Week 3(3/3)

Activation Function

Machine Learning with Python

Handong Global University
Prof. Youngsup Kim
idebtor@gmail.com

Activation Function

- Goals
  - Understanding Activation Function
  - Learning Several Activation Functions

- Content
  - Concepts of Activation Function
  - Sigmoid function
  - Step functions
  - tanh function
  - ReLU function

C to F Converter

- **Formula(1)**

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

C to F Converter

- **Formula(1)**

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

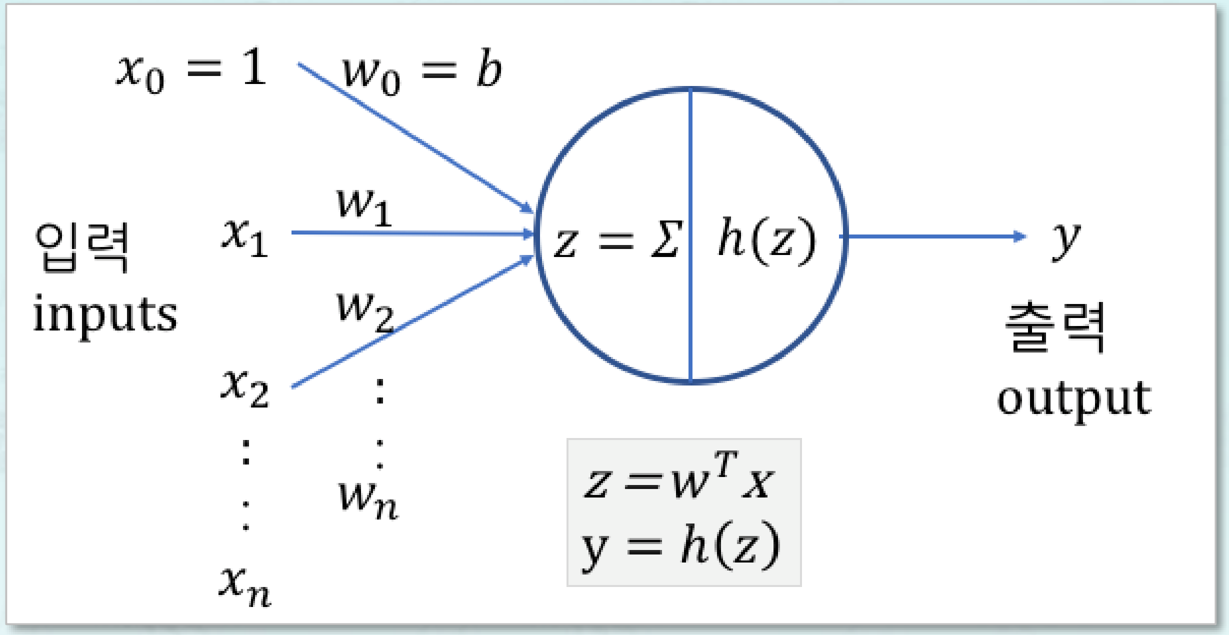
$$F = \begin{cases} \frac{9}{5}C + 32 & \textit{if } C \geq 0 \\ 32 & \textit{otherwise} \end{cases}$$

C to F Converter

- **Formula(1)**

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

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

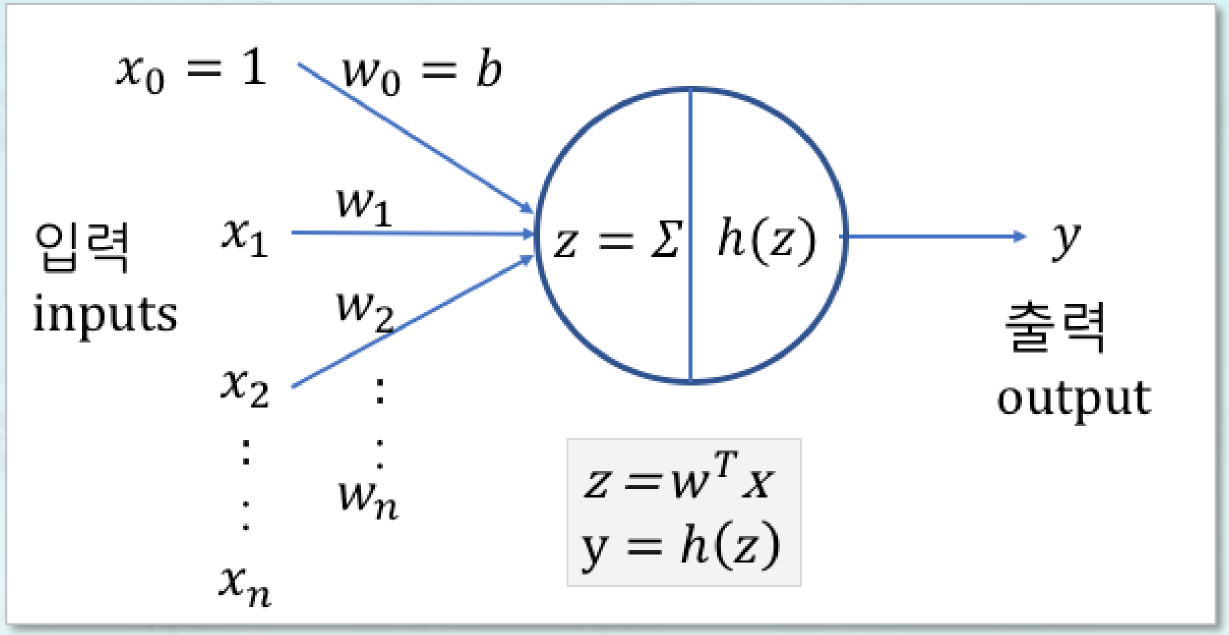


C to F Converter

- **Formula(1)**

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

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



- **Formula for Neuron(2)**

$$z = w_0 x_0 + w_1 x_1$$

단, $x_0 = 1, w_0 = b$

$$y = h(z)$$

C to F Converter

- **Formula(1)**

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

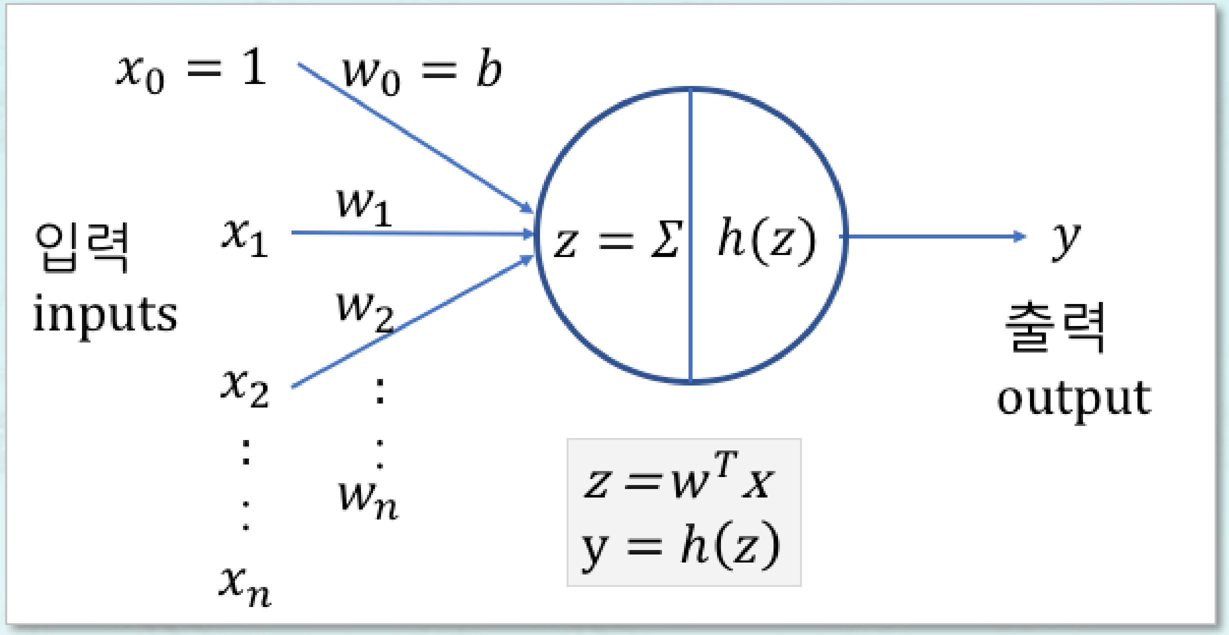
$$F = \begin{cases} \frac{9}{5}C + 32 & \text{if } C \geq 0 \\ 32 & \text{otherwise} \end{cases}$$

- **Formula for Neuron(2)**

$$z = w_0x_0 + w_1x_1$$

$$\text{단}, x_0 = 1, w_0 = b$$

$$y = h(z)$$



C to F Converter

- **Formula(1)**

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

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

- **Formula for Neuron(2)**

$$z = w_0 x_0 + w_1 x_1$$

$$\text{단}, x_0 = 1, w_0 = b$$

$$y = h(z)$$

- **C to F Formula(3)**

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

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

C2F Neuron

- C2F Neuron
- C to F Formula(3)

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

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

Activation Function

C2F Neuron

- **C2F Neuron**

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

- **C to F Formula(3)**

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

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

Activation Function

C2F Neuron

- **C2F Neuron**

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

- **C to F Formula(3)**

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

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

Activation Function

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

C2F Neuron

- C2F Neuron

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

- C to F Formula(3)

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

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

Activation Function

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

C2F Neuron

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

C2F Neuron

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

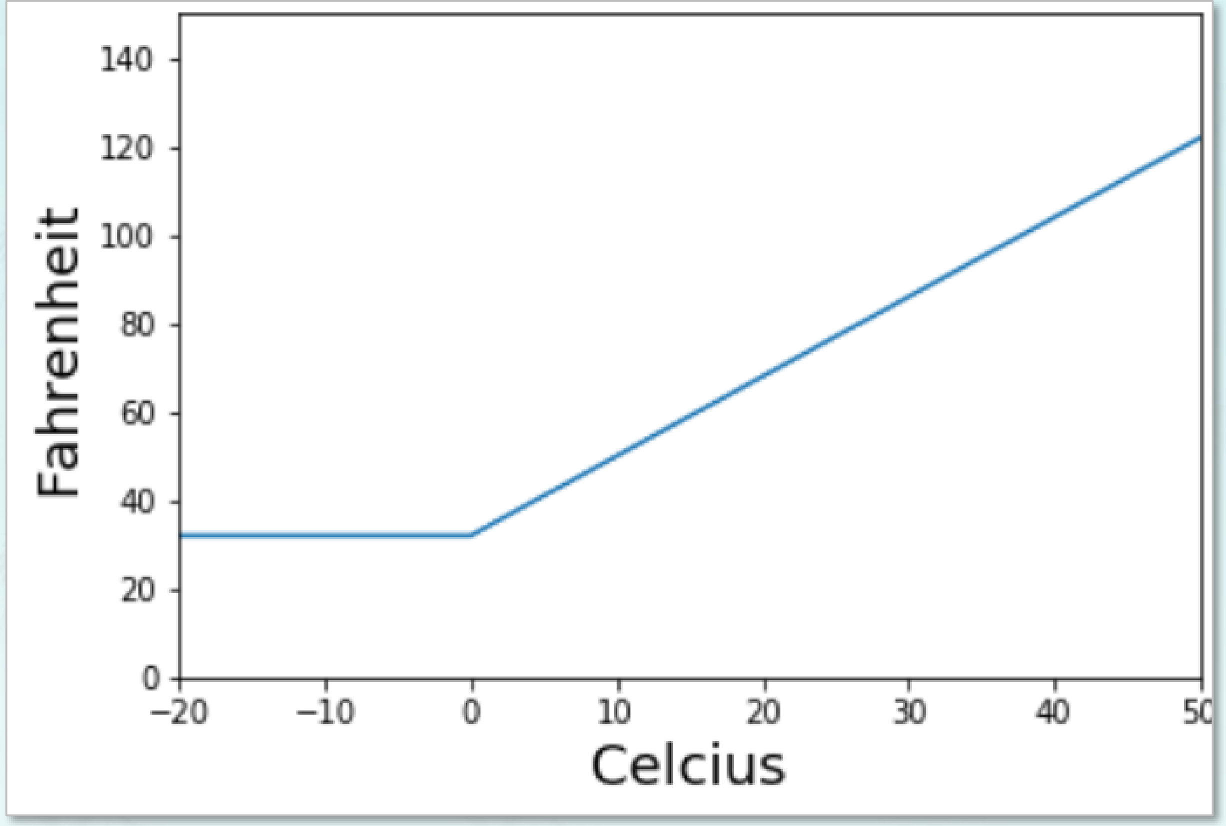
```
[32, 32, 32.0, 97.7, 104.0, 122.0, 212.0]
```

C2F Neuron Test

```
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]
```

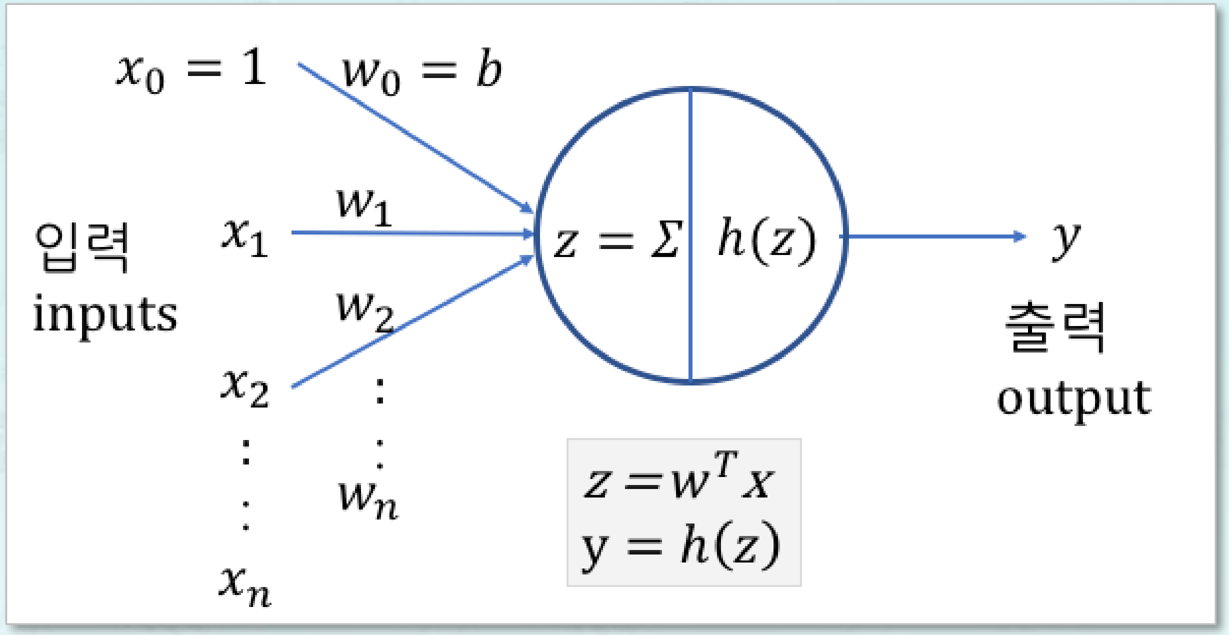
C2F Neuron Test

```python
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()
```



Activation Function

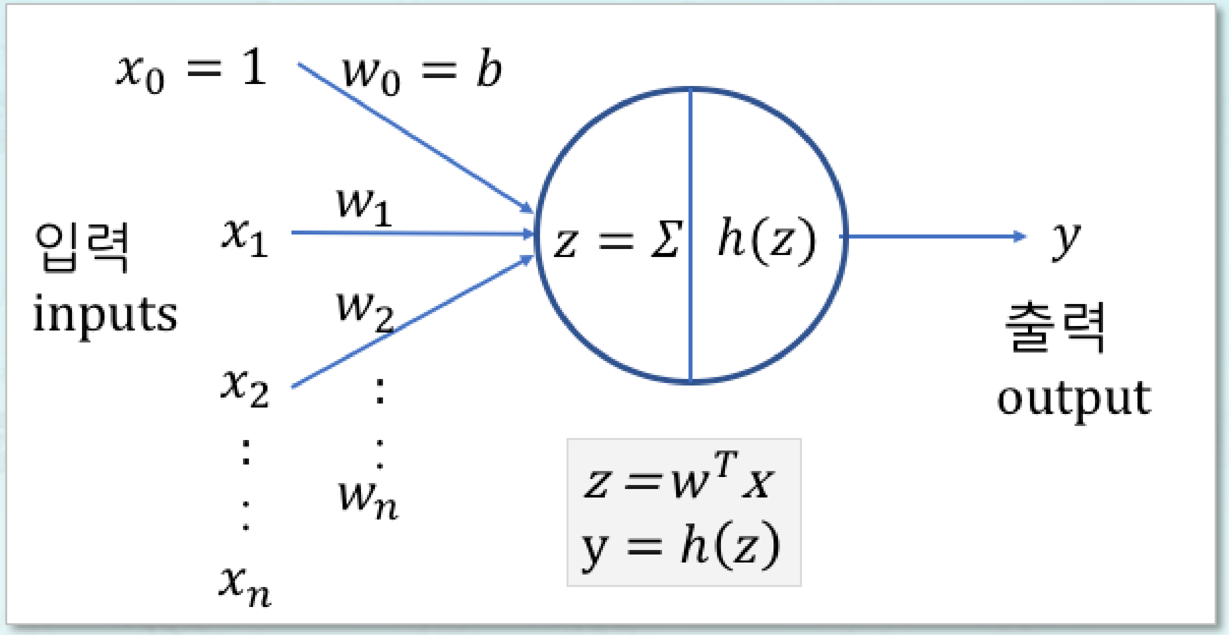
- **JoyQuiz:**
- **Why did we make C2F Neuron?**



Activation Function

- **JoyQuiz:**
- **Why did we make C2F Neuron?**

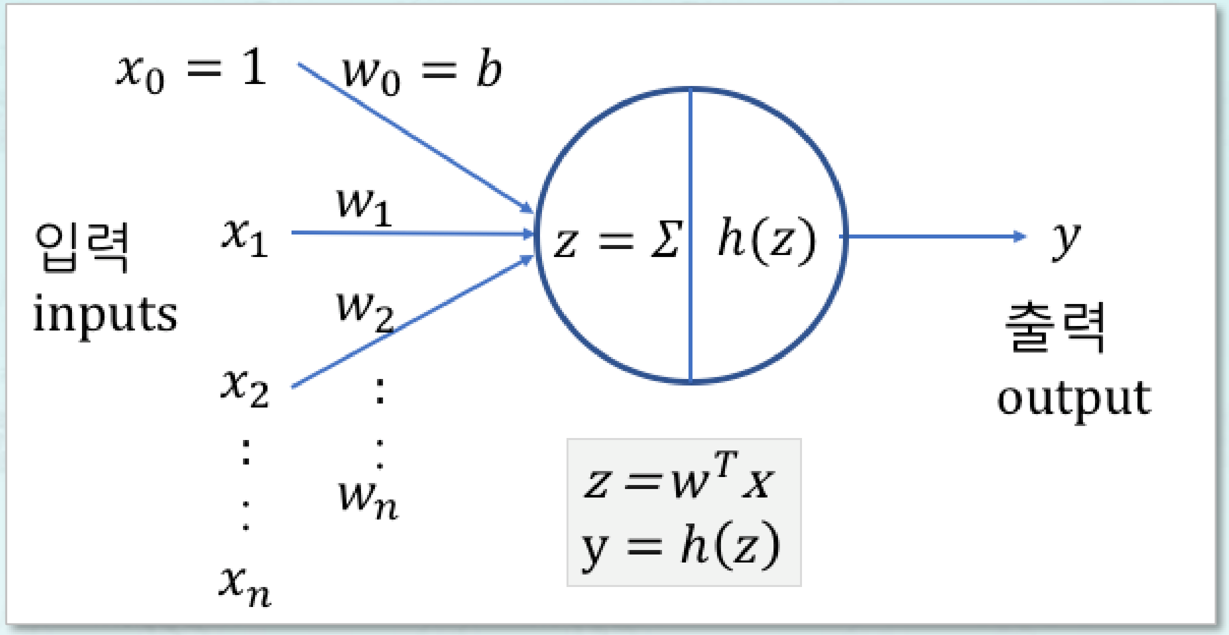
  (1) input x
  (2) weight w
  (3) bias b
  (4) net input z
  (5) Activation Function h(z)
  (6) output y



Activation Function

- **JoyQuiz:**
- **Why did we make C2F Neuron?**

  (1) input x
  (2) weight w
  (3) bias b
  (4) net input z
  (5) Activation Function h(z)
  (6) output y



Activation Function – Sigmoid

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

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

- e : 2.7182…

Activation Function – Sigmoid

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

- e : **2.7182…**

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

- $\sigma(0) = \frac{1}{1+\frac{1}{e^{0}}} = ?$

Activation Function – Sigmoid

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

- e : **2.7182...**

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

Activation Function – Sigmoid

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

- e : **2.7182...**

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

Activation Function – Sigmoid

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

- e : **2.7182...**

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

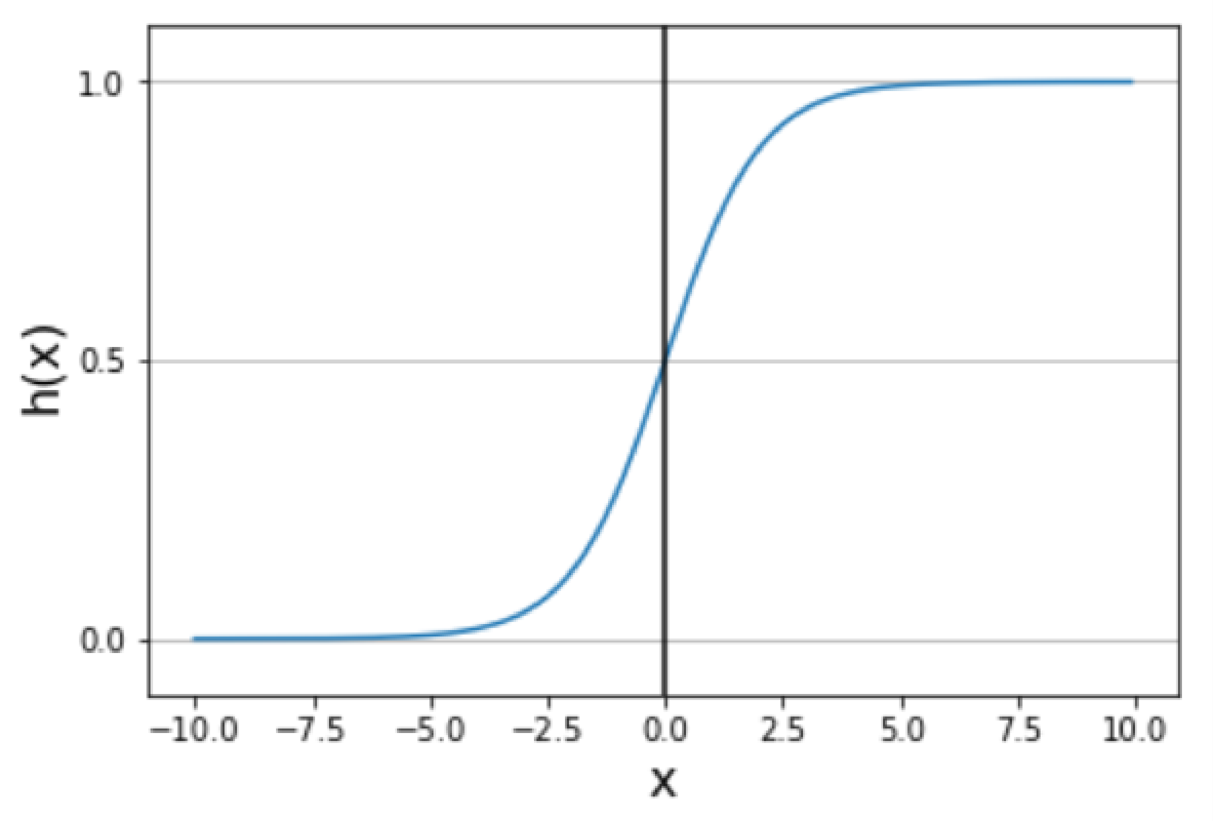
Activation Function – Sigmoid

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

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

Activation Function – Sigmoid

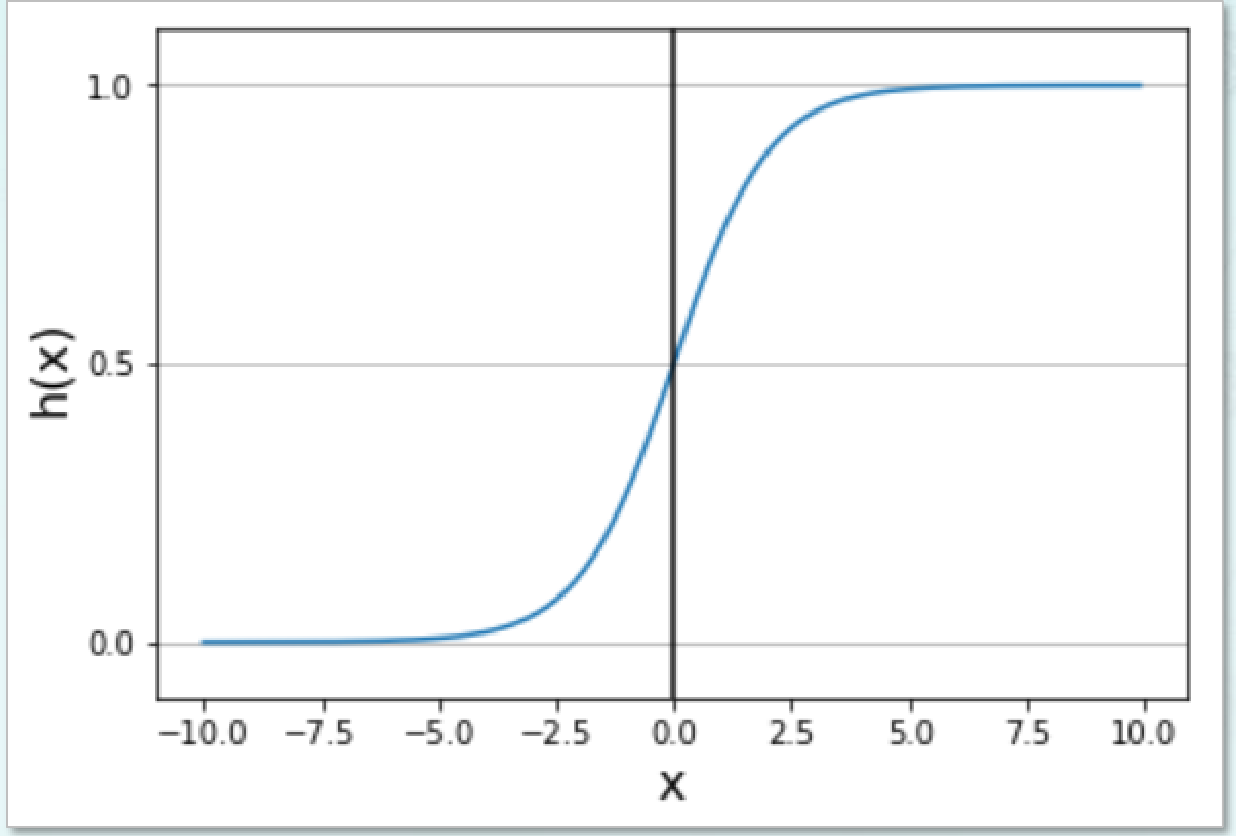
$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$




```
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()
```

Activation Function – Sigmoid

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



- Convert input to something that is differentiable between 0 and 1.
- Use logistic classification and cost function.
- The output is between 0 and 1.

Activation Function – Sigmoid

- **Derivative Rules**

$$(e^x)' = e^x \quad ①$$

$$(e^{-x})' = -e^{-x} \quad ②$$

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

Activation Function – Sigmoid

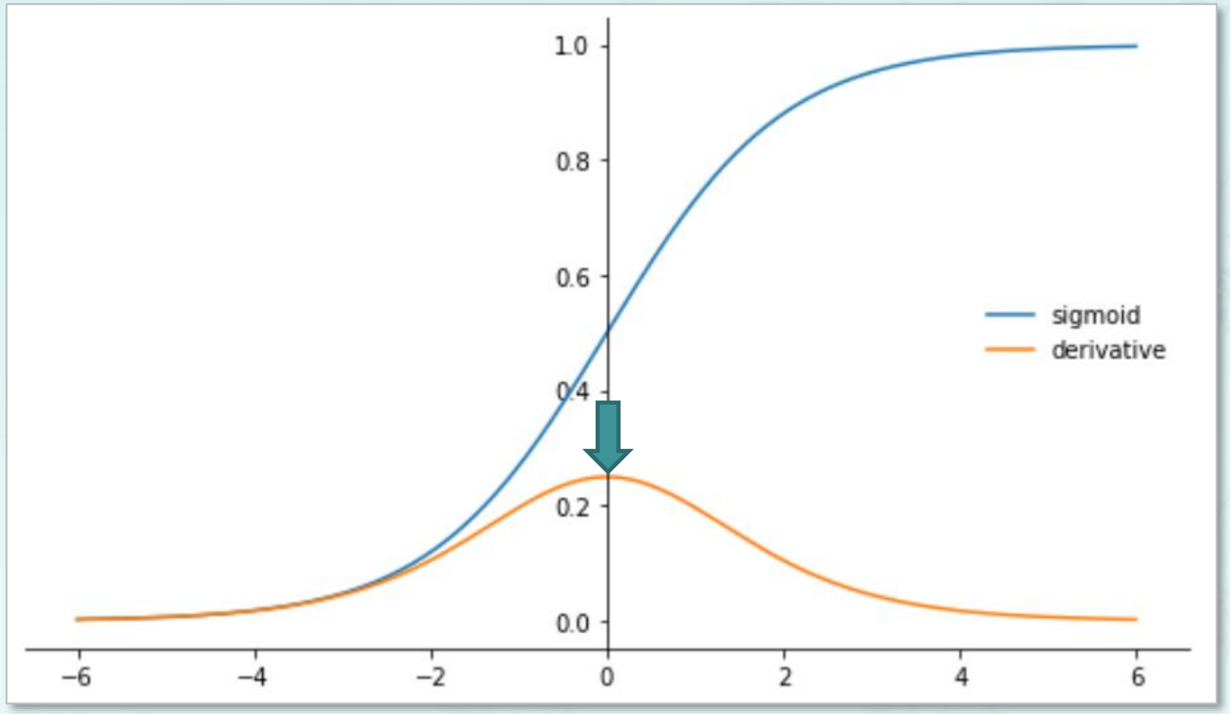
- **Derivative Rules**

$$(e^x)' = e^x \quad ①$$
$$(e^{-x})' = -e^{-x} \quad ②$$
$$\frac{du^n}{dx} = nu^{n-1}\frac{du}{dx} \quad ③$$

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

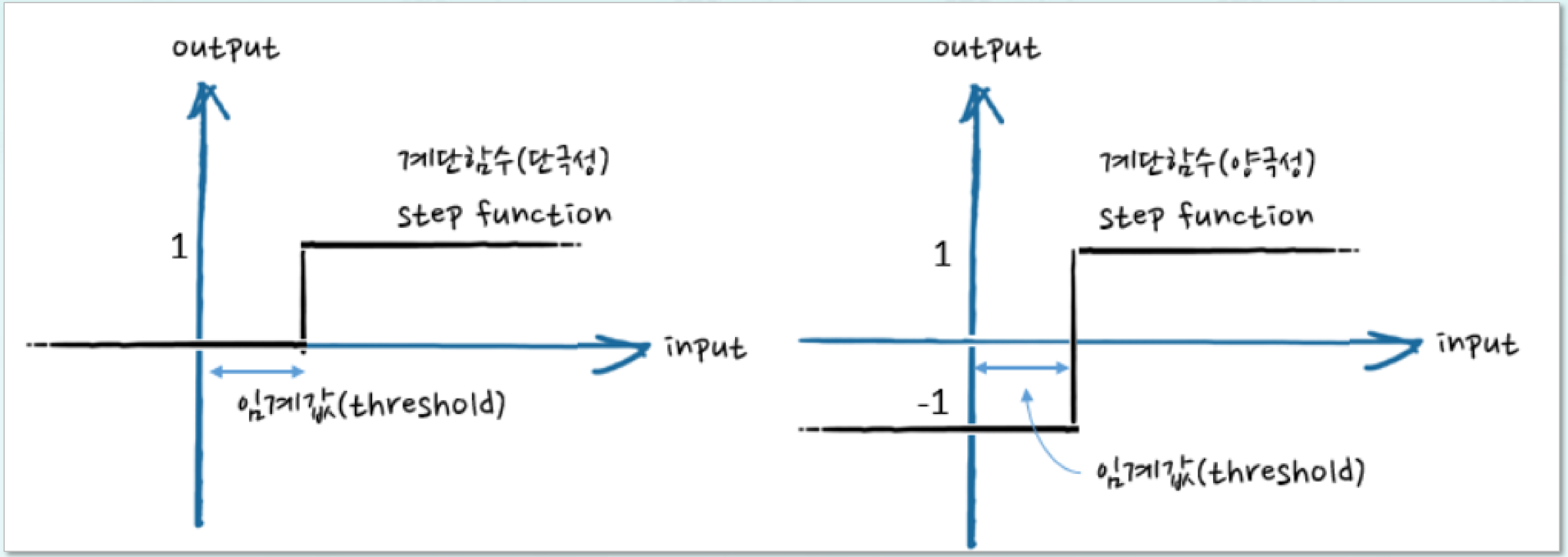
Activation Function – Sigmoid

- **Sigmoid and its Derivative**



- **Max Derivative: 0.25**

Activation Function – Step



Activation Function – Step

- **Unipolar and Bipolar Step Function**

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

$$y = h(z)$$

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

단극성

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

양극성

Activation Function – Step

- **Unipolar and Bipolar Step Function**

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \ldots$$
$$y = h(z)$$

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

단극성

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

양극성

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

Activation Function – Step

- **Unipolar and Bipolar Step Function**

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \ldots$$
$$y = h(z)$$

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

단극성

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

양극성

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

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

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

Activation Function – Step

- **Unipolar and Bipolar Step Function**

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

$$y = h(z)$$

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

단극성

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

양극성

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

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

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

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

Activation Function – Step

- **Unipolar and Bipolar Step Function**

```
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()
```

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

버그

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

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

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

Activation Function – Step

- **Using Boolean indexing**

```
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
```

Bug

방법 1

방법 2

Activation Function – 계단 함수 구현

- **Using Boolean indexing**

```
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

Activation Function – 계단 함수 구현

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

버그

방법 1

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

방법 2

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

Activation Function – tanh

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

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

- **Similar to sigmoid**
    -

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

Activation Function – tanh

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

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

- **Similar to sigmoid**
  - **Output range (-1, 1)**
  - **Converge fast**

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

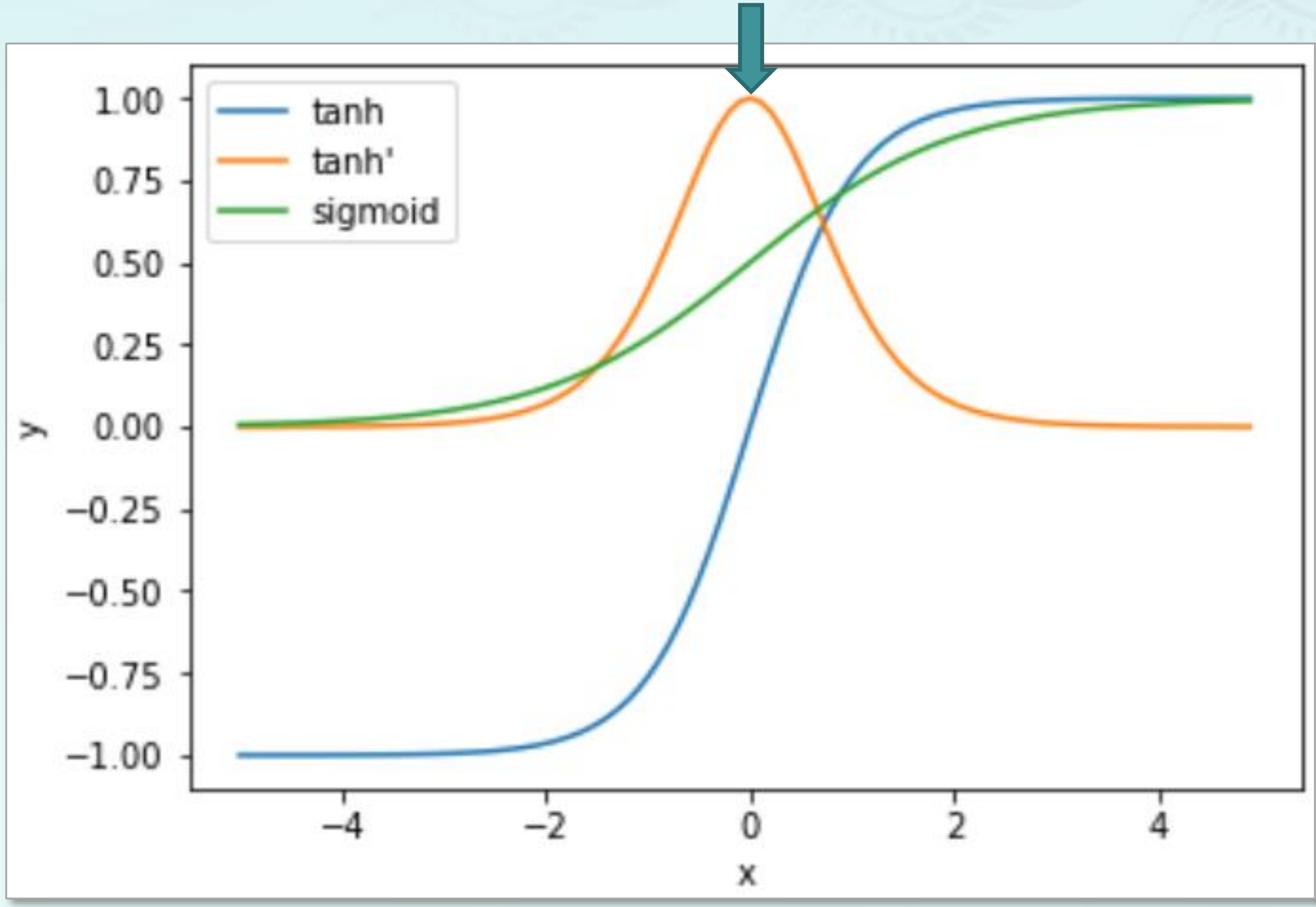
Activation Function – tanh

- **Derivative**

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

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

Activation Function – tanh



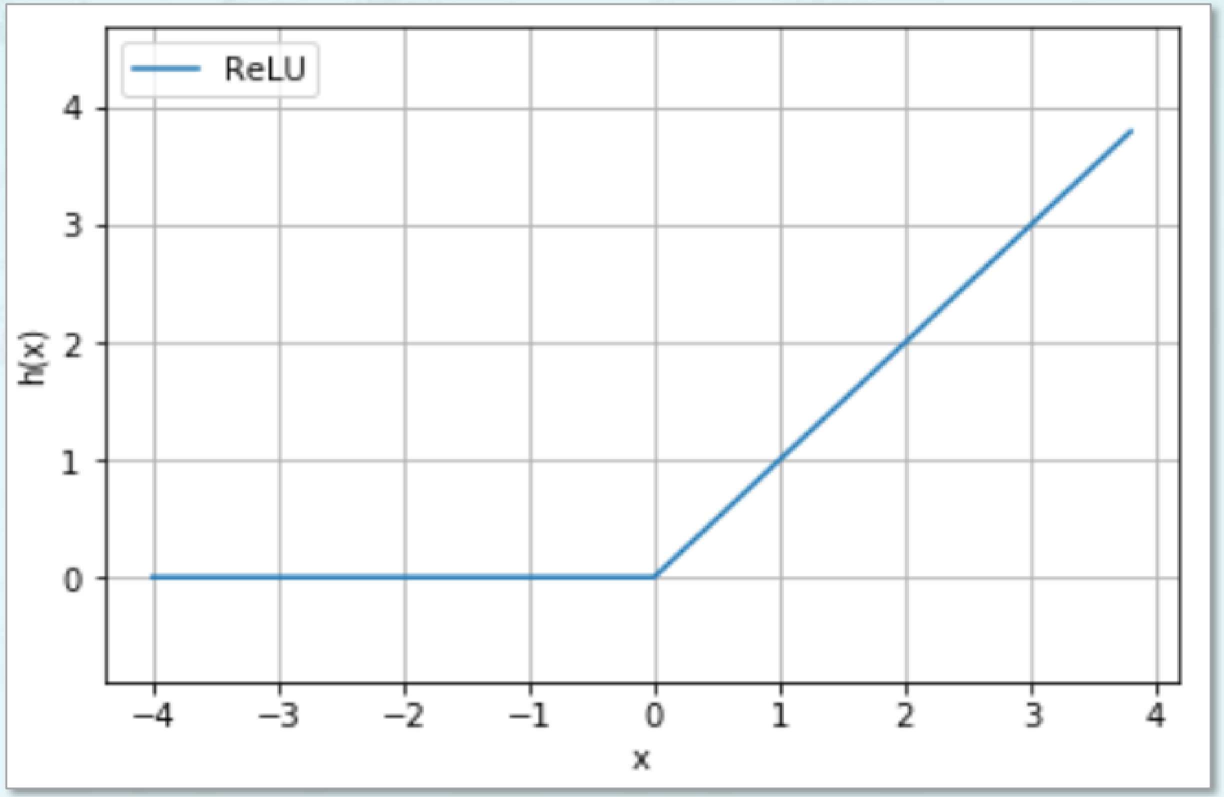

```python
x = np.arange(-5.0, 5.0, 0.1)
y = tanh(x)
plt.plot(x, y, label='tanh')
y = (1-tanh(x))*(1+tanh(x))
plt.plot(x, y, label="tanh'")
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(-1.1, 1.1)
y = sigmoid(x)
plt.plot(x, y, label='sigmoid')
plt.legend(loc='best')
plt.show()
```

Activation Function – ReLU

- Rectified Linear Unit

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

- Implementation



Activation Function – ReLU

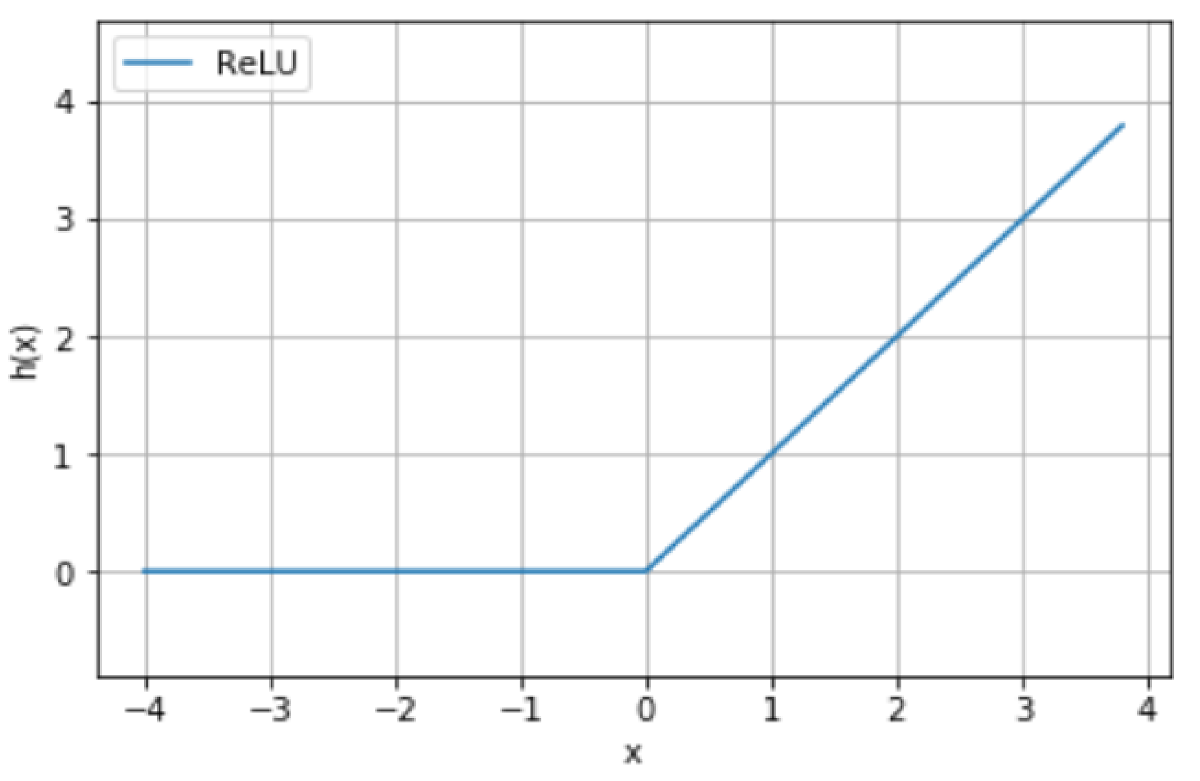
- Rectified Linear Unit

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

- No "vanishing gradient problem"
- Linear function
- Simple derivative

- Implementation

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



Activation Function

- Summary
  - Understanding Activation Function
  - Activation Functions
    - Step functions
    - tanh
    - sigmoid
    - ReLU

- Next
  - 4-1 Perceptron

Week3(3/3)

Activation Function

Machine Learning with Python

Handong Global University
Prof. Youngsup Kim
idebtor@gmail.com