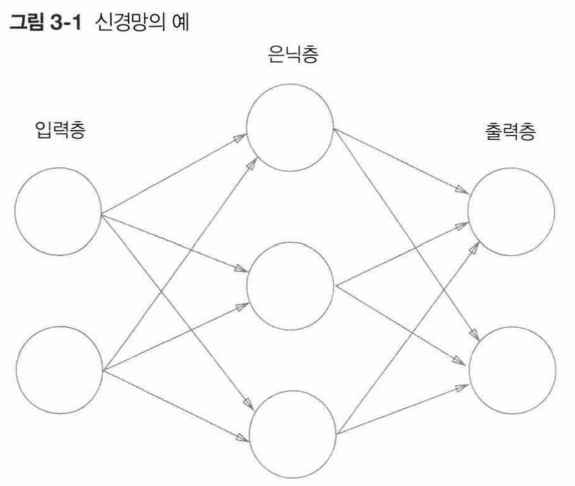
Chapter 3 신경망

3.1 퍼셉트론에서 신경망으로

3.1.1 신경망의 예

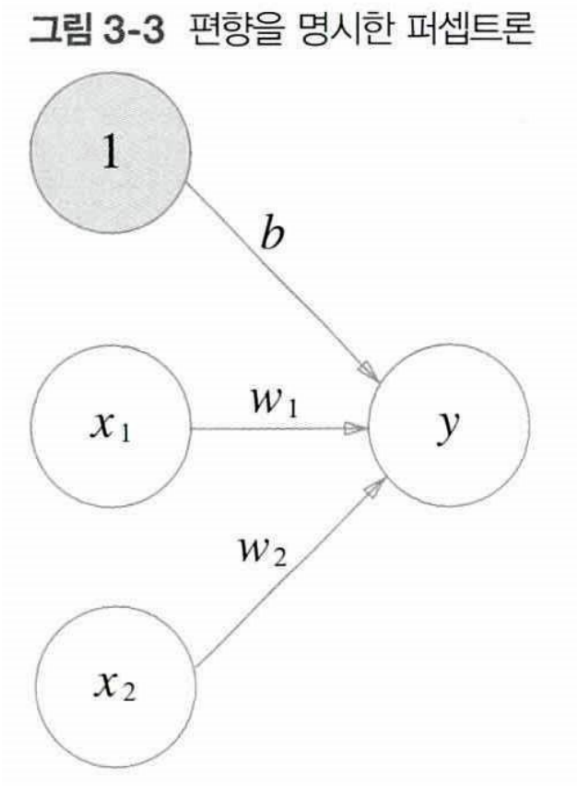
- 신경망은 입력층, 출력층, 은닉층으로 구성되며 은닉층의 뉴런은 사람 눈에 보이지 않는다.

그림 3-1 신경망의 예



3.1.2 퍼셉트론 복습

그림 3-3 편향을 명시한 퍼셉트론



- h(x) : 0을 넘으면 1을 출력하고 그렇지 않으면 0을 출력하는 함수

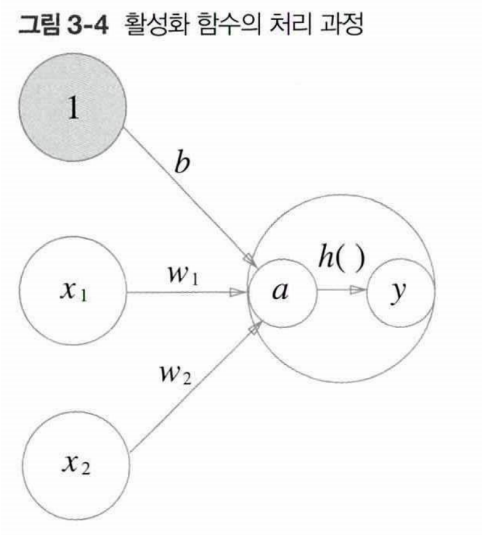
$$y = h(b + w_1 x_1 + w_2 x_2)$$

$$h(x) = \begin{cases} 0 \ (x \leq 0) \\ 1 \ (x > 0) \end{cases}$$

3.1.3 활성화 함수의 등장

- 활성화 함수(activation function) : 입력 신호의 총합을 출력 신호로 변환화는 함수

그림 3-4 활성화 함수의 처리 과정



- 뉴런 = 노드(책 기준) → 큰 동그라미를 가리킨다.

3.2 활성화 함수

- 계단함수(step function) : 임계값(threshold)를 경계로 출력이 바뀌는 함수

3.2.1 시그모이드 함수(sigmoid function)

$$h(x) = \frac{1}{1 + exp(-x)}$$

- 퍼셉트론과 신경망의 차이는 활성화 함수 뿐이다.

3.2.2 계단 함수 구현하기

```
#계단함수 구현하기
def step_function(x):
  if x > 0 :
    return 1
  else:
    return 0
```

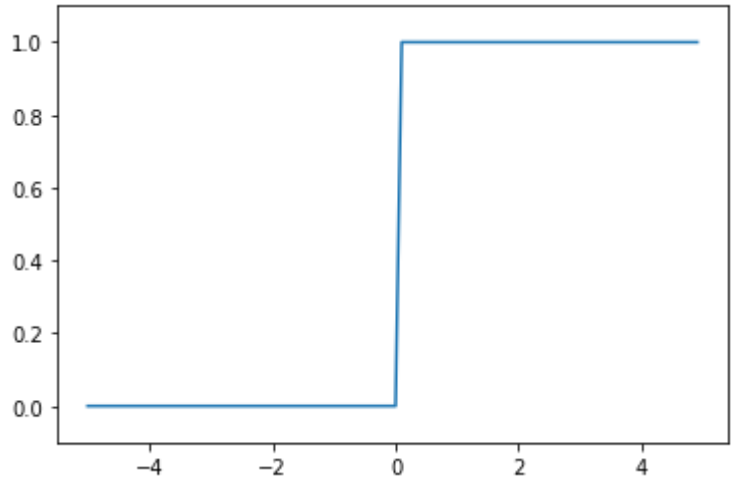
- 위 구현은 간단하지만 넘파이 배열을 인수로 넣을 수 없음(아래와 같이 수정하자)

```
# 계단함수 업그레이드(넘파이 포함)
def step_function(x):
  y = x > 0
  return y.astype(np.int)
```

3.2.3 계단 함수의 그래프

```
#계단함수 그래프 확인
import numpy as np
import matplotlib.pylab as plt
def step_function(x):
  return np.array(x > 0, dtype = np.int)

x = np.arange(-5.0, 5.0, 0.1)
y = step_function(x)
plt.plot(x,y)
plt.ylim(-0.1, 1.1), #y축의 범위 지정
plt.show()
```

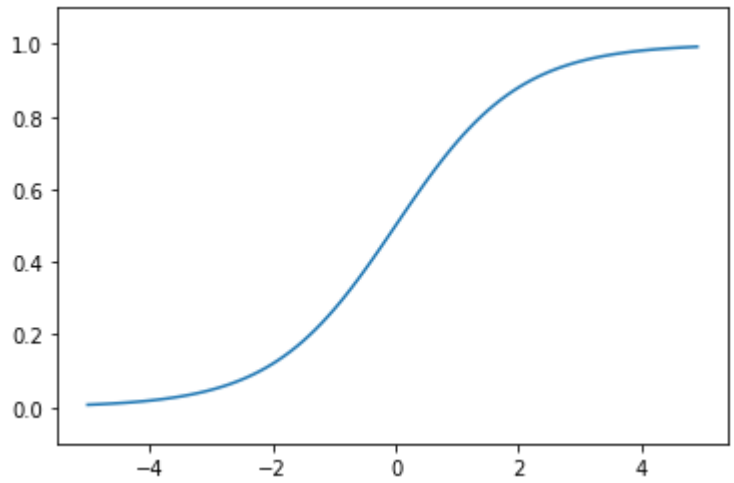


3.2.4 시그모이드 함수 구현하기

```
def sigmoid(x):
  return 1 / (1 + np.exp(-x))
```

- 계단함수와 다르게 인수가 넘파이 배열이여도 잘 출력됨
- → 넘파이의 브로드캐스트 기능 : 넘파이 배열과 스칼라값의 연산을 넘파이의 배열의 원소 각각과 스칼라값의 연산으로 바꿔 수행하는 것!

```
# 시그모이드 함수 시각화
x = np.arange(-5.0, 5.0, 0.1)
y = sigmoid(x)
plt.plot(x, y)
plt.ylim(-0.1, 1.1) #y축 범위 지정
plt.show()
```



3.2.5 시그모이드 함수와 계단 함수 비교

차이점 : 매끄러움의 차이→ 출력 값이 확연히 다르다

공통점 : 입력이 아무리 작거나 커도 출력은 0에서 1사이의 값을 반환한다.

3.2.6 비선형 함수