퍼셉트론

학습 목표

- 신경망에 대하여 이해한다.
- 신경망의 초기 모델인 퍼셉트론을 이해한다.
- 퍼셉트론의 작동원리를 이해한다.
- 퍼셉트론 학습 알고리즘을 이해한다.
- 퍼셉트론의 한계점을 인식한다.

신경망

- 최근에 많은 인기를 끌고 있는 딥러닝(deep learning)의 시작은 1950년대부터 연구되어 온 인공 신경망(artificial neural network: ANN)이다.
- 인공 신경망은 생물학적인 신경망에서 영감을 받아서 만들어진 컴퓨팅 구조이다.

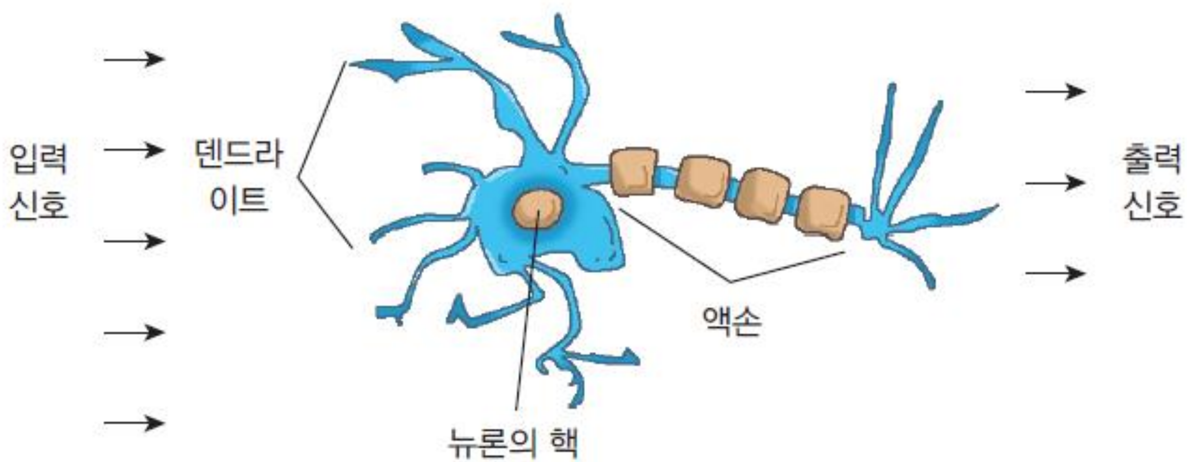
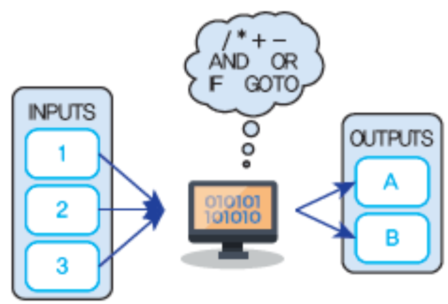


그림 5-1 뉴런의 구조

전통적인 컴퓨터 vs 인공신경망

	기존의 컴퓨터	인간의 두뇌
처리소자의 개수	10^8개의 트랜지스터	10^{10}개의 뉴런
처리소자의 속도	10^{12}Hz	10^2Hz
학습기능	없음	있음
계산 스타일	중앙집중식, 순차적인 처리	분산 병렬 처리
		

신경망의 장점

- 첫 번째는 학습이 가능하다는 점이다. 데이터만 주어지면 신경망은 예제로부터 배울 수 있다.

- 두 번째는 몇 개의 소자가 오동작하더라도 전체적으로는 큰 문제가 발생하지 않는다는 점이다.

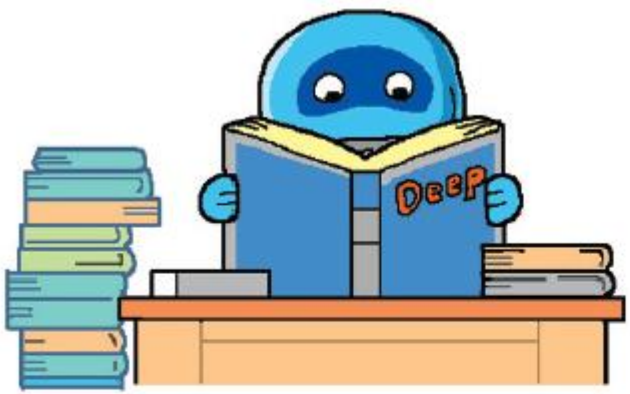


그림 5-3 학습하는 컴퓨터

신경망이 필요한 분야

- 예를 들어 강아지 이미지와 고양이 이미지를 식별하는 작업을 생각해보자. 인간은 쉽게 이미지를 인식하지만 인간도 인식의 메커니즘을 정확히 모르기 때문에 인식 알고리즘을 명시적으로 만드는 것은 아주 어려운 일이다.

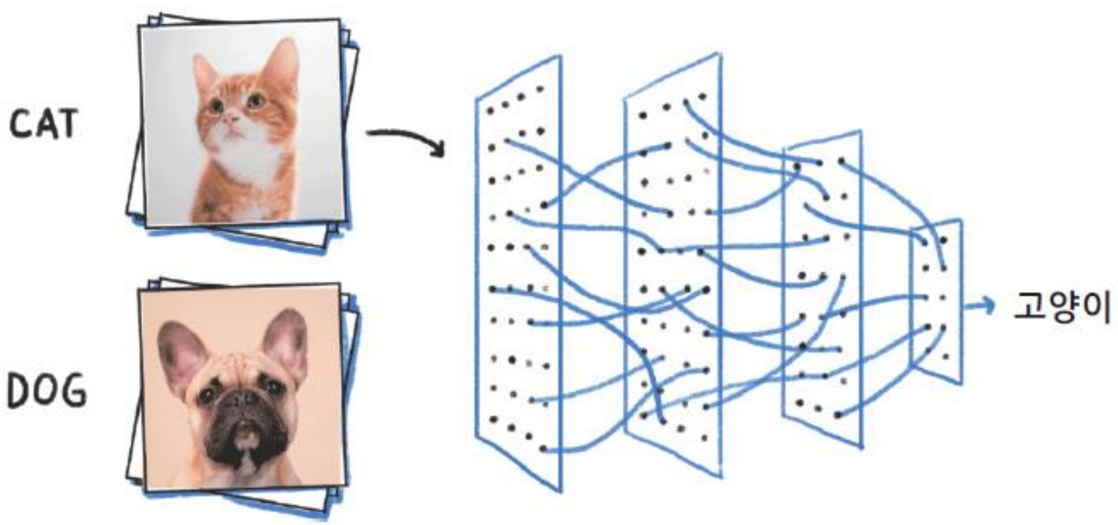


그림 5-4 신경망을 이용한 이미지 인식

퍼셉트론

- 퍼셉트론(perceptron)은 1957년에 로젠블라트(Frank Rosenblatt)가 고안한 인공 신경망이다.

선형회귀 일반식.

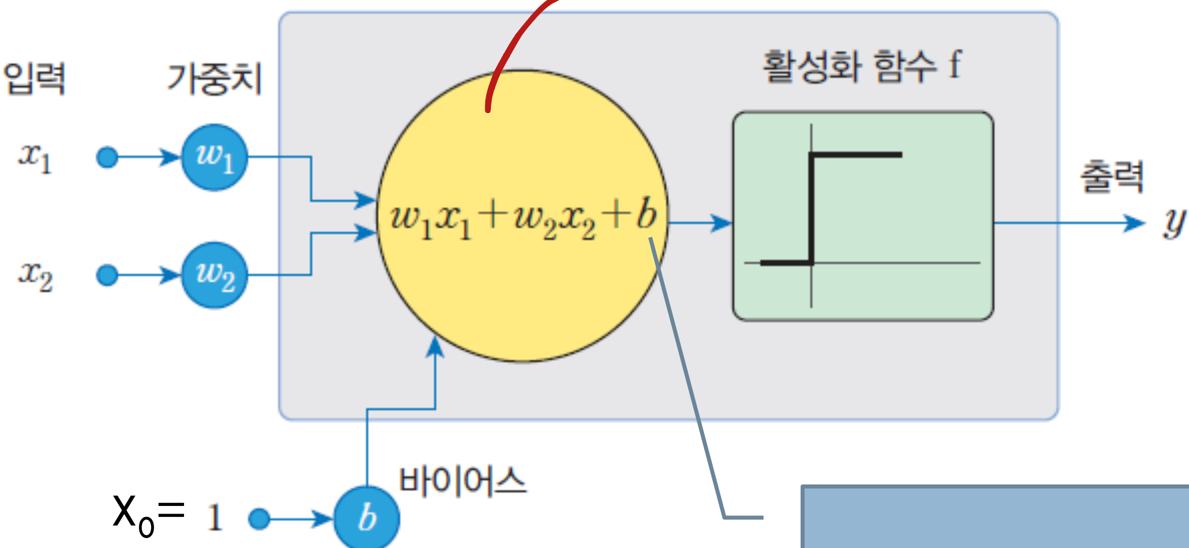


그림 5-5 퍼셉트론에서의 뉴런

$b = -\theta$

퍼셉트론

- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.
- 활성화 함수의 역할은 수도꼭지와 같음

$$
y = \begin{cases} 1 & \text{if } (w_1x_1 + w_2x_2 + b \geq 0) \\ 0 & otherwise \end{cases}
$$

참이면 1

거짓이면 0

퍼셉트론은 논리 연산을 학습할 수 있을까?

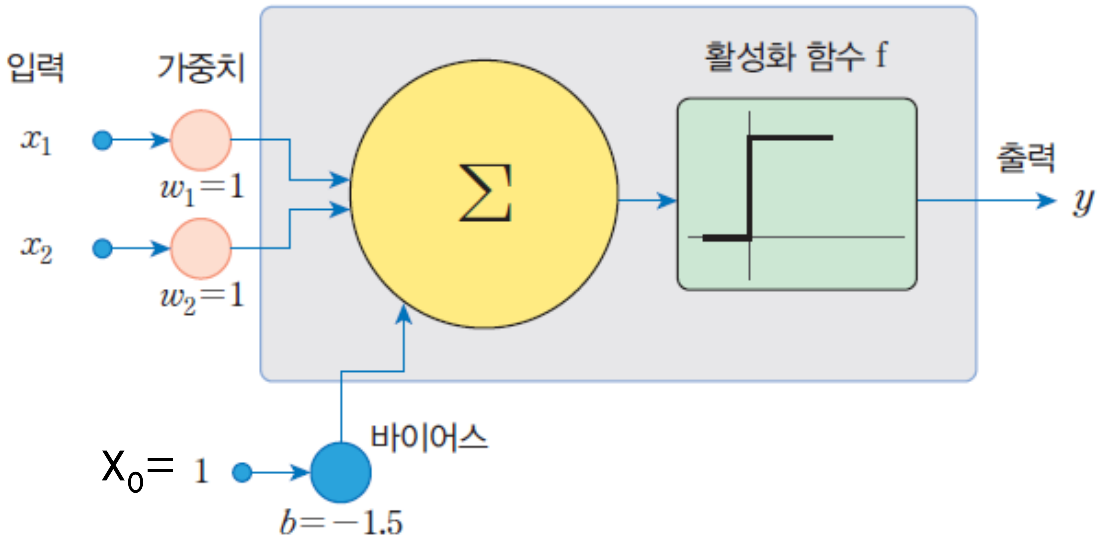


그림 5-6 논리 연산을 하는 퍼셉트론

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

And? Or?

퍼셉트론은 논리 연산을 학습할 수 있을까?

$*W_1 = 1, \ W_2 = 1, \ b = -1.5$

표 5-2 퍼셉트론 출력 계산

x_1	x_2	$w_1x_1 + w_2x_2$	b	y
0	0	1*0+1*0=0	–1.5 <0	0
1	0	1*1+1*0=1	–1.5 <0	0
0	1	1*0+1*1=1	–1.5 <0	0
1	1	1*1+1*1=2	–1.5 >0	1

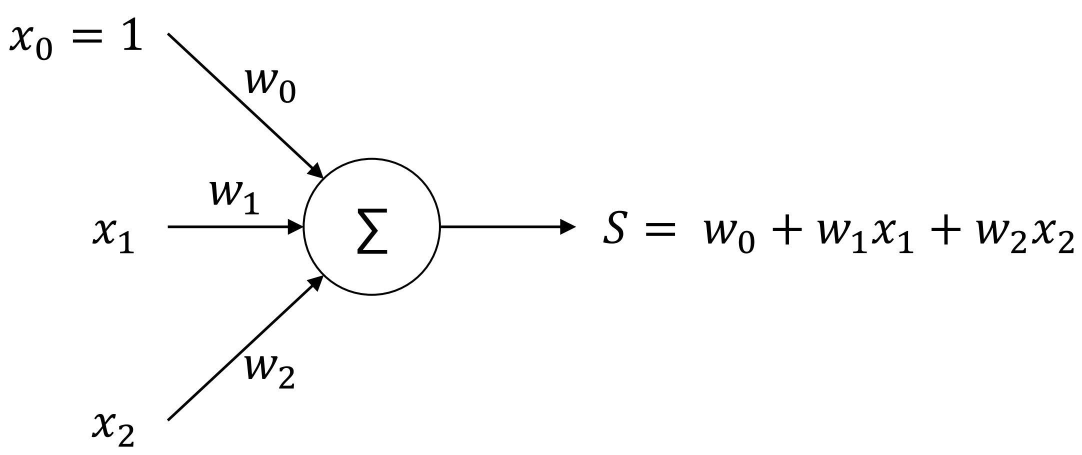
$$y = \begin{cases} 1 & \text{if } (w_1x_1 + w_2x_2 + b \geq 0) \\ 0 & otherwise \end{cases}$$

퍼셉트론의 동작 원리

◆ 퍼셉트론의 구조와 표현 수식

벡터내적.

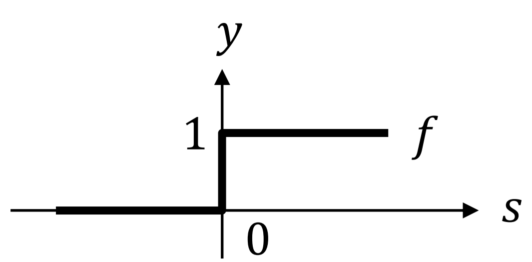
$$s = \sum_{i=1}^{n} w_i x_i + w_0 = \sum_{i=0}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}$$



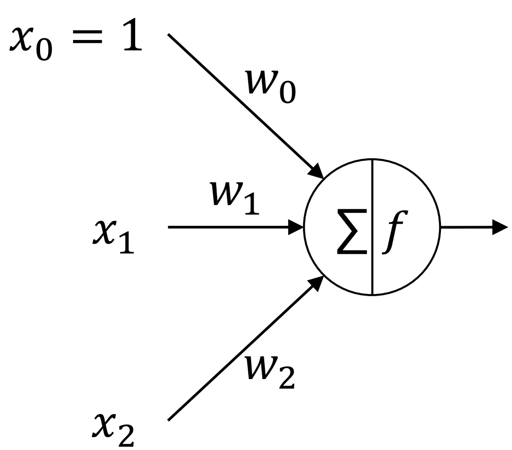
퍼셉트론의 동작 원리

◆ 퍼셉트론의 구조와 표현 수식 - 활성함수 (Activation Function) 포함

$$s = \sum_{i=1}^{n} w_i x_i + w_0 = \sum_{i=0}^{n} w_i x_i = \mathbf{w} \cdot \mathbf{x} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}$$



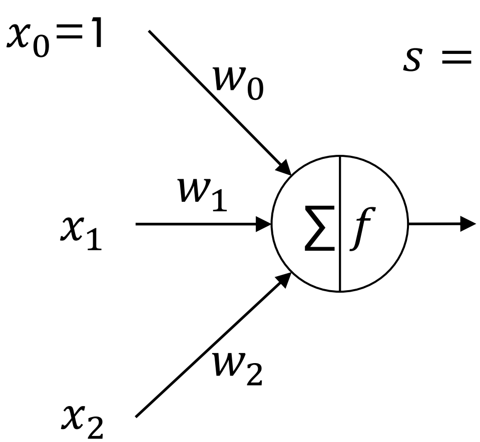
Activation Function
: Hard Limit



$$y = \begin{cases} 1 & s > 0 \\ 0 & otherwise \end{cases}$$

퍼셉트론이 할 수 있는 일 - 1

◆ 퍼셉트론으로 만들어낸 AND 연산



$$s = w_0 + w_1x_1 + w_2x_2$$

-1.5+0+0

-1.5+0+1

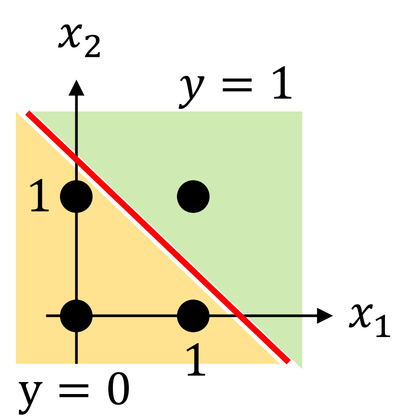
$$y = \begin{cases} 1 & s > 0 \\ 0 & otherwise \end{cases}$$

-1.5+1+0

-1.5+1+1

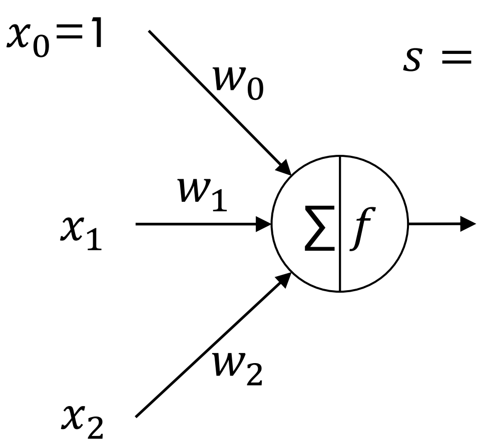
$$w_0 = -1.5, w_1 = 1.0, w_2 = 1.0$$

x_1	x_2	Σ	y
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1



퍼셉트론이 할 수 있는 일 - 2

◆ 퍼셉트론으로 만들어낸 OR 연산

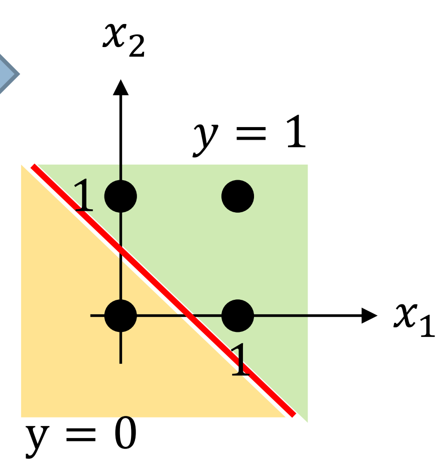


$$s = w_0 + w_1 x_1 + w_2 x_2$$

$$y = \begin{cases} 1 & s > 0 \\ 0 & otherwise \end{cases}$$

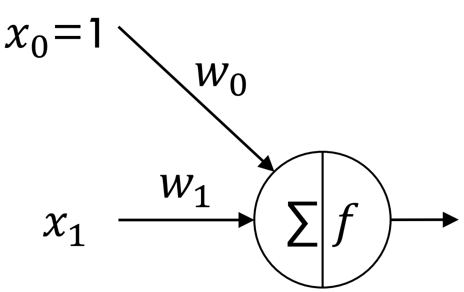
$$w_0 = -0.5, w_1 = 1.0, w_2 = 1.0$$

x_1	x_2	Σ	y
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1



퍼셉트론이 할 수 있는 일 - 3

◆ 퍼셉트론으로 만들어낸 NOT 연산

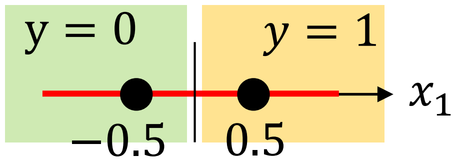


$$s = w_0 + w_1 x_1$$

$$y = \begin{cases} 1 & s > 0 \\ 0 & otherwise \end{cases}$$

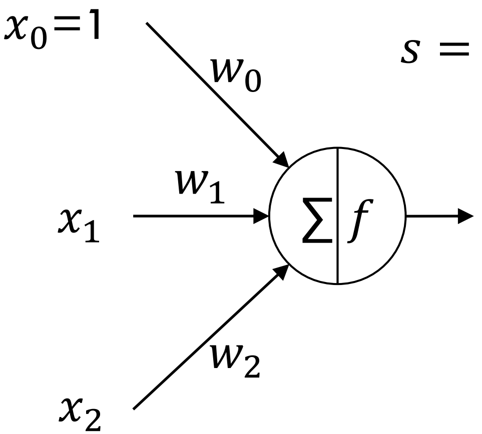
$$w_0 = 0.5, w_1 = -1.0$$

x_1	Σ	y
0	0.5	1
1	-0.5	0



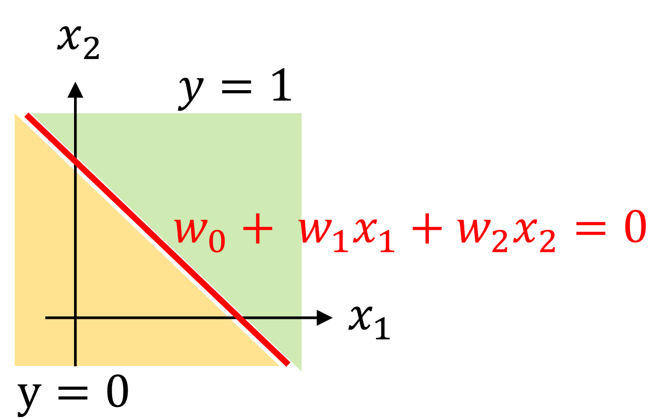
퍼셉트론이 할 수 있는 일 - 4

◆ 퍼셉트론으로 풀 수 있는 평면상의 경계 구분 문제



$$s = w_0 + w_1 x_1 + w_2 x_2$$

$$y = \begin{cases} 1 & s > 0 \\ 0 & otherwise \end{cases}$$



퍼셉트론 학습 알고리즘

- 학습이라고 부르려면 신경망이 스스로 가중치를 자동으로 설정해주는 알고리즘이 필요하다. 퍼셉트론에서도 학습 알고리즘이 존재한다.

→ 경사하강법 적용가능.

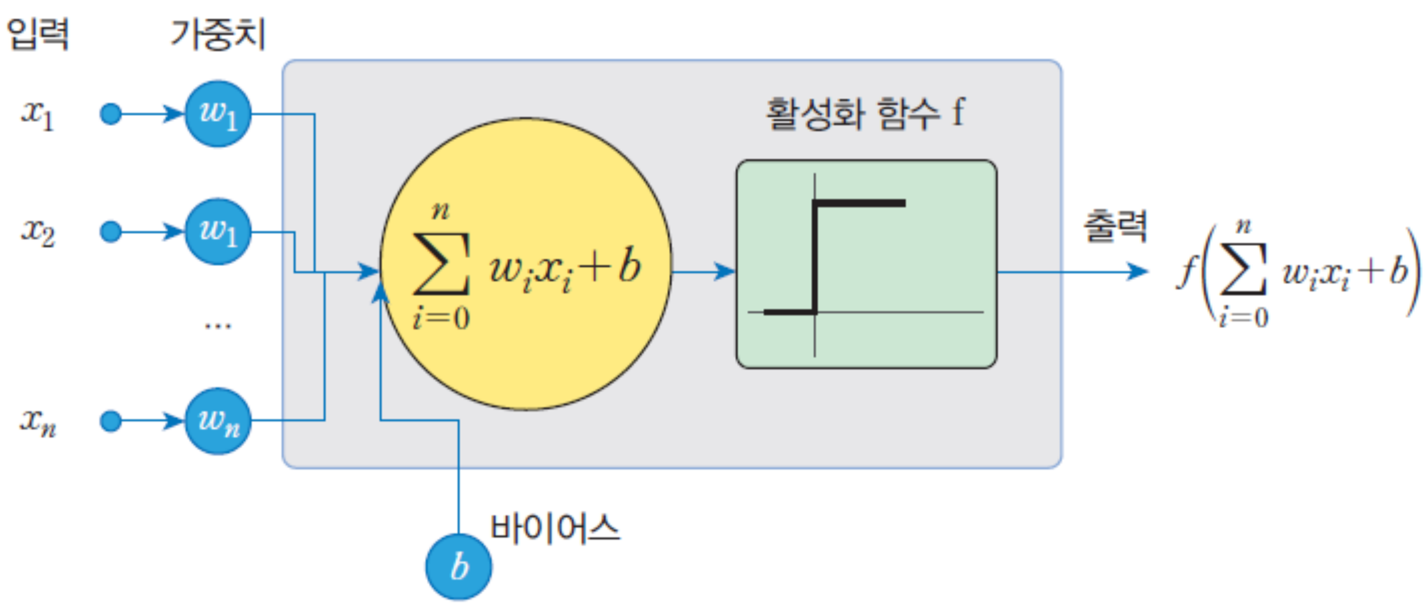


그림 5-8 퍼셉트론

확률적 경사하강법

- 확률적 경사하강법 (Stochastic Gradient Descent)

=> 학습 데이터가 n개 일 때,

경사하강법(GD)은 n개의 error의 평균을 사용하여 weights를 업데이트 한다면,

확률적 경사하강법(SGD)은 1개의 error를 사용하여 weights를 업데이트 한다.

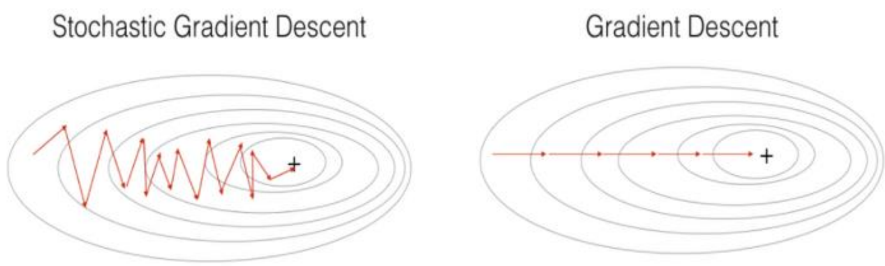
경사하강법(GD) $Loss(W, b) = \frac{1}{n}\sum_{i=1}^{n}((Wx_i + b) - y_i)^2$ n개의 데이터셋 다 사용

확률적 경사하강법(SGD) $Loss(W, b) = ((Wx_i + b) - y_i)^2$ 1개만 쓰기에 다 필요X

간단

메모리·연산↓

< 최소값을 찾는 과정 >

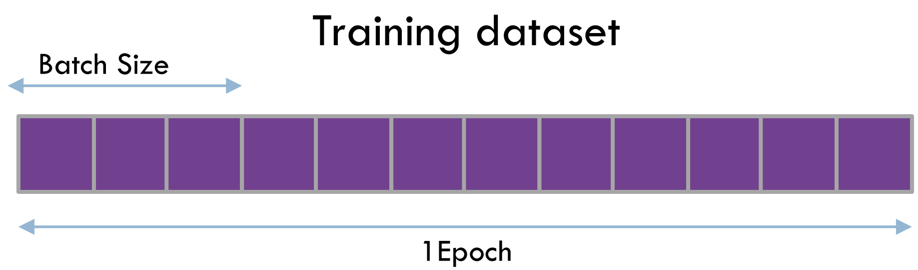


- 노이즈가 심하지만, 반복이 충분하면 SGD로 GD만큼의 결과를 얻을 수 있음.

(최종 투표 결과가 GD라면, 사전 출구 조사가 SGD임)

Global minimum에 도달!

확률적 경사하강법 (Epoch, Batch)



차이 이해.

주

1 Epoch = 데이터 셋의 모든 데이터를 이용하여 학습했음을 의미 사용한 데이터. 주로 epoch 사용.

1 iteration = 1회 학습 (= weights(가중치) 1회 업데이트) 경사하강법 사용 횟수

Minibatch (그냥 batch라고도 함) = 데이터 셋을 batch size 크기로 쪼개서 학습함

일부 데이터셋.

(쪼개지 않은 것을 full-batch라고 하기도 함)

전체 데이터 셋

Ex> 데이터 개수가 100개일 때, = Epoch

데이터 갯수 = mini batch x iteration

경사하강법(GD)로 학습하면 1 Epoch에 1 iteration이 된다면

minbatch의 크기가 1인 확률적 경사하강법(SGD)로 학습하면 1 Epoch에 100 iteration이 됨

= Epoch

데이터 개수가 12개이고, 10 Epoch 학습시켰을 때,

12 = 3 x iteration

경사하강법(GD)과 minbatch의 크기가 3인 확률적 경사하강법 (SGD)의 iteration 횟수는 ?

4회

퍼셉트론 학습 알고리즘

input: 학습 데이터 $(x^1, d^1), \ldots, (x^m, d^m)$

① 모든 w와 바이어스 b를 0 또는 작은 난수로 초기화한다.
② while (가중치가 변경되지 않을 때까지 반복)
③ for 각 학습 데이터 $\boldsymbol{x}^{\boldsymbol{k}}$와 정답 d^k
④ $y^k(t) = f(\boldsymbol{w}(\boldsymbol{t}) \bullet \boldsymbol{x}^{\boldsymbol{k}})$
⑤ 모든 가중치 w_i에 대하여 $w_i(t+1) = w_i(t) + \eta \bullet (d^k - y^k(t)) \bullet x_i^k$

논리 연산자 학습 과정 (교재 설명)

$$w_i(t+1) = w_i(t) + \eta \cdot (d^k - y^k(t)) \cdot x_i^k$$

- 퍼셉트론이 1을 0으로 잘못 식별했다고 하자. 가중치의 변화량은 η * (1-0) * x_i^k 가 된다. 따라서 가중치는 증가된다. 가중치가 증가되면 출력도 증가되어서 출력이 0에서 1이 될 가능성이 있다.

- 반대로 0을 1로 잘못 식별했다고 하자. 가중치의 변화량은 η * (0-1) * x_i^k 가 된다. 따라서 가중치는 줄어든다. 가중치가 줄어들면 출력도 감소되어서 출력이 1에서 0이 될 가능성이 있다.

논리 연산자 학습 과정 (교수자 설명)

$$w_i(t+1) = w_i(t) + \eta \cdot (d^k - y^k(t)) \cdot x_i^k$$

k는 데이터를 구분하기 위해 필요
i 는 독립변수와 가중치의 구분을 위해 필요
t 와 t+1 은 업데이트 전과 후를 구분을 위해 필요

$y^k(t) = f(w(t) \cdot x) = w_2^K(t) \cdot x_2^K + w_1^K(t) \cdot x_1^K + w_0^K(t) \cdot x_0^K \quad (x_0^K=1)$

(이해를 돕기 위해 독립변수가 2개인 것으로 예시를 들었음)

개수 빠뜨X , 미분시 상쇄위해

$loss(w) = 1/2(d^k - y^k(t))^2$ (loss를 k번 째 데이터만 고려한다는 것 자체가 확률적 경사하강법을 쓴다는 것임)

답지 예측

$\partial\, loss(w) / \partial w_2^K(t) = (d^k - \underbrace{(w_2^K(t) \cdot x_2^K + w_1^K(t) \cdot x_1^K + w_0^K(t) \cdot x_0^K)}_{= y^k(t)}) \cdot (-x_2^K)$

얘에 대해 편미분

경사하강법 공식 $w_i(t+1) = w_i(t) - \alpha \cdot \{\partial\, loss(w) / \partial w_i\}$ 에 구해진 경사(Gradient)를 대입하면

$$\begin{aligned} w_i(t+1) &= w_i(t) - \alpha \cdot (d^k - (w_2^K(t) \cdot x_2^K + w_1^K(t) \cdot x_1^K + w_0^K(t) \cdot x_0^K)) \cdot (-x_i^K) \\ &= w_i(t) + \alpha \cdot (d^k - (w_2^K(t) \cdot x_2^K + w_1^K(t) \cdot x_1^K + w_0^K(t) \cdot x_0^K)) \cdot (x_i^K) \\ &= w_i(t) + \alpha \cdot (d^k - y^k(t)) \cdot (x_i^K) \end{aligned}$$

$$\begin{aligned} w_2(t+1) &= w_2(t) + \alpha \cdot (d^k - y^k(t)) \cdot (x_2^K) \\ w_1(t+1) &= w_1(t) + \alpha \cdot (d^k - y^k(t)) \cdot (x_1^K) \\ w_0(t+1) &= w_0(t) + \alpha \cdot (d^k - y^k(t)) \cdot (x_0^K) \end{aligned}$$

퍼셉트론 학습 알고리즘


```
import numpy as np

epsilon = 0.0000001                    # 부동소수점 오차 방지

def step_func(t):              # 퍼셉트론의 활성화 함수
    if t > epsilon: return 1
    else: return 0

X = np.array([                 # 훈련 데이터 세트
    [0, 0, 1],                 # 맨 끝의 1은 바이어스를 위한 입력 신호 1이다.
    [0, 1, 1],                 # 맨 끝의 1은 바이어스를 위한 입력 신호 1이다.
    [1, 0, 1],                 # 맨 끝의 1은 바이어스를 위한 입력 신호 1이다.
    [1, 1, 1]                  # 맨 끝의 1은 바이어스를 위한 입력 신호 1이다.
])

y = np.array([0, 0, 0, 1])     # 정답을 저장하는 넘파이 행렬
W = np.zeros(len(X[0]))        # 가중치를 저장하는 넘파이 행렬
```

퍼셉트론 학습 알고리즘


```
def perceptron_fit(X, Y, epochs=10):   # 퍼셉트론 학습 알고리즘 구현
    global W
    eta = 0.2                                           # 학습률

    for t in range(epochs):
        print("epoch=", t, "======================")
        for i in range(len(X)):
            predict = step_func(np.dot(X[i], W))
            error = Y[i] - predict                      # 오차 계산
            W += eta * error * X[i]                     # 가중치 업데이트
            print("현재 처리 입력=",X[i],"정답=",Y[i],"출력=",predict,"변경된 가중치=", W)
        print("================================")
```

퍼셉트론 학습 알고리즘


```
def perceptron_predict(X, Y):                    # 예측
    global W
    for x in X:
        print(x[0], x[1], "->", step_func(np.dot(x, W)))

perceptron_fit(X, y, 6)
perceptron_predict(X, y)
```

```
epoch= 0 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 0 변경된 가중치= [0. 0. 0.]
현재 처리 입력= [0 1 1] 정답= 0 출력= 0 변경된 가중치= [0. 0. 0.]
현재 처리 입력= [1 0 1] 정답= 0 출력= 0 변경된 가중치= [0. 0. 0.]
현재 처리 입력= [1 1 1] 정답= 1 출력= 0 변경된 가중치= [0.2 0.2 0.2]
==================================
epoch= 1 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 1 변경된 가중치= [0.2 0.2 0. ]
현재 처리 입력= [0 1 1] 정답= 0 출력= 1 변경된 가중치= [ 0.2  0.  -0.2]
현재 처리 입력= [1 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.2  0.  -0.2]
현재 처리 입력= [1 1 1] 정답= 1 출력= 0 변경된 가중치= [0.4 0.2 0. ]
==================================
epoch= 2 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 0 변경된 가중치= [0.4 0.2 0. ]
현재 처리 입력= [0 1 1] 정답= 0 출력= 1 변경된 가중치= [ 0.4  0.  -0.2]
현재 처리 입력= [1 0 1] 정답= 0 출력= 1 변경된 가중치= [ 0.2  0.  -0.4]
현재 처리 입력= [1 1 1] 정답= 1 출력= 0 변경된 가중치= [ 0.4  0.2 -0.2]
==================================
epoch= 3 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.2]
현재 처리 입력= [0 1 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.2]
현재 처리 입력= [1 0 1] 정답= 0 출력= 1 변경된 가중치= [ 0.2  0.2 -0.4]
현재 처리 입력= [1 1 1] 정답= 1 출력= 0 변경된 가중치= [ 0.4  0.4 -0.2]
==================================
```

```
epoch= 4 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.4 -0.2]
현재 처리 입력= [0 1 1] 정답= 0 출력= 1 변경된 가중치= [ 0.4  0.2 -0.4]
현재 처리 입력= [1 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.4]
현재 처리 입력= [1 1 1] 정답= 1 출력= 1 변경된 가중치= [ 0.4  0.2 -0.4]
=================================
epoch= 5 =======================
현재 처리 입력= [0 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.4]
현재 처리 입력= [0 1 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.4]
현재 처리 입력= [1 0 1] 정답= 0 출력= 0 변경된 가중치= [ 0.4  0.2 -0.4]
현재 처리 입력= [1 1 1] 정답= 1 출력= 1 변경된 가중치= [ 0.4  0.2 -0.4]
=================================
0 0 -> 0
0 1 -> 0
1 0 -> 0
1 1 -> 1
```

퍼셉트론의 한계점

- XOR 연산

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

배타적일 때 True.

```
...
0 0 -> 1
0 1 -> 1
1 0 -> 0
1 1 -> 0
```

원하는 출력이 나오지 않는다.

선형 분류 가능 문제

- 패턴 인식 측면에서 보면 퍼셉트론은 직선을 이용하여 입력 패턴을 분류하는 선형 분류자(linear classifier)의 일종이라고 말할 수 있다.

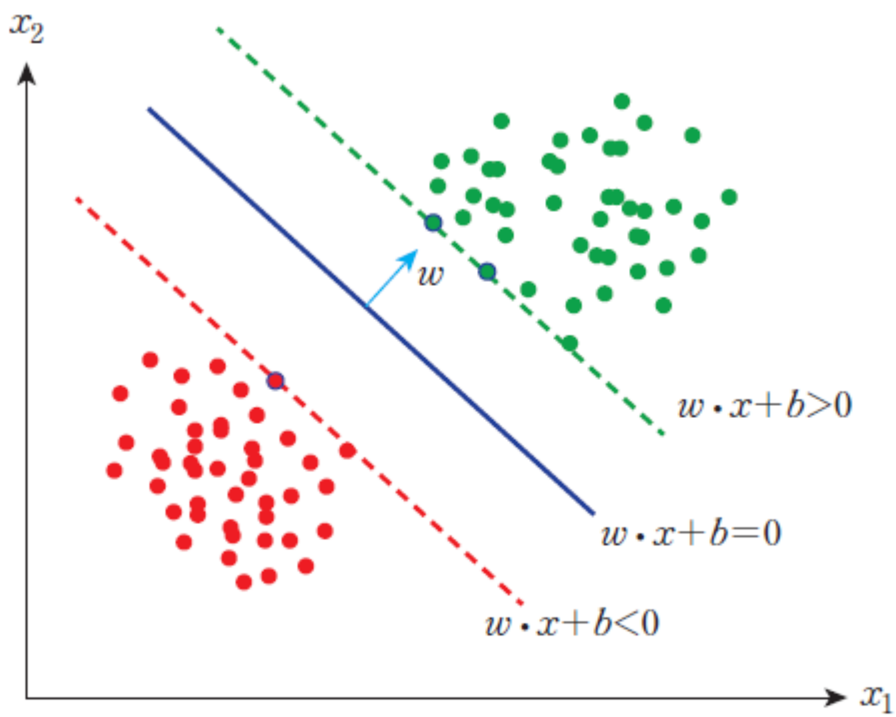


그림 5-10 선형 분류자

선형 분류 가능 문제

- Minsky와 Papert는 1969년에 발간된 책 "Perceptrons"에서 1개의 레이어(layer, 계층)으로 구성된 퍼셉트론은 XOR 문제를 학습할 수 없다는 것을 수학적으로 증명

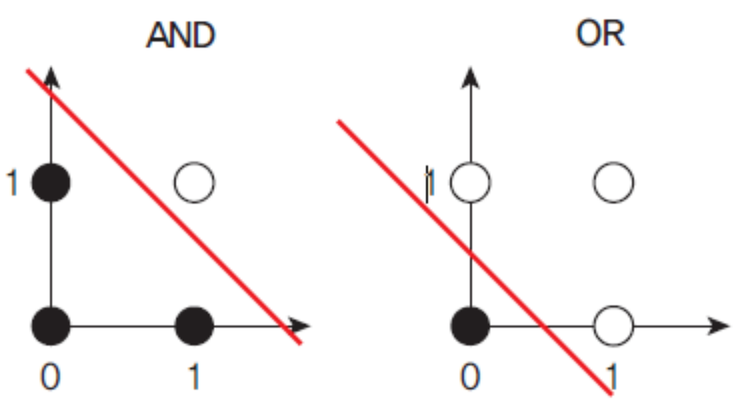


그림 5-11 선형 분리 가능한 문제

그림 5-12 선형 분리 불가능한 문제

다층 퍼셉트론으로 XOR 문제를 해결

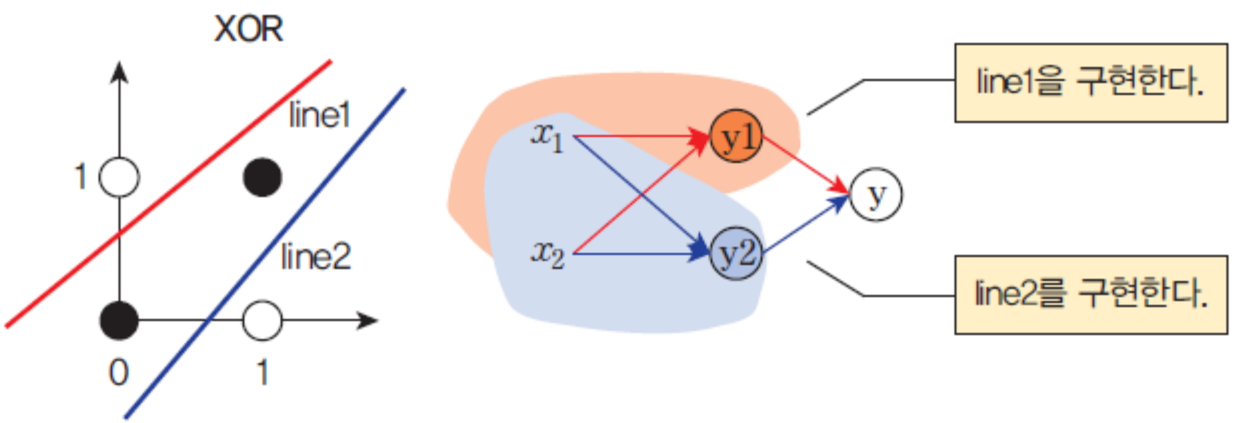
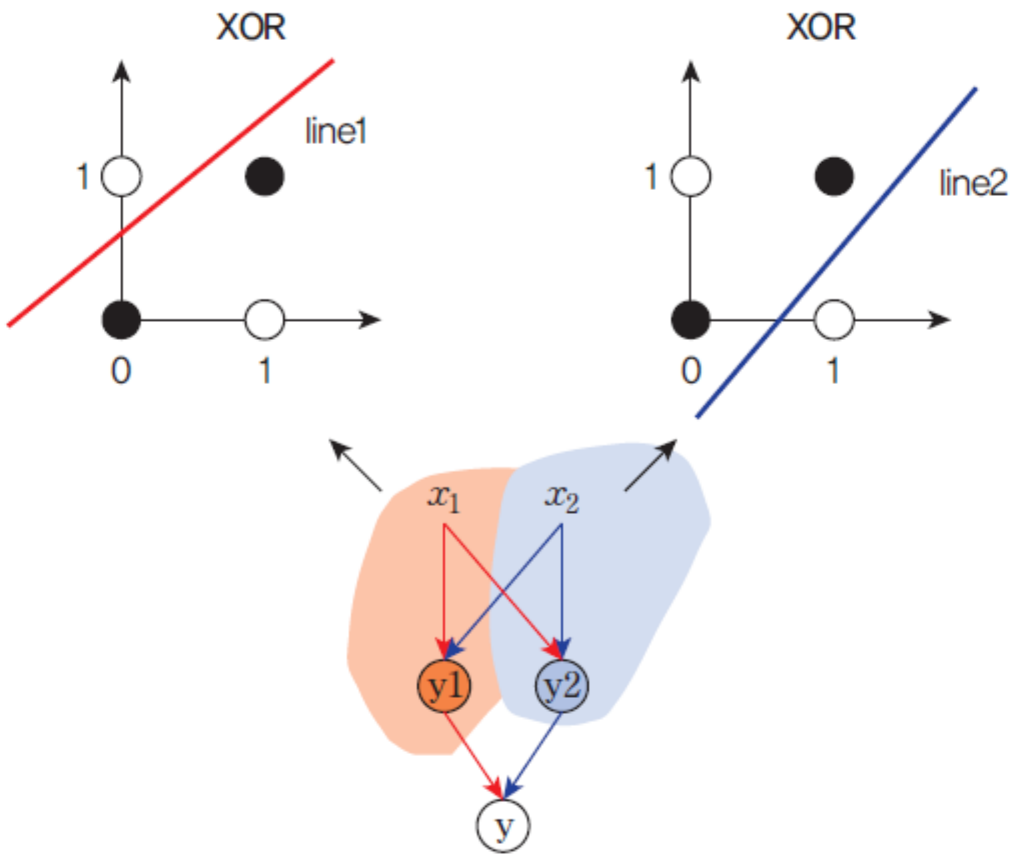


그림 5-13 다층을 사용하는 퍼셉트론

Deep. 한층 : 퍼셉트론

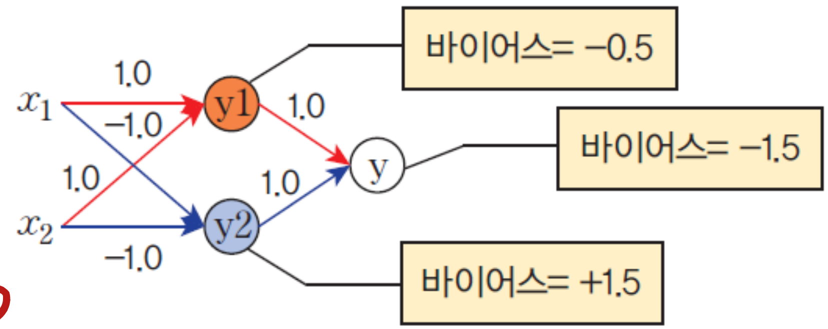
↳ 여러 층의 퍼셉트론(MLP)

다층 퍼셉트론으로 XOR 문제를 해결



다층 퍼셉트론으로 XOR 문제를 해결

$W_{11} = 1.0$
$W_{12} = -1.0$
$W_{21} = 1.0$
$W_{22} = -1.0$



$W_1 = 1.0$
$W_2 = 1.0$

그림 5-14 다층 퍼셉트론에서 XOR 문제 해결

x_1	x_2	y1	y2	y	XOR 출력
0	0	0	1	0	0
1	0	1	1	1	1
0	1	1	1	1	1
1	1	1	0	0	0

5주차 과제.

Mini Project: 퍼셉트론으로 분류

- 대학생들의 신장과 체중을 받아서 성별을 출력하는 퍼셉트론을 만들어보자. (체중은 100 kg으로, 키는 200 cm로 스케일링(데이터를 0~1값으로 전환) 해서 푸세요)

WB 구하기.

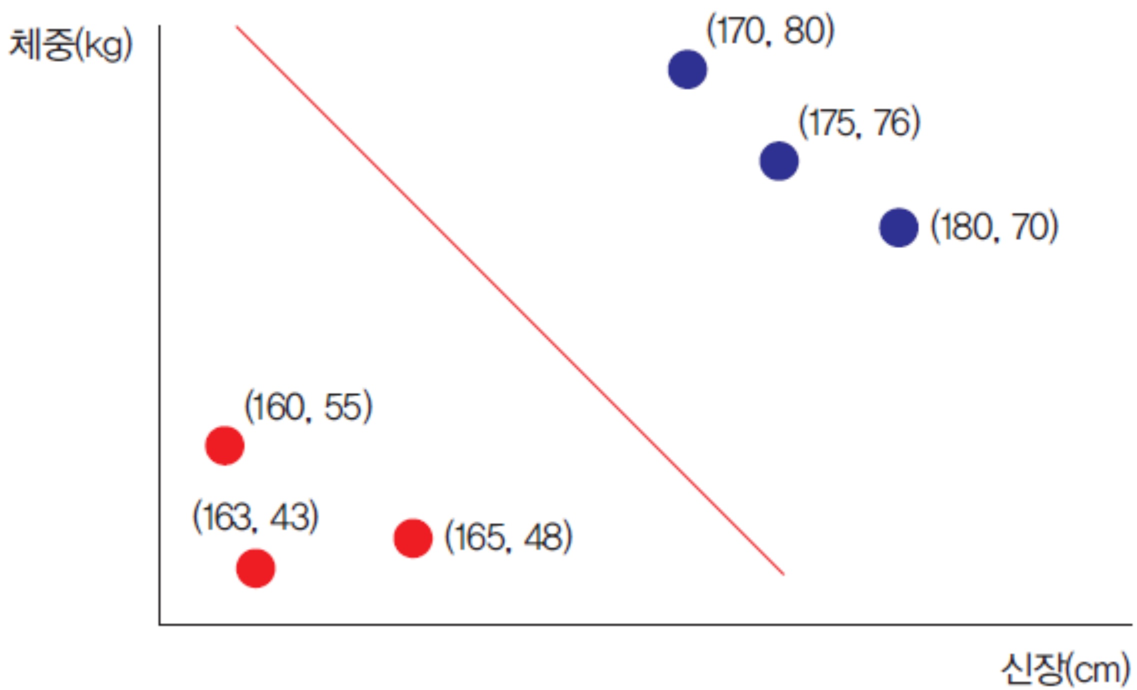


그림 5-15 신장과 체중으로 남녀를 구분하는 문제

Summary

- 딥러닝(deep learning)의 시작은 1950년대부터 연구되어 온 인공 신경망(artificial neural network: ANN)이다.
- 신경망의 가장 큰 장점은 학습이 가능하다는 점이다. 데이터만 주어지면 신경망은 예제로부터 배울 수 있다.
- 뉴런은 다른 뉴런들로부터 신호를 받아서 모두 합한 후에 비선형 함수를 적용하여 출력을 계산한다. 연결선은 가중치를 가지고 있고 이 가중치에 학습의 결과가 저장된다.
- 입력을 받아서 뉴런을 활성화시키는 함수를 활성화 함수(activation function)라고 한다.
- 퍼셉트론은 하나의 뉴런만을 사용한다. 다수의 입력을 받아서 하나의 신호를 출력하는 장치이다.
- 퍼셉트론은 AND나 OR 같은 논리적인 연산을 학습할 수 있었지만 XOR 연산은 학습할 수 없었다. 선형 분리 가능한 문제만 학습할 수 있었다.

Q & A