

Lecture 2

Deep Neural Network



Seoul National University



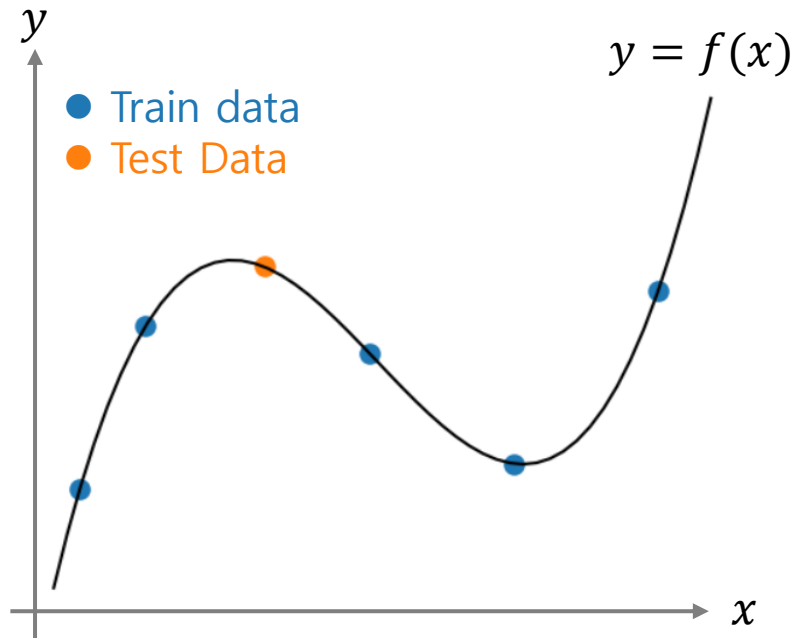
Human Interface Laboratory

Contents

- **Neural Network is a Function Approximator**
- **Perceptron**
- **Deep Neural Network**



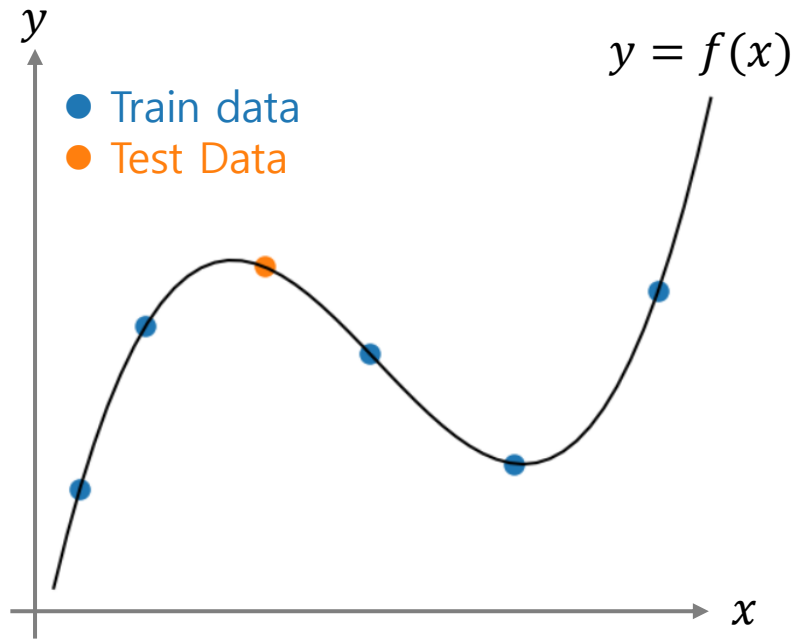
Neural Network is a Function Approximator




- 매우 많은 $(x_i, y_i) \in TrainData$ 가 주어질 때, $y_i \approx f_{\theta}(x_i)$ 가 되도록 Neural Network f_{θ} 를 학습하겠다
- [희망 사항] 학습때 쓰이지 않은 $(x_i, y_i) \in TestData$ 에 대해서도 $y_i \approx f_{\theta}(x_i)$ 가 될 것이다!

➡ Unknown function f 를
N.N. (f_{θ}) 으로 approximate 하겠다!

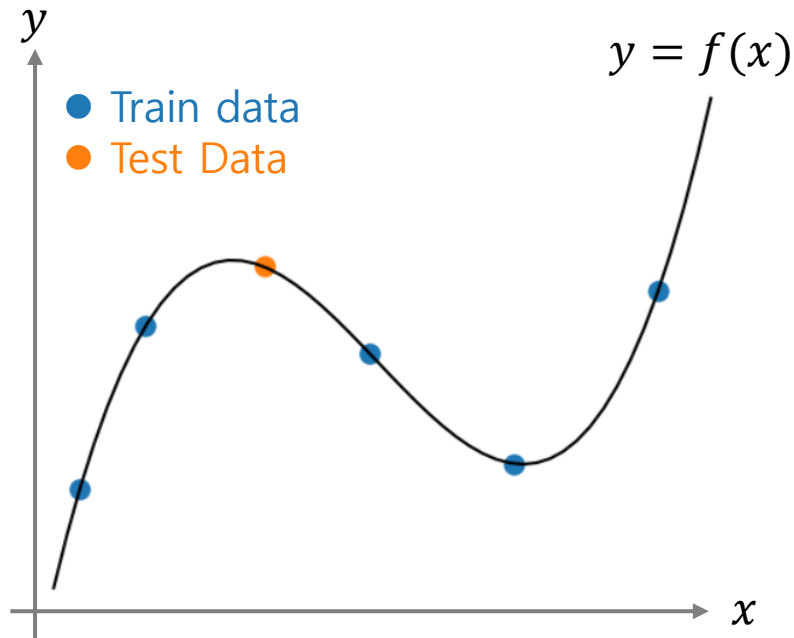
Neural Network is a Function Approximator



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)	"안녕하세요"	"요"
	"Question"	"Answer"

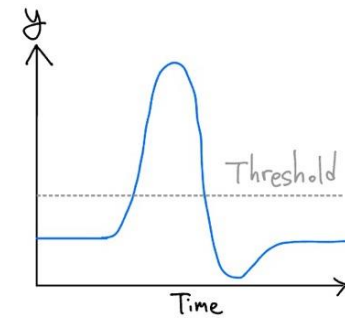
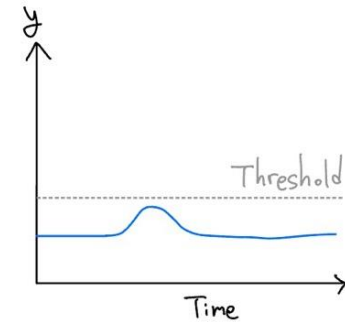
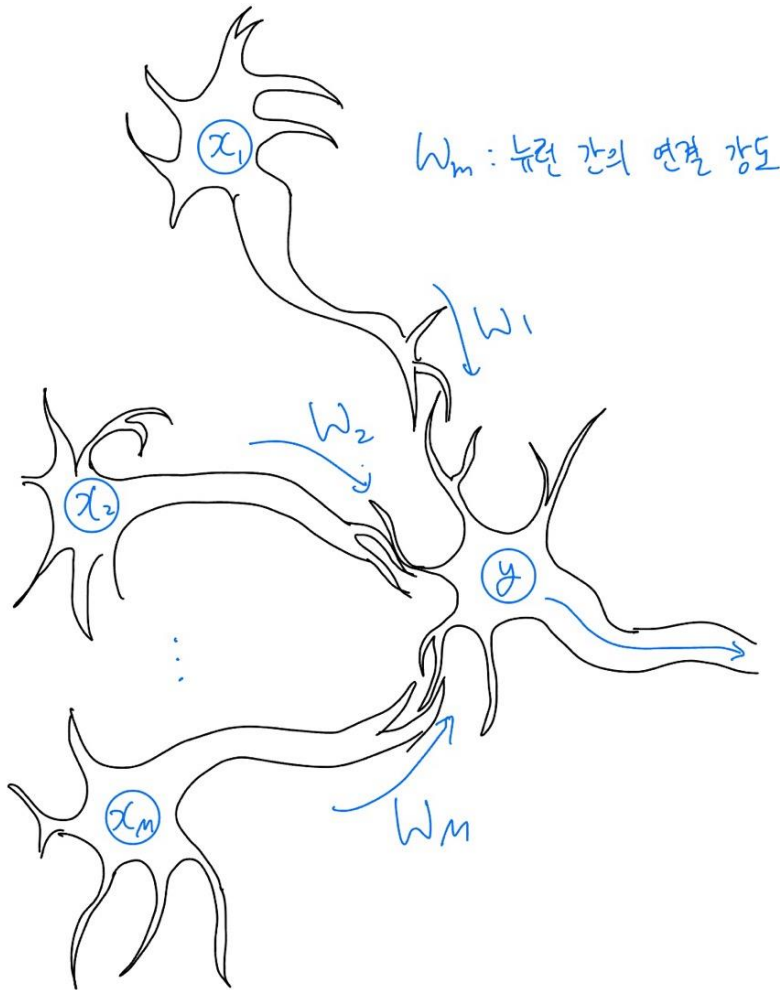
Neural Network is a Function Approximator



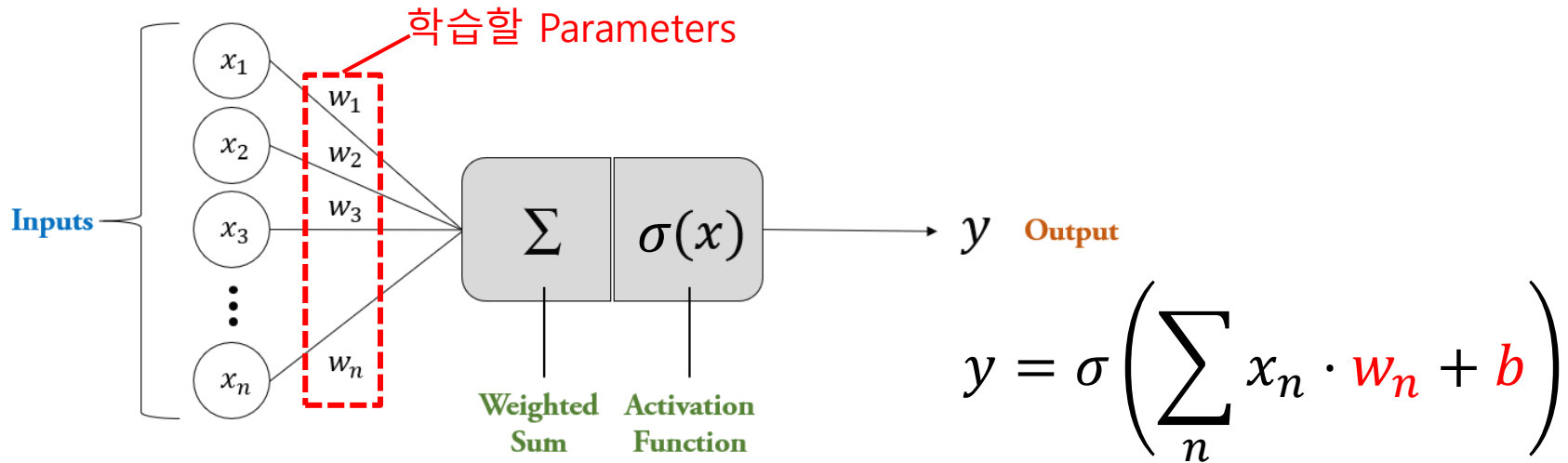
➡ Unknown function f 를
N.N. (f_{θ}) 으로 approximate 하겠다!

- 매우 많은 $(x_i, y_i) \in TrainData$ 가 주어질 때, $y_i \approx f_{\theta}(x_i)$ 가 되도록 Neural Network f_{θ} 를 학습하겠다
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!

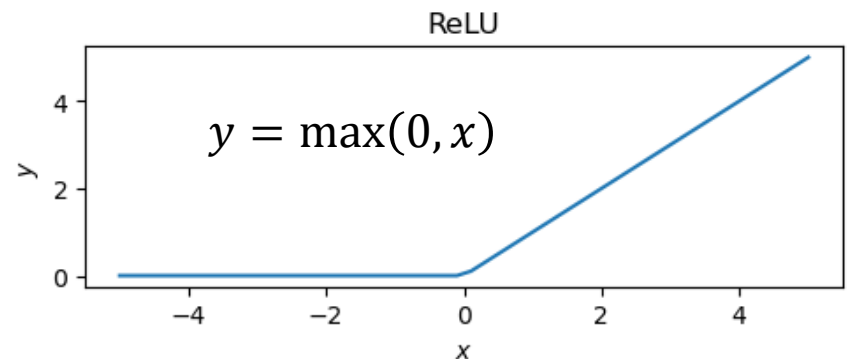
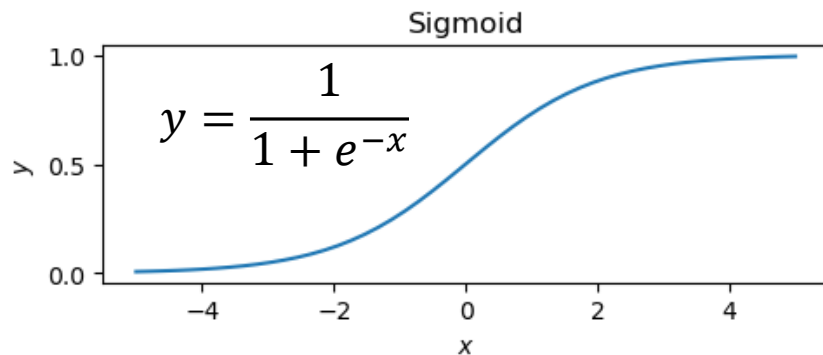
Motivation (Synapse & Neuron)



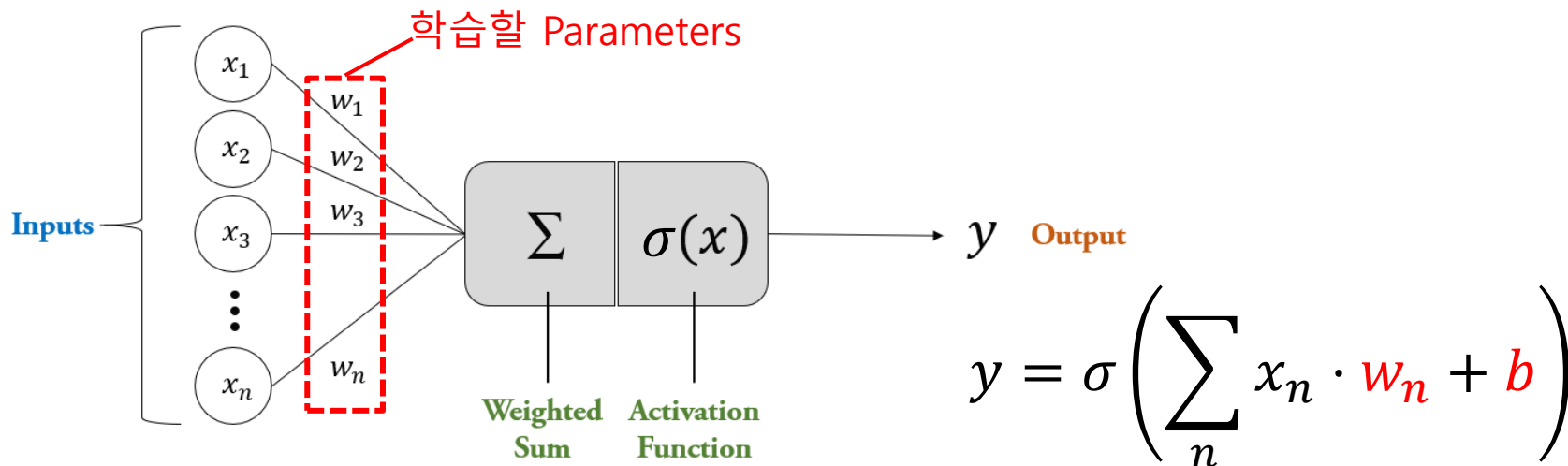
Perceptron



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

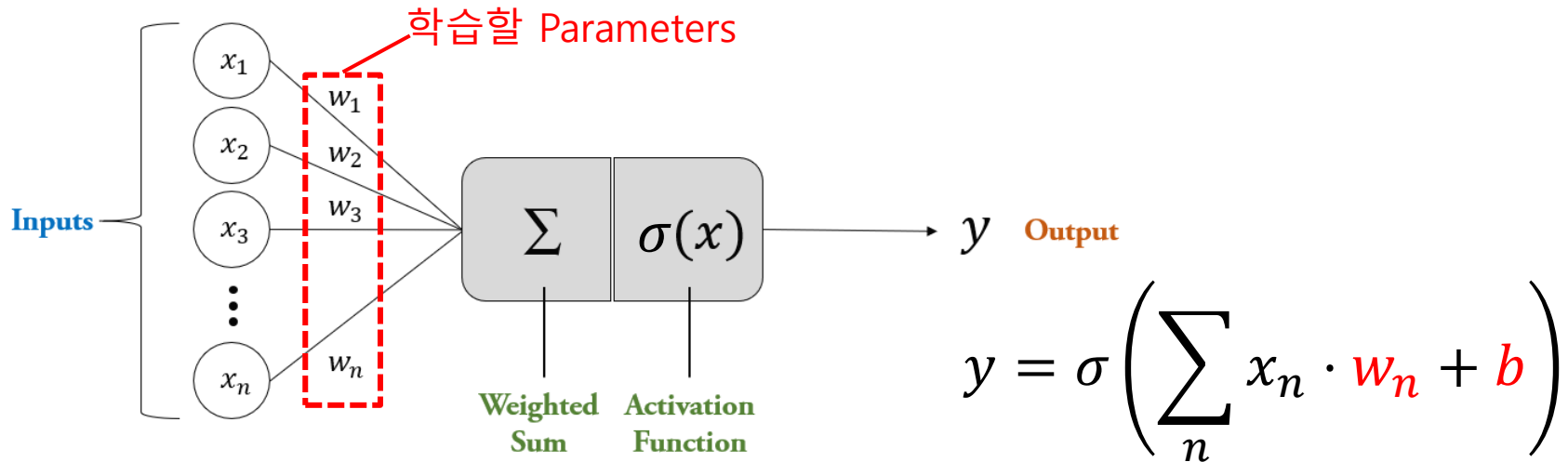


Perceptron 실습 (AND Gate)



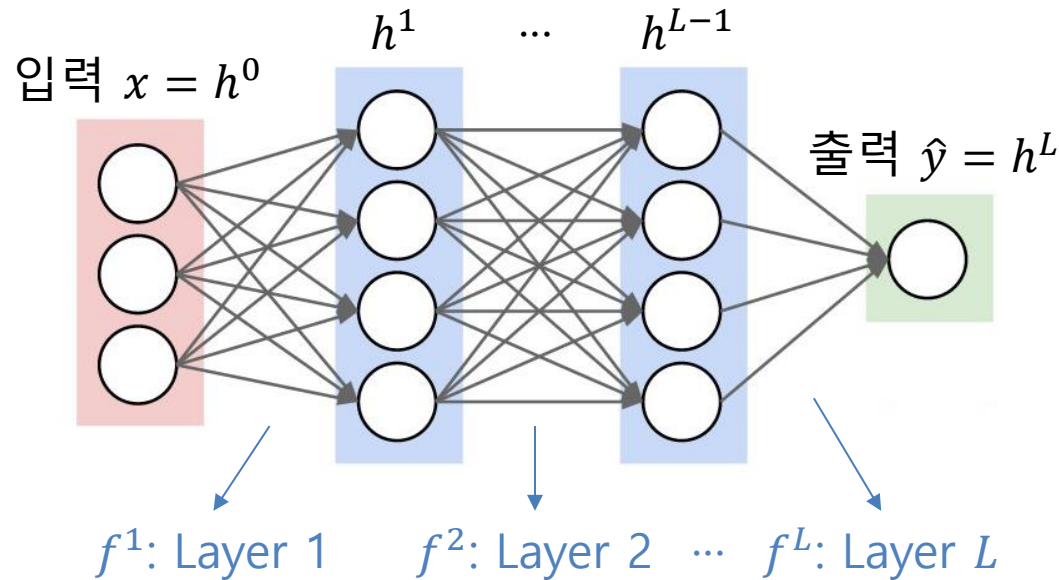
- Lab2-1. PyTorch Tensor
- Lab2-2. Perceptron (AND Gate)

Perceptron 학습



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 θ 를 학습
- [한계] 매우 간단한 함수 (ex. AND gate)만 approximate 가능. 조금만 복잡해져도 (ex. XOR gate) approximate 불가능.

Deep Neural Network



- Deep Neural Network (DNN)는 여러 Layer들로 이루어져 있음

$$\hat{y} = f(x) = f^L \left(f^{L-1} (\dots f^1(x)) \right)$$

- Layer 종류: Fully Connected Layer, 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

Chain Rule

- $y = g(f(x))$
- $y' = g'(f(x)) \cdot f'(x)$
- $\Leftrightarrow \frac{\partial y}{\partial x} = \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial x}$

Backpropagation Using Chain Rule

- $h = g_{\theta_1}(x)$
- $\hat{y} = f_{\theta_2}(h)$
- $\mathcal{L} = \text{Loss}(\hat{y}, y)$

- $\frac{\partial \mathcal{L}}{\partial \theta_2} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_2}$
- $\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h} \cdot \frac{\partial h}{\partial \theta_1}$

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 Connected Layer, Convolutional Layer, Recurrent Layer, Transformer, etc.

- **Training Algorithm**

- 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)

