FastCampus Pytorch

Ch3. Neural Networks

HARRY KIM

Lecture Content

1. Neural Network
2. Activation Function
3. MNIST
4. Batch Training

강의 자료

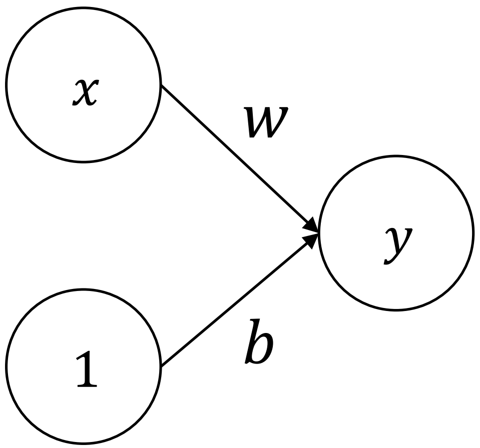
- Books
  - Pattern Classification Second Edition [Duda, 2001]
  - Pattern Recognition And Machine Learning [Bishop, 2006]
  - 밑바닥부터 시작하는 딥러닝 [사이토 고키, 2017]
  - 머신러닝, 딥러닝 실전개발 입문 [쿠지라 히코우즈쿠에, 2017]
- Online
  - UVA DEEP LEARNING COURSE [University of Amsterdam, 2018]
  - CS231n [http://cs231n.stanford.edu/, 2018]
  - Machine Learning [https://ko.coursera.org/learn/machine-learning, 2018]

1. Neural Network

Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 선형 회귀식은 하나의 퍼셉트론으로도 구현 가능
  - 퍼셉트론 : 다수의 신호를 입력받아 하나의 신호를 출력
    - $x, 1$: 입력 신호
    - w, b : 가중치
    - y : 출력 신호

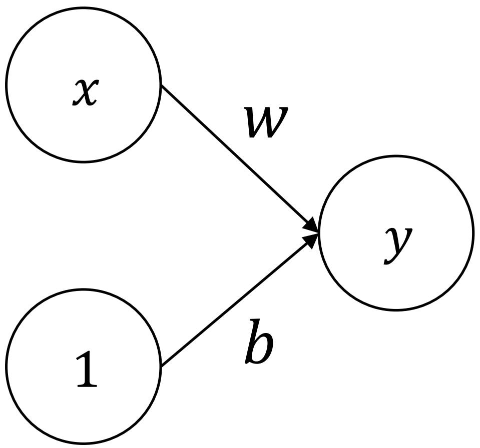
$$y = wx + b$$



Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 선형 회귀식은 하나의 퍼셉트론으로도 구현 가능
  - 퍼셉트론 : 다수의 신호를 입력받아 하나의 신호를 출력
    - $x, 1$: 입력 신호
    - w, b : 가중치
    - y : 출력 신호

$$y = wx + b$$



역전파를 활용하여 좋은 가중치(w, b)를 찾는 것이 목표

Neural Network

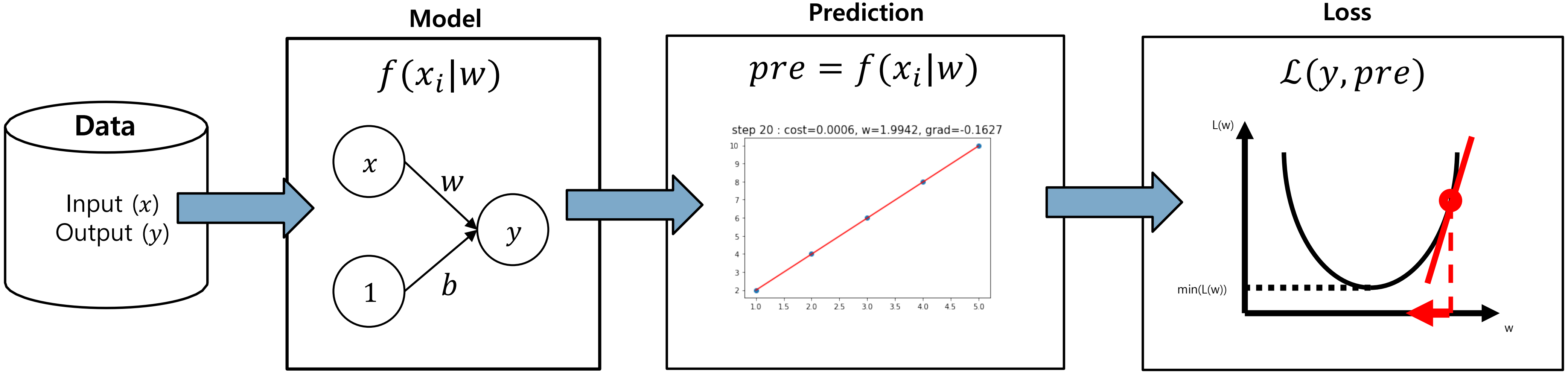
Neural Network

Activation Function

MNIST

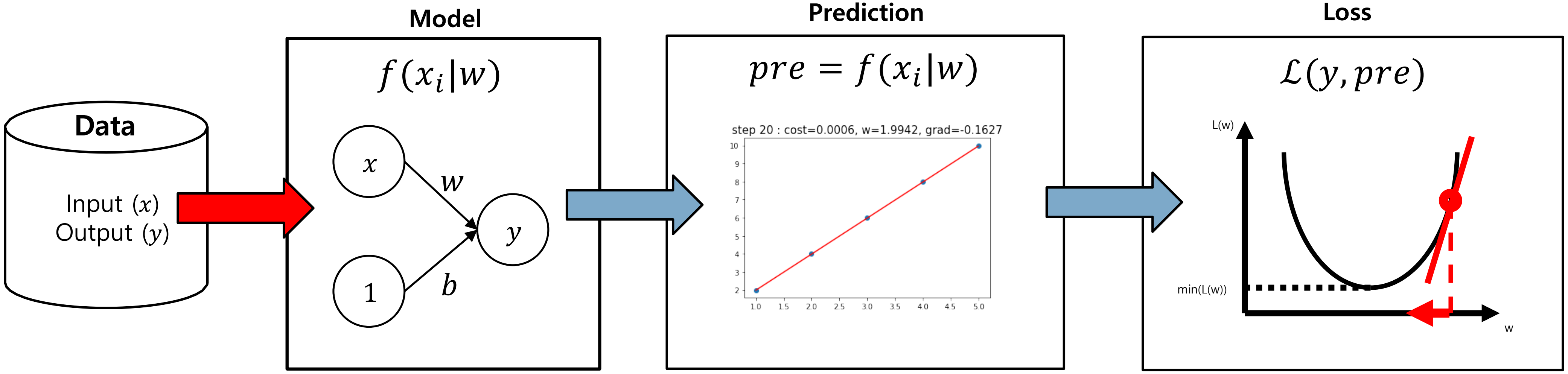
Batch Training

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리



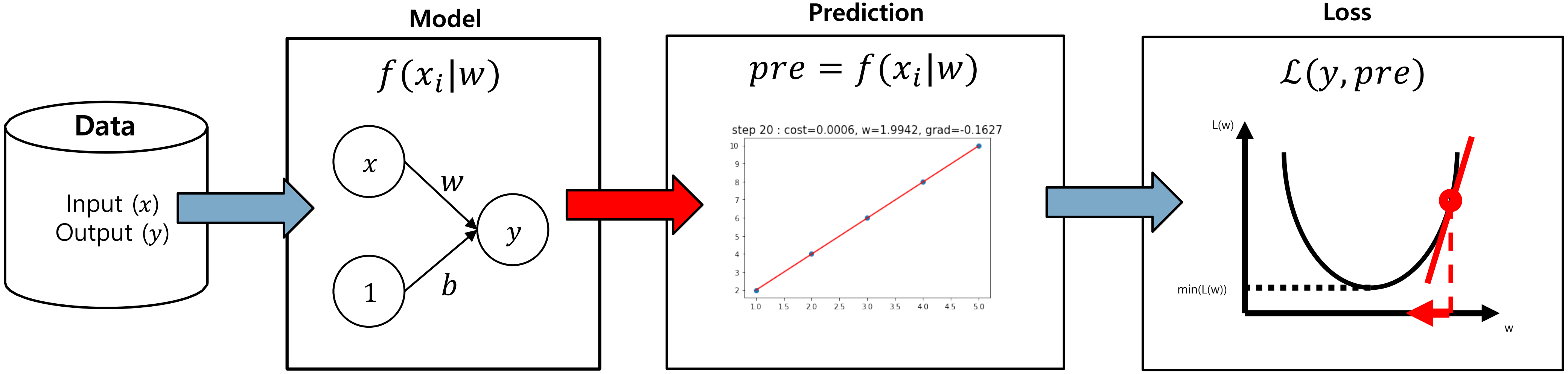
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현



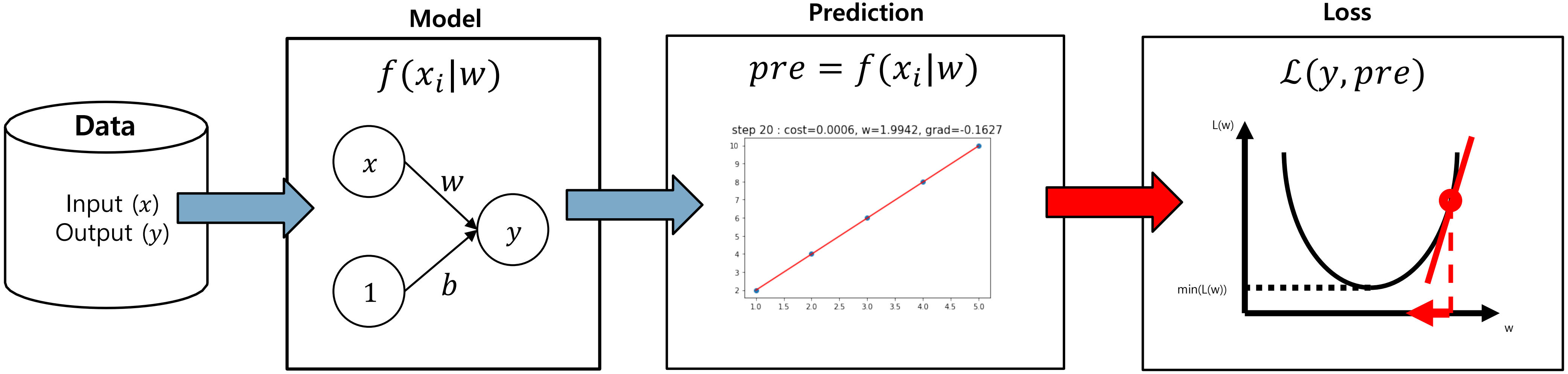
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현 > 예측값 도출



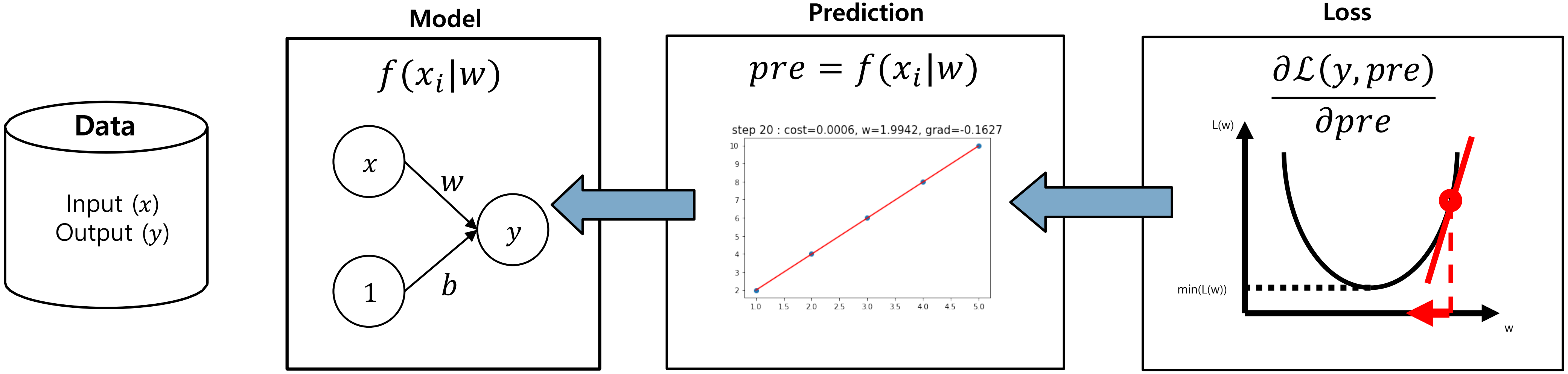
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현 > 예측값 도출 > 손실함수 계산



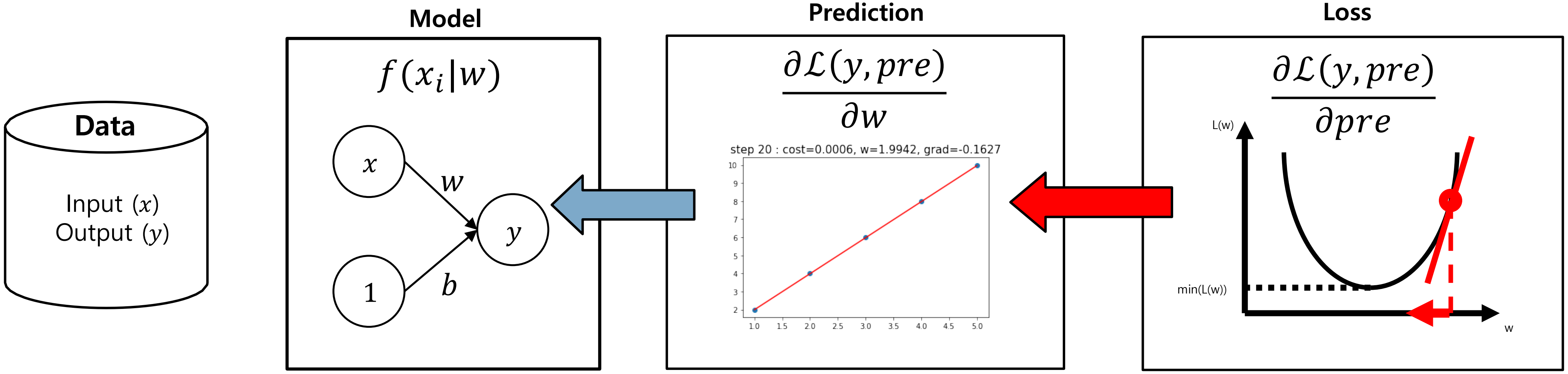
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현 > 예측값 도출 > 손실함수 계산
  - 역전파(Backward)
    - 기울기 계산



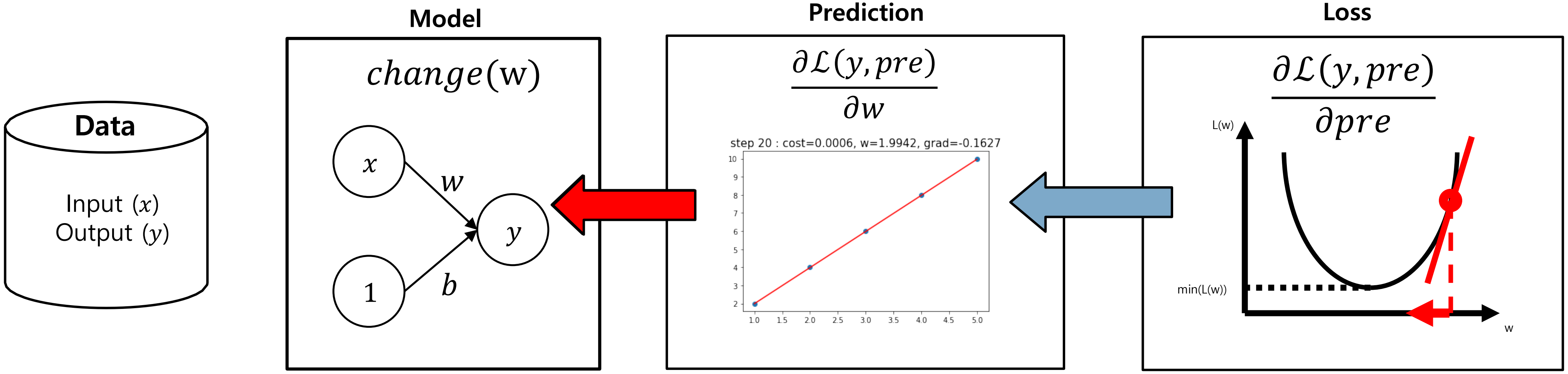
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현 > 예측값 도출 > 손실함수 계산
  - 역전파(Backward)
    - 기울기 계산 > 개선 방향 확정



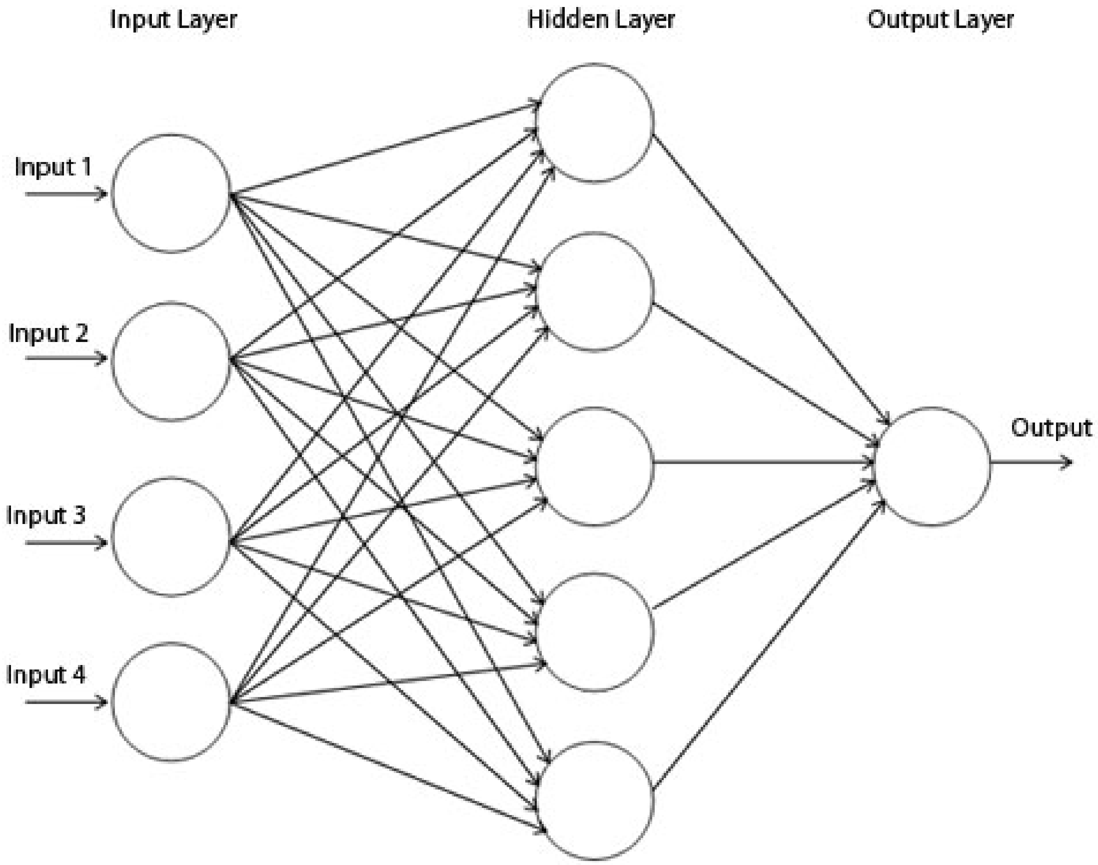
Neural Network

- **Prev : 선형 회귀(Linear Regression)와 신경망(Neural Network)**
  - 순전파(Forward)
    - 데이터 처리 > 모델 구현 > 예측값 도출 > 손실함수 계산
  - 역전파(Backward)
    - 기울기 계산 > 개선 방향 확정 > **가중치 개선 = $\boldsymbol{change}$ w!!!**



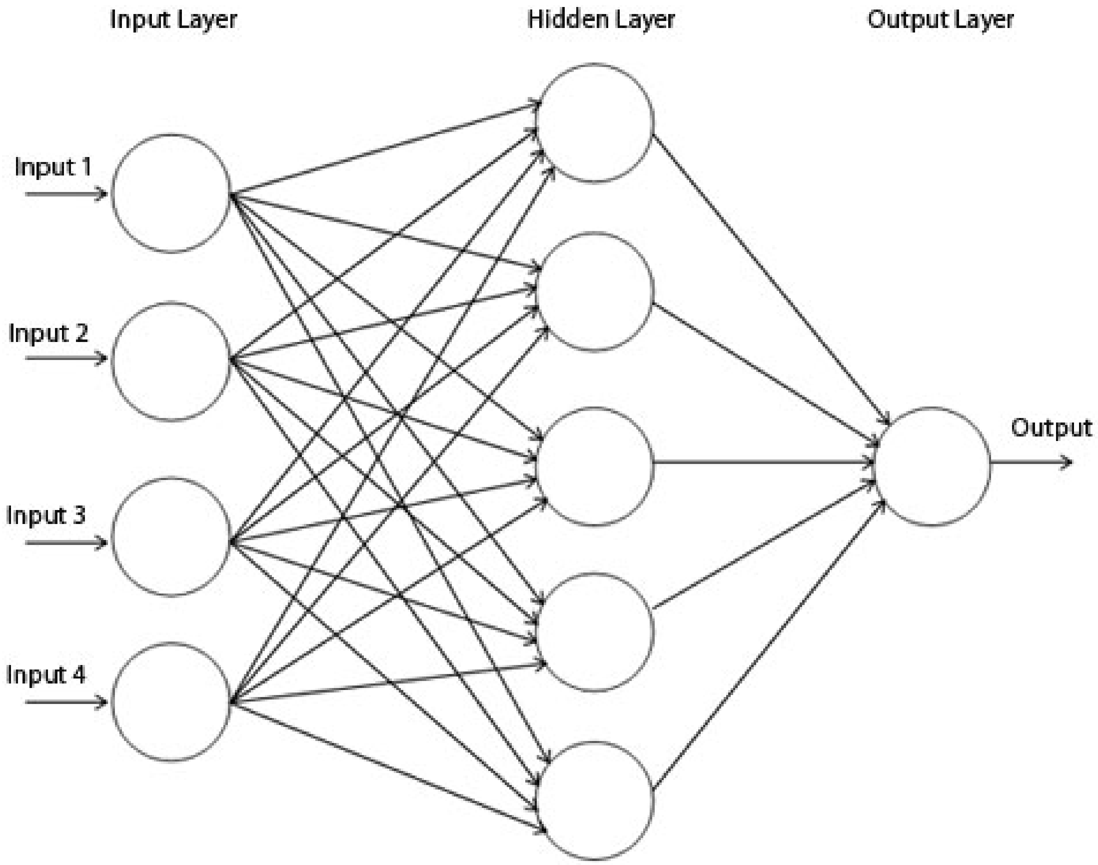
Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - 입력층(Input Layer)
  - 은닉층(Hidden Layer)
  - 출력층(Output Layer)



Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - 입력층(Input Layer)
  - 은닉층(Hidden Layer)
  - 출력층(Output Layer)



Q) 모든 뉴런(퍼셉트론)을
선형(Linear)으로 구현하면?

Neural Network

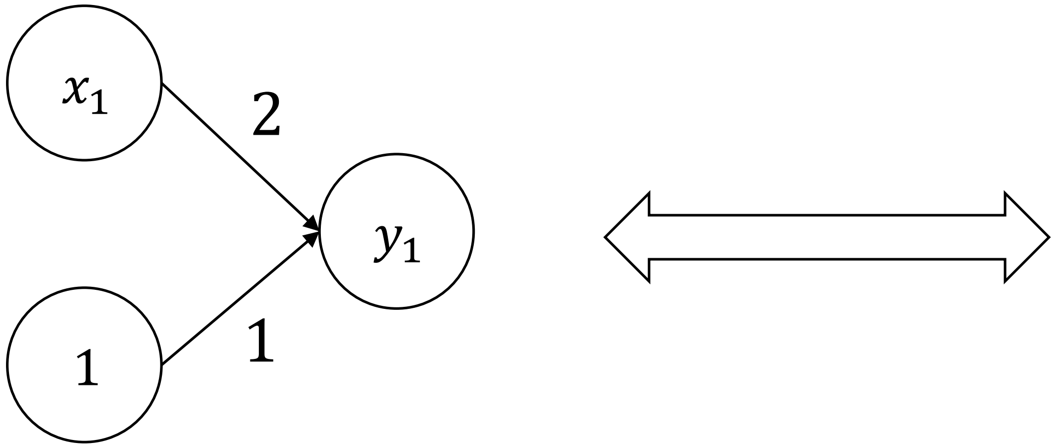
Neural Network

Activation Function

MNIST

Batch Training

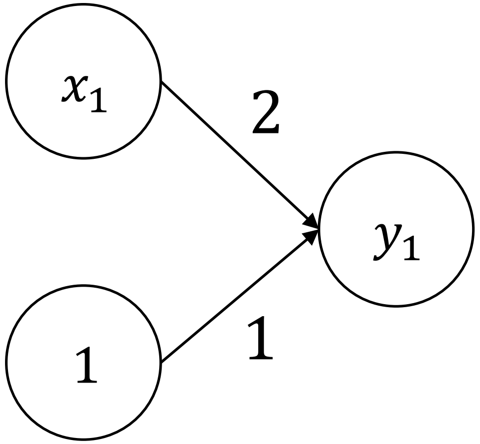
- **선형 신경망(Neural Network with Linear Module)**



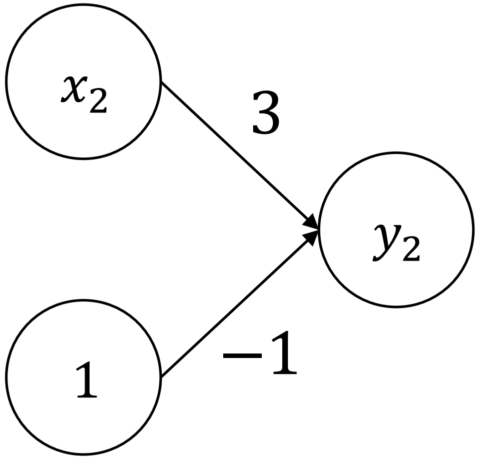
$$y_1 = 2x_1 + 1$$

Neural Network

- **선형 신경망(Neural Network with Linear Module)**



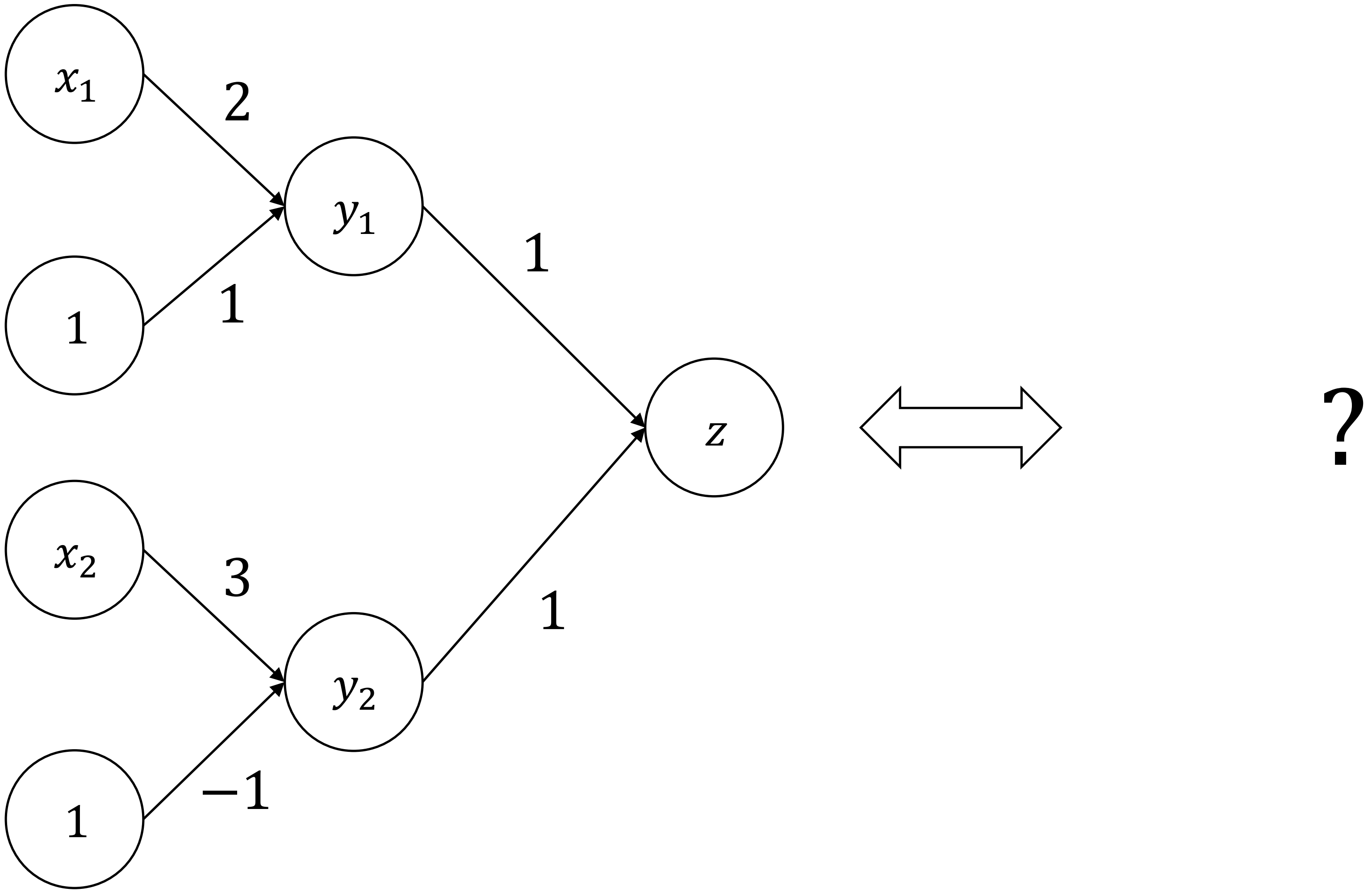
$$y_1 = 2x_1 + 1$$



$$y_2 = 3x_2 - 1$$

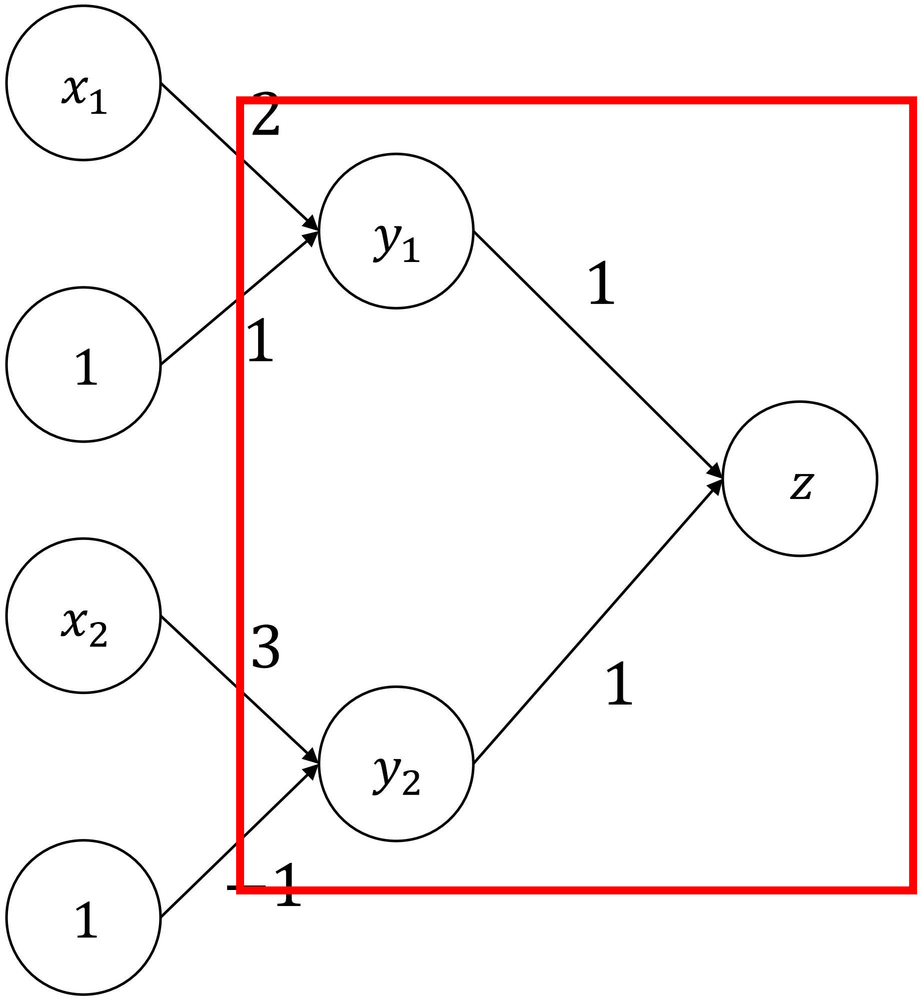
Neural Network

- **선형 신경망(Neural Network with Linear Module)**



Neural Network

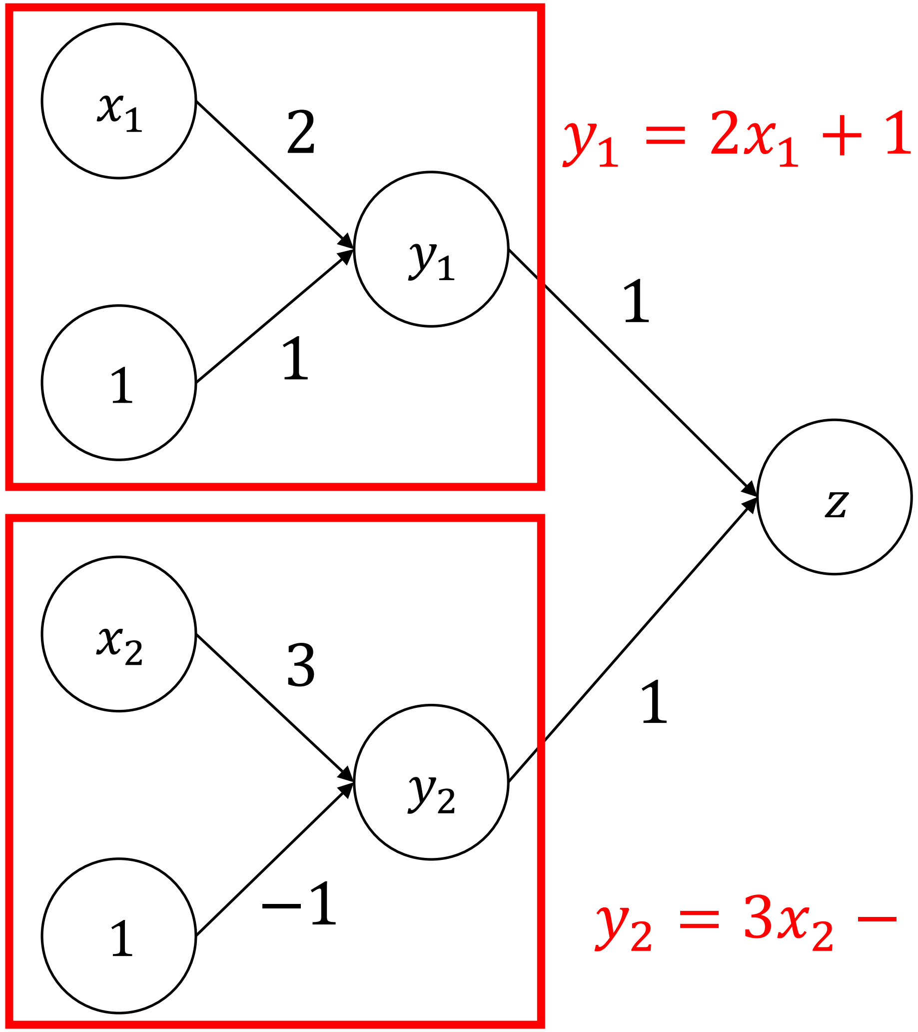
- **선형 신경망(Neural Network with Linear Module)**



$$z = 1 * y_1 + 1 * y_2$$

Neural Network

- 선형 신경망(Neural Network with Linear Module)



$y_1 = 2x_1 + 1$

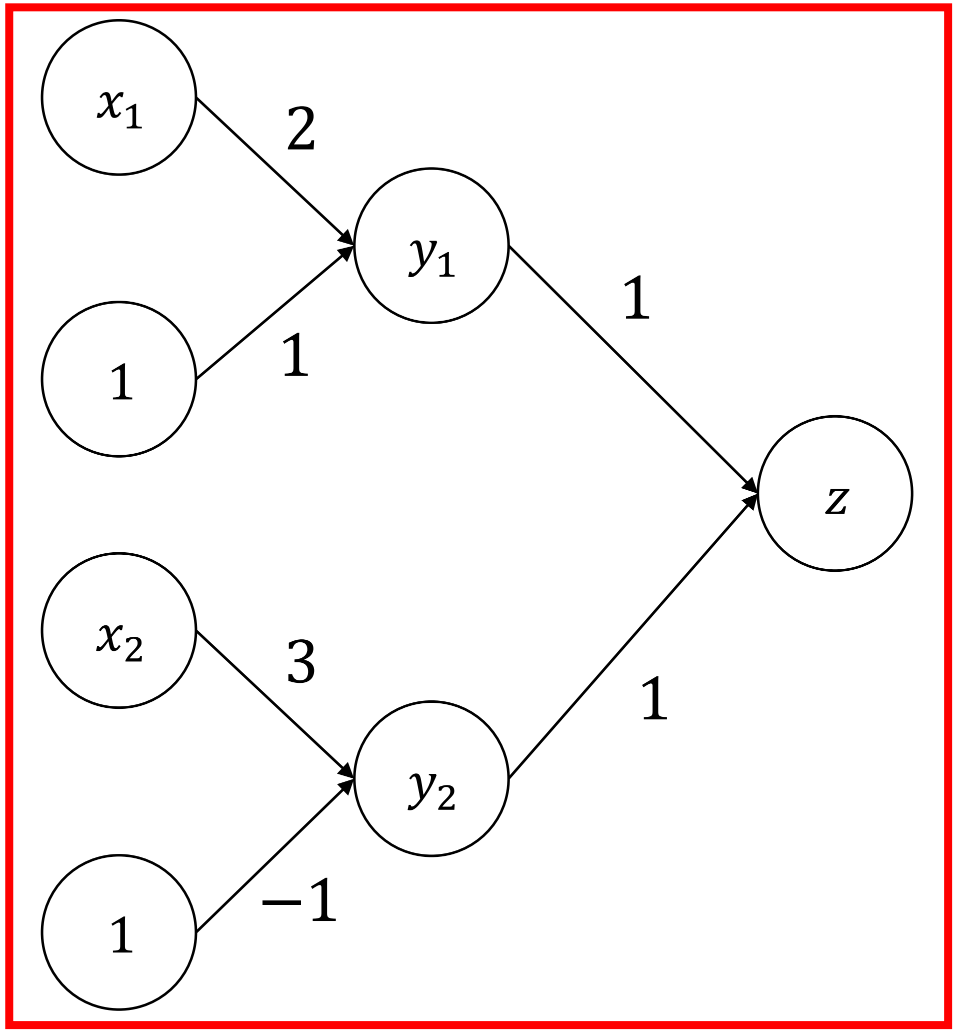
$y_2 = 3x_2 - 1$

$$z = 1 * y_1 + 1 * y_2$$

$$= 1 * (2x_1 + 1) + 1 * (3x_2 - 1)$$

Neural Network

- 선형 신경망(Neural Network with Linear Module)



$$z = 1 * y_1 + 1 * y_2$$

$$= 1 * (2x_1 + 1)$$
$$+1 * (3x_2 - 1)$$

$$= 2x_1 + 1 + 3x_2 - 1$$

$$= 2x_1 + 3x_2$$

Fast campus

Neural Network

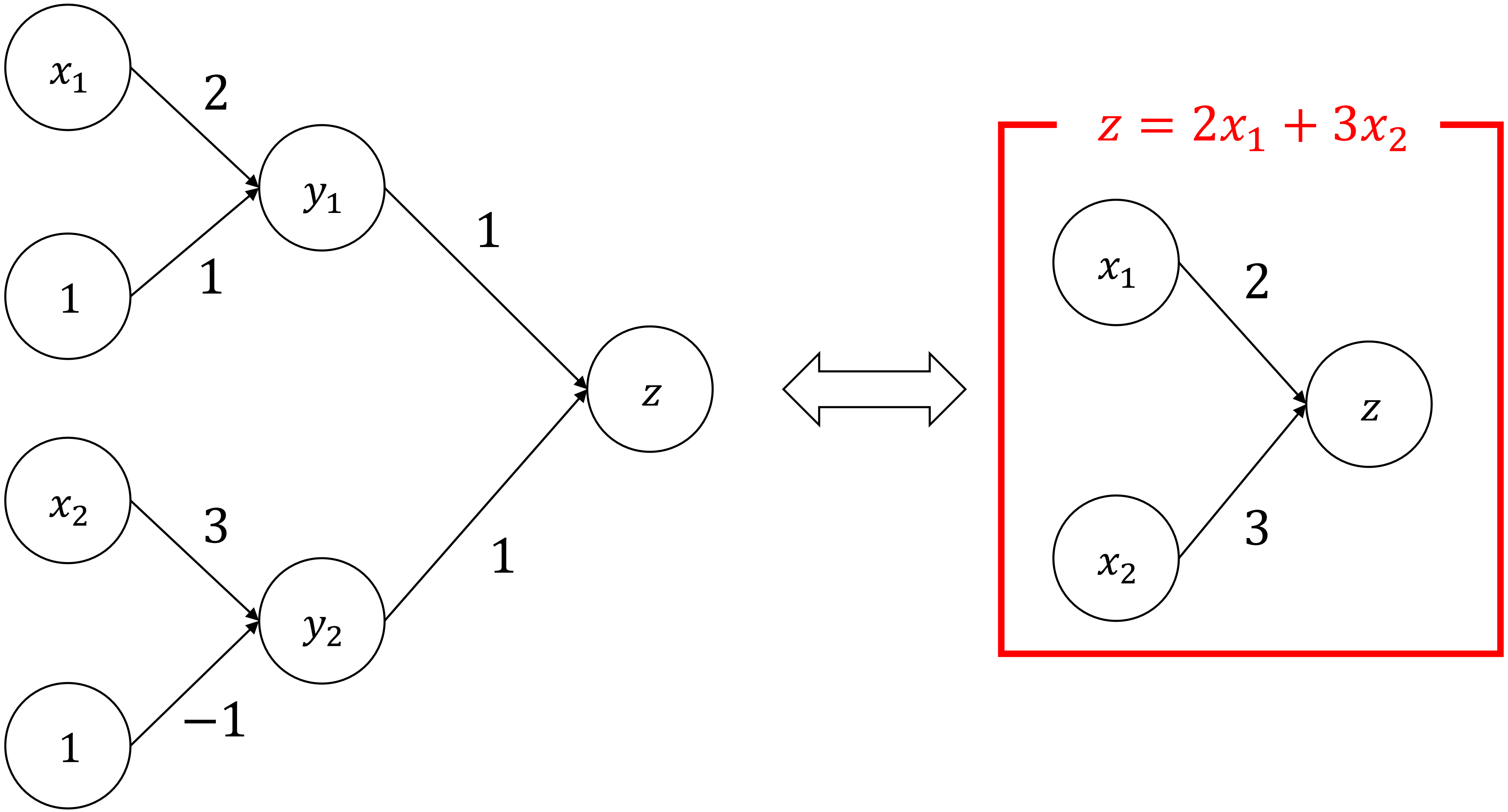
Neural Network

Activation Function

MNIST

Batch Training

- **선형 신경망(Neural Network with Linear Module)**



Neural Network

Neural Network

Activation Function

MNIST

Batch Training

- **선형 신경망(Neural Network with Linear Module)**
  - x : 입력값
  - w_l : l 번째 층의 가중치
  - $a_l = f_l(x|w_l) = w_l x + b_l$: l 층의 결과값

Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - x : 입력값
  - w_l : l 번째 층의 가중치
  - $a_l = f_l(x|w_l) = w_l x + b_l$: l 층의 결과값

$$a_L(x|w_{1,\ldots,L}) = f_L(f_{L-1}(\ldots f_1(x|w_1)|\ldots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\ldots f_2(w_1 x + b_1|w_2) \ldots w_{L-1})|\ w_L)$$

Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - x : 입력값
  - w_l : l 번째 층의 가중치
  - $a_l = f_l(x|w_l) = w_l x + b_l$: l 층의 결과값

$$a_L(x|w_{1,\dots,L}) = f_L(f_{L-1}(\dots f_1(x|w_1)|\dots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\dots f_2(w_1 x + b_1|w_2)\dots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\dots w_2(w_1 x + b_1) + b_2)\dots w_{L-1})|\ w_L)$$

Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - x : 입력값
  - w_l : l 번째 층의 가중치
  - $a_l = f_l(x|w_l) = w_l x + b_l$: l 층의 결과값

$$a_L(x|w_{1,\ldots,L}) = f_L(f_{L-1}(\ldots f_1(x|w_1)|\ldots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\ldots f_2(w_1 x + b_1|w_2) \ldots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\ldots w_2(w_1 x + b_1) + b_2) \ldots w_{L-1})|\ w_L)$$
$$= \cdots$$
$$= w_L w_{L-1} \ldots w_1 x + (w_L w_{L-1} \ldots w_2 b_1 + w_L w_{L-1} \ldots w_3 b_2 + \cdots)$$

Neural Network

- **선형 신경망(Neural Network with Linear Module)**
  - x : 입력값
  - w_l : l 번째 층의 가중치
  - $a_l = f_l(x|w_l) = w_l x + b_l$: l 층의 결과값

$$\therefore\ f_L(f_{L-1}(\ldots f_1(x|w_1)|\ldots w_{L-1})|\ w_L) = wx + b$$

  - 따라서, 선형 신경망을 여러 번 쌓는 것은 의미 없음

2. Activation Function

Activation Function

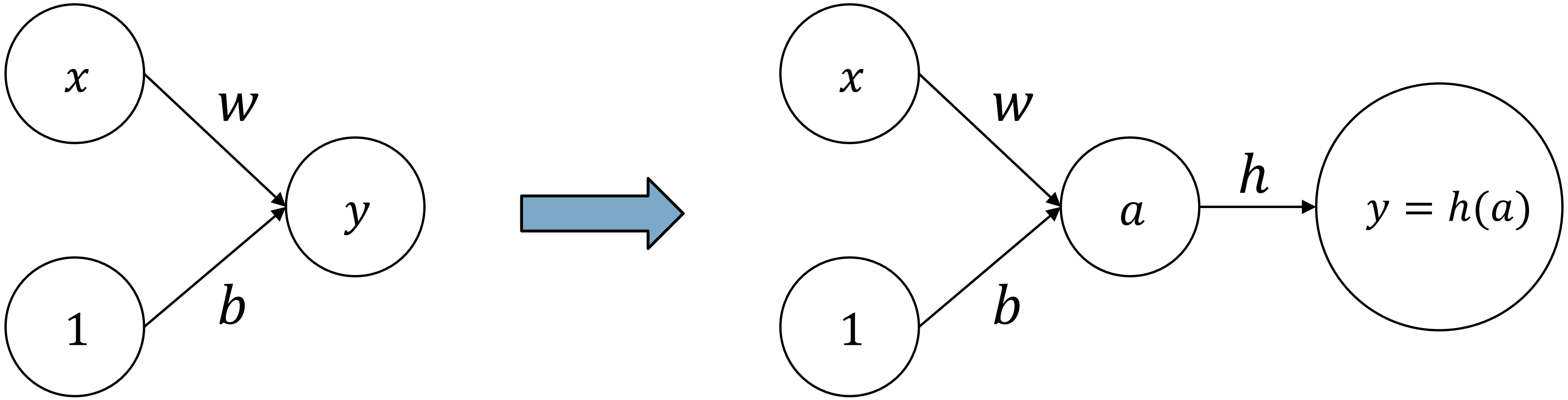
- **활성화 함수(Activation Function)**

$$a_L(x|w_{1,\dots,L}) = f_L(f_{L-1}(\dots f_1(x|w_1)|\dots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\dots f_2(w_1x + b_1|w_2)\dots w_{L-1})|\ w_L)$$
$$= f_L(f_{L-1}(\dots w_2(w_1x + b_1) + b_2)\dots w_{L-1})|\ w_L)$$
$$= \cdots$$
$$= wx + b\ (\dots?)$$

- f가 선형이면 여러 번 쌓는 것이 의미 없음
- **선형이 아닌 다른 무언가**가 필요함

Activation Function

- **활성화 함수(Activation Function)**
  - 선형이 아닌 다른 무언가 = 활성화 함수



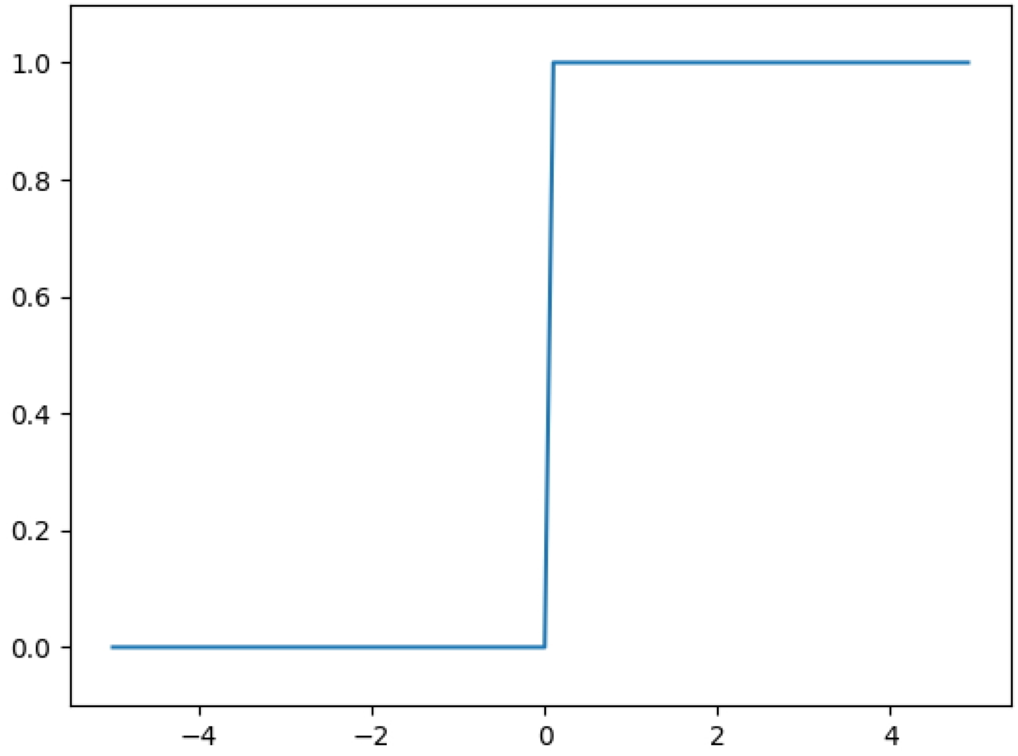
fast campus

Activation Function

- Neural Network
- Activation Function
- MNIST
- Batch Training

- 활성화 함수(Activation Function) – 계단 함수(Step Function)

  - $$h(a) = \begin{cases} 1 & (a > 0) \\ 0 & (a \leq 0) \end{cases}$$

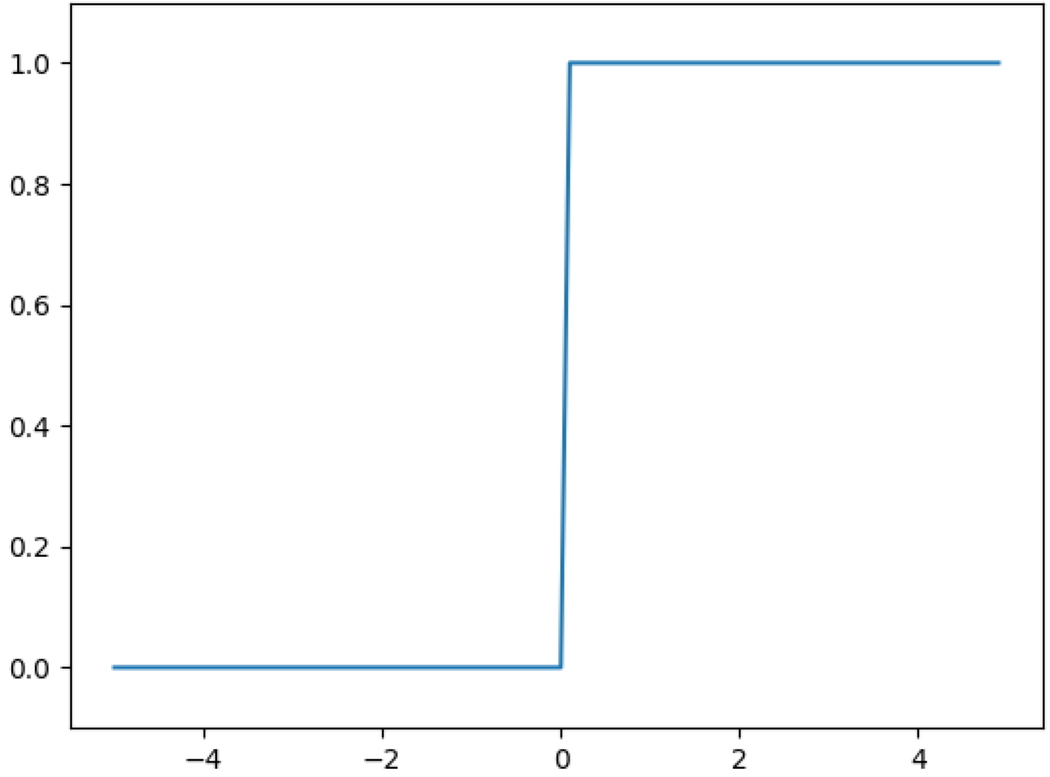


Activation Function

- Neural Network
- Activation Function
- MNIST
- Batch Training

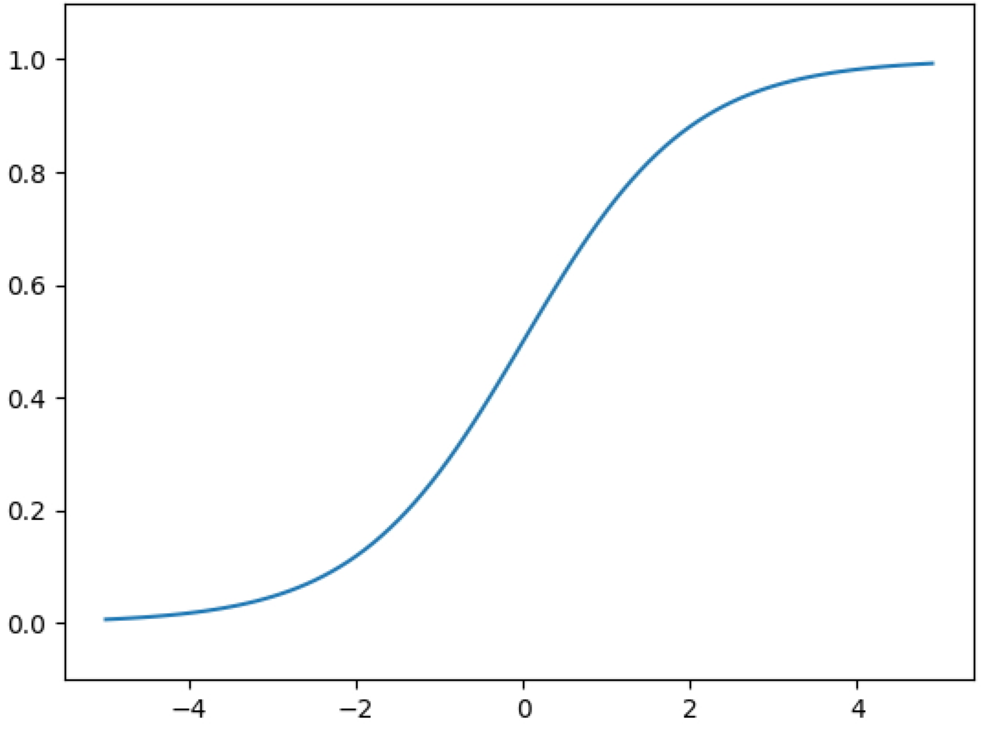
▪ 활성화 함수(Activation Function) – 계단 함수(Step Function)

- $$h(a) = \begin{cases} 1 & (a > 0) \\ 0 & (a \leq 0) \end{cases}$$
- 너무 극단적인 변화
- 기울기가 무한 or 0
- 0에서 미분 불가능



Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - 로지스틱 함수(Logistic Function) or 시그모이드 함수(Sigmoid Function)
  - $h(a) = \frac{1}{1+\exp(-a)}$



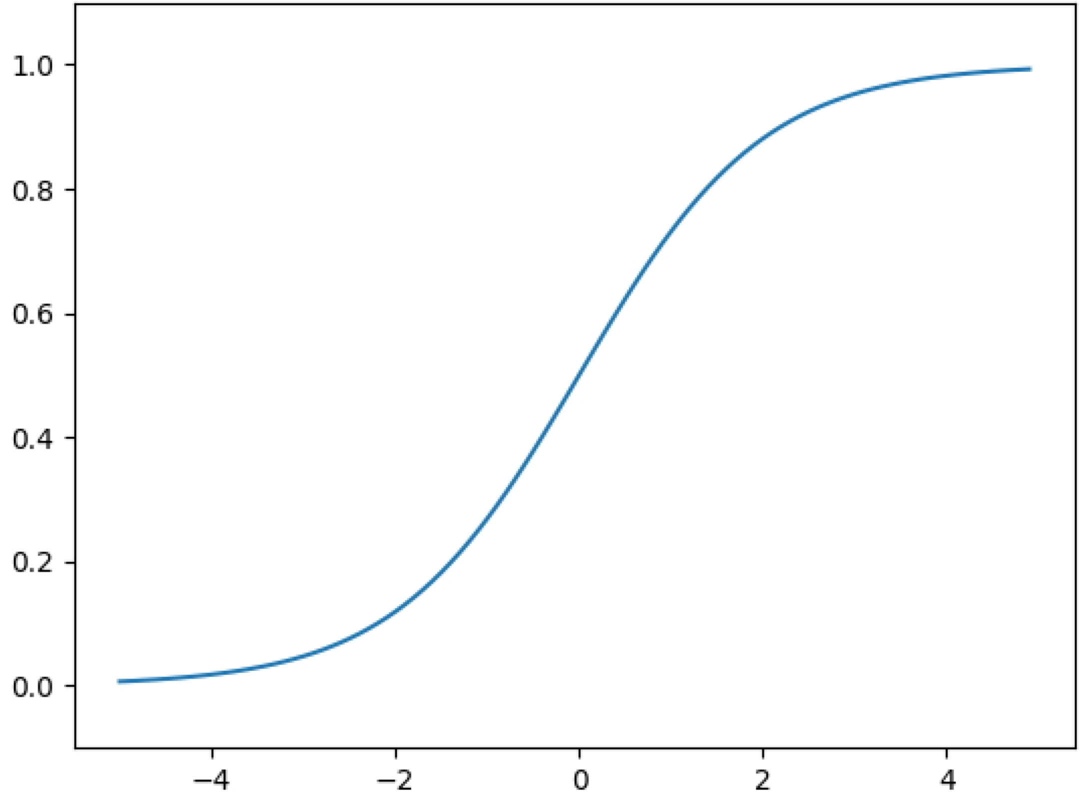
Fast campus

Activation Function

- Neural Network
- Activation Function
- MNIST
- Batch Training

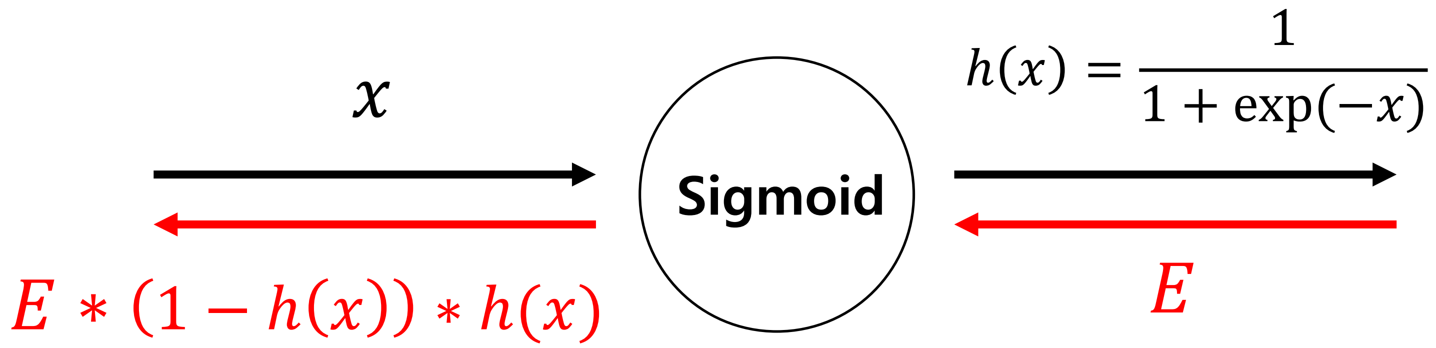
▪ 활성화 함수(Activation Function) – 시그모이드(Sigmoid)

- $h(a) = \frac{1}{1+\exp(-a)}$
- 계단 함수보다 더 부드러워진 변화
- 미분 가능
- 하지만, 기존 계단 함수와 똑같이 왼쪽/오른쪽으로 갈수록 기울기가 0에 가까워짐



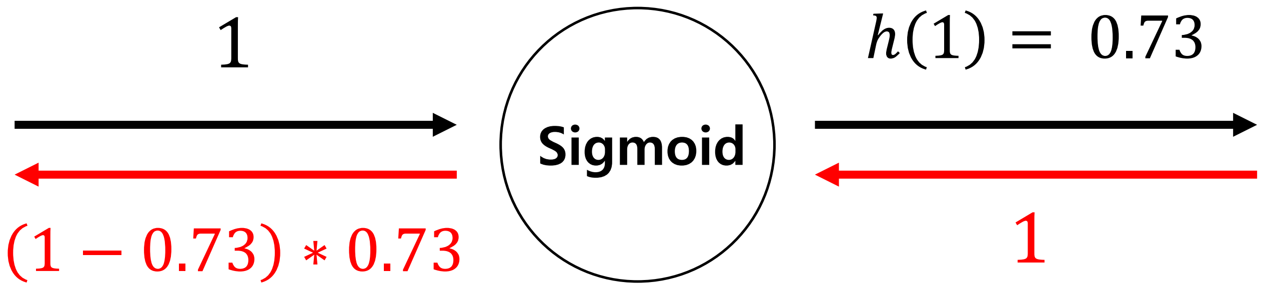
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $$\frac{\partial h(a)}{\partial a} = \big(1 - h(a)\big) * h(a)$$
  - 어느 점에서나 **기울기가 1보다 작음**



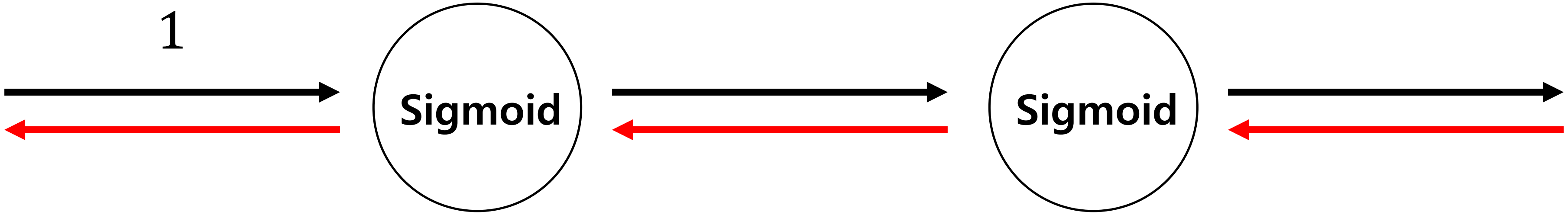
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\dfrac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**



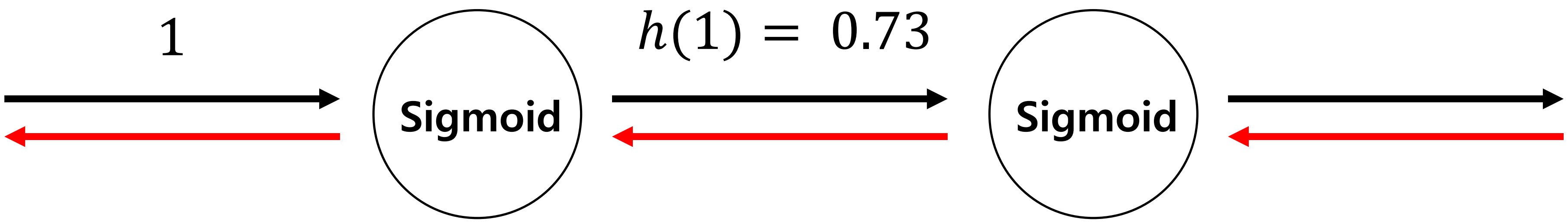
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**



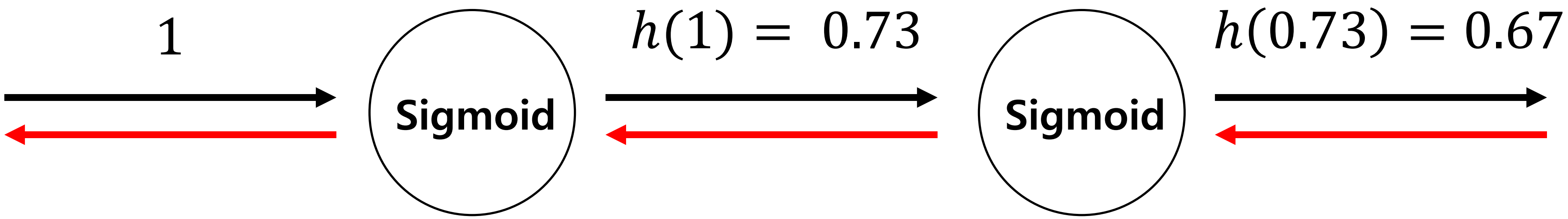
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $$\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$$
  - 어느 점에서나 **기울기가 1보다 작음**



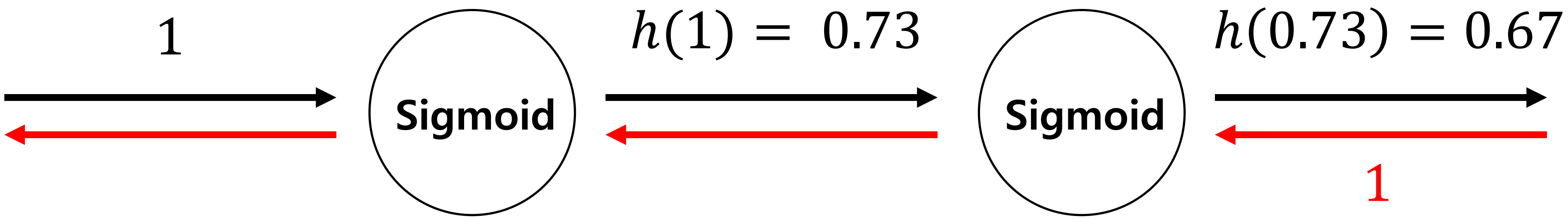
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\frac{\partial h(a)}{\partial a} = \big(1 - h(a)\big) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**



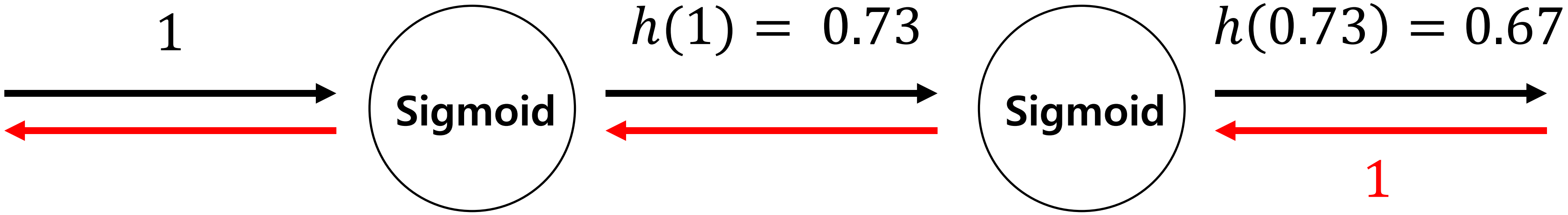
Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**



Activation Function

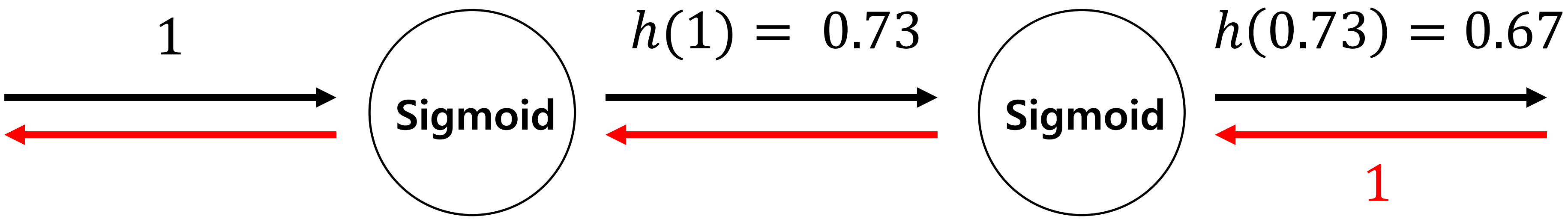
- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**



$$1 * (1 - 0.67) * 0.67$$
$$= 0.22$$

Activation Function

- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$
  - 어느 점에서나 **기울기가 1보다 작음**

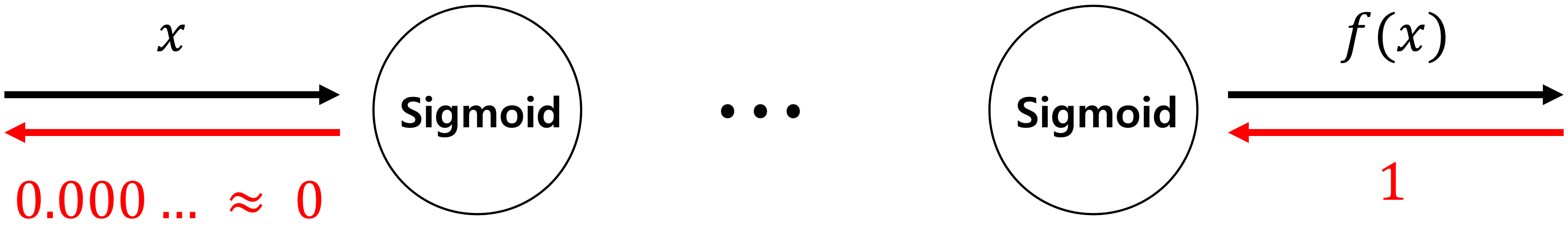


$0.22 * (1 - 0.73) * 0.73 = 0.043$

$1 * (1 - 0.67) * 0.67 = 0.22$

Activation Function

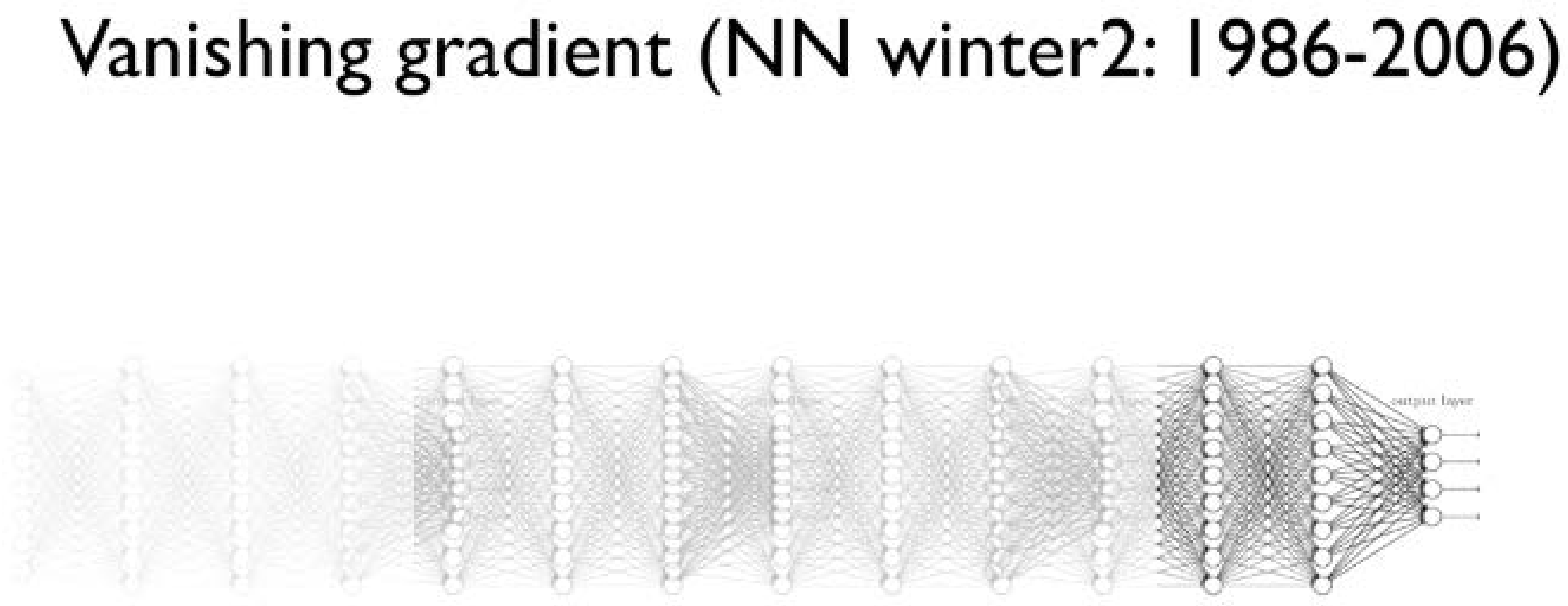
- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - $$\frac{\partial h(a)}{\partial a} = \left(1 - h(a)\right) * h(a)$$
  - 어느 점에서나 **기울기가 1보다 작음**



Activation Function

Neural Network
Activation Function
MNIST
Batch Training

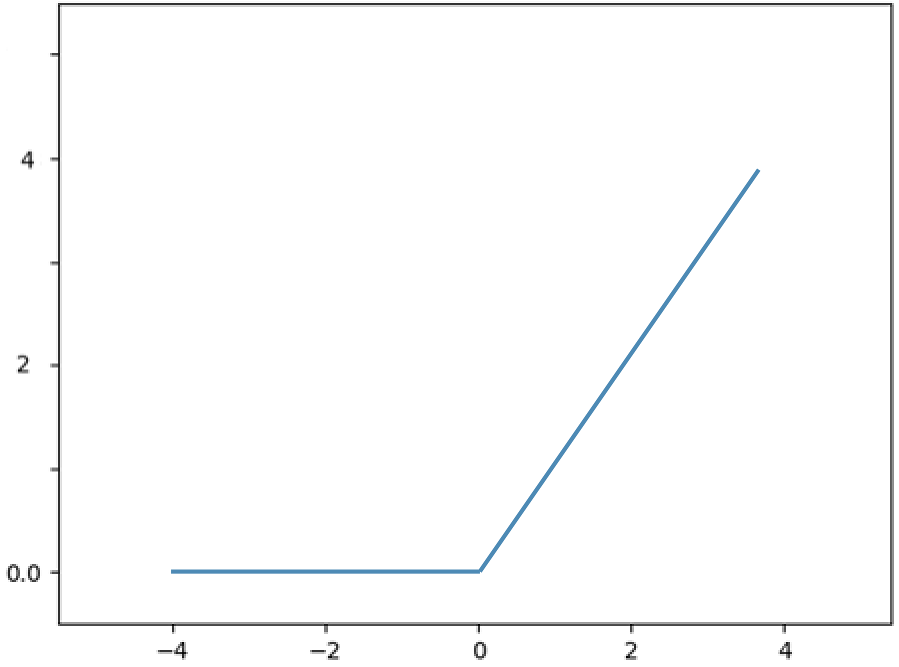
- **활성화 함수(Activation Function) – 시그모이드(Sigmoid)**
  - 어느 점에서나 기울기가 1보다 작음
    → 여러 층을 쌓으면 기울기 ~ 0 $(0.01 * 0.01 = 0.0001)$
    → 기울기 소실(Vanishing Gradient) 문제로 이어짐



Fast campus

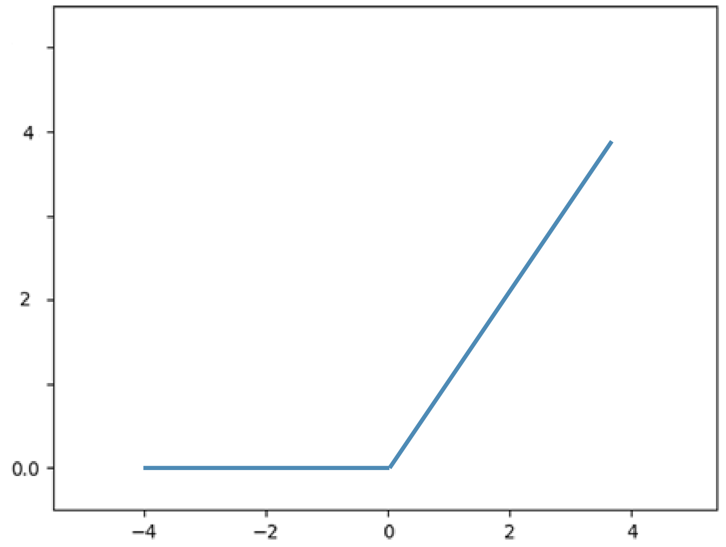
Activation Function

- **활성화 함수(Activation Function) – ReLU Function**
  - AlexNet에서 사용
  - $$h(a) = \begin{cases} a & (a > 0) \\ 0 & (a \leq 0) \end{cases}$$



Activation Function

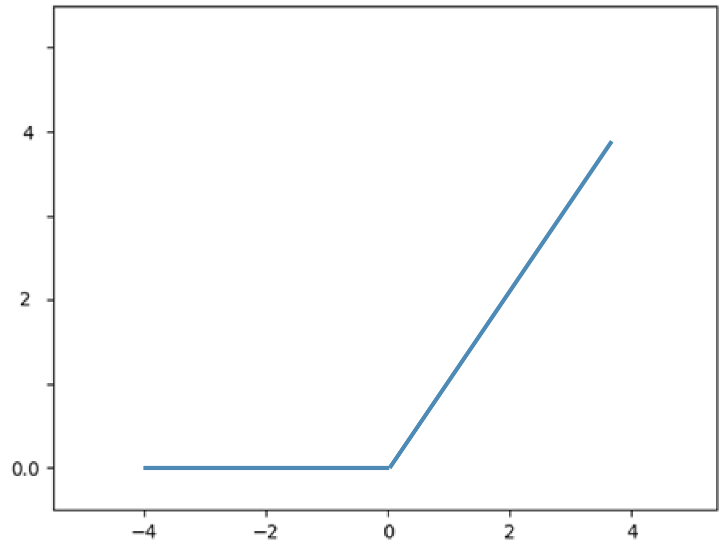
- **활성화 함수(Activation Function) – ReLU Function**
  - 함수의 기울기가 0 or 1
  - 그대로 값을 내보냄
    - 계산 및 학습이 빠름
    - 기울기 소실 문제 해결
  - 하지만
    - 대칭적이지 않음
    - **0에서 미분 불가능**
    - 0보다 작은 값들은 '죽을' 수 있다



Fast campus

Activation Function

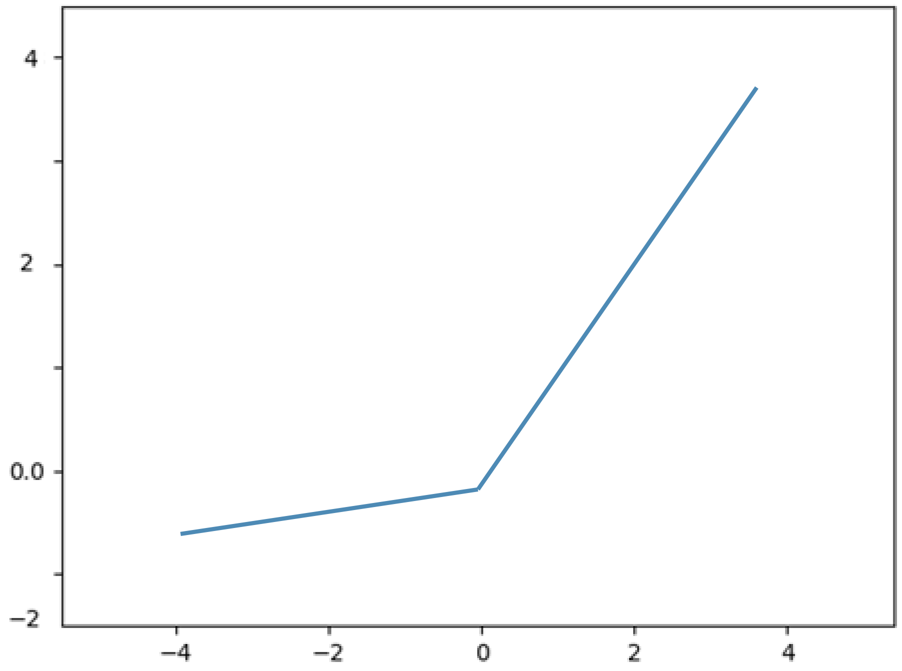
- **활성화 함수(Activation Function) – ReLU Function**
  - 함수의 기울기가 0 or 1
  - 그대로 값을 내보냄
    - 계산 및 학습이 빠름
    - 기울기 소실 문제 해결
  - 하지만
    - 대칭적이지 않음
    - **0에서 미분 불가능**
      - **sub-gradient descent로 해결 가능**
      - x = 0, it has *subdifferential* [0,1]
    - 0보다 작은 값들은 '죽을' 수 있다



Activation Function

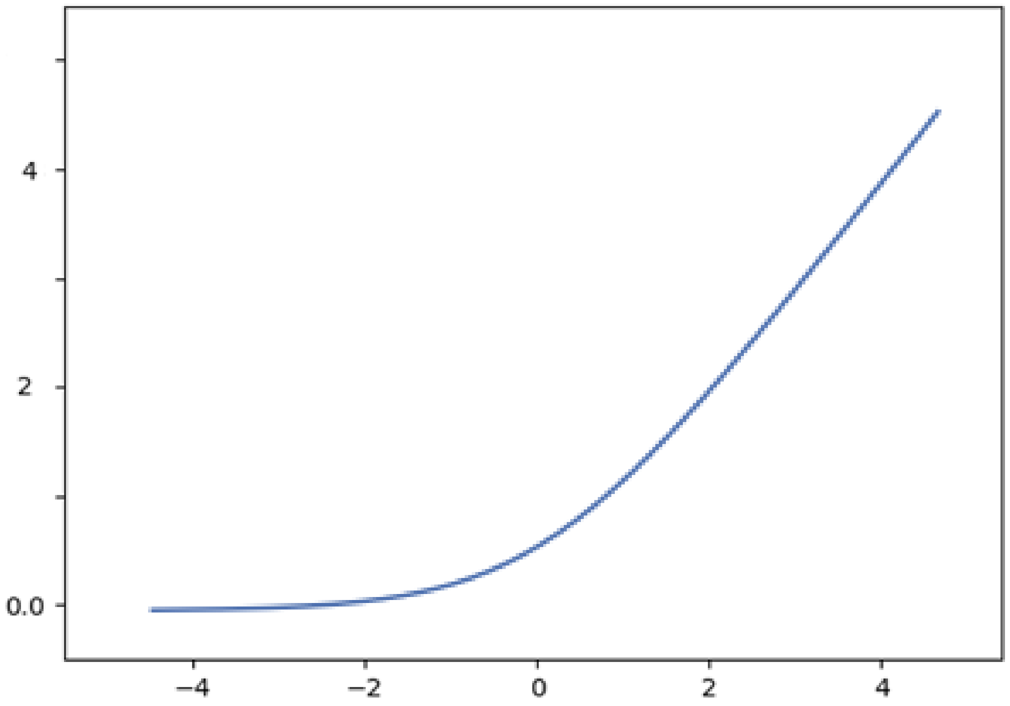
▪ 활성화 함수(Activation Function) – Leaky ReLU Function

▪ $$h(a) = \begin{cases} a & (a > 0) \\ 0.1 * a & (a \leq 0) \end{cases}$$



Activation Function

- **활성화 함수(Activation Function) – Softplus Function**
  - $h(a) = \log(1 + \exp(a))$



Fast campus

Activation Function

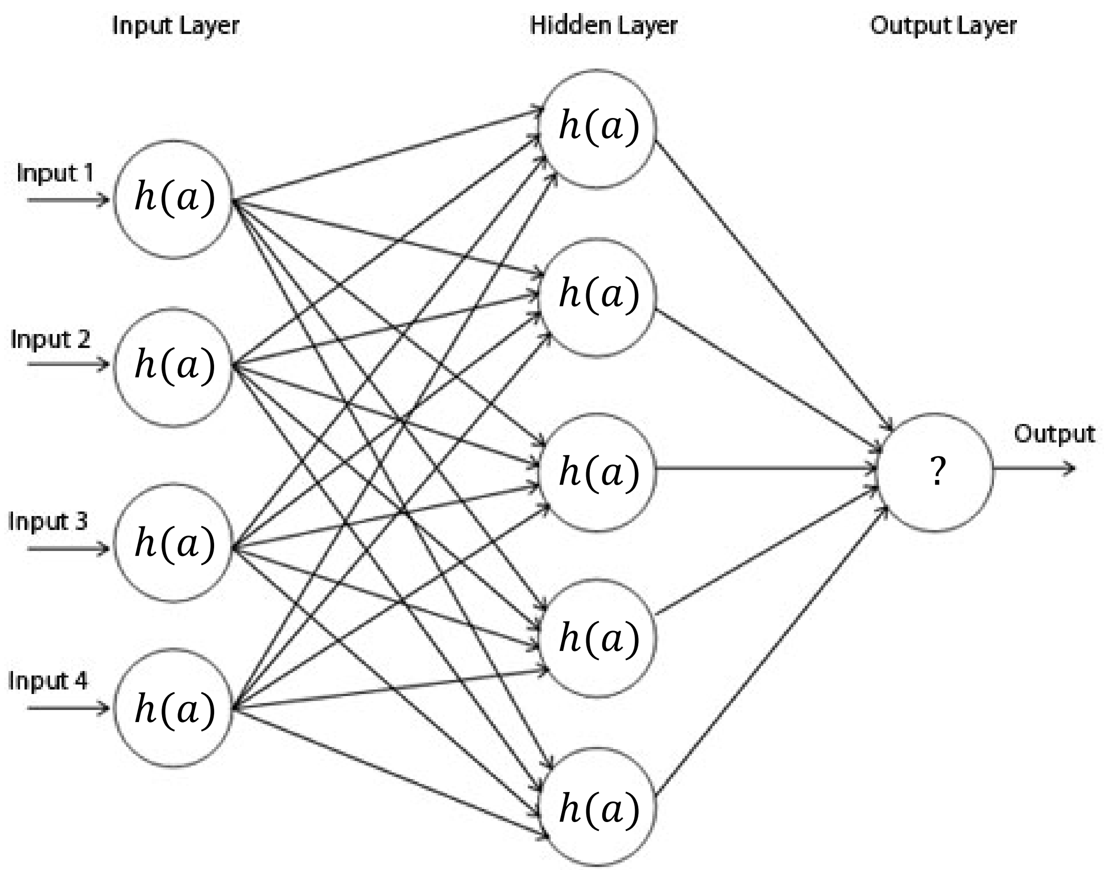
Neural Network

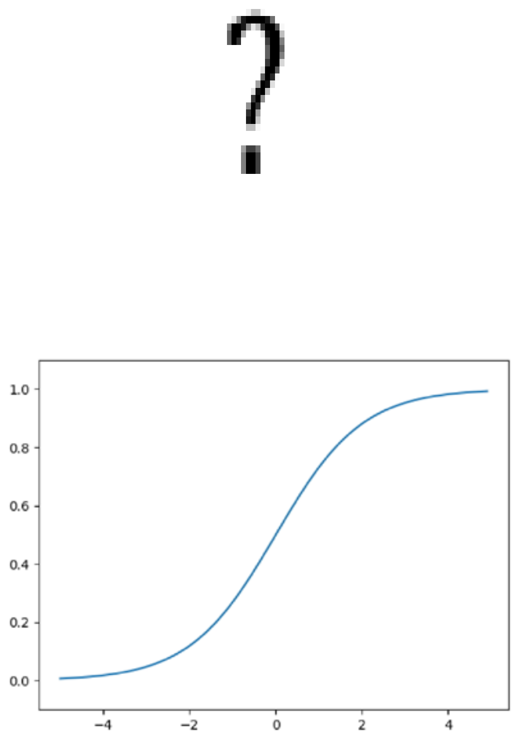
Activation Function

MNIST

Batch Training

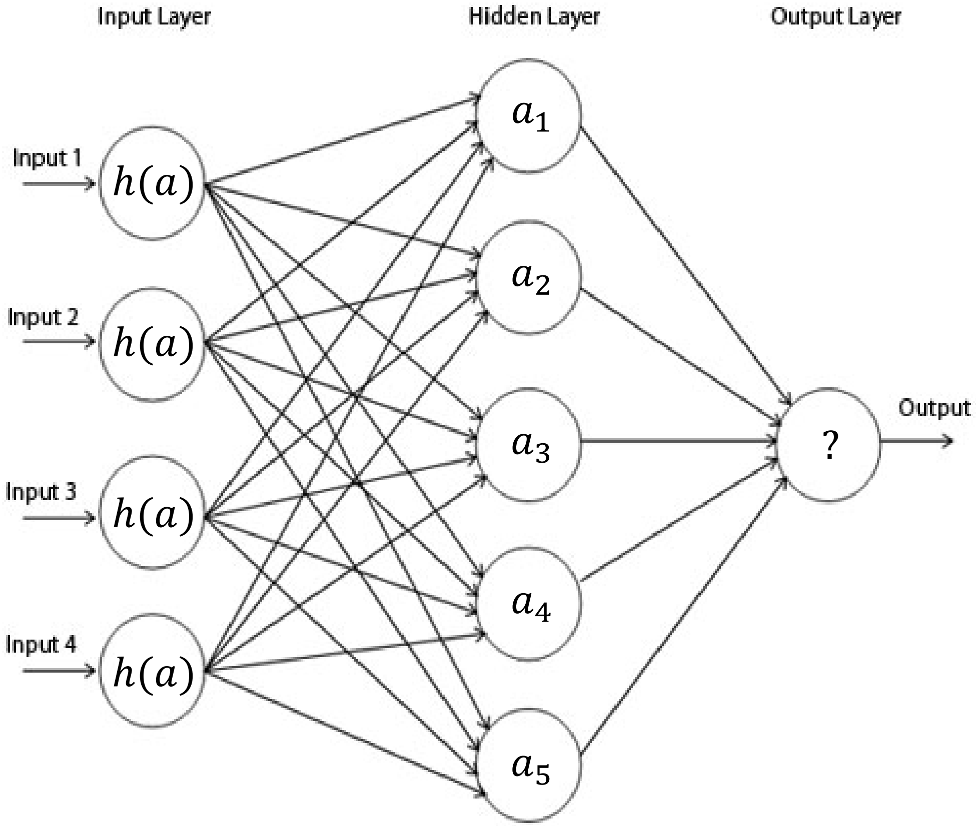
- 활성화 함수(Activation Function) - 출력층





Activation Function

- **활성화 함수(Activation Function) – 출력층**
  - 회귀 - 항등 함수(Identity Function)
    - $y_k = a_k$
  - 분류 – 소프트맥스 함수(Softmax Function)
    - $y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$
    - $y_k = \frac{\exp(a_k + C)}{\sum_{i=1}^{n} \exp(a_i + C)}$



Fast campus

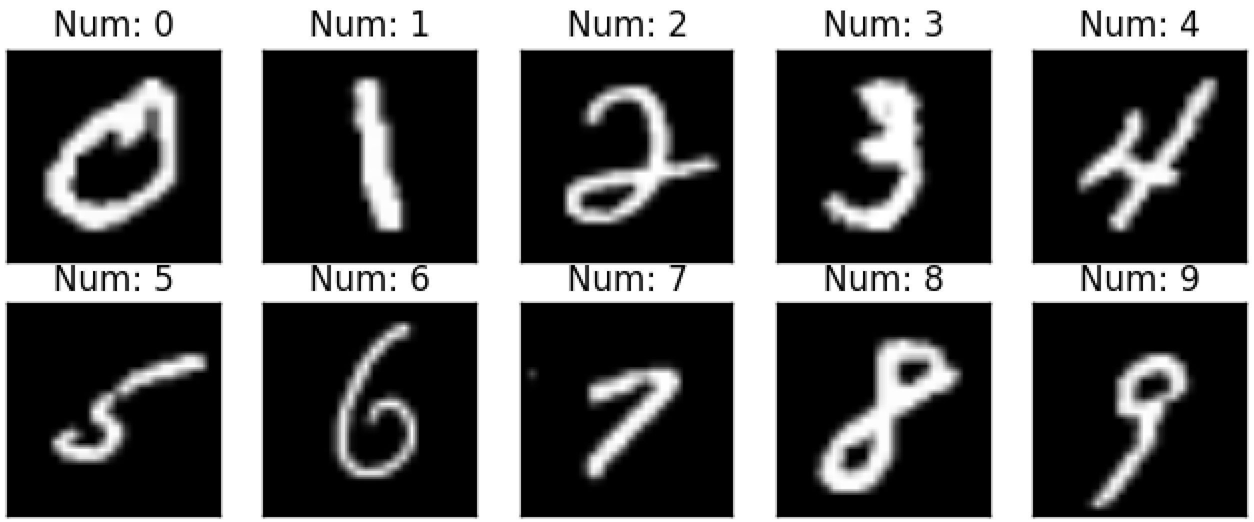
Activation Function

- **활성화 함수(Activation Function) – 출력층**
  - 회귀 - 항등 함수(Identity Function)
    - $y_k = a_k$
    - $[0.3, 2.9, 4.0] \rightarrow [0.3, 2.9, 4.0]$
  - 분류 – 소프트맥스 함수(Softmax Function)
    - $y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$
    - $[0.3, 2.9, 4.0] \rightarrow [0.018, 0.245, 0.737]$
    - $0.018 + 0.245 + 0.737 = 1$

3. MNIST

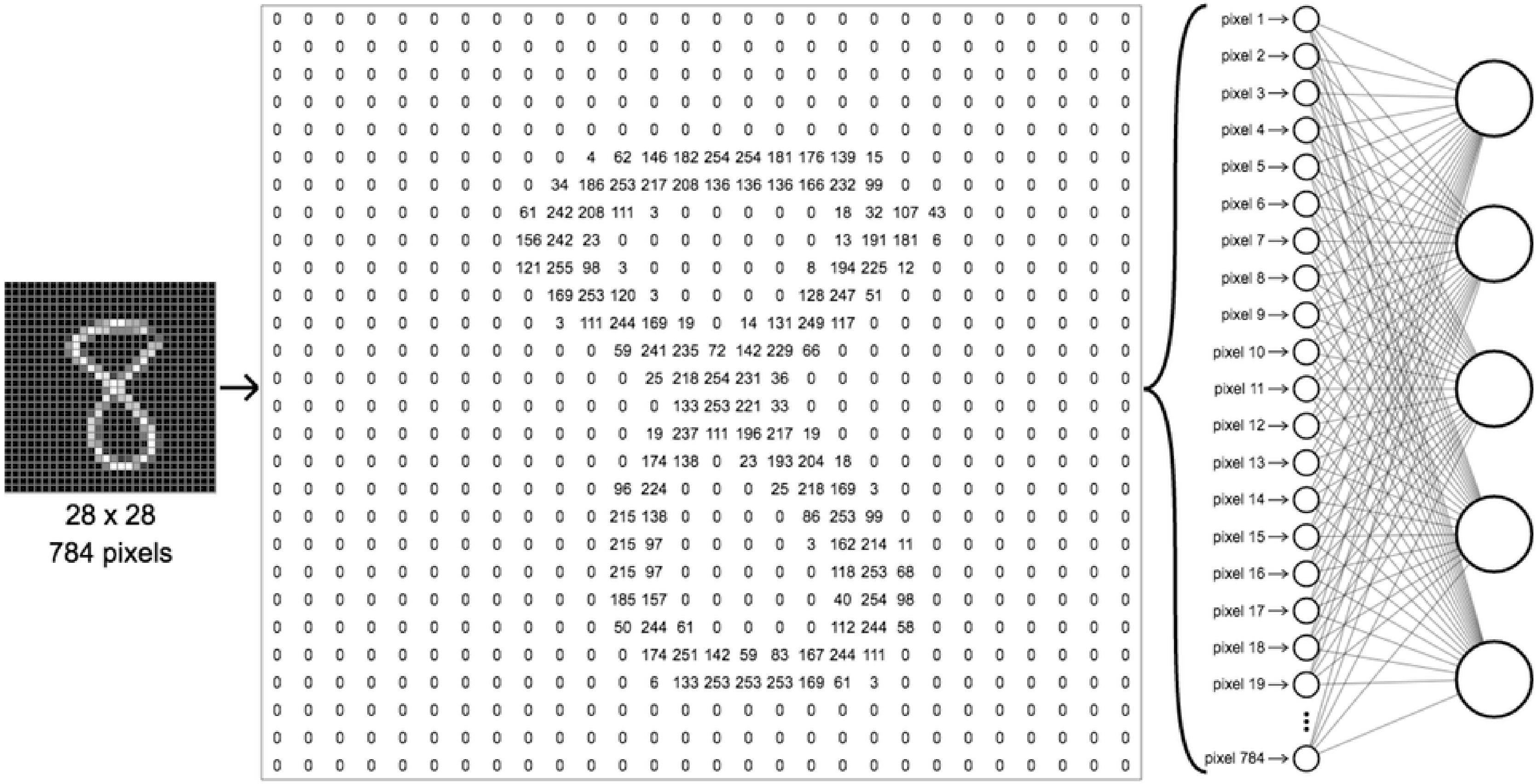
MNIST

- **MNIST : 손글씨 숫자 이미지 데이터**
  - 60000장의 Training Set
  - 10000장의 Test Set
  - Size : 28 X 28
  - Color : Gray



MNIST

- **MNIST를 위한 Neural Network**
  - Size : 28 X 28 = 784



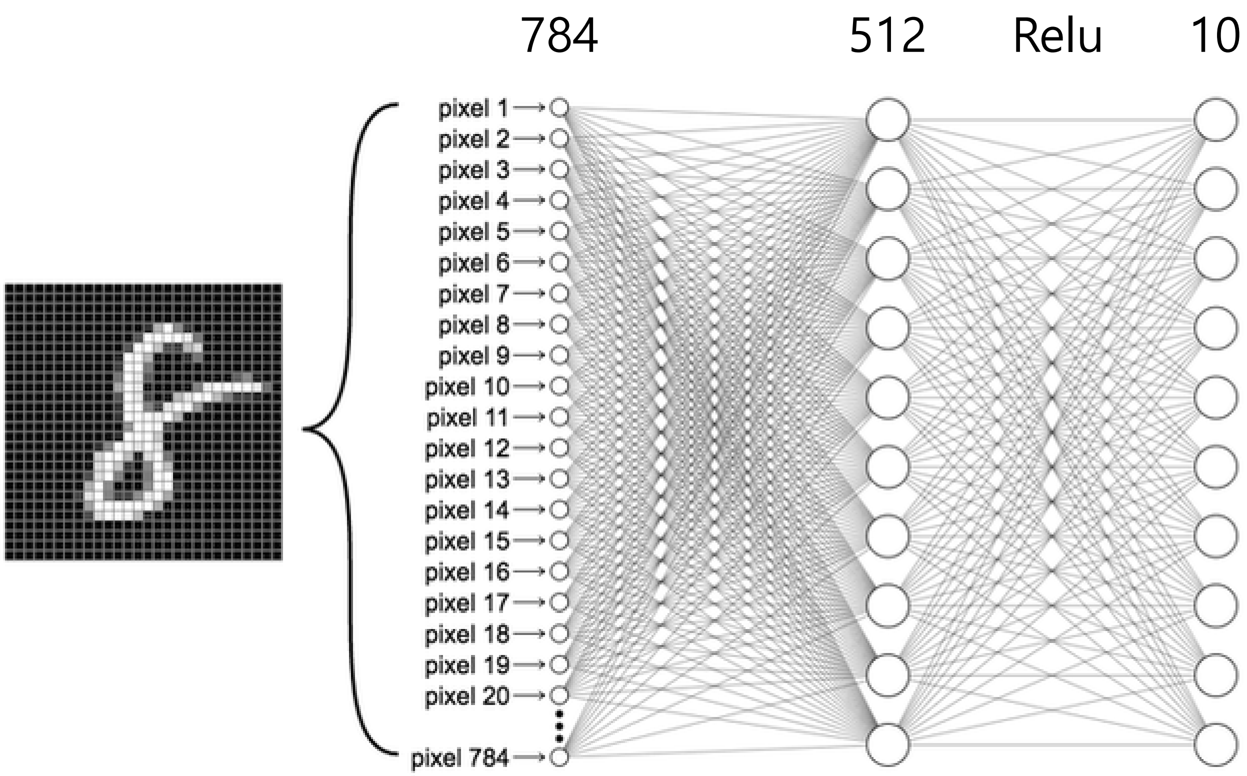
http://lucenaresearch.com/deep-neural-networks/

Fast campus

MNIST

▪ MNIST를 위한 Neural Network

- Size : 28 X 28 = 784
- Output : 10개의 값 (확률) = 이미지가 [0일 확률, 1일 확률, ..., 9일 확률]



https://achintavarna.wordpress.com/2017/11/17/keras-tutorial-for-beginners-a-simple-neural-network-to-identify-numbers-mnist-data/

MNIST

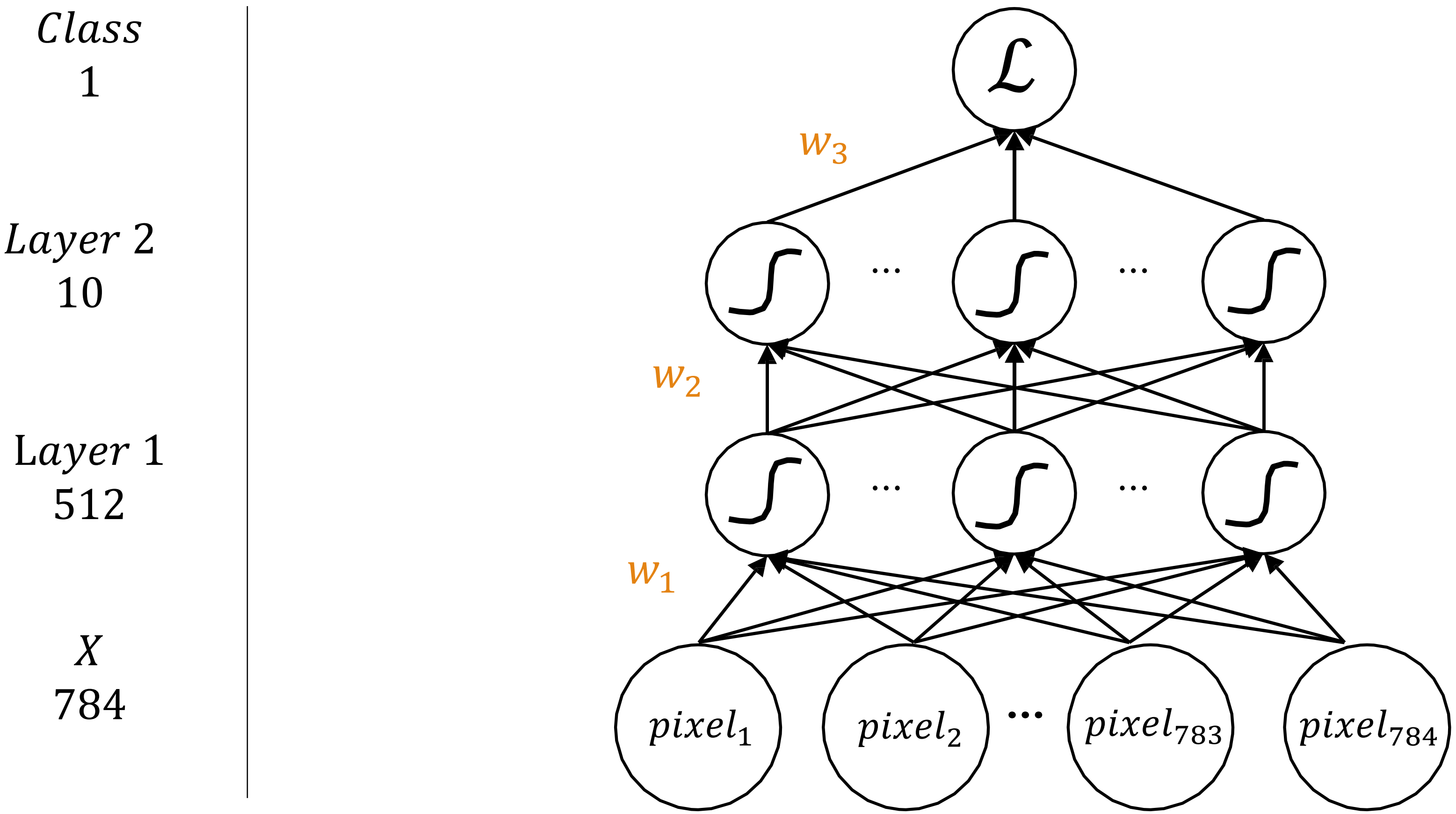
Neural Network
Activation Function
MNIST
Batch Training

- **MNIST를 위한 Neural Network**
  - Size : 28 X 28 = 784
  - Output : 10개의 값 (확률) = 이미지가 [0일 확률, 1일 확률, ..., 9일 확률]

- **각 층의 배열 형상**
  - $X \rightarrow W1 \rightarrow W2 \rightarrow Y$
  - $784 \rightarrow 784 \times 512 \rightarrow 512 \times 10 \rightarrow 10$

MNIST

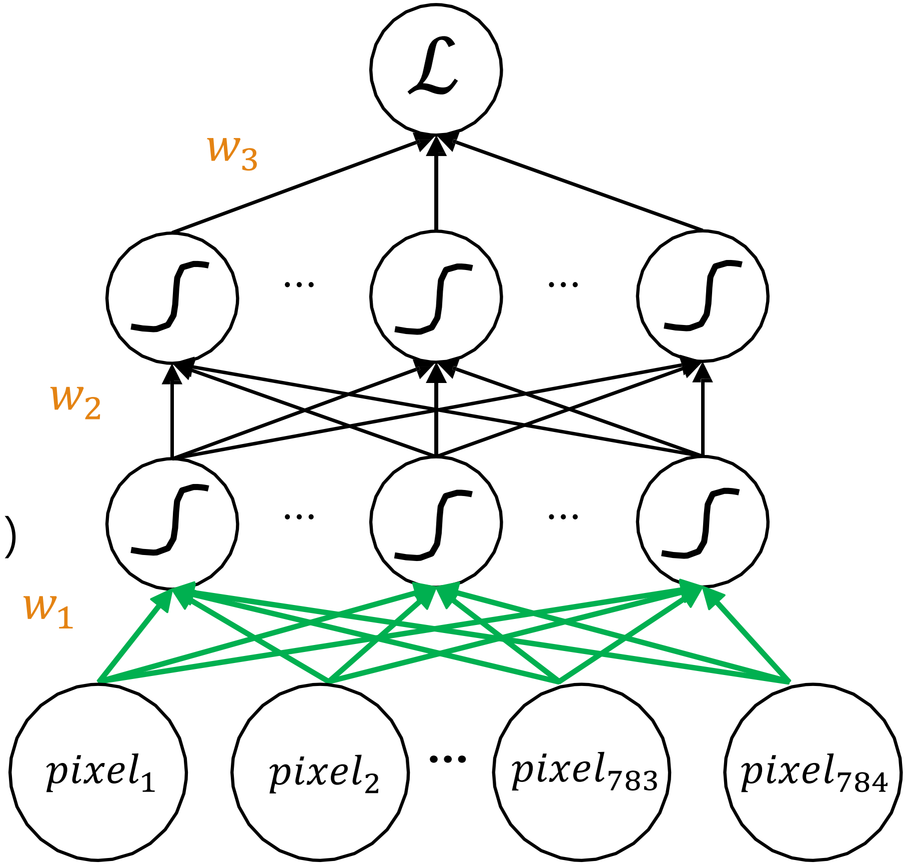
- **MNIST를 위한 Neural Network**
  - 학습 방법



MNIST

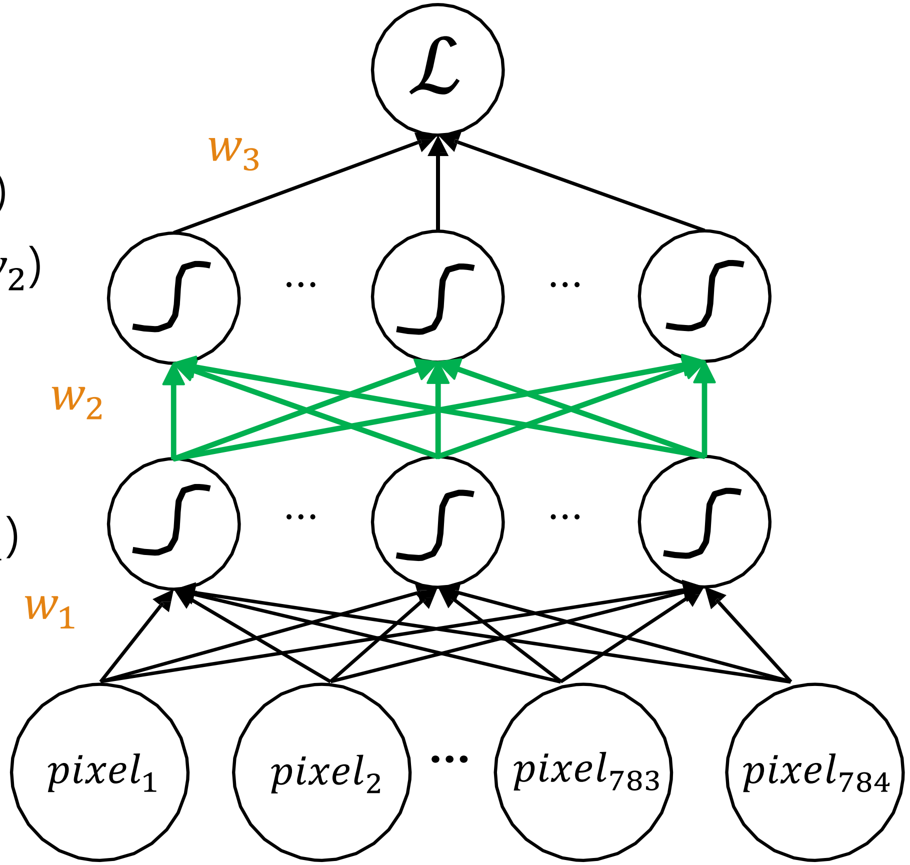
- **MNIST를 위한 Neural Network**
  - 학습 방법

$Class$ 1	
$Layer\ 2$ 10	
$Layer\ 1$ 512	$y_1 = h_1(a_1)$ $a_1 = f_1(x_1 \mid w_1)$
X 784	



- **MNIST를 위한 Neural Network**
  - 학습 방법

$Class$ 1	
$Layer\ 2$ 10	$y_2 = h_2(a_2)$ $a_2 = f_2(x_2 \vert w_2)$
$Layer\ 1$ 512	$y_1 = h_1(a_1)$ $a_1 = f_1(x_1 \vert w_1)$
X 784	



MNIST

- **MNIST를 위한 Neural Network**
  - 학습 방법

Class 1

Layer 2 10

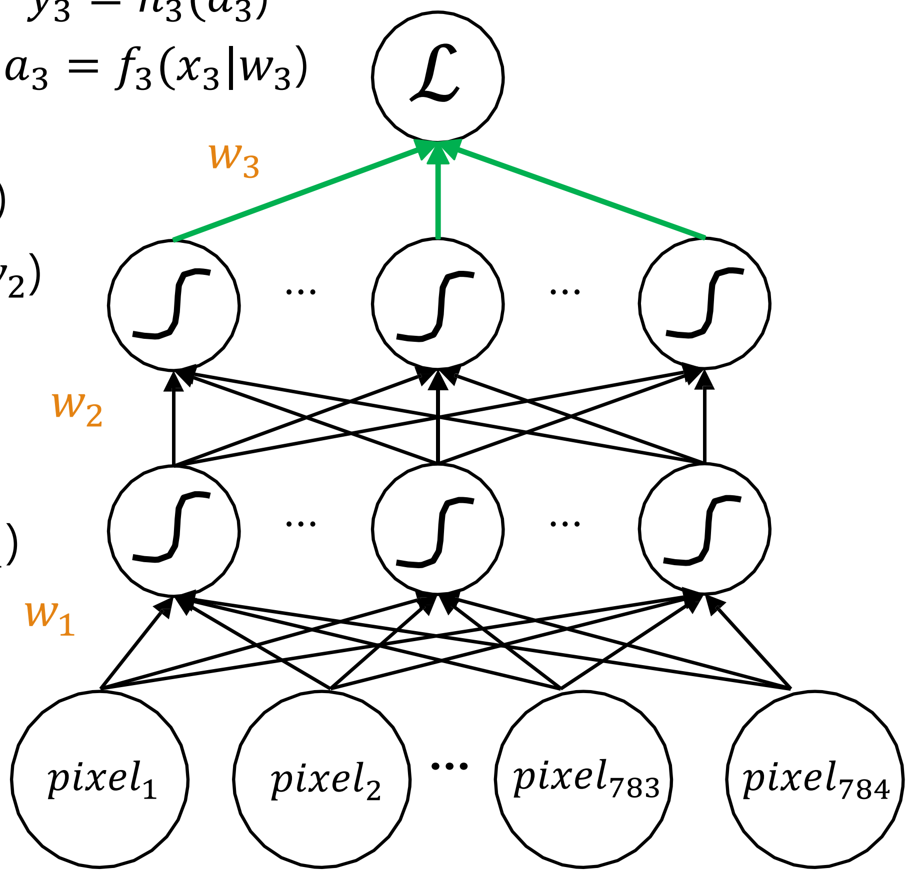
Layer 1 512

X 784

$y_3 = h_3(a_3)$
$a_3 = f_3(x_3|w_3)$

$y_2 = h_2(a_2)$
$a_2 = f_2(x_2|w_2)$

$y_1 = h_1(a_1)$
$a_1 = f_1(x_1|w_1)$



MNIST

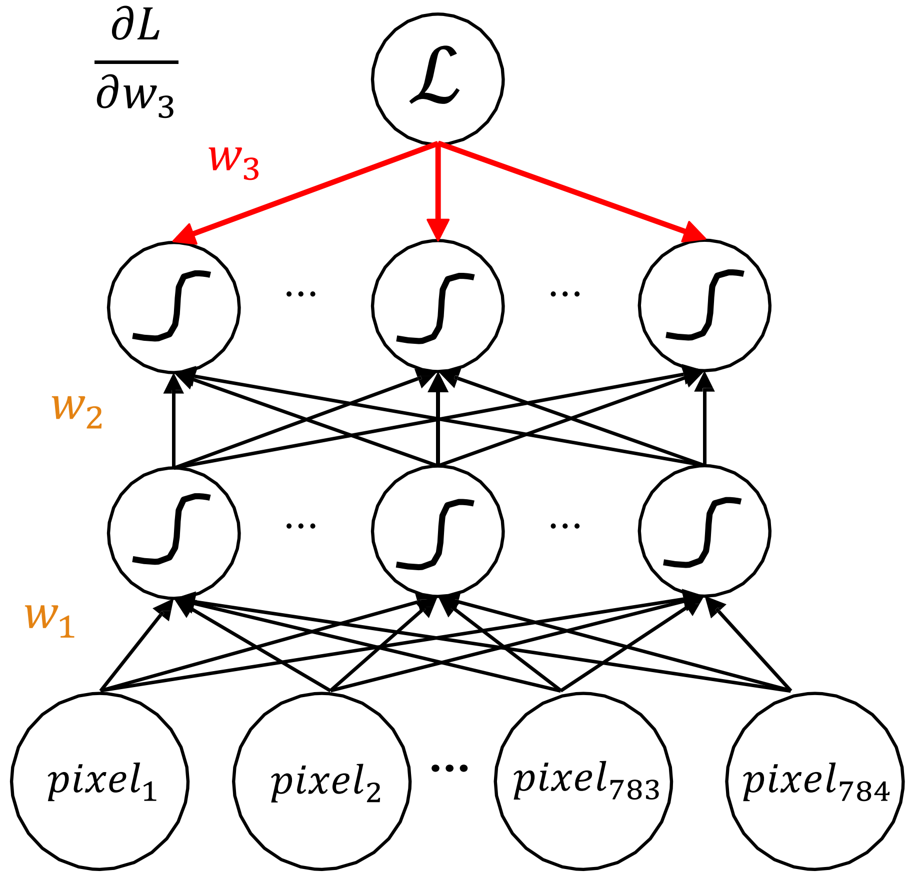
- **MNIST를 위한 Neural Network**
  - 학습 방법

$Class$ 1

$Layer\ 2$ 10

$Layer\ 1$ 512

X 784



MNIST

Neural Network
Activation Function
MNIST
Batch Training

▪ MNIST를 위한 Neural Network

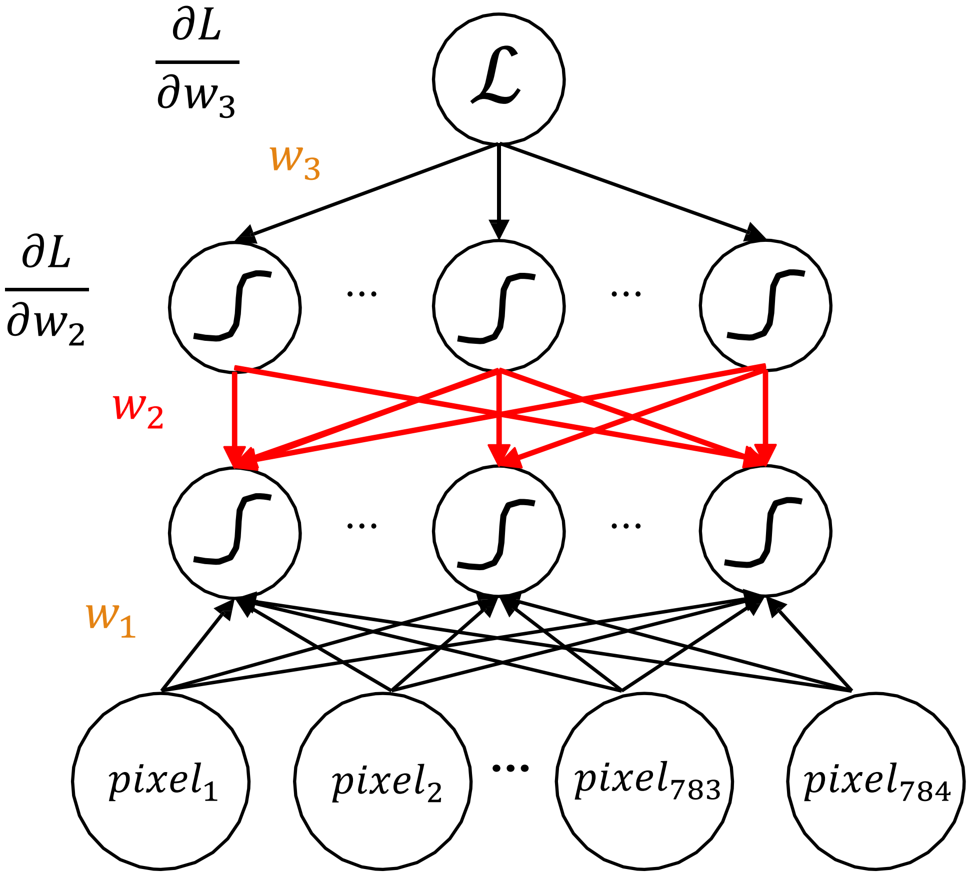
- 학습 방법

$Class$
1

$Layer\ 2$
10

$Layer\ 1$
512

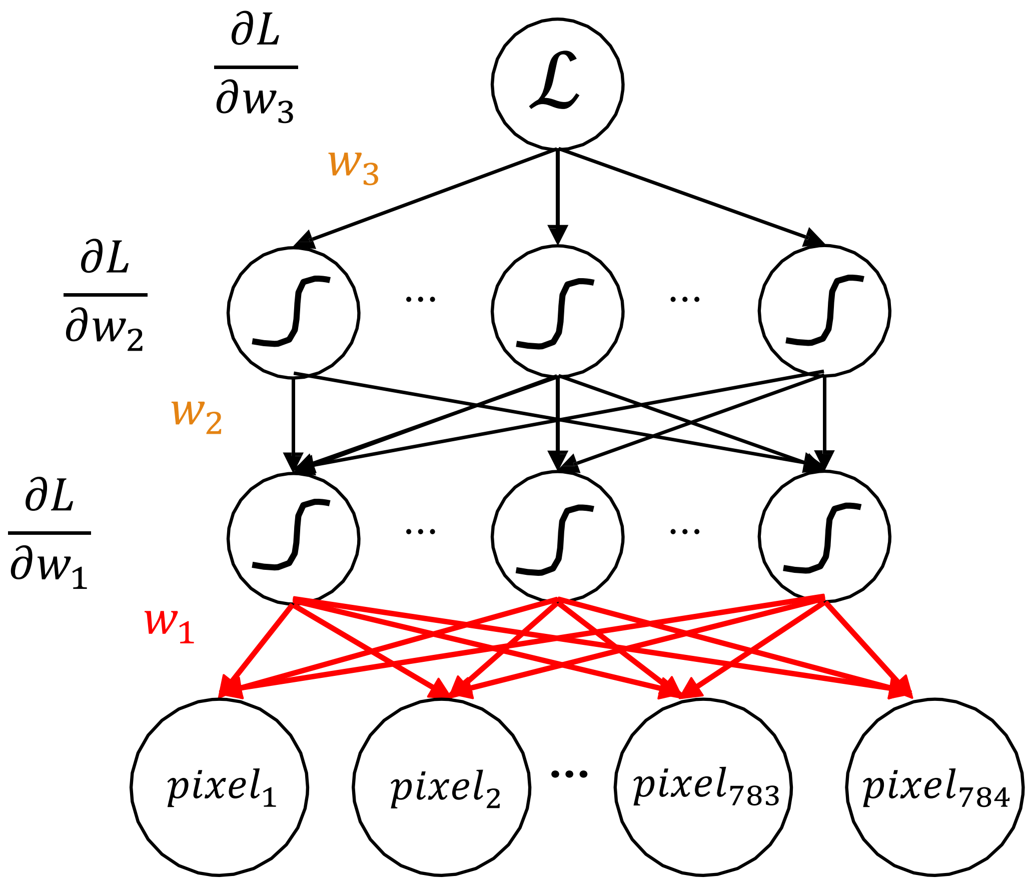
X
784



MNIST

- **MNIST를 위한 Neural Network**
  - 학습 방법

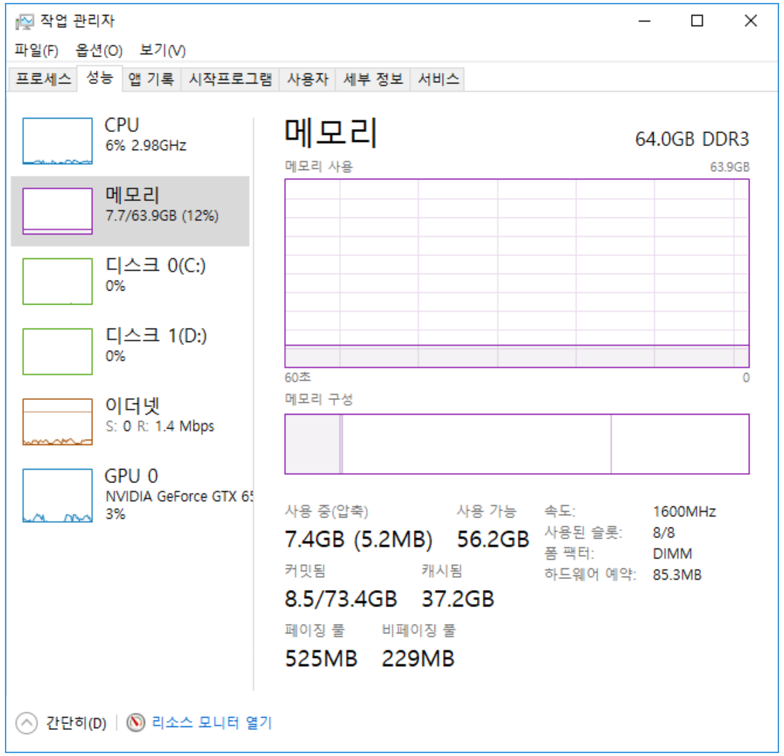
$Class$	1
$Layer\ 2$	10
$Layer\ 1$	512
X	784



4. Batch Training

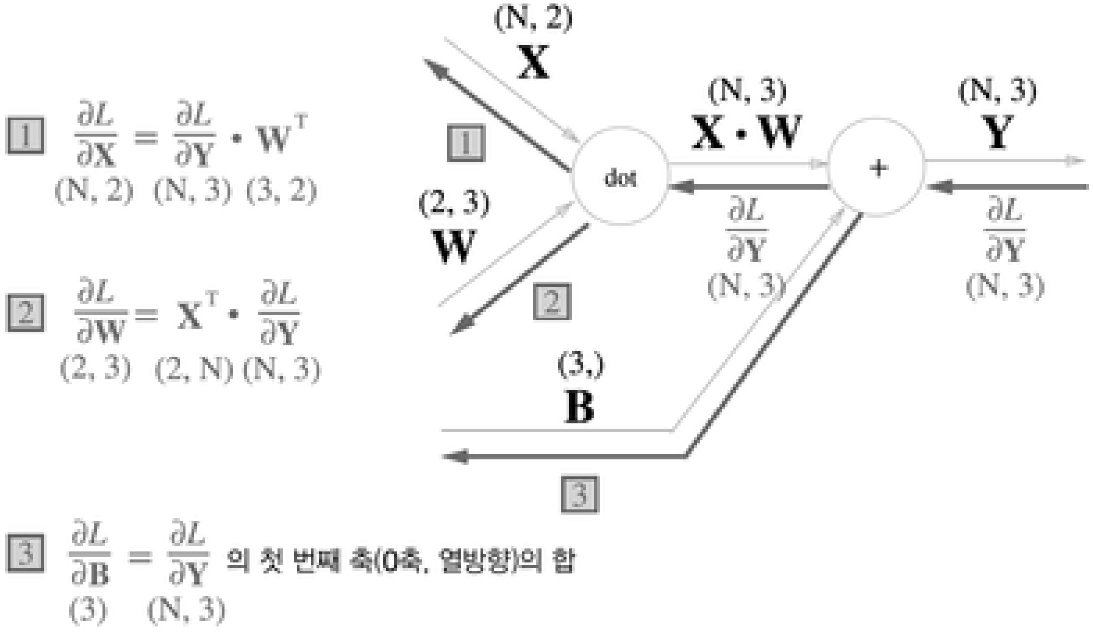
Batch Training

- **배치 학습(Batch Training)**
  - 이미지를 1개씩 학습시킬 것이냐?
    - 학습 속도 너무 느림
    - 컴퓨터는 큰 배열을 한꺼번에 계산하는 것이 빠름
  - 이미지를 모두다 학습시킬 것이냐?
    - Out of Memory Error
    - 학습을 시키기 위해서는 메모리에 담아야 하는데,
    - 메모리는 용량이 한정되어 있음



Batch Training

- **배치 학습(Batch Training)**
  - 적당한 양의 데이터들을 묶어 한 번에 학습시킴
  - Batch size = 100 :
    - $X \quad \rightarrow \quad W1 \quad \rightarrow \quad W2 \quad \rightarrow \quad Y$
    - $100 \times 784 \rightarrow 784 \times 512 \rightarrow 512 \times 10 \rightarrow 100 \times 10$

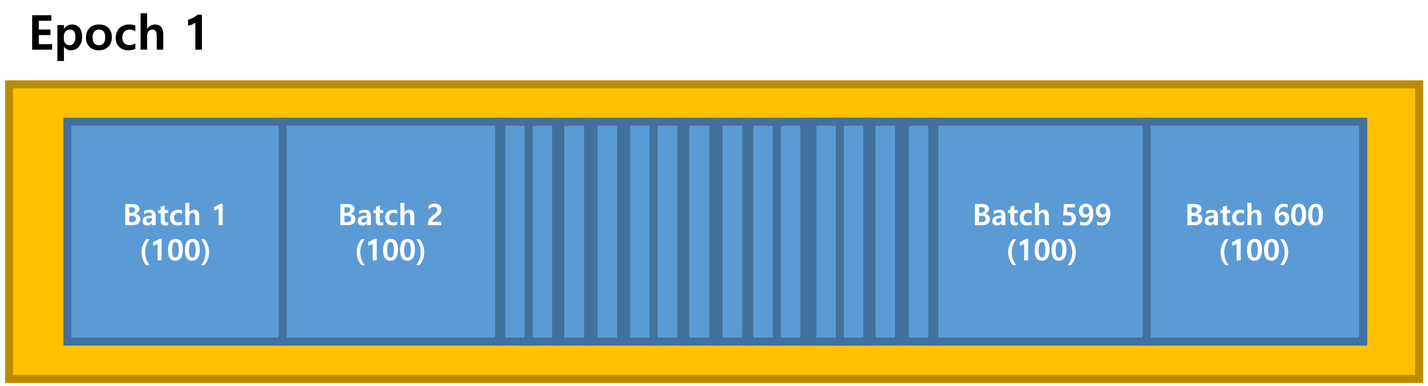


Batch Training

- **배치 학습을 위한 용어**
  - 에폭(Epoch)
    - **한 데이터**가 **총 몇 번 학습**에 사용되는가?
  - 반복(Iteration)
    - **1 에폭**에 **몇 개의 배치**를 사용해서 학습할 것인가?

Batch Training

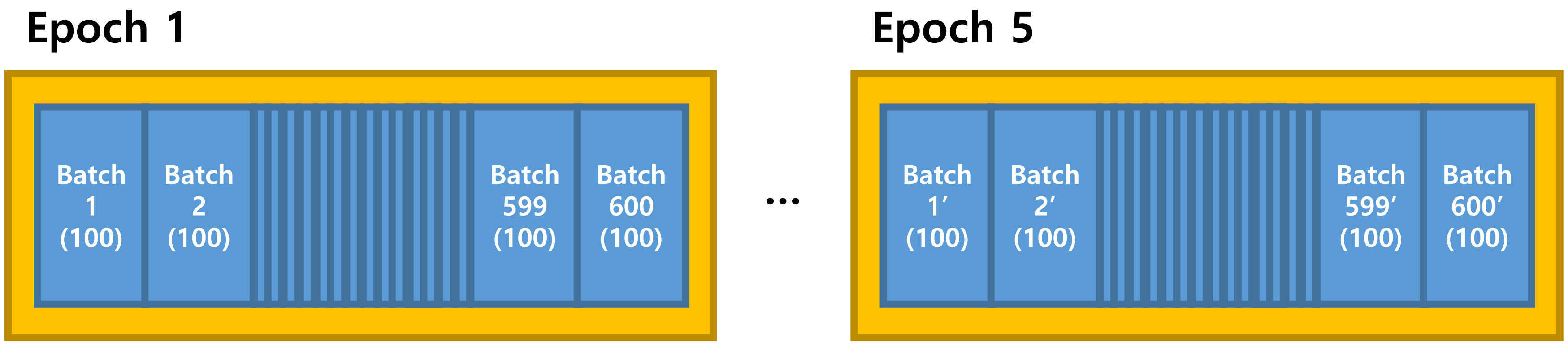
- **배치 학습을 위한 용어**
  - EX1) **총 Data 60000개**
    - **Batch size = 100** : 한 번 학습에 100개씩 학습
    - **Epoch = 1** : 모든 데이터를 1번씩 학습에 사용
    - **Iter = 60000/100 = 600** : 1 에폭은 600개의 배치로 학습



Batch Training

Neural Network
Activation Function
MNIST
Batch Training

- **배치 학습을 위한 용어**
  - EX2) **총 Data 60000개**
    - **Batch size = 100** : 한 번 학습에 100개씩 학습
    - **Epoch = 5** : 모든 데이터를 5번씩 학습에 사용
    - **Iter = 60000/100 = 600** : 1 에폭은 600개의 배치로 학습



Batch Training

- **배치 학습을 위한 용어**
  - EX3) 총 Data 60000개
    - **Batch size = 200** :
    - **Epoch = 10** :
    - **Iter = ?** :

Batch Training

- **배치 학습을 위한 용어**
  - EX3) 총 Data 60000개
    - **Batch size = 200** : 한 번 학습에 200개씩 학습
    - **Epoch = 10** : 모든 데이터를 10번씩 학습에 사용
    - **Iter = 60000/200 = 300** : 1 에폭은 300개의 배치로 학습