# 임상의를 위한 AI 교육 – 기초과정 2주차

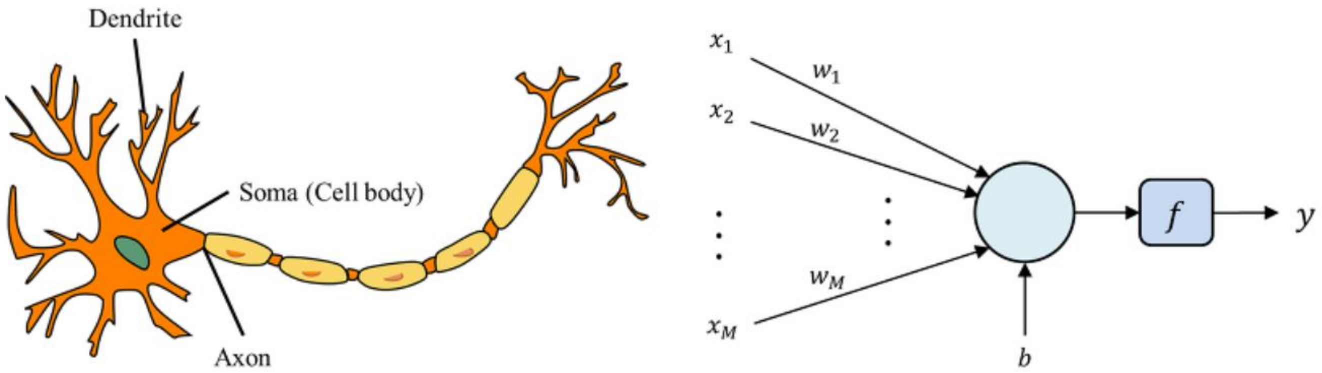
# 딥 러닝의 개념과 실습

서울대학교병원 융합의학과 김영곤 교수

※ 본 수업자료는 **"서울대학교병원 데이터사이언스연구부 AI지원실"** 학습서기반으로 제작되었습니다.

# Part 4. 딥 러닝

- **인공신경망 (Artificial Neural Network, ANN)**
  - 인간이 학습하는 방법을 본떠 만든 네트워크 구조
  - 인간의 뇌가 가지는 생물학적 특성인 뉴런의 연결 구조를 함수로 표현 → **퍼셉트론(Perceptron)**

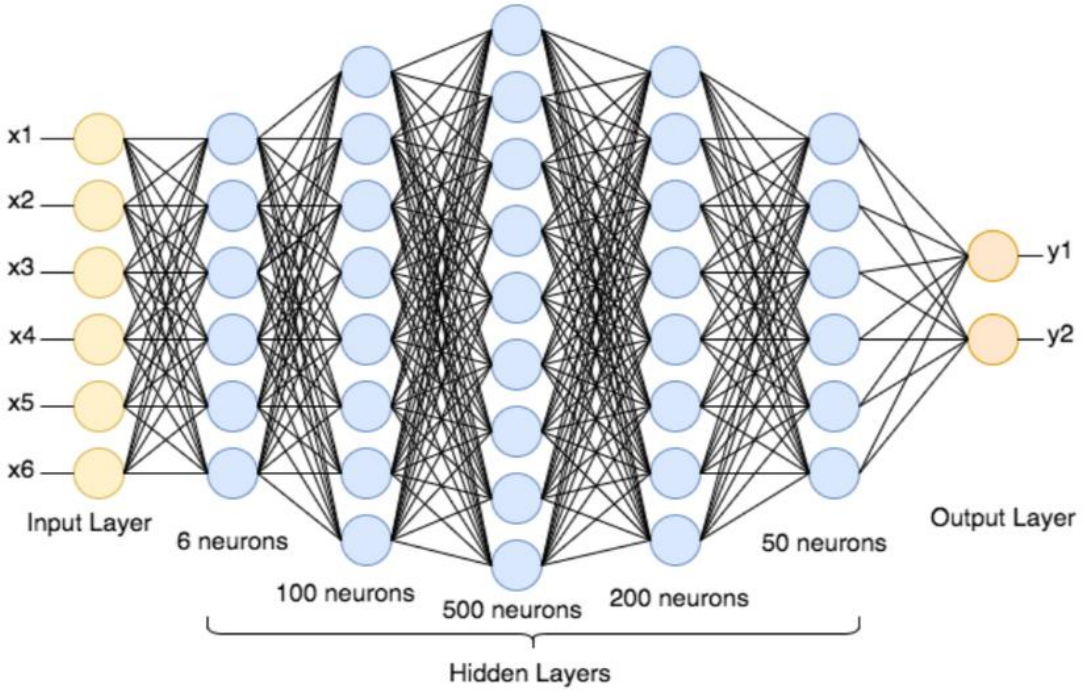



[그림 1] 생물체의 neuron (좌)과 artificial neuron (우)

# Part 4. 딥 러닝



- **딥러닝 (Deep Learning)**
  - 인공신경망을 여러 층을 쌓아 올려(=Deep), 입력에 따른 출력을 학습시키는 구조
  - 인공신경망의 구조를 깊게 구성하여 머신러닝을 수행한다는 측면에서, 머신러닝의 한 종류라고 볼 수 있음



# Part 4. 딥 러닝

- **딥러닝의 역사**
  - 퍼셉트론의 개념은 1957년 프랑크 로젠블라트(Frank Rosenblatt)에 의해 개발
  - 하지만, 퍼셉트론을 이용한 인공신경망을 구현하는 데에 여러 문제가 발생
    1. 신경망의 층(Layer)를 증가시킴에 따른 문제를 해결하지 못함 – **기울기 소실(Gradient Vanishing)**
    2. 데이터가 증가함에 따라 오히려 신경망의 성능이 떨어지는 문제를 해결하지 못함 – **과적합(Overfitting)**
    3. 큰 규모의 신경망을 학습할 수 있는 **하드웨어**와 **데이터 양**의 한계
  - 2000년대 이후, 위의 문제들을 해결할 수 있는 이론이 개발되고, 하드웨어의 발전과 데이터의 축적을 통해, 머신러닝/딥러닝의 연구가 다시 활발해지기 시작

## 1. 인공 신경망 Neural Network

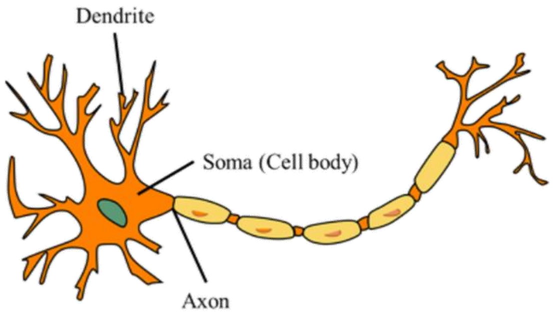
**4-1-1** | **Perceptron**

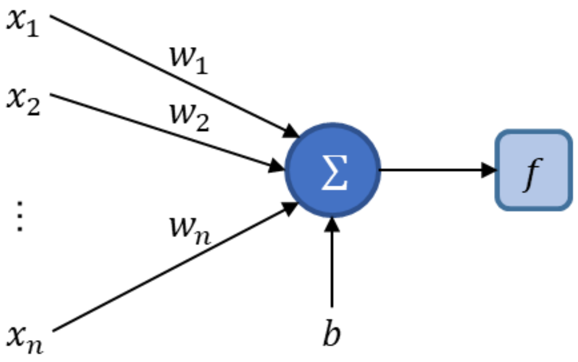
- **퍼셉트론의 개념**

  - 인간의 뇌가 가지는 생물학적 특성인 <u>뉴런의 연결 구조를 함수로 표현</u>

    → **퍼셉트론(Perceptron)**

  - 뉴런과 퍼셉트론의 비교




외부 자극

↓

외부 자극을
하나의 신호로 통합

↓

통합된 신호를 다음 뉴런으로 전달할지 판단

입력 $x_i$

↓

가중치를 부여하여 연산
$\sum w_i x_i + b$

↓

활성화 함수를 통해 출력
$y = f(\sum w_i x_i + b)$

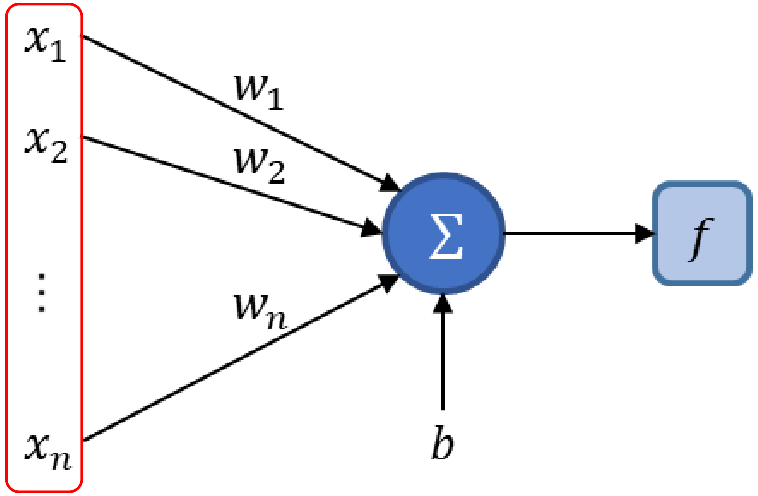
SNUH

## 1. 인공 신경망 Neural Network

| 4-1-2 | Perceptron |
| --- | --- |

- **퍼셉트론의 과정**

  1. 데이터 $x_1, x_2, \ldots, x_n$을 입력받음



SNUH

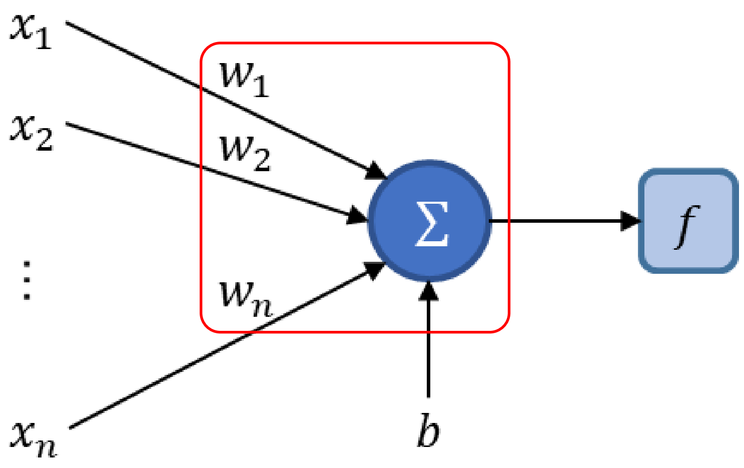
## 1. 인공 신경망 Neural Network

| 4-1-3 | Perceptron |
| --- | --- |

- **퍼셉트론의 과정**

  2. 데이터 $x_1, x_2, \ldots, x_n$에 각각의 가중치(weight)인 $w_1, w_2, \ldots, w_n$를 곱하여 더함 → $\sum w_i x_i$

  - **가중치(weight)**는 퍼셉트론이 학습해야 하는 파라미터(parameter)로, 학습을 진행할수록 최적의 가중치 값을 찾아가는 것이 목적. 입력신호가 결과에 미치는 영향을 조절



SNUH

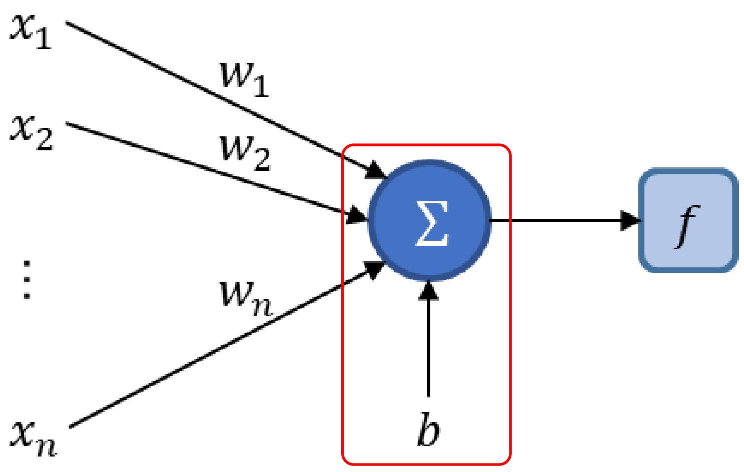
## 1. 인공 신경망 Neural Network

| 4-1-4 | Perceptron |
|---|---|

- **퍼셉트론의 과정**

3. $\sum w_i x_i$ 에 편향(bias)를 더해줌 → $\sum w_i x_i + b$

- **편향(bias)**은 가중치와 마찬가지로 퍼셉트론이 학습해야 하는 파라미터.
  퍼셉트론의 활성화에 대한 역치(threshold)를 조정하는 역할



SNUH

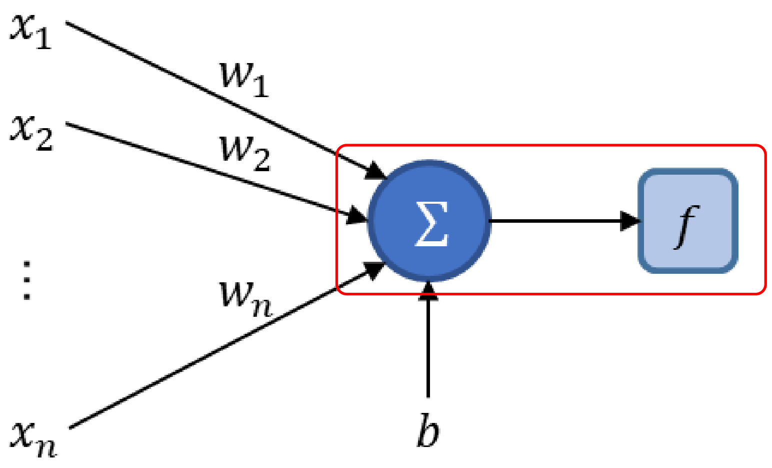
## 1. 인공 신경망 Neural Network

**4-1-5** | **Perceptron**

- **퍼셉트론의 과정**

4. $\sum w_i x_i + b$ 를 활성화 함수(Activation function) $f$ 를 통해 출력 $y$를 계산 → $y = f(\sum w_i x_i + b)$

- **활성화 함수(Activation function)**는 퍼셉트론의 연산 결과를 유의미한 결과로 변환



SNUH

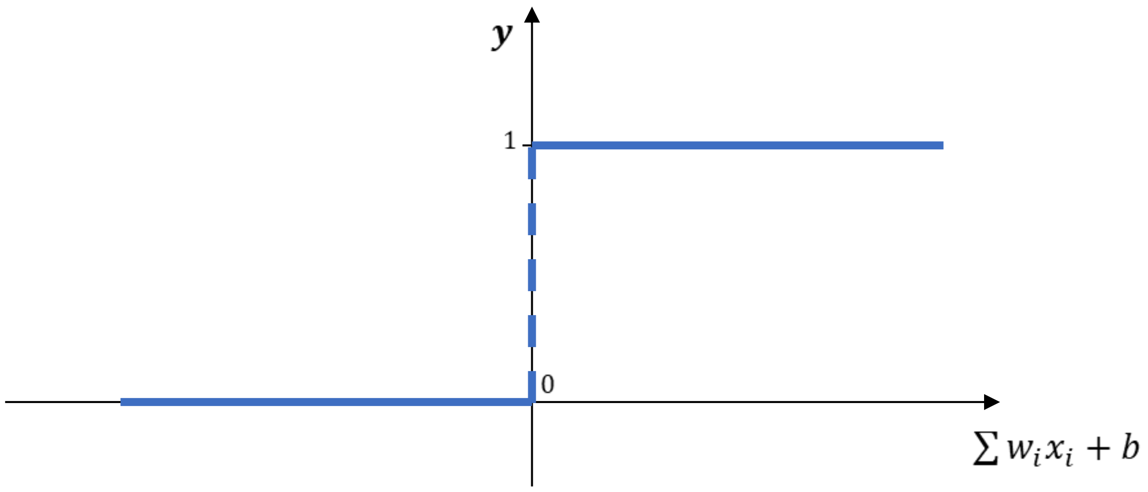
## 1. 인공 신경망 Neural Network

| 4-1-6 | Perceptron |
|---|---|

- **퍼셉트론의 과정**
  - 활성화 함수의 예시 – 계단 함수 (Step function)

$$if, \sum w_i x_i + b \geq 0 \rightarrow y = 1$$

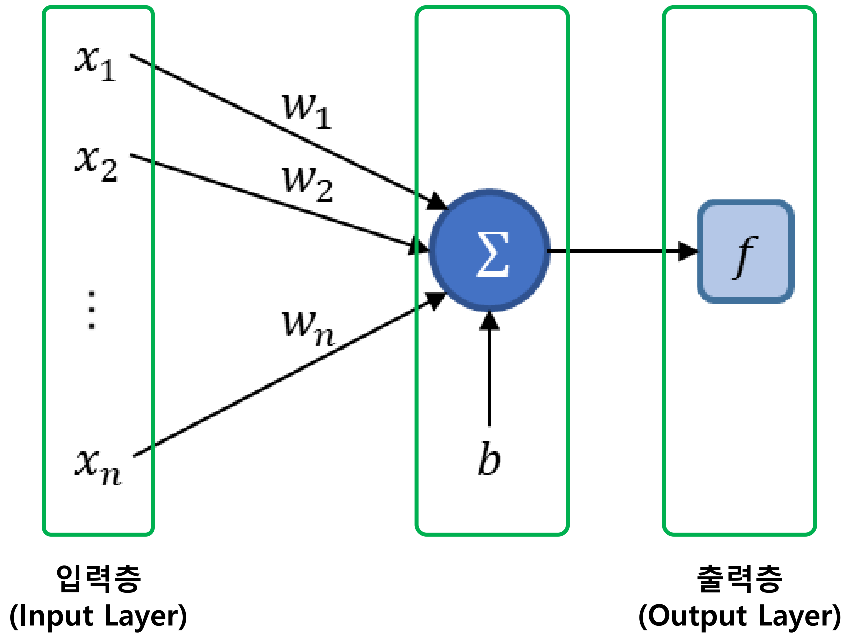
$$if, \sum w_i x_i + b < 0 \rightarrow y = 0$$



## 1. 인공 신경망 Neural Network

| 4-1-7 | Perceptron |
| --- | --- |

- **단층 퍼셉트론 (Single-layer Perceptron)**
  - 한 쌍의 입력층과 출력층으로 이루어진 퍼셉트론



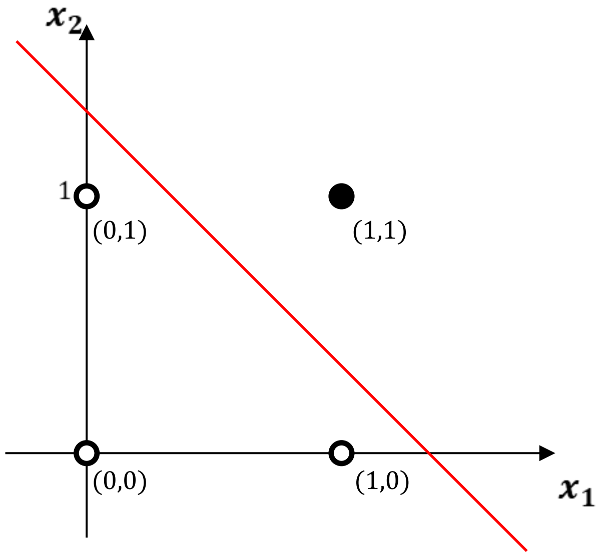
## 1. 인공 신경망 Neural Network

| 4-1-8 | Perceptron |
|---|---|

- **단층 퍼셉트론 (Single-layer Perceptron)**
  - 예시) 단층 퍼셉트론($\sum w_i x_i + b$)은 선형식으로 표현 가능

1) AND 게이트

| 입력 | | 출력 |
|---|---|---|
| $x_1$ | $x_2$ | $y$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



SNUH

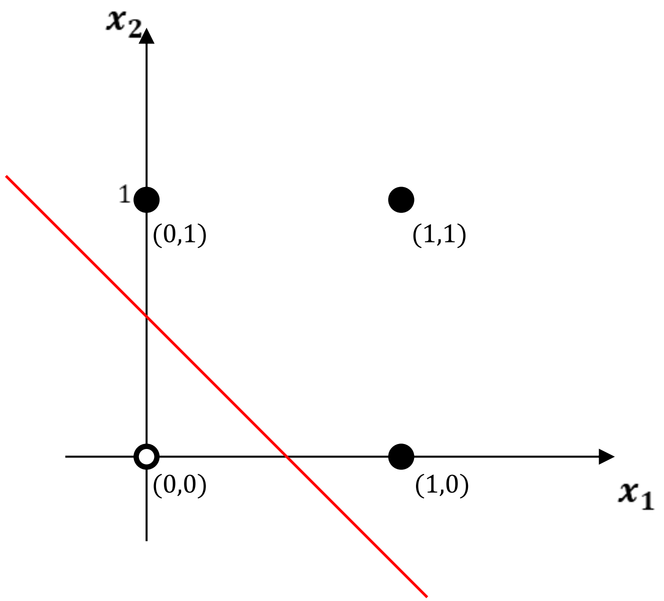
## 1. 인공 신경망 Neural Network

| 4-1-9 | Perceptron |
|---|---|

- **단층 퍼셉트론 (Single-layer Perceptron)**

  2) OR 게이트

| 입력 | | 출력 |
|---|---|---|
| $x_1$ | $x_2$ | $y$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



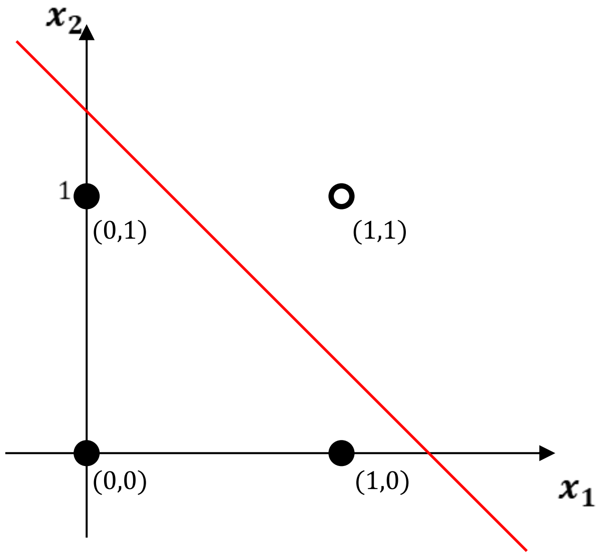
## 1. 인공 신경망 Neural Network

| 4-1-10 | Perceptron |
|---|---|

- **단층 퍼셉트론 (Single-layer Perceptron)**

  3) NAND 게이트

| 입력 | | 출력 |
|---|---|---|
| $x_1$ | $x_2$ | $y$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

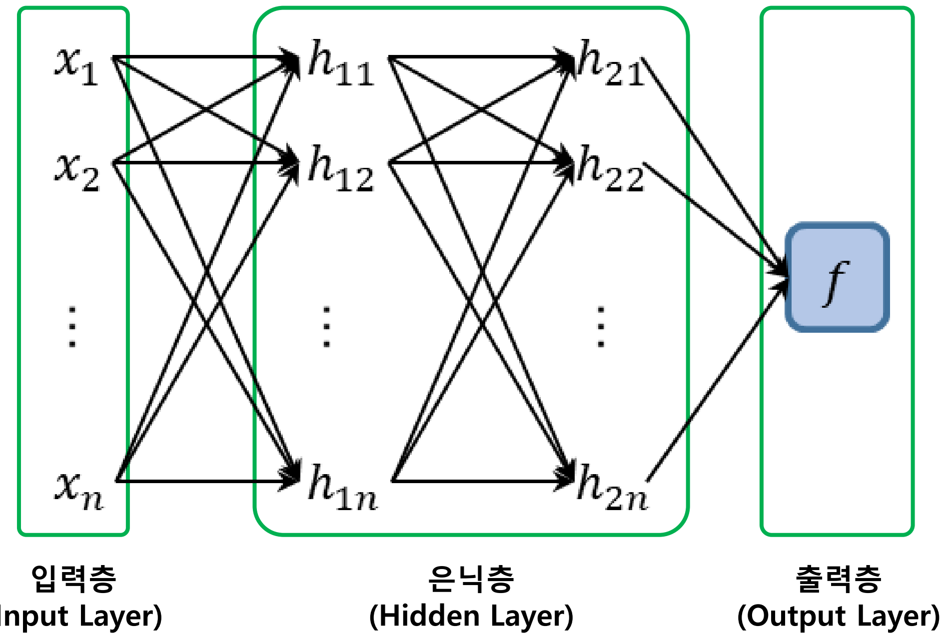


SNUH

## 1. 인공 신경망 Neural Network

| 4-1-11 | Perceptron |
|---|---|

- **다층 퍼셉트론(Multi-layer Perceptron)**
  - 단층 퍼셉트론으로 구현할 수 있는 AND, OR, NAND 게이트를 **조합**하여, XOR 게이트 이상의 복잡도를 구현 가능



SNUH

## 1. 인공 신경망 Neural Network

4-2-1 | **Activation Function**

- **활성화 함수의 비선형성 (Non-linearity)**
  - 예시)

    선형 활성화 함수 $f(x) = wx + b$

    두 번째 레이어의 활성화함수는 $f\big(f(x)\big) = w_2(w_1x + b_1) + b_2 = (w_1w_2)x + (b_1w_1 + b_2)$

    위의 식은 결국 $wx + b$ 의 형태로 표현 가능

    ∴ 레이어를 깊게 쌓아도, 결국 **하나의 레이어의 역할에 불과**

  - **비선형 활성화 함수**를 이용하여, 레이어를 깊게 쌓아 더 복잡한 문제를 해결할 수 있음

SNUH

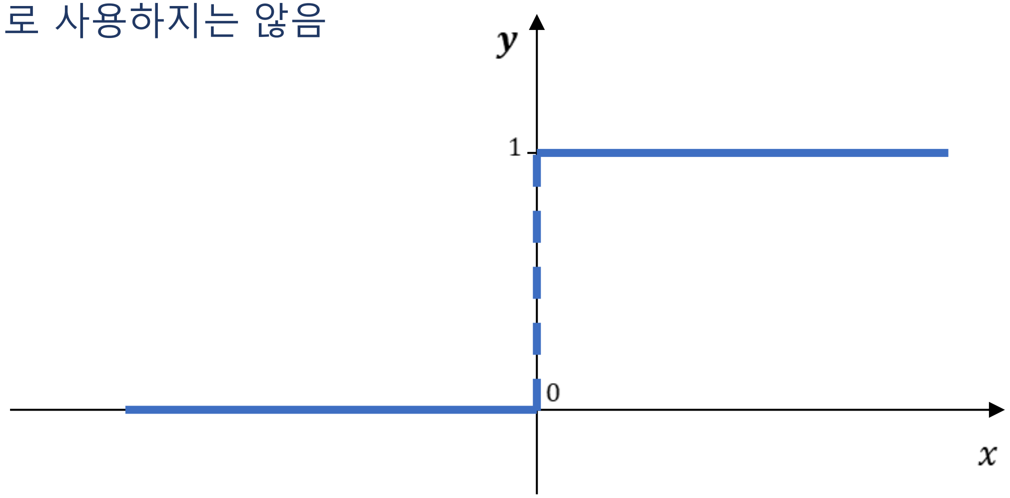
## 1. 인공 신경망 Neural Network

4-2-2 | Activation Function

- **1) 계단 함수 (Step function)**

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

  - 입력에 따라 0 또는 1의 값을 출력 (이산적)
  - 미세한 학습이 어려워 실제로 사용하지는 않음



SNUH

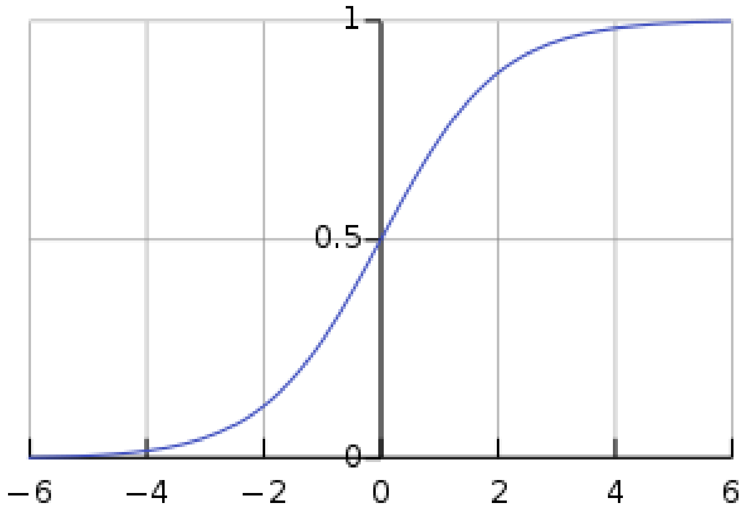
## 1. 인공 신경망 Neural Network

4-2-3 | Activation Function

- **2) 시그모이드 함수 (Sigmoid function)**

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

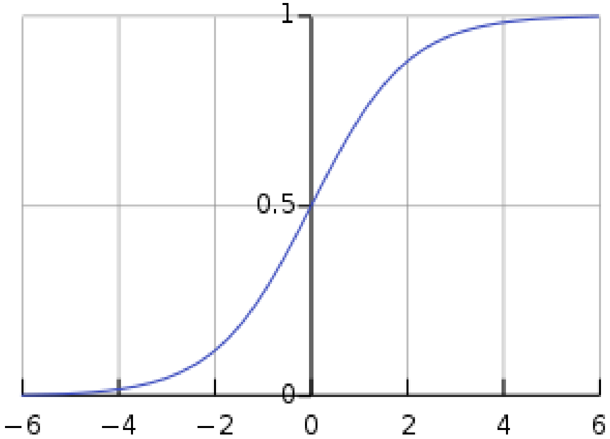
  - 입력에 따라 0 과 1 사이의 연속적인 값을 출력
  - 입력의 미세한 변화에도 학습이 가능

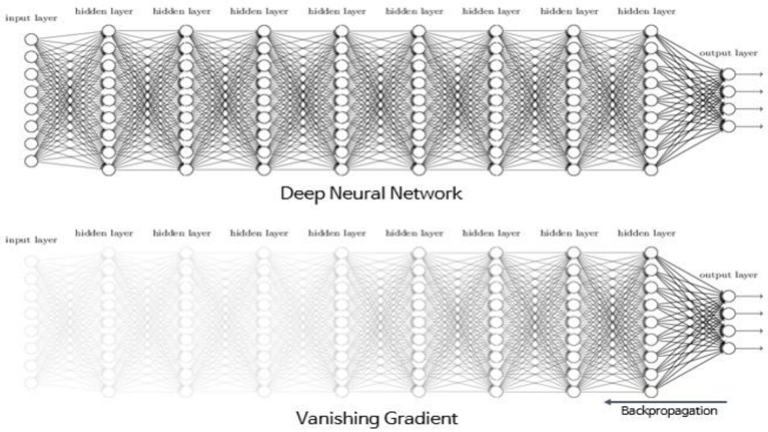


## 1. 인공 신경망 Neural Network

4-2-4 Activation Function

- **2) 시그모이드 함수 (Sigmoid function)**
  - 인공 신경망은 **활성화 함수의 미분값**을 이용하여 학습을 진행
  - 시그모이드 함수의 미분값(기울기)은 (0,0.25] 범위의 값을 가짐
  - 신경망이 깊어져 미분값을 계속 곱하게 되면, 그 값이 0에 가까워져 학습이 잘 되지 않음
  - 이를 **기울기 소실(Gradient Vanishing)** 현상이라 함



SNUH

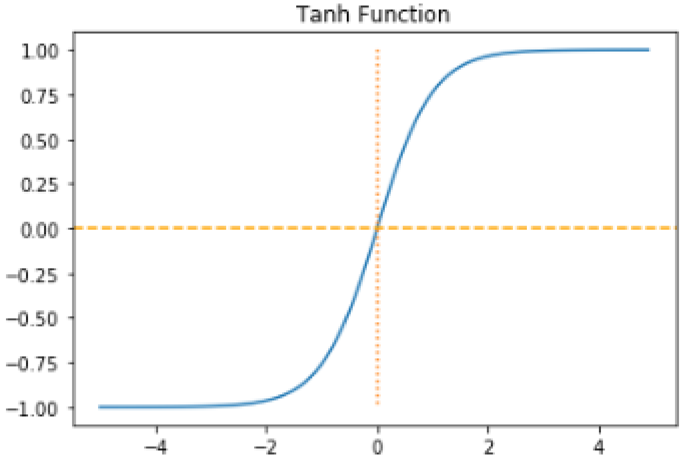
## 1. 인공 신경망 Neural Network

4-2-5 | Activation Function

### • 3) 하이퍼볼릭탄젠트 함수 (Hyperbolic tangent function)

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

- 입력에 따라 -1 과 1 사이의 연속적인 값을 출력
- 미분했을 때의 최대값이 시그모이드 함수보다는 커서, 기울기 소실 문제를 더 방지할 수 있음



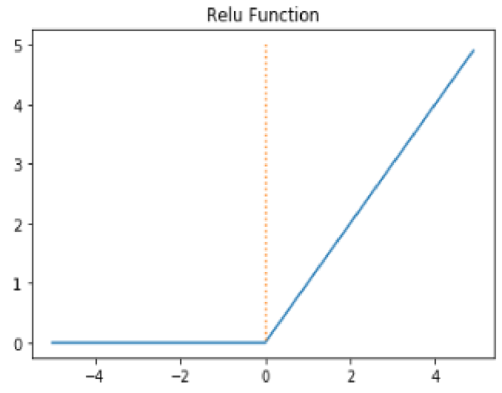
## 1. 인공 신경망 Neural Network

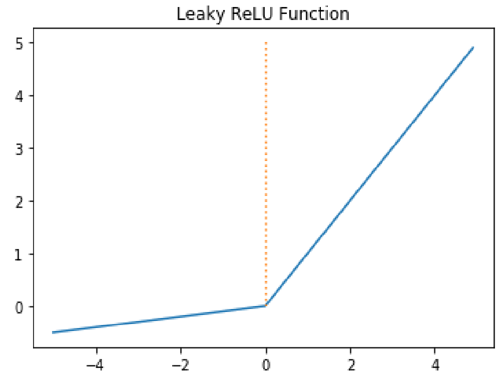
4-2-6 Activation Function

- **4) 렐루 함수 (ReLU function, Rectified Linear Unit)**

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

  - 입력이 음수이면 0, 양수이면 입력을 그대로 출력
  - 학습 속도가 빠르고, 기울기 소실 문제가 발생하지 않음
  - 입력이 음수인 경우, 가중치가 업데이트 되지 않는 현상(Dying ReLU)이 발생할 수 있음
  → 약간의 기울기를 적용한 Leaky ReLU 함수를 통해 해결 가능



SNUH

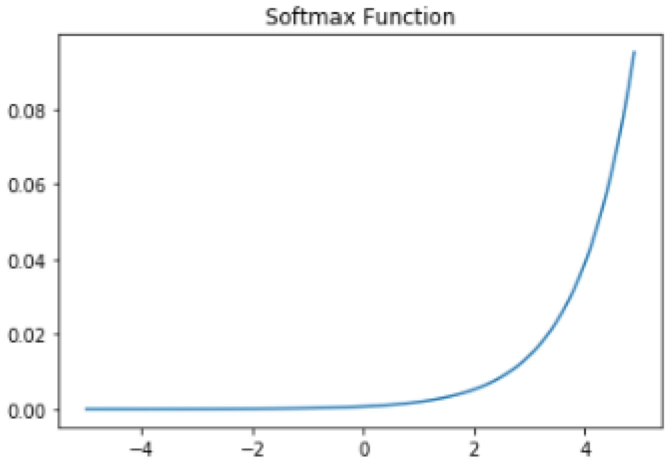
## 1. 인공 신경망 Neural Network

| 4-2-7 | Activation Function |
| --- | --- |

- **5) 소프트맥스 함수 (Softmax function)**

$$f(x_i) = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

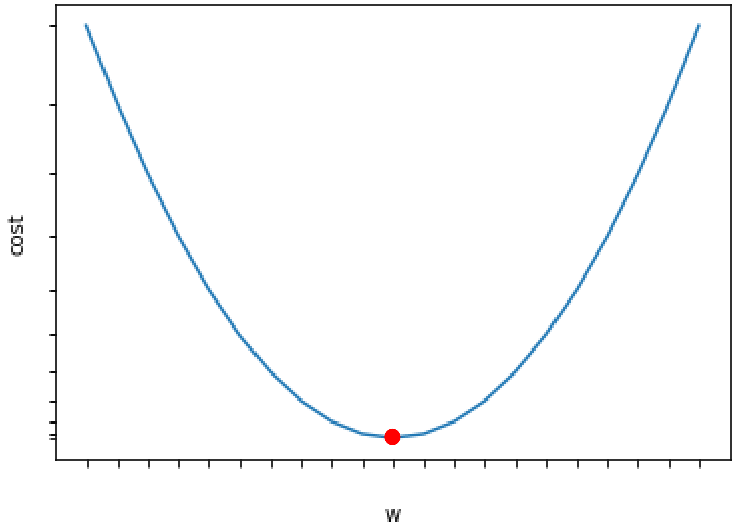
  - 여러 클래스에 대한 입력을 각 클래스에 대한 확률로 변환하여 출력
  - 다중 클래스 분류 모델의 출력층에 이용



## 1. 인공 신경망 Neural Network

4-3-1 Cost Function

- **오차 (손실, Cost, Loss)**
  - 모델의 예측 값과 실제 값의 차이
  - 오차가 작을수록 학습이 잘 되었다고 볼 수 있음
  - 반대로 오차가 크다면, 학습이 잘 되지 않았으므로 모델을 더 많이 학습시켜야 함 → $w$와 $b$를 더 많이 수정해야 함




- **오차 함수(손실 함수, Loss function, Cost function)**
  - 오차를 수치화 해주는 함수
  - **오차 함수의 값을 최소화**하는 최적의 **$w$와 $b$를 찾는 것**이 학습의 목표

## 1. 인공 신경망 Neural Network

4-3-2 Cost Function

### 1) 평균절대오차(MAE, Mean Absolute Error)

- 오차 함수 $Cost(w, b)$를 예측값 $\hat{Y}$와 실제값 $Y$ 간 차이의 절대값인 $|\hat{Y} - Y|$로 정의

$$Cost(W, b) = \frac{1}{n}\sum_{i}|\hat{Y_i} - Y_i|$$

- MAE는 미분이 불가능하다는 단점이 있음

$$\frac{\partial}{\partial w}Cost(W, b) = \frac{\partial}{\partial w}\frac{1}{n}\sum_{i}|\hat{Y_i} - Y_i|$$

$$= \frac{\partial}{\partial w}\frac{1}{n}\sum_{i}|wx_i + b - Y_i|$$

SNUH

## 1. 인공 신경망 Neural Network

4-3-3 | Cost Function

### 2) 평균제곱오차(MSE, Mean Squared Error)

$$Cost(W,b) = \frac{1}{n}\sum_{i}\left(\widehat{Y}_i - Y_i\right)^2$$

- MSE는 미분 가능

$$\frac{\partial}{\partial w}Cost(W,b) = \frac{\partial}{\partial w}\frac{1}{n}\sum_{i}\left(\widehat{Y}_i - Y_i\right)^2 = \frac{\partial}{\partial w}\frac{1}{n}\sum_{i}(wx_i + b - Y_i)^2$$

$$= \frac{1}{n}\sum_{i} 2x_i(wx_i + b - Y_i) = \frac{2}{n}\sum_{i} x_i(\widehat{Y}_i - Y_i)$$

## 1. 인공 신경망 Neural Network

**4-3-4** Cost Function

### 3) 교차 엔트로피 에러(Cross Entropy Error)

- 분류(Classification) 문제에서 사용되는 오차 함수

$$Cost = \sum_{c=1}^{N} y_c \log p_c$$

- $y_c$는 실제값(0 or 1), $p_c$는 예측값(0 ~ 1 사이의 값)

SNUH

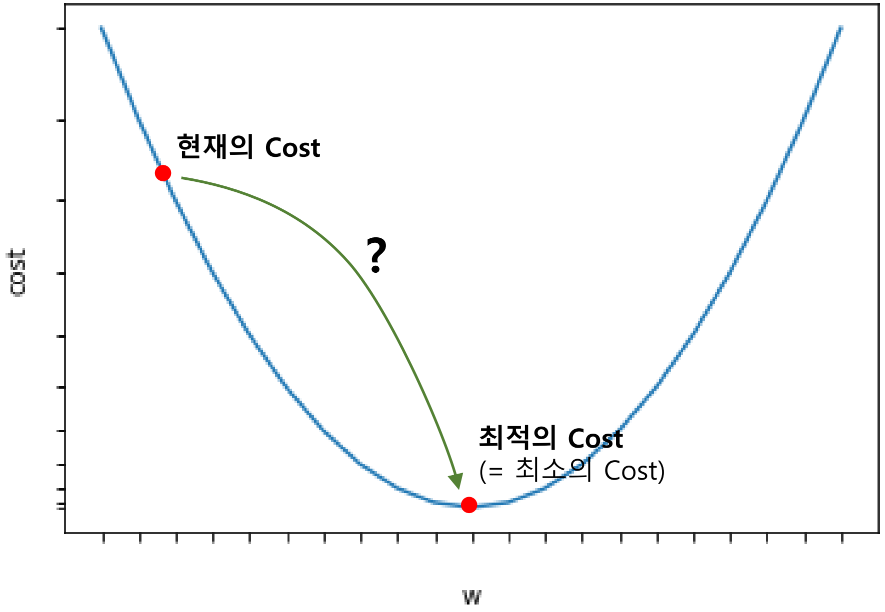
## 1. 인공 신경망 Neural Network

**4-4-1** | **Optimizer**

- **옵티마이저(Optimizer)**

  - 오차 함수를 통해 최적의 $w$를 찾는 방법

    → 계산된 오차를 모델에게 **어떻게** 학습 시킬 것인가?



SNUH

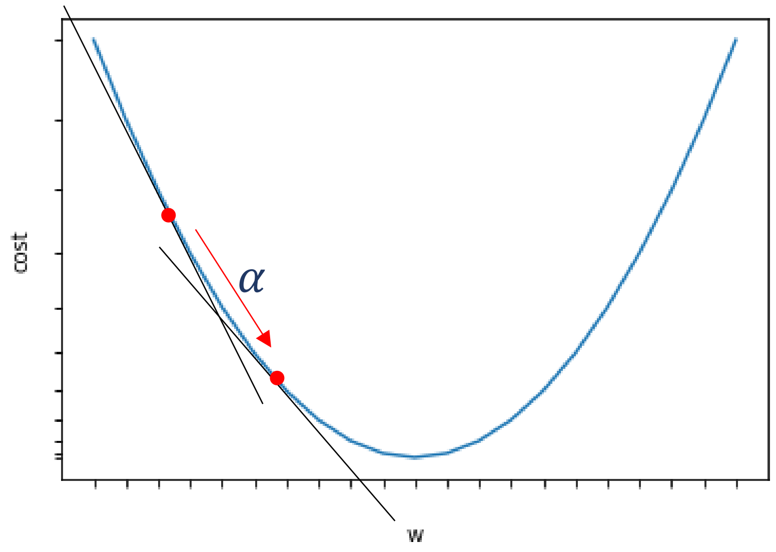
## 1. 인공 신경망 Neural Network

4-4-2 **Optimizer**

### 1) 경사 하강법(Gradient descent)

- 가장 기본적인 옵티마이저
- 손실 함수의 기울기를 더 낮은 쪽으로 학습 → $w$ 값을 업데이트
- 손실 함수를 $w$에 대해 미분하여, 그 미분값을 연산하여 $w$를 수정

$$w := w - \alpha \frac{\partial}{\partial w} cost(w, b)$$

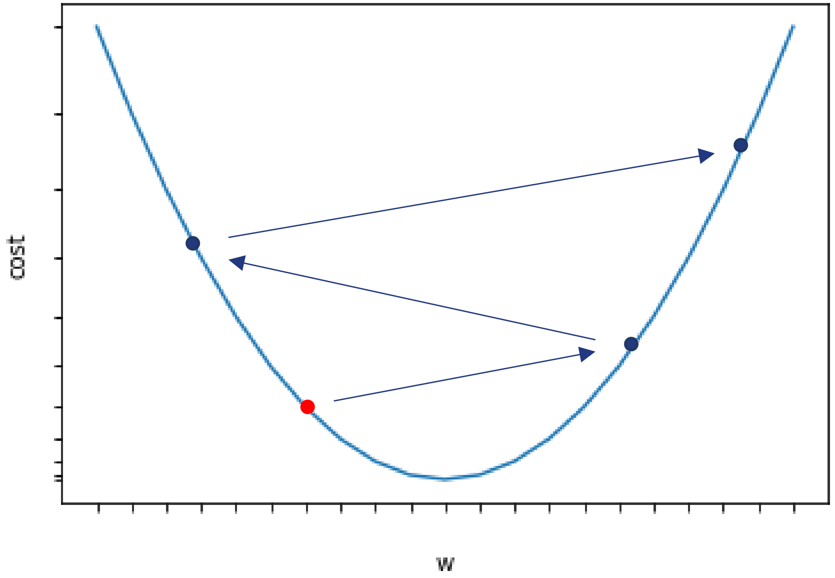



- $\alpha$는 학습률(Learning rate)로, $w$를 얼마나 수정할지를 결정하는 파라미터

## 1. 인공 신경망 Neural Network

| 4-4-3 | Optimizer |
| --- | --- |

- **학습률(Learning rate)의 의미**
  - 학습률이 너무 큰 경우, 최적값의 방향으로 수렴하지 않고 발산할 수 있음

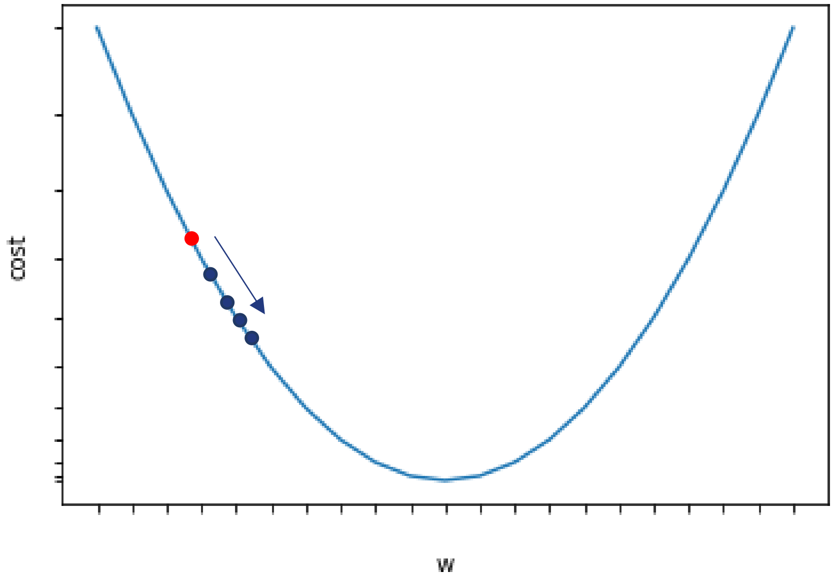


SNUH

## 1. 인공 신경망 Neural Network

| 4-4-4 | Optimizer |
| --- | --- |

- **학습률(Learning rate)의 의미**
  - 학습률이 너무 작은 경우, 수렴의 속도가 느려져 학습의 효율성이 떨어질 수 있음



## 1. 인공 신경망 Neural Network

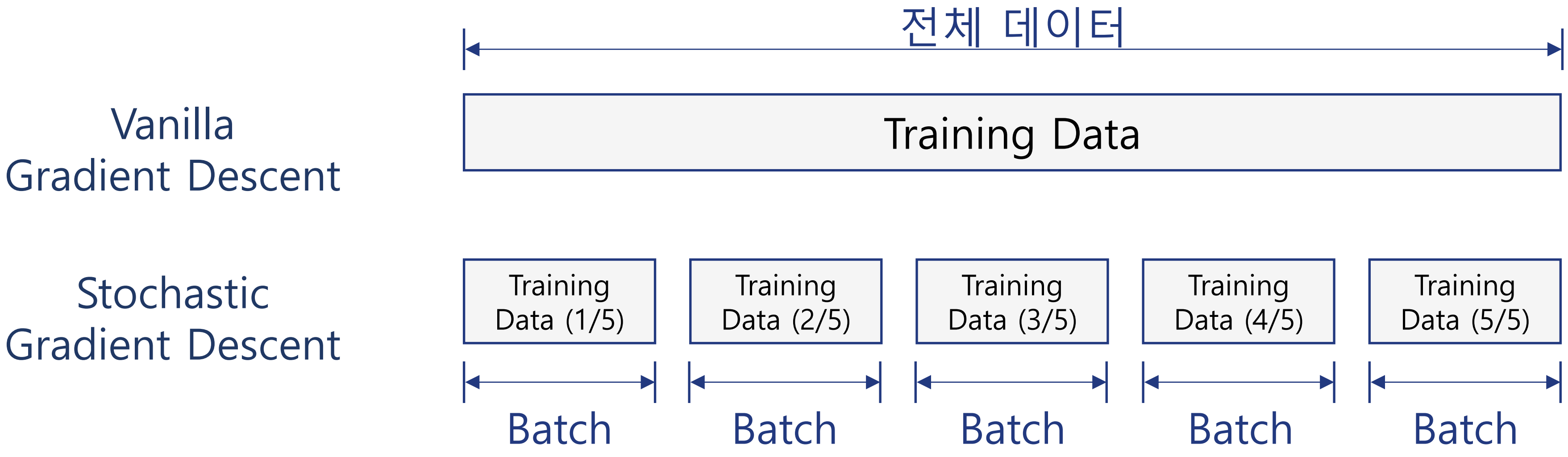
| 4-4-5 | Optimizer |
|---|---|

### 2) 확률적 경사 하강법(SGD, Stochastic Gradient Descent)

- 일반적인 경사 하강법은 전체 데이터에 대해서 한 번의 학습을 진행

  → 학습이 너무 느리다는 단점이 있음

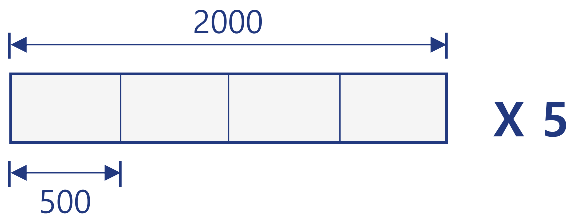
- 전체 데이터 중 **일부(Batch)**만 이용하여 학습하여, 학습의 속도를 향상



## 1. 인공 신경망 Neural Network

| 4-4-6 | Optimizer |
|---|---|

- **Epoch / Batch size / Iteration**
    - **Epoch** (에포크) : 전체 데이터셋을 한 번 학습한 상태
    - **Batch size** (배치 사이즈) : 한 번 학습할 때 주어지는 데이터 사이즈
    - **Iteration** (반복) : 1번의 Epoch을 달성하기 위한 Batch size



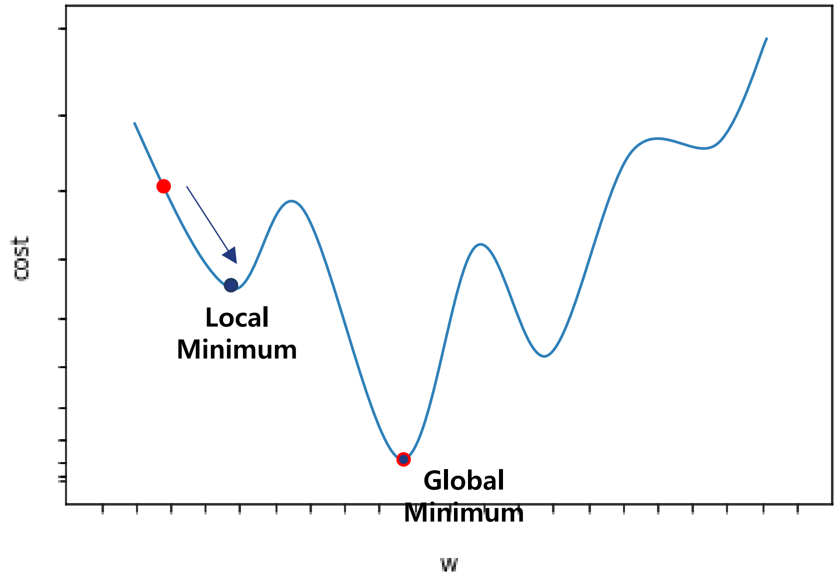

- Ex) 2000개의 데이터를 500개의 Batch size로 나누어 5번의 Epoch 만큼 학습시키려면, 1번의 Epoch을 학습하는 데에 필요한 Iteration은 4번(500 / 2000 = 4)이며, 5번의 Epoch을 학습하기 위해서는 20번(4 X 5 = 20)의 Iteration이 필요하다.

SNUH

## 1. 인공 신경망 Neural Network

4-4-7 **Optimizer**

- **경사 하강법의 한계**
  - 경사 하강법은 한 번 **극소값(Local minimum)**에 수렴하면, **최소값(Global minimum)**을 찾을 수 없음
  - 극소값을 벗어나기 위한 방법이 필요



SNUH

## 1. 인공 신경망 Neural Network

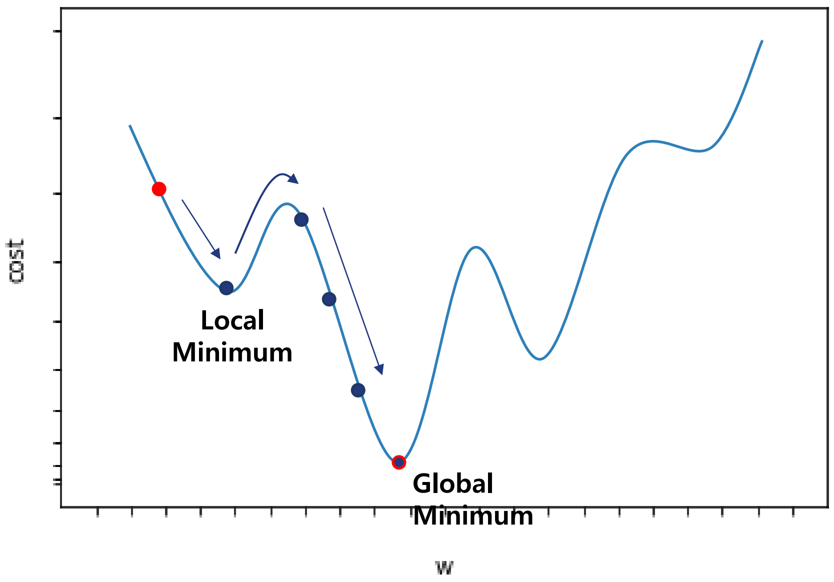
4-4-8 | **Optimizer**

### 3) 모멘텀 (Momentum)

- 경사 하강법의 알고리즘에 **이전 업데이트의 관성**(Momentum, $\gamma$)을 **추가**하여, Local minimum을 넘어갈 수 있음

$$v_t = \gamma v_{t-1} + \alpha \frac{\partial}{\partial w} cost_t(w, b)$$

$$w_{t+1} = w_t - v_t$$



SNUH

## 1. 인공 신경망 Neural Network

| 4-4-9 | Optimizer |
|---|---|

### 4) Adagrad (Adaptive Gradient)

- 학습률 $\alpha$를 조절하며 학습하는 방법
- 현재까지 **학습이 많이 된**(=가중치 $w$가 업데이트된 총량이 높은) 가중치는 **학습률을 감소**시켜 더 **세밀한 학습**을 시킴
- 현재까지 **학습이 적게 된**(=가중치 $w$가 업데이트된 총량이 낮은) 가중치는 **학습률을 증가**시켜 더 **빠른 학습**을 시킴

$$G_t = G_{t-1} + \left\{\frac{\partial}{\partial w} cost_t(w,b)\right\}^2 = \sum_1^t \left\{\frac{\partial}{\partial w} cost_i(w,b)\right\}^2$$

$$w_{t+1} := w_t - \alpha * \frac{1}{\sqrt{G_t + \epsilon}} * \frac{\partial}{\partial w} cost_t(w,b)$$

- $G_t$는 현재까지 업데이트된 총량, $\epsilon$은 분모가 0이 되는 것을 방지하기 위한 값

SNUH

## 1. 인공 신경망 Neural Network

4-4-10 Optimizer

### 5) RMSprop (Root Mean Square Propagation)

- Adagrad 방법은 학습이 반복됨에 따라 학습률은 필연적으로 작아진다는 단점이 있음 ($G_t$가 무한히 증가하기 때문)
- 이를 보완하기 위하여 RMSprop 방법에서는 $G_t$를 전체 총합이 아닌 **지수 이동평균**으로 이용
- 최근 값들에는 큰 가중치를 주어 영향력을 높이고, 이전 값들에는 작은 가중치를 주어 영향력을 점점 감소

$$G_t = \gamma G_{t-1} + (1-\gamma)\left\{\frac{\partial}{\partial w} cost_t(w,b)\right\}^2$$

$$w_{t+1} := w_t - \alpha * \frac{1}{\sqrt{G_t + \epsilon}} * \frac{\partial}{\partial w} cost_t(w,b)$$

## 1. 인공 신경망 Neural Network

**4-4-11** | **Optimizer**

### 6) Adam (Adaptive Moment Estimation)

- RMSprop(Adaptive) 방법과 Momentum 방법을 모두 적용

Momentum 항 : $M_t = \beta_1 M_{t-1} + (1-\beta_1)\frac{\partial}{\partial w}cost_t(w,b)$

이동평균 항 : $V_t = \beta_2 V_{t-1} + (1-\beta_2)\left\{\frac{\partial}{\partial w}cost_t(w,b)\right\}^2$

$$\widehat{M}_t = \frac{M_t}{1-\beta_1}$$

$$\widehat{V}_t = \frac{V_t}{1-\beta_2}$$

학습 초기에 0 으로 편향되는 현상을 방지

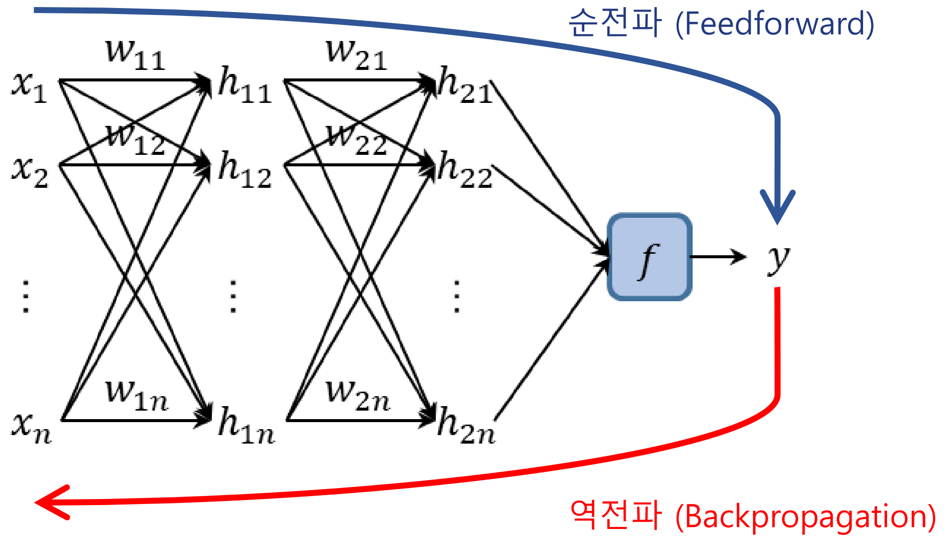
$$w_{t+1} := w_t - \alpha * \frac{1}{\sqrt{\widehat{V}_t + \epsilon}} * \widehat{M}_t$$

SNUH

## 1. 인공 신경망 Neural Network

| 4-5-1 | Backpropagation |
|---|---|

- **역전파(Backpropagation)**
  - 신경망의 가중치를 업데이트
  - 신경망의 최종 출력인 오차(Error)에 각 가중치가 주는 영향을 역으로 계산



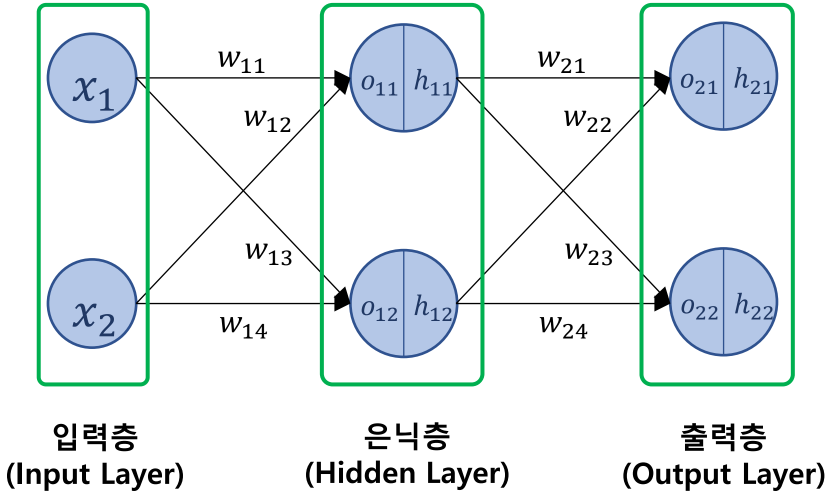
SNUH

## 1. 인공 신경망 Neural Network

4-5-2 | Backpropagation

### • 예시 – 순전파

- 각 1개의 입력층, 은닉층, 출력층로 구성된 신경망
- 학습해야하는 가중치는 총 8가지 → $(w_{11}, w_{12}, w_{13}, w_{14}, w_{21}, w_{22}, w_{23}, w_{24} w_{25})$
- 편향 $b_{ij}$는 없다고 가정

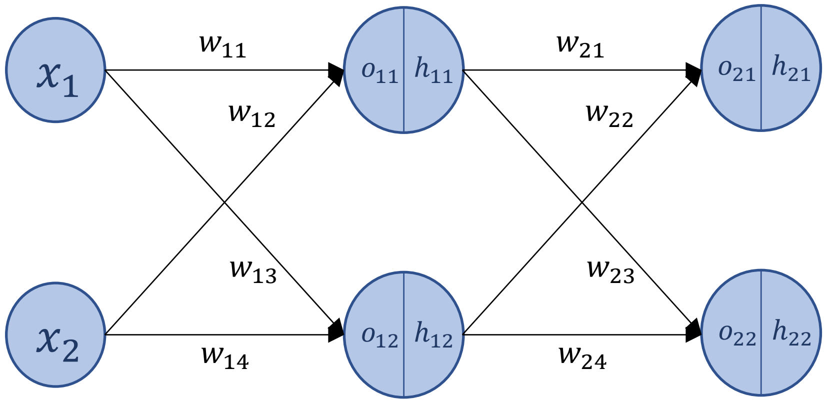


SNUH

## 1. 인공 신경망 Neural Network

4-5-3 | Backpropagation

- **예시 – 순전파**
  - $o_{ij}$는 입력값과 가중치 $w_{ij}$의 곱
  - $h_{ij}$는 $o_{ij}$에 활성화 함수를 적용한 값
  - 활성화 함수는 모두 시그모이드(Sigmoid) 함수로 가정

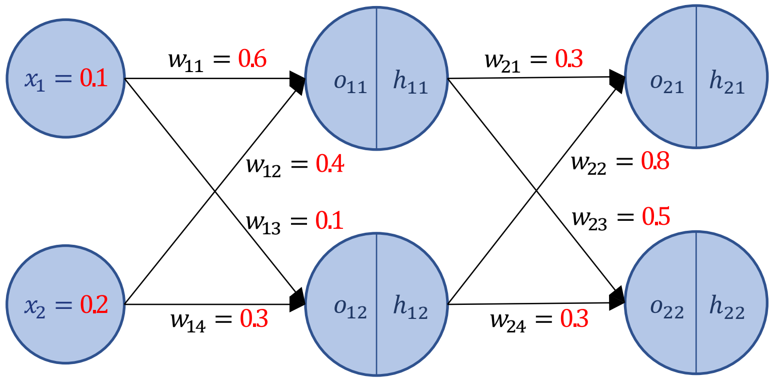


SNUH

## 1. 인공 신경망 Neural Network

| 4-5-4 | Backpropagation |
| --- | --- |

- **예시 – 순전파**
  - 가중치의 초기 값을 랜덤하게 설정
  - 입력값으로 $(x_1, x_2) = (0.1, 0.2)$를 입력

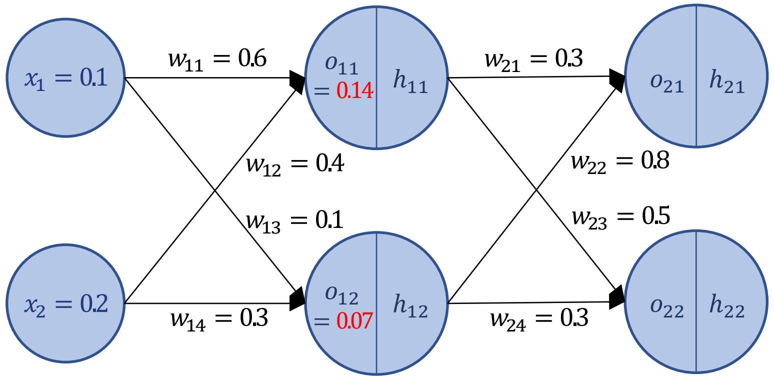


SNUH

## 1. 인공 신경망 Neural Network

**4-5-5** Backpropagation

- **예시 – 순전파**
  - $o_{11} = w_{11}x_1 + w_{12}x_2 = 0.6 \cdot 0.1 + 0.4 \cdot 0.2 = 0.14$
  - $o_{12} = w_{13}x_1 + w_{14}x_2 = 0.1 \cdot 0.1 + 0.3 \cdot 0.2 = 0.07$

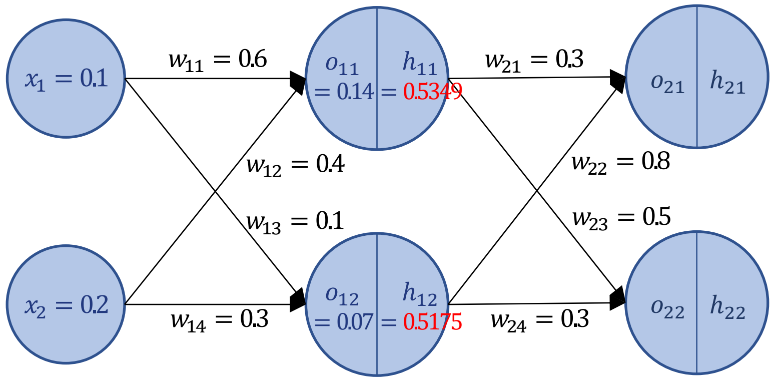


SNUH

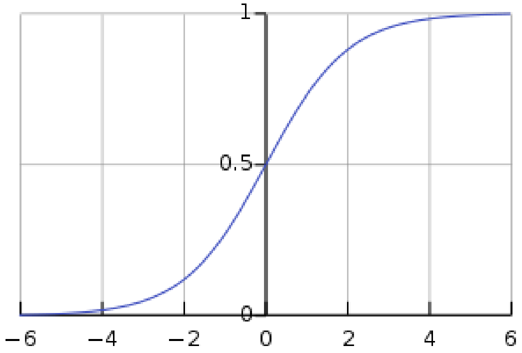
## 1. 인공 신경망 Neural Network

**4-5-6** Backpropagation

- **예시 – 순전파**
  - $h_{11} = f(o_{11}) = \frac{1}{1+e^{-o_{11}}} = \frac{1}{1+e^{-0.14}} = 0.5349$
  - $h_{12} = f(o_{12}) = \frac{1}{1+e^{-o_{12}}} = \frac{1}{1+e^{-0.07}} = 0.5175$




**Sigmoid Function**




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

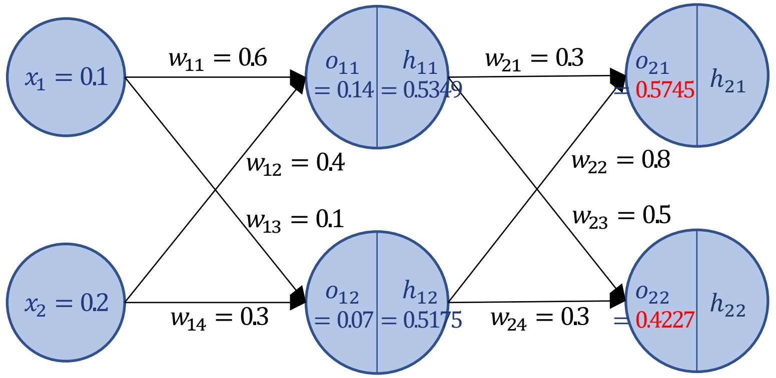
SNUH

## 1. 인공 신경망 Neural Network

4-5-7 Backpropagation

### • 예시 – 순전파

- $o_{21} = w_{21}h_{11} + w_{22}h_{12} = 0.3 \cdot 0.5349 + 0.8 \cdot 0.5175 = 0.5745$
- $o_{22} = w_{23}h_{11} + w_{24}h_{12} = 0.5 \cdot 0.5349 + 0.3 \cdot 0.5175 = 0.4227$

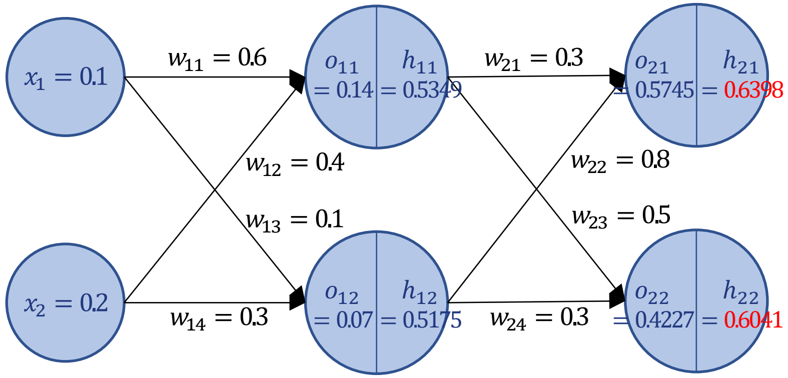


SNUH

## 1. 인공 신경망 Neural Network

4-5-8 Backpropagation

- **예시 – 순전파**
  - $h_{21} = f(o_{21}) = \frac{1}{1+e^{-o_{21}}} = \frac{1}{1+e^{-0.5745}} = 0.6398$
  - $h_{22} = f(o_{22}) = \frac{1}{1+e^{-o_{22}}} = \frac{1}{1+e^{-0.4227}} = 0.6041$
  - 신경망의 최종 출력인 $(h_{21}, h_{22})$이 입력 $(x_1, x_2) = (0.1, 0.2)$에 따른 신경망(모델)의 예측값

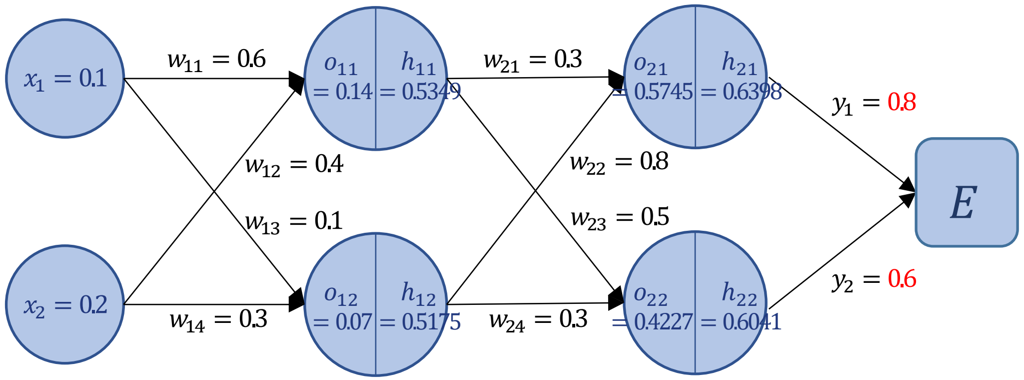


SNUH

## 1. 인공 신경망 Neural Network

4-5-9 **Backpropagation**

- **예시 – 순전파**
  - 신경망인 최종 출력인 $(h_{21}, h_{22}) = (0.6398, 0.6041)$를 통해 오차를 계산
  - 입력 $(x_1, x_2) = (0.1, 0.2)$에 따른 실제 값은 $(y_1, y_2) = (0.8, 0.6)$이라고 가정
  - 오차 함수(Loss function)는 평균제곱오차(MSE)를 사용



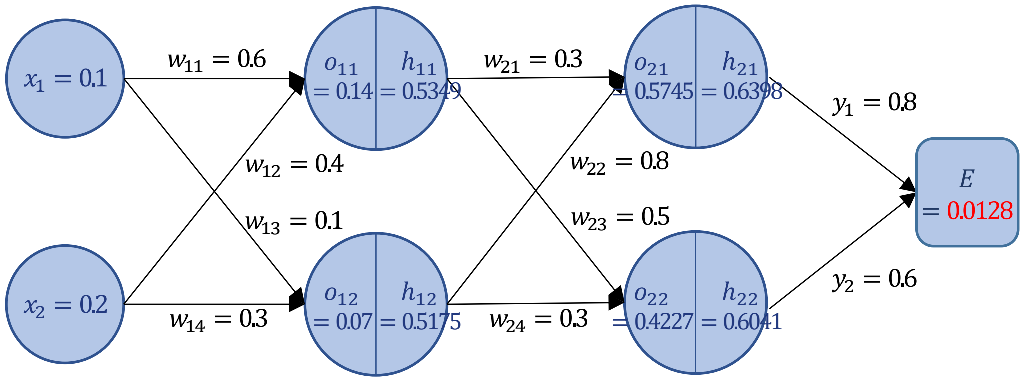
SNUH

## 1. 인공 신경망 Neural Network

| 4-5-10 | Backpropagation |
| --- | --- |

### • 예시 – 순전파

- 총 오차 $E = \frac{1}{2}\sum_i (y_i - h_{2i})^2$
- $E = \frac{1}{2}\{(y_1 - h_{21})^2 + (y_2 - h_{22})^2\} = \frac{1}{2}\{(0.8 - 0.6398)^2 + (0.6 - 0.6041)^2\} = 0.0128$



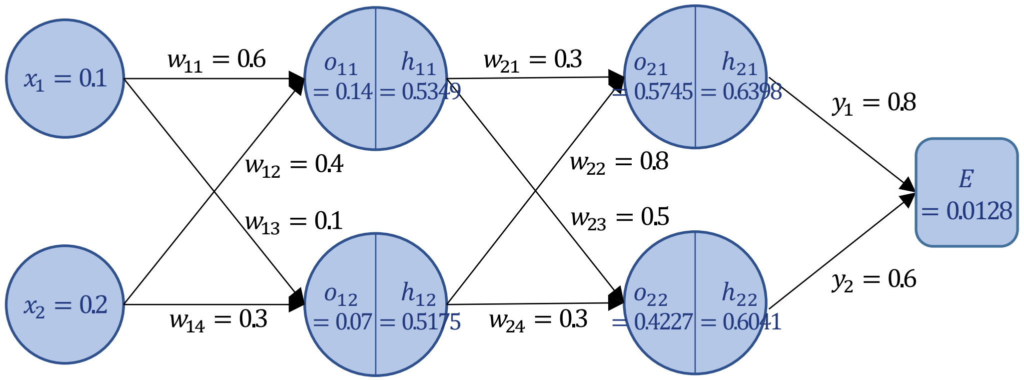
SNUH

## 1. 인공 신경망 Neural Network

**4-5-11** **Backpropagation**

- **예시 – 역전파**
  - 총 오차 $E$를 각 가중치 $w_{ij}$에 대해 미분하여 경사 하강법으로 가중치를 업데이트

$$w_{ij} := w_{ij} - \alpha \frac{\partial}{\partial w_{ij}} E$$



SNUH

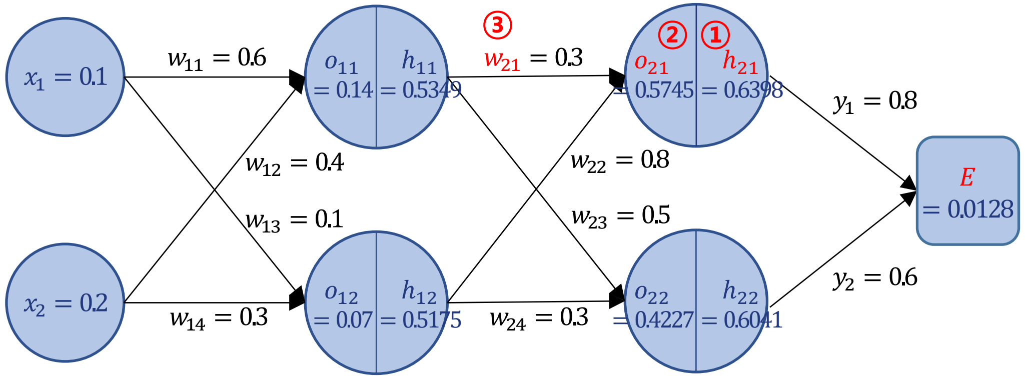
## 1. 인공 신경망 Neural Network

| 4-5-12 | Backpropagation |
| --- | --- |

### • 예시 – 역전파

- $w_{21}$에 대한 총 오차의 미분값인 $\frac{\partial E}{\partial w_{21}}$를 계산
- $w_{21}$에 대해 미분하기 위해서는, ①$E$를 계산하는 손실 함수, ②$h_{21}$를 계산하는 활성화 함수, ③$o_{21}$를 계산하는 함수

  에 대한 미분이 필요



SNUH

## 1. 인공 신경망 Neural Network

| 4-5-13 | Backpropagation |
| --- | --- |

- **예시 – 역전파**
  - 연쇄 법칙(Chain rule)
    - 함수 $g$가 $x_0$에서 미분 가능하고, 함수 $f$가 $g(x_0)$에서 미분 가능할 때, $(f \circ g)'(x_0) = f'\big(g(x_0)\big)g'(x_0)$
    - 즉, $y = f(u)$, $u = f(x)$로 표현하면,

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

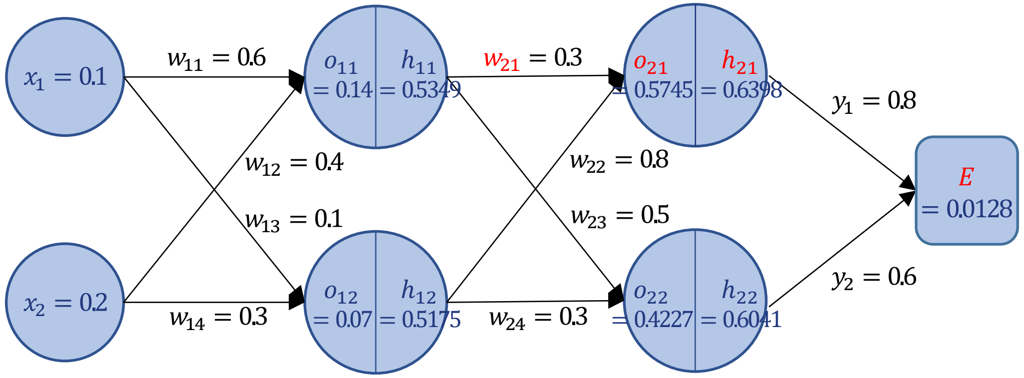
  - 연쇄 법칙을 이용하면, 역전파 과정에서 총 오차를 각 기울기에 대해 미분할 수 있음

SNUH

## 1. 인공 신경망 Neural Network

| 4-5-14 | Backpropagation |
|---|---|

- **예시 – 역전파**
  - 연쇄법칙에 의해, $\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w_{21}}$ 로 미분 가능



SNUH

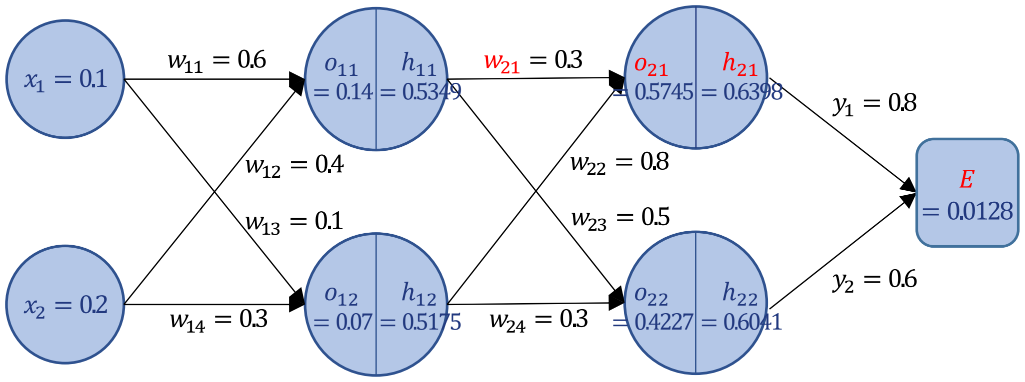
## 1. 인공 신경망 Neural Network

4-5-15 | Backpropagation

- **예시 – 역전파**

① $\frac{\partial E}{\partial h_{21}} = \frac{\partial}{\partial h_{21}} \frac{1}{2} \sum_i (y_i - h_{2i})^2 = \frac{\partial}{\partial h_{21}} \frac{1}{2} \{(y_1 - h_{21})^2 + (y_2 - h_{22})^2\}$

$= \frac{1}{2}\{2(y_1 - h_{21})\} = \frac{1}{2}\{2(0.6398 - 0.8)\} = -0.1602$



SNUH

## 1. 인공 신경망 Neural Network

| 4-5-16 | Backpropagation |
| --- | --- |

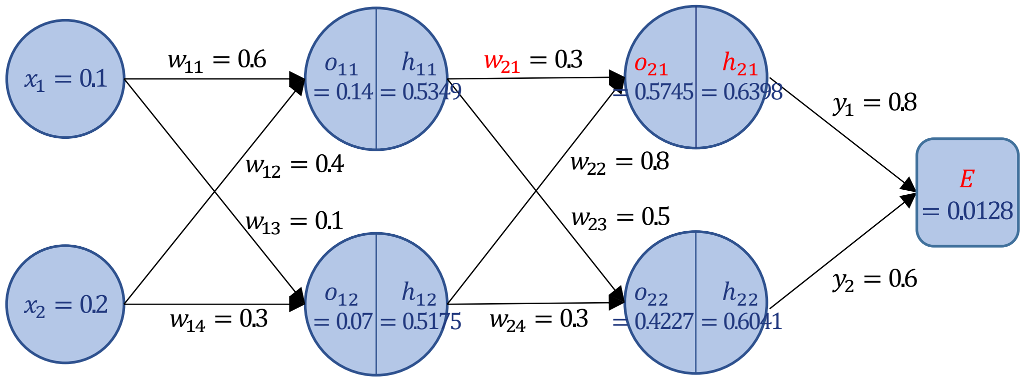
- **예시 – 역전파**

② $\frac{\partial h_{21}}{\partial o_{21}} = h_{21}(1 - h_{21}) = 0.6398(1 - 0.6398) = 0.2305$

**Sigmoid Function**

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

$$f'(x) = f(x) \cdot (1 - f(x))$$



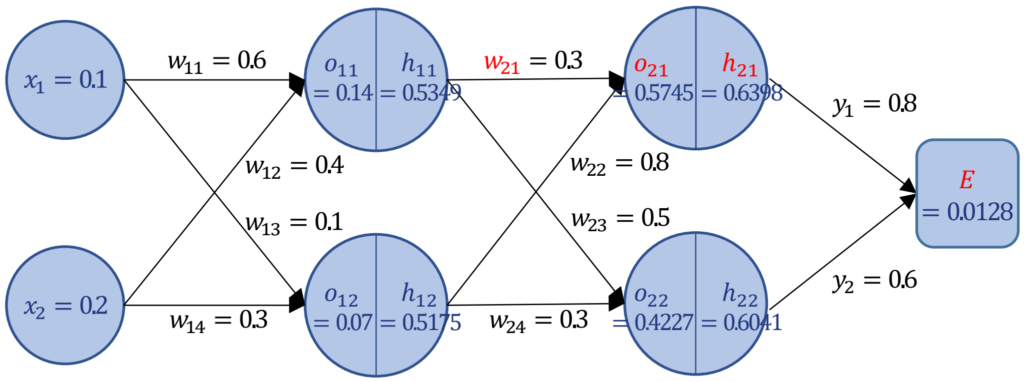
SNUH

## 1. 인공 신경망 Neural Network

| 4-5-17 | Backpropagation |
|---|---|

### • 예시 – 역전파

③ $\frac{\partial o_{21}}{\partial w_{21}} = \frac{\partial}{\partial w_{21}}(w_{21}h_{11} + w_{22}h_{12}) = h_{11} = 0.5349$



SNUH

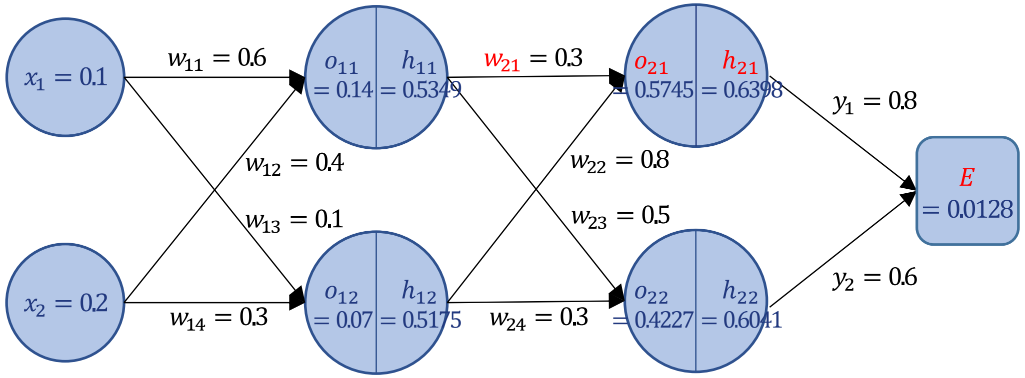
## 1. 인공 신경망 Neural Network

| 4-5-18 | Backpropagation |
| --- | --- |

- **예시 – 역전파**
  - 따라서, 학습률 $\alpha = 0.5$ 로 가중치 $w_{21}$을 업데이트 한다면,

$$w_{21} := w_{21} - \alpha \frac{\partial}{\partial w_{21}} E$$

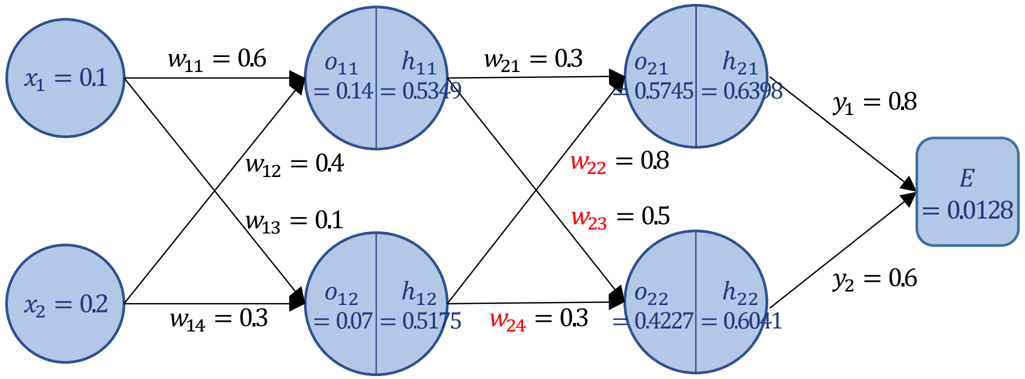
$$= w_{21} - \alpha \frac{\partial E}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w_{21}} = 0.3 - (0.5) \cdot (-0.1602) \cdot 0.2305 \cdot 0.5349 = 0.3 + 0.0099 = 0.3099$$



## 1. 인공 신경망 Neural Network

| 4-5-19 | Backpropagation |
|---|---|

- **예시 – 역전파** (같은 방법으로, $w_{22}, w_{23}, w_{24}$ 를 업데이트)
  - $w_{22} := w_{22} - \alpha \frac{\partial}{\partial w_{22}} E = w_{22} - \alpha \frac{\partial E}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial w_{22}} = 0.8 - (0.5) \cdot (-0.1602) \cdot 0.2305 \cdot 0.5175 = 0.8096$
  - $w_{23} := w_{23} - \alpha \frac{\partial}{\partial w_{23}} E = w_{23} - \alpha \frac{\partial E}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial o_{22}} \cdot \frac{\partial o_{22}}{\partial w_{23}} = 0.5 - (0.5) \cdot 0.0041 \cdot 0.2392 \cdot 0.5349 = 0.4997$
  - $w_{24} := w_{24} - \alpha \frac{\partial}{\partial w_{24}} E = w_{24} - \alpha \frac{\partial E}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial o_{22}} \cdot \frac{\partial o_{22}}{\partial w_{24}} = 0.3 - (0.5) \cdot 0.0041 \cdot 0.2392 \cdot 0.5175 = 0.2997$

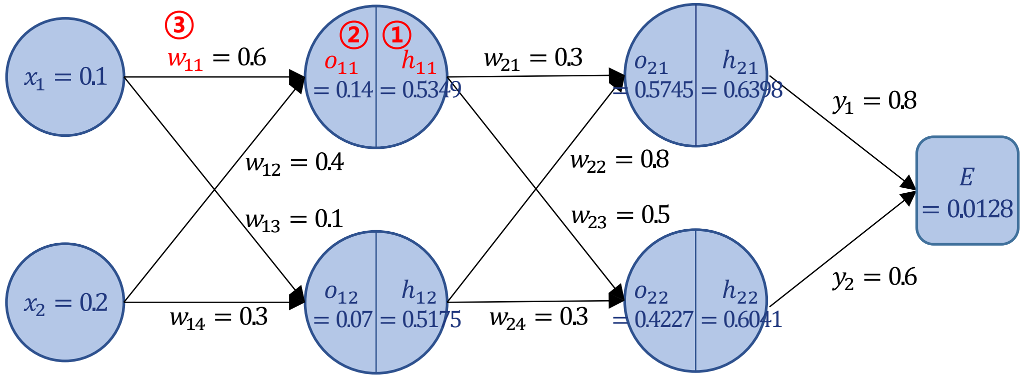


SNUH

## 1. 인공 신경망 Neural Network

| 4-5-20 | Backpropagation |
|---|---|

- **예시 – 역전파**
  - $w_{11}$에 대한 총 오차의 미분값인 $\frac{\partial E}{\partial w_{11}}$를 계산
  - 연쇄법칙에 의해, $\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}}$ 로 미분 가능

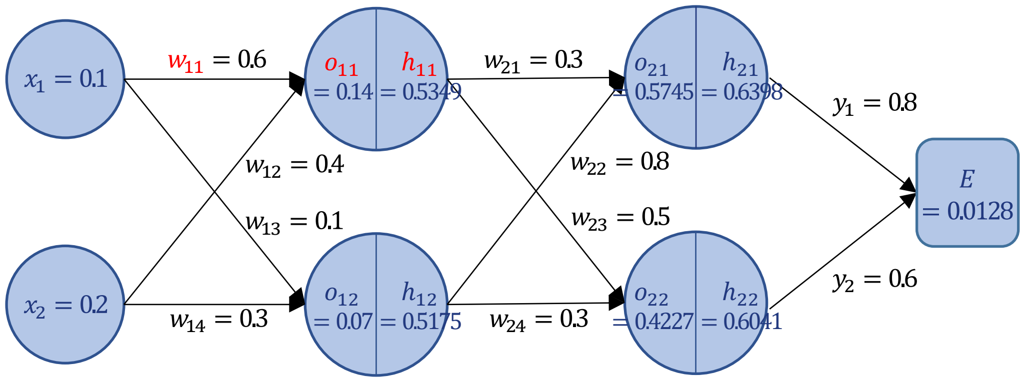


SNUH

## 1. 인공 신경망 Neural Network

**4-5-21** | **Backpropagation**

- **예시 – 역전파**




  - $\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}}$

$$\frac{\partial E}{\partial h_{11}} = \frac{\partial E_{y_1}}{\partial h_{11}} + \frac{\partial E_{y_2}}{\partial h_{11}} = \underbrace{\frac{\partial E_{y_1}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial h_{11}}}_{①} + \underbrace{\frac{\partial E_{y_2}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial o_{22}} \cdot \frac{\partial o_{22}}{\partial h_{11}}}_{②}$$

① $\frac{\partial E_{y_1}}{\partial h_{21}} \cdot \frac{\partial h_{21}}{\partial o_{21}} \cdot \frac{\partial o_{21}}{\partial h_{11}} = (y_1 - h_{21}) \cdot h_{21}(1 - h_{21}) \cdot w_{21} = 0.1602 \cdot 0.2305 \cdot 0.3 = 0.0111$

② $\frac{\partial E_{y_2}}{\partial h_{22}} \cdot \frac{\partial h_{22}}{\partial o_{22}} \cdot \frac{\partial o_{22}}{\partial h_{11}} = (y_2 - h_{22}) \cdot h_{22}(1 - h_{22}) \cdot w_{23} = (-0.0041) \cdot 0.2392 \cdot 0.5 = -0.0005$

$\frac{\partial E}{\partial h_{11}} = 0.0106$

SNUH

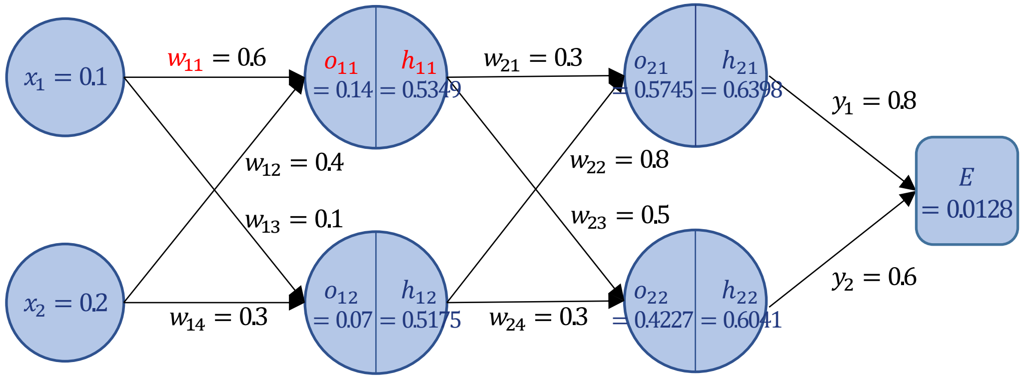
## 1. 인공 신경망 Neural Network

| 4-5-22 | Backpropagation |
| --- | --- |

### • 예시 – 역전파

- $\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}}$

$$\frac{\partial h_{11}}{\partial o_{11}} = h_{11}(1 - h_{11}) = 0.5349(1 - 0.5349) = 0.2488$$



SNUH

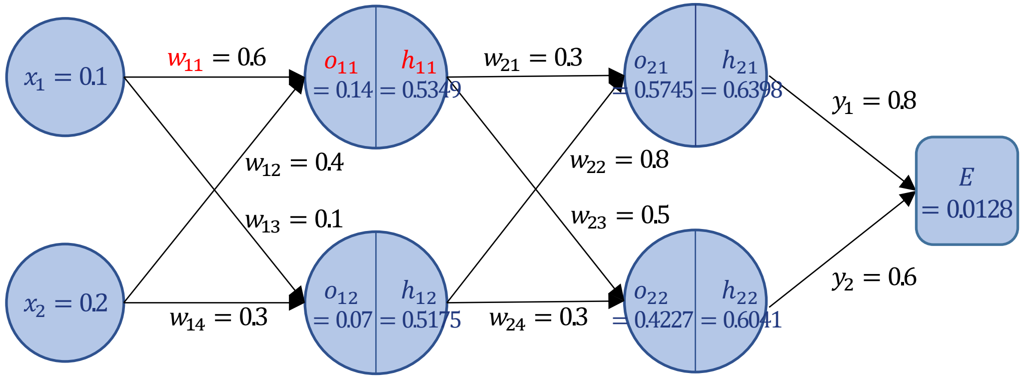
## 1. 인공 신경망 Neural Network

4-5-23 | Backpropagation

### • 예시 – 역전파

- $\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}}$

$$\frac{\partial o_{11}}{\partial w_{11}} = \frac{\partial}{\partial w_{11}}(w_{11}x_1 + w_{12}x_2) = x_1 = 0.1$$



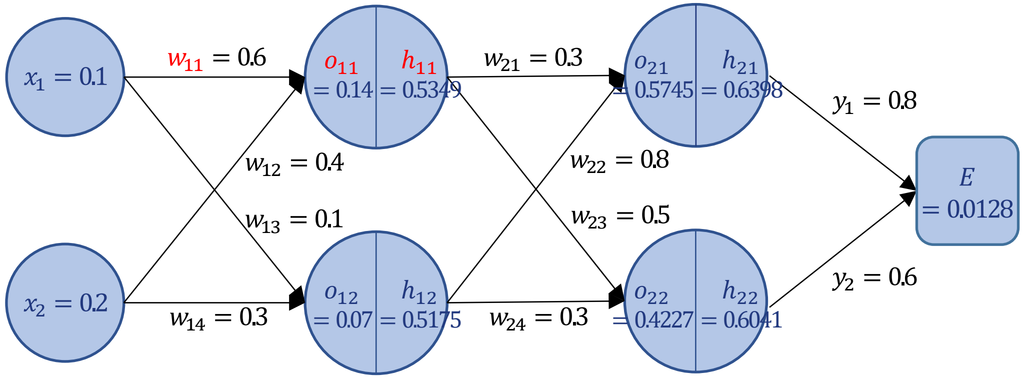
## 1. 인공 신경망 Neural Network

**4-5-24** Backpropagation

- **예시 – 역전파**
  - 따라서, 학습률 $\alpha = 0.5$ 로 가중치 $w_{11}$을 업데이트 한다면,

$$w_{11} := w_{11} - \alpha \frac{\partial}{\partial w_{11}} E$$

$$= w_{11} - \alpha \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{11}} = 0.6 - (0.5) \cdot 0.0106 \cdot 0.2488 \cdot 0.1 = 0.6 + 0.00013 = 0.59986$$

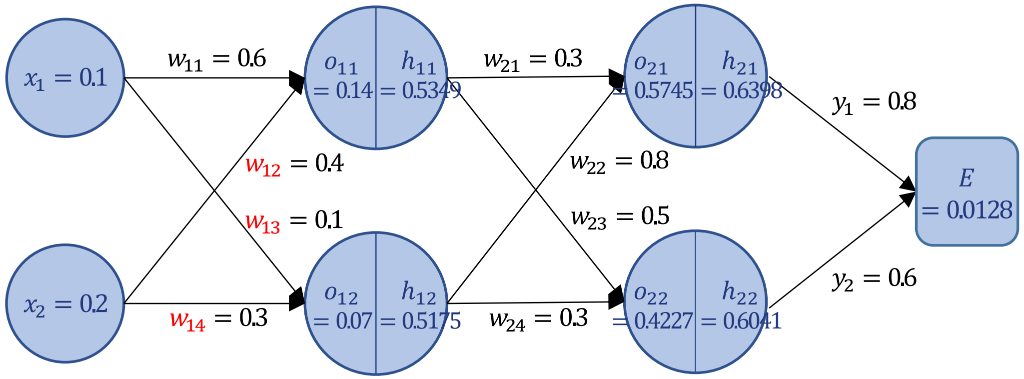


SNUH

## 1. 인공 신경망 Neural Network

| 4-5-25 | Backpropagation |
|---|---|

- **예시 – 역전파** (같은 방법으로, $w_{12}, w_{13}, w_{14}$ 를 업데이트)
  - $w_{12} := w_{12} - \alpha \frac{\partial}{\partial w_{12}} E = w_{12} - \alpha \frac{\partial E}{\partial h_{11}} \cdot \frac{\partial h_{11}}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial w_{12}} = 0.4 - (0.5) \cdot 0.0106 \cdot 0.2488 \cdot 0.2 = 0.39974$
  - $w_{13} := w_{13} - \alpha \frac{\partial}{\partial w_{13}} E = w_{13} - \alpha \frac{\partial E}{\partial h_{12}} \cdot \frac{\partial h_{12}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w_{13}} = 0.1 - (0.5) \cdot 0.0292 \cdot 0.2497 \cdot 0.1 = 0.09964$
  - $w_{14} := w_{14} - \alpha \frac{\partial}{\partial w_{14}} E = w_{14} - \alpha \frac{\partial E}{\partial h_{12}} \cdot \frac{\partial h_{12}}{\partial o_{12}} \cdot \frac{\partial o_{12}}{\partial w_{14}} = 0.3 - (0.5) \cdot 0.0292 \cdot 0.2497 \cdot 0.2 = 0.29927$



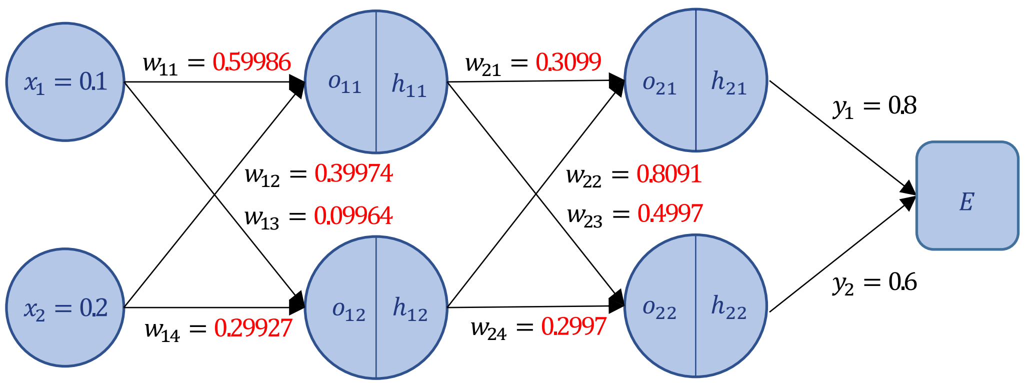
SNUH

## 1. 인공 신경망 Neural Network

4-5-26 | Backpropagation

### • 예시 – 업데이트 결과 확인

- 업데이트된 가중치를 이용하여 순전파 과정을 통해 오차를 계산하여, 오차가 감소했는지 확인
- 기존 오차 $E = 0.0128$

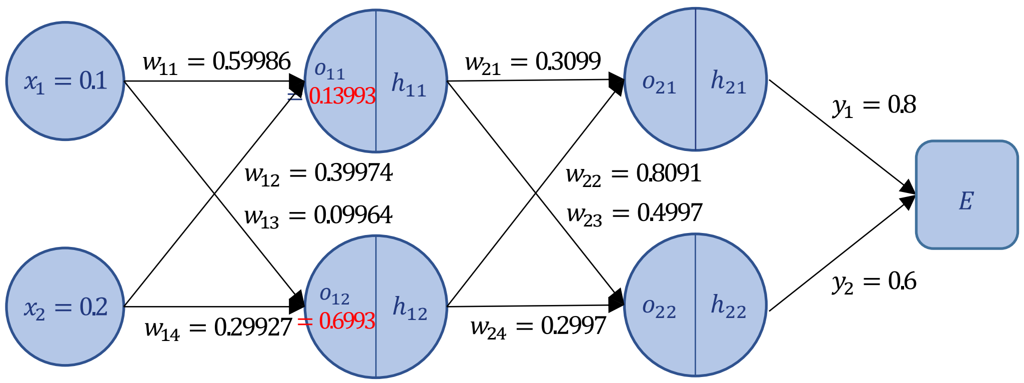


SNUH

## 1. 인공 신경망 Neural Network

4-5-27 | Backpropagation

- **예시 – 업데이트 결과 확인**
  - $o_{11} = w_{11}x_1 + w_{12}x_2 = 0.59986 \cdot 0.1 + 0.39974 \cdot 0.2 = 0.13993$
  - $o_{12} = w_{13}x_1 + w_{14}x_2 = 0.09964 \cdot 0.1 + 0.29985 \cdot 0.2 = 0.06993$

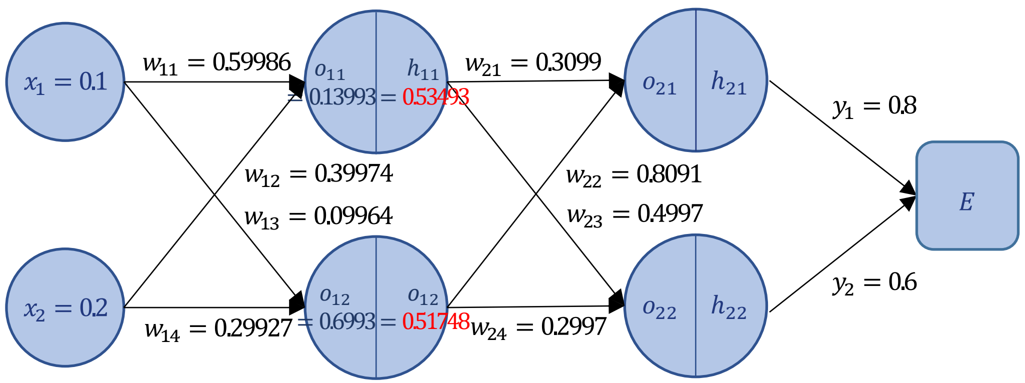


SNUH

## 1. 인공 신경망 Neural Network

4-5-28 | Backpropagation

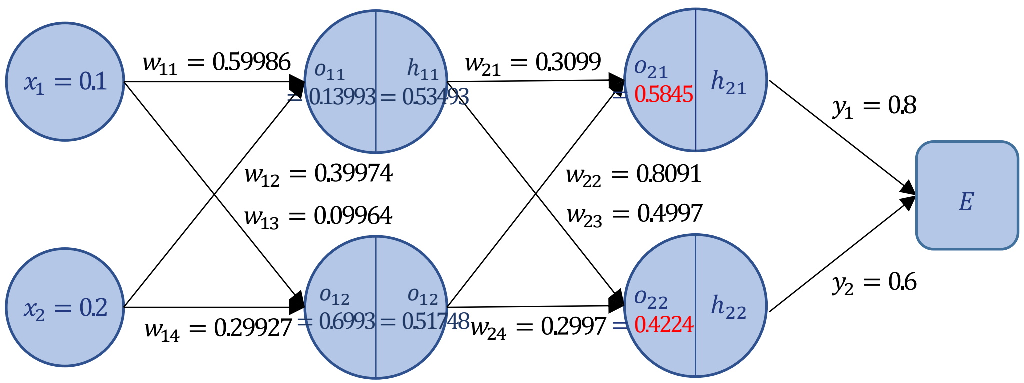
- **예시 – 업데이트 결과 확인**
  - $h_{11} = f(o_{11}) = \frac{1}{1+e^{-o_{11}}} = \frac{1}{1+e^{-0.13993}} = 0.53493$
  - $h_{12} = f(o_{12}) = \frac{1}{1+e^{-o_{12}}} = \frac{1}{1+e^{-0.06993}} = 0.51748$



## 1. 인공 신경망 Neural Network

**4-5-29** Backpropagation

- **예시 – 업데이트 결과 확인**
  - $o_{21} = w_{21}h_{11} + w_{22}h_{12} = 0.3099 \cdot 0.53493 + 0.8091 \cdot 0.51748 = 0.5845$
  - $o_{22} = w_{23}h_{11} + w_{24}h_{12} = 0.4997 \cdot 0.53493 + 0.2997 \cdot 0.51748 = 0.4224$

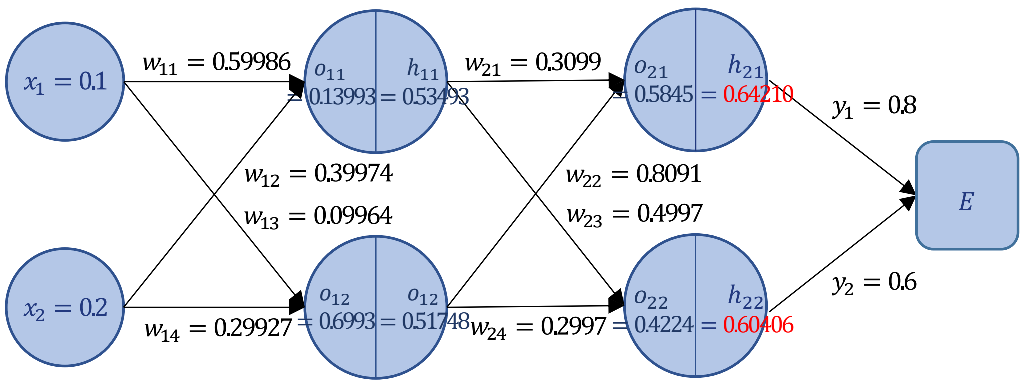


SNUH

## 1. 인공 신경망 Neural Network

4-5-30 | Backpropagation

- **예시 – 업데이트 결과 확인**
  - $h_{21} = f(o_{21}) = \frac{1}{1+e^{-o_{21}}} = \frac{1}{1+e^{-0.5845}} = 0.64210$
  - $h_{22} = f(o_{22}) = \frac{1}{1+e^{-o_{22}}} = \frac{1}{1+e^{-0.4224}} = 0.60406$



SNUH

## 1. 인공 신경망 Neural Network

| 4-5-31 | Backpropagation |
| --- | --- |

- **예시 – 업데이트 결과 확인**
  - 총 오차 $E = \frac{1}{2}\sum_i (h_{2i} - y_i)^2$
  - $E = \frac{1}{2}\{(h_{21} - y_1)^2 + (h_{22} - y_2)^2\} = \frac{1}{2}\{(0.64210 - 0.8)^2 + (0.60406 - 0.6)^2\} = 0.0125$  ∴ 기존 오차값 0.0128 보다 0.0003 감소