3주차(3/3)

활성화 함수

파이썬으로 배우는 기계학습

한 동 대 학 교
김영섭 교수

활성화 함수

- 학습 목표
  - 활성화 함수의 역할을 이해한다.
  - 다양한 활성화 함수를 익힌다.

- 학습 내용
  - 시그모이드 함수
  - 계단 함수
  - 쌍곡탄젠트 함수
  - 렐루 함수

체온 변환기

- 섭씨를 화씨로 변환하는 수식(1)

$$F = \frac{9}{5}C + 32$$

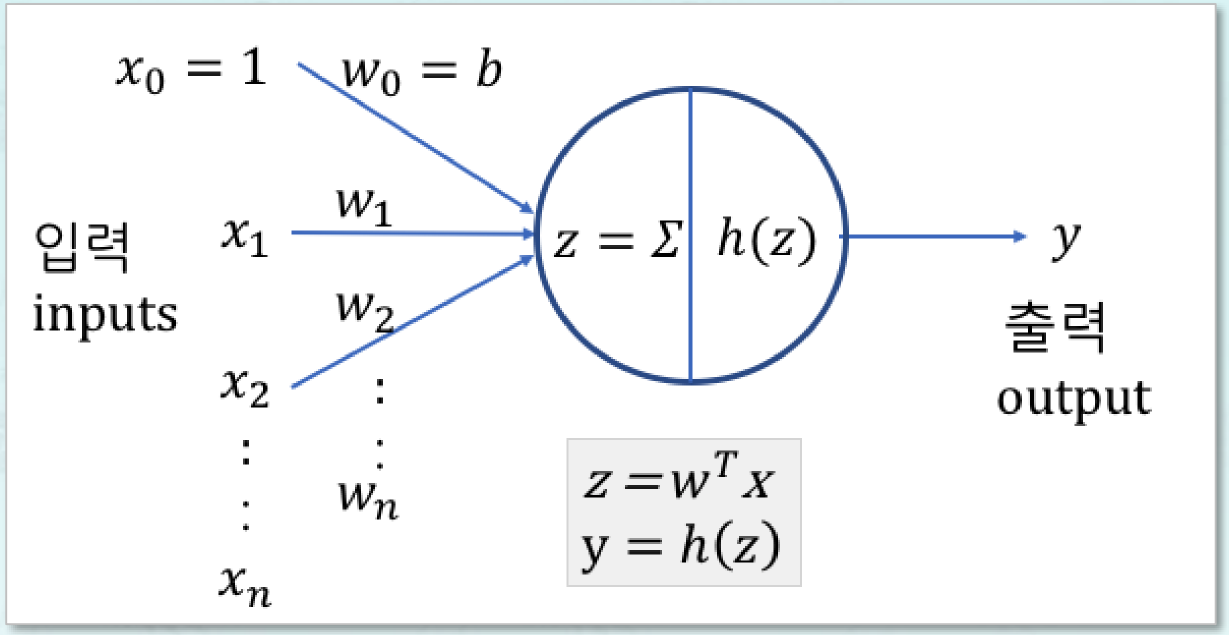
$$F = \begin{cases} \frac{9}{5}C + 32 & \text{if } C \geq 0 \\ 32 & \text{otherwise} \end{cases}$$

- 뉴론의 수식(2)

$$z = w_0 x_0 + w_1 x_1$$

$$\text{단}, x_0 = 1, w_0 = b$$

$$y = h(z)$$



체온 변환기

- 섭씨를 화씨로 변환하는 수식(1)

$$F = \frac{9}{5}C + 32$$

$$F = \begin{cases} \frac{9}{5}C + 32 & \mathit{if}\ C \geq 0 \\ 32 & \mathit{otherwise} \end{cases}$$

- 뉴론의 수식(2)

$$z = w_0 x_0 + w_1 x_1$$

$$\text{단}, x_0 = 1, w_0 = b$$

$$y = h(z)$$

- 체온 변환기 수식(3)

$$z = 32 + \frac{9}{5}x_1$$

$$y = h(z) \quad \begin{cases} 32 & \text{if } z < 32). \\ z & \text{if } z \geq 32). \end{cases}$$

체온 변환기 C2F 뉴론 구현

- 체온 변환기 뉴론 **C2F**

```python
def activate(z):
    """returns 32 if z < 32"""
    if z < 32 :
        z = 32
    return z
```

```python
def C2F(C):
    """ converts Celcius to Fahrenheit"""
    F = 9/5.0 * C + 32
    return activate(F)
```

- 체온 변환기 수식**(3)**

$$z = 32 + \frac{9}{5}x_1$$

$$y = h(z) \quad \begin{cases} 32 & \text{if } z < 32). \\ z & \text{if } z \geq 32). \end{cases}$$

활성화 함수

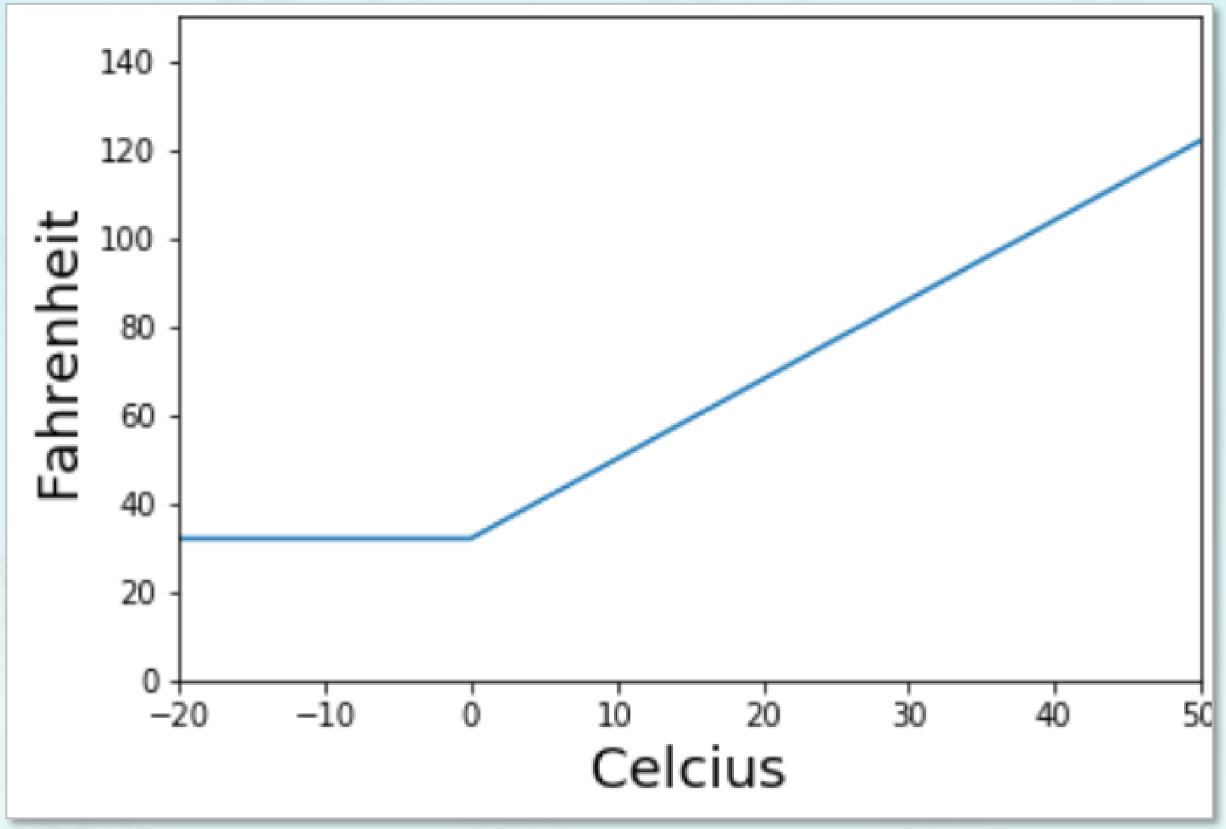
체온 변환기 C2F 뉴론 테스트

```
test_c = [-20, -10, 0, 36.5, 40, 50, 100]
test_f = [ C2F(c) for c in test_c ]
print(test_f)
```

```
[32, 32, 32.0, 97.7, 104.0, 122.0, 212.0]
```

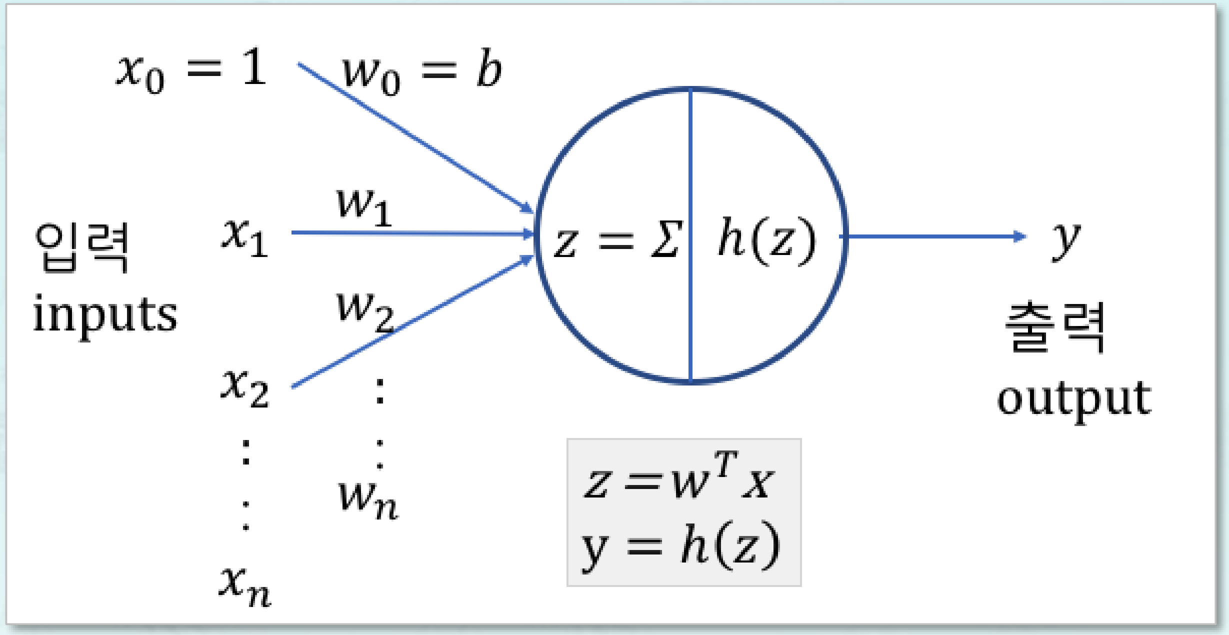
체온 변환기 C2F 뉴론 테스트

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

# Plotting the simple neuron
x = np.arange(-100, 100, .1)
y = [C2F(ix) for ix in x]

plt.figure()
plt.plot(x, y)
plt.axis([-20, 50, 0, 150])
plt.xlabel('Celcius')
plt.ylabel('Fahrenheit')
plt.show()
```



활성화 함수

- JoyPop 퀴즈:
- 그런데, 김 교수는 팔리지도 않을 체온변환기는 왜 만들었는가?
  (1) 입력 x
  (2) 가중치 w
  (3) 편향 b
  (4) 순입력 z
  (5) 활성화 함수 h(z)
  (6) 출력 y



- 큰소리로 답 외치기
  - 하나, 둘, 셋할 때, 다같이

활성화 함수 – 시그모이드(sigmoid)

$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

- e : 자연 상수, **2.7182...**
- 단순한 함수

- $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{1}{1+\frac{1}{e^x}}$
- $\sigma(0) = \frac{1}{1+\frac{1}{e^0}} = \frac{1}{2}$
- $\sigma(x \rightarrow \infty) = \frac{1}{1+\frac{1}{e^\infty}} = 1$
- $\sigma(x \rightarrow -\infty) = \frac{1}{1+e^\infty} = 0$

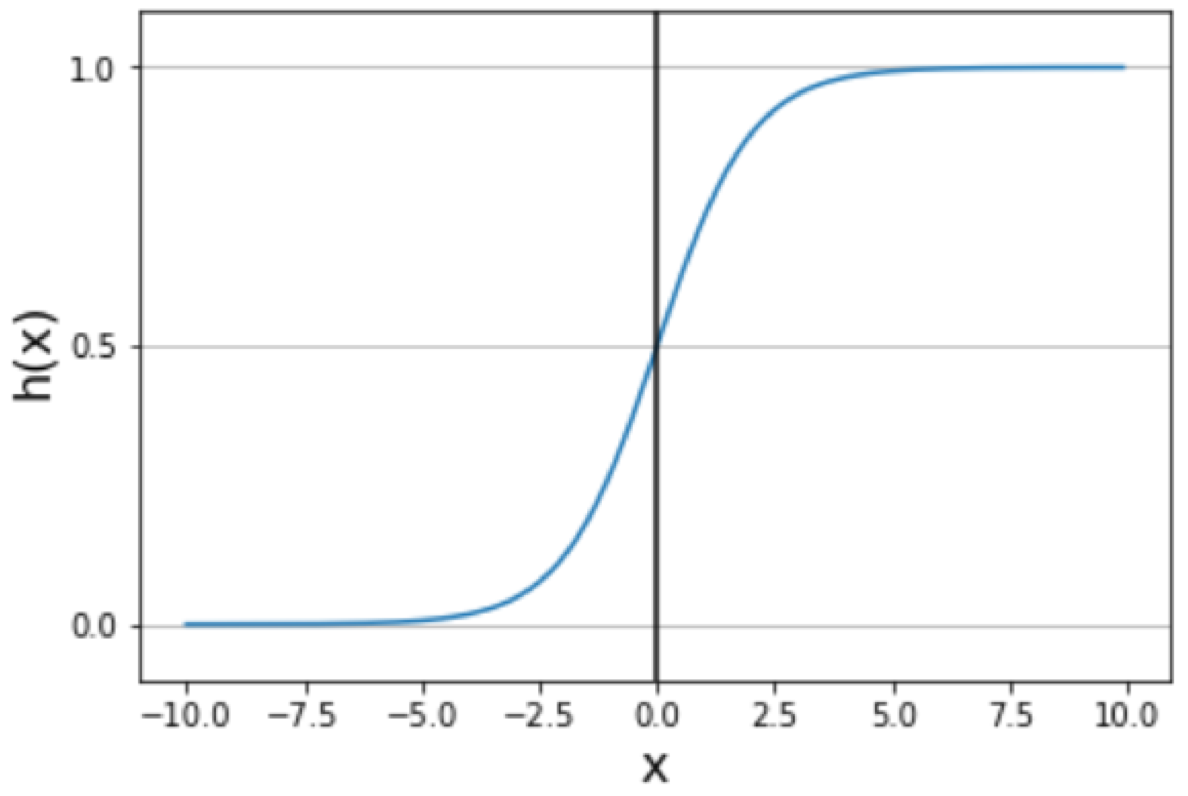
활성화 함수 – 시그모이드(sigmoid)

$$sigmoid(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$

```python
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

활성화 함수 – 시그모이드(sigmoid)

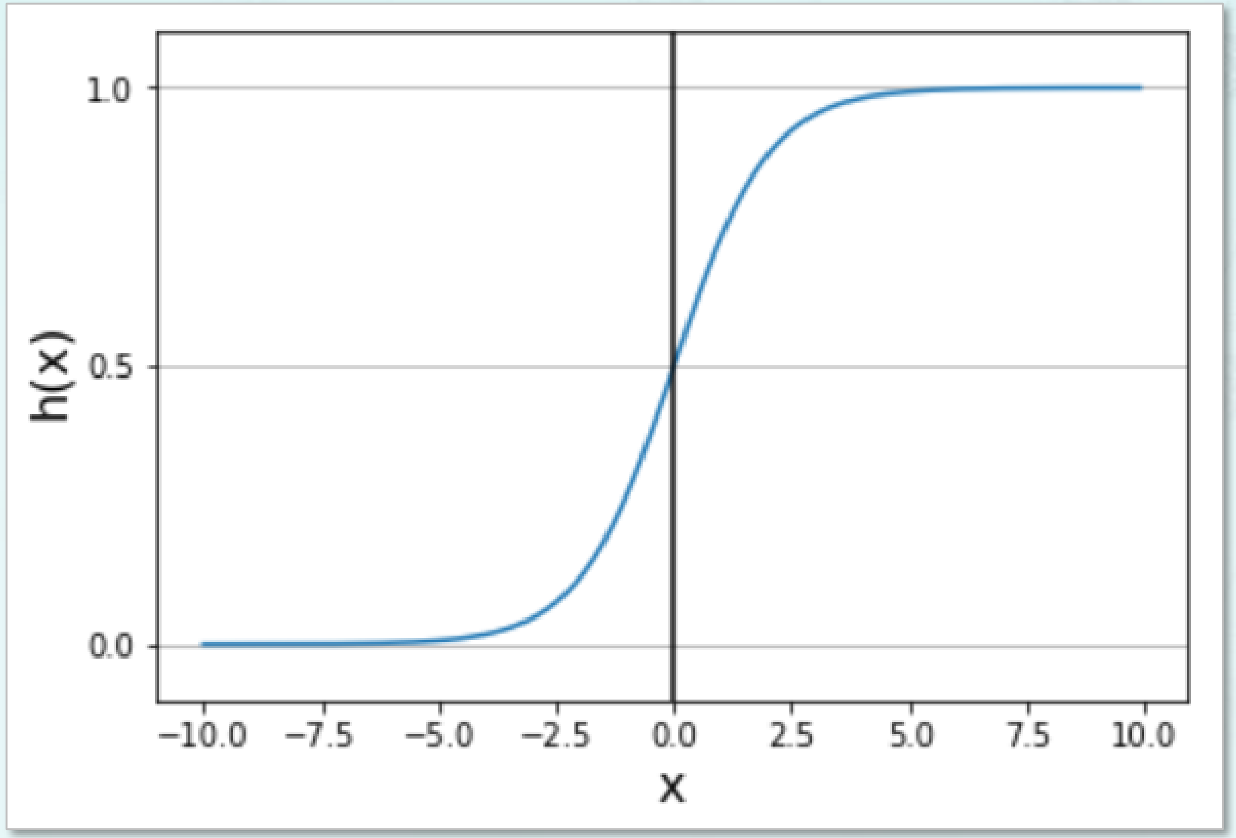
$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$




```
x = np.arange(-10.0, 10.0, 0.1)
y = sigmoid(x)
plt.plot(x,y)
plt.axvline(0, color='black')
plt.xlabel('x', fontsize=16)
plt.ylabel('h(x)', fontsize=16)
plt.ylim(-0.1, 1.1)
plt.yticks([0.0, 0.5, 1.0])
plt.grid(axis='y')
plt.show()
```

활성화 함수 – 시그모이드(sigmoid)

$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$



- 입력 값을 **0**보다 크고 **1**보다 작은 미분 가능한 수로 변환
- 로지스틱 분류, 비용함수에 사용
- 출력값이 **0**과 **1**사이

활성화 함수 – 시그모이드(sigmoid)

- 미분 공식

$$(e^x)' = e^x \quad ①$$

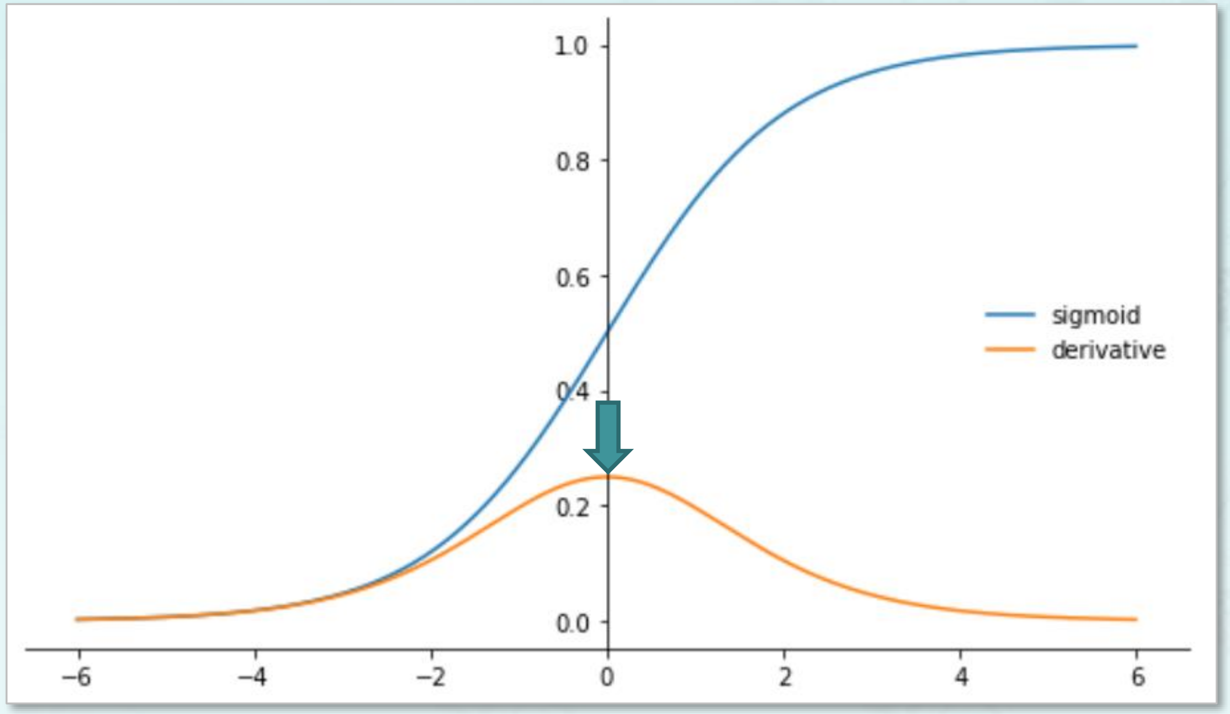
$$(e^{-x})' = -e^{-x} \quad ②$$

$$\frac{du^n}{dx} = nu^{n-1}\frac{du}{dx} \quad ③$$

$$\begin{aligned}
\frac{d}{dx}sigmoid(x) &= \frac{d}{dx}(1+e^{-x})^{-1} \\
&\overset{③}{=} (-1)\frac{1}{(1+e^{-x})^2}\frac{d}{dx}(1+e^{-x}) \\
&\overset{②}{=} (-1)\frac{1}{(1+e^{-x})^2}(0-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \\
&= \frac{(1+e^{-x})}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right) \\
&= sigmoid(x)(1-sigmoid(x))
\end{aligned}$$

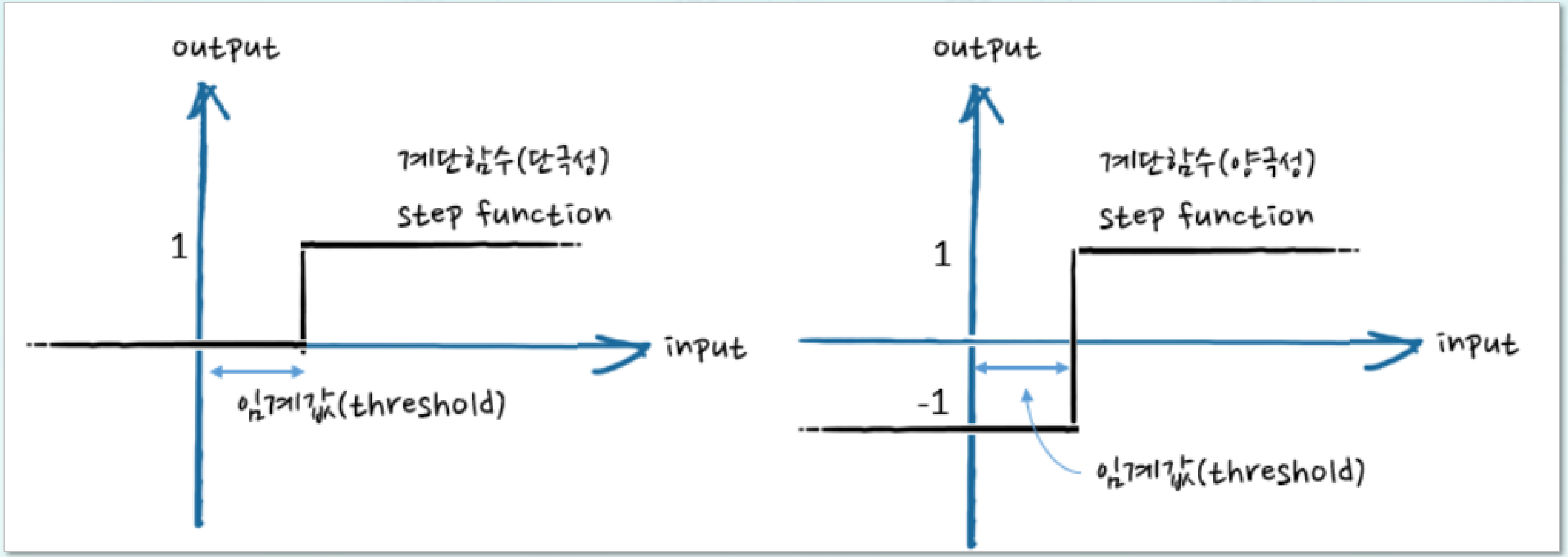
활성화 함수 – 시그모이드(sigmoid)

- 시그모이드 함수와 도함수



- 미분계수 – 최대값 **0.25**

활성화 함수 – 계단 함수



output
계단함수(단극성)
step function
1
input
임계값(threshold)
output
계단함수(양극성)
step function
1
-1
input
임계값(threshold)

활성화 함수 – 계단 함수 구현

- 단극성 및 양극성 계단함수

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \ldots$$

$$y = h(z)$$

$$h(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

단극성

$$h(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

양극성

```python
def step(x):
    if x >= 0:
        return 1
    else:
        return 0
```

```python
print('step(3) = ',step(3))
```

```
step(3) =  1
```

```python
z = step(np.array([-1, 2, 3]))
print('step([-1, 2, 3]) = ', z)
```

활성화 함수 – 계단 함수 구현

- 단극성 및 양극성 계단함수

```python
z = step(np.array([-1, 2, 3]))
print('step([-1, 2, 3]) = ', z)
```

```
----------------------------------------
ValueError
<ipython-input-22-1f7da1ffc932> in <modu
----> 1 z = step(np.array([-1, 2, 3]))
      2 print('step([-1, 2, 3]) = ', z)

<ipython-input-14-2622e94088b1> in step(
      1 def step(x):
----> 2     if x >= 0:
      3         return 1
      4     else:
      5         return 0

ValueError: The truth value of an array
us. Use a.any() or a.all()
```

```python
def step(x):
    if x >= 0:
        return 1
    else:
        return 0
```

버그

```python
print('step(3) = ',step(3))
```

```
step(3) =  1
```

```python
z = step(np.array([-1, 2, 3]))
print('step([-1, 2, 3]) = ', z)
```

활성화 함수 – 계단 함수 구현

- 넘파이의 불린 인덱싱 이용

```
x = np.array([-1, 2, 3])
print(x > 0)
```

배열의 로직

```
[False  True  True]
```

```
def step(x):
    if x >= 0:
        return 1
    else:
        return 0
```

버그

```
x = np.array([-1, 2, 3])
print((x > 0) * 1)
```

```
[0 1 1]
```

방법 1

```
x = np.array(x > 0, dtype=np.int)
print(x)
```

```
[0 1 1]
```

방법 2

활성화 함수 – 계단 함수 구현

```
def step(x):
    if x >= 0:
        return 1
    else:
        return 0
```

버그

방법 1

```
def step(x):
    return (x > 0) * 1
```

방법 2

```
def step(x):
    return np.array(x > 0, dtype=np.int)
```

활성화 함수 – 쌍곡탄젠트(tanh)

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$tanh(x) = 2sigmoid(2x) - 1$$

$$\begin{aligned} tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \frac{e^{-x}}{e^{-x}} \\ &= \frac{1 - e^{-2x}}{1 + e^{-2x}} \\ &= \frac{2 - (1 + e^{-2x})}{1 + e^{-2x}} \\ &= \frac{2}{1 + e^{-2x}} - 1 \qquad \because \sigma(2x) = \frac{1}{1 + e^{-2x}} \\ &= 2sigmoid(2x) - 1 \end{aligned}$$

- 시그모이드와 유사
  - 출력범위 **(-1, 1)**
  - 빠르게 수렴하는 특성

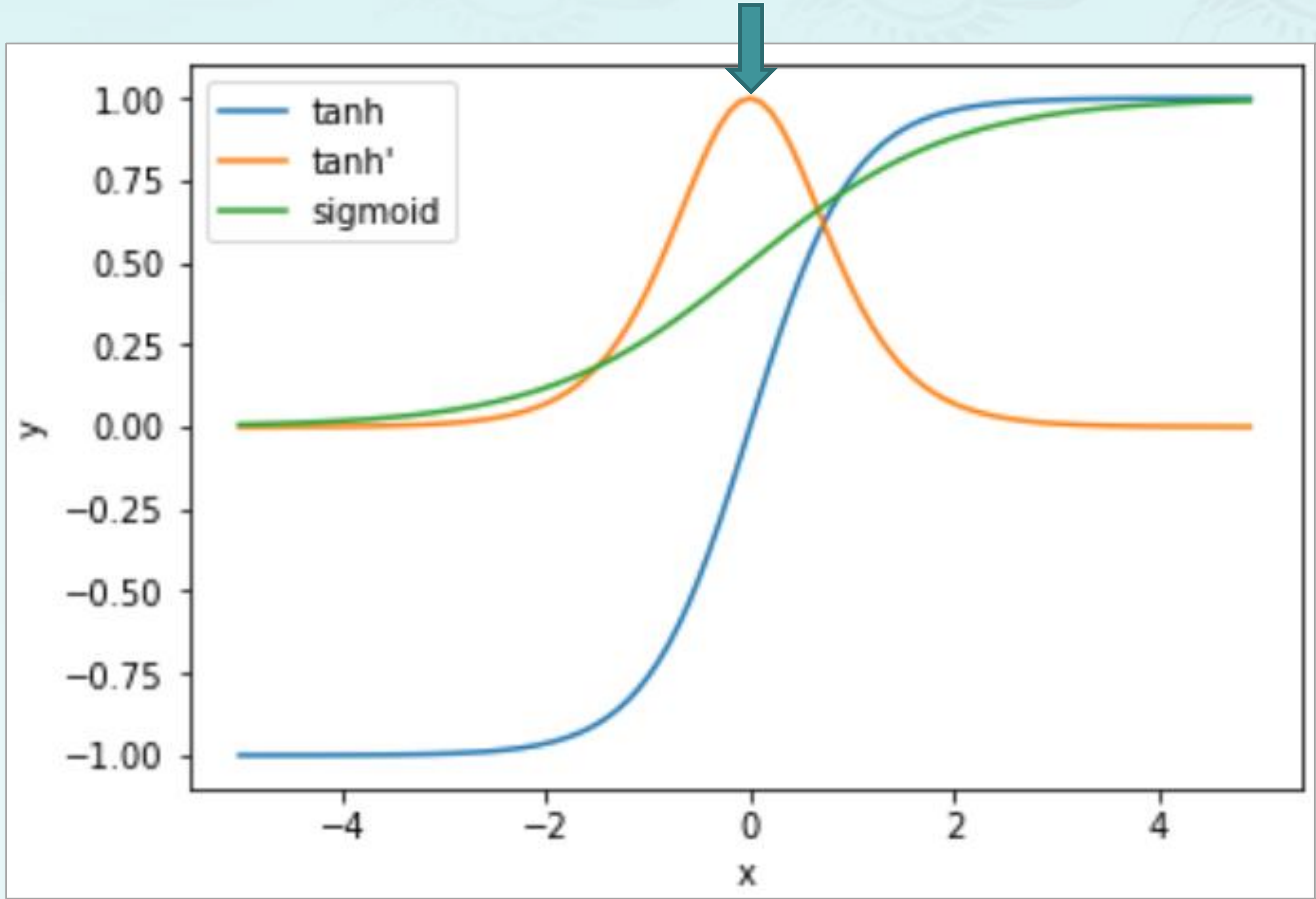
활성화 함수 – 쌍곡탄젠트(tanh)

- 미분

$$f(x) = e^x - e^{-x}$$
$$g(x) = e^x + e^{-x}$$
$$\frac{d}{dx}tanh(x) = [\frac{e^x - e^{-x}}{e^x + e^{-x}}]'$$
$$= [\frac{f(x)}{g(x)}]'$$
$$= \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$
$$= \frac{(e^x - (-e^{-x}))(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(1 + e^{-x})^2}$$
$$= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$
$$= 1 - [\frac{(e^x - e^{-x})}{(e^x + e^{-x})}]^2$$
$$= 1 - tanh^2(x)$$

$$(e^x)' = e^x$$
$$(e^{-x})' = -e^{-x}$$
$$[\frac{f(x)}{g(x)}]' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

활성화 함수 – 쌍곡탄젠트(tanh)




```
x = np.arange(-5.0, 5.0, 0.1)
y = tanh(x)
plt.plot(x, y, label='tanh')
y = (1-tanh(x))*(1+tanh(x))
plt.plot(x, y, label="tanh'")
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(-1.1, 1.1)
y = sigmoid(x)
plt.plot(x, y, label='sigmoid')
plt.legend(loc='best')
plt.show()
```

활성화 함수 – 렐루(ReLU)

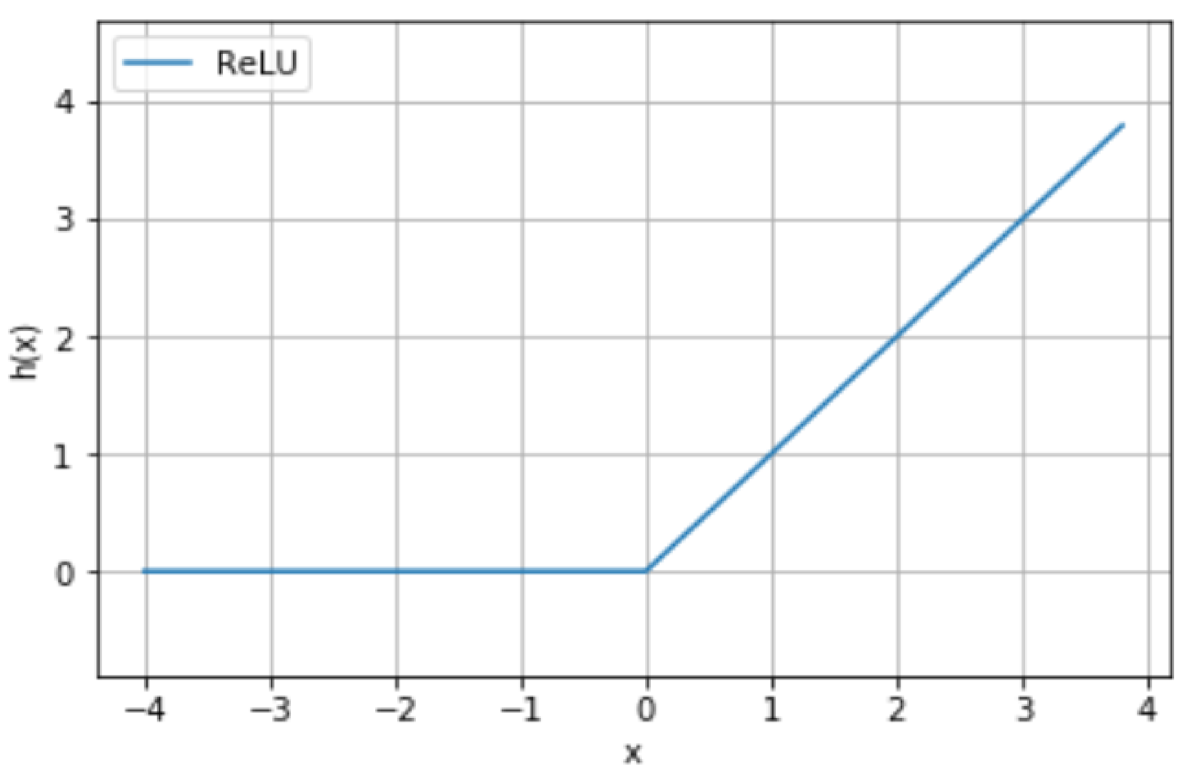
- 렐루: **Re**ctified **L**inear **U**nit

$$h(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- 특성
  - “소멸하는 기울기” 문제가 없음
  - 선형함수 – 간단한 미분

- 렐루 함수 구현

```python
def relu(x):
    return np.maximum(0, x)
```



활성화 함수

- 학습 정리
  - 활성화 함수의 역할과 기능
  - 활성화 함수의 미분
  - 다양한 활성화 함수
    - 시그모이드 함수
    - 계단 함수
    - 쌍곡탄젠트 함수
    - 렐루 함수