Lecture 2

Deep Neural Network
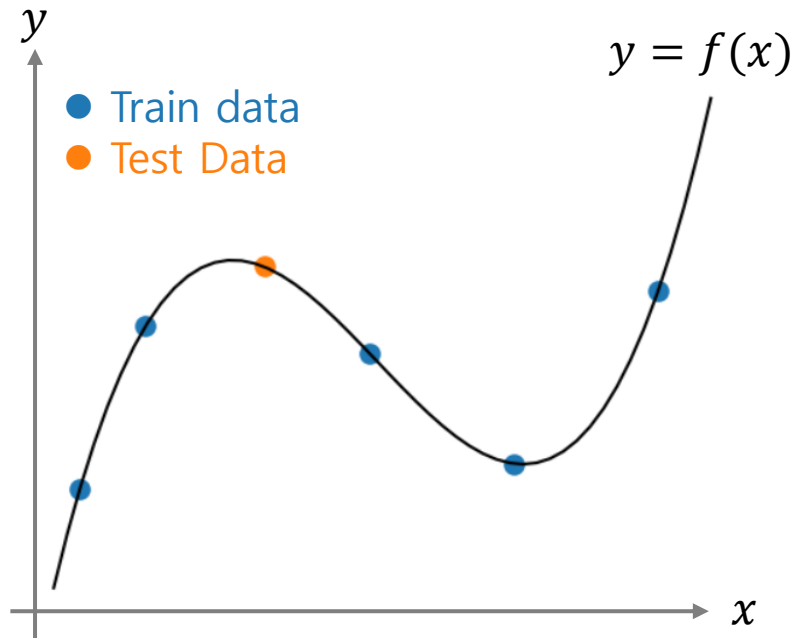
Seoul National University

Human Interface Laboratory

Contents

- **Neural Network is a Function Approximator**

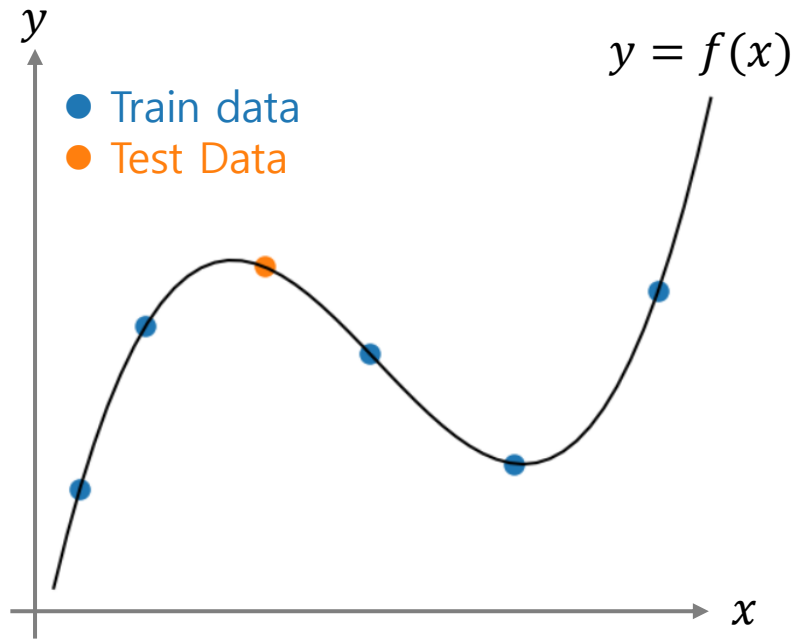- **Perceptron**

- **Deep Neural Network**

Neural Network is a Function Approximator

$y = f(x)$

● Train data
● Test Data

➥ Unknown function f를
N.N. (f_θ) 으로 approximate 하겠다!

- 매우 많은 $(x_i, y_i) \in TrainData$가 주어질 때, $y_i \approx f_\theta(x_i)$ 가 되도록 Neural Network f_θ를 학습하겠다

- [희망 사항] 학습때 쓰이지 않은 $(x_i, y_i) \in TestData$ 에 대해서도 $y_i \approx f_\theta(x_i)$ 가 될 것이다!

Neural Network is a Function Approximator

y

● Train data
● Test Data

$y = f(x)$

x

➥ Unknown function f 를
N.N. (f_θ) 으로 approximate 하겠다!

f (함수)	x (입력)	y (출력)
$f(x) = x^3$	1.1	1.331
	-0.3	-0.027
And Gate	(1, 1)	1
	(1, 0)	0
Image Classifier		"강아지"
Language Model (ex. ChatGPT)	"안녕하세"	"요"
	"Question"	"Answer"

Neural Network is a Function Approximator
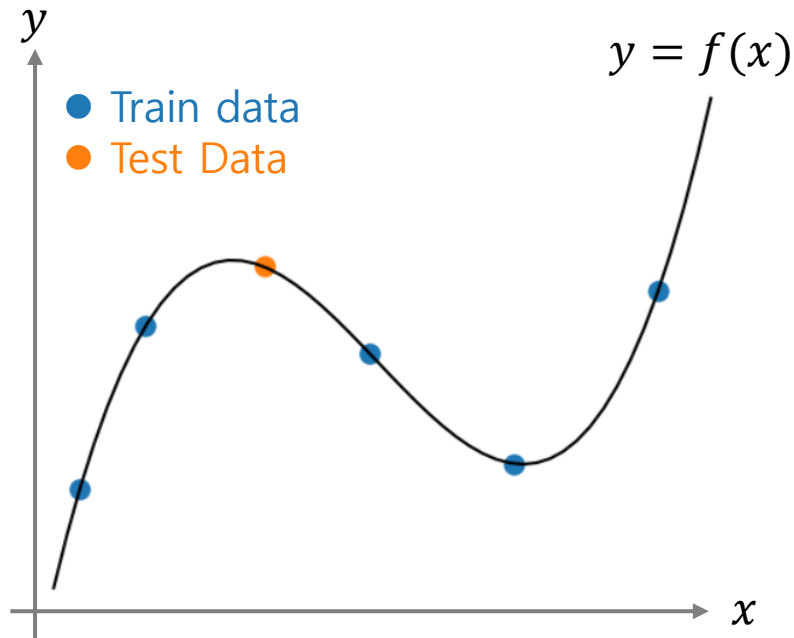
$y = f(x)$

● Train data
● Test Data

x
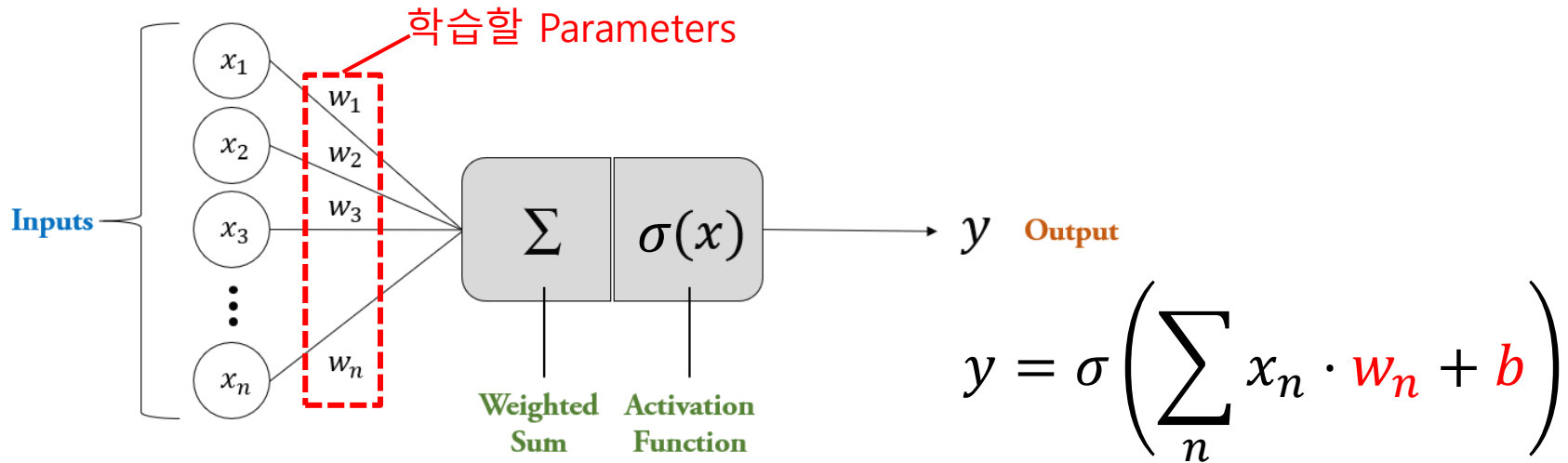
↳ Unknown function f를
N.N. (f_θ) 으로 approximate 하겠다!

- 매우 많은 $(x_i, y_i) \in TrainData$가
  주어질 때, $y_i \approx f_\theta(x_i)$ 가 되도록
  <u>Neural Network f_θ</u>를 <u>학습</u>하겠다
  ①   ②

- [희망 사항] 학습때 쓰이지 않은
  $(x_i, y_i) \in TestData$ 에 대해서도
  $y_i \approx f_\theta(x_i)$ 가 될 것이다!

Motivation (Synapse & Neuron)

W_m : 뉴런 간의 연결 강도

Perceptron

학습할 Parameters

$$y = \sigma\left(\sum_n x_n \cdot w_n + b\right)$$

Inputs

x_1

x_2

x_3

x_n

w_1

w_2

w_3

w_n

Σ $\sigma(x)$

y Output
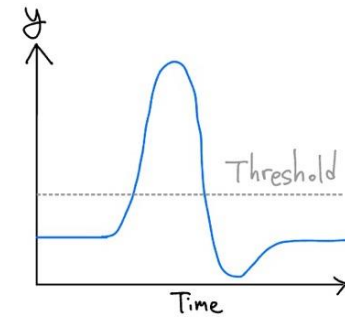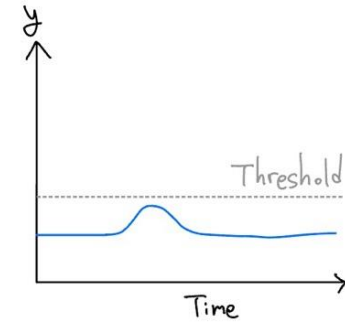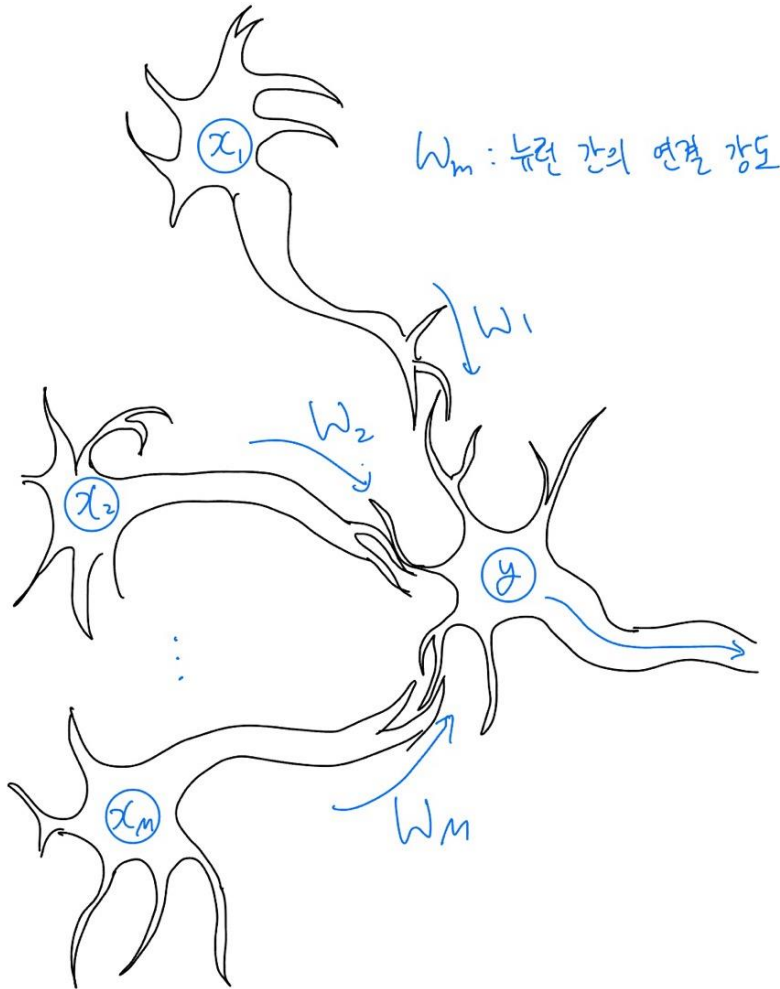
Weighted Sum   Activation Function

- $\sigma(\cdot)$: Activation function (비선형 함수)

Sigmoid

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

ReLU

$$y = \max(0, x)$$

Perceptron 실습 (AND Gate)

학습할 Parameters

Inputs

x_1
x_2
x_3
x_n

w_1
w_2
w_3
w_n

$\sum$ $\sigma(x)$

Weighted Sum   Activation Function

y Output

$$y = \sigma\left(\sum_n x_n \cdot w_n + b\right)$$
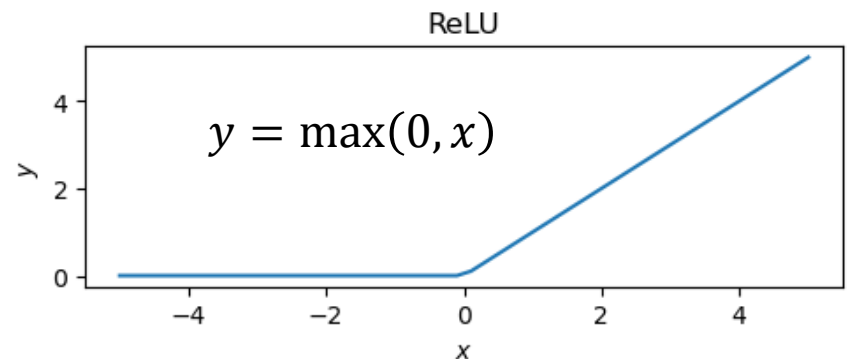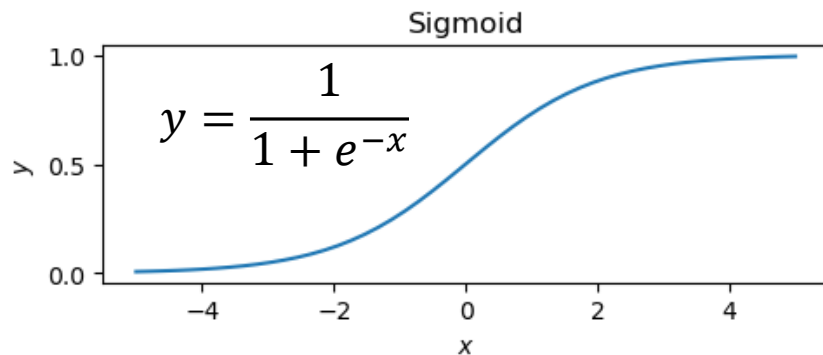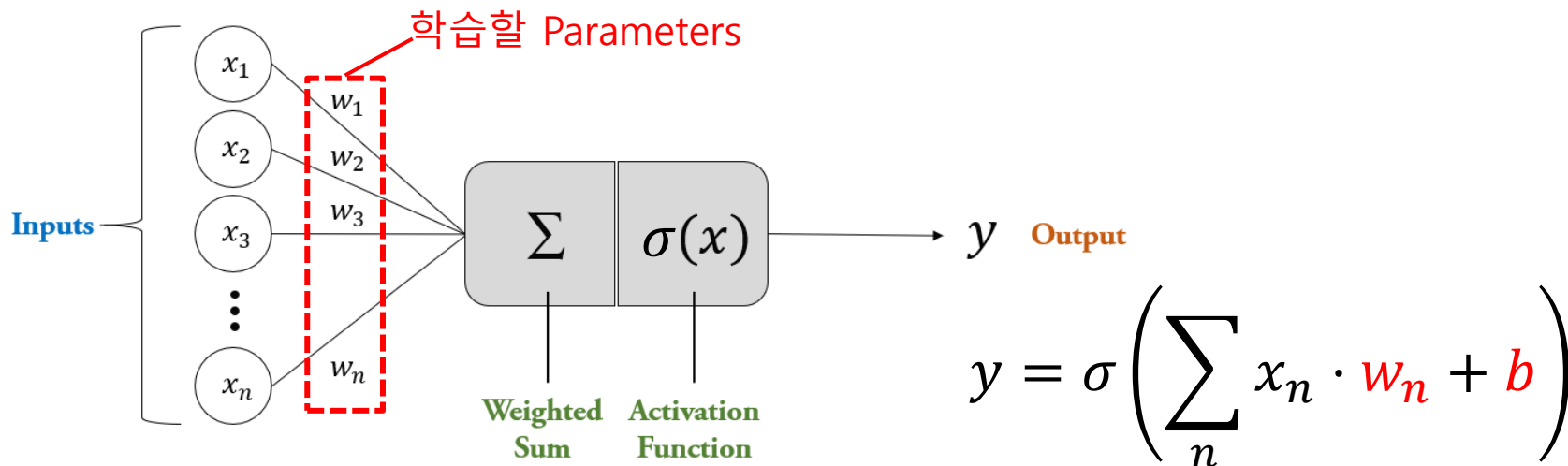
- Lab2-1. PyTorch Tensor

- Lab2-2. Perceptron (AND Gate)

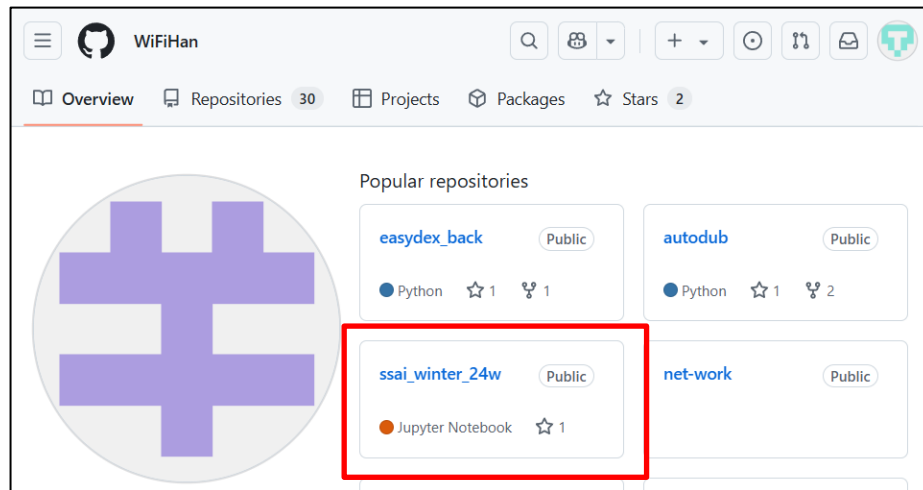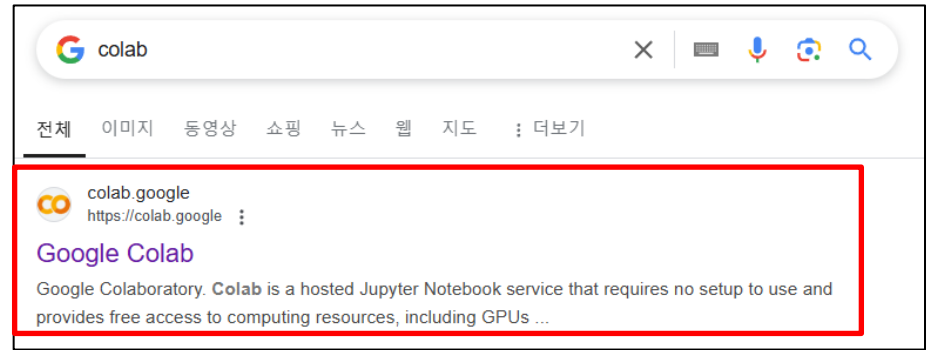# Perceptron 실습 (AND Gate)

1. github.com/wifihan 접속

2. ssai_winter_24w 클릭

3. lec2/lec2.ipynb 다운로드
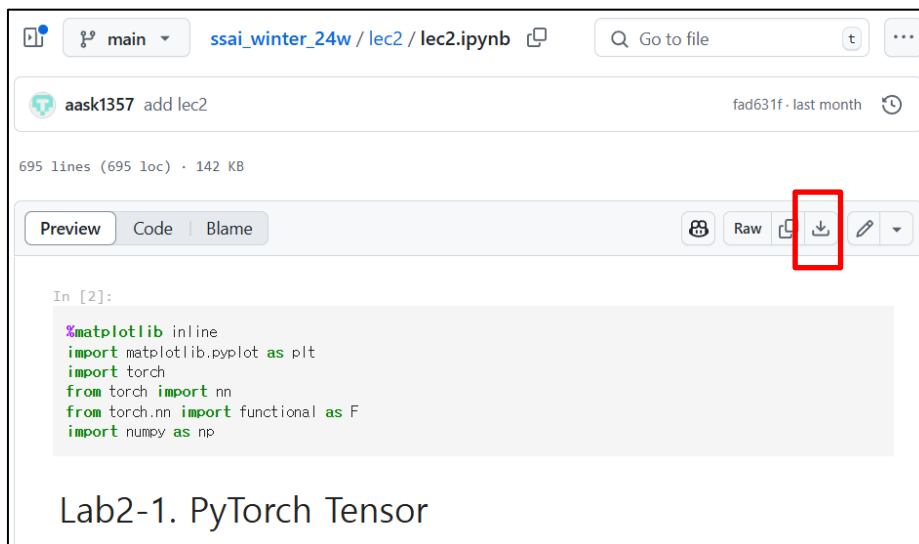
Perceptron 실습 (AND Gate)

4. 구글에 colab 검색 후 접속

4. Open Colab -> 업로드 (3.에서 다운받았던 lec2.ipynb)
5. 런타임 유형 GPU로 변경
6. 연결 확인

Perceptron 학습

학습할 Parameters

Inputs

x_1
x_2
x_3
$\vdots$
x_n

w_1
w_2
w_3
w_n

$\sum$ $\sigma(x)$ → y Output

Weighted Sum   Activation Function

$$y = \sigma\left(\sum_n x_n \cdot w_n + b\right)$$
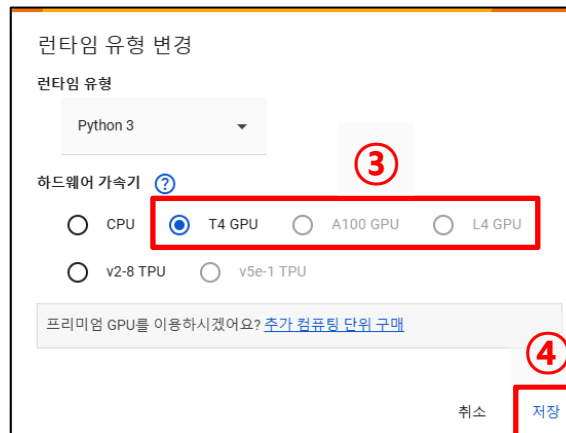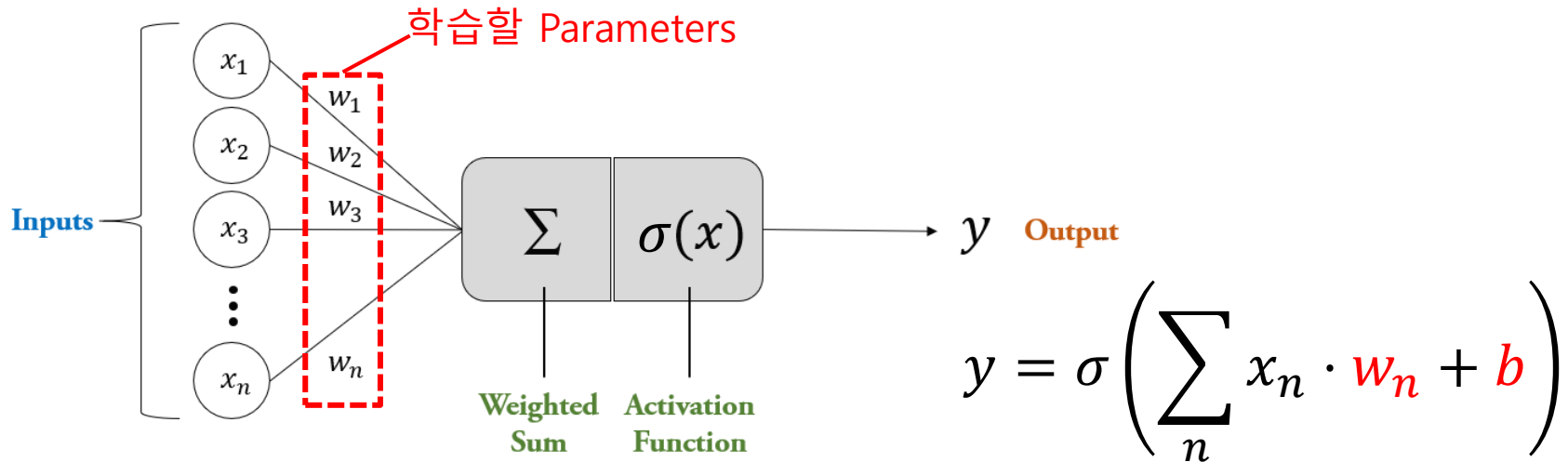
1. Parameter $\theta = [w, b]$ 를 random한 값으로 초기화

2. $(x_i, y_i) \in TrainData$ 에 대해 $y_i \approx f_\theta(x_i)$ 되도록 모델의 parameter θ를 학습

➢ [한계] 매우 간단한 함수 (ex. AND gate)만 approximate 가능. 조금만 복잡해져도 (ex. XOR gate) approximate 불가능.

Deep Neural Network

h^1 … h^{L-1}

입력 $x = h^0$

출력 $\hat{y} = h^L$

f^1: Layer 1 f^2: Layer 2 … f^L: Layer L

- Deep Neural Network (DNN)는 여러 Layer들로 이루어져 있음
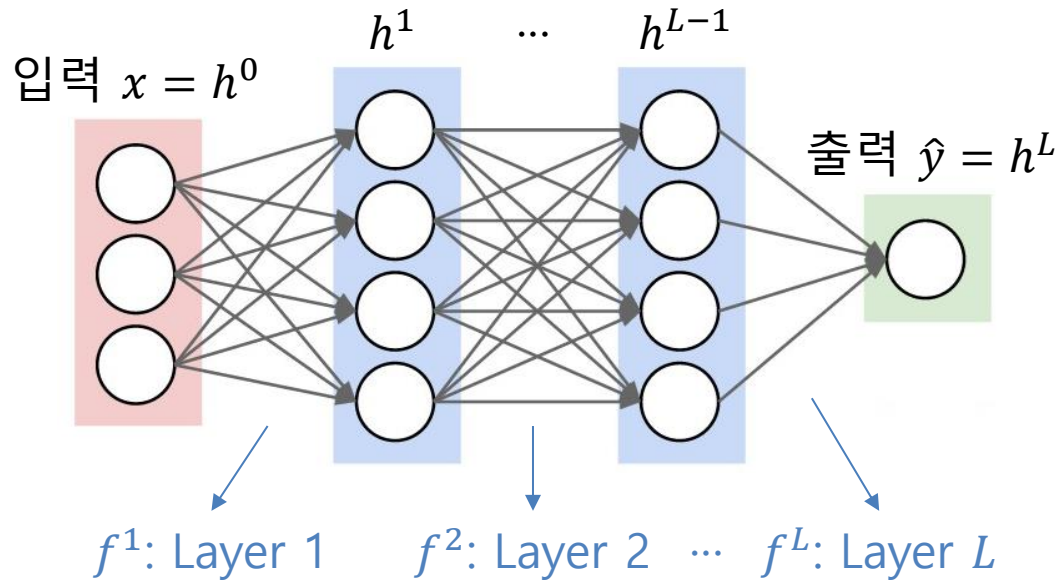
$$\hat{y} = f(x) = f^L \left(f^{L-1} \left(\cdots f^1(x) \right) \right)$$

- Layer 종류: <u>Fully Connected Layer</u>, Convolutional Layer, Recurrent Layer, Transformer, …

Neural Network 학습 알고리즘

1. $\hat{y} = f_\theta(x)$: Forward Propagation

2. $Loss = \mathcal{L}(\hat{y}, y)$: 정답과의 차이 계산

   Ex) Mean squared error (MSE) loss $= \frac{1}{B}\sum_{i=1}^{B}(y_i - \hat{y}_i)^2$

3. $\frac{\partial \mathcal{L}}{\partial \theta}$ 계산   : Backpropagation

4. $\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$: Gradient descent

   Learning rate, 충분히 작은 상수 (ex. 0.001)

위 1~4를 계속 반복하면 Loss가 줄어든다

➢ $(x_i, y_i) \in TrainData$ 에 대해 $y_i \approx f_\theta(x_i)$ 가 된다

➢ $(x_i, y_i) \in TestData$ 에 대해서도 $y_i \approx f_\theta(x_i)$ 가 되길 희망

Neural Network 학습 알고리즘

1. $\hat{y} = f_\theta(x)$: Forward Propagation

2. $Loss = \mathcal{L}(\hat{y}, y)$: 정답과의 차이 계산

   Ex) Mean squared error (MSE) loss $= \frac{1}{B}\sum_{i=1}^{B}(y_i - \hat{y}_i)^2$

3. $\frac{\partial \mathcal{L}}{\partial \theta}$ 계산 : <u>Backpropagation</u>

4. $\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$: <u>Gradient descent</u>

   Learning rate, 충분히 작은 상수 (ex. 0.001)

위 1~4를 계속 반복하면 Loss가 줄어든다
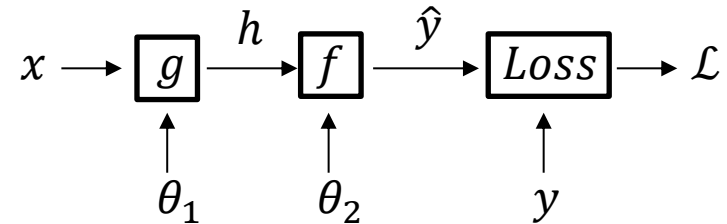- $(x_i, y_i) \in TrainData$ 에 대해 $y_i \approx f_\theta(x_i)$ 가 된다
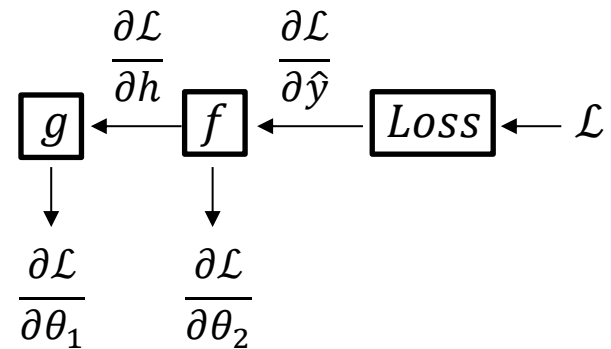- $(x_i, y_i) \in TestData$ 에 대해서도 $y_i \approx f_\theta(x_i)$ 가 되길 희망

Chain Rule

- $y = g\big(f(x)\big)$

- $y' = g'\big(f(x)\big) \cdot f'(x)$

- $\Longleftrightarrow \dfrac{\partial y}{\partial x} = \dfrac{\partial g}{\partial f} \cdot \dfrac{\partial f}{\partial x}$

Backpropagation Using Chain Rule

- $h = g_{\theta_1}(x)$

- $\hat{y} = f_{\theta_2}(h)$

- $\mathcal{L} = Loss(\hat{y}, y)$

- $\dfrac{\partial \mathcal{L}}{\partial \theta_2} = \dfrac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial \theta_2}$

- $\dfrac{\partial \mathcal{L}}{\partial \theta_1} = \dfrac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \dfrac{\partial \hat{y}}{\partial h} \cdot \dfrac{\partial h}{\partial \theta_1}$
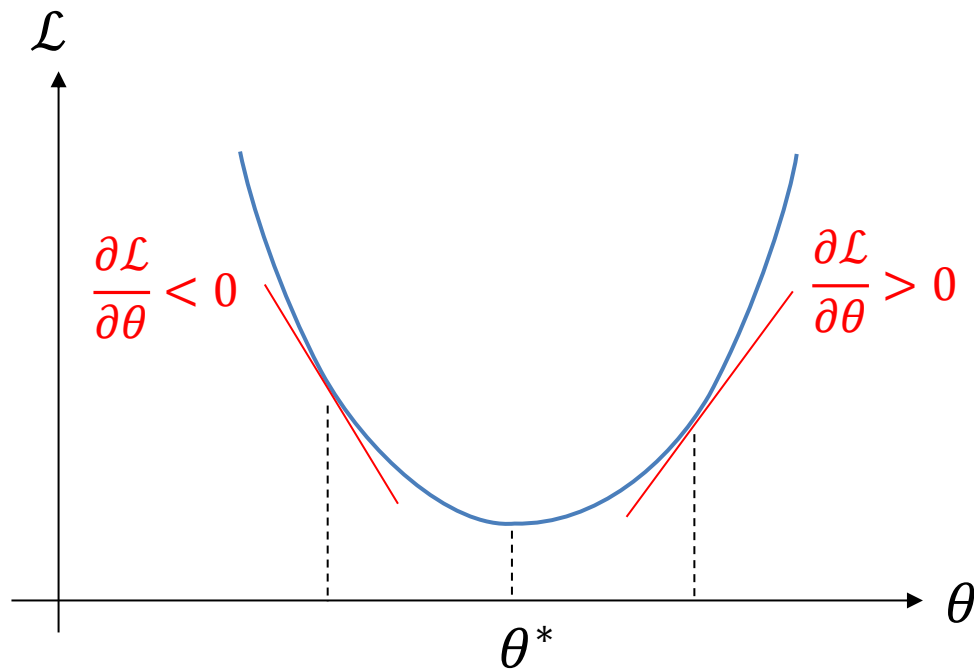
Gradient Descent

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$$

Learning rate, 충분히 작은 상수 (ex. 0.001)

Neural Network 학습을 위한 준비물

- **Data**
  - 학습을 위한 $(x_i, y_i) \in TrainData$
  - 검증을 위한 $(x_i, y_i) \in TestData$

- **Model**
  - Fully Connected Layer, Convolutional Layer, Recurrent Layer, Transformer, …

- **Training Algorithm**
  - Loss
  - Backpropagation
  - Optimizer (ex. Gradient descent)

Neural Network 학습을 위한 준비물

- **Data**
  - 학습을 위한 $(x_i, y_i) \in TrainData$
  - 검증을 위한 $(x_i, y_i) \in TestData$
- **Model**
  - Fully Co ... ent Layer, Transfo ...
- **Training Algori...**
  - Loss
  - Backpropagation
  - Optimizer (ex. Gradient descent)

Neural Network 실습 (Regression)

- Lab2-3 ~ 2-5. Training a Perceptron (AND Gate)
- Lab2.6 Deep Neural Network (Regression)