

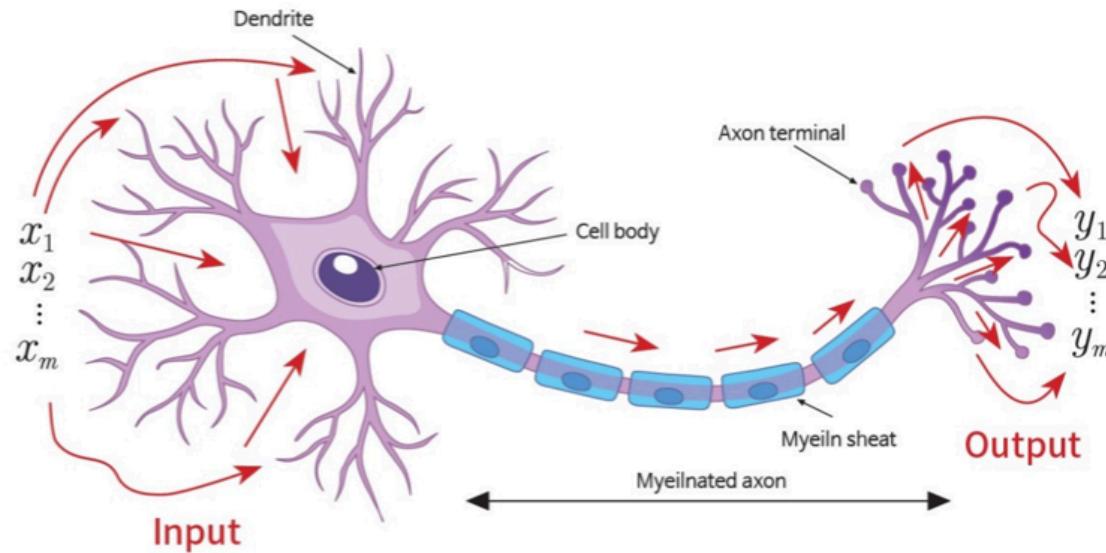
# Perceptron

# 내용

- 신경망
- 퍼셉트론의 이해
- 퍼셉트론의 작동원리
- 퍼셉트론의 한계점

# 신경망

- 딥러닝(deep learning)의 시작은 1950년대부터 연구되어 온 인공 신경망(artificial neural network: ANN)
- 인공 신경망은 생물학적인 신경망에서 영감을 받아서 만들어짐



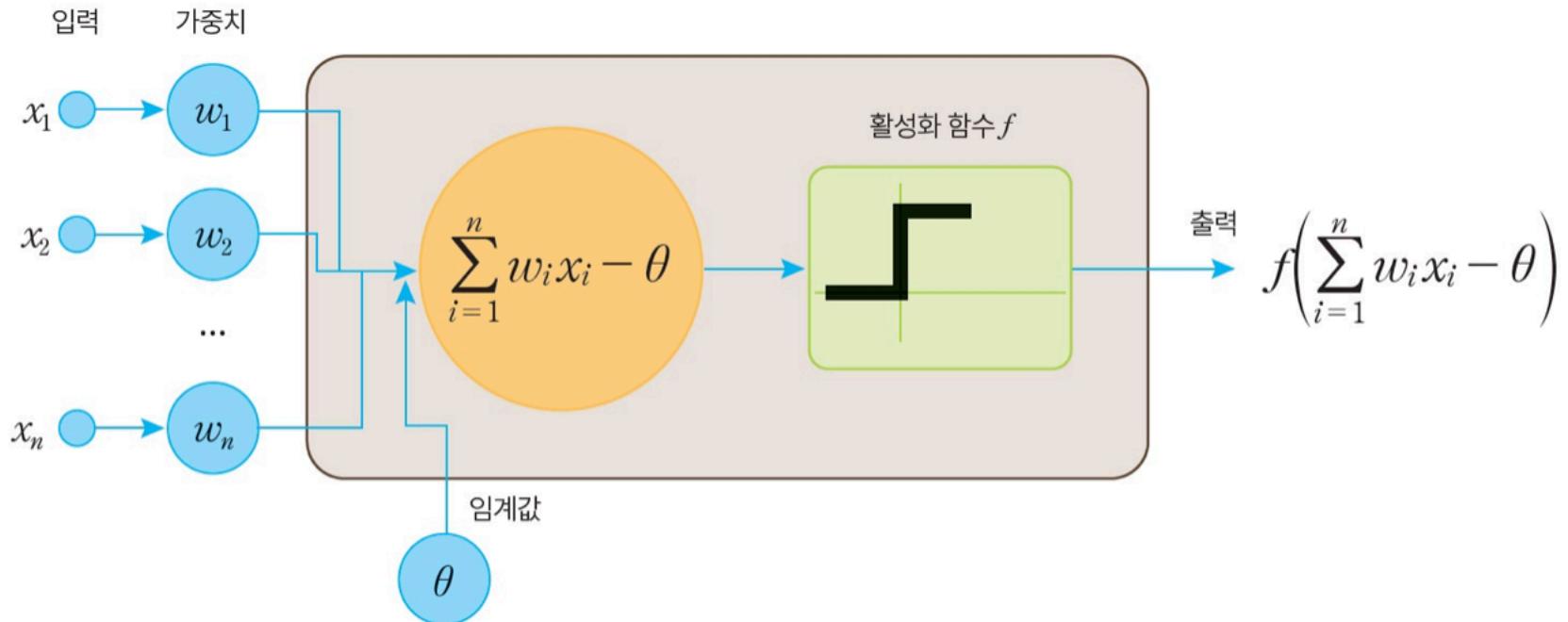
# 전통적인 컴퓨터 vs 인공신경망

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

# 신경망의 장단점

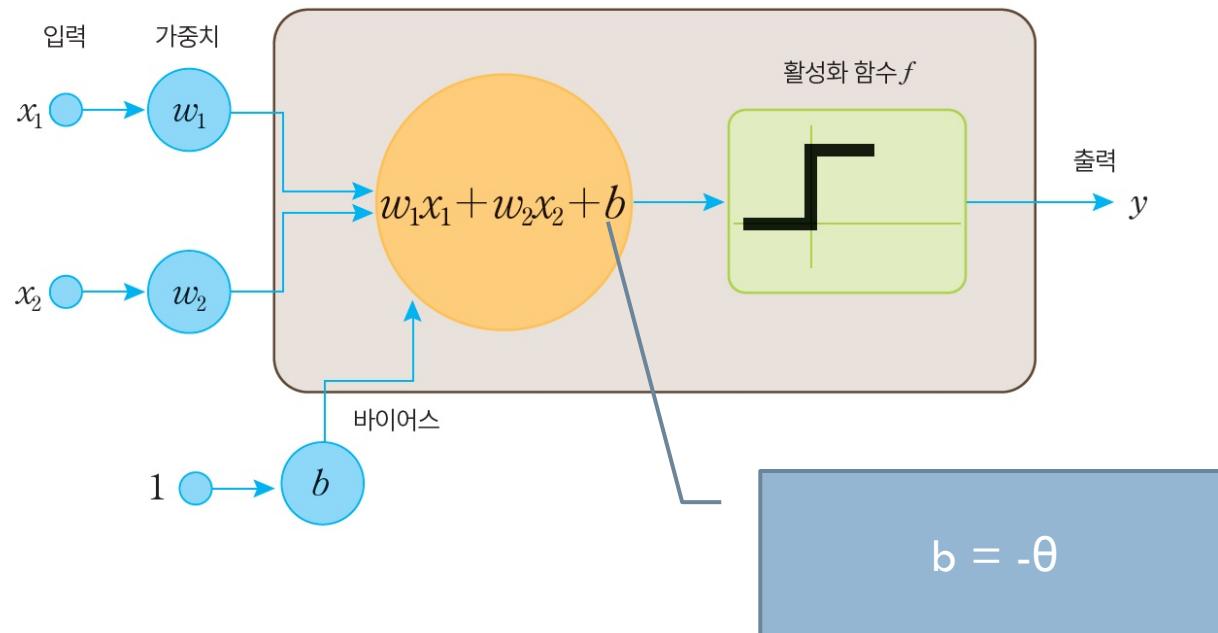
- 학습 : 주어진 데이터로부터 분류 또는 군집화 규칙을 도출
- 일부 오류가 있는 데이터에도 예측 값을 제공
- 주어진 데이터에 민감

# 뉴론의 수학적인 모델



# 퍼셉트론

- 퍼셉트론(perceptron)은 1957년에 Frank Rosenblatt가 단순 신경망

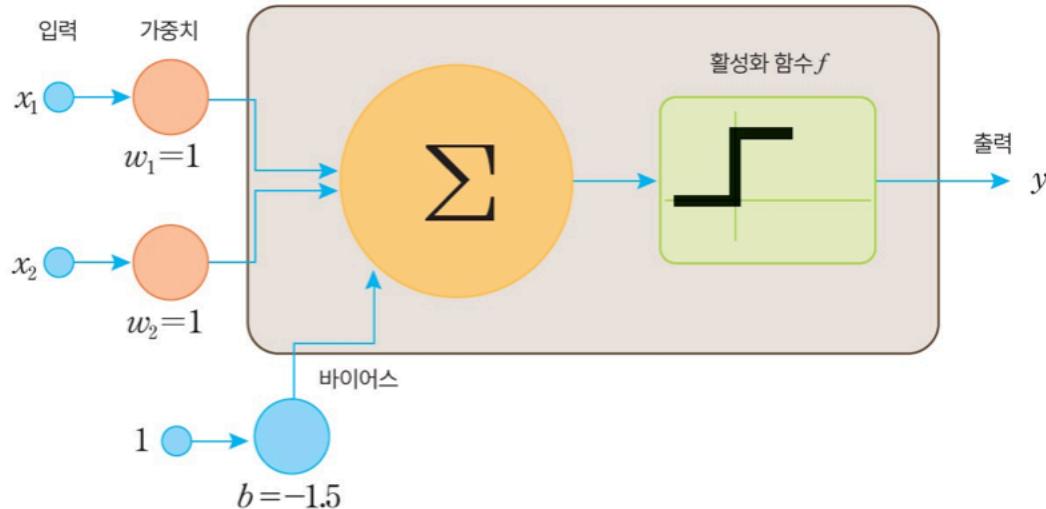


# 퍼셉트론

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

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

# 퍼셉트론의 논리 연산



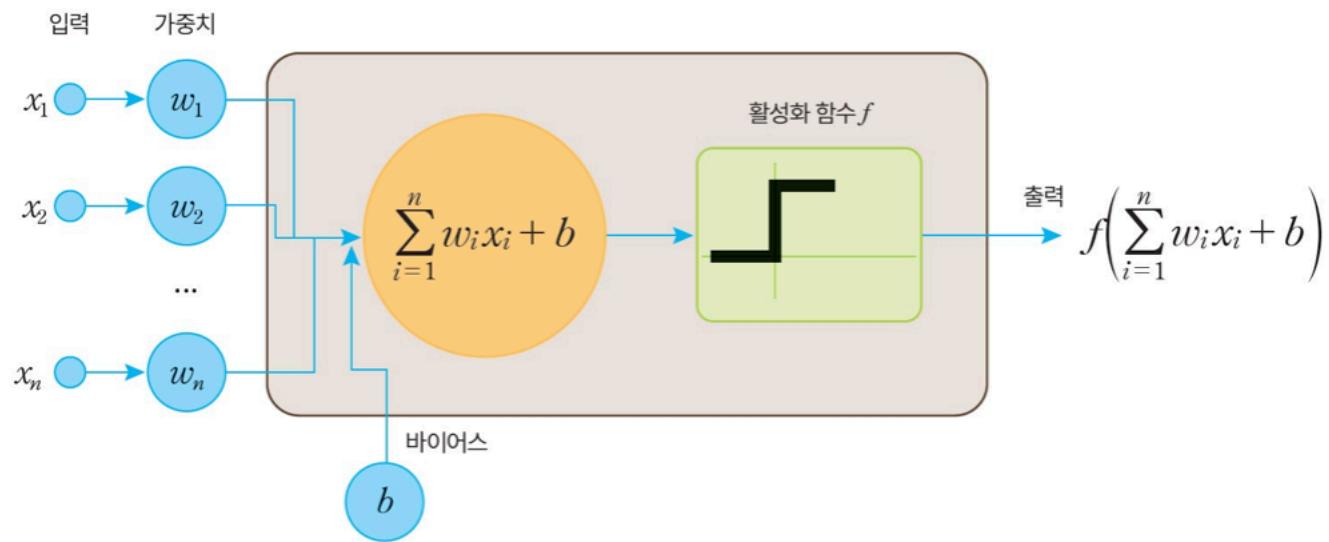
x1	x2	y
0	0	0
1	0	0
0	1	0
1	1	1

# 퍼셉트론의 논리 연산

x1	x2	$w_1x_1+w_2x_2$	b	$w_1x_1+w_2x_2+b$	출력
0	0	$1*0+1*0=0$	-1.5	-1.5	0
1	0	$1*1+1*0=1$	-1.5	-0.5	0
0	1	$1*0+1*1=1$	-1.5	-0.5	0
1	1	$1*1+1*1=2$	-1.5	+0.5	1

# 퍼셉트론 학습 알고리즘

- 퍼셉트론은 선형함수 모델이 가능



# 퍼셉트론 학습 알고리즘

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

- ① 모든 w들과 바이어스 b를 0 또는 작은 난수로 초기화한다.
- ② **while** (가중치가 변경되지 않을 때까지 반복)

③ 각 학습 데이터  $x^k$ 와 정답  $d^k$ 에 대하여

④  $y^k(t) = f(w \cdot x)$

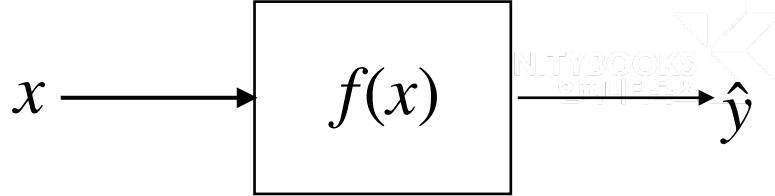
⑤ **if**  $d^k == y^k(t)$

⑥                   **continue**

⑦                   **else**

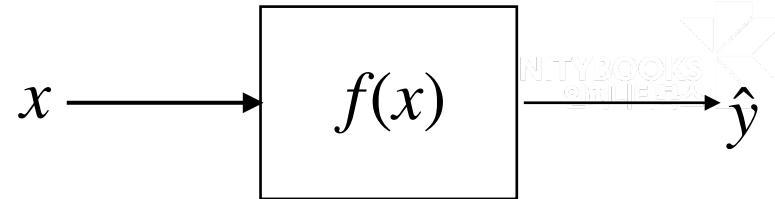
⑧                   모든 가중치  $w_i$ 에 대하여  $w_i(t+1) = w_i(t) + \eta \cdot (d^k - y^k(t)) \cdot x_i^k$

## 2-class 분류 문제 평가



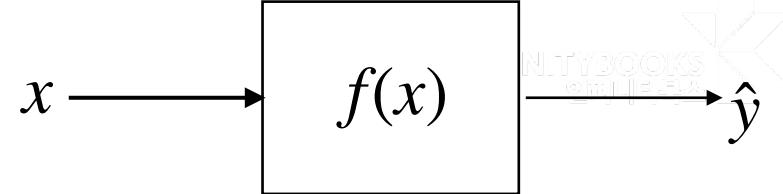
- 데이터셋  $D = \{(x_i, y_i) \mid i = 1, \dots, N\}$ 
  - 입력값  $x_i \in \mathbb{R}^2$
  - 실제값  $y_i \in \{0, 1\}$
- 학습된 모델  $\hat{y} = f(x)$ 
  - 입력값  $x \in \mathbb{R}^2$
  - 출력값  $\hat{y} \in \{0, 1\}$
- 데이터  $(x_i, y_i)$ 의 평가
  - 모델  $\hat{y}_i = f(x_i)$
  - 만약  $\hat{y}_i \neq y_i$  이면 오류가 발생

## 2-class 분류 문제 평가



- 객관적 평가를 여러 경우에 모델을 평가
- 데이터셋을 훈련과 테스트 데이터셋으로 분리  $D = D_{train} \cup D_{test}$ 
  - 훈련 데이터셋  $D_{train}, N_1 = |D_{train}|$
  - 테스트 데이터셋  $D_{test}, N_2 = |D_{test}|$
  - 나누는 비율은 8:2가 보편적으로 선택. 그러나 데이터셋 크기에 따라 다름
  - 임의 선택을 이용하여 데이터셋을 나눔
- 주어진 데이터셋에서 반복적으로 평가를 진행
  - 훈련과 테스트 데이터셋에 대해 동일하게 평가
  - k-way 교차 평가: 훈련과 테스트 데이터셋으로 나누어 k번 평가 진행
  - k-way 검증 교차 평가: 훈련, 검증 과 테스트 데이터셋으로 나누어 k번 평가 진행

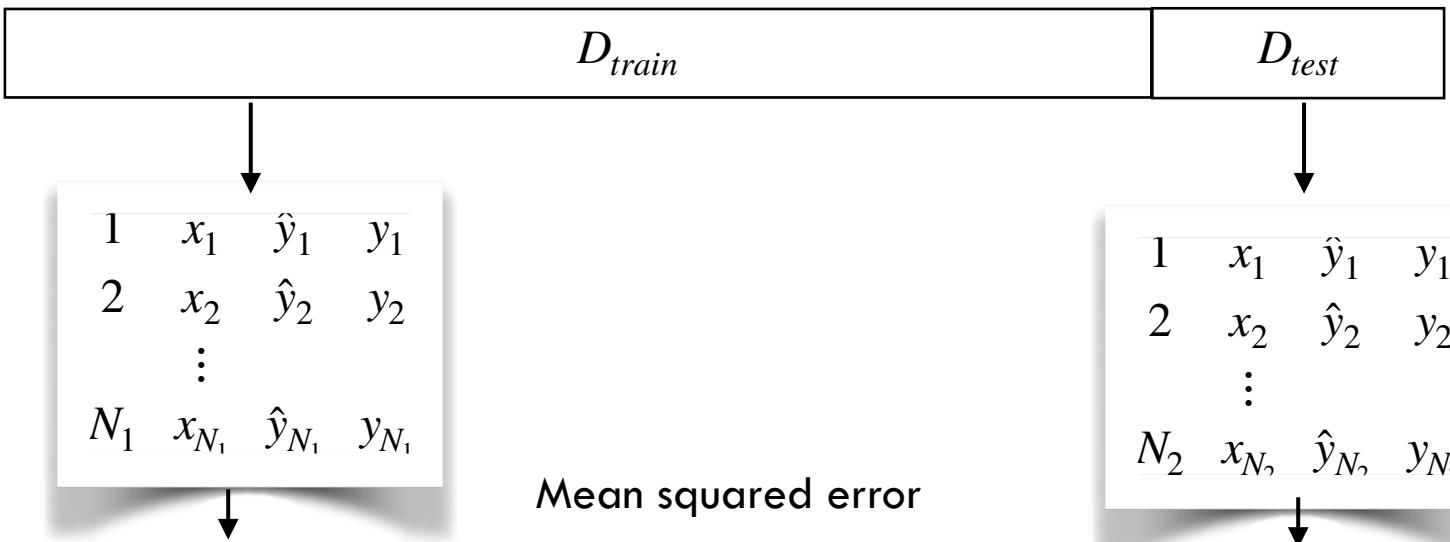
# 2-class 분류 문제 평가



$$D = \{(x_i, y_i) \mid i = 1, \dots, N\}$$

--	--	--	--	--

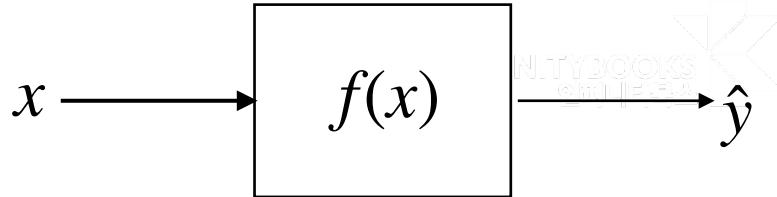
$$D = D_{train} \cup D_{test}$$



$$\text{Accuracy}_{train} = \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbb{1}(y_i = \hat{y}_i)$$

$$\text{Accuracy}_{test} = \frac{1}{N_2} \sum_{i=1}^{N_2} \mathbb{1}(y_i = \hat{y}_i)$$

# 5-way 2-class 분류 문제 평가



- Accuracy와 error의 관계

$$\text{Accuray} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(y_i = \hat{y}_i) \quad \text{Error} = 1 - \text{Accuray}$$

- 테스트셋 선택이 가능한 모든 조합에 대해 평가
- 모델 최종 평가는 모든 평가의 평균임
- 5-way 교차 검증의 경우 훈련 데이터셋에서 검증 데이터셋을 일부 선택
  - 훈련 모델은 검증 데이터셋을 제외한 데이터로 구성
  - 훈련 모델을 검증 데이터셋으로 평가 후 가장 성능이 높은 훈련 모델을 선택하여 테스트셋을 평가

# 논리 연산자 AND 학습 과정

에포크(epoch)	입력		원하는 출력	실제 출력	오차	변경된 가중치		
	$x_1$	$x_2$	$d$	$y$	$d - y$	$w_0(b)$	$w_1$	$w_2$
1	0	0	0	1	-1	-0.1	0.0	0.0
	0	1	0	0	0	-0.1	0.0	0.0
	1	0	0	0	0	-0.1	0.0	0.0
	1	1	1	0	1	0.0	0.1	0.1
2	0	0	0	1	-1	-0.1	0.1	0.1
	0	1	0	1	-1	-0.2	0.1	0.0
	1	0	0	0	0	-0.2	0.1	0.0
	1	1	1	0	1	-0.1	0.2	0.1
3	0	0	0	0	0	-0.1	0.2	0.1
	0	1	0	1	-1	-0.2	0.1	0.0
	1	0	0	1	-1	-0.3	0.1	0.0
	1	1	1	0	1	-0.2	0.2	0.1
4	0	0	0	0	0	-0.2	0.2	0.1
	0	1	0	0	0	-0.2	0.2	0.1
	1	0	0	0	0	-0.2	0.2	0.1
	1	1	1	1	1	-0.2	0.2	0.1

# 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]

# 퍼셉트론 프로그래밍

```
# 뉴론의 출력 계산 함수
def calculate(input):
    global weights
    global bias
    activation = bias          # 바이어스
    for i in range(2):          # 입력신호 총합 계산
        activation += weights[i] * input[i]
    if activation >= 0.0:        # 스텝 활성화 함수
        return 1.0
    else:
        return 0.0
```

# 퍼셉트론 프로그래밍

```
# 학습 알고리즘
def train_weights(X, y, l_rate, n_epoch):
    global weights
    global bias
    for epoch in range(n_epoch):                      # 에포크 반복
        sum_error = 0.0
        for row, target in zip(X, y):                  # 데이터셋을 반복
            actual = calculate(row)                   # 실제 출력 계산
            error = target - actual                  # 실제 출력 계산
            bias = bias + l_rate * error
            sum_error += error**2                    # 오류의 제곱 계산
            for i in range(2):                      # 가중치 변경
                weights[i] = weights[i] + l_rate * error * row[i]
            print(weights, bias)
    print('에포크 번호=%d, 학습률=%.3f, 오류=%.3f' % (epoch, l_rate, sum_error))
return weights
```

# 퍼셉트론 프로그래밍

```
# AND 연산 학습 데이터셋, 샘플과 레이블이다.  
X = [[0,0],[0,1],[1,0],[1,1]]  
y = [0, 0, 0, 1]  
  
# 가중치와 바이어스 초기값  
weights = [0.0, 0.0]  
bias = 0.0  
  
l_rate = 0.1          # 학습률  
n_epoch = 5           # 에포크 횟수  
weights = train_weights(X, y, l_rate, n_epoch)  
print(weights, bias)
```

## 실행결과

[0.0, 0.0] -0.1

[0.0, 0.0] -0.1

[0.0, 0.0] -0.1

[0.1, 0.1] 0.0

에포크 번호=0, 학습률=0.100, 오류=2.000

[0.1, 0.1] -0.1

[0.1, 0.0] -0.2

[0.1, 0.0] -0.2

[0.2, 0.1] -0.1

에포크 번호=1, 학습률=0.100, 오류=3.000

[0.2, 0.1] -0.1

[0.2, 0.0] -0.2

[0.1, 0.0] -0.30000000000000004

[0.2, 0.1] -0.20000000000000004

에포크 번호=2, 학습률=0.100, 오류=3.000

[0.2, 0.1] -0.20000000000000004

[0.2, 0.1] -0.20000000000000004

[0.2, 0.1] -0.20000000000000004

[0.2, 0.1] -0.20000000000000004

에포크 번호=3, 학습률=0.100, 오류=0.000

[0.2, 0.1] -0.20000000000000004

[0.2, 0.1] -0.20000000000000004

[0.2, 0.1] -0.20000000000000004

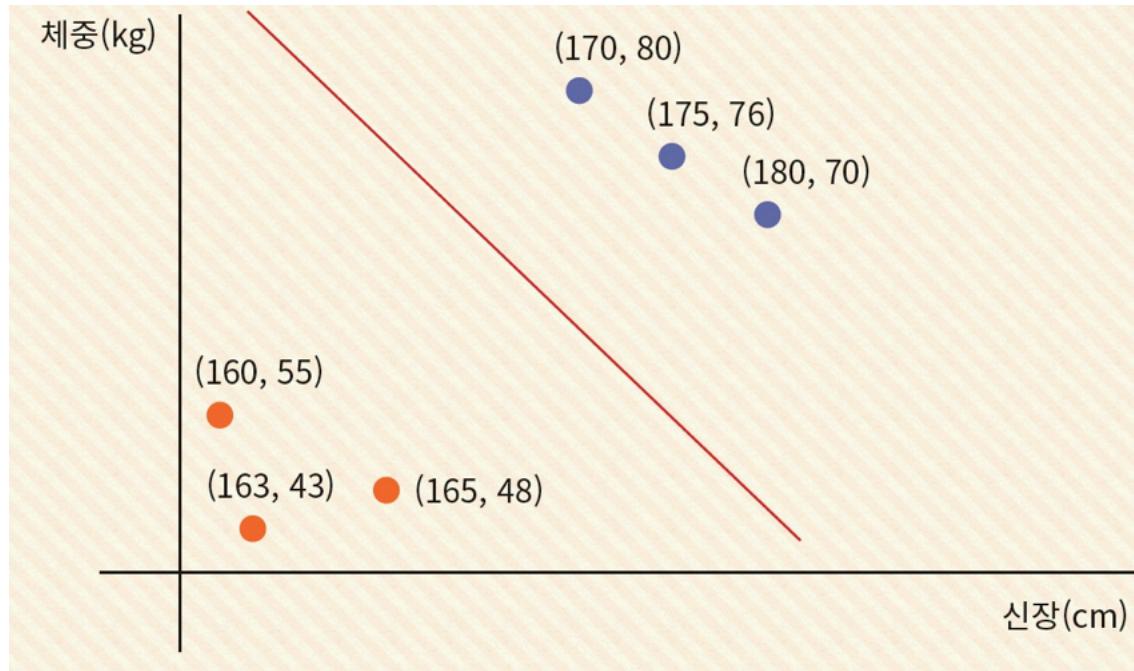
[0.2, 0.1] -0.20000000000000004

에포크 번호=4, 학습률=0.100, 오류=0.000

[0.2, 0.1] -0.20000000000000004

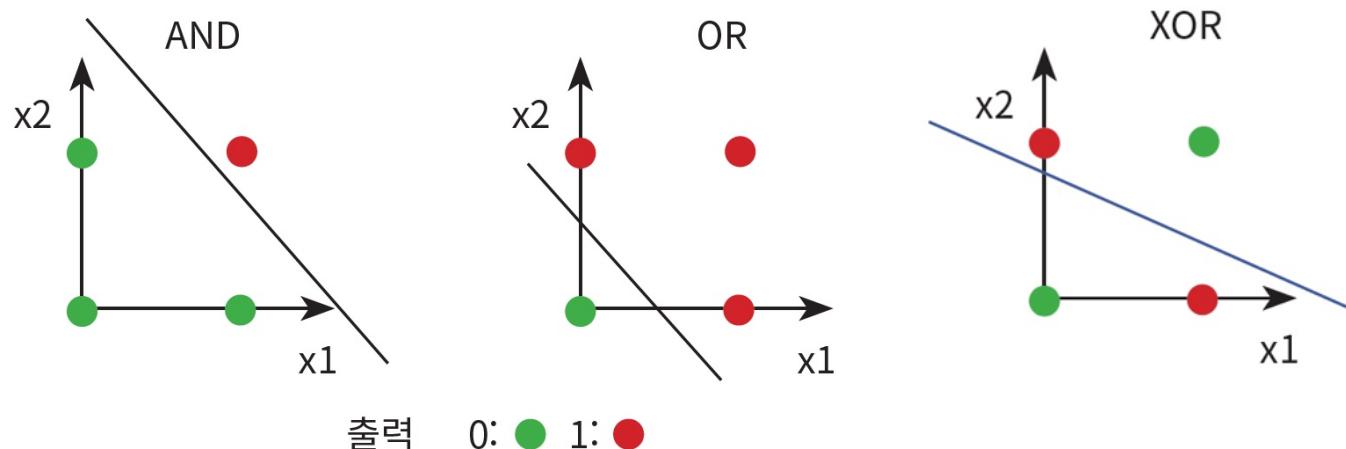
# Lab: 퍼셉트론으로 분류

- 대학생들의 신장과 체중을 받아서 성별을 출력하는 퍼셉트론을 학습



# 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



# XOR 학습 문제

```
...
# XOR 연산 학습 데이터셋
# 샘플과 레이블이다.
X = [[0,0],[0,1],[1,0],[1,1]]
y = [0, 1, 1, 0]

weights = [0.0, 0.0]
bias = 0.0

l_rate = 0.1          # 학습률
n_epoch = 100         # 에포크 횟수
weights = train_weights(X, y, l_rate, n_epoch)
print(weights, bias)
```

# 실행결과

...

에포크 번호=97, 학습률=0.100, 오류=4.000

[-0.1, 0.0] -0.1

[-0.1, 0.1] 0.0

[0.0, 0.1] 0.1

[-0.1, 0.0] 0.0

에포크 번호=98, 학습률=0.100, 오류=4.000

[-0.1, 0.0] -0.1

[-0.1, 0.1] 0.0

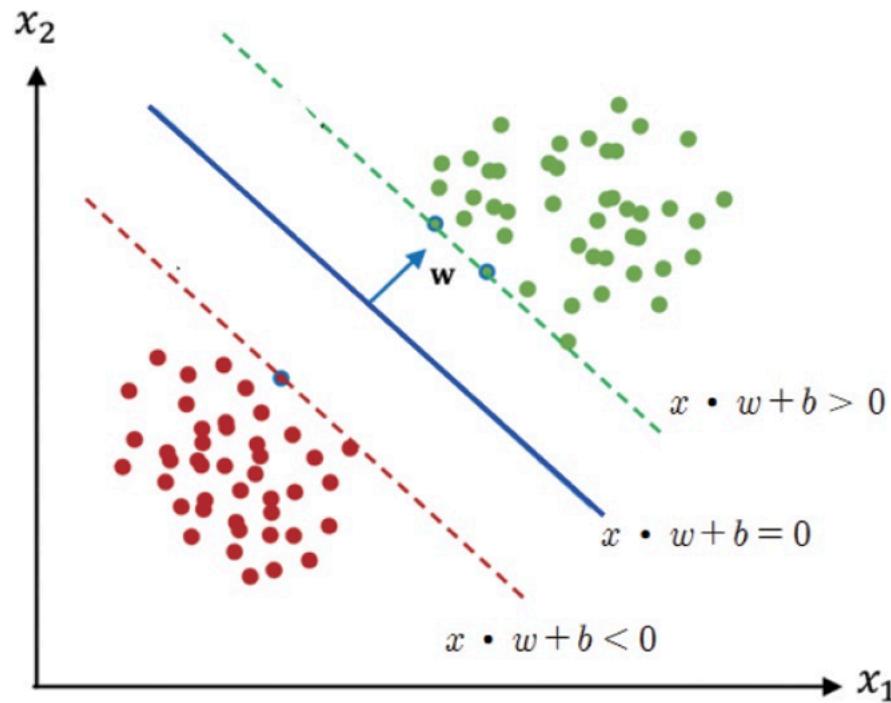
[0.0, 0.1] 0.1

[-0.1, 0.0] 0.0

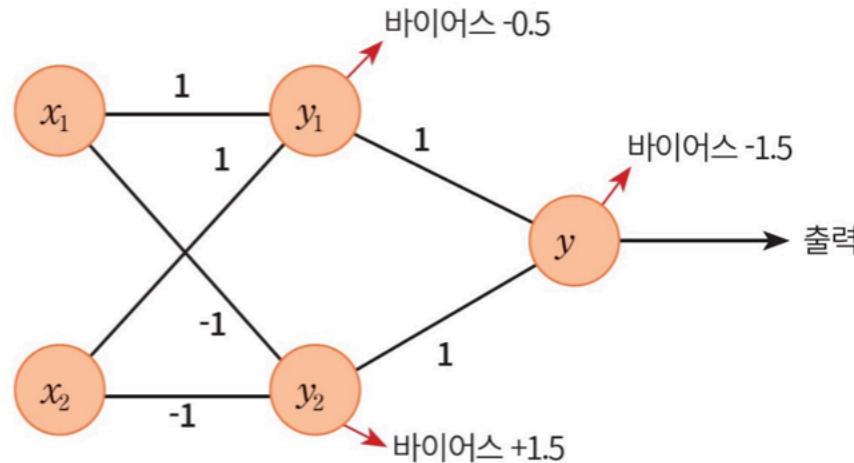
에포크 번호=99, 학습률=0.100, 오류=4.000

# 선형 분류 가능 문제

- 퍼셉트론은 직선을 이용하여 입력 패턴을 분류하는 선형 분류기(linear classifier)임



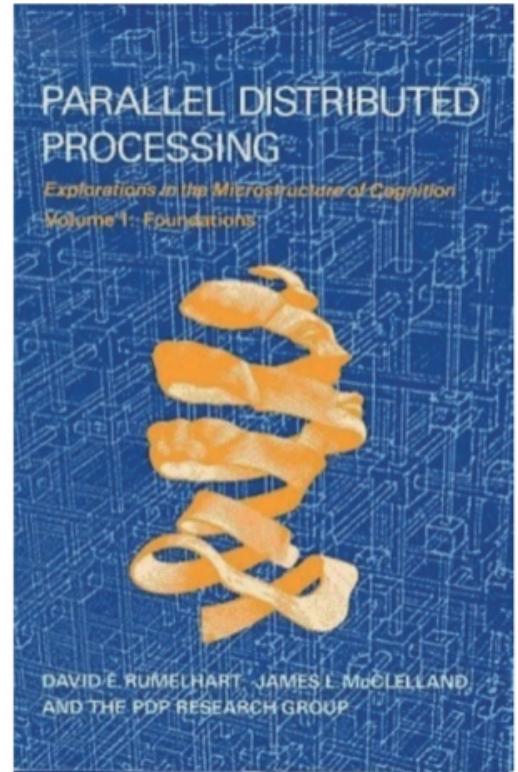
# 다층 퍼셉트론으로 XOR 문제를 해결



$x_1$	$x_2$	$y_1$	$y_2$	$y$	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

# 다층 퍼셉트론의 학습 알고리즘

- Minsky와 Papert는 다층 퍼셉트론을 학습시키는 알고리즘을 찾기가 아주 어려울 것이라고 예언
- 1980년대 중반에 Rumelhart와 Hinton 등은 다층 퍼셉트론을 위한 학습 알고리즘이 개발됨



# 정리

- 신경망의 가장 큰 장점은 학습이 가능
- 뉴론은 다른 뉴론들로부터 신호를 받아서 모두 합한 후에 비선형 함수를 적용하여 출력을 계산
- 연결선은 가중치를 가지고 있고 이 가중치에 학습의 결과를 저장
- 퍼셉트론은 하나의 뉴론만을 사용하는 다수의 입력을 받아서 하나의 신호를 출력
- 퍼셉트론은 AND나 OR 같은 논리적인 연산을 학습할 수 있었지만 XOR 연산은 학습할 수 없음
- 선형 분리 가능한 문제만 해결 가능