

퍼셉트론



학습 목표

- 신경망에 대하여 이해한다.
- 신경망의 초기 모델인 **퍼셉트론**을 이해한다.
- 퍼셉트론의 **작동원리**를 이해한다. dnn 가
- 퍼셉트론 **학습 알고리즘**을 이해한다.
- 퍼셉트론의 한계점을 인식한다.



퍼셉트론은 아주
오래된 모델이지만, 이해하기
쉽고, 많은 사람들에게 영감을 준,
전설적인 모델입니다. 우리는
즐겁게 신경망을 시작해봅시다.
시작이 중요합니다.

- 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.

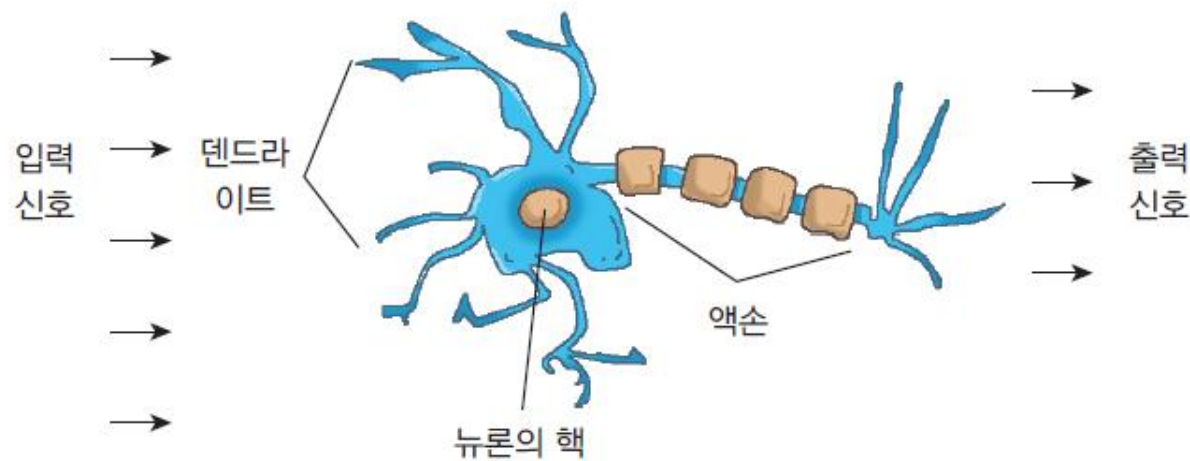
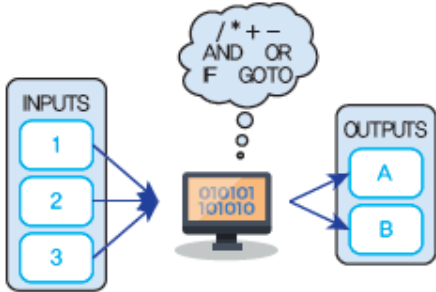



그림 5-1 뉴런의 구조



전통적인 컴퓨터 vs 인공지능망

	기존의 컴퓨터	인간의 두뇌
처리소자의 개수	10^8 개의 트랜지스터	10^{10} 개의 뉴런
처리소자의 속도	10^{12} Hz	10^2 Hz
<u>학습기능</u>	없음	있음
계산 스타일	중앙집중식, 순차적인 처리	분산 병렬 처리
	 <p>The diagram illustrates a traditional computer architecture. On the left, a box labeled 'INPUTS' contains three numbered slots (1, 2, 3). Arrows point from these slots to a central computer monitor. Above the monitor is a thought bubble containing the text '/ * + - AND OR F GOTO'. The monitor itself displays the binary code '010101' and '101010'. Arrows point from the monitor to a box on the right labeled 'OUTPUTS', which contains two slots labeled 'A' and 'B'.</p>	 <p>The diagram shows a human brain filled with a dense, interconnected network of neurons, represented by colored dots (red, blue, green, yellow) connected by lines, illustrating a distributed and parallel processing system.</p>



신경망의 장점

- 첫 번째는 **학습이 가능**하다는 점이다. 데이터만 주어진다면 신경망은 예제로부터 배울 수 있다.
- 두 번째는 몇 개의 소자가 **오동작하더라도 전체적으로는 큰 문제가 발생하지 않는다**는 점이다.

가

가



그림 5-3 학습하는 컴퓨터



신경망이 필요한 분야

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

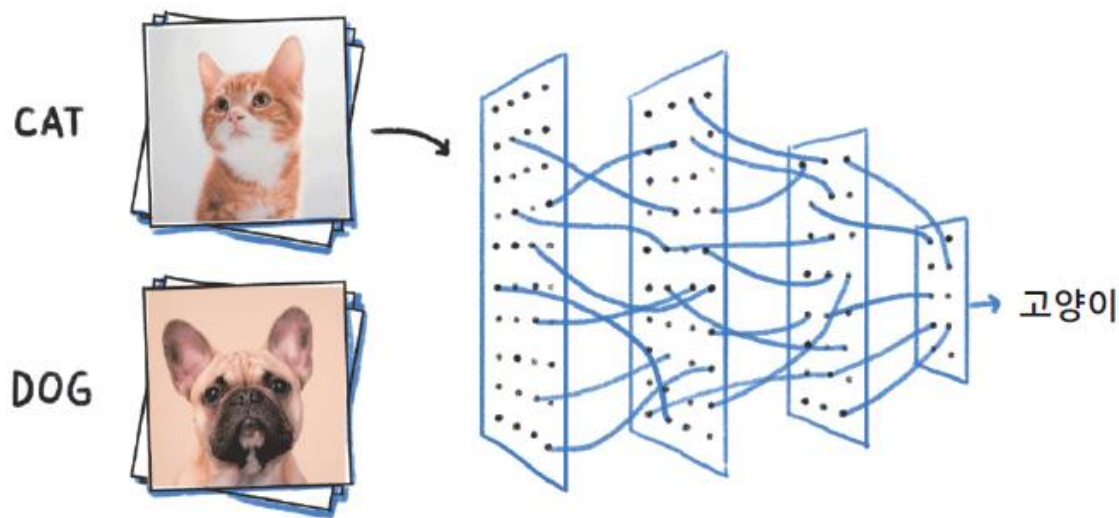
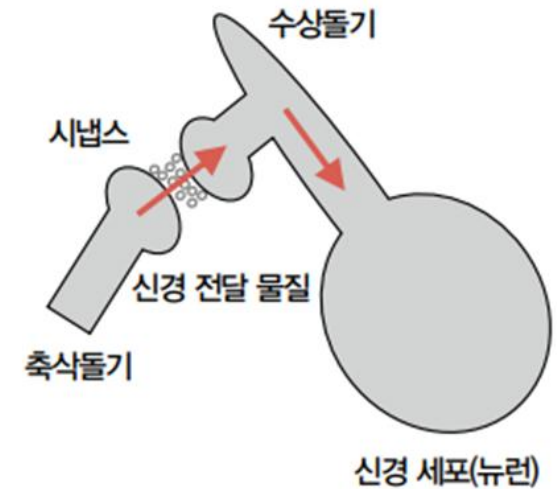
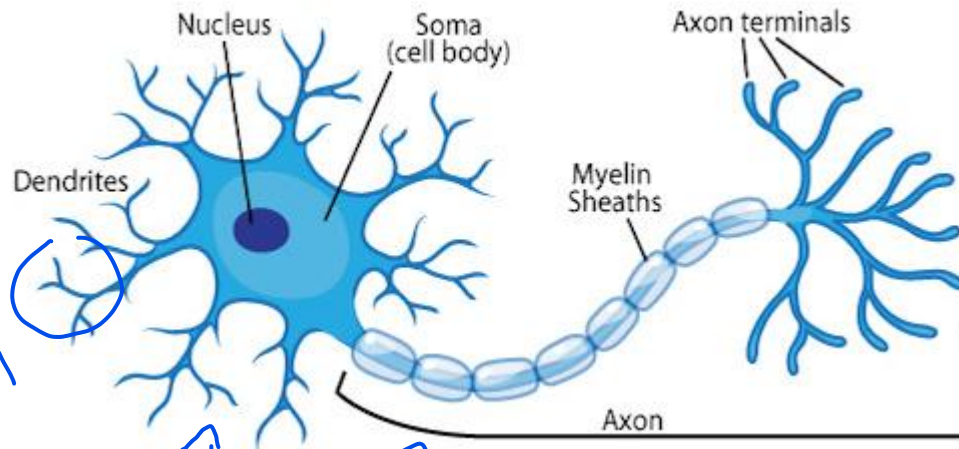


그림 5-4 신경망을 이용한 이미지 인식



Human's neuron



- 신경세포체(cell body) - 세포의 핵 및 세포기관
- 수상돌기(dendrites) - 축삭말단과 연결되어 신호를 수신하는 부분
- 축삭돌기(axon) - 신호를 전달하는 부분
- 시냅스(synapse) - 수상돌기와 축삭돌기 사이에 있는 부분으로 신호 전달의 세기를 담당함



Artificial neural networks

- 인간 두뇌의 생물학적 뉴런의 작용을 모방한 모델
- 뉴런들로부터의 **입력**을 **일정한 함수를 거쳐 출력**

McCulloch-Pitts model

뉴런이 다른 뉴런으로부터 전기신호를 받았을 때, 일정 기준을 넘으면 다음 뉴런으로 신호를 전달하는 현상을 수학적으로 고안





퍼셉트론의 시작

- 1949년 Donald Hebb이 제안

시냅스의 가소성 있는 변화

- 학습(learning)이나 훈련(training)이 진행됨에 따라 신경세포들 사이의 연결강도(connection weight)가 변화함!
- 상호작용 증가 - 시냅스 강화 >> weight 증가
- 상호작용 감소 - 시냅스 약화 >> weight 감소





퍼셉트론

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

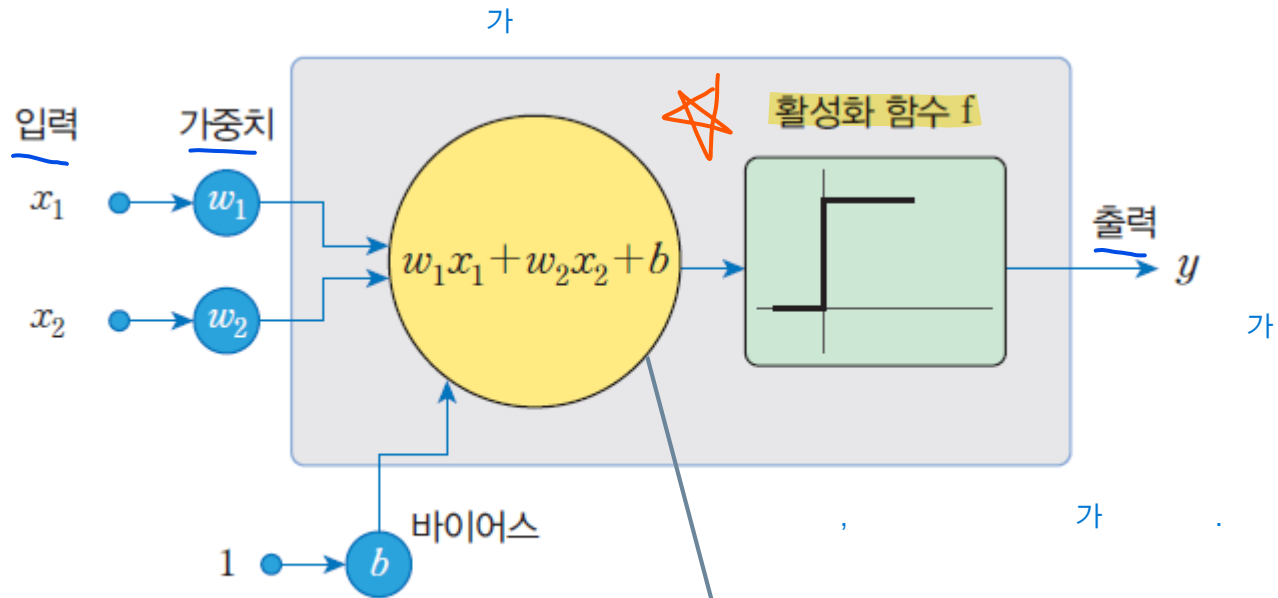
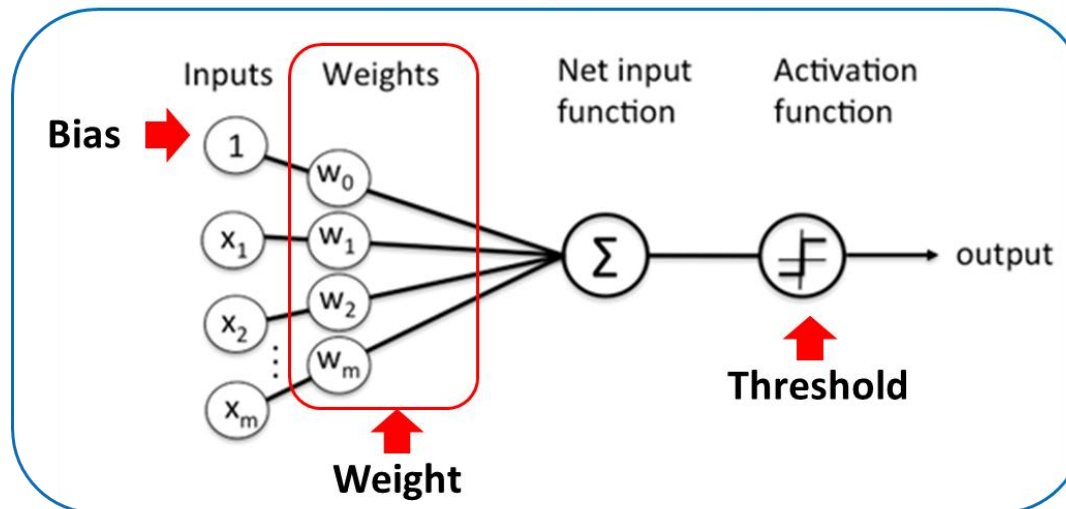


그림 5-5 퍼셉트론에서의 뉴런

$$b = -\theta$$

퍼셉트론

Perceptron



- 바이어스(bias): 선형 경계의 절편을 나타내는 값, 상수
- 가중치(weight): 선형 경계의 **방향성** 또는 **형태**를 나타내는 값
- 임계치(threshold): 어떠한 값이 활성화되기 위한 최소값
- 활성화함수(activation function): perceptron에서 계산된 net값이 임계치보다 크면 1을 출력하고, 임계치보다 작은 경우에는 0을 출력하는 함수



퍼셉트론

- 뉴런에서는 입력 신호의 가중치 합이 어떤 임계값을 넘는 경우에만 뉴런이 활성화되어서 1을 출력한다. 그렇지 않으면 0을 출력한다.

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



퍼셉트론은 논리 연산을 학습할 수 있을까?

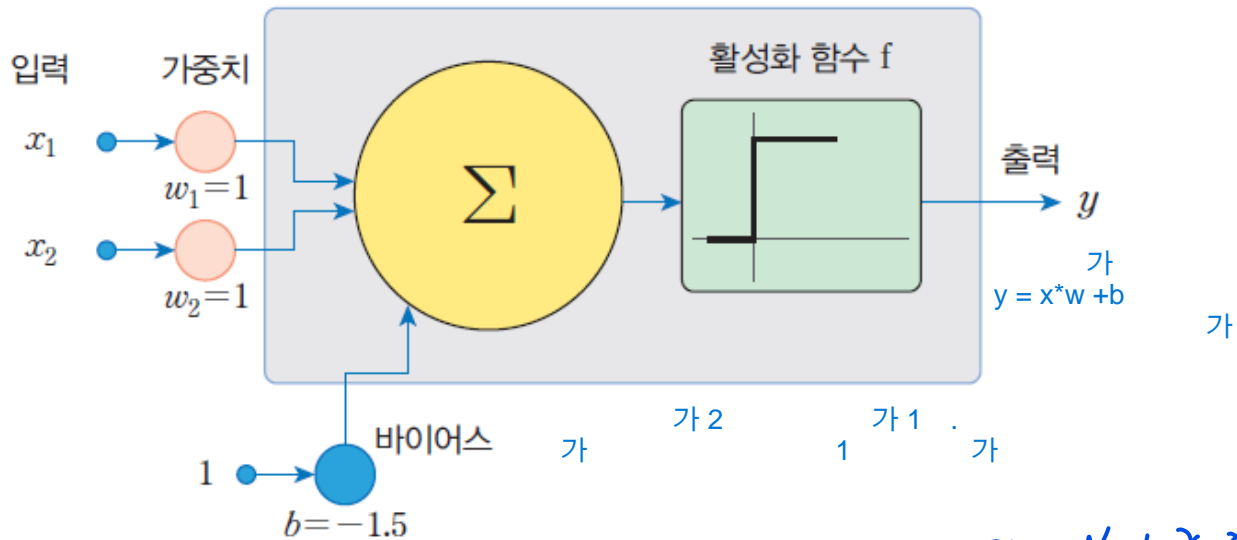


그림 5-6 논리 연산을 하는 퍼셉트론

$$y = x_1 w_1 + x_2 w_2 + b$$
$$y \geq 0$$

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



퍼셉트론은 논리 연산을 학습할 수 있을까?

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
1	0	$1*1+1*0=1$	-1.5	0
0	1	$1*0+1*1=1$	-1.5	0
1	1	$1*1+1*1=2$	-1.5	1



활성화 함수

- 계단 함수

information loss 가

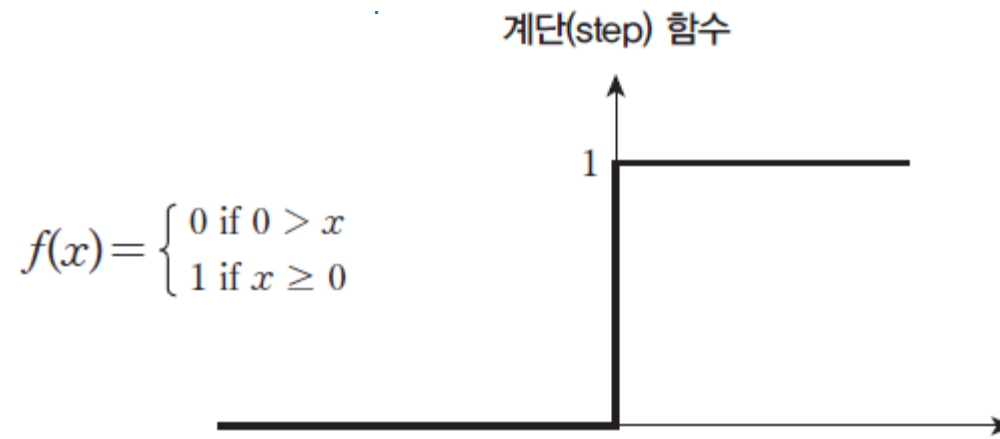


그림 5-7 퍼셉트론에서의 활성화 함수



퍼셉트론 구현 #1(순수 파이썬 사용)

```
epsilon = 0.0000001
```

```
def perceptron(x1, x2):  
    w1, w2, b = 1.0, 1.0, -1.5  
    sum = x1*w1+x2*w2+b  
    if sum > epsilon :  
        return 1  
    else :  
        return 0
```

부동소수점 오차를 방지하기 위하여

```
print(perceptron(0, 0))  
print(perceptron(1, 0))  
print(perceptron(0, 1))  
print(perceptron(1, 1))
```

```
0  
0  
0  
1
```




퍼셉트론 구현 #2(넘파이 사용)

```
import numpy as np
epsilon = 0.0000001

def perceptron(x1, x2):
    X = np.array([x1, x2])
    W = np.array([1.0, 1.0])
    B = -1.5
    sum = np.dot(W, X)+B
    if sum > epsilon :
        return 1
    else :
        return 0

print(perceptron(0, 0))
print(perceptron(1, 0))
print(perceptron(0, 1))
print(perceptron(1, 1))
```

```
0
0
0
1
```



퍼셉트론 학습 알고리즘

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

가

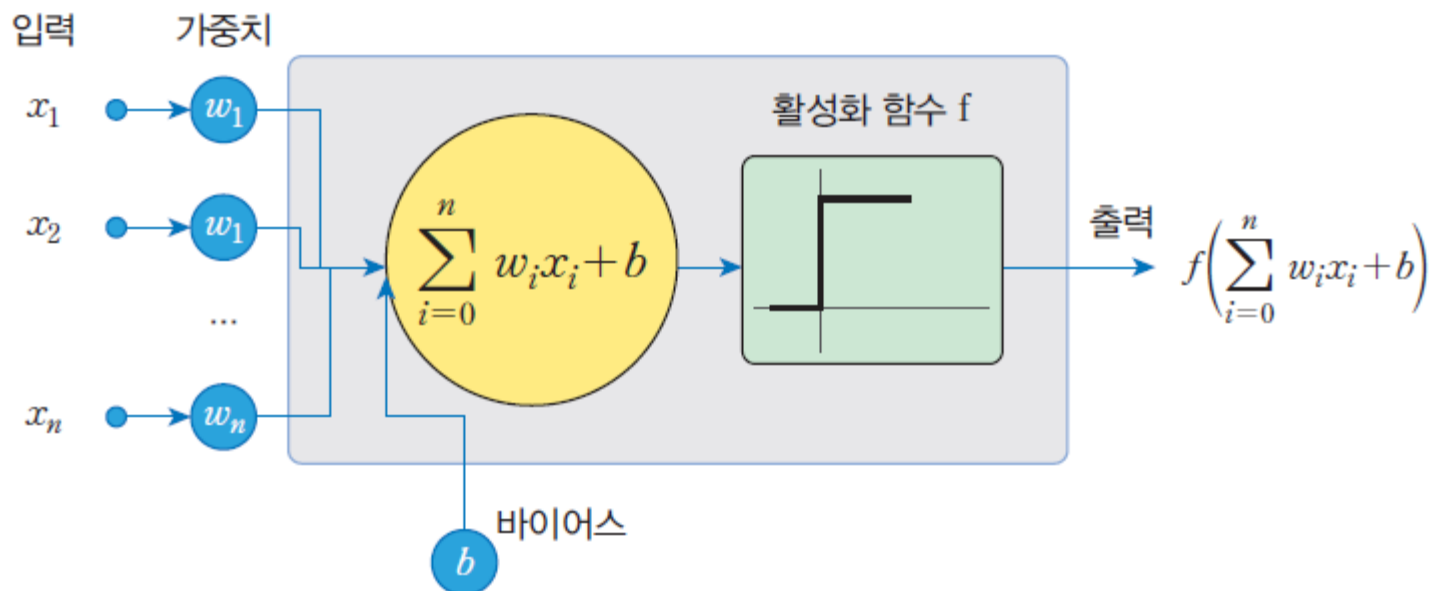


그림 5-8 퍼셉트론

가



퍼셉트론 학습 알고리즘

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

① 모든 w 와 바이어스 b 를 0 또는 작은 난수로 초기화한다.

② while (가중치가 변경되지 않을 때까지 반복)

③ for 각 학습 데이터 x^k 와 정답 d^k

④ $y^k(t) = f(w(t) \cdot x^k)$ 가

⑤ 모든 가중치 w_i 에 대하여 $w_i(t+1) = \underline{w_i(t)} + \eta \cdot \underline{(d^k - y^k(t))} \cdot \underline{x_i^k}$

가



논리 연산자 학습 과정

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

가

- 퍼셉트론이 1을 0으로 잘못 식별했다고 하자. 가중치의 변화량은 $\eta * (1-0) * x_i^k$ 가 된다. 따라서 가중치는 증가된다. 가중치가 증가되면 출력도 증가되어서 출력이 0에서 1이 될 가능성이 있다.
- 반대로 0을 1로 잘못 식별했다고 하자. 가중치의 변화량은 $\eta * (0-1) * x_i^k$ 가 된다. 따라서 가중치는 줄어든다. 가중치가 줄어들면 출력도 감소되어서 출력이 1에서 0이 될 가능성이 있다.



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



퍼셉트론 시뮬레이터

- http://vlabs.iitb.ac.in/vlabs-dev/labs/machine_learning/labs/exp1/simulation.php



sklearn으로 퍼셉트론 실습하기

```
from sklearn.linear_model import Perceptron
```

```
# 샘플과 레이블이다.
```

```
X = [[0,0],[0,1],[1,0],[1,1]]
```

```
y = [0, 0, 0, 1]
```

```
# 퍼셉트론을 생성한다. tol는 종료 조건이다. random_state는 난수의 시드이다.
```

```
clf = Perceptron(tol=1e-3, random_state=0)
```

```
# 학습을 수행한다.
```

```
clf.fit(X, y)
```

```
# 테스트를 수행한다.
```

```
print(clf.predict(X))
```

```
[0 0 0 1]
```



퍼셉트론의 한계점

- XOR 연산

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

...
00 -> 1
01 -> 1
10 -> 0
11 -> 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.

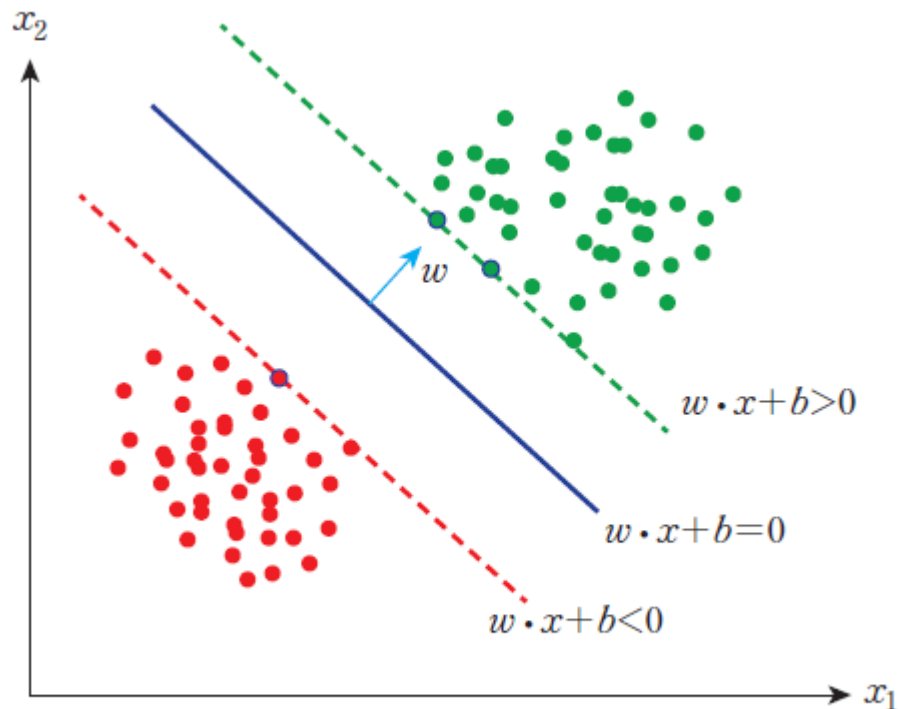


그림 5-10 선형 분류자



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

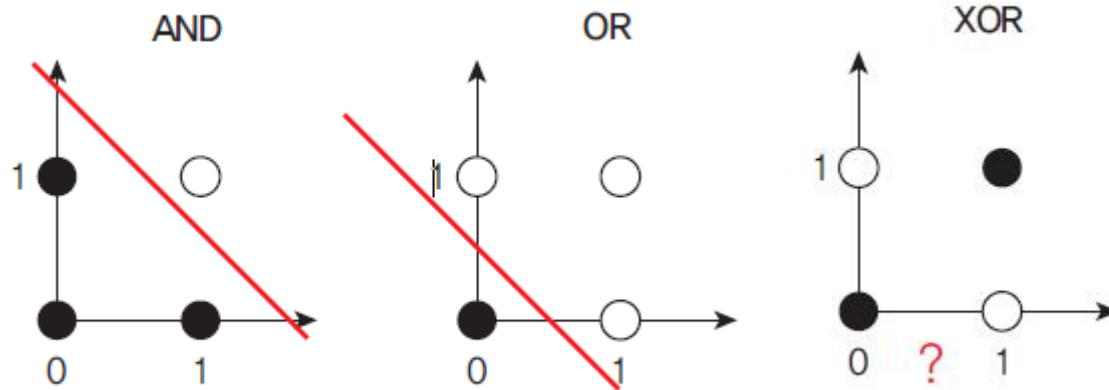


그림 5-11 선형 분리 가능한 문제

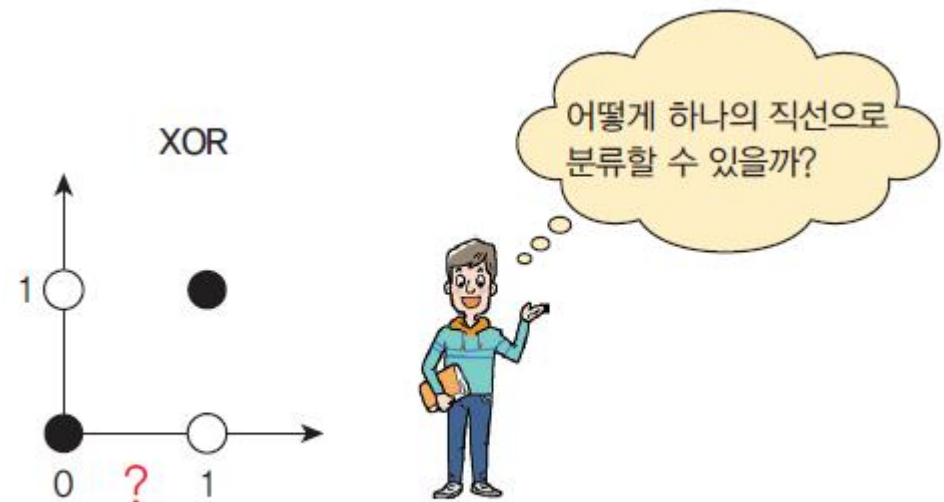


그림 5-12 선형 분리 불가능한 문제

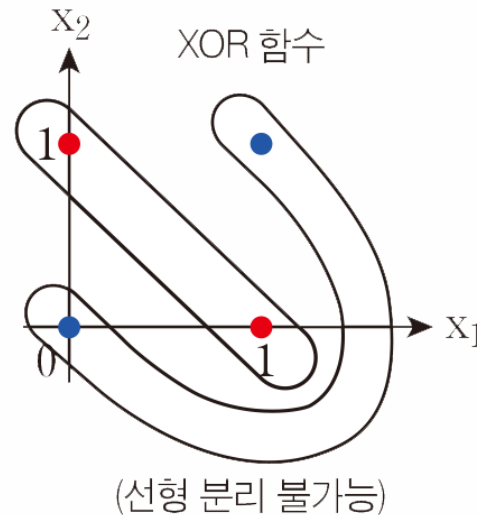


Linear inseparability

- XOR (exclusive-OR) function

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

(XOR 함수)

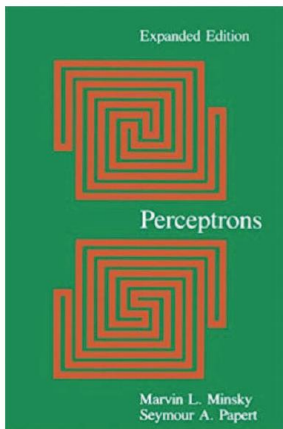


- **선형 분리가 불가능!**
- 한 직선으로 두 집합을 교차하지 않고 나눌 수 없음
- 이 점은 단층 퍼셉트론 학습에서 매우 심각한 문제점
- Limitation of single-layer perceptron



Limitation of single-layer perceptron

- 1969, Marvin Minsky - Perceptions
 - Perceptron의 한계를 지적
 - 단층 퍼셉트론은 학습 모델로서는 적절하지 않음
 - XOR 문제 해결 불가, 10여 년 동안 관심이 멀어짐



single layer perceptron

[그림 9.15] 『퍼셉트론즈』

- 1974, Paul Werbos - Backpropagation Algorithm
- 1986, David Rumelhart (Parallel Distributed Process)
Multi-layers perceptron (다층 퍼셉트론)



다층 퍼셉트론으로 XOR 문제를 해결

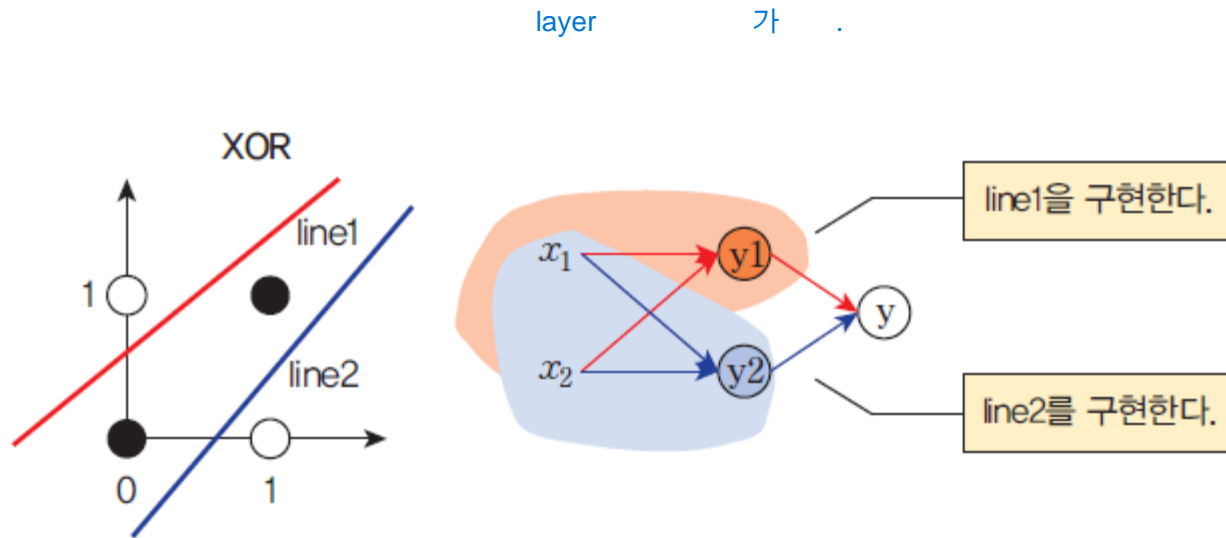
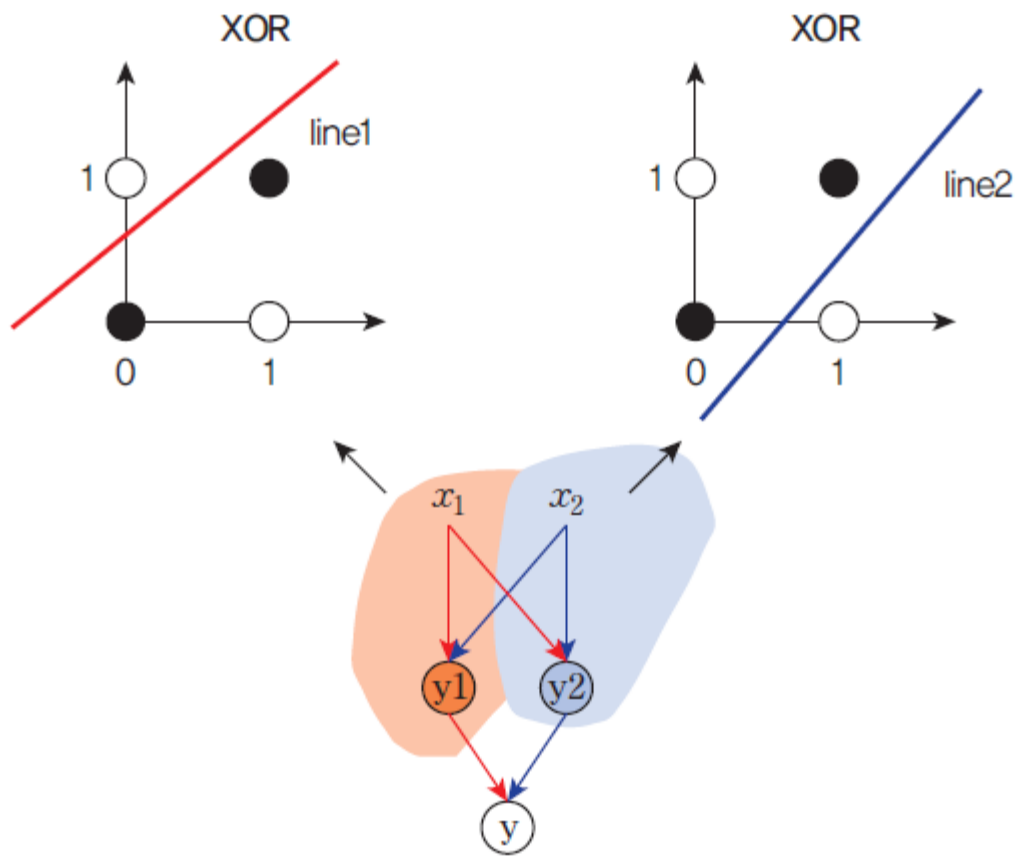


그림 5-13 다층을 사용하는 퍼셉트론



다층 퍼셉트론으로 XOR 문제를 해결





다층 퍼셉트론으로 XOR 문제를 해결

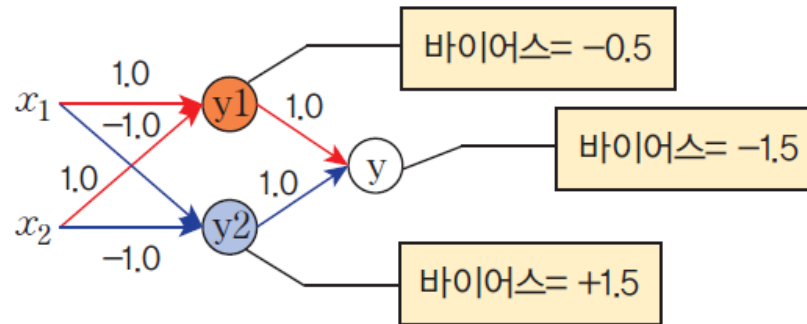


그림 5-14 다층 퍼셉트론에서 XOR 문제 해결

x_1	x_2	y_1	y_2	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



Mini Project: 퍼셉트론으로 분류

- 대학생들의 신장과 체중을 받아서 성별을 출력하는 퍼셉트론을 만들어보자.

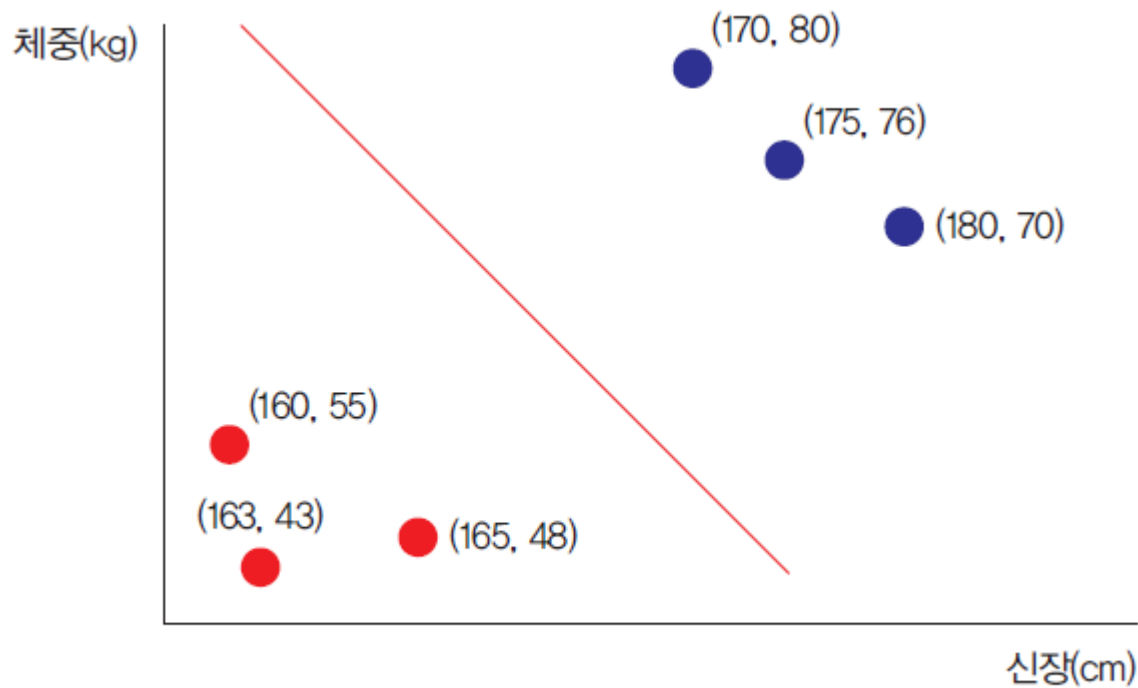


그림 5-15 신장과 체중으로 남녀를 구분하는 문제



Summary

- 딥러닝(deep learning)의 시작은 1950년대부터 연구되어 온 인공 신경망(artificial neural network: ANN)이다. ann: 가 가
- 신경망의 가장 큰 장점은 학습이 가능하다는 점이다. 데이터만 주어진다면 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.