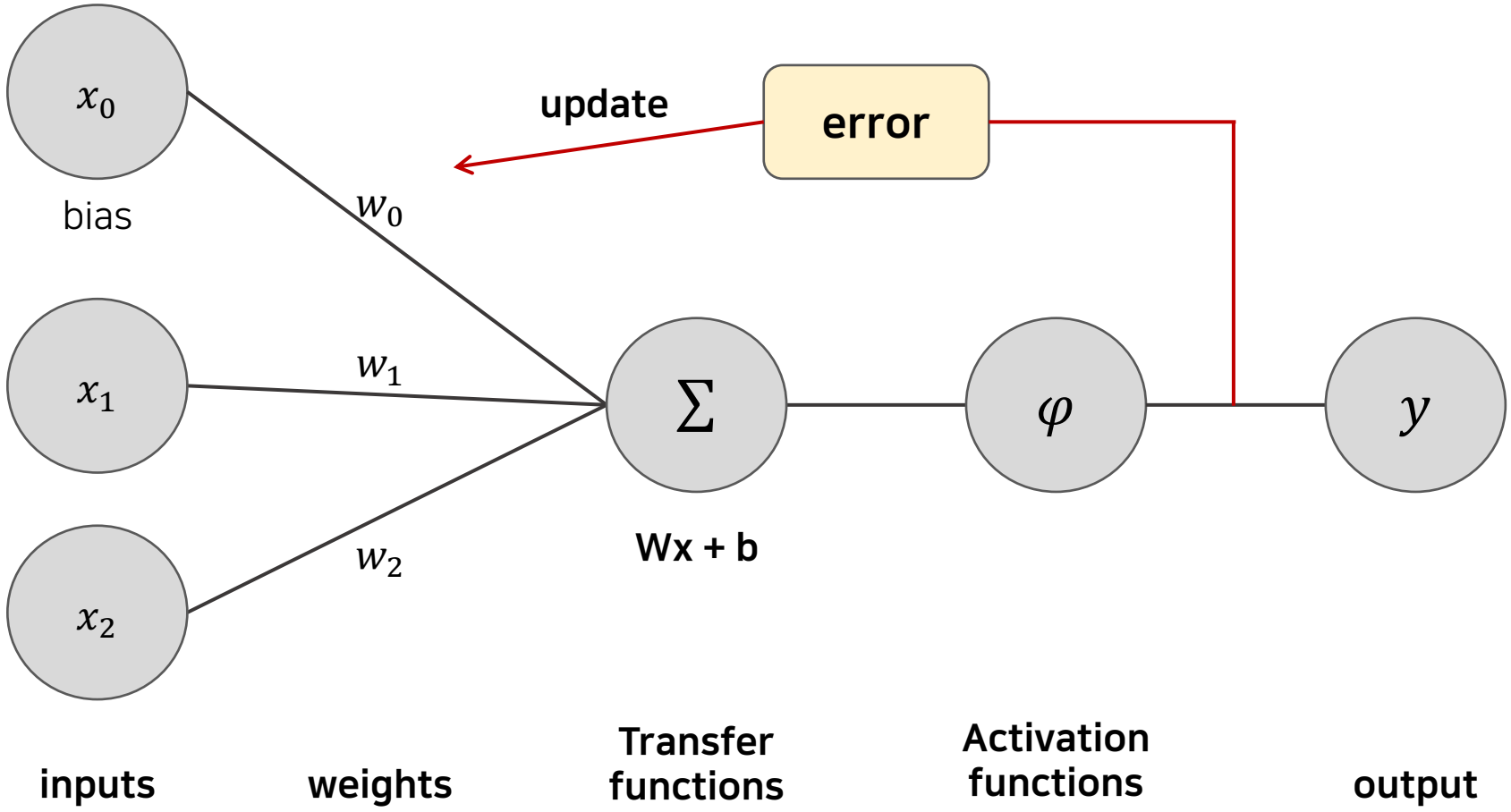


사람 신경망	인공 신경망
세포체	노드
수상돌기	입력
축삭돌기	출력
시냅스	가중치

Neuron의 정보 처리 이해를 모방한 인공 신경망 연구

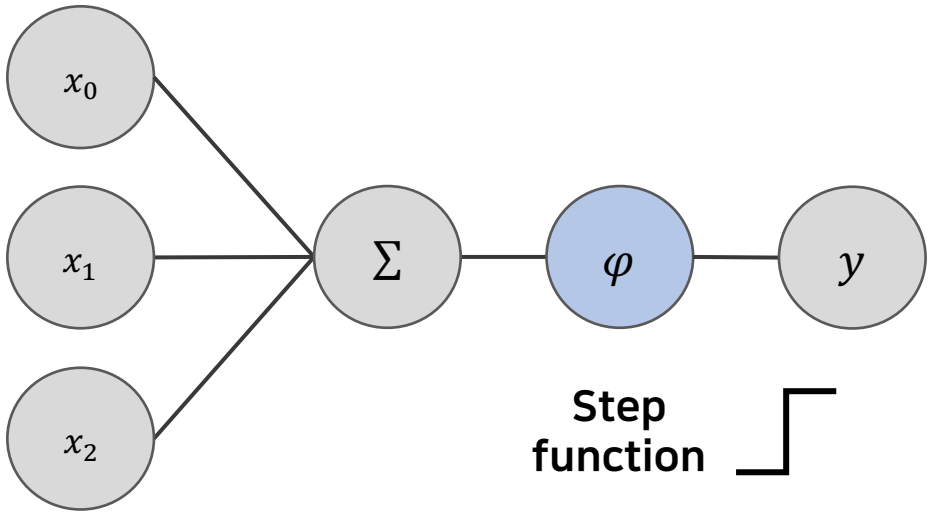
1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- Perceptron



1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- 로젠블란트 초기 퍼셉트론 학습 규칙

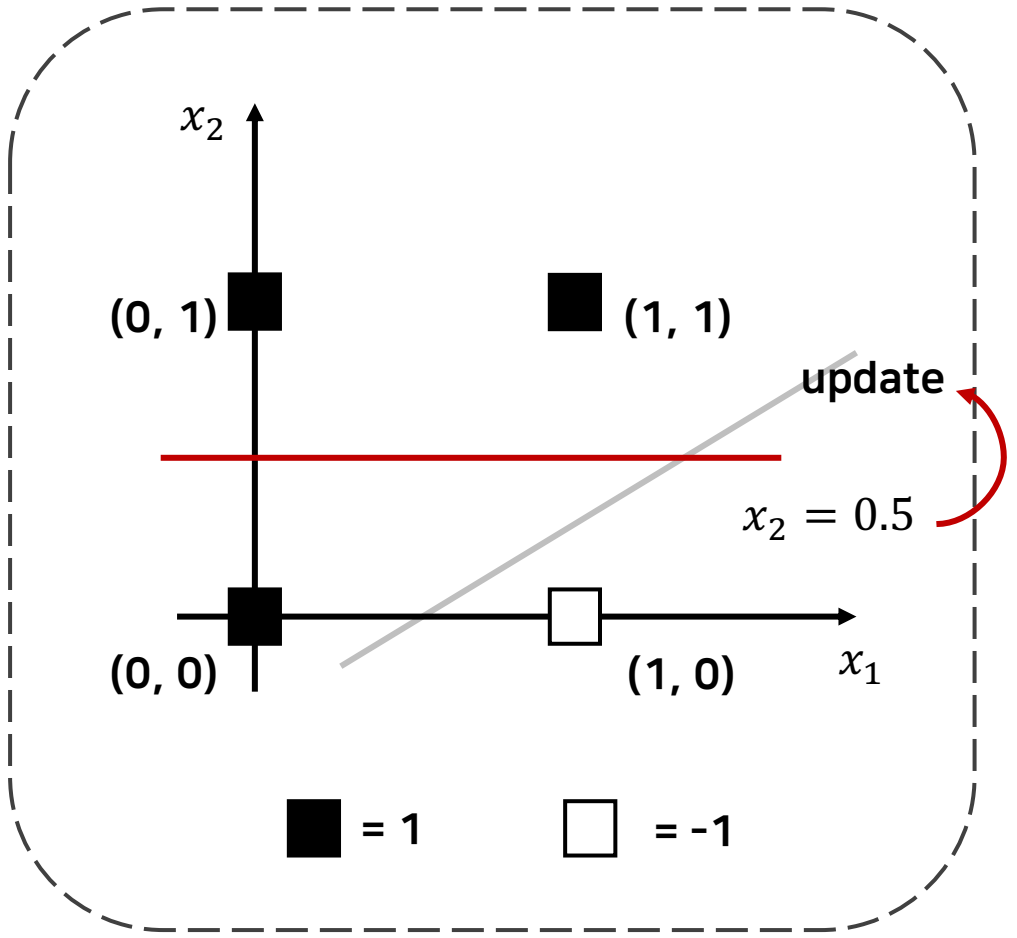


로젠블란트 초기 퍼셉트론 학습 규칙

1. 가중치를 0 또는 랜덤한 작은 값으로 초기화 합니다.
2. 각 훈련 샘플 $x^{(i)}$ 에서 다음 작업을 합니다.
   a. 출력 값 $\hat{y}$ 을 계산합니다.
   b. 가중치를 업데이트 합니다.

1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- 로젠블란트 초기 퍼셉트론 학습 규칙



$$-0.5 + 0 \cdot x_1 + 1 \cdot x_2 = 0$$

■ (0, 0) : -0.5 → 1 , 오분류

■ (0, 1) : 0.5 → 1

■ (1, 1) : 0.5 → 1

□ (1, 0) : 0.5 → -1

$$\phi(z) = \begin{cases} 1 & if\ z > \theta \\ -1 & otherwise \end{cases}$$

1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- 로젠블란트 초기 퍼셉트론 학습 규칙

■ (0, 0) : **오분류**

$$x_0 = 1$$
$$x_1 = 0$$
$$x_2 = 0$$

$$w_0 = -0.5$$
$$w_1 = 0$$
$$w_2 = 1$$

$$\eta = 0.05$$

$$w_j \leftarrow w_j + \eta(y - o)x_j$$

$$w_0 \leftarrow w_0 + 0.05 \cdot (\overset{\text{실제}}{1} - (\overset{\text{예측}}{-1})) \cdot 1$$

$$w_1 \leftarrow w_1 + 0.05 \cdot (1 - (-1)) \cdot 0$$

$$w_2 \leftarrow w_2 + 0.05 \cdot (1 - (-1)) \cdot 0$$

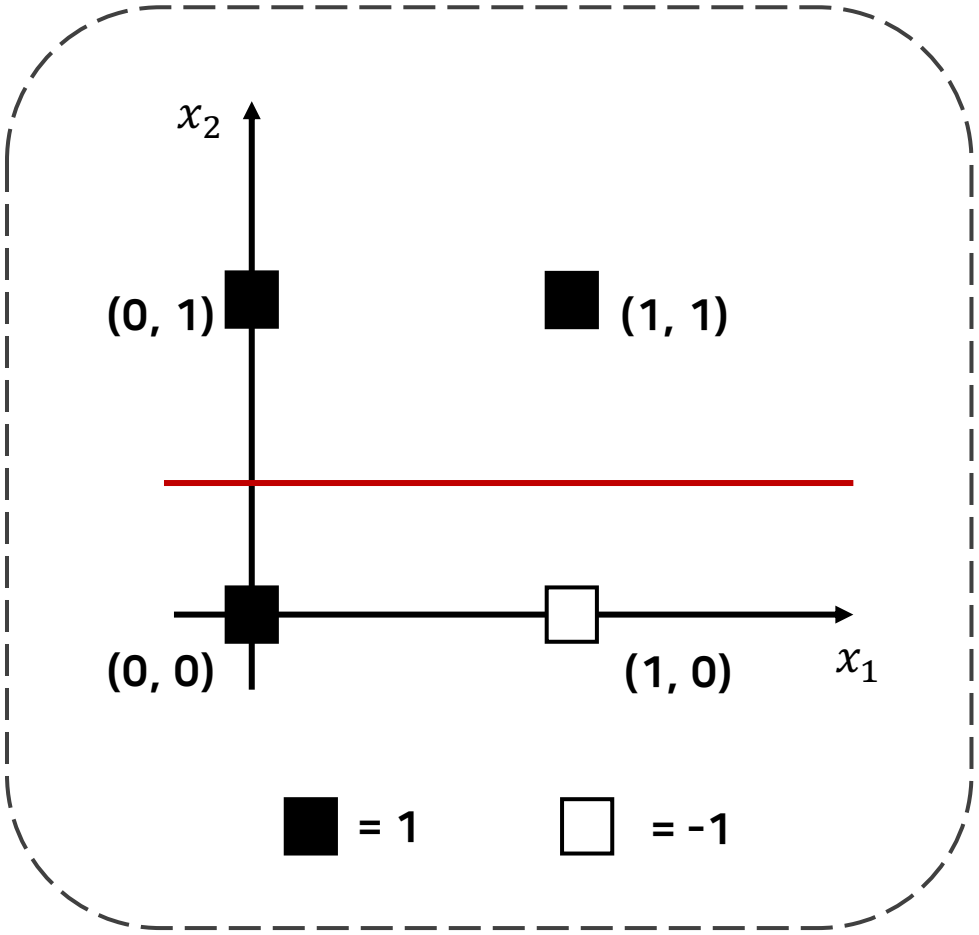
$$w_0 \leftarrow -0.5 + 0.1 = -0.4$$

$$w_1 \leftarrow 0 + 0 = 0$$

$$w_2 \leftarrow 1 + 0 = 1$$

1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- 로젠블란트 초기 퍼셉트론 학습 규칙

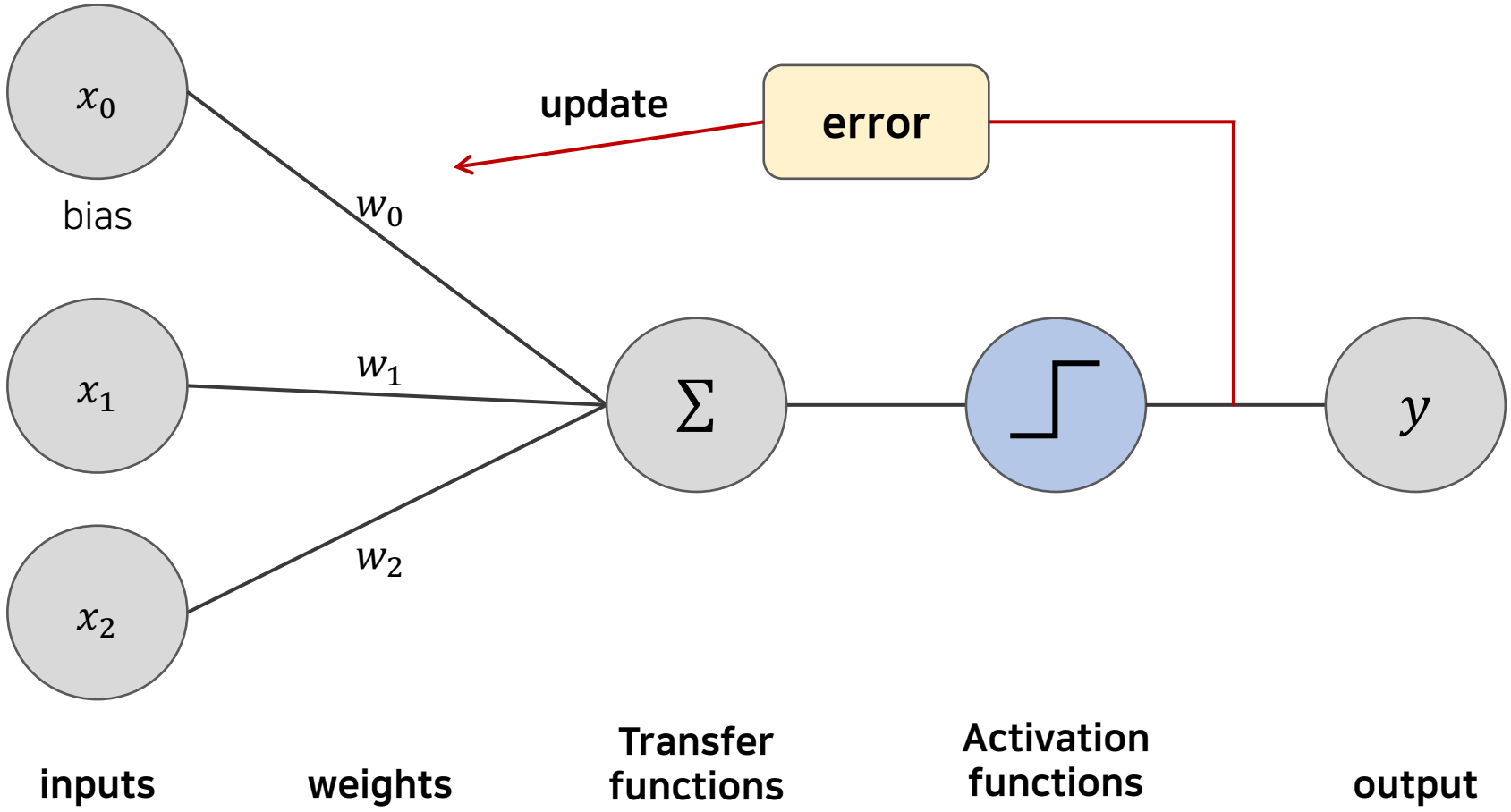


update

$-0.4 + 0 \cdot x_1 + 1 \cdot x_2 = 0$

1. 인공 뉴런 : 초기 머신 러닝의 간단한 역사

- 로젠블란트 초기 퍼셉트론 학습 규칙

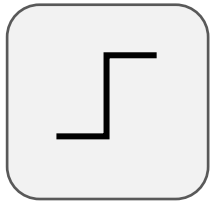


02 ,
파이썬으로 퍼셉트론 학습 알고리즘 구현

03 ,
적응형 선형 뉴런과 학습의 수렴

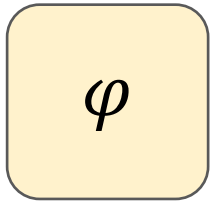
3. 적응형 선형 뉴런과 학습의 수렴

- Rosenblatt vs ADALINE



Rosenblatt

- Step Function
- 실제 Class Label vs 예측 Class Label



ADAptive Linear NEuron

- Continuous Activate Function
- 실제 Class Label vs 출력 값

Activation functions

3. 적응형 선형 뉴런과 학습의 수렴

- ADALINE

$$J(W) = \frac{1}{2}\sum(y^{(i)} - (w^T x^{(i)}))^2$$

목적함수 (SSE) : 실제값과 예측값의 차이 최소화

$$\frac{\partial J}{\partial w_j} = -\sum\left(y^{(i)} - \left(w^T x^{(i)}\right)\right)x_j^{(i)}$$

목적함수 미분

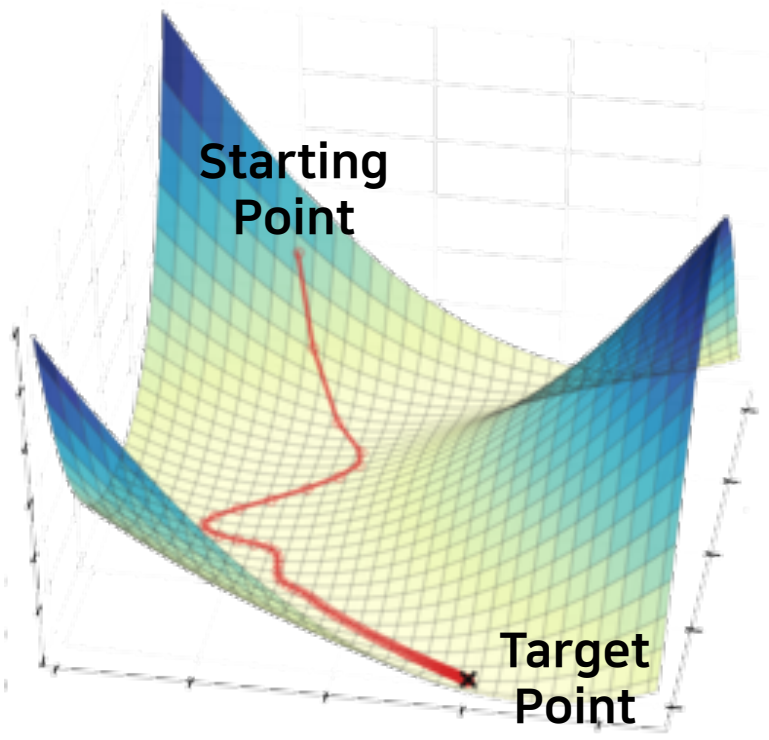
$$\Delta w_j = -\eta\,\frac{\partial J}{\partial w_j} = \eta\sum\left(y^{(i)} - \left(w^T x^{(i)}\right)\right)x_j^{(i)}$$

가중치 update : learning rate * 목적함수 미분

$$w := w + \Delta w$$

3. 적응형 선형 뉴런과 학습의 수렴

- Gradient Descent



Gradient Descent (경사하강법)

최적 해 (최솟값)을 찾을 때 까지 언덕을 내려온다 !

ex. 등산을 하고 이제 하산을 하는 상황,

현재 내 위치로부터 한 발자국을 내딛었을 때

어느 정도 높이 차이가 나는지에 대한 정보 밖에 없다고 가정했을 때,

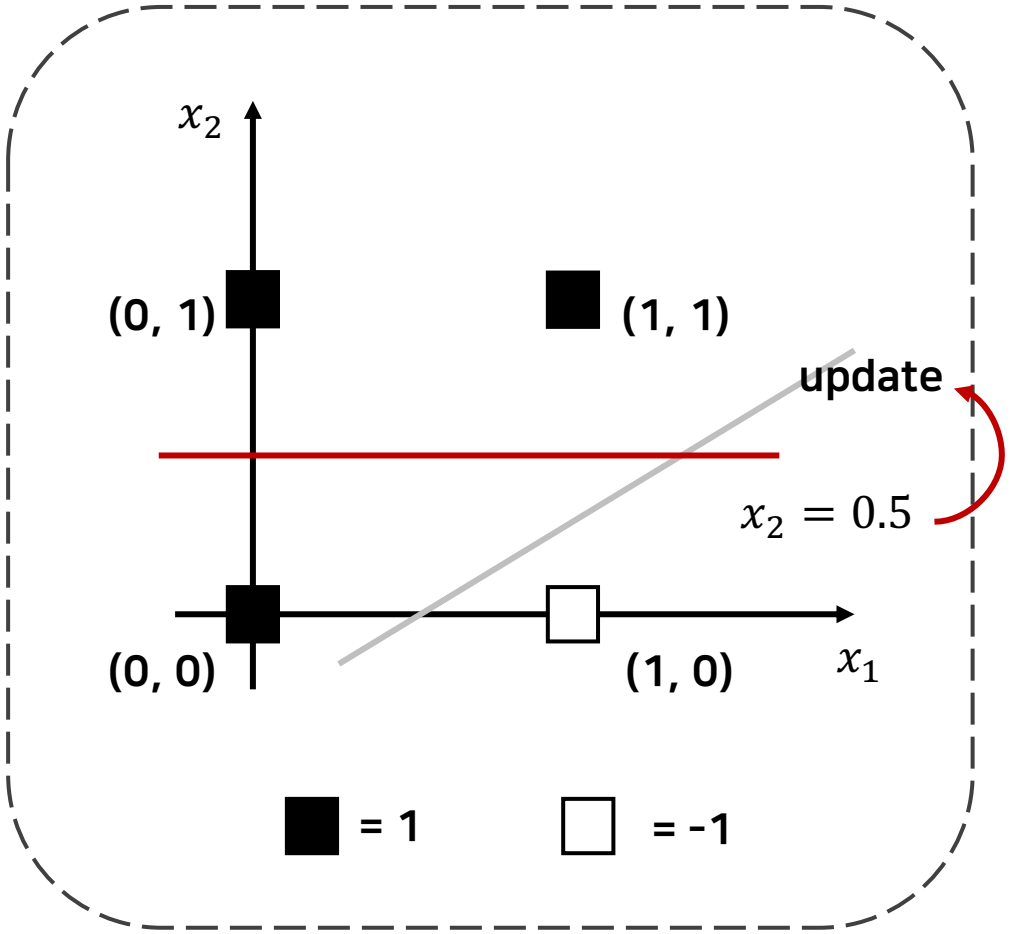
산 아래로 내려갈 수 있는 가장 간단한 방법은

한 발자국을 내딛었을 때 가장 높이 차이가 많이 나는 방향,

즉, **경사가 가장 가파른 방향으로 이동**하는 것

3. 적응형 선형 뉴런과 학습의 수렴

- ADALINE Gradient Descent



$$-0.5 + 0 \cdot x_1 + 1 \cdot x_2 = 0\,, \eta = 0.05$$

예측

- (0, 0) : -1
- (0, 1) : 1
- (1, 1) : 1
- (1, 0) : -1

실제

x_0	x_1	x_2	y	$w^T x$	$y - w^T x$
1	0	0	1	-0.5	1.5
1	0	1	1	0.5	0.5
1	1	1	1	0.5	0.5
1	1	0	-1	-0.5	-0.5

3. 적응형 선형 뉴런과 학습의 수렴

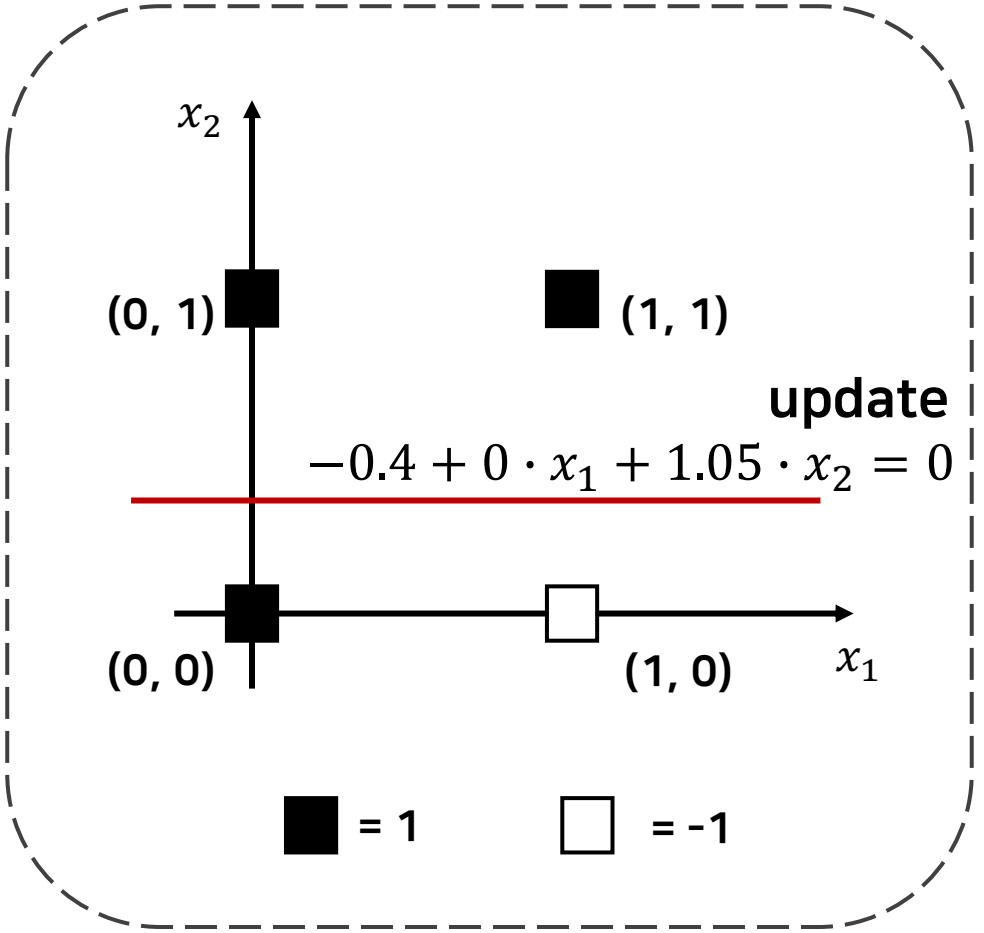
- ADALINE Gradient Descent

x_0	x_1	x_2	y	w^Tx	$y - w^Tx$
1	0	0	1	-0.5	1.5
1	0	1	1	0.5	0.5
1	1	1	1	0.5	0.5
1	1	0	-1	-0.5	-0.5

$w_0 \leftarrow w_0 + 0.05 \cdot 2$
($\because x_0$ 는 다 1 이므로)

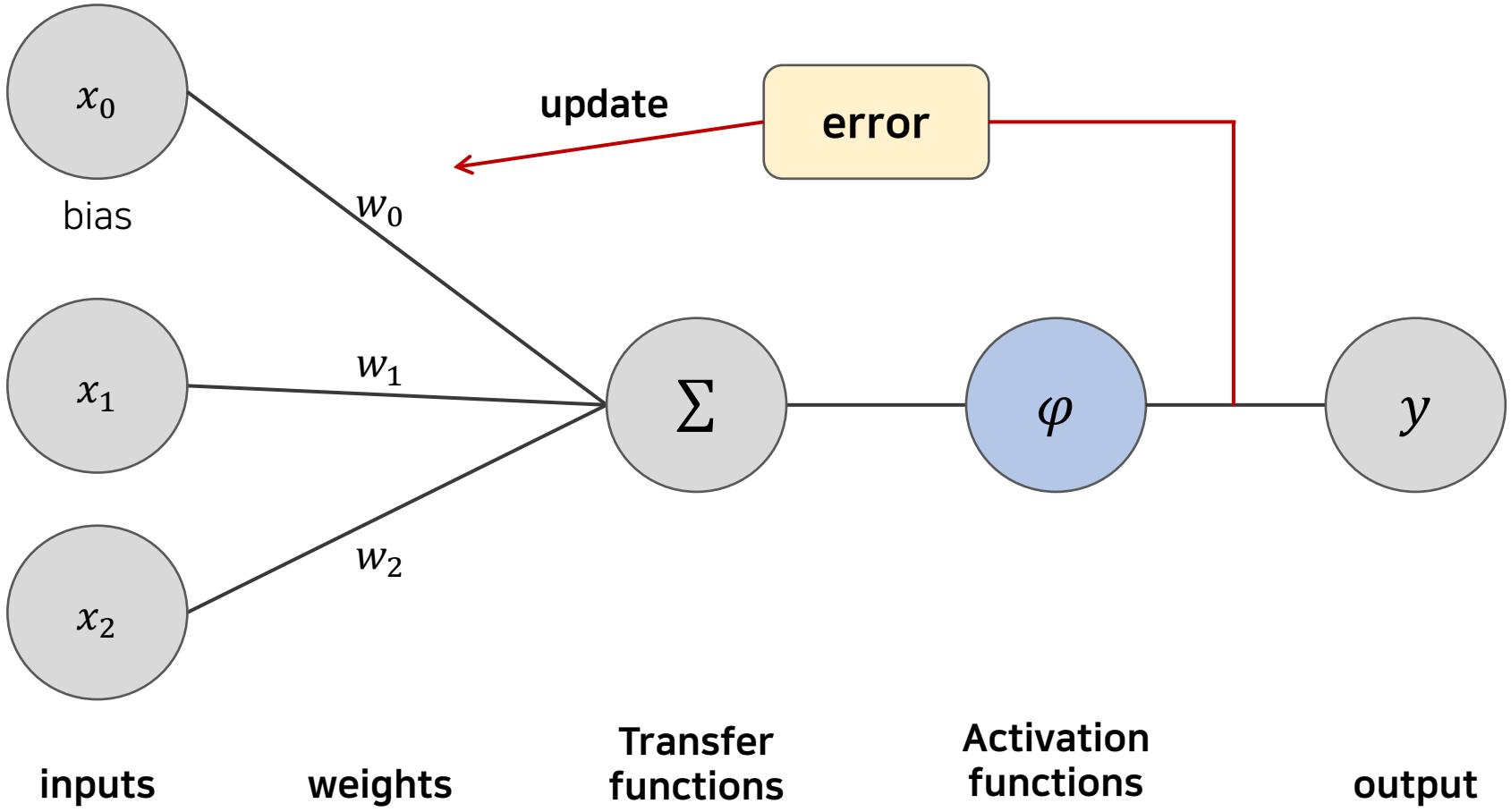
$w_1 \leftarrow w_1 + 0.05 \cdot 0$
($\because x_1 = 1$ 인 데이터에 대해 다 더한 값)

$w_2 \leftarrow w_2 + 0.05 \cdot 1$
($\because x_2 = 1$ 인 데이터에 대해 다 더한 값)



3. 적응형 선형 뉴런과 학습의 수렴

- ADALINE



3. 적응형 선형 뉴런과 학습의 수렴
- Scaling

- Scale 의 범위가 너무 크면 Noise 데이터가 생성되거나 Overfitting 이 될 가능성이 높아짐
- 값이 한 쪽으로 쏠리기 때문에, 활성화 함수(Activation Function)을 거치는 의미가 사라짐

Scaling

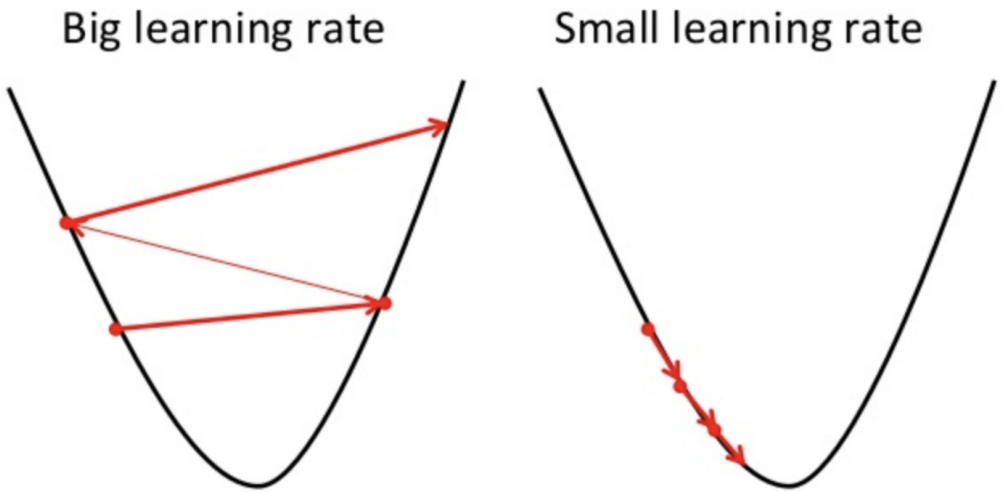
$$x_i = \frac{x_i - \mu}{\sigma}$$

Standardization
mean, variance 고려

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

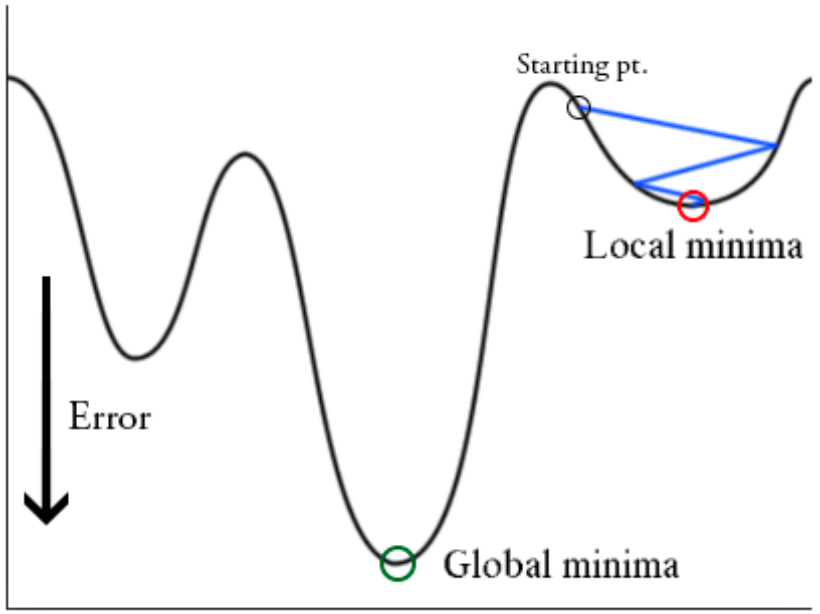
min-max normalization
0 ~ 1 사이의 값

3. 적응형 선형 뉴런과 학습의 수렴
- Gradient Descent Problem



Overshooting

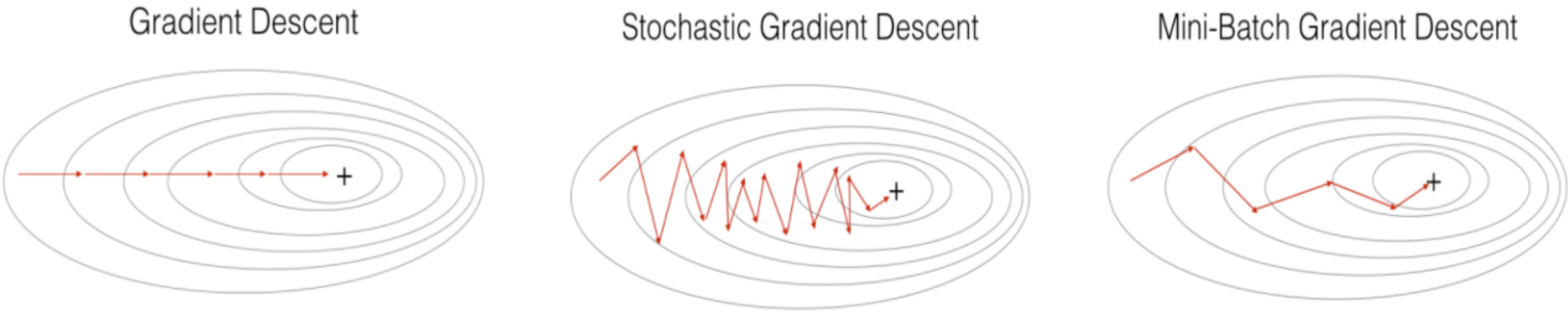
Learn
too slow



- Learning Rate 에 따라 발산할 가능성이 존재하며, 너무 작은 Learning Rate 를 택할 시에 수렴이 늦어짐
- Initial Point 를 어디에서 시작하는지에 따라 수렴 지점이 달라질 수 있음

3. 적응형 선형 뉴런과 학습의 수렴

- Stochastic Gradient Descent (SGD)



- 한 step 마다 모든 데이터들에 대한 Gradient 를 계산하는 것은 연산량이 많아, 비효율적
- [SGD] 한 번에 한 개만, Random Sampling 을 통해 추출해서, 해당 Point에 대한 Gradient를 계산

- (+) 실제 Gradient 계산하는 것 보다 연산 시간 단축
  (+) Shooting이 일어나기 때문에 Local Minima에 빠질 Risk가 적음
  (-) Global Minima를 찾지 못할 가능성 존재

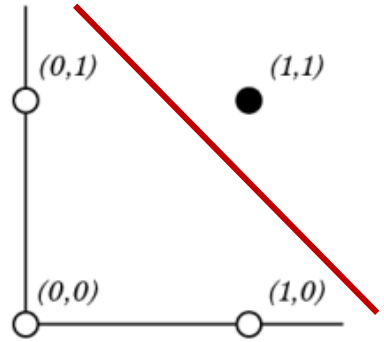
- [Mini Batch Stochastic Descent] 여러 개 Point Sampling 하여 Gradient 계산

3. 적응형 선형 뉴런과 학습의 수렴

- cf. Perceptron XOR

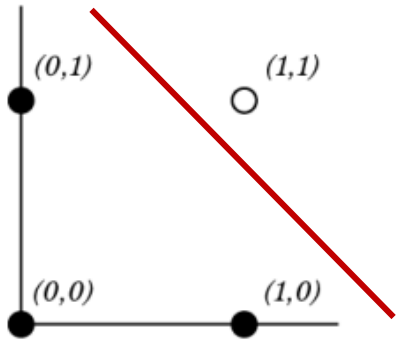
1. AND GATE

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1



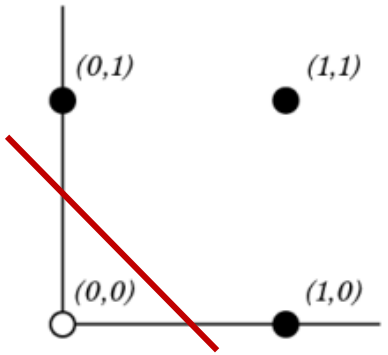
2. NAND GATE

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0



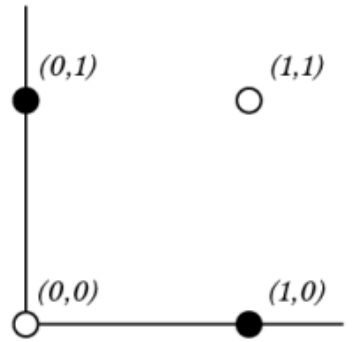
3. OR GATE

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1



4. XOR GATE

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0



XOR Gate 는 하나의 선으로 문제를 풀 수 없음 -> MultiLayer Neural Network 의 등장

감사합니다 😊