Neural Network

Lab Seminar - Senseable AI Lab

Byeongjoon Noh

powernoh@sch.ac.kr

SCH SOON CHUN HYANG UNIVERSITY

Most of the slides are available on Senseable AI Lab homepage: https://sailab.space

Data

Multivariate data

인자(변수) / 관측치	X_1	…	X_i	…	X_p	Y
N_1	X_{11}	…	X_{i1}	…	X_{1p}	Y_1
…	…	…	…	…	…	…
N_i	X_{i1}	…	X_{ii}	…	X_{ip}	Y_i
…	…	…	…	…	…	…
N_n	X_{n1}	…	X_{ni}	…	X_{np}	Y_n

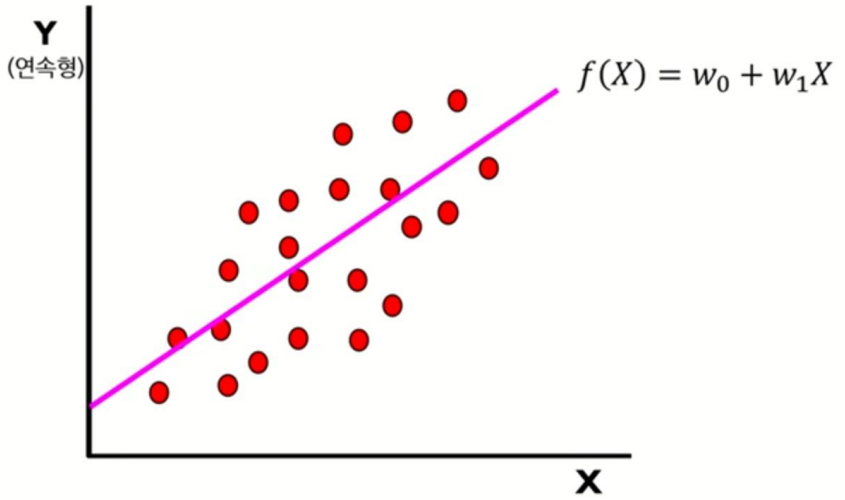
Linear regression model

$$Y = f(X) = w_0X_0 + w_1X_1 + \cdots + w_nX_n = \sum_{i=0}^{n} w_iX \qquad (X_0=1)$$

- 무수히 많은 β의 집합(파라미터) 중 단 하나의 **파라미터 조합** 찾는 것 ➔ cost function을 최소화하도록

$$\min_{\beta} \sum_{i=1}^{n} \left(Y_i - \sum_{i=0}^{n} w_iX_i \right)^2$$

- Algorithm: Least square estimation, gradient descent, etc.



Logistic regression model

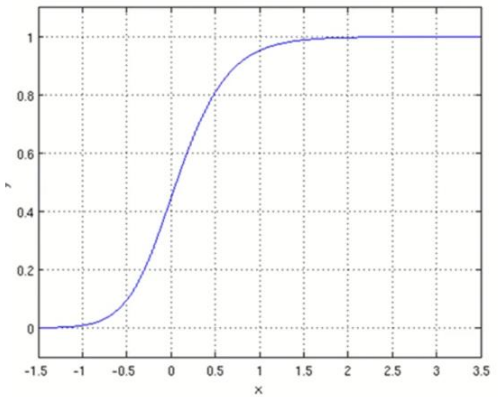
$$Y = f(X) = \frac{1}{1 + e^{-(w_0X_0 + w_1X_1 + \cdots + w_nX_n)}} = \frac{1}{1 + e^{-\left(\sum_{i=0}^{n} w_iX\right)}} \qquad (X_0=1)$$

- 무수히 많은 β의 집합(파라미터) 중 단 하나의 **파라미터 조합** 찾는 것 ➔ cost function을 최소화하도록

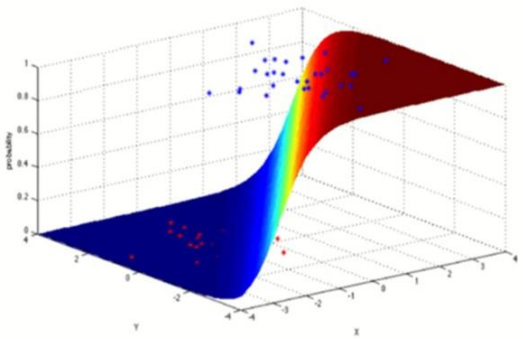
$$\min_{\beta} \sum_{i=1}^{n} \left(Y_i - \sum_{i=0}^{n} \beta_i x_i \right)^2$$

- Algorithm: Conjugate gradient method, etc.

$f(X) = \frac{1}{1+e^{-(w_0+w_1X_1)}}$

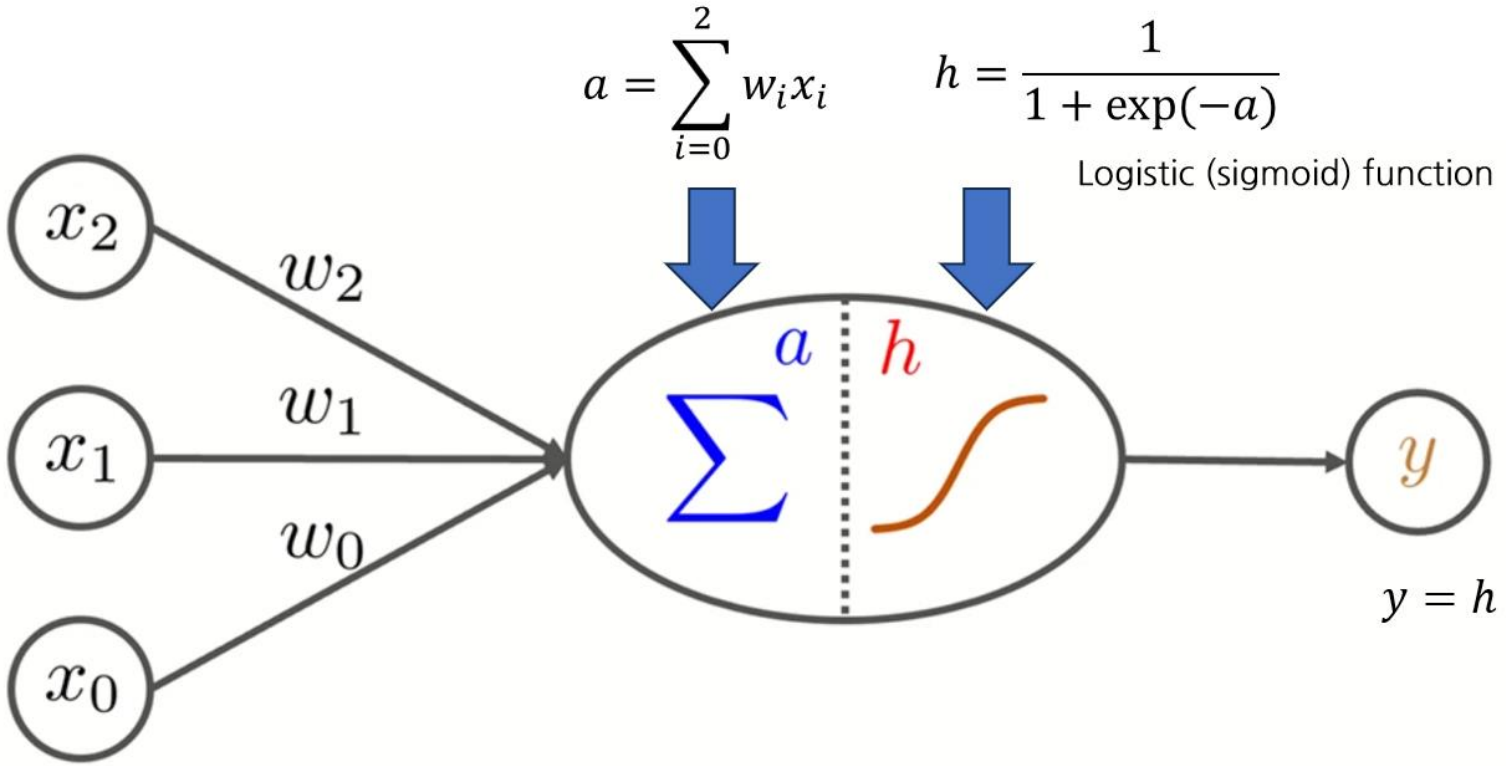


$f(X) = \frac{1}{1+e^{-(w_0+w_1X_1+w_2X_2)}}$



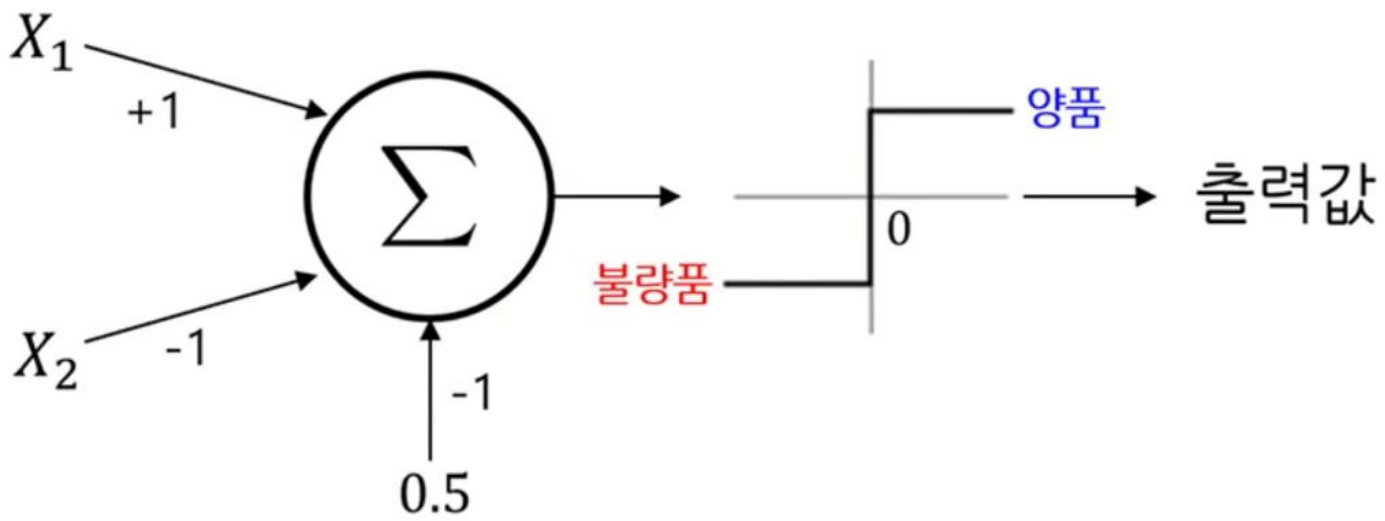
Logistic regression model

- 1) 입력변수의 선형 결합
- 2) 선형결합 값의 비선형 변환 (Nonlinear transformation)



Perceptron

- Single layer perceptron
  - 초기 뉴럴네트워크 (1957)
  - 입력값의 선형결합 값을 구하고 그 값이 0보다 큰지 여부로 분류

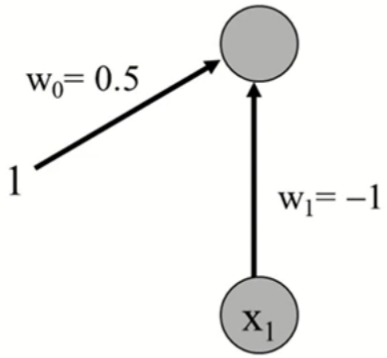


$X_1 - X_2 - 0.5 > 0$ ➔ 양품

$X_1 - X_2 - 0.5 \leq 0$ ➔ 불량품

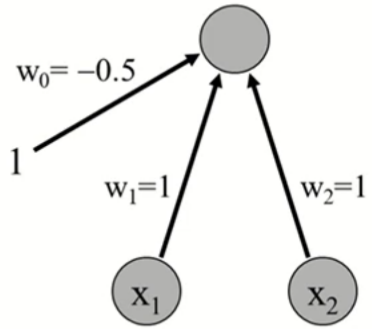
Perceptron

input x1	output
0	1
1	0



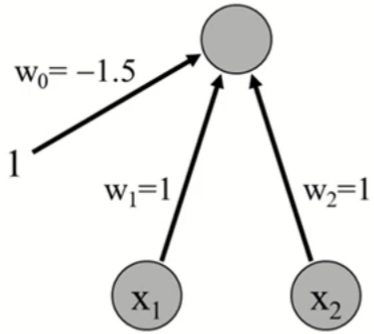
변환기

input x1	input x2	output
0	0	0
0	1	1
1	0	1
1	1	1



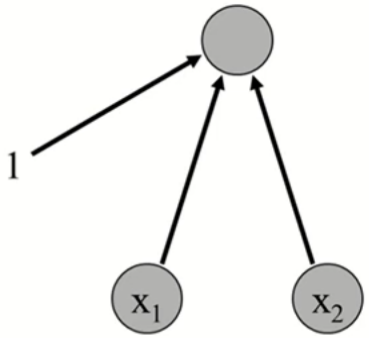
Boolean OR

input x1	input x2	output
0	0	0
0	1	0
1	0	0
1	1	1



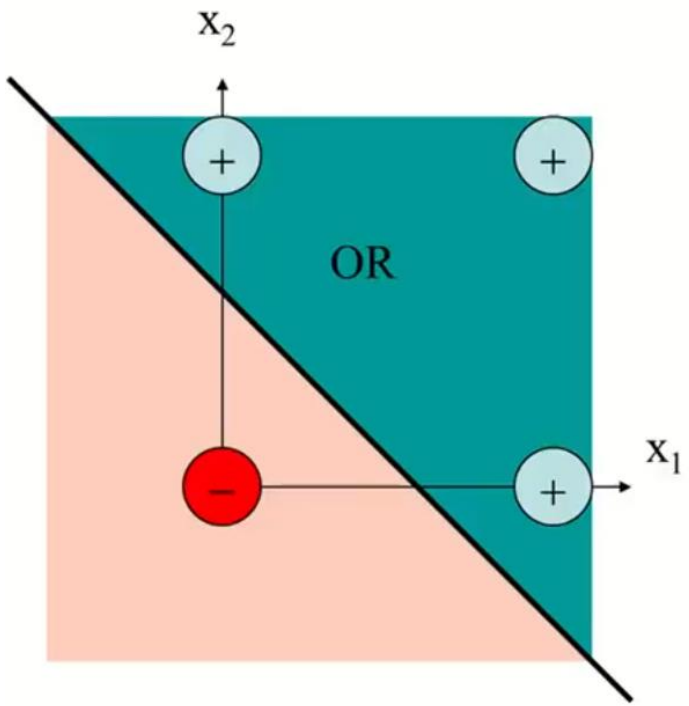
Boolean AND

input x1	input x2	output
0	0	0
0	1	1
1	0	1
1	1	0

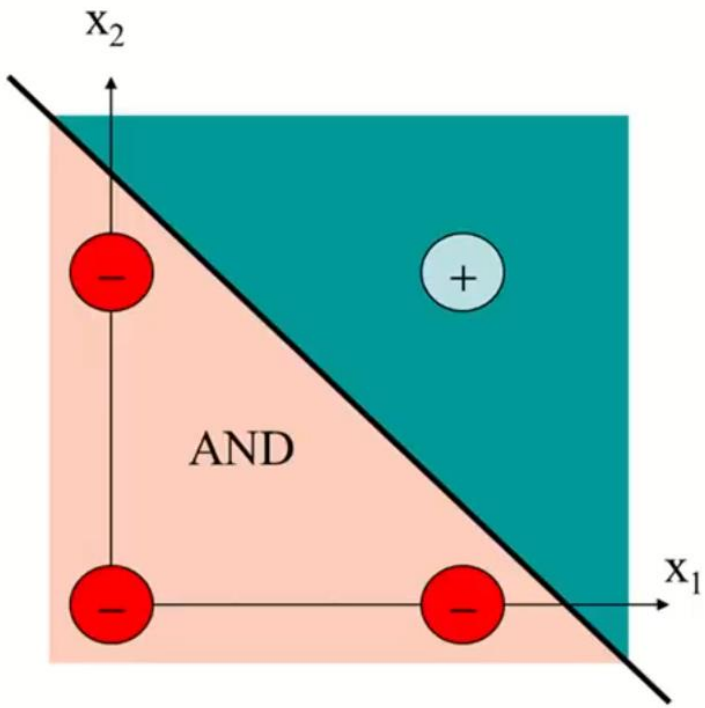


Boolean XOR
Value = 1 iff x1 ≠ x2

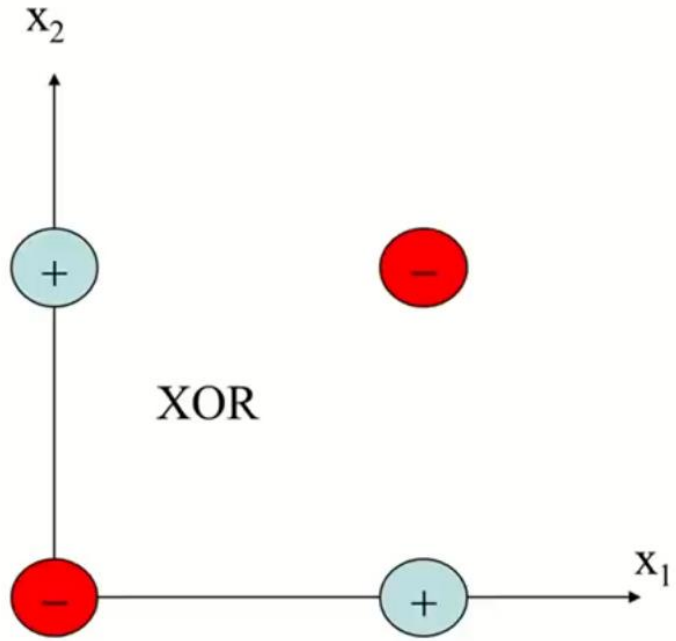
Perceptron



Boolean OR



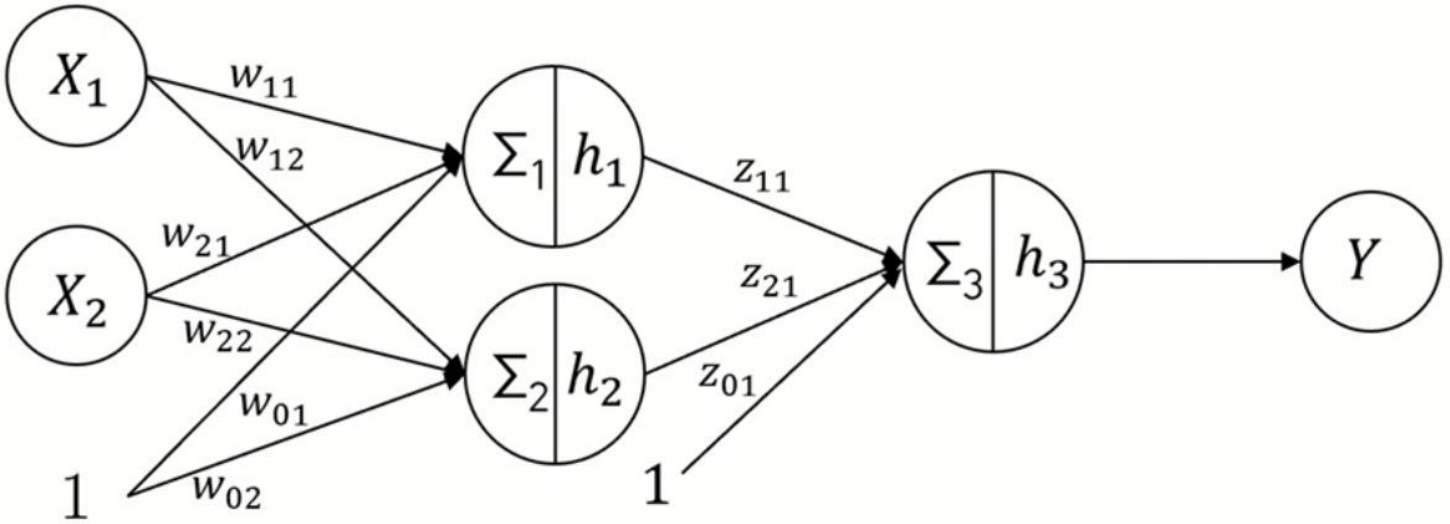
Boolean AND



Boolean XOR

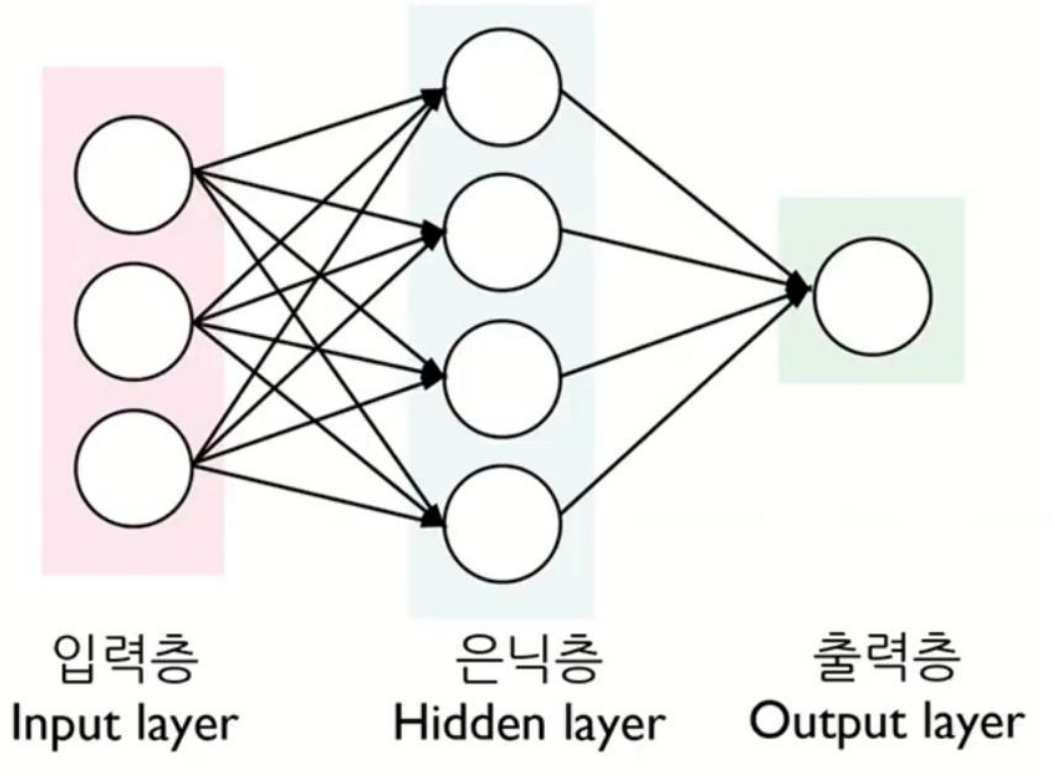
Multi-layer perceptron

- 두 개의 perceptron을 결합



Multi-layer perceptron

- 두 개의 perceptron을 결합



입력층

- 입력변수의 값이 들어오는 곳
- 입력 변수의 수 == 입력 노드의 수

은닉층

- 은닉층에는 다수 노드 포함 가능
- 다수의 은닉층을 형성 가능

출력층

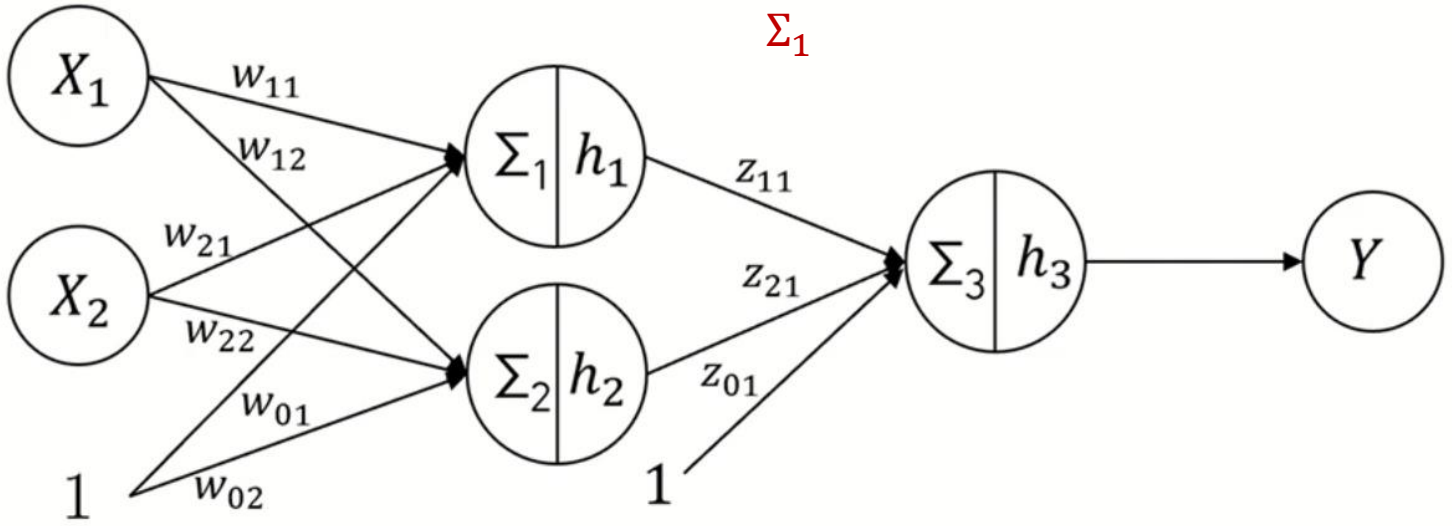
- (범주형) 출력노드의 수 = 출력변수의 범주 개수
- (연속형) 출력노드의 수 = 출력변수의 개수

Multi-layer perceptron

- 두 개의 perceptron을 결합

$$h = \frac{1}{1 + \exp(-a)}$$

$$h(\Sigma_1) = h_1 = \frac{1}{1 + e^{-(w_{01} + w_{11}X_1 + w_{21}X_2)}}$$



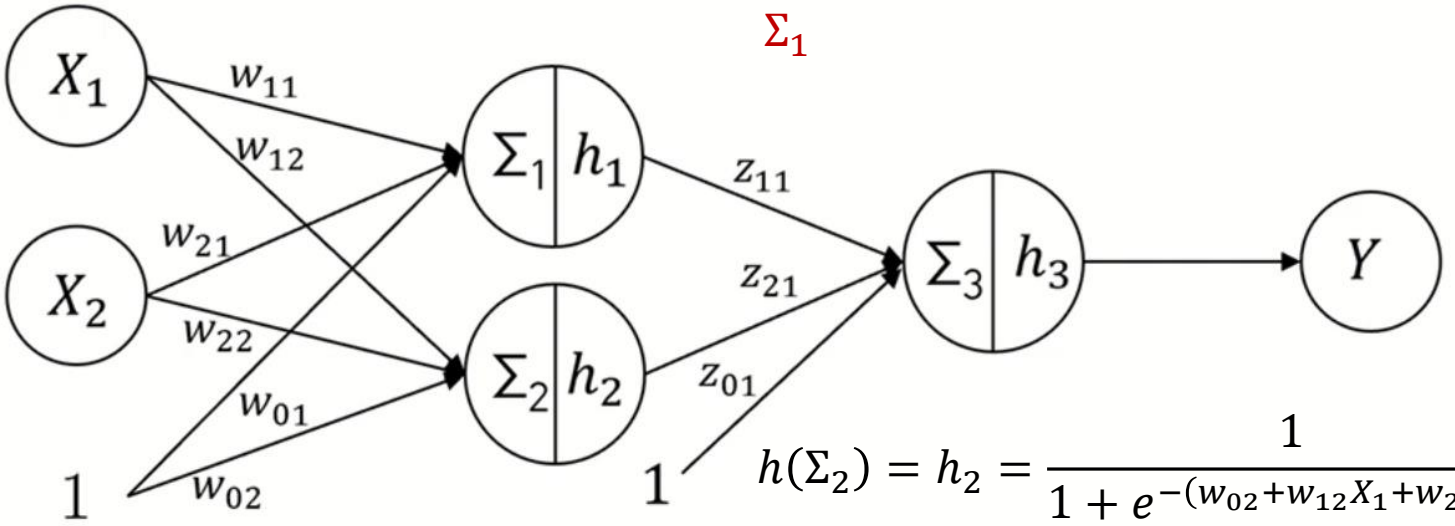
Multi-layer perceptron

- 두 개의 perceptron을 결합

$$h = \frac{1}{1 + \exp(-a)}$$

$$h(\Sigma_1) = h_1 = \frac{1}{1 + e^{-(w_{01} + w_{11}X_1 + w_{21}X_2)}}$$

Σ_1



$$h(\Sigma_2) = h_2 = \frac{1}{1 + e^{-(w_{02} + w_{12}X_1 + w_{22}X_2)}}$$

Σ_2

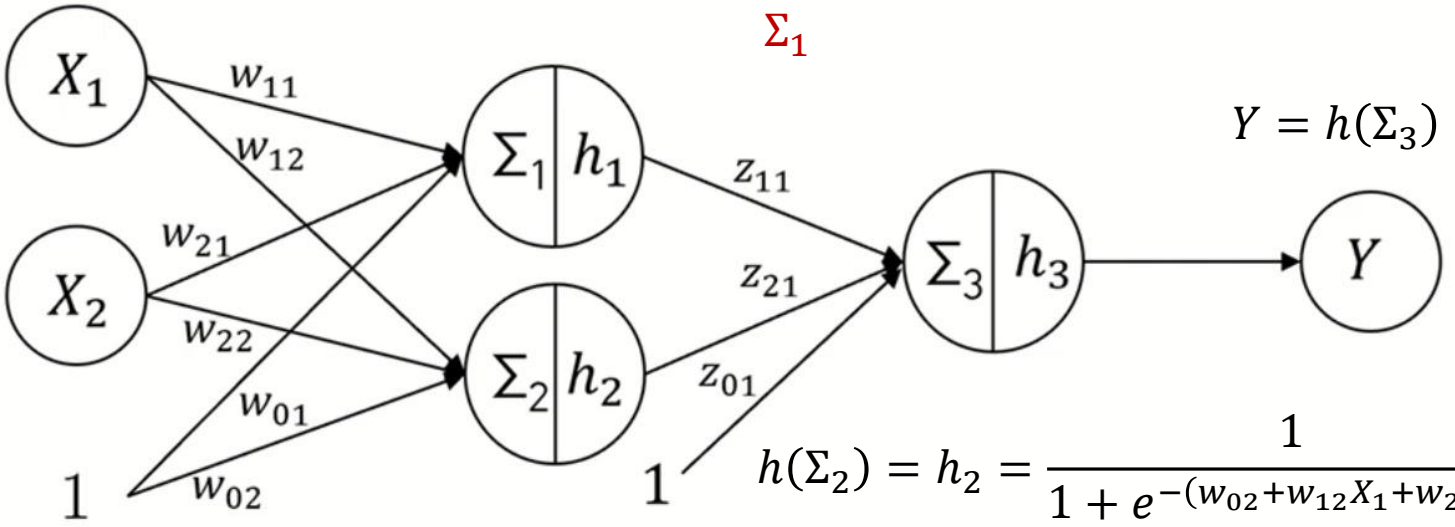
Multi-layer perceptron

- 두 개의 perceptron을 결합

$$h = \frac{1}{1 + \exp(-a)}$$

$$h(\Sigma_1) = h_1 = \frac{1}{1 + e^{-(w_{01} + w_{11}X_1 + w_{21}X_2)}}$$

Σ_1



$$Y = h(\Sigma_3) = h_3 = \frac{1}{1 + e^{-(z_{01} + z_{11}h_1 + z_{21}h_2)}}$$

Σ_3

$$h(\Sigma_2) = h_2 = \frac{1}{1 + e^{-(w_{02} + w_{12}X_1 + w_{22}X_2)}}$$

Σ_2

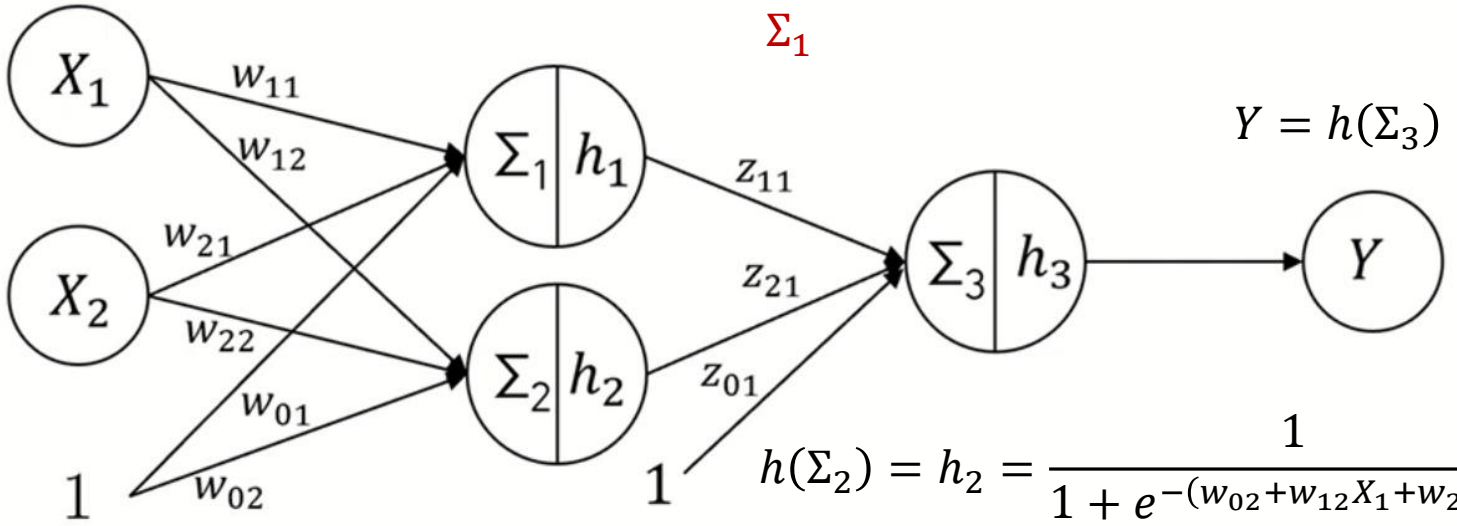
Multi-layer perceptron

- 두 개의 perceptron을 결합

$$h = \frac{1}{1+\exp(-a)}$$

$$h(\Sigma_1) = h_1 = \frac{1}{1+e^{-(w_{01}+w_{11}X_1+w_{21}X_2)}}$$

Σ_1



$$Y = h(\Sigma_3) = h_3 = \frac{1}{1+e^{-(z_{01}+z_{11}h_1+z_{21}h_2)}}$$

Σ_3

$$h(\Sigma_2) = h_2 = \frac{1}{1+e^{-(w_{02}+w_{12}X_1+w_{22}X_2)}}$$

Σ_2

$$Y = \frac{1}{1+\exp\left(-\left(z_{01}+z_{11}\left(\frac{1}{1+e^{-(w_{01}+w_{11}X_1+w_{21}X_2)}}\right)\right)+z_{21}\left(\frac{1}{1+e^{-(w_{02}+w_{12}X_1+w_{22}X_2)}}\right)\right)}$$

Multi-layer perceptron

- Linear regression model

$$f(X) = w_0 + w_1 X_1 + \cdots + w_p X_p$$

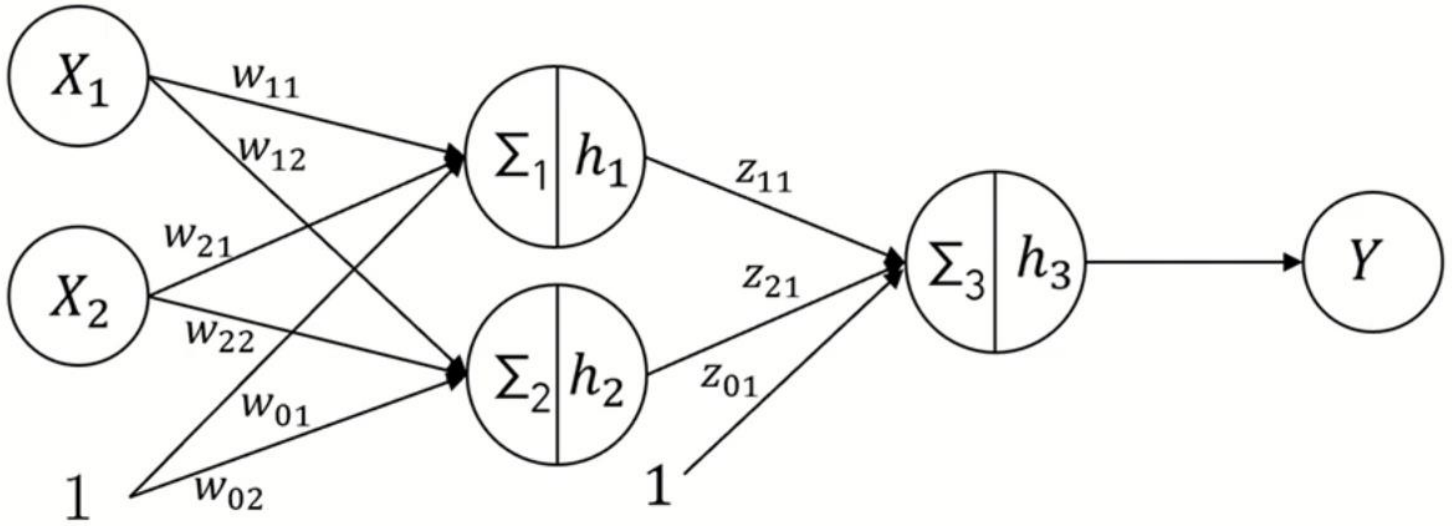
- Logistic regression model

$$f(X) = \frac{1}{1 + e^{-(w_0 + w_1 X_1 + \cdots + w_p X_p)}}$$

- Multi-layer perceptron (Neural network) model

$$f(X) = \frac{1}{1 + \exp\left(-\left(z_{01} + z_{11}\left(\frac{1}{1 + e^{-(w_{01} + w_{11} X_1 + w_{21} X_2)}}\right)\right) + z_{21}\left(\frac{1}{1 + e^{-(w_{02} + w_{12} X_1 + w_{22} X_2)}}\right)\right)}$$

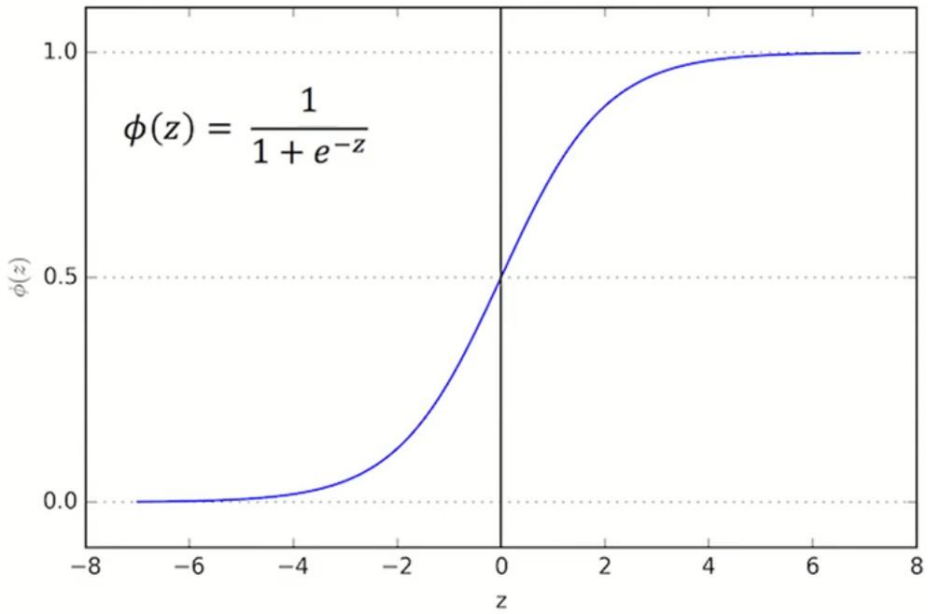
Hyperparameter



- 파라미터 (Parameters)
  - 층 간 노드를 연결하는 가중치 ($w_{11}, w_{12}, \dots, z_{21}$) ➔ 알고리즘으로 결정

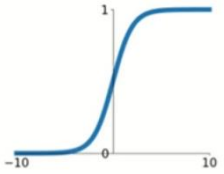
- 하이퍼파라미터 (Hyperparameters)
  - 은닉층 개수, 은닉노드 개수, activation function ➔ 사용자가 임의로 결정

Activation function



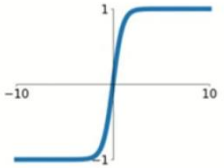
Sigmoid

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



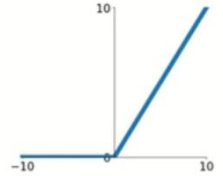
tanh

$\tanh(x)$



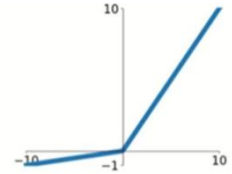
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

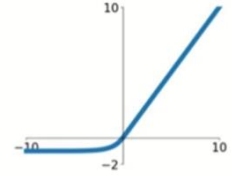


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



- Logistic function, sigmoid function, squashing function (Large input → small output)
- Output 범위: 0~1
- Input값에 대해 단조증가 (혹은 단조감소) 함수
- 미분결과를 output의 함수로 표현 가능 (gradient learning method에 유용하게 사용)

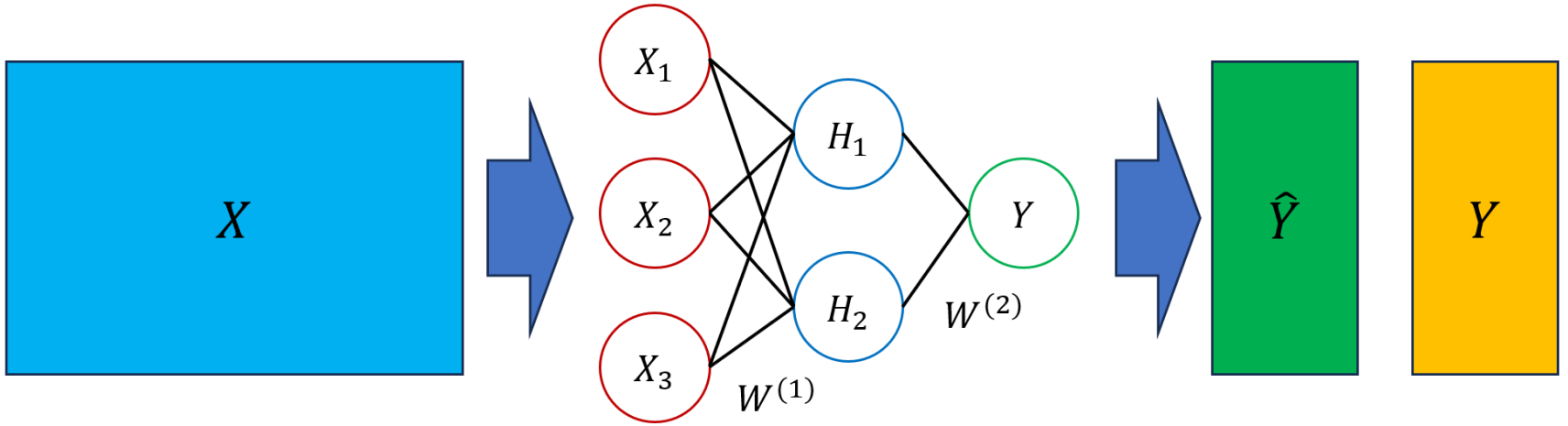
$$\frac{d\phi(z)}{dz} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = \phi(z)\big(1-\phi(z)\big)$$

Goal of neural network

- 모델로부터 나온 $f(X;w)$ 값 (== $\hat{Y}$) 과 실제 Y 값의 차이를 최소로하는 가중치 w를 찾아라

$$\underset{w}{\operatorname{argmin}} \sum_{i} L\big(Y, f(X;w)\big) \quad \leftarrow \text{Cost (loss function)}$$

  - Algorithm: Gradient descent method (경사하강법)



순전파 (Feed forward) 과정

오류 역전파 (Error Backpropagation) 과정 → Gradient 계산 방법

Cost (loss function)

- Cost function example – Numerical prediction problem
- Regression: mean squared error (MSE)
  - $L = \frac{1}{n}\sum_{i=1}^{n}(o_i - t_i)^2$
  - o_i: 예측값, t_i: 실제값
- 미분가능해야 함

예측값	실제값
11	10
19	20
30	30
43	40
47	50

$$L = \frac{1}{5}\sum_{i=15}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{5}[(10-11)^2 + (20-19)^2 + (30-30)^2 + (40-43)^2 + (50-47)^2] = 4$$

Cost (loss function)

- Cost function example – classification (softmax) problem
- Classification: cross entropy
  - $L = -\sum_i t_i \log p_i$
  - p_i: 예측값, t_i: 실제값
- 미분가능해야 함

예측값

Class 1	Class 2	Class 3
0.3	0.2	0.5
0.1	0.8	0.1
0.6	0.2	0.2
0.1	0.5	0.4

실제값

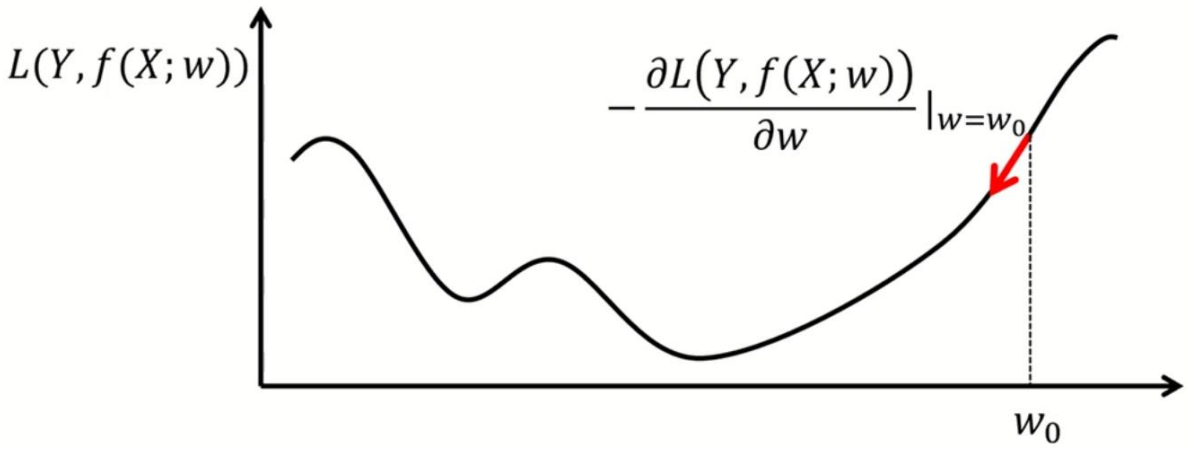
Class 1	Class 2	Class 3
1	0	0
0	1	0
1	0	0
0	0	1

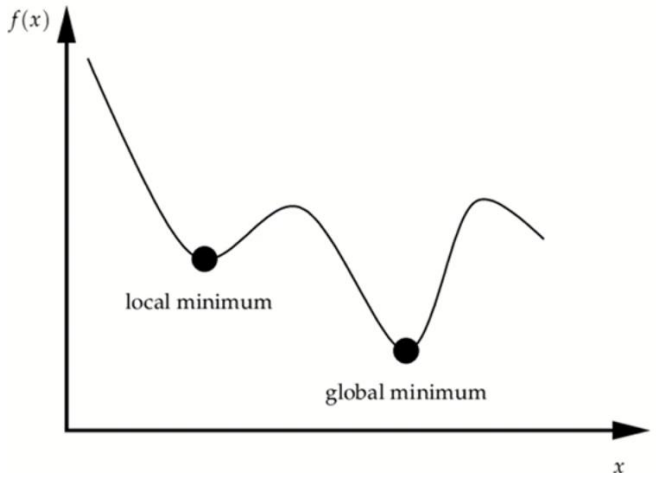
$$L = -\sum_i t_i \log p_i$$

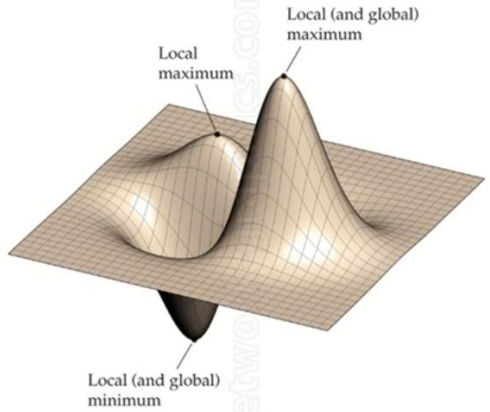
$$= -[\ln(0.3) + \ln(0.8) + \ln(0.6) + \ln(0.4)] = 2.85$$

Gradient descent method

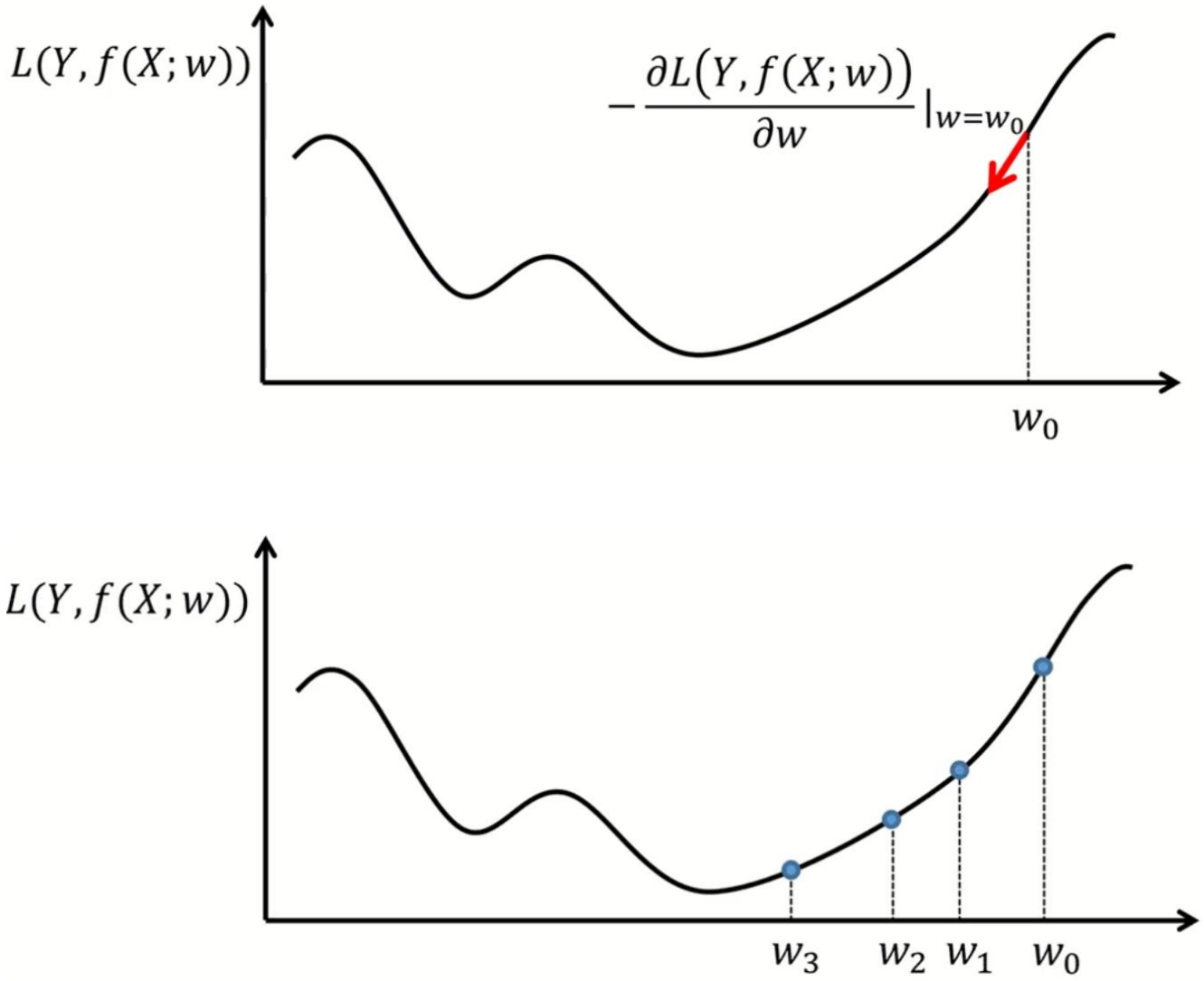
- Gradient: 함수의 기울기
- Optimization: 함수의 최솟값 혹은 최댓값을 찾는 과정
- 최솟값들 중 가장 작은 최솟값: 전역 최솟값 (global minimum)
- 지역적인 최솟값: 지역 최솟값 (local minimum)







Gradient descent method

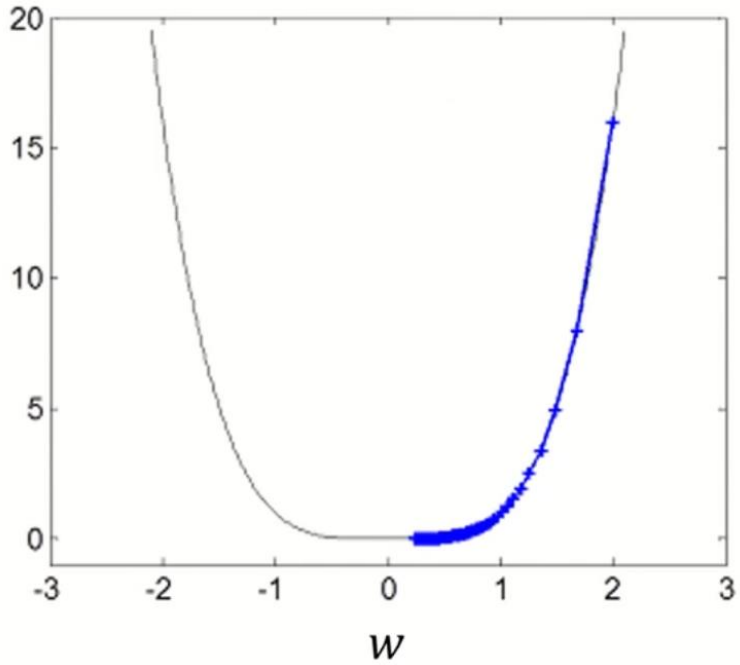


Gradient descent method

$$w_{\tau+1} = w_\tau - \alpha \cdot L'(w_\tau)$$

- $0 < \alpha < 1$: Learning rate (학습률)
  - 좀 더 섬세하고 촘촘히? ➔ Small α
  - 좀 더 빠르게? ➔ Large α

$L(w) = w^4$



$w_0 = 2 \qquad \alpha = 0.01$

$w_1 = w_0 - \alpha L'(w_0) = 1.6800$

$w_2 = w_1 - \alpha L'(w_1) = 1.4903$

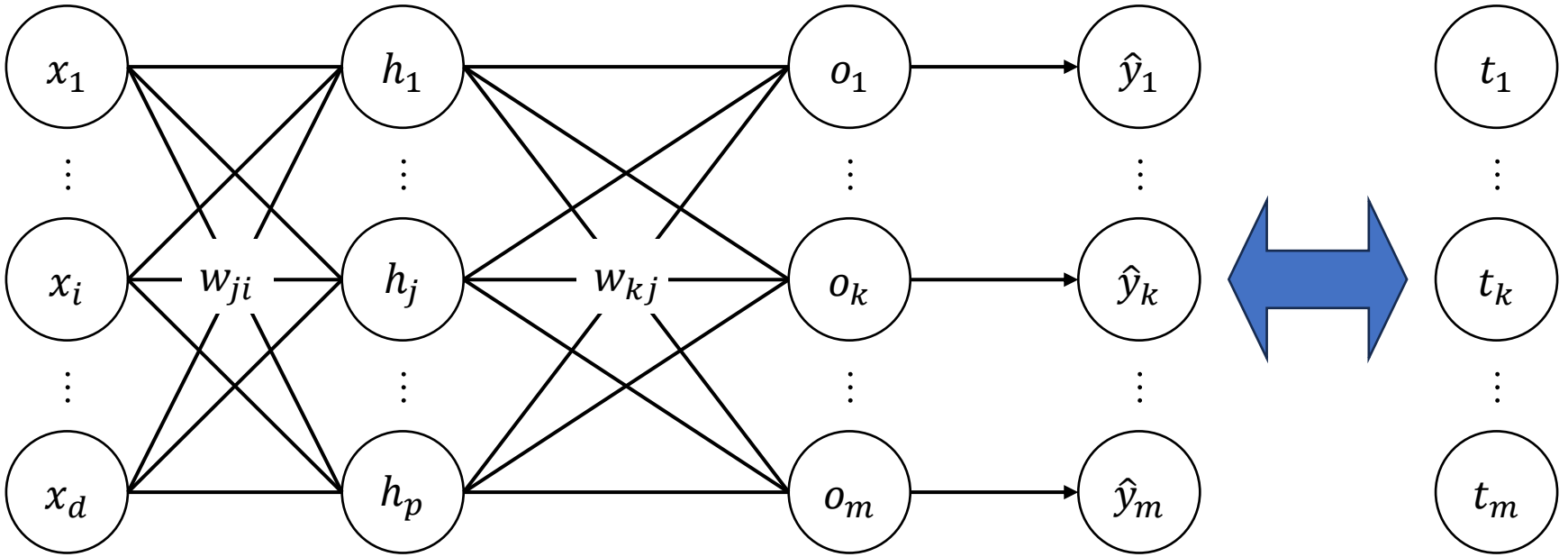
$w_3 = w_2 - \alpha L'(w_2) = 1.3579$

…

$w_{200} = w_{199} - \alpha L'(w_{199}) = 0.2461$

$\alpha = 0.02$ 라면?

NN training mechanism



입력 X, 출력 Y, 예측 O

$$D_1 = (x_{11}, x_{12}, \dots, x_{1d}, t_{11}, t_{12}, \dots, t_{1m}) \qquad (o_{11}, o_{12}, \dots, o_{1m})$$

$$D_2 = (x_{21}, x_{22}, \dots, x_{2d}, t_{21}, t_{22}, \dots, t_{2m}) \qquad (o_{21}, o_{22}, \dots, o_{2m})$$

...

$$D_N = (x_{N1}, x_{N2}, \dots, x_{Nd}, t_{N1}, t_{N2}, \dots, t_{Nm}) \qquad (o_{N1}, o_{N2}, \dots, o_{Nm})$$

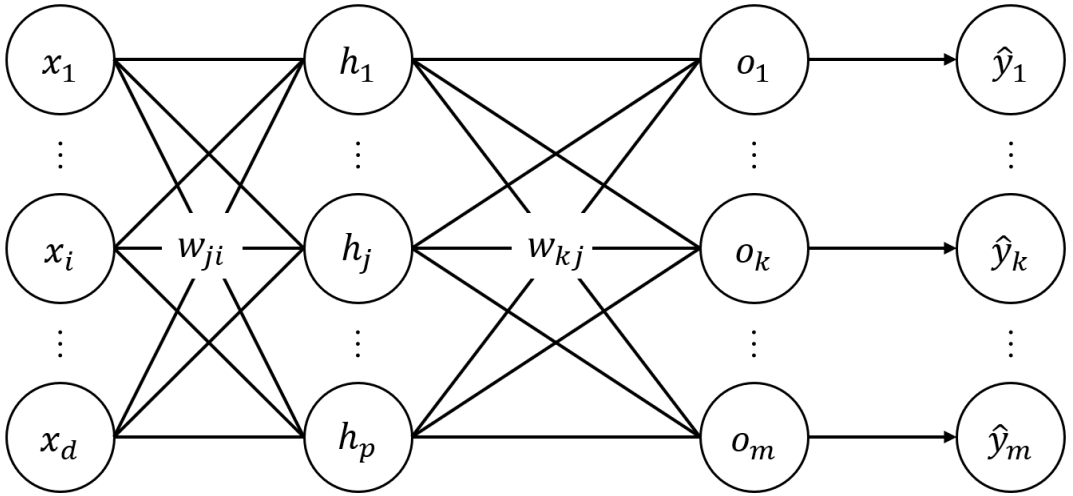
NN training mechanism

- 목표: 비용함수 $E(w)$를 최소로 하는 w를 찾자 (*Y가 연속형)

$$E(w) = \sum_{n=1}^{N} E_n(w) \qquad E_n(w) = \frac{1}{2}\sum_{n=1}^{m}(t_{nk} - o_{nk})^2$$

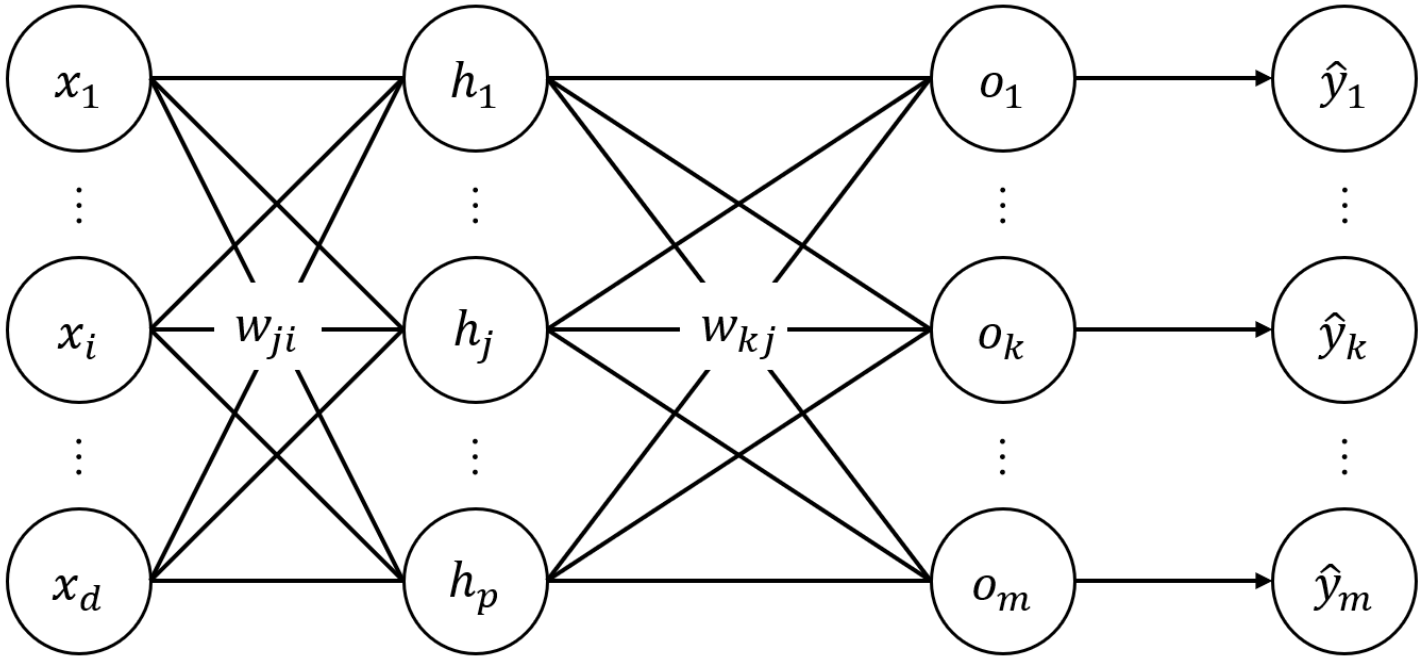
- w에 대해서 미분 가능 ➔ $\frac{\partial E_n}{\partial w_{kj}}$와 $\frac{\partial E_n}{\partial w_{ji}}$를 구할 수 있다

출력층-은닉층 은닉층-입력층



NN training mechanism

$$o_k = \frac{1}{1 + \exp\left(-\left(w_{k0} + \sum_{i=1}^{p} w_{kj} h_j\right)\right)}$$



$$h_j = \frac{1}{1 + \exp\left(-\left(w_{j0} + \sum_{i=1}^{d} w_{ji} x_i\right)\right)}$$

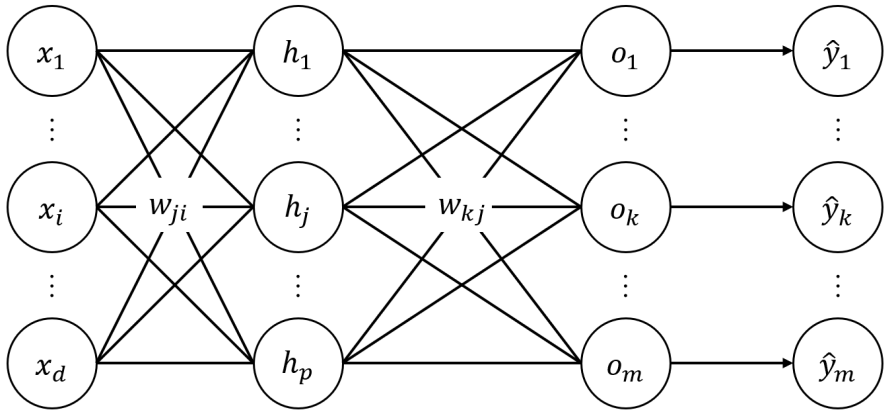
NN training mechanism

$$E_n(w) = \frac{1}{2}\sum_{n=1}^{m}(t_{nk} - o_{nk})^2 \qquad o_k = \frac{1}{1 + \exp\left(-\left(w_{k0} + \sum_{i=1}^{p} w_{kj} h_j\right)\right)}$$

$$E_n(w) = \frac{1}{2}\sum_{n=1}^{m}\left(t_k - \left(\frac{1}{1 + \exp\left(-\left(w_{k0} + \sum_{j=1}^{p} w_{kj}\left(\frac{1}{1 + \exp\left(-\left(w_{j0} + \sum_{i=1}^{d} w_{ji} x_i\right)\right)}\right)\right)\right)}\right)\right)$$

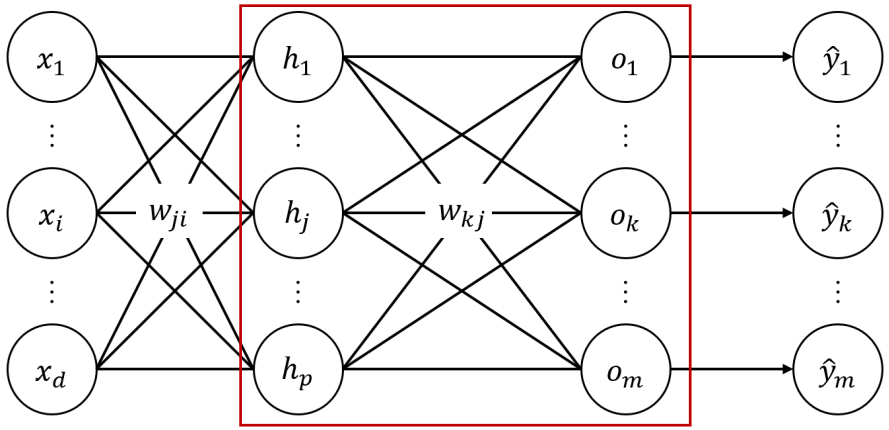
미분을 바로 하기에 너무 복잡

- **1) 출력층-은닉층 사이**
- **2) 은닉층-입력층 사이**



NN training mechanism

1) 출력층-은닉층 사이



$$net_k = h_1 w_{k1} + h_2 w_{k2} + \cdots + h_j w_{kj} + \cdots + h_p w_{kp} + w_{k0}$$

- h_j: **은닉층의 j번째 노드값**
- $o_k = sigmoid(net_k) = \frac{1}{1+\exp(-net_k)}$
- $E_n(w) = \frac{1}{2}\sum_{n=1}^{m}(t_k - o_k)^2$

경사하강법을 이용하여 w_{kj}를 계산하기 위해 $\Delta w_{kj} = -\alpha \frac{\partial E_n}{\partial w_{kj}}$

$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial E_n}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial E_n}{\partial o_k}\frac{\partial o_k}{\partial net_k} h_j$$

NN training mechanism

1) 출력층-은닉층 사이

$$\frac{\partial E_n}{\partial w_{kj}} = \frac{\partial E_n}{\partial o_k}\frac{\partial o_k}{\partial net_k}h_j$$

$$\frac{\partial E_n}{\partial o_k} = \frac{\partial}{\partial o_k}\frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2 = \frac{\partial}{\partial o_k}\frac{1}{2}(t_k - o_k)^2 = -(t_k - o_k)$$

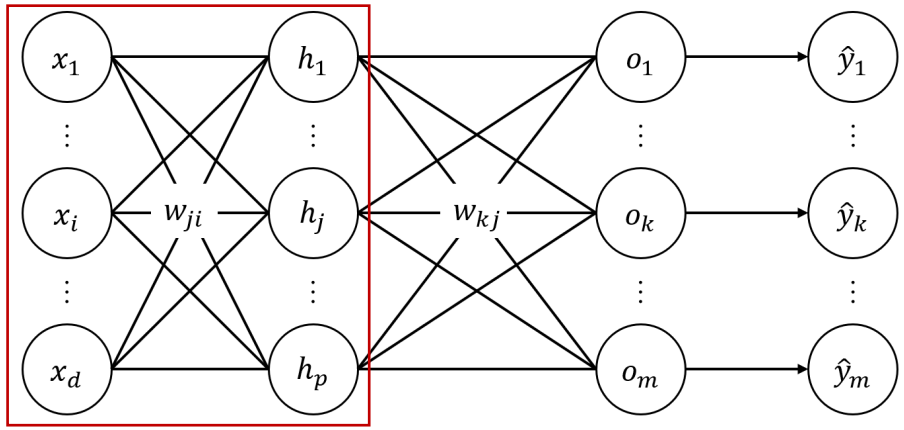
$$\frac{\partial o_k}{\partial net_k} = \frac{\partial}{\partial net_k}\frac{1}{1+\exp(-net_k)} = \frac{-(-\exp(-net_k))}{(1+\exp(-net_k))^2} = \frac{\exp(-net_k)}{(1+\exp(-net_k))^2}$$

$$= \frac{1}{1+\exp(-net_k)}\frac{\exp(-net_k)}{1+\exp(-net_k)} = o_k(1-o_k)$$

$$\Delta w_{kj} = -\alpha\frac{\partial E_n}{\partial w_{kj}} = \alpha(t_k - o_k)o_k(1-o_k)h_j$$

NN training mechanism

2) 은닉층-입력층 사이



$$net_j = x_1 w_{j1} + x_2 w_{j2} + \cdots + x_i w_{ji} + \cdots + x_d w_{jd} + w_{j0}$$

- $h_j = sigmoid(net_j) = \frac{1}{1+\exp(-net_j)}$

$$net_k = h_1 w_{k1} + h_2 w_{k2} + \cdots + h_j w_{kj} + \cdots + h_p w_{kp} + w_{k0}$$

- $o_k = sigmoid(net_k) = \frac{1}{1+\exp(-net_k)}$

$$E_n(w) = \frac{1}{2}\sum_{n=1}^{m}(t_k - o_k)^2$$

경사하강법을 이용하여 w_{ji}를 계산하기 위해 $\Delta w_{ji} = -\alpha \frac{\partial E_n}{\partial w_{ji}}$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_n}{\partial net_j} x_i$$

NN training mechanism

2) 은닉층-입력층 사이

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial net_j} x_i$$

$$\frac{\partial E_n}{\partial net_j} = \frac{\partial}{\partial net_j} \frac{1}{2} \sum_{k=1}^{m} (t_k - o_k)^2 = \frac{1}{2} \sum_{k=1}^{m} \frac{\partial}{\partial net_j} (t_k - o_k)^2$$

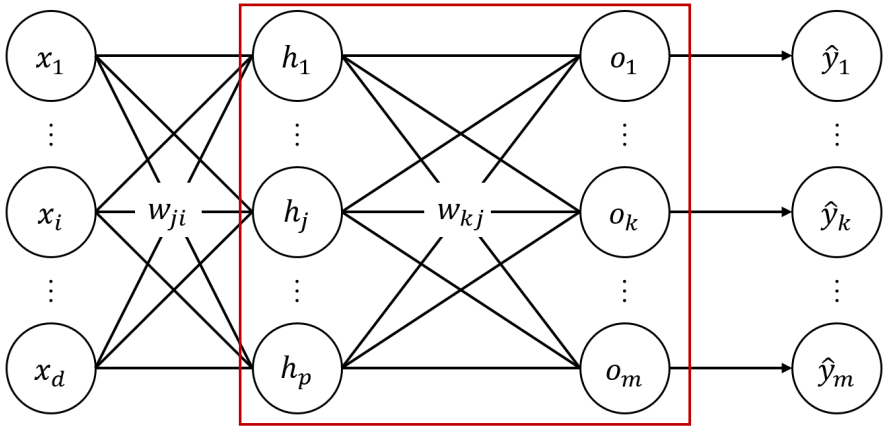
$$= \frac{1}{2} \sum_{k=1}^{m} \frac{\partial h_j}{\partial net_j} \frac{\partial net_k}{\partial h_j} \frac{\partial o_k}{\partial net_k} \frac{\partial (t_k - o_k)^2}{\partial o_k}$$

$$\frac{\partial h_j}{\partial net_j} = h_j(1 - h_j) \qquad \frac{\partial net_k}{\partial h_j} = w_{kj} \qquad \frac{\partial o_k}{\partial net_k} = o_k(1 - o_k) \qquad \frac{\partial (t_k - o_k)^2}{\partial o_k} = -2(t_k - o_k)$$

$$\Delta w_{jh} = -\alpha \frac{\partial E_n}{\partial w_{ji}} = \alpha x_i h_j(1 - h_j) \sum_{k=1}^{m} w_{kj}\, o_k(1 - o_k)(t_k - o_k)$$

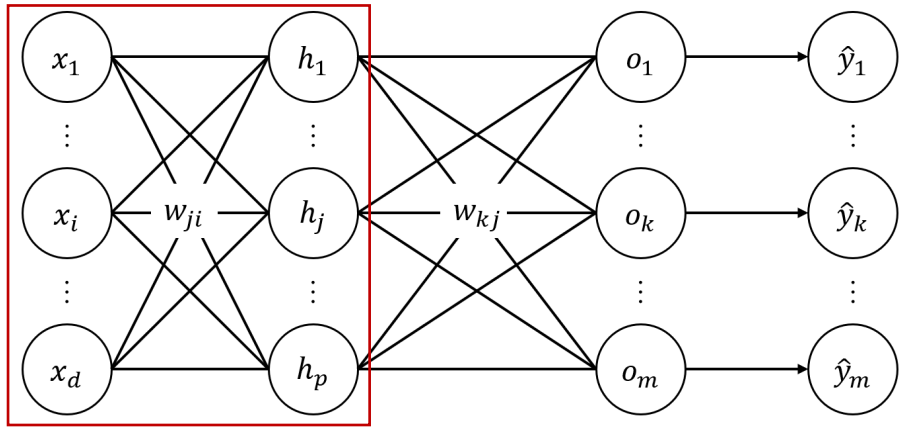
NN training mechanism

1) 출력층-은닉층 사이



$$\Delta w_{kj} = -\alpha \frac{\partial E_n}{\partial w_{kj}} = \alpha(t_k - o_k)o_k(1 - o_k)h_j$$

2) 은닉층-입력층 사이



$$\Delta w_{jh} = -\alpha \frac{\partial E_n}{\partial w_{ji}} = \alpha x_i h_j(1 - h_j) \sum_{k=1}^{m} w_{kj}\, o_k(1 - o_k)(t_k - o_k)$$

NN training mechanism

- Step 1: 모든 가중치 w를 임의로 생성
- Step 2: 입력변수 값과 입력층과 은닉층 사이의 w값을 이용하여 은닉노드의 값을 계산

  ➔ 선형 결합 후 activation

$$h_j = f\left(w_{j0} + \sum_{i=1}^{d} w_{ji} x_i\right)$$

- Step 3: 은닉노드의 값과 은닉층과 출력층 사이의 w값을 이용하여 출력노드의 값을 계산

  ➔ 선형 결합 후 activation

$$o_k = f\left(w_{k0} + \sum_{j=1}^{p} w_{kj} h_j\right)$$

- Step 4: 계산된 출력노드의 값과 실제 출력변수의 값의 차이를 줄일 수 있도록 은닉층-출력층 사이의 w값 업데이트 $w_{kj} = w_{kj} - \alpha \frac{\partial E_n}{\partial w_{kj}}$
- Step 5: 계산된 출력노드의 값과 실제 출력변수의 값의 차이를 줄일 수 있도록 입력층-은닉층 사이의 w값 업데이트 $w_{ji} = w_{ji} - \alpha \frac{\partial E_n}{\partial w_{ji}}$
- Step 6: 에러가 충분히 줄어 들 때 까지 Step2 ~ Step 6반복

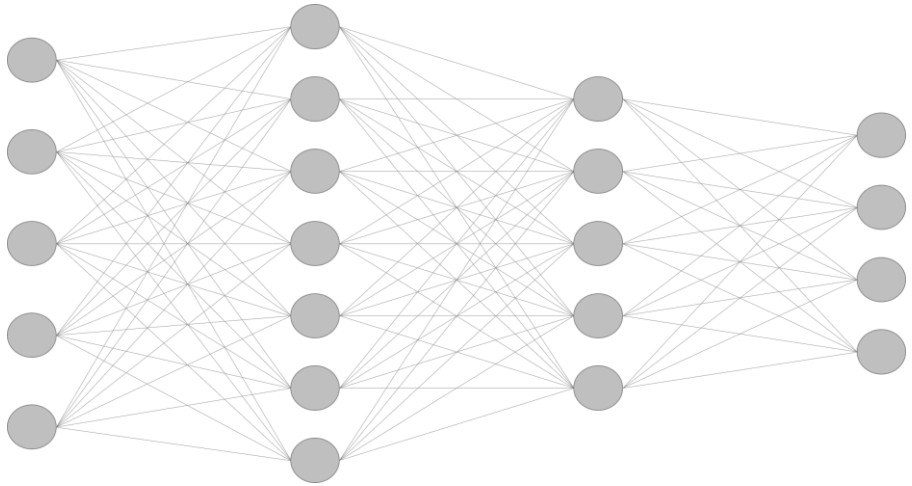
- 순전파 알고리즘 ➔ 입력층-은닉층-출력층에서 각 노드의 값을 계산해나가는 과정 (Step2, 3)
- 오류 역전파 알고리즘 ➔ 오차를 구하는 과정 (Step4, 5에서 $\partial E_n/\partial w_{kj}$, $\partial E_n/\partial w_{ji}$ 를 구하는 과정)
- 경사하강법 ➔ 오차의 기여도 (기울기, 미분값)을 통해 오류를 최소화하는 방향으로 weight를 업데이트 하는 과정 (Step4, 5)

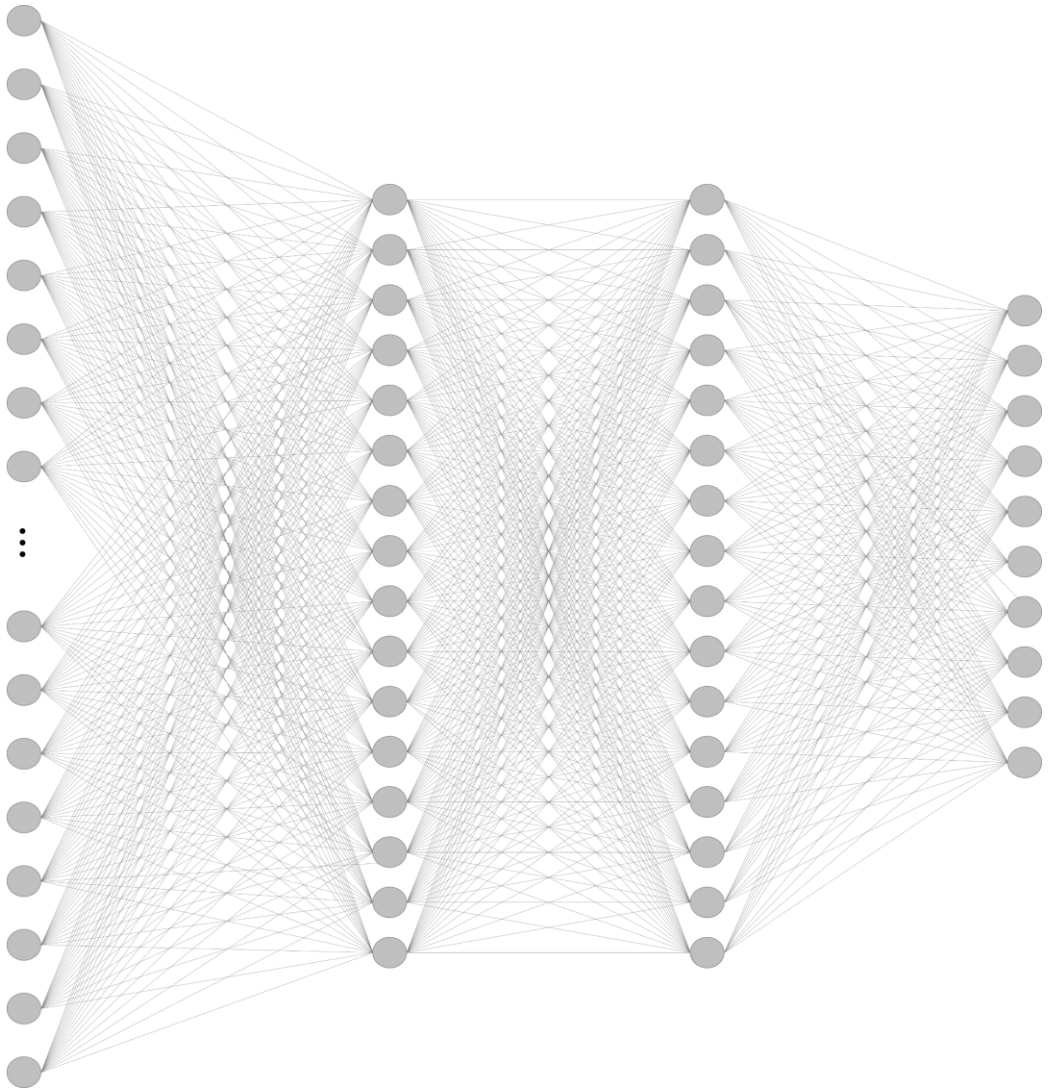
Summary

- Input layer, hidden layer, output layer
- Input nodes, hidden nodes, output nodes
- Cost function
- Activation function
- Feed forward and error backpropagation
- Gradient descent method

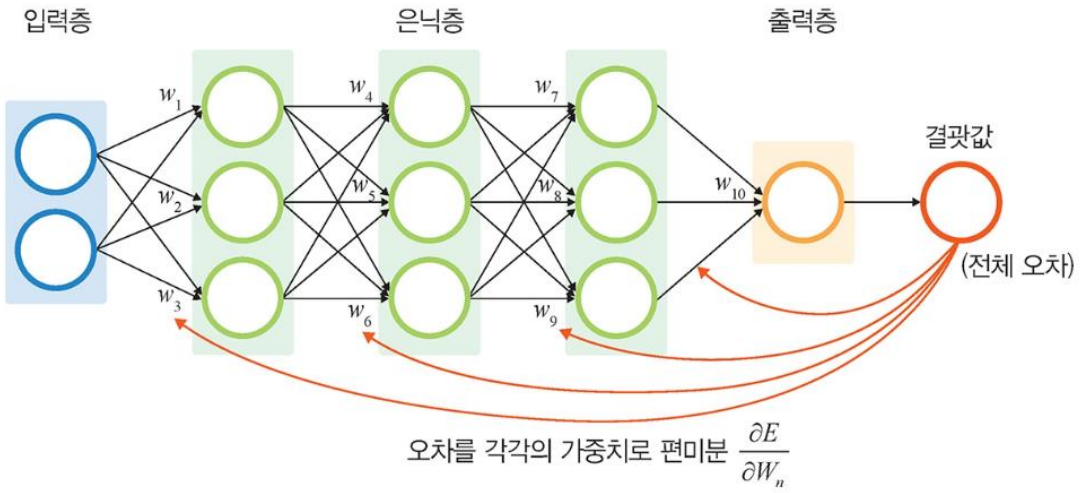
Deep learning model

- Multi-layer perceptron에서 은닉층이 매우 많은 구조
  - Deep MLP, Deep Neural Network (DNN), etc.





Performance issues on naïve DNN model



$$w_k = f(net_k)$$

- **오차를 각각의 가중치로 편미분 하는 과정에서**

$$\frac{\partial E_n}{\partial w_{10}} = \frac{\partial E_n}{\partial o} \times \frac{\partial o}{\partial net_{10}} \times \frac{\partial net_{10}}{\partial w_{10}}$$

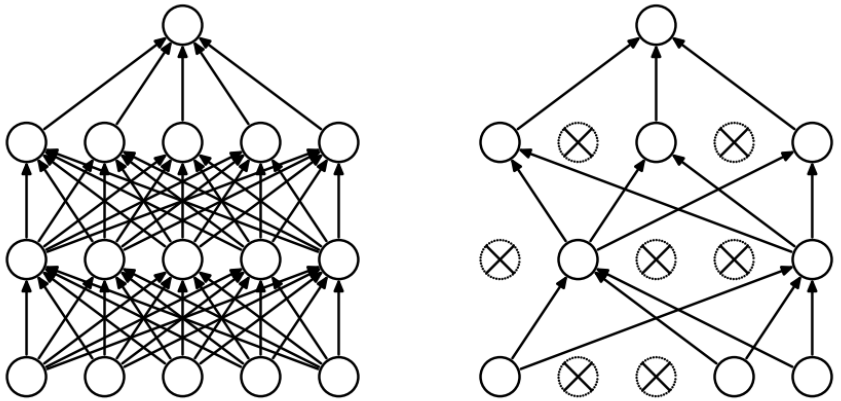
$$\frac{\partial E_n}{\partial w_9} = \frac{\partial E_n}{\partial o} \times \frac{\partial o}{\partial net_{10}} \times \frac{\partial net_{10}}{\partial w_{10}} \times \frac{\partial w_{10}}{\partial net_9} \times \frac{\partial net_9}{\partial w_9}$$

$$\frac{\partial E_n}{\partial w_3} = \frac{\partial E_n}{\partial o} \times \frac{\partial o}{\partial net_{10}} \times \frac{\partial net_{10}}{\partial w_{10}} \times \frac{\partial w_{10}}{\partial net_9} \times \frac{\partial net_9}{\partial w_9} \ldots \times \frac{\partial w_4}{\partial net_3} \times \frac{\partial net_3}{\partial w_3} \Rightarrow 0.0000000000 \ldots$$

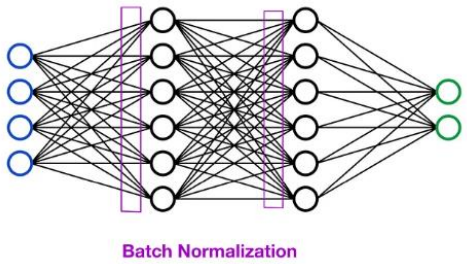
Gradient vanishing problem

Solutions

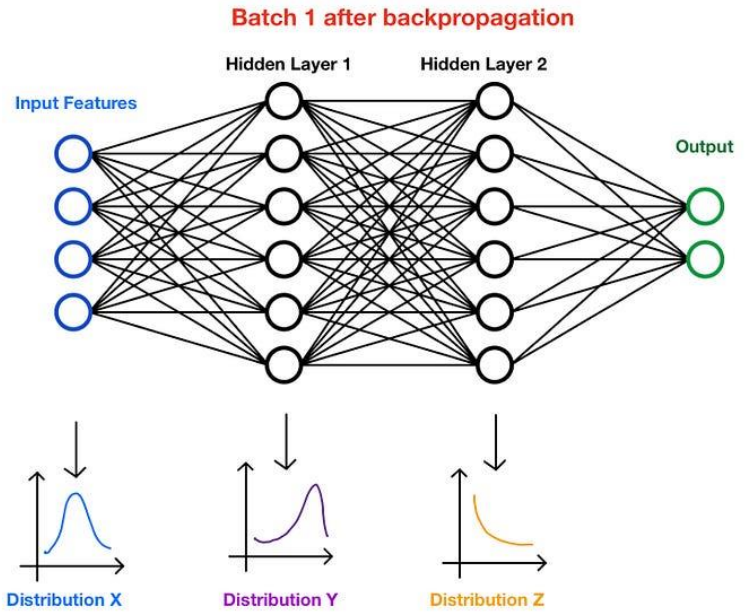
- Fine-tuning techniques
  - Dropout
  - Activation function (tanh ➔ ReLU)
  - Batch normalization (BN)
  - Weight initialization
  - Residual network (ResNets)
  - …



(a) Standard Neural Net (b) After applying dropout.



$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$



End of slide