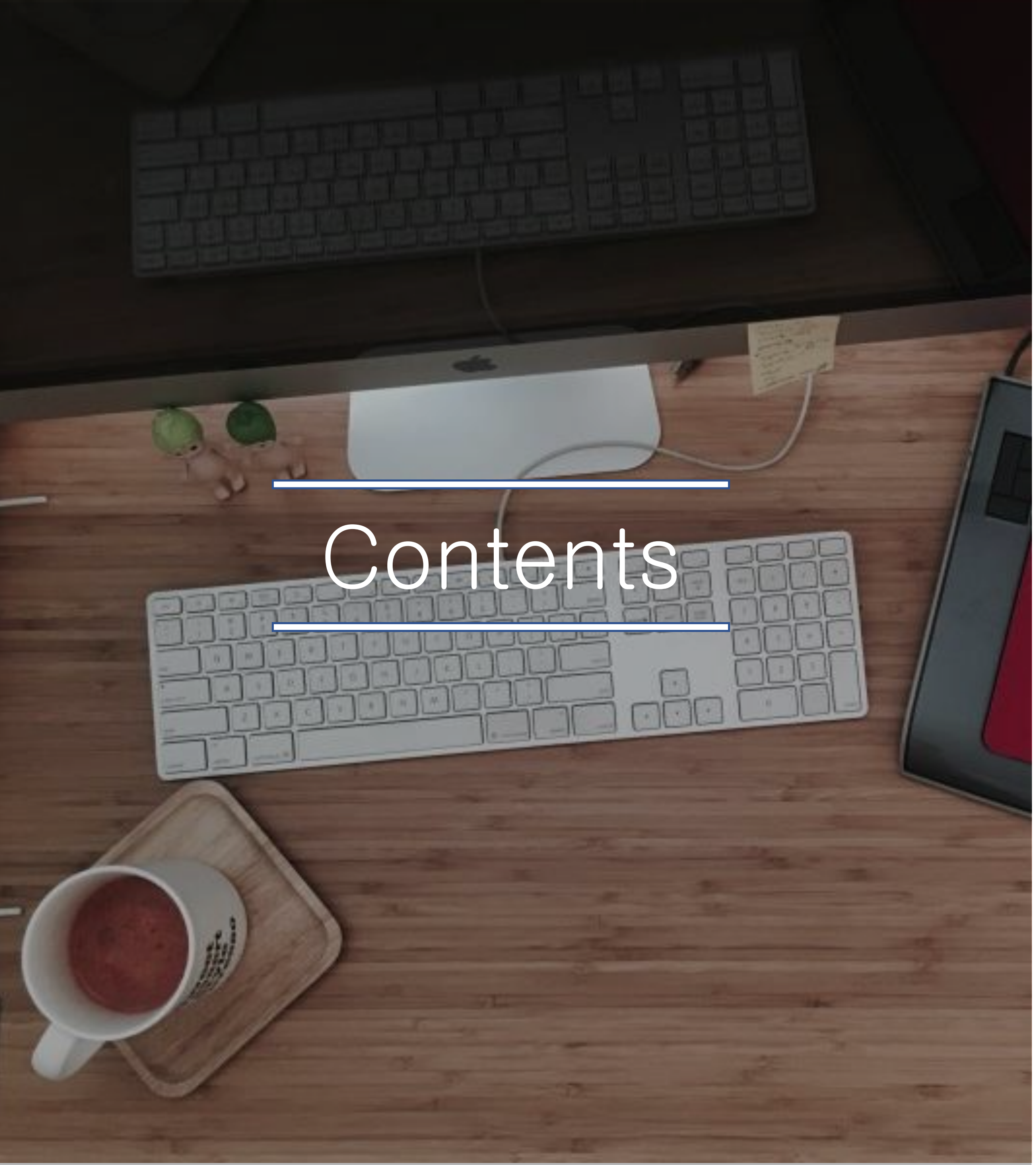
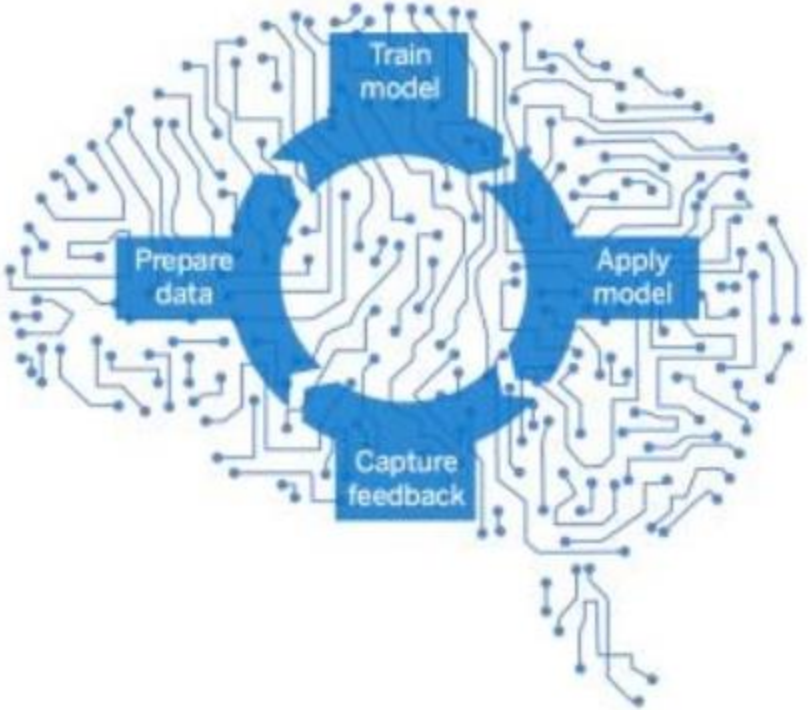
인공지능(Artificial Intelligence)

인하대학교

HCI Lab
Inha University
Human-Computer Interaction Lab



Contents

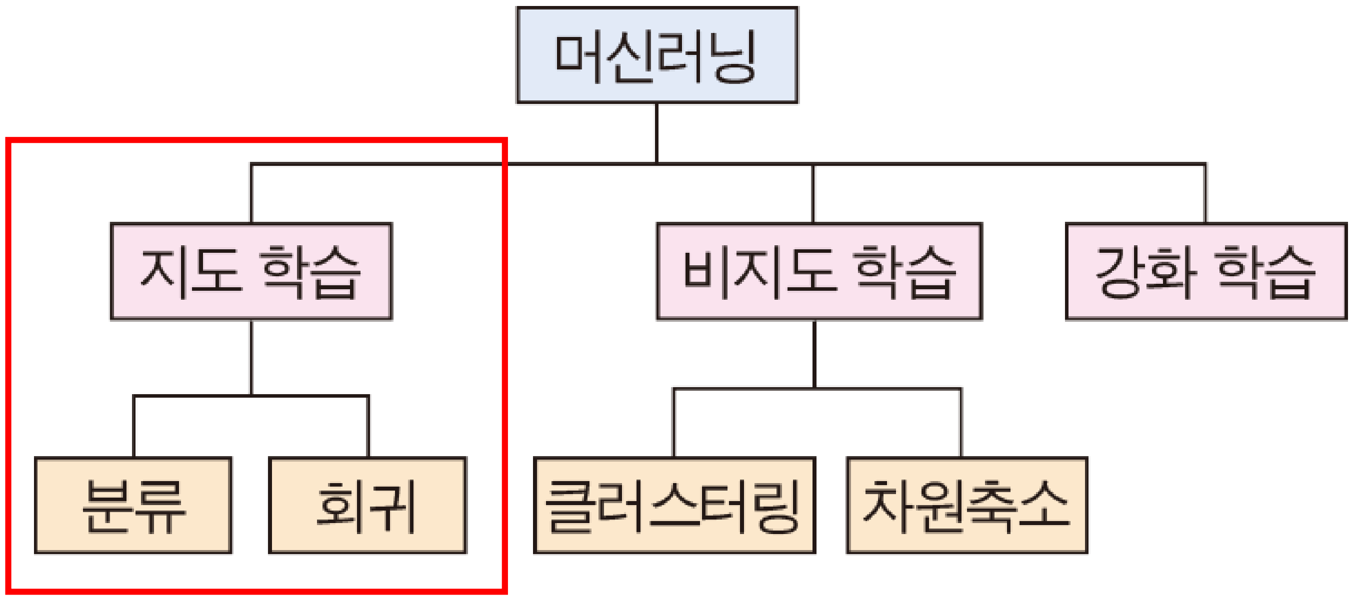


Let’s Recap

머신러닝의 학습 방법

(1) 머신러닝에서의 학습 방법

- 학습의 형태에 따라 3가지 학습 방법
- 지도 학습, 비지도 학습, 강화 학습으로 구분
- 지도학습은 분류와 회귀, 비지도 학습은 클러스터링과 차원 축소로 다시 나누어짐



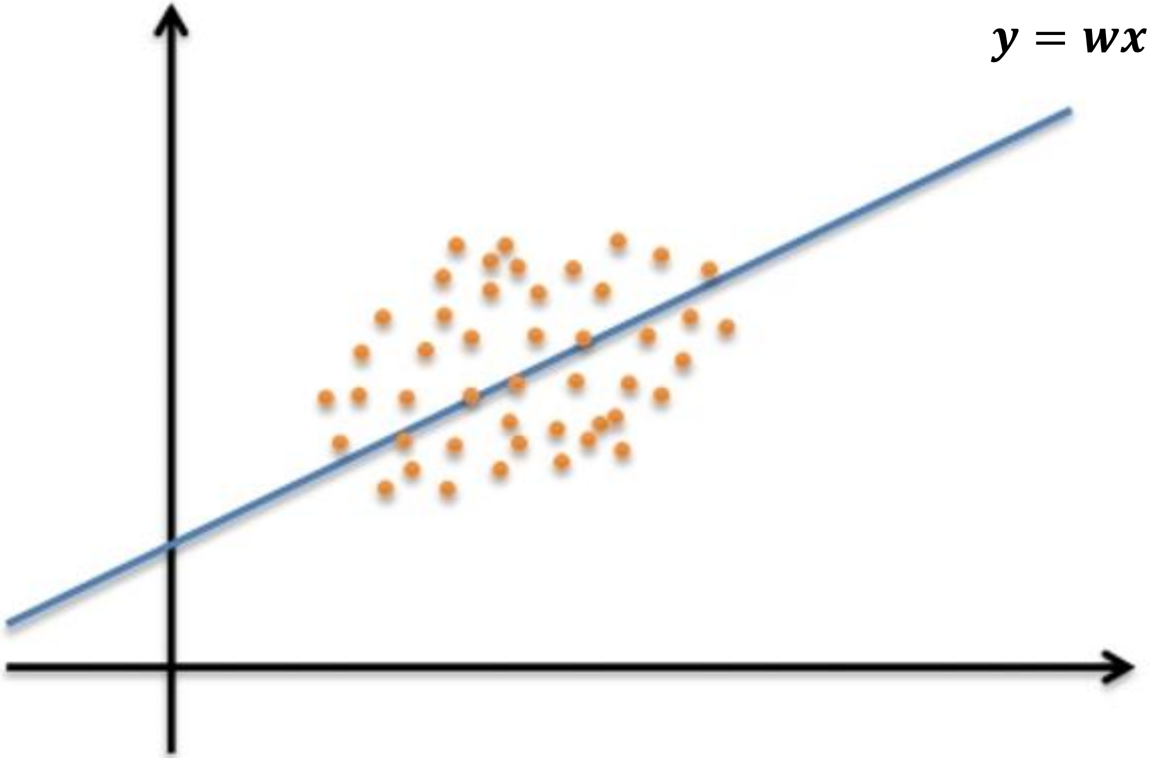
1 Linear & Logistic Regression

Linear Regression

선형 회귀(Linear Regression) : 종속 변수 y와 한 개 이상의 독립 변수 X의 선형 상관 관계를 모델링하는 회귀분석 기법

x 에 대한 y 값을 가장 잘 설명하는 변수 w, b 를 찾는 것

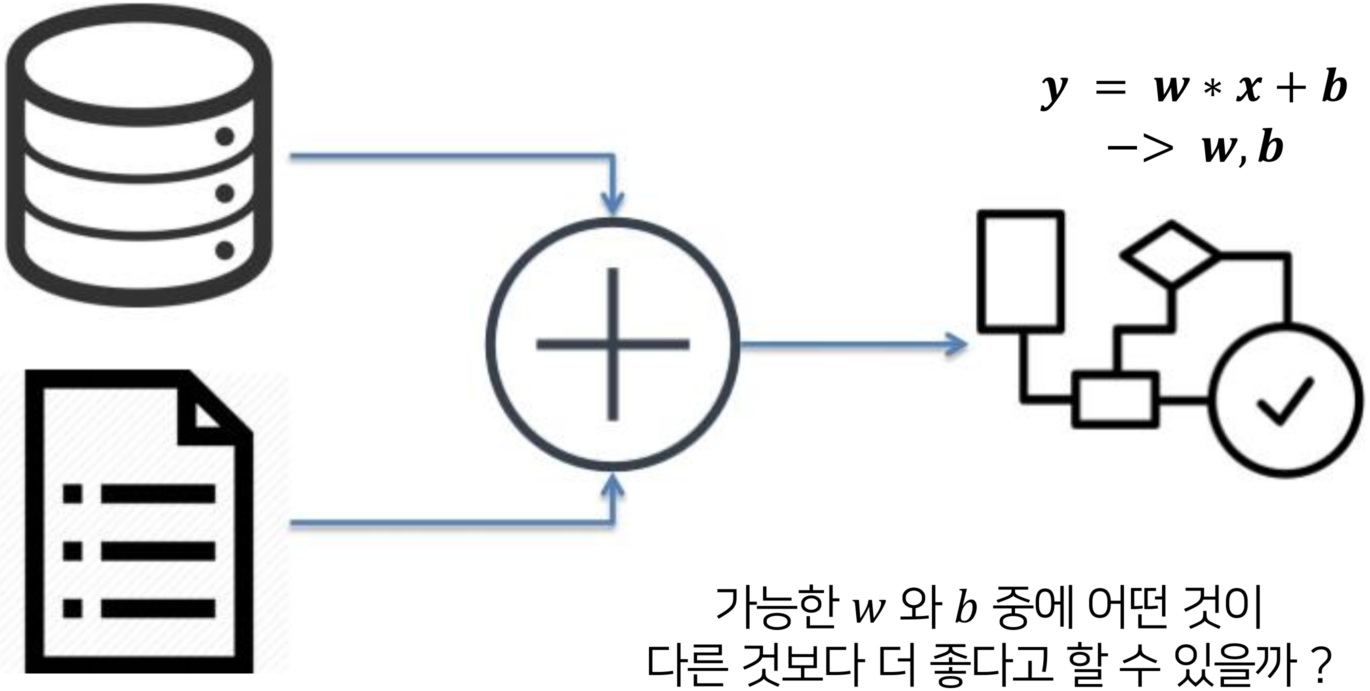
$$\boldsymbol{y = wx + b}$$



Linear Regression

$x1, x2, x3, \ldots$

$y1, y2, y3, \ldots$



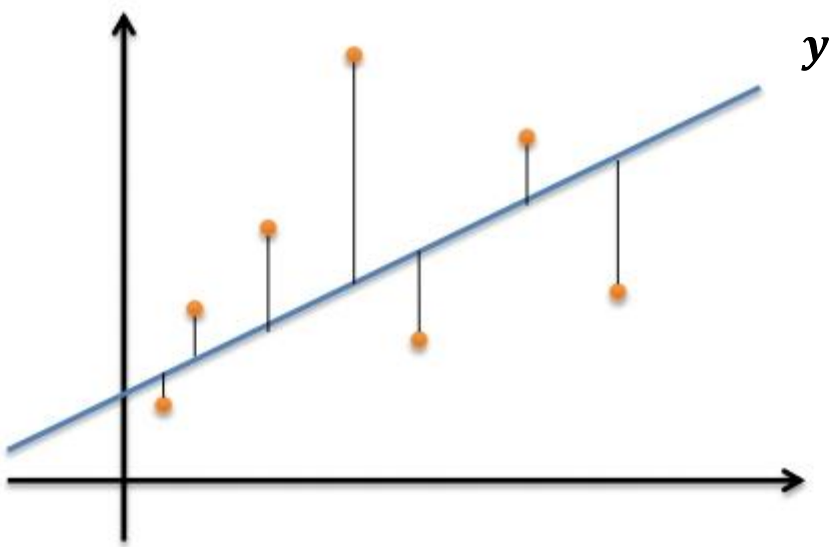
$\boldsymbol{y} = \boldsymbol{w} * \boldsymbol{x} + \boldsymbol{b}$

$-> \boldsymbol{w}, \boldsymbol{b}$

가능한 w 와 b 중에 어떤 것이
다른 것보다 더 좋다고 할 수 있을까 ?

Linear Regression

잘 예측했는지 아닌지 측정할 **척도(metric)** 가 필요함



Mean Squared Error(MSE)

$$MSE = \frac{(x1 - x2)^2}{n}$$

두 값의 거리의 제곱의 평균

임의의 w*,b* 를 초기값으로 한다면

$$Loss = \frac{(y^* - y)^2}{n} = \frac{(w^*x + b^* - y)^2}{n}$$

loss 값은 고정된 x, y(데이터) 에서 w*,b*에 의해 구해짐

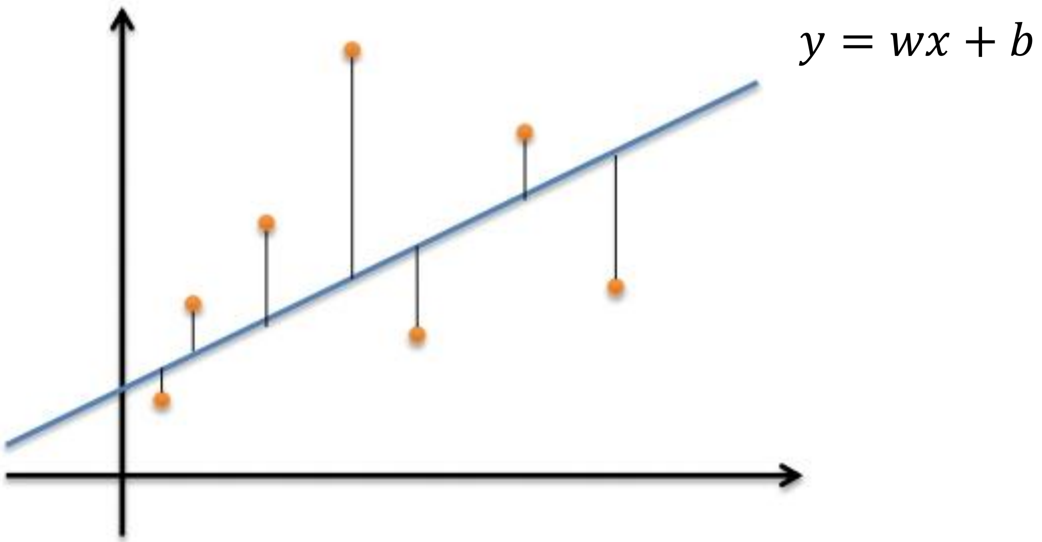
Linear Regression

잘 예측했는지 아닌지 측정할 **척도(metric)** 가 필요함

Mean Squared Error(MSE)

$$MSE = \frac{(x1 - x2)^2}{n}$$

두 값의 거리의 제곱의 평균



$$Loss = \frac{(y^* - y)^2}{n} = \frac{(w^*x + b^* - y)^2}{n}$$

loss 값은 고정된 x, y(데이터) 에서 w*,b*에 의해 구해짐

예시)

y = 0.5*x+4 이고 w*=2,

b*=2 일 때 , x=3 에서 loss 는 ?

$$Loss = \frac{(8 - 5.5)^2}{n} = \frac{(2.5)^2}{n}$$

Linear Regression

- Loss 를 minimize 하는 w,b 를 구하고자 함

  → Grid/Random Search?

Linear Regression

- Loss 를 minimize 하는 w,b 를 구하고자 함

  → Grid/Random Search?

시간이 너무 오래 걸림

Linear Regression

- Loss 를 minimize 하는 w,b 를 구하고자 함

  → Grid/Random Search?

  → Loss값을 최소화 해서 구하자!

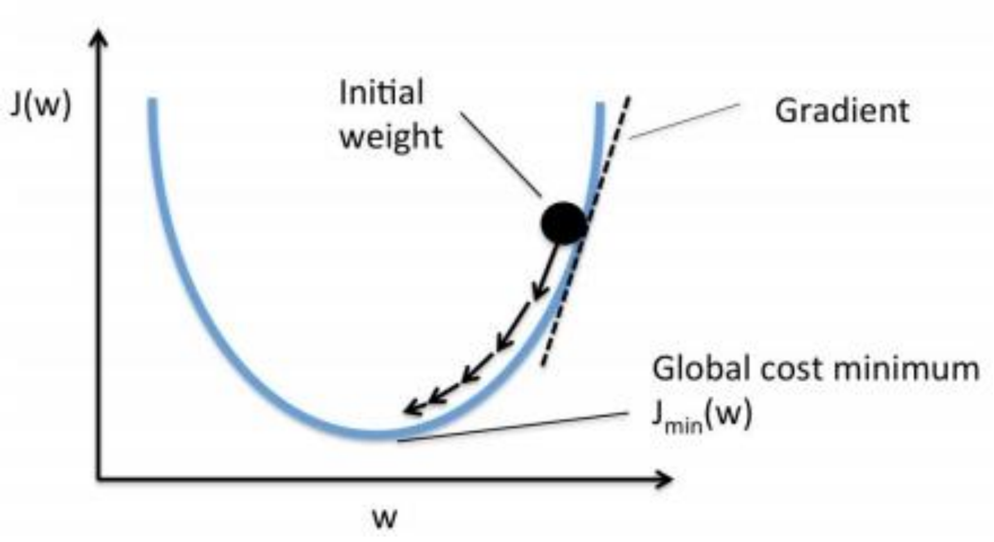
Gradient Descent

- Loss 를 minimize 하는 w,b 를 구하고자 함

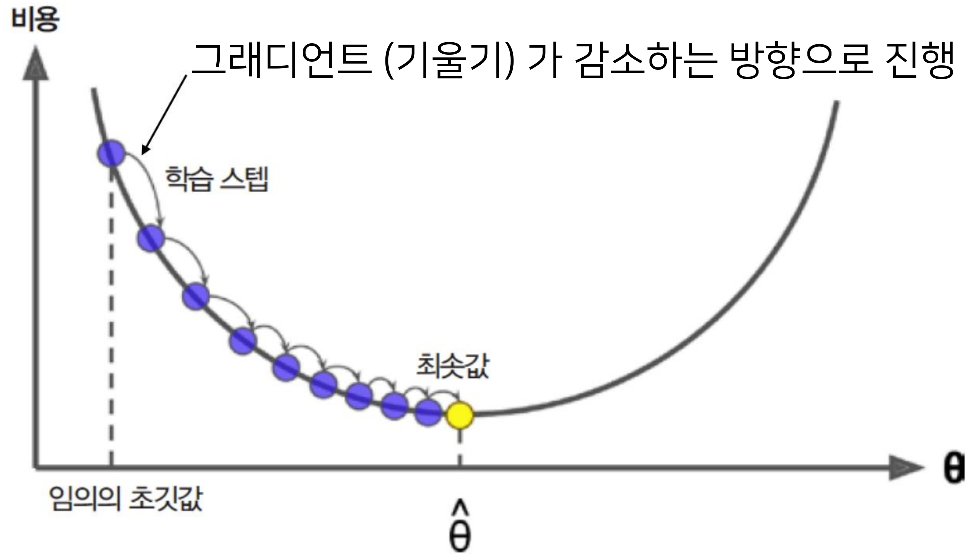
  → Grid/Random Search?

  → Loss값을 최소화 해서 구하자!

경사하강법(Gradient Descent!)



Gradient Descent

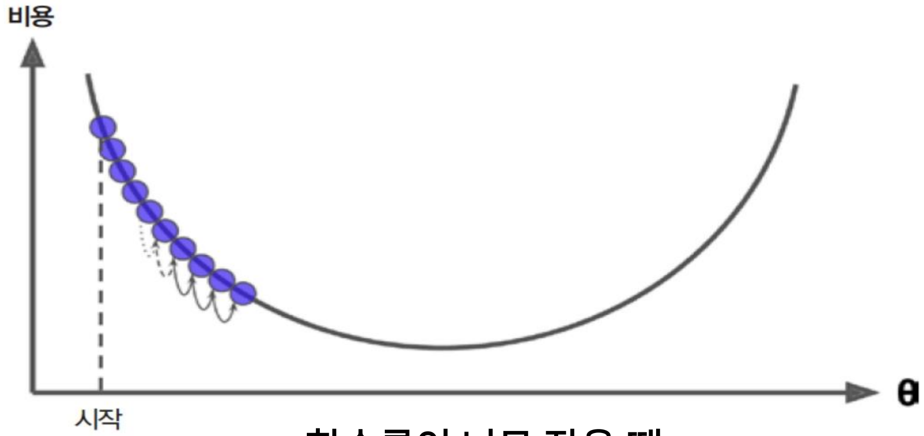


현재 loss 에 대한 w 의 $gradient$(경사도) 를 구하여

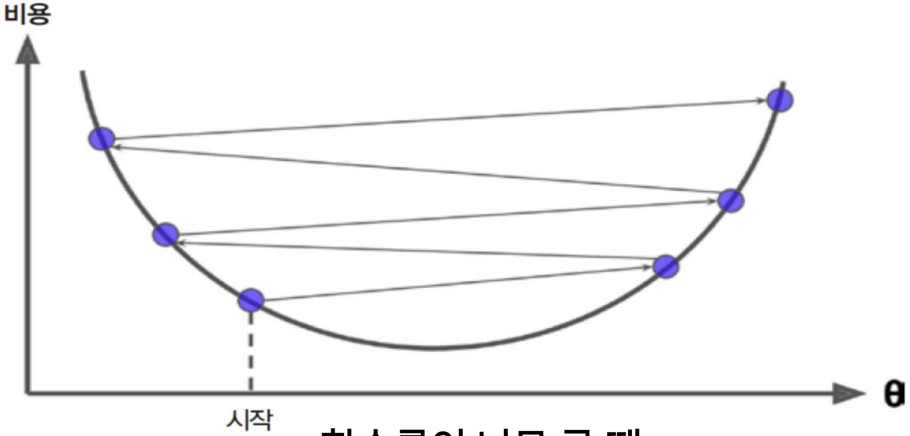
$gradient \times learning\ rate$ 만큼 w 를 업데이트

$$\boxed{w_{t+1} = w_t - \text{gradient} * \text{learning\_rate}}$$

계속 업데이트 하다 보면 optimal(최적의)한 w 에 근접



학습률이 너무 작을 때



학습률이 너무 클 때

CI Lab

논리회로

- **학습과정**

1. 임의의 w와 b를 설정
2. 주어진 훈련 데이터를 이용하여, 결과값(y)를 도출
3. 결과값(y)와 실제 결과값($\hat{y}$) 사이의 오차 계산
4. 오차(Loss)를 줄이는 방향으로 w와 b를 재설정(학습)
5. 오차를 최소한으로 줄이도록 2~4의 과정을 계속반복진행

분류(Classification) / 회귀(Regression)

- **분류 (Classification)** : 입력 데이터가 어느 Class에 속하는 지에 대한 문제
  ex) 인물 사진 -> 남 or 여

- **회귀 (Regression)** : 입력 데이터에서 연속적인 수치를 예측하는 문제
  ex) 인물 사진 -> 키 200cm

- **출력층에서 사용하는 활성화 함수**
  - 항등 함수 : 입력을 그대로 출력 (Identity function)
  - Softmax 함수 : 지수 함수를 활용한 분류기. 각 요소의 값은 0~1, 합은 1이 됨

$$y_k = \frac{exp(a_k)}{\sum_{i=1}^{n} exp(a_i)} = \frac{\exp(a_k - C)}{\sum_{i=1}^{n} \exp(a_i - C)}$$

(임의의 정수 C는 지수 오버플로 방지 대책, 입력 최댓값 사용이 일반적)

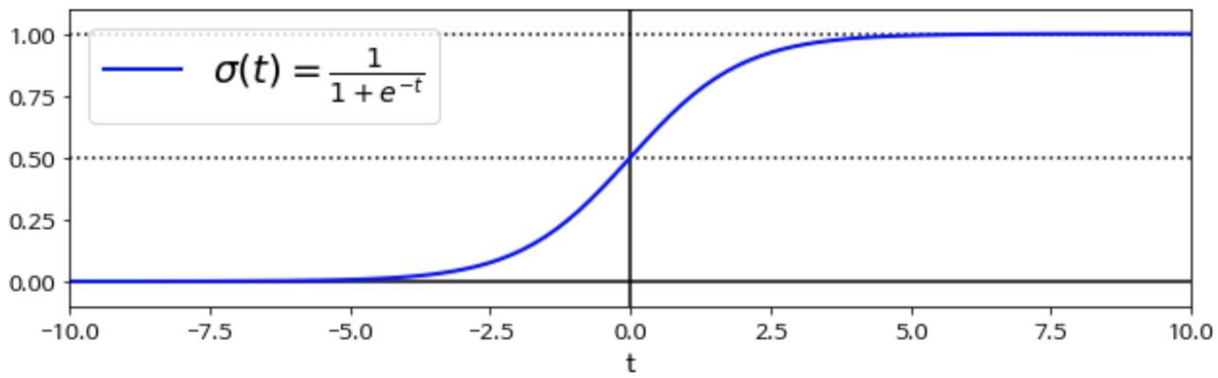
HCI Lab
Inha University
Human-Computer Interaction Lab

로지스틱 회귀(logistic regression)

- 로짓(logit) 회귀이라고도 부르며 분류하고자 할 때 사용
- 이진 분류는 양성 (1) 클래스와 음성 (0) 클래스를 분류
- 선형 방정식을 시그모이드 (sigmoid) 함수에 통과시켜 0~1 사이의 확률을 얻음
- 예측

$$\hat{y} = \begin{cases} 0 & \hat{p} < 0.5 \quad or \quad w \cdot x < 0 \text{일 때} \\ 1 & \hat{p} \geq 0.5 \quad or \quad w \cdot x \geq 0 \text{일 때} \end{cases}$$

클래스 확률 예측값



Cross Entropy Error

- 분류일 경우, Cross Entropy Error(CEE)를 사용!

$$L = -\sum_{i=1}^{n} t_i \log y_i$$

- t는 정답 값, y는 추론 값

- 정답의 개수와 추론의 개수는 같음

ex) 2개이면 이진분류(Binary Classification), 2개보다 많으면 다중분류(Multiple Classification)

✓ y값은 신경망 여러 개를 거쳐 산출된 값이 최종 마지막 어떤 특별한 활성함수의 입력값이 되어 산출된 결과값

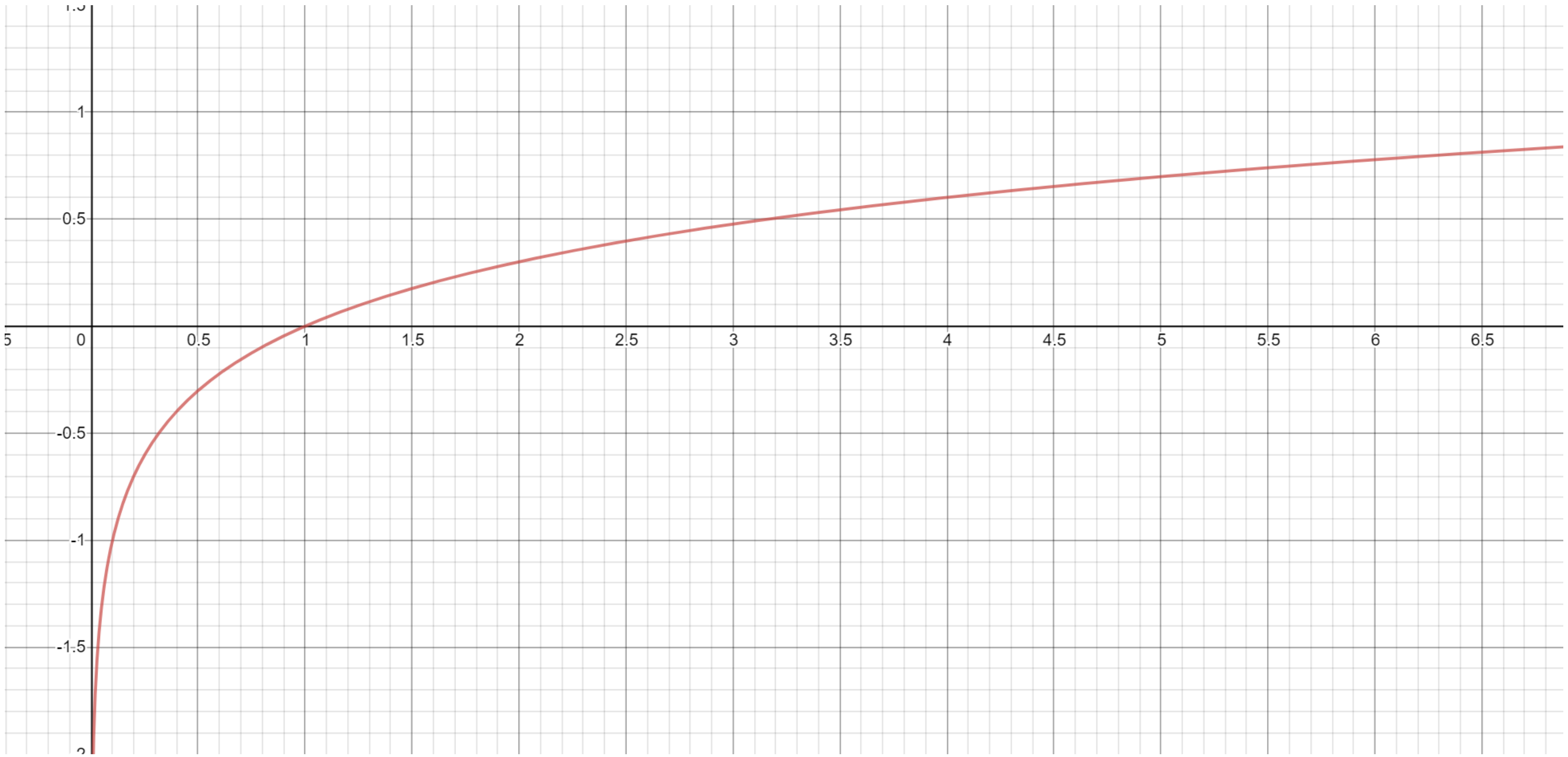
✓ **이진분류**로 가장 많이 사용되는 활성함수는 Sigmoid이고 **다중분류**로 가장 많이 사용되는 것이 Softmax

Cross Entropy Error

- **이진분류 예시**

  - L은 참 일 때(t=1) –logy가 되고, 거짓 (t=0)일 때 -log(1-y)가 되며, 다음과 같이 그래프로 나타낼 수 있음

$$L = -(t\log(y) + (1-t)\log(1-y))$$



Cross Entropy Error

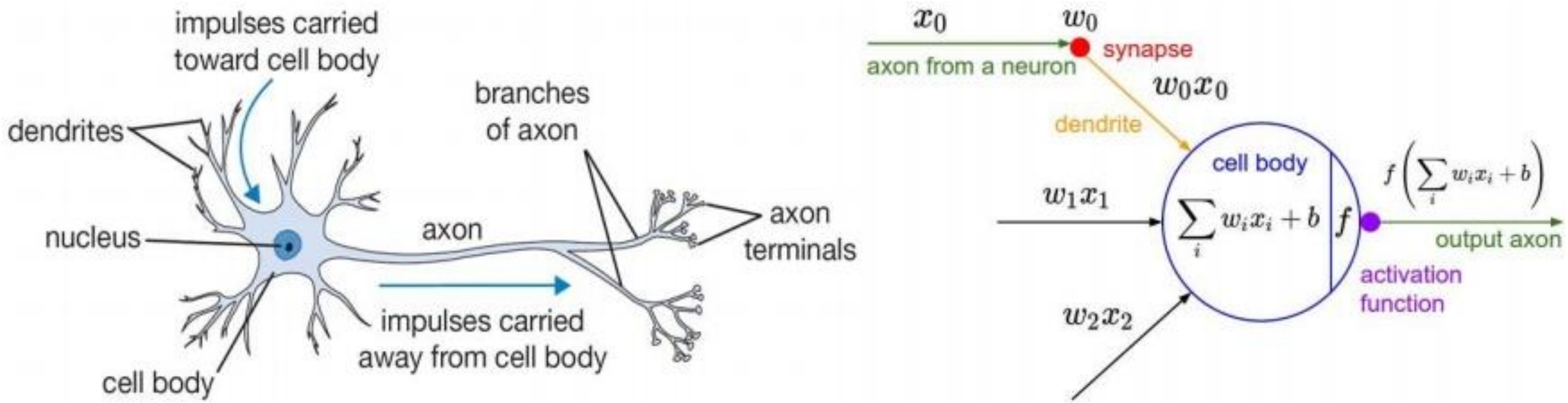
$$L = -(t\log(y) + (1-t)\log(1-y))$$

- y는 항상 0~1사이이므로 y가 참 일 때(제대로 학습한 경우), 0은 무한대이고, 1은 0임.
- 즉, y가 참(t=1)에 가까울수록 L(손실함수)는 0에 가까워지고 y가 거짓(t=0)에 가까울수록 L은 무한대에 가까워 짐
- y가 거짓일때(제대로 학습하지 못한 경우), y가 참(t=1)에 가까울수록 L(손실함수)는 무한대에 가까워지고 y가 거짓(t=0)에 가까울수록 L은 0에 가까워 짐

2 Perceptron

퍼셉트론이란?

- **퍼셉트론**(Perceptron, 1957): 뇌의 신경(Neuron)을 모델화
- 다수의 신호를 입력으로 받아 하나의 신호를 출력
- 퍼셉트론은 신호 1과 0을 가질 수 있으며 1은 신호 흐름, 0 은 신호 흐르지 않음을 의미



여러 자극이 들어오고 일정 기준을 넘으면 이를 다른 뉴런에 전달하는 구조

(출처 : http://cs231n.github.io/neural-networks-1/)

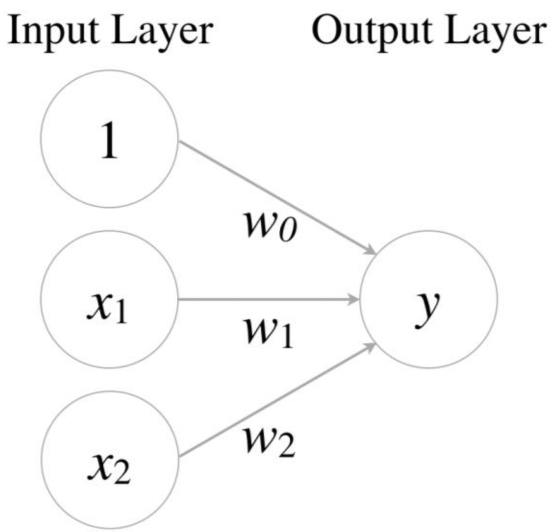
HCI Lab Inha University Human-Computer Interaction Lab

Linear Regression to Perceptron

- 입력이 2개, 출력이 1개인 Perceptron을 생각해보자

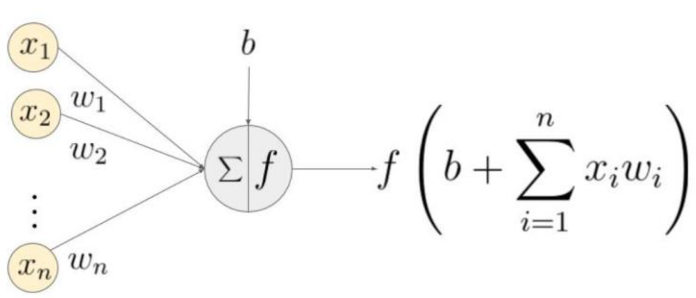
$$y = Wx + b$$

$$\begin{aligned} y &= W_1 x_1 + W_2 x_2 + W_0 x_0 \\ &= W_1 x_1 + W_2 x_2 + b_0 * 1 \\ &= W_1 x_1 + W_2 x_2 + b_0 \end{aligned}$$

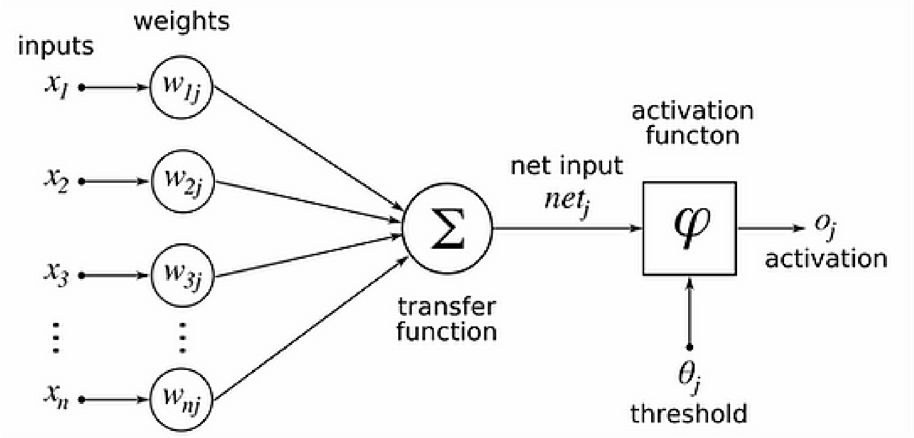


Linear Regression to Perceptron

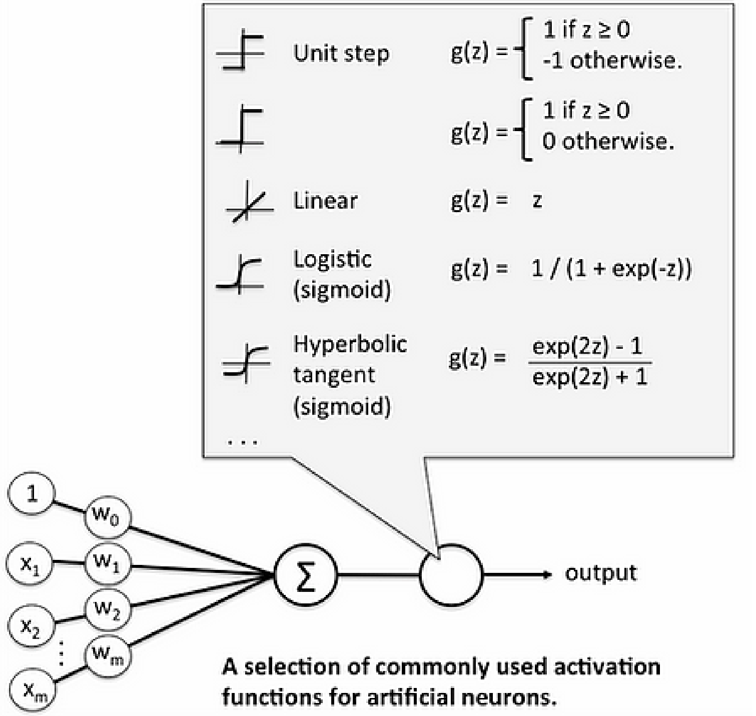
- 출력값 0, 1을 갖기 위해서 활성화 함수 사용



An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.



Source: Isaac Changhau

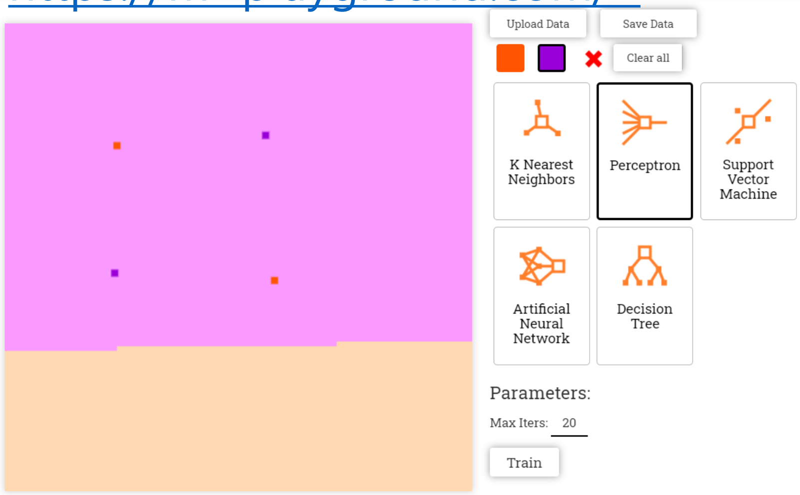


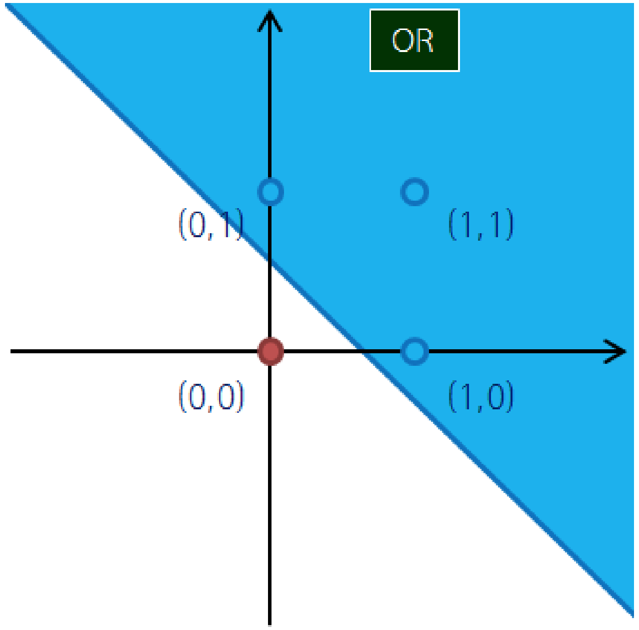
Source: Sebastian Raschka

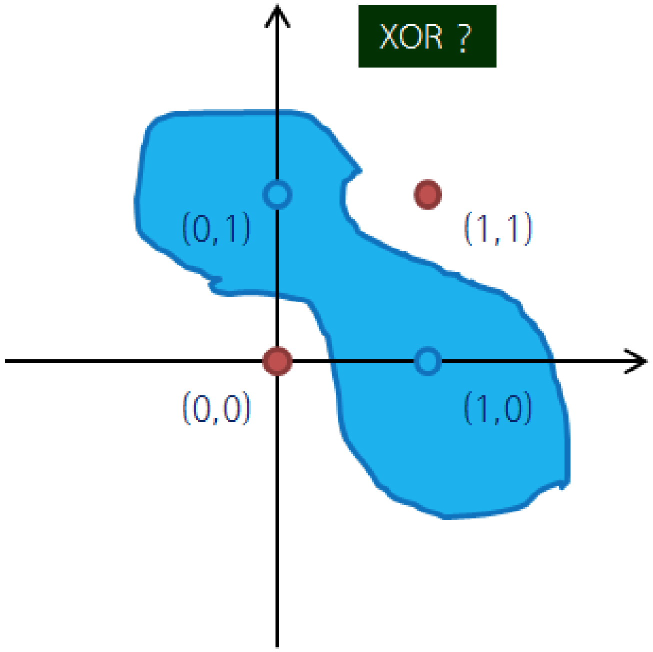
단층 퍼셉트론의 한계

https://ml-playground.com/#

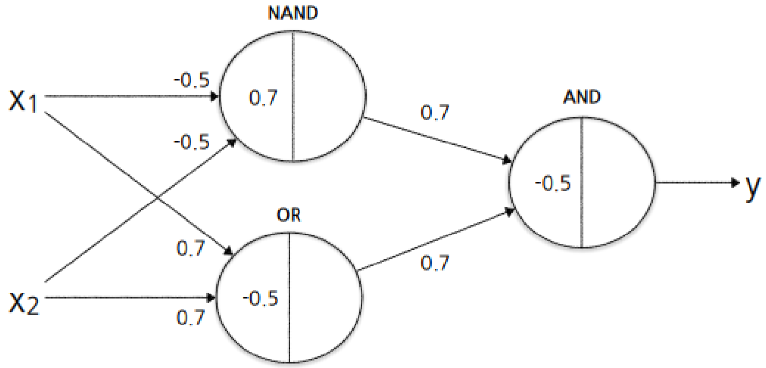
$$y = \begin{cases} 0(-0.5 + x1 + x2 <= 0) \\ 1(-0.5 + x1 + x2 \ > \ 0) \end{cases}$$







MLP: XOR 게이트

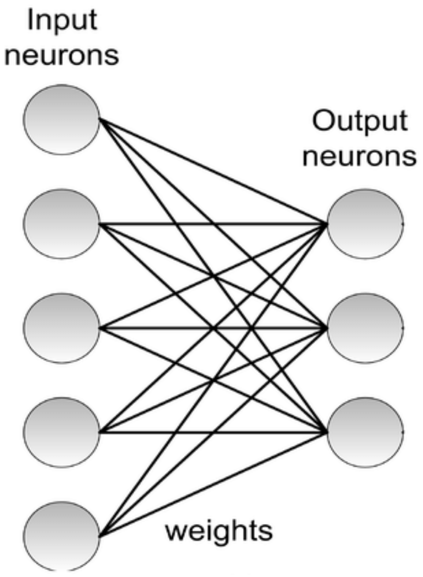


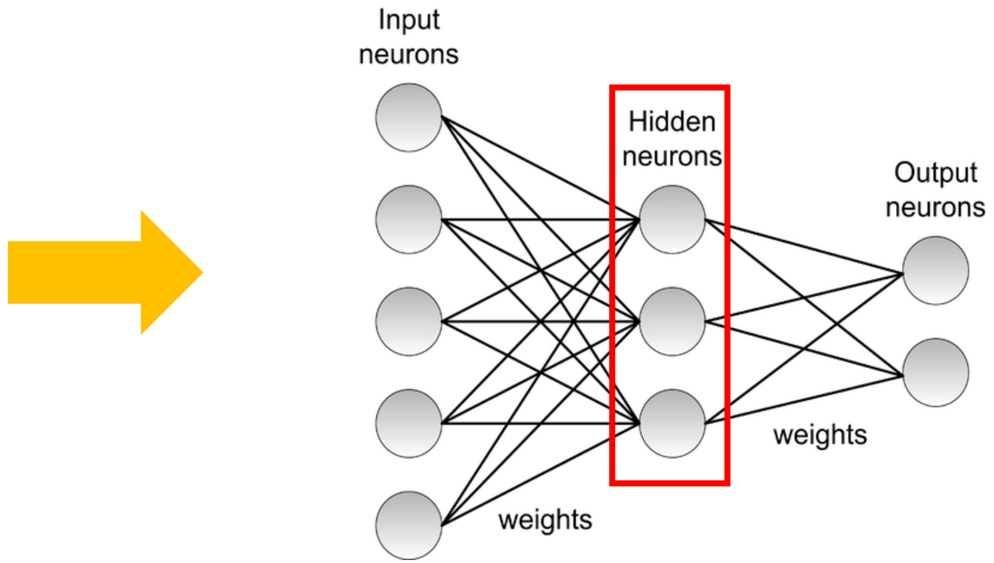
2층 퍼셉트론을 XOR 표현이 가능

다층 퍼셉트론을 이용하면 복잡한 문제 해결 가능 **But, 학습이 불가**

단층 → 다층 퍼셉트론

- 입력이 n개, 출력이 m개인 4층 Perceptron을 생각해보자





$$y = W_1 x_1 + W_2 x_2 + b_0$$

$$y = w4\left(act\left(w3\left(act(w2(act(w1 * input + b1)) + b2)\right) + b3\right)\right) + b4$$

(출처 : http://neuralnetworksanddeeplearning.com/chap5.html)

Linear Regression to Perceptron

- 입력이 n개, 출력이 m개인 2층 Perceptron을 생각해보자

$$y = Wx + b$$

$$y_0 + y_1$$

$$y_0 + y_1 \cdots + y_m = W_0x_0 + W_1x_1 + W_2x_2 + \cdots + W_nx_n + b_0 + b_1 \cdots + b_n$$

Neural Network

$$y = W \cdot x + b$$

$$= \begin{array}{cccc} x & W & & b \\ \begin{bmatrix} x_{00} & x_{01} & \cdots & \\ x_{10} & x_{11} & & \\ \vdots & & \ddots & \\ \vdots & & & x_{mn} \end{bmatrix} & \begin{bmatrix} W_{00} & W_{01} & W_{02} & \cdots \\ W_{10} & W_{11} & & \\ W_{20} & & \ddots & \\ \vdots & & & W_{nl} \end{bmatrix} & + & \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix} \\ m \times n & n \times l & & m \times 1 \end{array}$$

Neural Network

$$y = W \cdot x + b$$

$$
= \begin{array}{c} x \\ \begin{bmatrix} x_{00} & x_{01} & \cdots & \cdots \\ x_{10} & x_{11} & & \\ \vdots & & \ddots & \\ \vdots & & & x_{mn} \end{bmatrix} \\ m \times n \end{array}
\begin{array}{c} W \\ \begin{bmatrix} w_{00} & w_{01} & w_{02} & \cdots \\ w_{10} & w_{11} & & \\ w_{20} & & \ddots & \\ \vdots & & & w_{nl} \end{bmatrix} \\ n \times l \end{array}
+
\begin{array}{c} b \\ \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix} \\ m \times 1 \end{array}
$$

Neural Network

$$y = act(wx+b)$$

$$= \text{activation}\left(\underbrace{\begin{bmatrix} wx + b \end{bmatrix}}_{m \times 1} \right)$$

Neural Network

만약 activation function 이 없다면 아래의 식은 결국 linear function

$$y = w4(act\left(w3\left(act\left(w2\left(act\left(w1 * input + b1\right)\right) + b2\right)\right) + b3\right)\right) + b4$$

행렬곱 계산을 생각해 볼 것!

ex) 행렬크기 $(\mathrm{M}, N) \times (N, K) \times (K, M) = (M, M)$

Neural Network

만약 activation function 이 없다면 아래의 식은 결국 linear function

$$y = w4(act\left(w3\left(act\left(w2\left(act\left(w1 * input + b1\right)\right) + b2\right)\right) + b3\right)\right) + b4$$

퍼셉트론이 끝난 뒤 activation function으로 non-linearity 를 추가해야 함

Neural Network

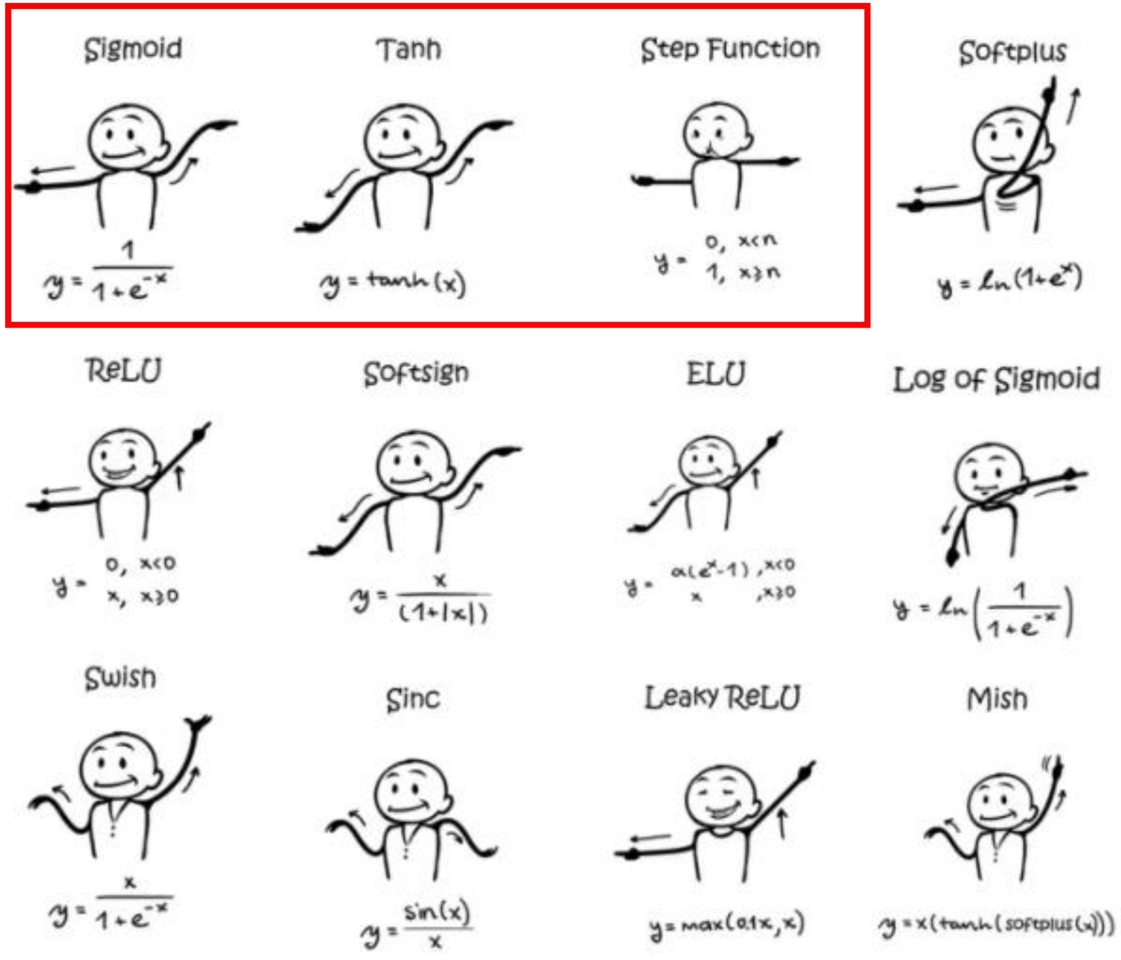
만약 activation function 이 없다면 아래의 식은 결국 linear function

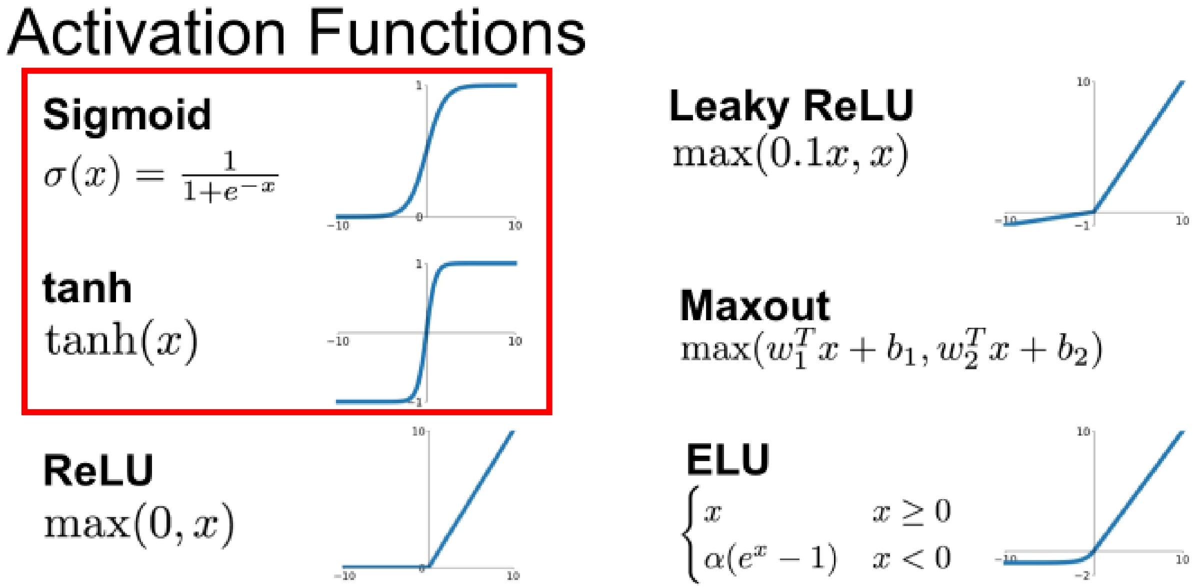
$$y = w4(act\left(w3\left(act(w2(act(w1 * input + b1)) + b2)\right) + b3)\right) + b4$$

퍼셉트론이 끝난 뒤 activation function으로 non-linearity 를 추가해야 함

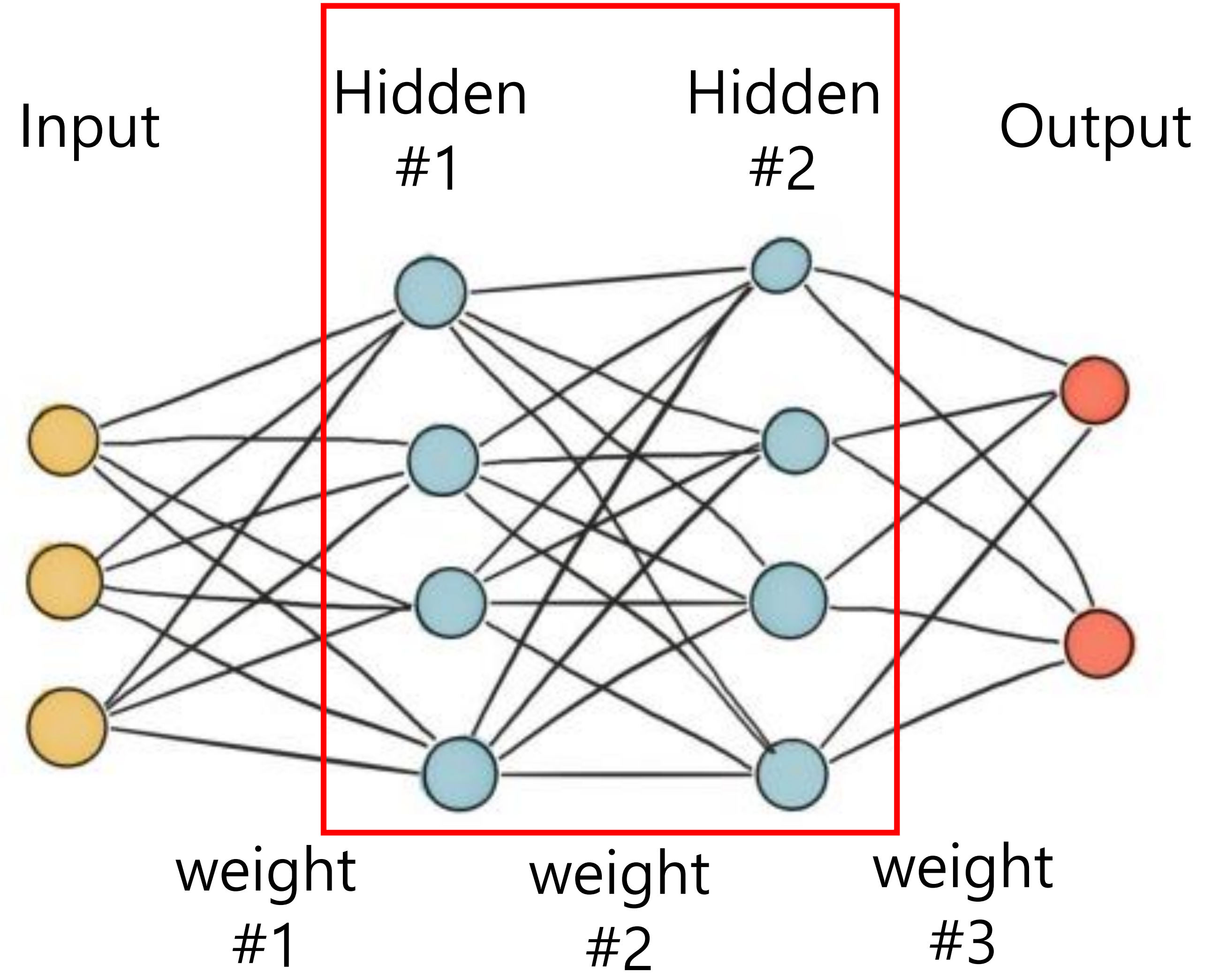
그렇다면 어떤 activation function 을 써야 할까 ?

Activation Function





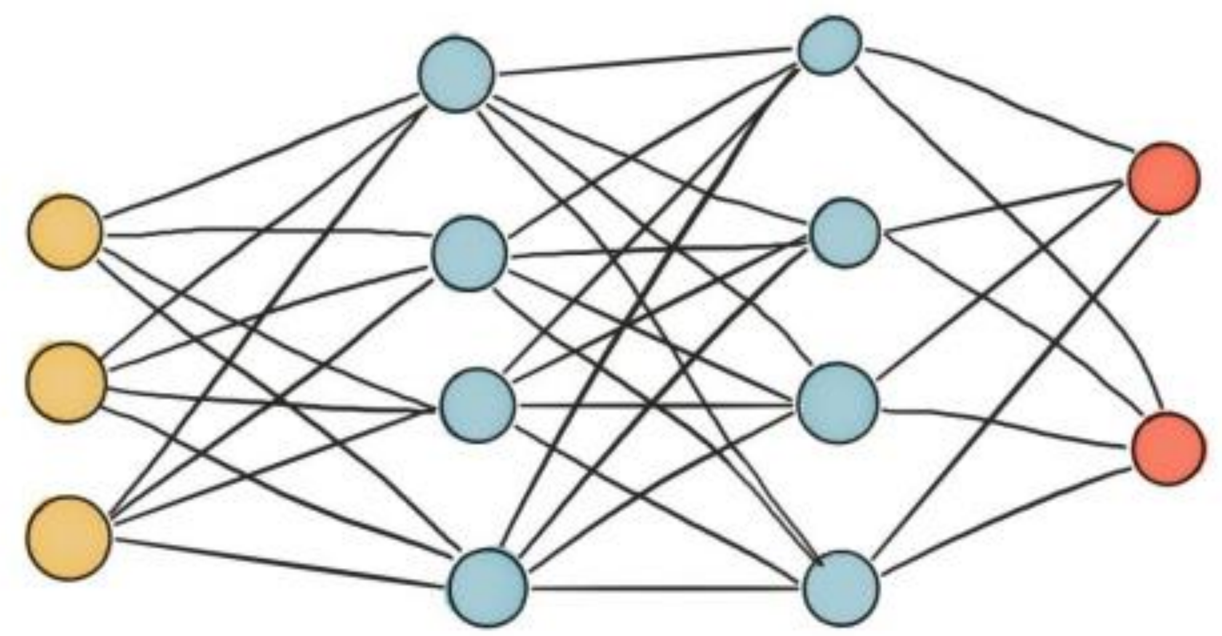
Backpropagation



2층 퍼셉트론을 XOR 표현이 가능

다층 퍼셉트론을 이용하면 복잡한 문제 해결 가능 But, **학습이 불가**

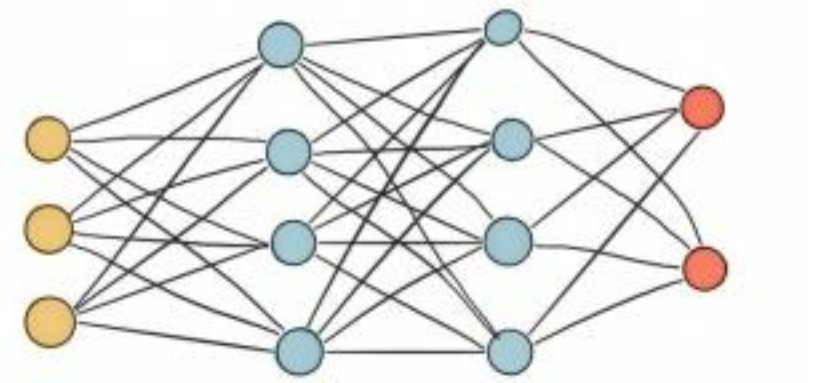
Forward & Back Prop.



$$\begin{bmatrix} w\,00 & w\,01 & w\,02 & w\,03 \\ w\,10 & w\,11 & w\,12 & w\,13 \\ w\,20 & w\,21 & w\,22 & w\,23 \end{bmatrix} \times \begin{bmatrix} w\,00 & w\,01 & w\,02 & w\,03 \\ w\,10 & w\,11 & w\,12 & w\,13 \\ w\,20 & w\,21 & w\,22 & w\,23 \\ w\,30 & w\,31 & w\,32 & w\,33 \end{bmatrix} \times \begin{bmatrix} w\,00 & w\,01 \\ w\,10 & w\,11 \\ w\,20 & w\,21 \\ w\,30 & w\,31 \end{bmatrix}$$

3x4 4x4 4x2

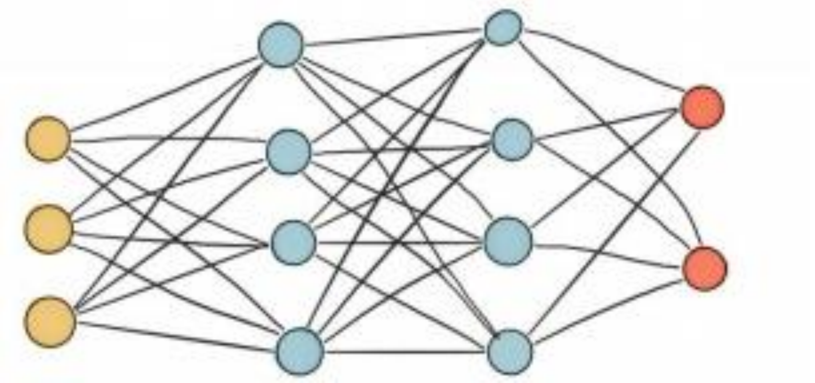
Forward & Back Prop.



쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y^* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

Forward & Back Prop.

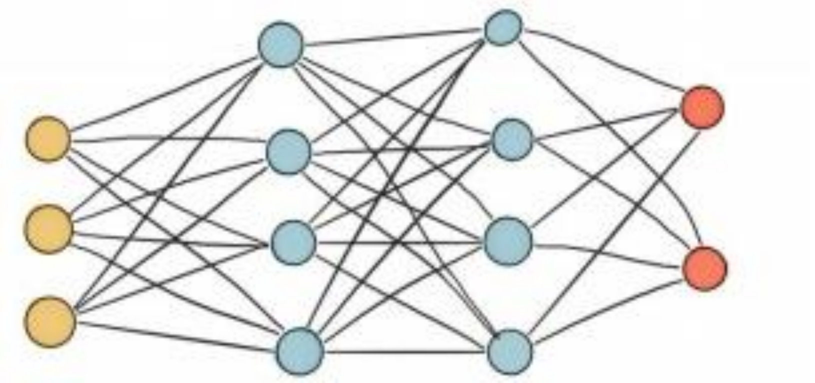


쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y^{*} = w3 * sig\ \ w2 * sig\ \ w1 * x + b1\ \ + b2\ \ + b3$$

$$\begin{aligned} loss\ \ &= y^{*} - y \\ &= w3 * sig\ \ w2 * sig\ \ w1 * x + b1\ \ + b2\ \ + b3 - y \end{aligned}$$

Forward & Back Prop.



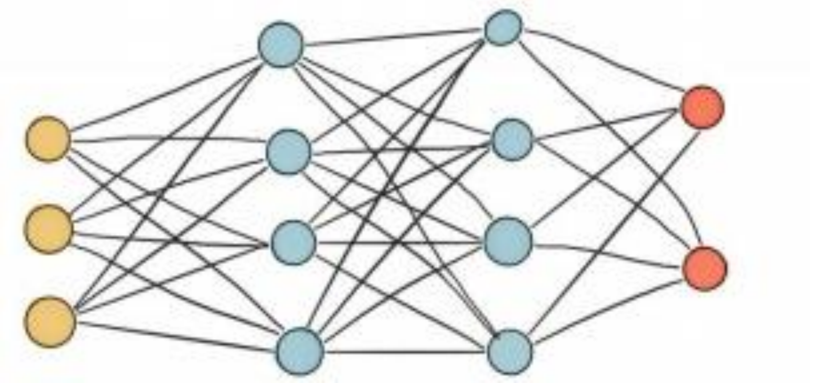
쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y^* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

$$\begin{aligned} loss &= y^* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

Forward & Back Prop.

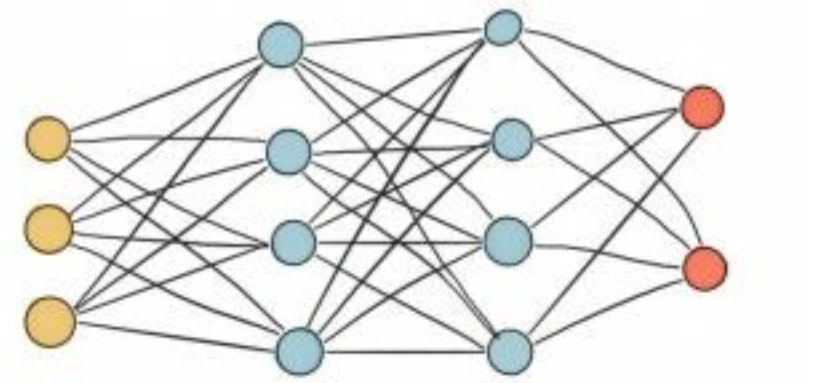


쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$\frac{\partial loss}{\partial w3} = sig\left(w2 * sig\left(w1 * x + b1\right) + b2\right)$$

$$\frac{\partial loss}{\partial b3} = 1$$

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

$$\begin{aligned} loss &= y* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

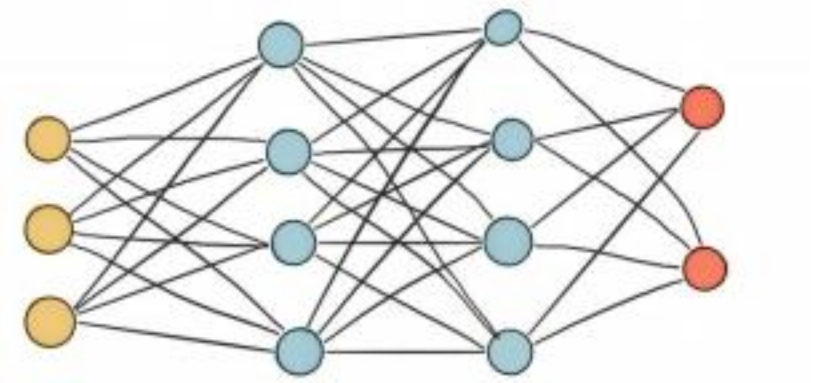
$$\frac{\partial loss}{\partial w3} = sig(w2 * sig(w1 * x + b1) + b2)$$

$$\frac{\partial loss}{\partial b3} = 1$$

$$\frac{\partial loss}{\partial w2} = ??$$

Inha University
Human-Computer Interaction Lab

Forward & Back Prop.



쉽게 이해되도록
loss = 예측값 - 실제로 설정

$$y* = w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3$$

$$\begin{aligned} loss &= y* - y \\ &= w3 * sig(w2 * sig(w1 * x + b1) + b2) + b3 - y \end{aligned}$$

$$\frac{\partial loss}{\partial w3} = sig\,(w2 * sig(w1 * x + b1) + b2)$$

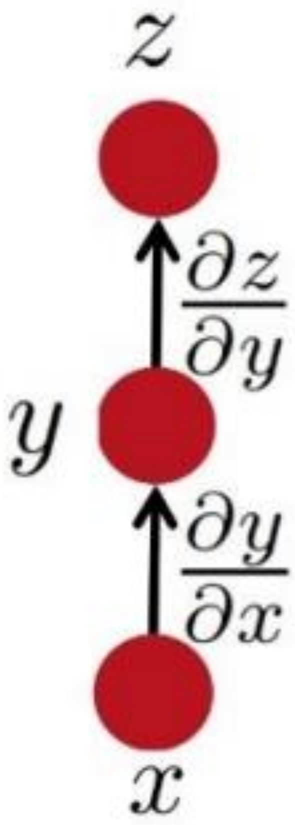
$$\frac{\partial loss}{\partial b3} = 1$$

$$\frac{\partial loss}{\partial w2} = chain\ rule!!$$

HCI Lab
Inha University
Human-Computer Interaction Lab

Forward & Back Prop.

Simple Chain Rule



$$\Delta z = \frac{\partial z}{\partial y} \Delta y$$

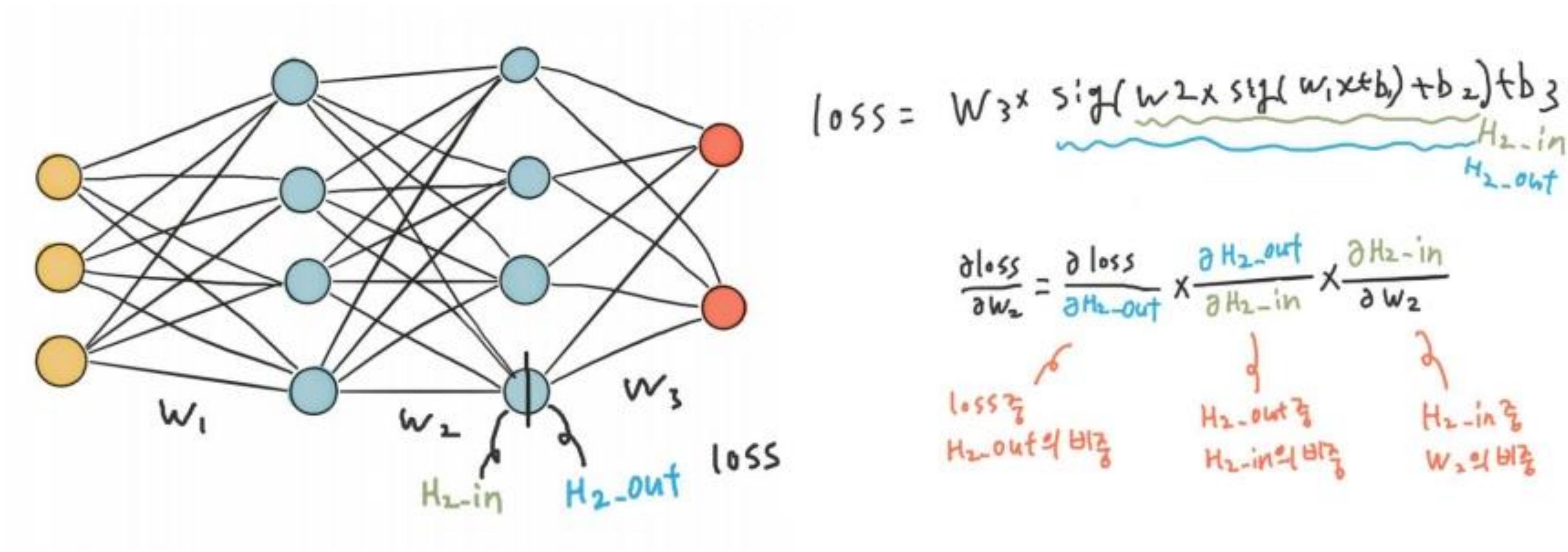
$$\Delta y = \frac{\partial y}{\partial x} \Delta x$$

$$\Delta z = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

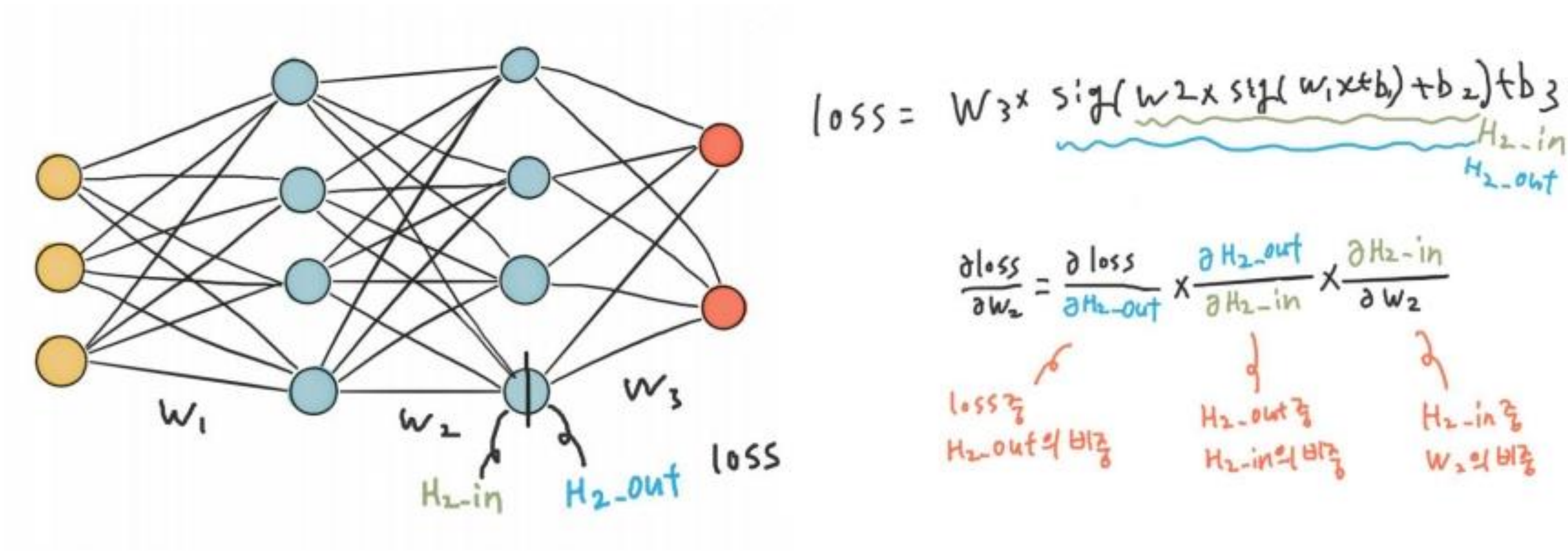
(출처 :https://www.slideshare.net/roelofp/041114-dl-nlpwordembeddings)

Forward & Back Prop.



HCI Lab
Inha University
Human-Computer Interaction Lab

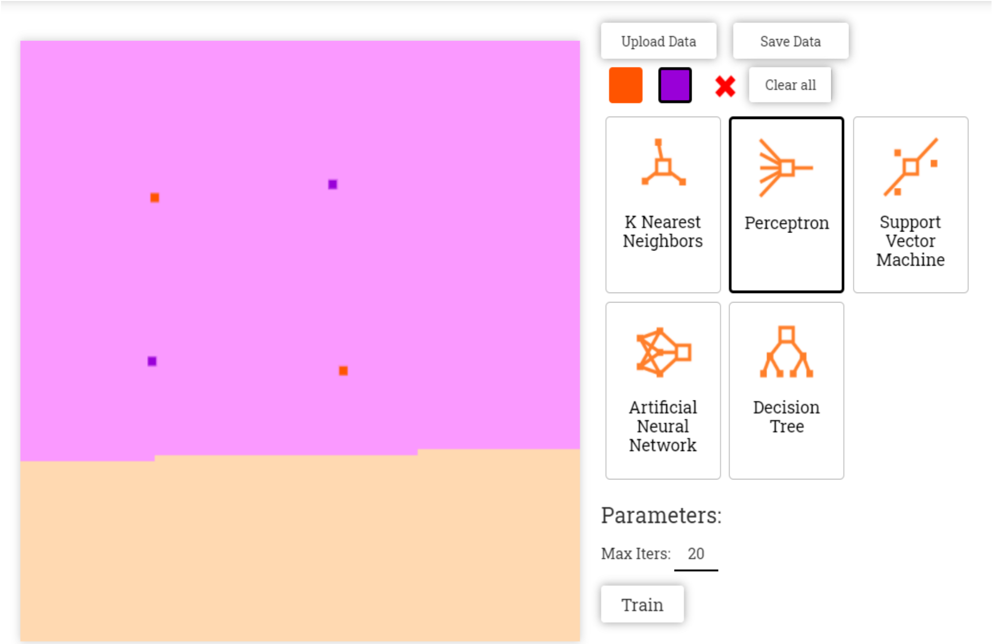
Forward & Back Prop.

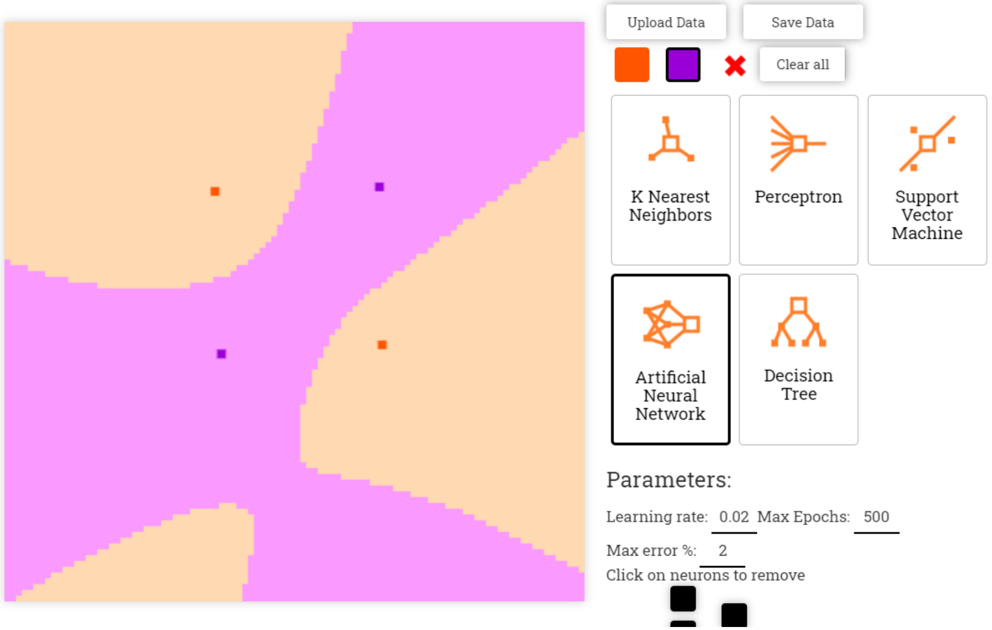


$$\frac{\partial loss}{\partial w2} = w3 * sigmoid'(h2\_in) * sigmoid(w1 * x + b)$$

XOR문제 해결!

https://ml-playground.com/#





4 Practice

Thank you