

신경망 구조

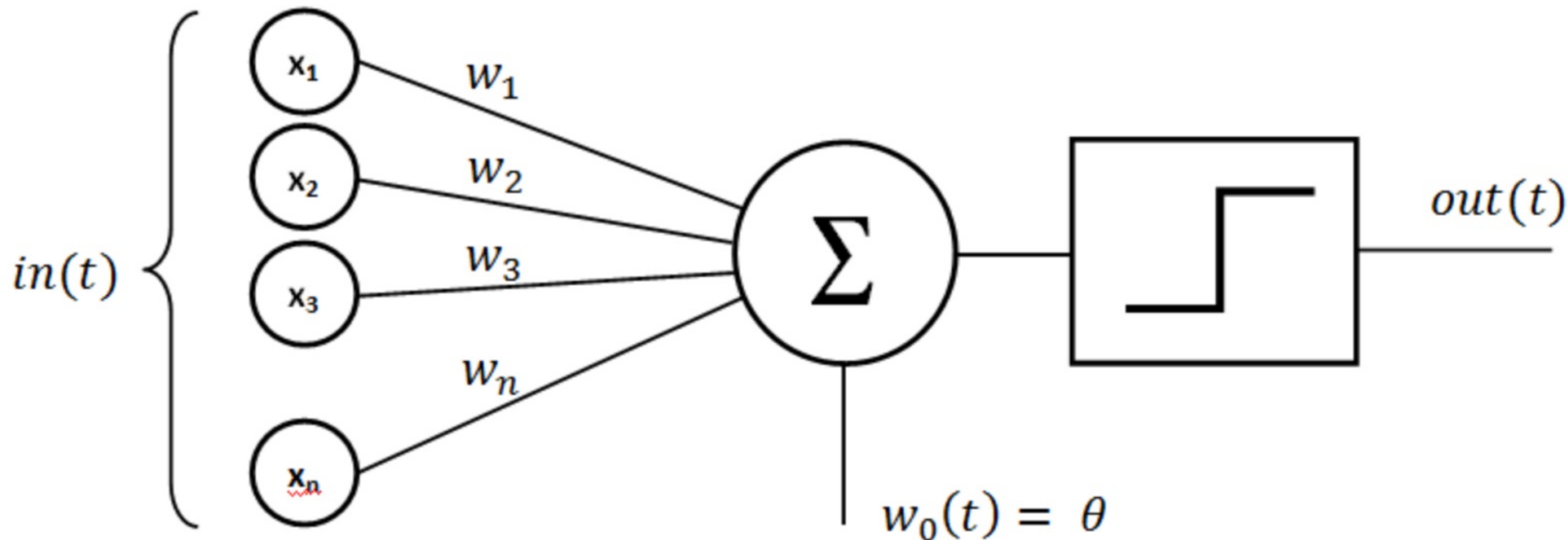
김영욱

Hello AI

```
3 require File.expand_path("../config/initializers/spec_helper.rb", __FILE__)
4 # Prevent database truncation if the environment is production
5 abort("The Rails environment is running in production mode!")
6 require 'spec_helper'
7 require 'rspec/rails'
8
9 require 'capybara/rspec'
10 require 'capybara/rails'
11
12 Capybara.javascript_driver = :webkit
13 Category.delete_all; Category.create
14 Shoulda::Matchers.configure do |config|
15   config.inherit_from(Shoulda::Matchers::Rails)
16   with.test_framework :rspec
17   with.library :rails
18 end
19 end
20
21 # Add additional requires below this line. See the documentation for more.
22 # Requires supporting ruby files from spec/support/ and its subdirectories.
23 # spec/support/ and its subdirectories. These files are required by default.
24 # run as spec files by default. This will be required when running rspec.
25 # in _spec.rb will both be required when running rspec.
26 # run twice. It is recommended that you do not use this option.
27 # end with _spec.rb. You can configure the behavior of this option
28 # on the command line using the --rspec option.
29
30 No results found for 'mongoid'
```

퍼셉트론

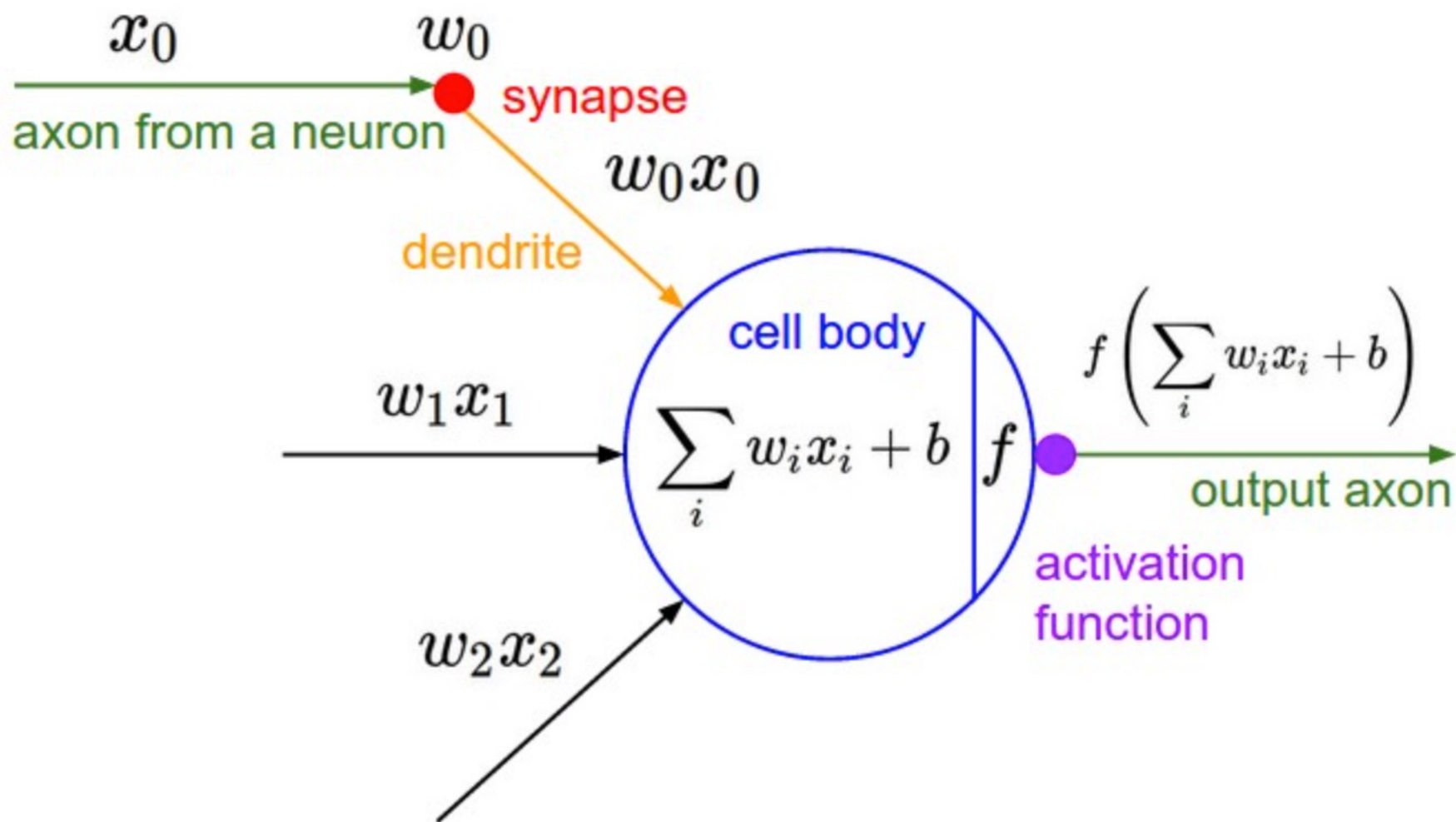
- 인공신경망의 한 종류
- 다수의 입력(x_1, x_2, \dots, x_n)과 가중치(w_1, w_2, \dots, w_n)를 곱하여 그 값에 편향(*bias*)을 더한 값이 어느 임계치 값(θ)을 초과하면 활성화 함수를 통과



한 출력값을 내보냄

출처: <https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a>

뉴런의 수학적 표현

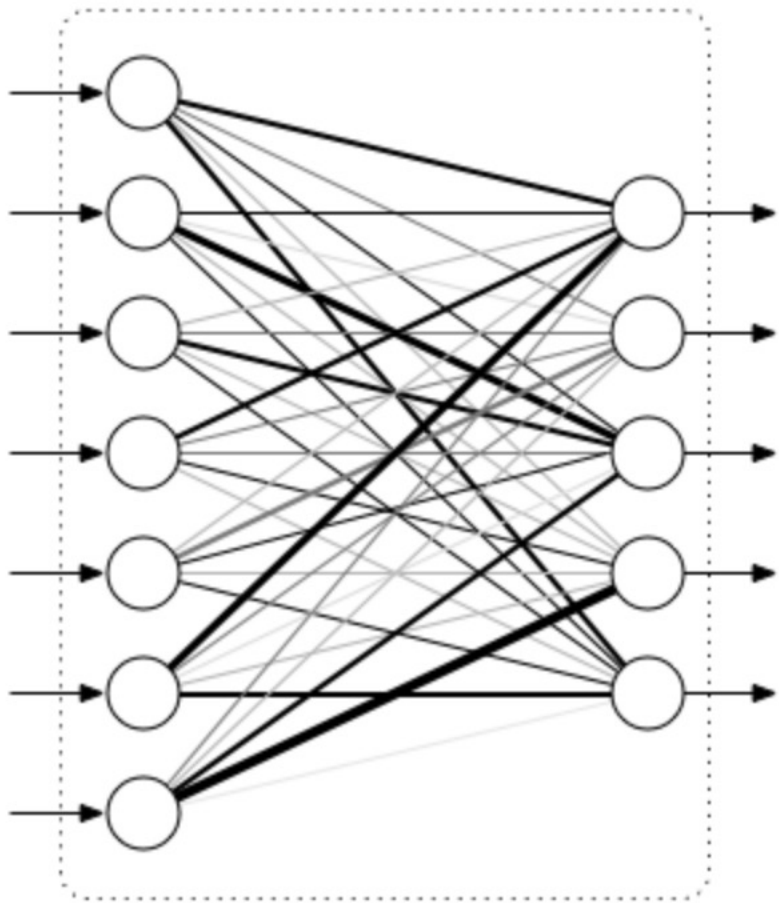


$$y = f(\sum_i w_i x_i + b)$$

- f : 활성화 함수
 - 임계값(θ)을 경계로 출력이 바뀜
- b : 편향
 - 결정 경계선을 원점에서부터 벗어나게 해줌
 - 따로 표현이 없어도 기본적으로 존재한다고 생각
- $\sum_i w_i x_i$: 두 벡터의 내적으로 표현 가능

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n = w^T x$$

완전 연결 계층(Fully-Connected Layer) 수학적 표현



$$W = [w_0, w_1, \dots, w_{M-1}]^T$$

각각의 w_k 는 $N \times 1$ 형태의 벡터

W 는 $N \times M$ 행렬

$$b = [b_0, b_1, \dots, b_{M-1}]$$

$$y_0 = f(w_0^T x + b_0)$$

$$y_1 = f(w_1^T x + b_1)$$

$$y_2 = f(w_2^T x + b_2)$$

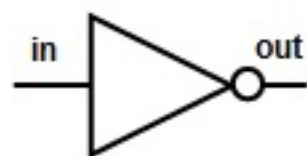
...

$$y_{M-1} = f(w_{M-1}^T x + b_{M-1})$$

$$\rightarrow y = f(Wx + b)$$

논리회로

- 논리 게이트(Logic Gates)
 - AND
 - OR
 - NOT
 - NAND
 - NOR



NOT

Input	Output
I	F
0	1
1	0



AND

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1



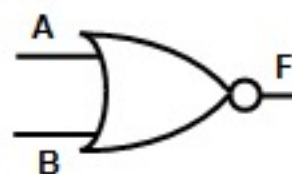
NAND

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0



OR

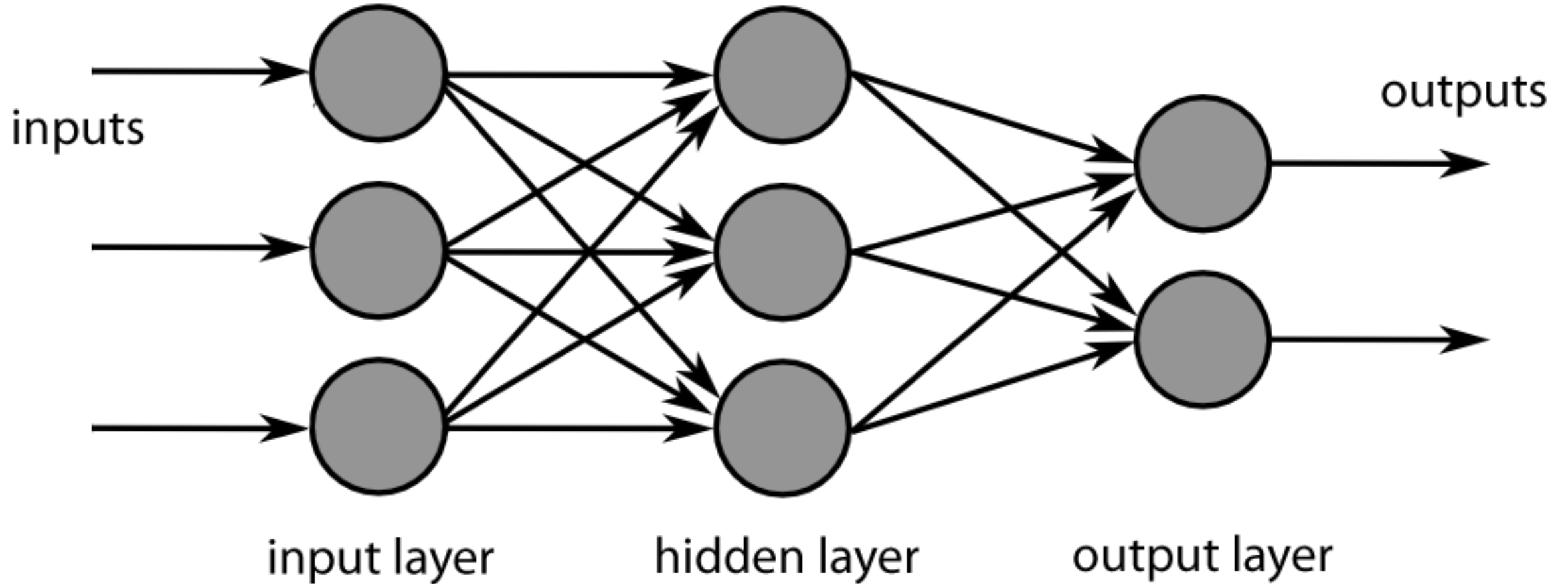
Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1



NOR

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0

다층 퍼셉트론(Multi Layer Perceptron, MLP)



다층 퍼셉트론의 구성

- 입력층(input layer)
- 은닉층(hidden layer)
 - 1개 이상 존재
 - 보통 5개 이상 존재하면 Deep Neural Network라고 칭함
- 출력층(output layer)

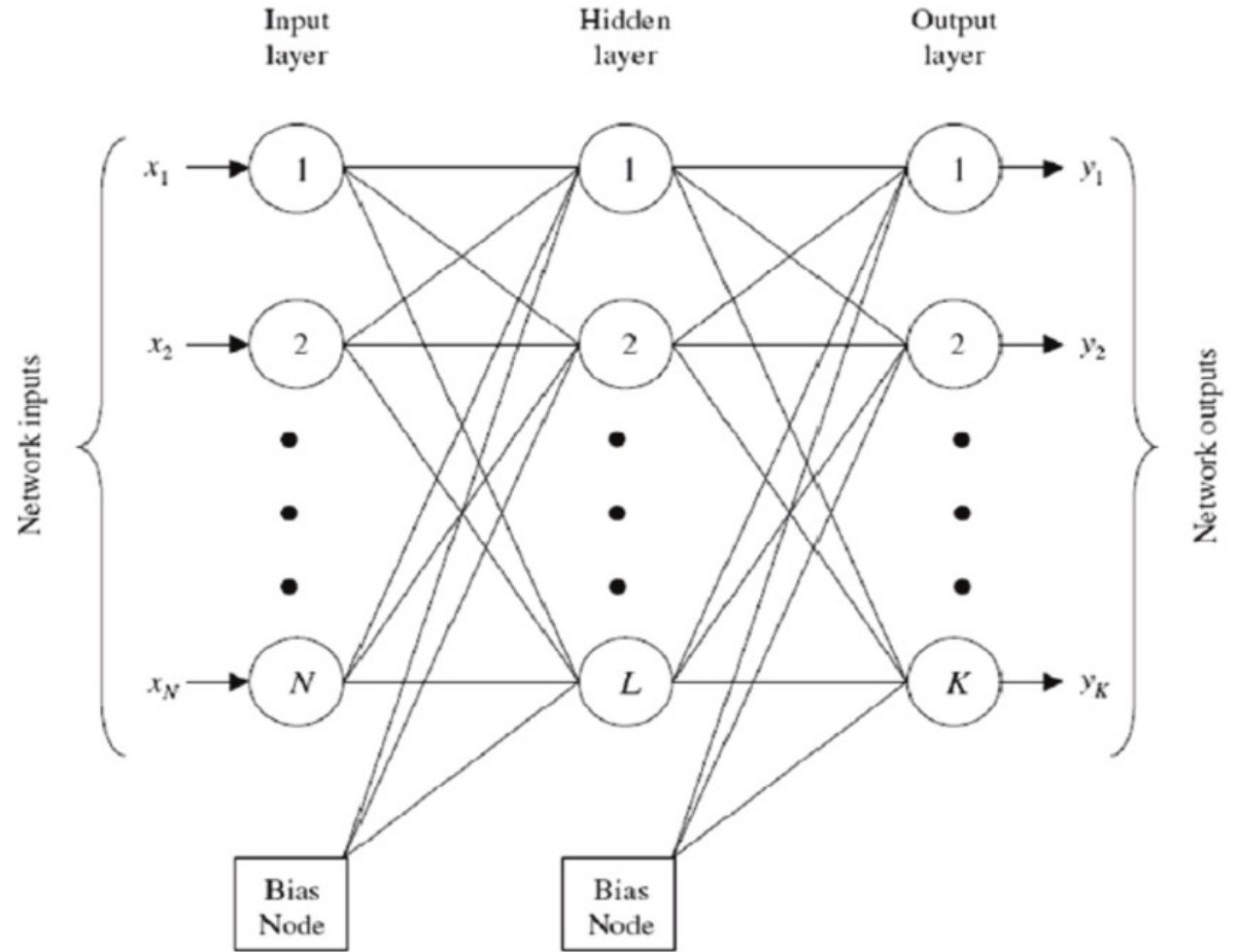
- 수식

- (input layer → hidden layer)

$$z = f_L(W_L x + b_L)$$

- (hidden layer → output layer)

$$y = a_K(W_K z + b_K)$$

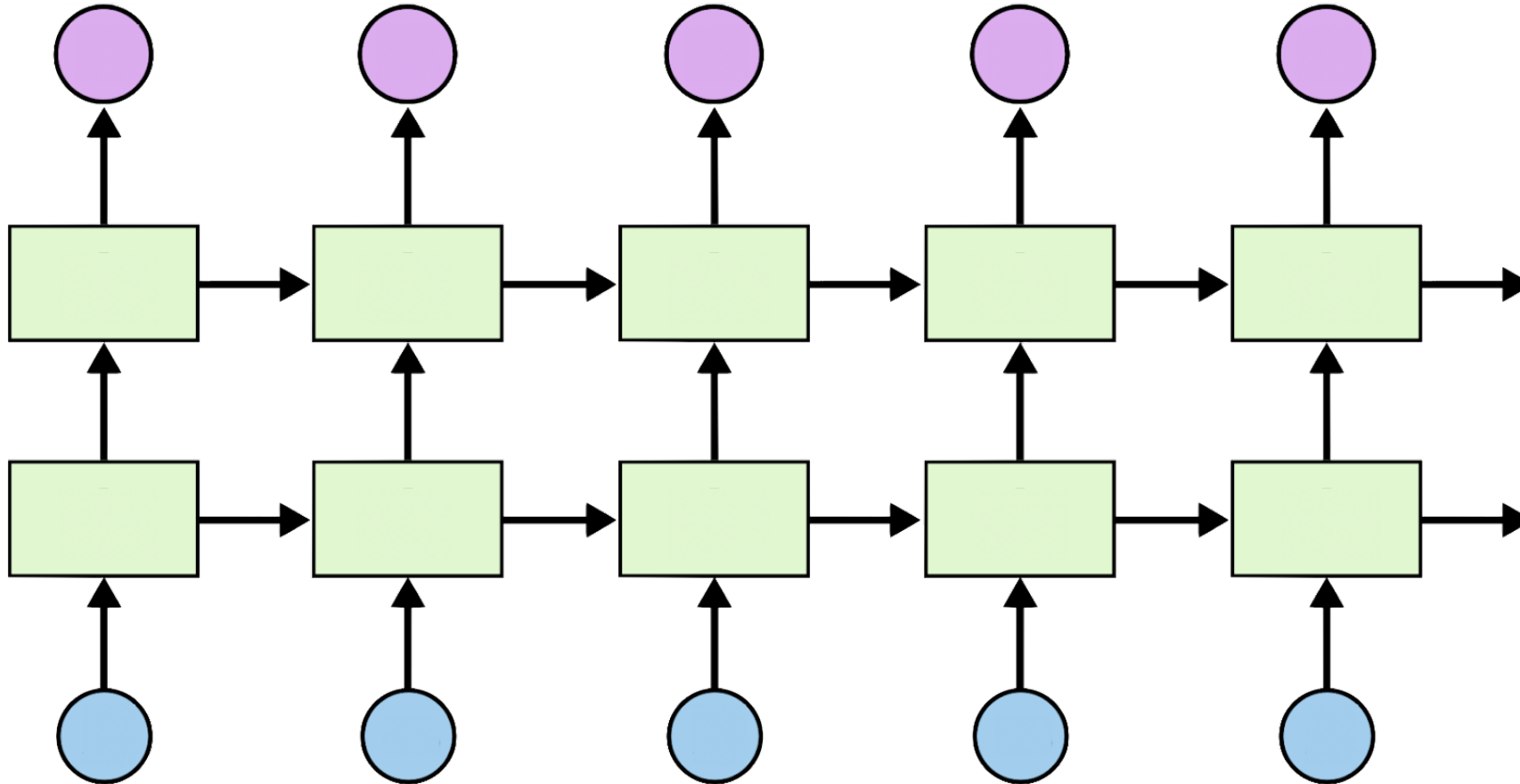


XOR 게이트

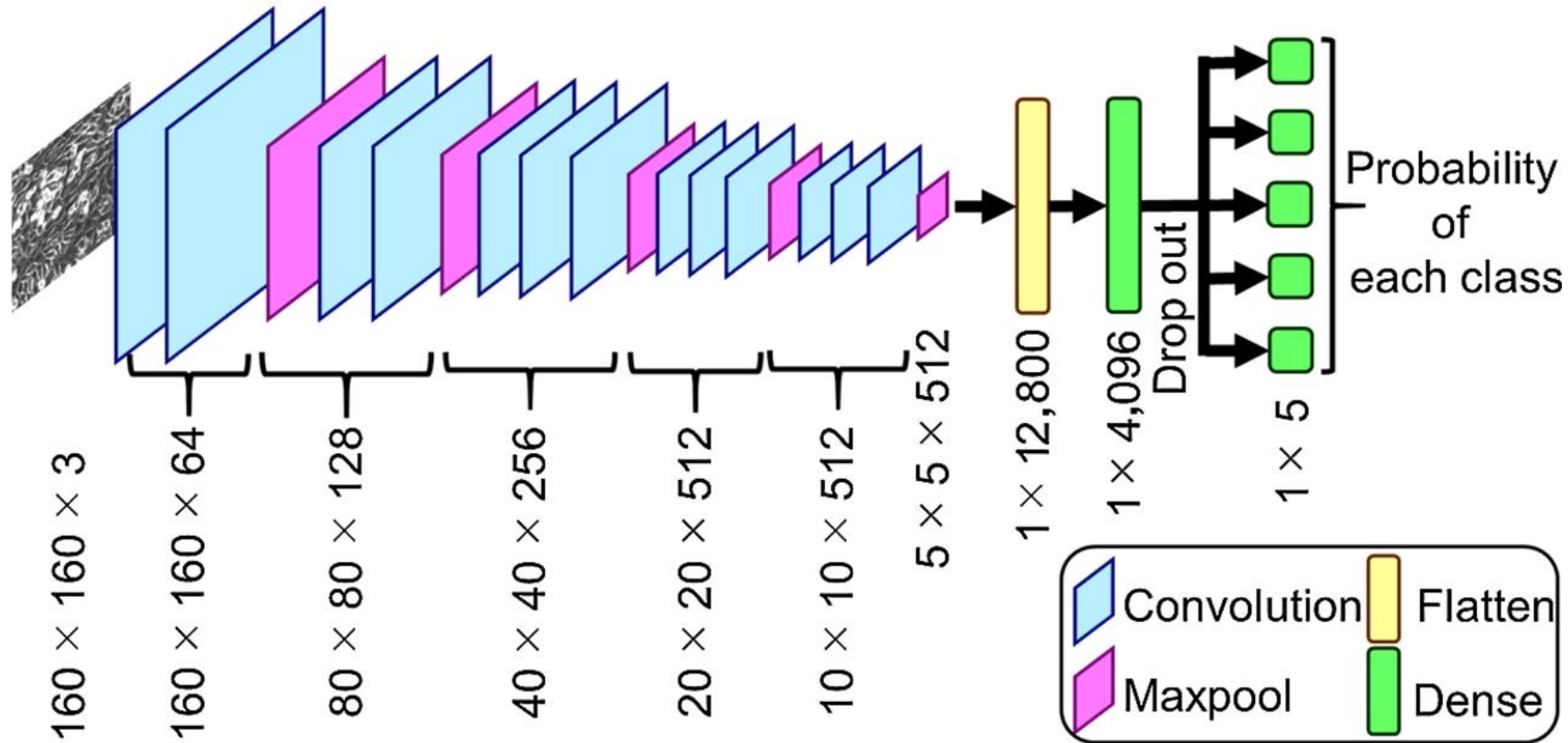
- 서로 다른 두 값이 입력으로 들어가면 1을 반환
- 진리표

Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

순환 신경망(Recurrent Neural Network, RNN)



합성곱 신경망(Convolutional Neural Network, CNN)

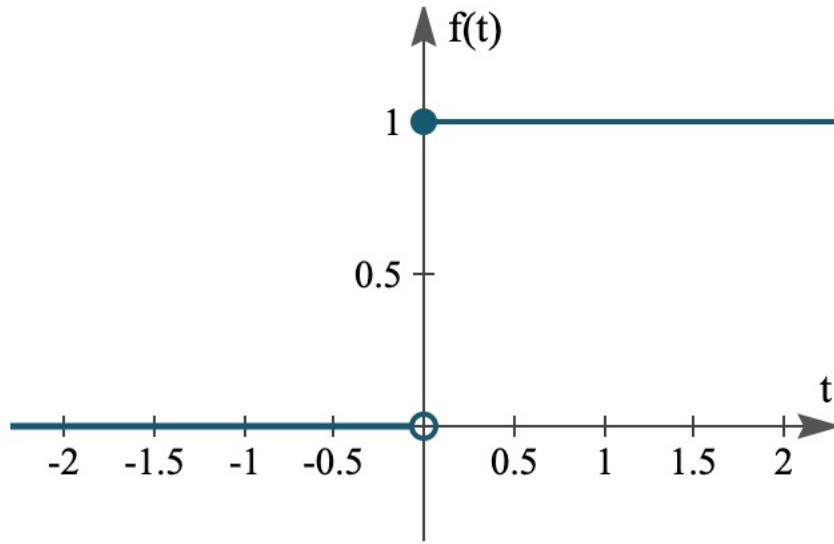


활성화 함수(Activation Function)

- 입력 신호의 총합을 출력 신호로 변환하는 함수
- 활성화 함수에 따라 출력값이 결정
- 단층, 다층 퍼셉트론 모두 사용
- 대표적인 활성화 함수
 - Sigmoid
 - ReLU
 - tanh
 - Identity Function
 - Softmax
- 하나의 layer에서 다음 layer로 넘어갈 때는 항상 활성화 함수를 통과
- [참고] 여러가지 활성화 함수

Step Function(계단 함수)

$$y = \begin{cases} 0 & (x < 0) \\ 1 & (x \geq 0) \end{cases}$$



출처: <https://www.intmath.com/laplace-transformation/1a-unit-step-functions-definition.php>

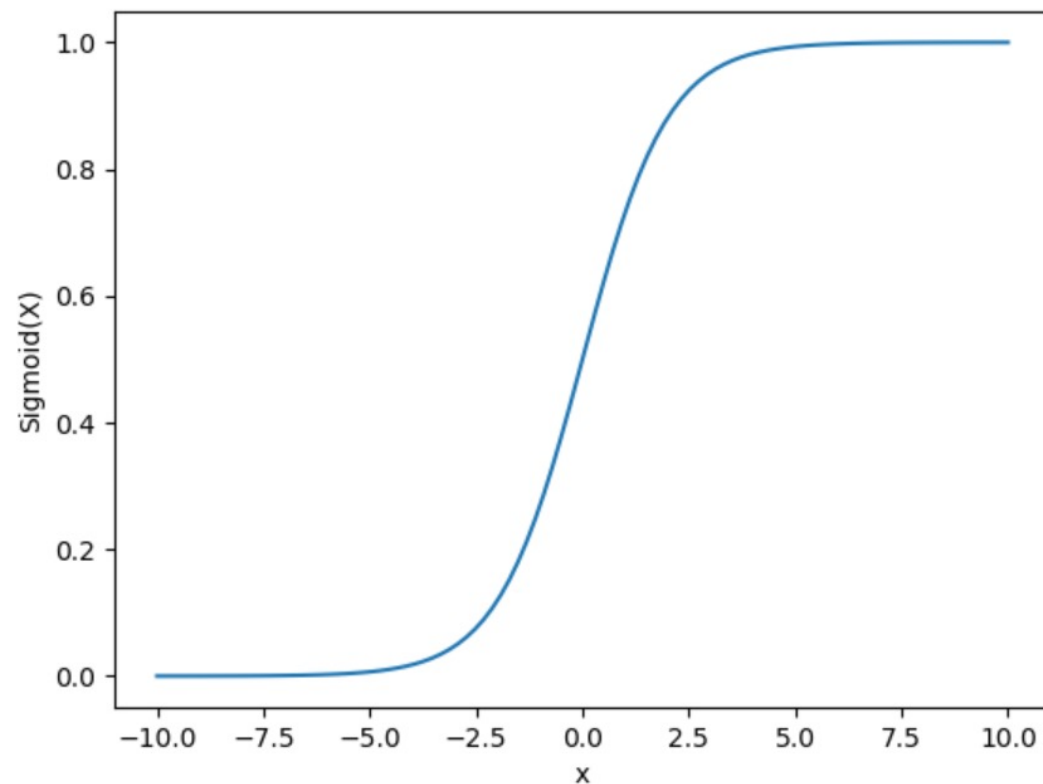
Figure 1



Sigmoid Function(시그모이드 함수)

- 이진분류(binary classification)에 주로 사용
 - 마지막 출력층의 활성화 함수로 사용
- 출력값이 0~1 의 값이며, 이는 확률로 표현 가능

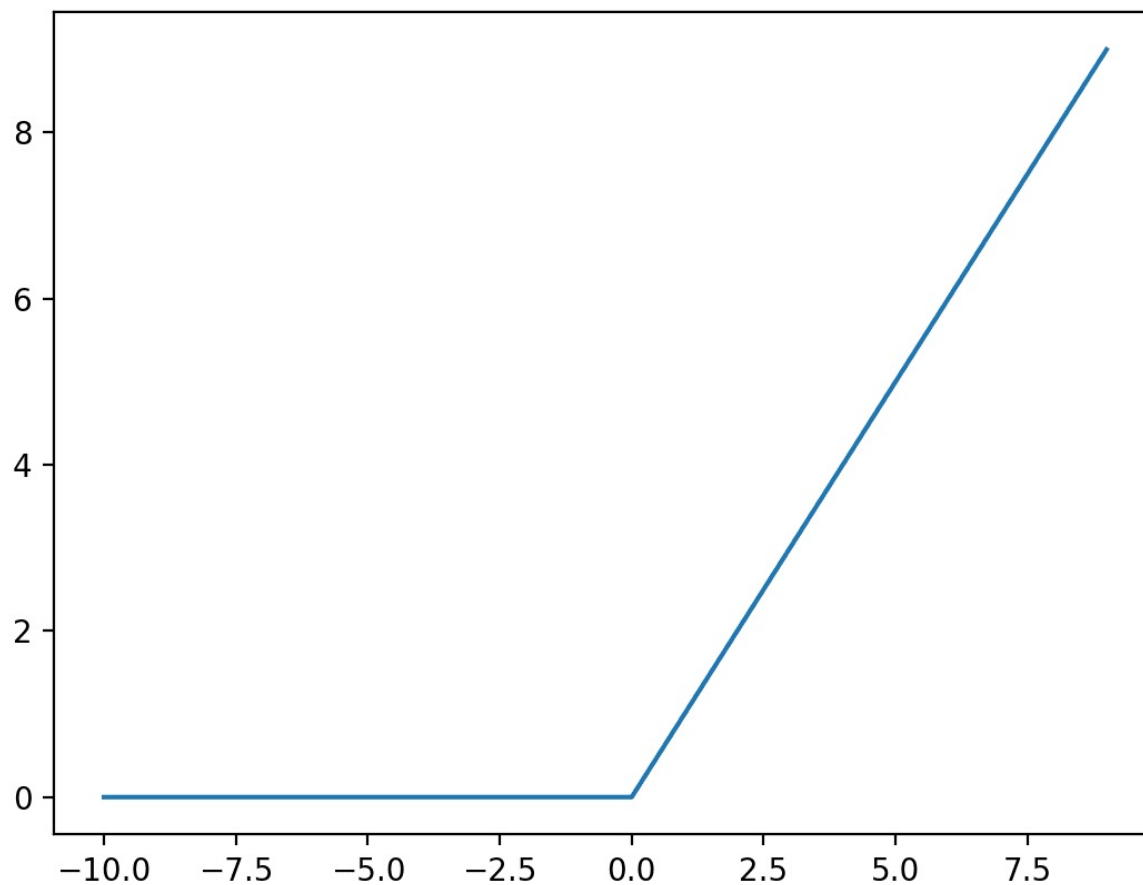
$$y = \frac{1}{1+e^{-x}}$$



ReLU(Rectified Linear Unit)

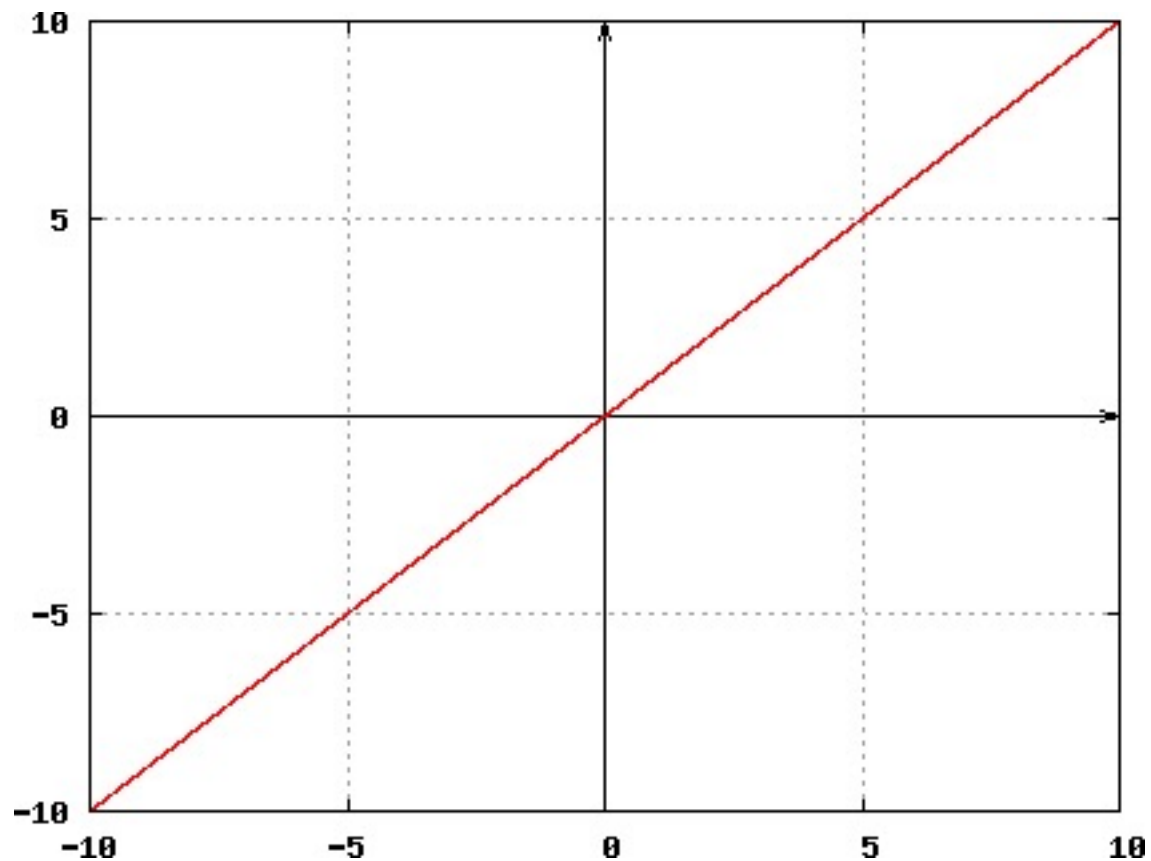
- 가장 많이 쓰이는 함수 중 하나

$$y = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases}$$



Identity Function(항등 함수)

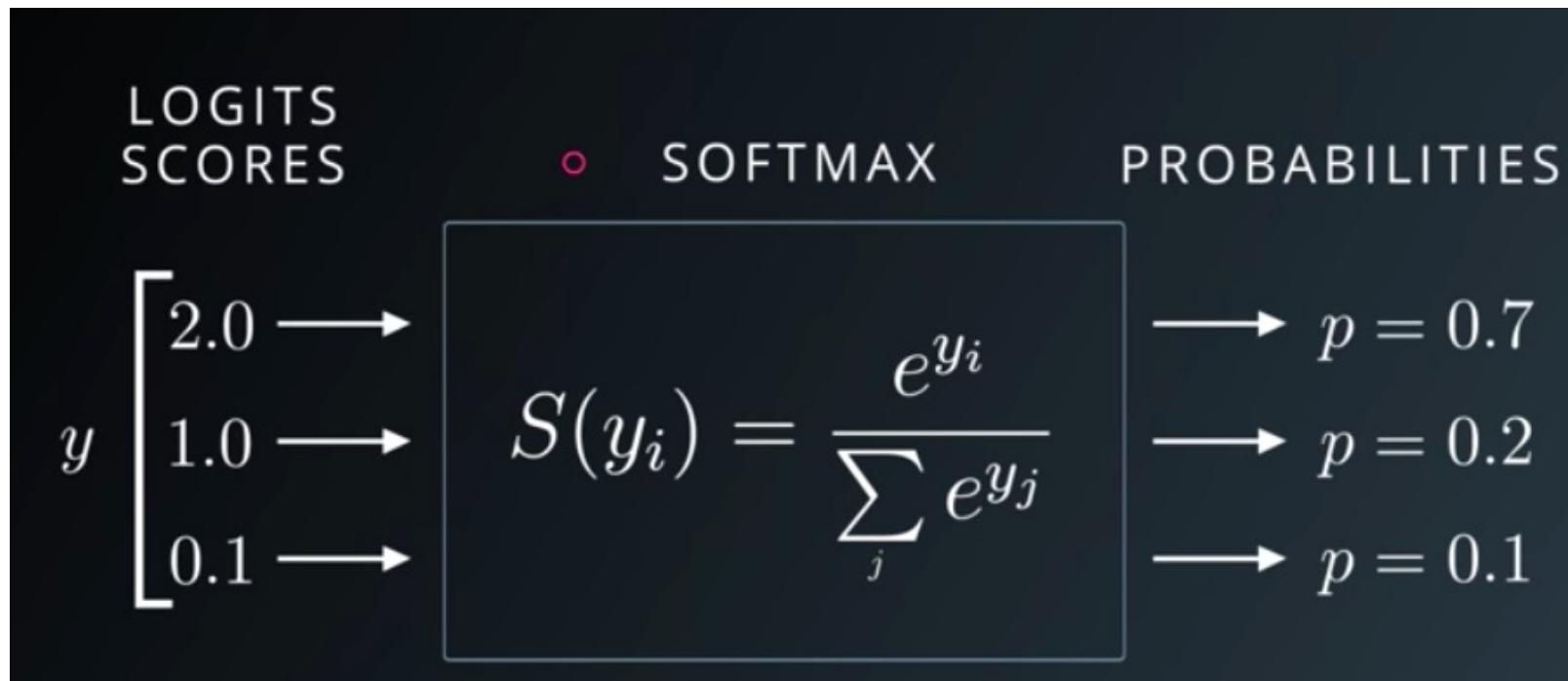
- 회귀(Regression) 문제에서 주로 사용
 - 출력층의 활성화 함수로 활용
- $y = x$
- 입력값 그대로 출력하기 때문에 굳이 정의할 필요는 없지만
신경망 중간 레이어 흐름과 통일하기 위해 사용



Softmax

- 다중 클래스 분류에 사용(Multi Class Classification)
- 입력값의 영향을 크게 받음
입력값이 크면 출력값도 큼
- 출력값을 확률에 대응가능
- 출력값의 총합은 1
- 수식

$$y_k = \frac{\exp(a_k)}{\sum_{i=1} \exp(a_i)}$$



소프트맥스 함수 주의점

- 오버플로우(overflow) 문제
- 지수함수(exponential function)을 사용하기 때문에
입력값이 너무 크면 무한대(inf)가 반환됨
- 개선한 수식

$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1} \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1} \exp(a_i)} \\ &= \frac{\exp(a_k + \log C)}{\sum_{i=1} \exp(a_i + \log C)} \\ &= \frac{\exp(a_k + C')}{\sum_{i=1} \exp(a_i + C')} \end{aligned}$$

활성화 함수를 비선형 함수(non-linear function)로 사용하는 이유

- 신경망을 깊게(deep) 하기 위함
- 만약 활성화 함수를 선형함수(linear function)으로 하게 되면 은닉층의 갯수가 여러개이더라도 의미가 없어짐
- 만약, $h(x) = cx$ 이고, 3개의 은닉층이 존재한다면

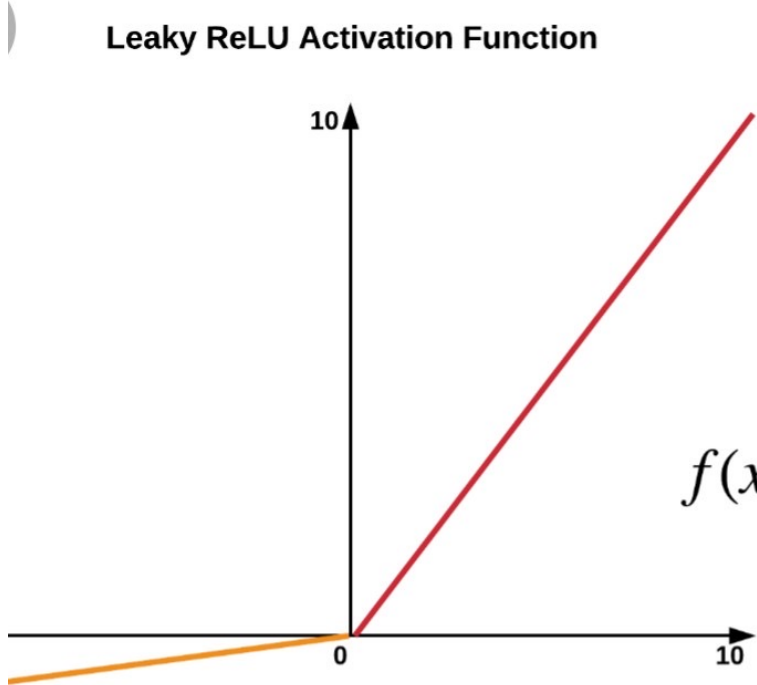
$$\begin{aligned}y &= h(h(h(x))) \\&= c * c * c * x \\&= c^3 x\end{aligned}$$

이므로 결국에는 선형함수가 되어버림

그 외의 활성화 함수

- LeakyReLU

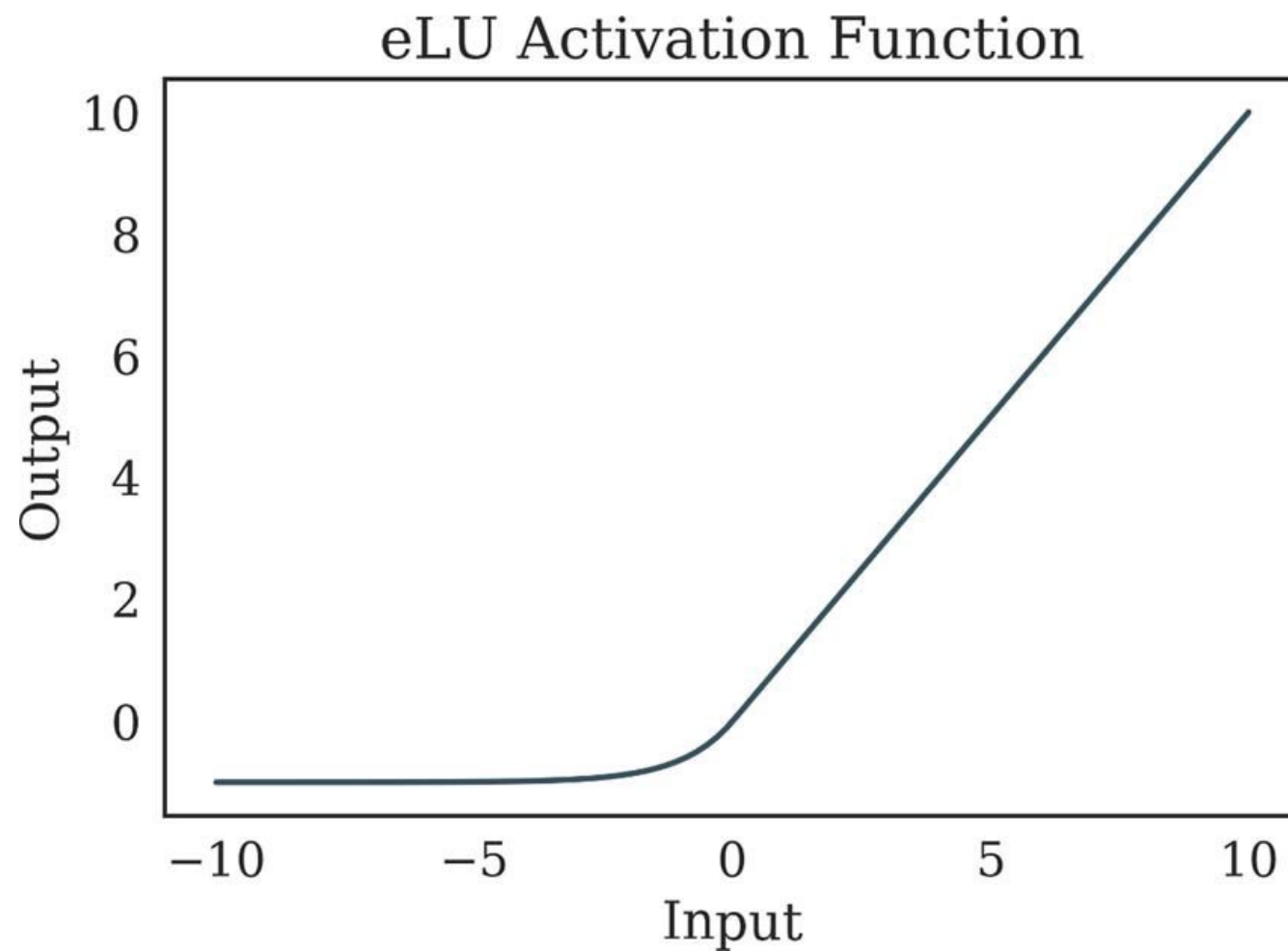
$$f_a(x) = \begin{cases} x & (x \geq 0) \\ ax & (x < 0) \end{cases}$$



$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

ELU(Exponential Linear Units)

$$f(\alpha, x) = \begin{cases} \alpha (e^x - 1) & (x \leq 0) \\ x & (x > 0) \end{cases}$$



3층 신경망 구현하기

- 2클래스 분류
- 입력층(Input Layer)
 - 뉴런수: 3
- 은닉층(Hidden Layers)
 - 첫번째 은닉층
 - 뉴런수: 3
 - 두번째 은닉층
 - 뉴런수: 2
- 출력층(Output Layer)
 - 뉴런수: 2