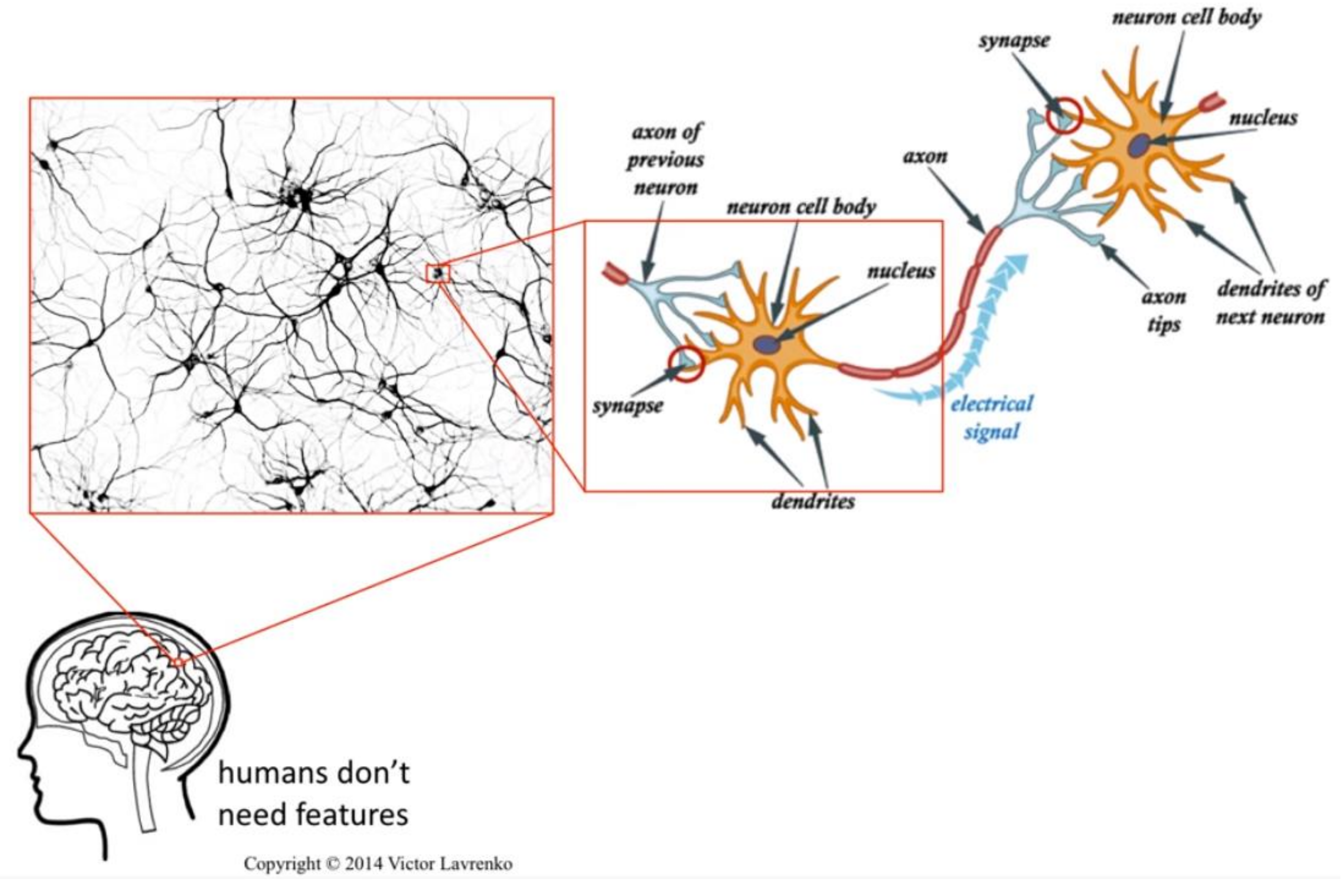
Perceptron & Gradient Descent

DONGDUK AI LEARING CREW_WEEK1

뉴런



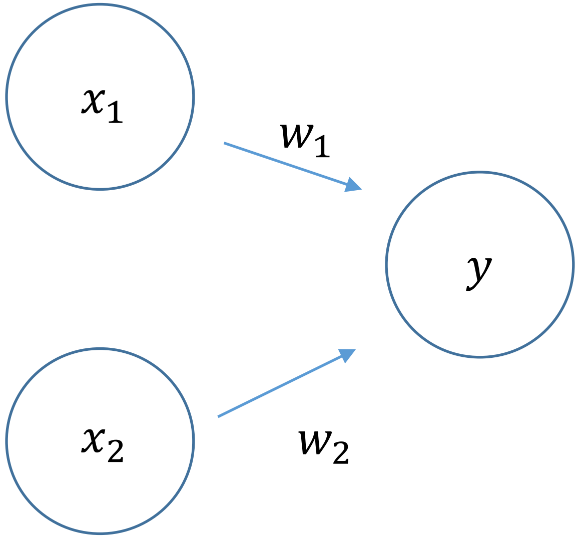
Dendrite : 이웃 뉴런에서 전기 신호를 받는다.

Synapse : 다른 뉴런과 Dendrite의 연결 부위에 있다.
전기신호의 세기를 재조정한다.

Soma (cell body) : Dendrite로부터 받은
여러 전기신호들을 모두 합친다.

Axon : Soma의 전위가 일정 이상이 되면
이웃 뉴런으로 전기 신호를 보낸다.

퍼셉트론



$$y = \begin{cases} 0, & w_1x_1 + w_2x_2 \le \theta \\ 1, & w_1x_1 + w_2x_2 > \theta \end{cases}$$

θ : 임계값, w_1, w_2 : 가중치(weight)

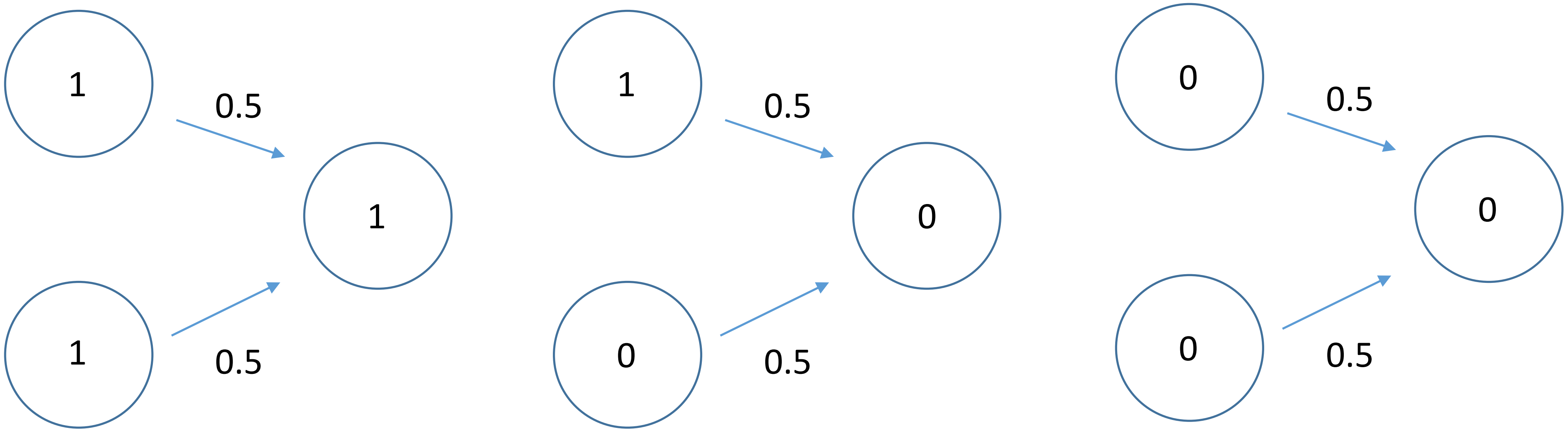
AND 게이트

x_1	x_2	y
True	True	True
True	False	False
False	True	False
False	False	False

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

전류가 흐른다 = 1 = True
전류가 흐르지 않는다 = 0 = False

AND 게이트



$0.5 \times 1 + 0.5 \times 1 = 1 > 0.7$ $\quad$ $0.5 \times 1 + 0.5 \times 0 = 0.5 \leq 0.7$ $\quad$ $0.5 \times 0 + 0.5 \times 0 = 0 \leq 0.7$

$w_1 = 0.5, w_2 = 0.5, \theta = 0.7$로 잡아보자

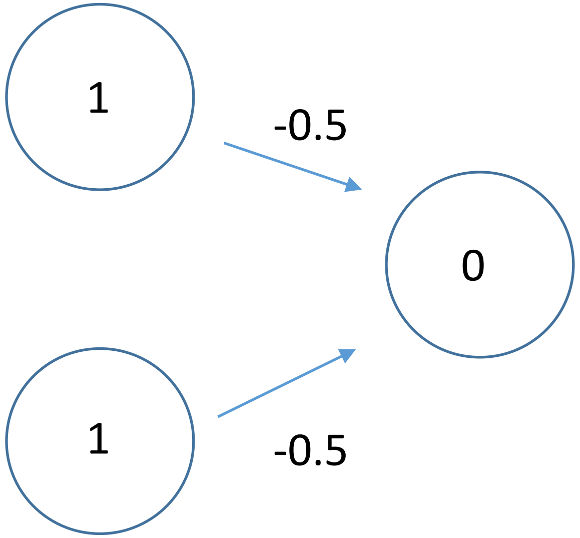
NAND 게이트

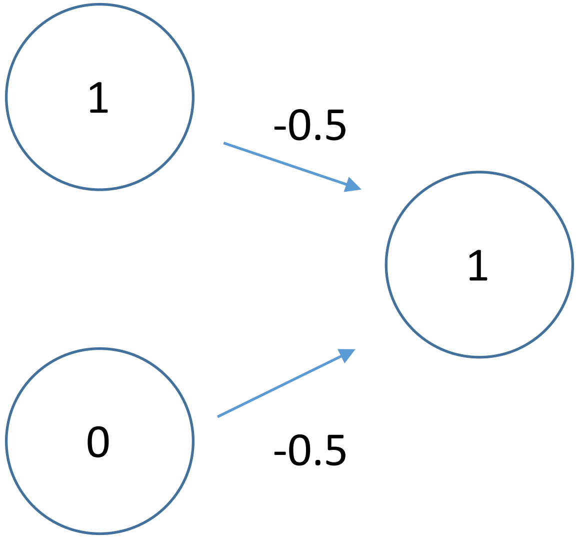
x_1	x_2	y
True	True	False
True	False	True
False	True	True
False	False	True

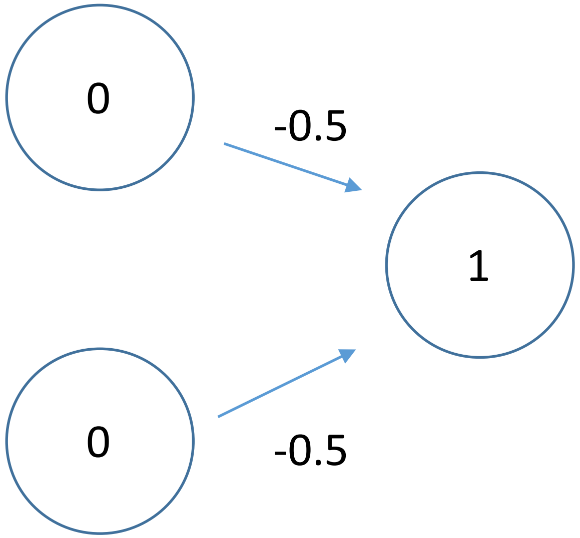
x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	1

전류가 흐른다 = 1 = True
전류가 흐르지 않는다 = 0 = False

NAND 게이트







$-0.5 \times 1 - 0.5 \times 1 = -1 \leq -0.7$

$-0.5 \times 1 - 0.5 \times 0 = -0.5 > -0.7$

$-0.5 \times 0 - 0.5 \times 0 = 0 > -0.7$

$w_1 = -0.5, w_2 = -0.5, \theta = -0.7$로 잡아보자

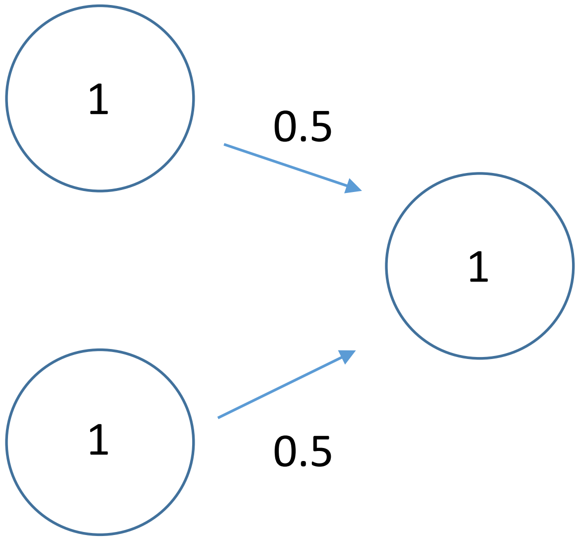
OR 게이트

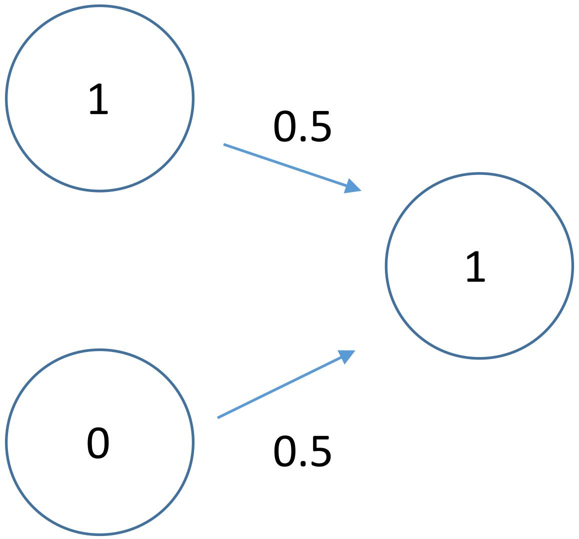
x_1	x_2	y
True	True	True
True	False	True
False	True	True
False	False	False

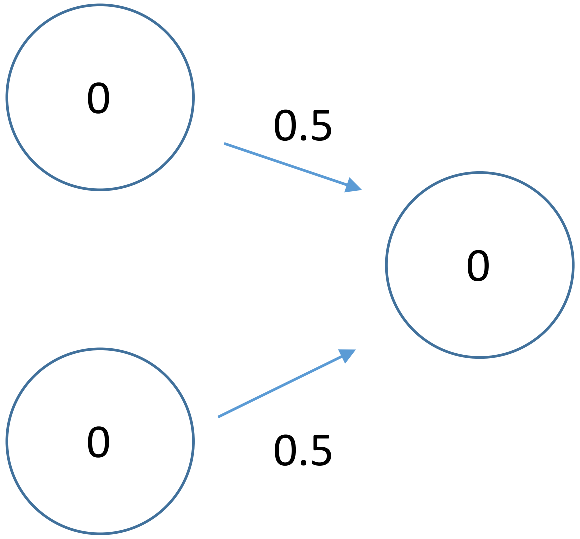
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

전류가 흐른다 = 1 = True
전류가 흐르지 않는다 = 0 = False

OR 게이트







$0.5 \times 1 + 0.5 \ \times 1 = 1 > 0.2$

$0.5 \times 1 + 0.5 \times 0 = 0.5 > 0.2$

$0.5 \times 0 + 0.5 \times 0 = 0 \leq 0.2$

$w_1 = 0.5, w_2 = 0.5, \theta = 0.2$로 잡아보자

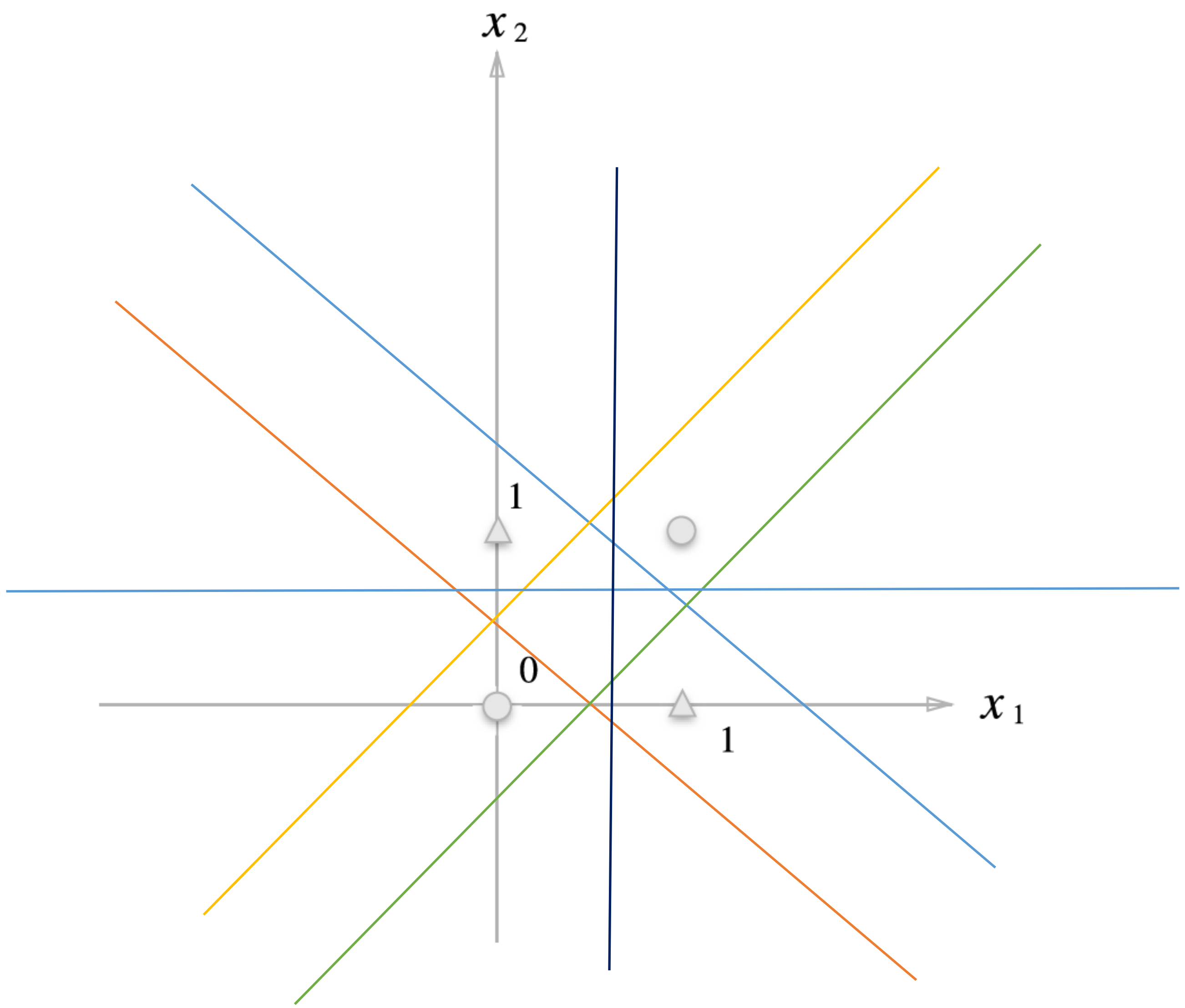
XOR 게이트

x_1	x_2	y
True	True	False
True	False	True
False	True	True
False	False	False

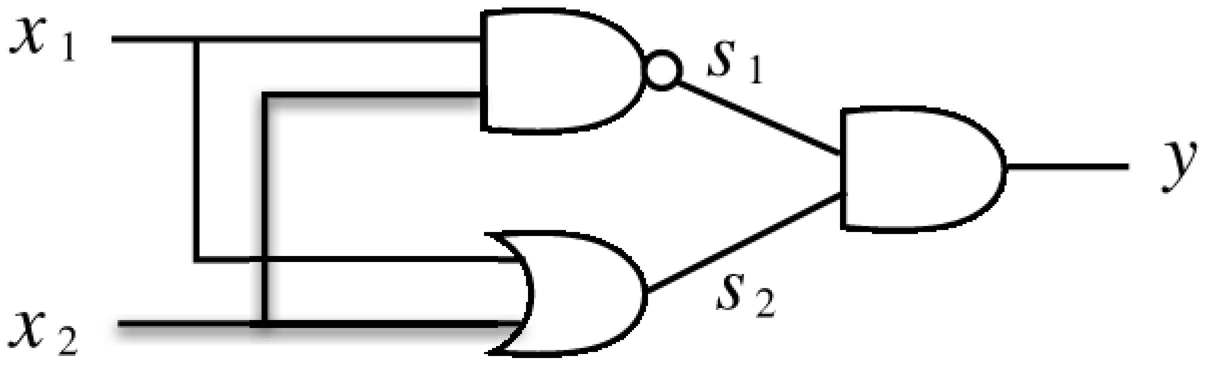
x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

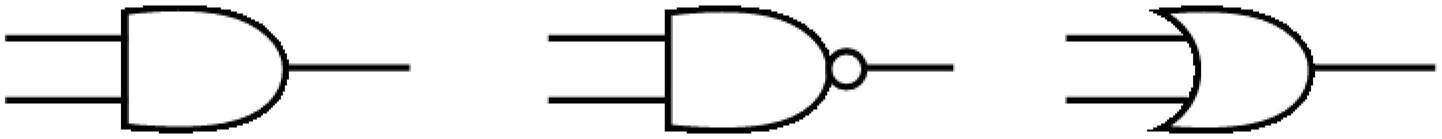
전류가 흐른다 = 1 = True
전류가 흐르지 않는다 = 0 = False

단층 퍼셉트론의 한계

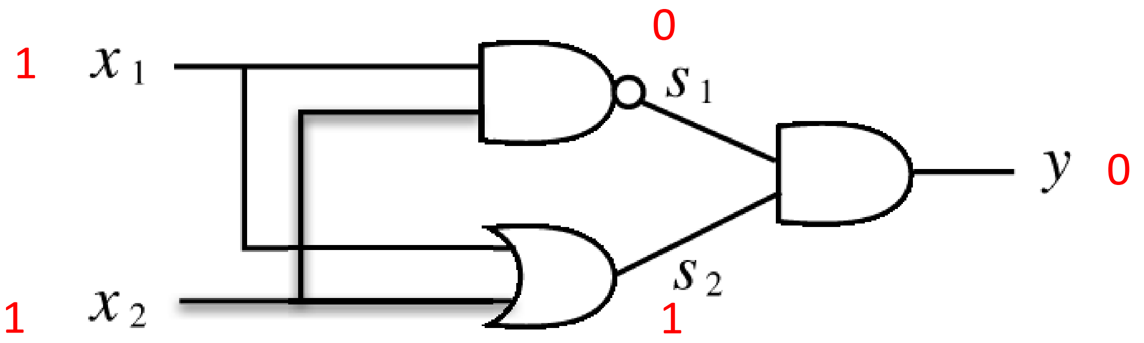


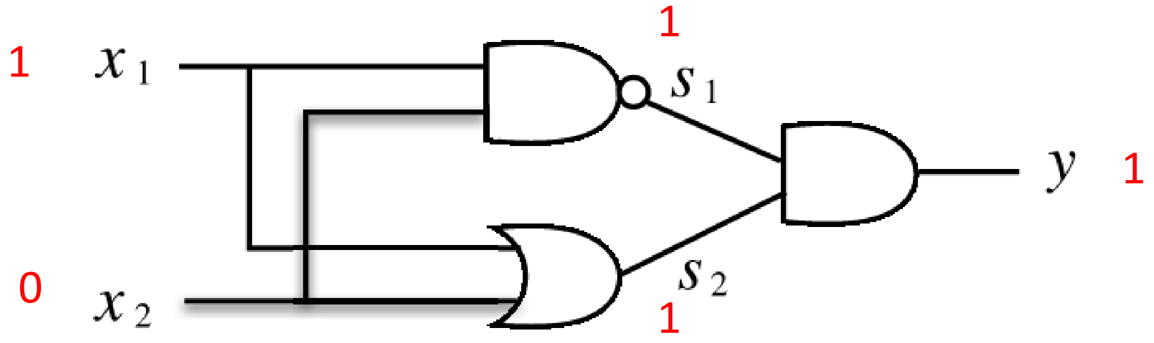
다층 퍼셉트론을 통한 XOR 게이트

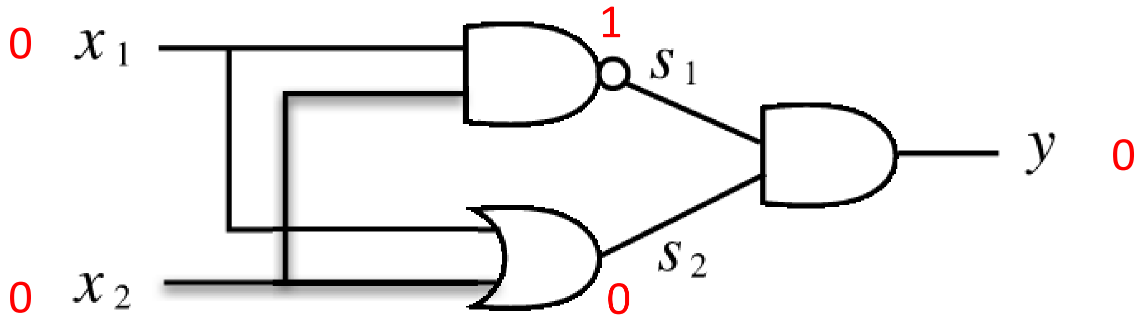




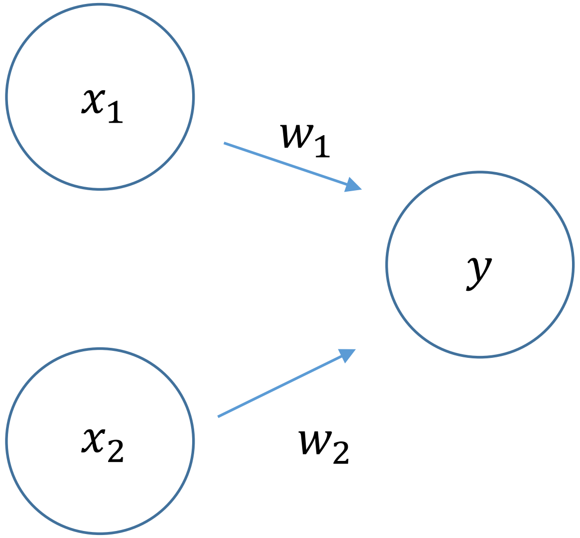
AND NAND OR

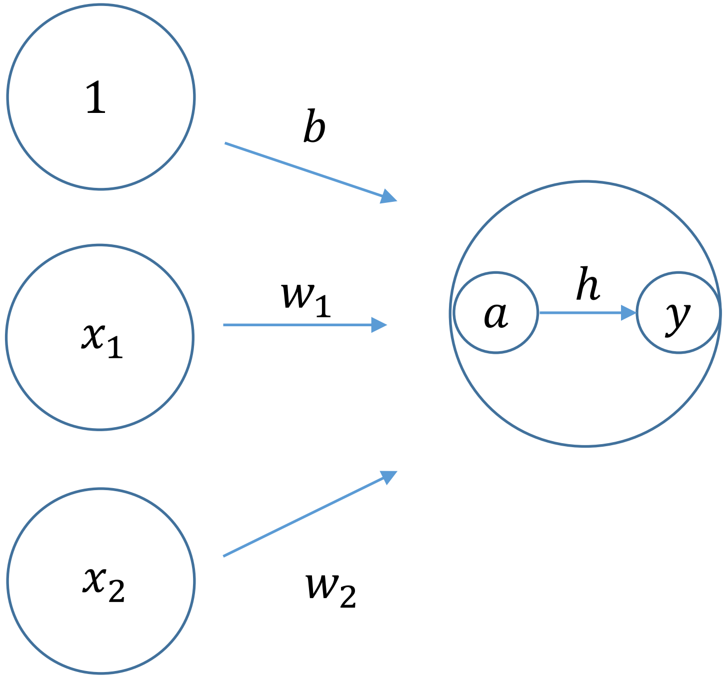

1
x1
0
s1
y
0
s2
1
1
x2


1
x1
1
s1
y
1
s2
1
0
x2


0
x1
1
s1
y
0
s2
0
0
x2

다층 퍼셉트론을 위한 개념





$$y = \begin{cases} 0, & w_1x_1 + w_2x_2 \le \theta \\ 1, & w_1x_1 + w_2x_2 > \theta \end{cases}$$

$$\Leftrightarrow \; y = \begin{cases} 0, & w_1x_1 + w_2x_2 + b \le 0 \\ 1, & w_1x_1 + w_2x_2 + b > 0 \end{cases}$$

$$\Leftrightarrow \; y = h(w_1x_1 + w_2x_2 + b)$$

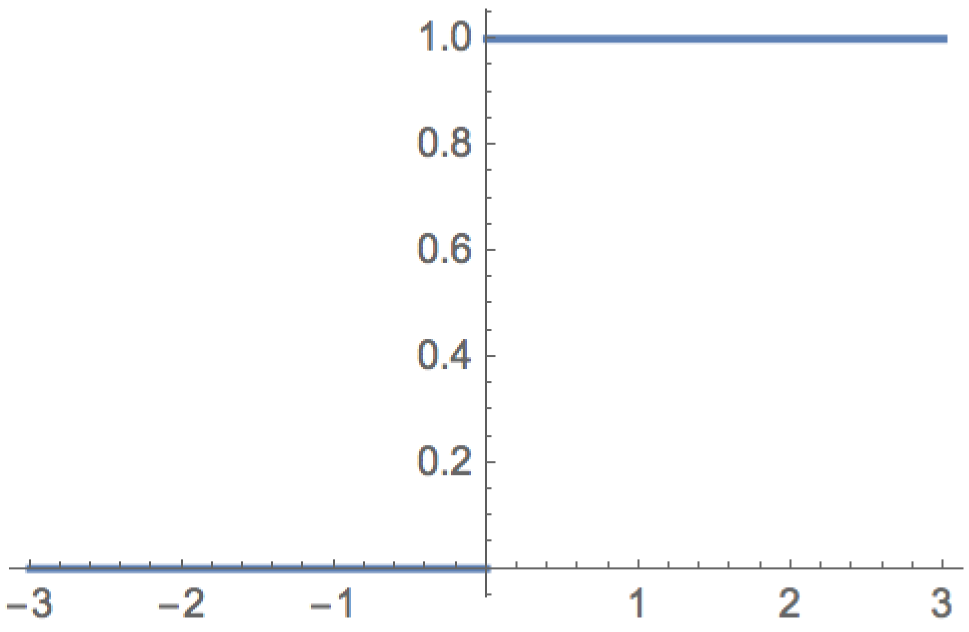
$b = -\theta$: 편향 (bias)

$h(x) = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$: Heaviside 함수

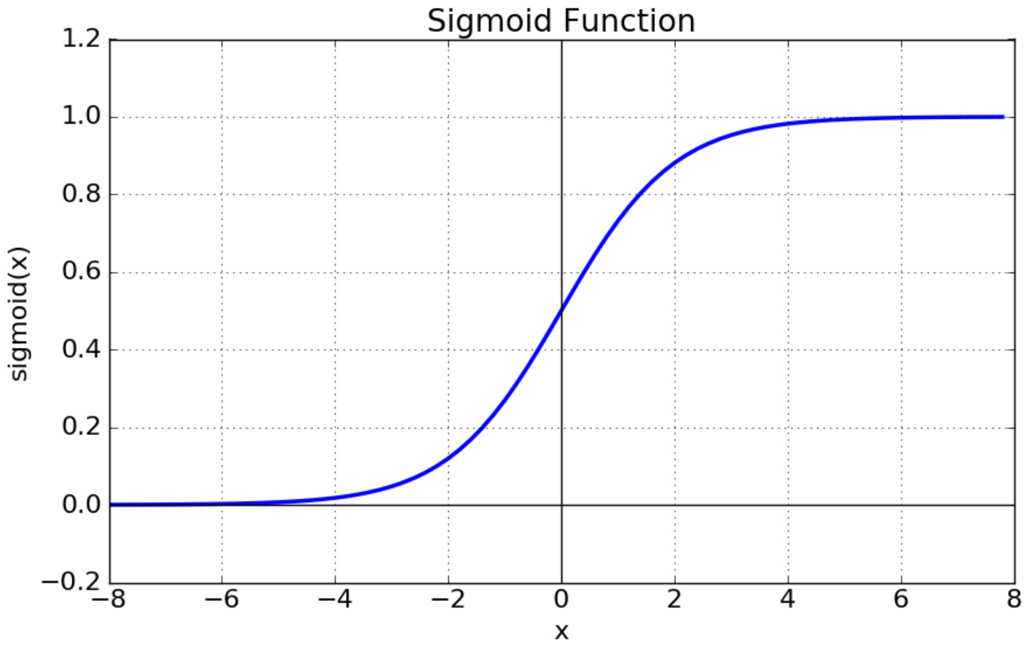
$a = w_1x_1 + w_2x_2 + b$

활성화 함수(Activation Function)

Heaviside 함수

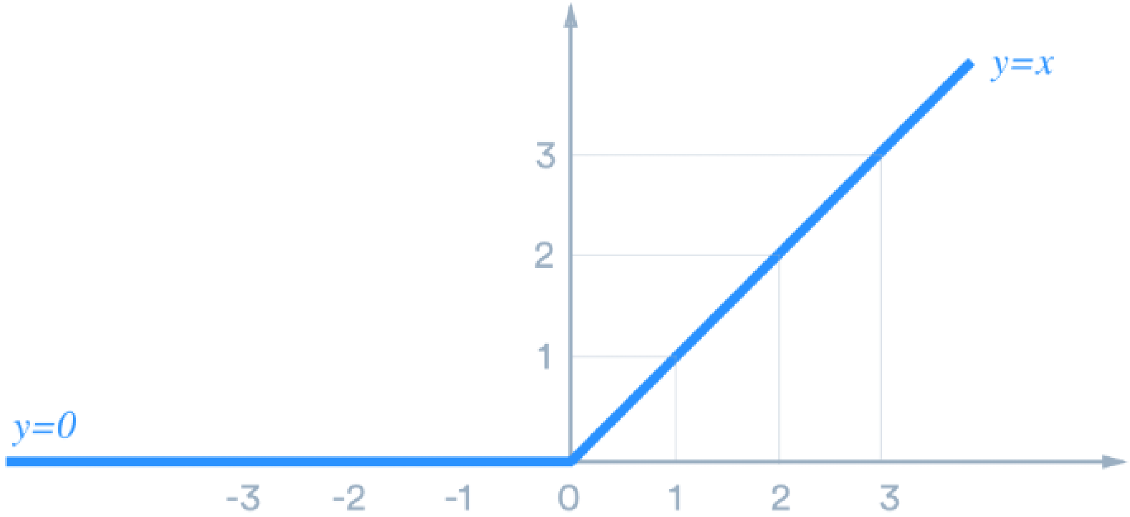


Sigmoid 함수 $f(x) = \frac{1}{1+e^{-x}}$

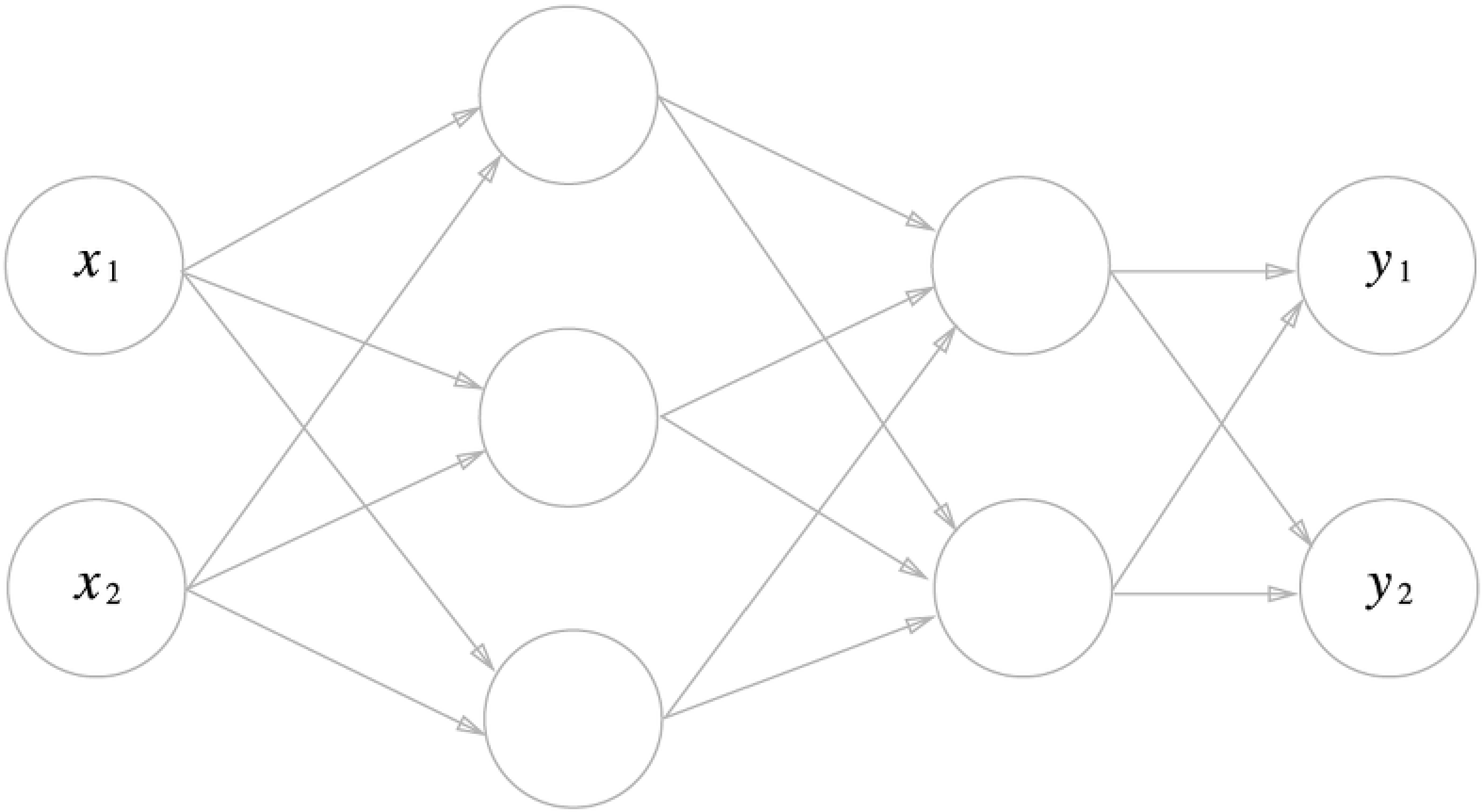


활성화 함수(Activation Function)

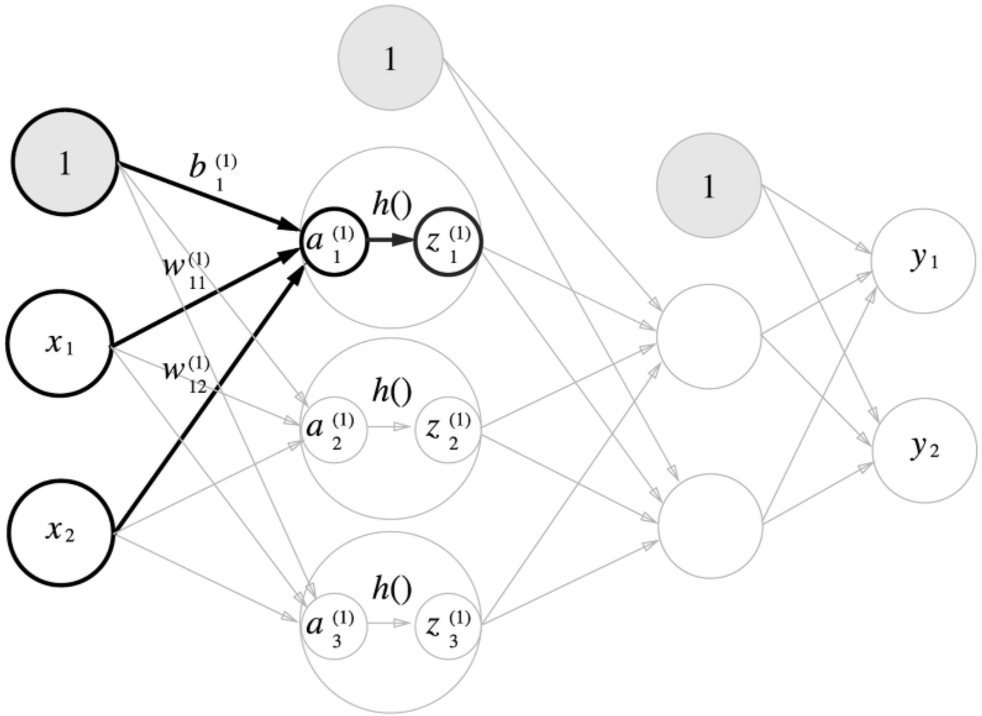
ReLU 함수



3층 신경망의 구조

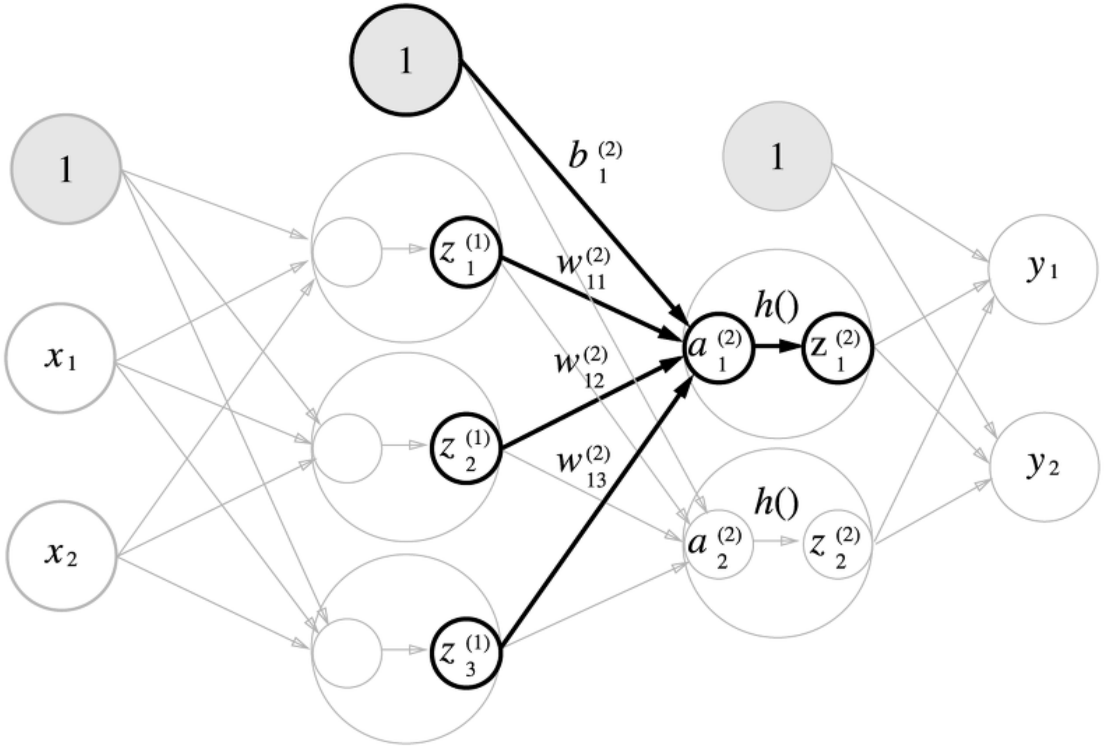


입력층에서 1층으로의 신호전달



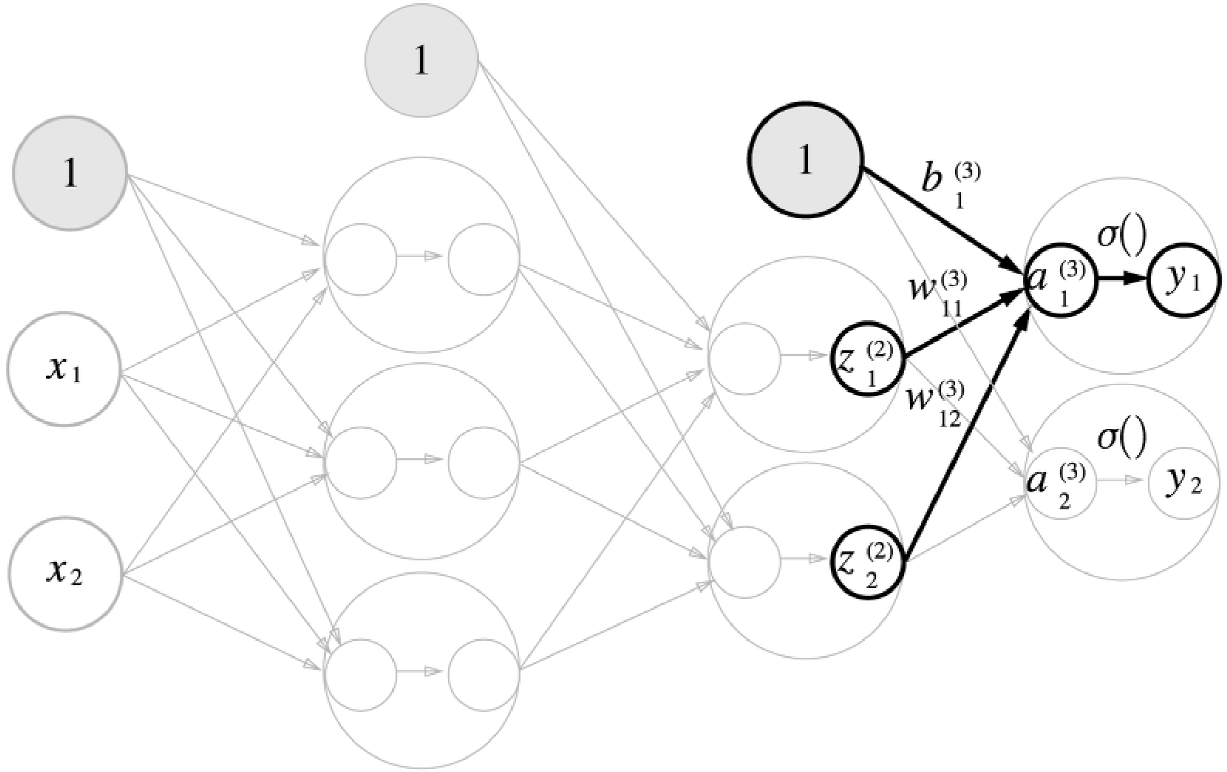
$$\begin{pmatrix} a_1^{(1)} & a_2^{(1)} & a_3^{(1)} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \end{pmatrix} + \begin{pmatrix} b_1^{(1)} & b_2^{(1)} & b_3^{(1)} \end{pmatrix}$$

1층에서 2층으로의 신호전달



$$\begin{pmatrix} a_1^{(2)} & a_2^{(2)} \end{pmatrix} = \begin{pmatrix} z_1^{(1)} & z_2^{(1)} & z_3^{(1)} \end{pmatrix} \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \\ w_{31}^{(2)} & w_{32}^{(2)} \end{pmatrix} + \begin{pmatrix} b_1^{(2)} & b_2^{(2)} \end{pmatrix}$$

2층에서 출력층으로의 신호전달

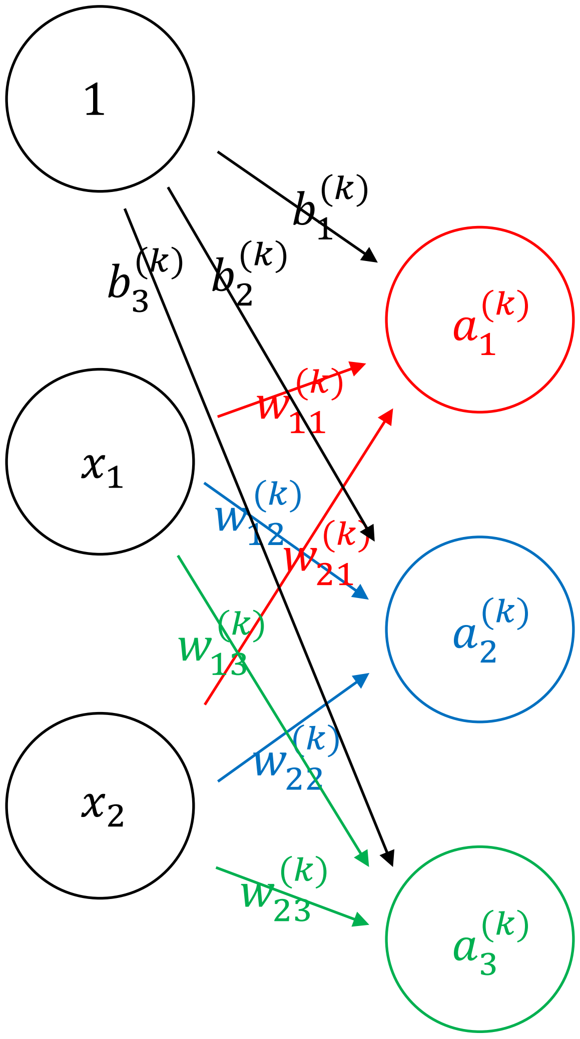


σ : softmax 함수

확률벡터로 변환하는 역할

$$\begin{pmatrix} a_1^{(3)} & a_2^{(3)} \end{pmatrix} = \begin{pmatrix} z_1^{(2)} & z_2^{(2)} \end{pmatrix} \begin{pmatrix} w_{21}^{(3)} & w_{22}^{(3)} \\ w_{31}^{(3)} & w_{32}^{(3)} \end{pmatrix} + \begin{pmatrix} b_1^{(3)} & b_2^{(3)} \end{pmatrix}$$

행렬과 퍼셉트론



$$W_{\text{입력, 출력}}^{(\text{층})}$$

$$a_1^{(k)} = w_{11}^{(k)} x_1 + w_{21}^{(k)} x_2 + b_1^{(k)}$$

$$a_2^{(k)} = w_{12}^{(k)} x_1 + w_{22}^{(k)} x_2 + b_2^{(k)}$$

$$a_3^{(k)} = w_{13}^{(k)} x_1 + w_{23}^{(k)} x_2 + b_3^{(k)}$$

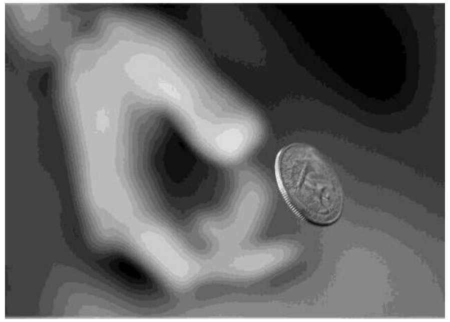
$$\Updownarrow$$

$$\begin{pmatrix} a_1^{(k)} & a_2^{(k)} & a_3^{(k)} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} w_{11}^{(k)} & w_{12}^{(k)} & w_{13}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} & w_{23}^{(k)} \end{pmatrix} + \begin{pmatrix} b_1^{(k)} & b_2^{(k)} & b_3^{(k)} \end{pmatrix}$$

확률벡터

$(p_1, p_2, p_3 \cdots, p_n)$: 확률벡터

$$\Leftrightarrow \quad p_k \geqslant 0, \qquad \sum_{k=1}^{n} p_k = 1$$



$(\frac{1}{2}, \frac{1}{2})$

$(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$

Softmax 변환

$$(a_1, a_2, a_3, \cdots, a_n)$$

e^x 을 적용해서 양수로

$$(e^{a_1}, e^{a_2}, e^{a_3}, \cdots, e^{a_n})$$

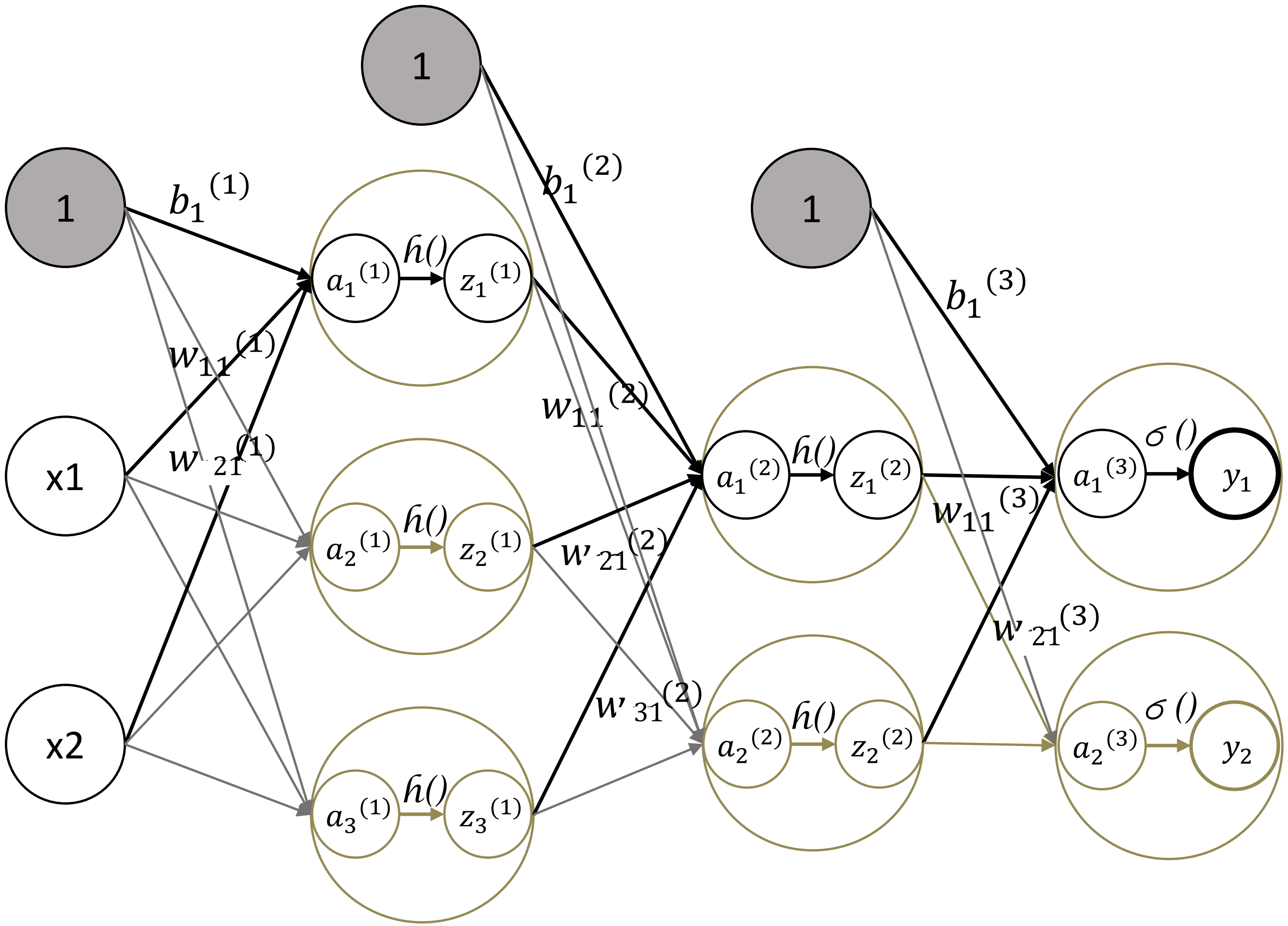
normalize

$$\left(\frac{e^{a_1}}{e^{a_1}+e^{a_2}+\cdots+e^{a_n}}, \frac{e^{a_2}}{e^{a_1}+e^{a_2}+\cdots+e^{a_n}}, \cdots, \frac{e^{a_n}}{e^{a_1}+e^{a_2}+\cdots+e^{a_n}}\right)$$

$(a_1, a_2, a_3, \cdots, a_n)$ 과 $(a_1 + C, a_2 + C, a_3 + C, \cdots, a_n + C)$ 의 softmax값은 동일하다.

이를 이용하여 overflow를 방지할 수 있다.

3층 신경망의 신호전달



Affine → Sigmoid → Affine → Sigmoid → Affine → Softmax

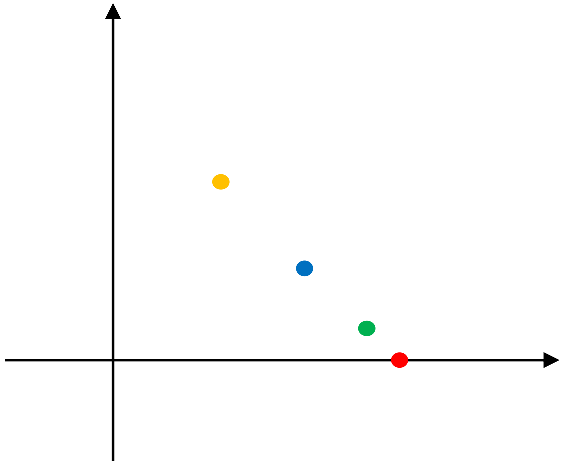
손실함수(Loss Function)

인공신경망에 손실함수라는 벌점을 준다.

인공신경망의 성능이 좋을수록 손실함수 값이 낮고 성능이 나쁠수록 손실함수 값이 높다.

머신러닝은 데이터를 통해 손실함수값을 낮추는 과정이다.

예) A 기상대는 내일 비올 확률을 90%, B 기상대는 60%,
C 기상대는 30%로 예상했다고 하자.
실제로 비가 왔을 경우 점 (1,0)과 점 (0.9, 0.1), (0.6,0.4),
(0.3,0.7)까지의 거리로 벌점을 줄 수 있다.
A 기상대 , B 기상대 모두 맞췄지만 A 기상대의 정확도가
더 높고 따라서 벌점은 낮다. C 기상대는 틀렸으므로
벌점이 가장 높다.



수치미분

미분계수

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

를 충분히 작은 h (예를 들면 10^{-4})를 잡아

$$\frac{f(x+h) - f(x)}{h}$$

로 근사시킨다.

수치미분

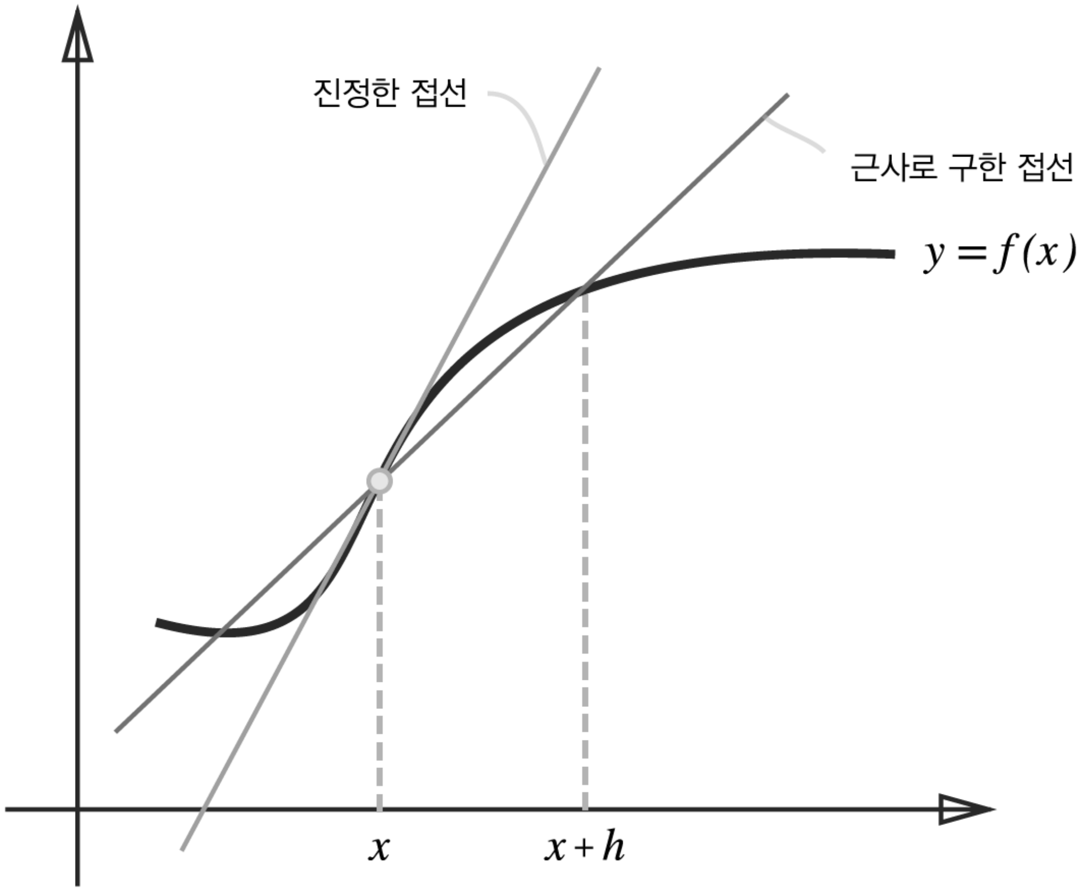
등식

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}=\lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h}$$

이 성립하므로 미분계수를

$$\frac{f(x+h)-f(x-h)}{2h}$$

로 근사시킬수 있는데 오차가 더 적다.



일변수 미분 vs 다변수 미분

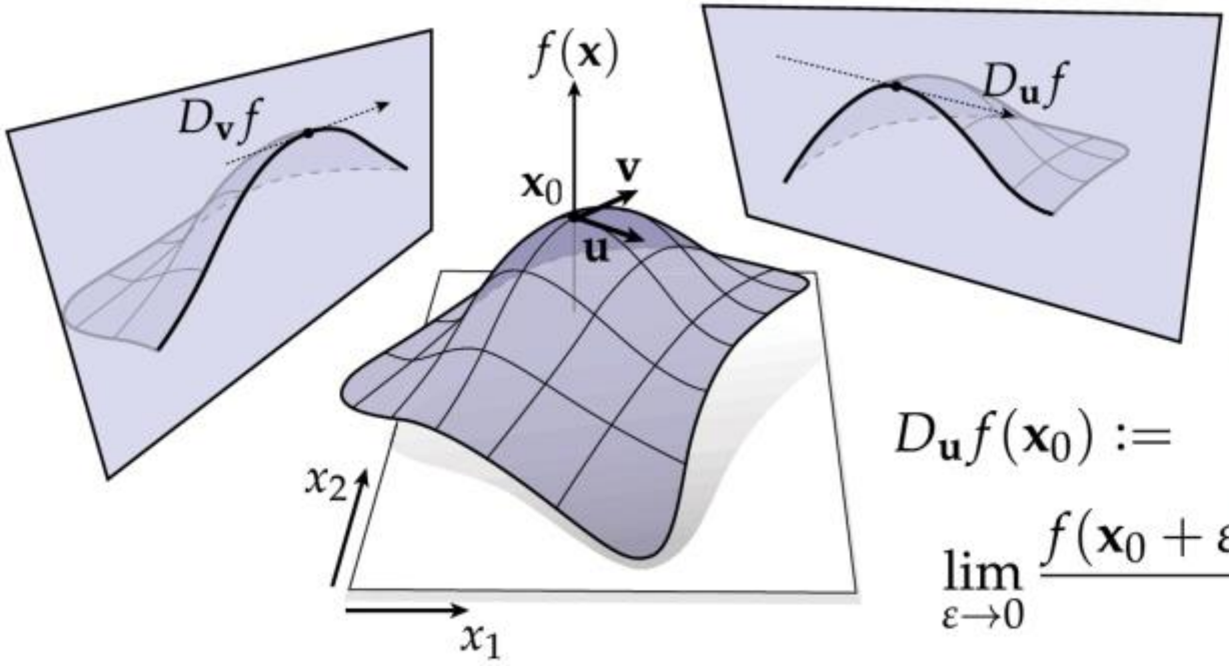
일변수 미분 $f'(x)$는 함수 f를 점 x에서 미분

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

다변수 방향미분 $D_{\vec{v}} f(\mathbf{x})$함수 f를 점 $\mathbf{x}$에서 미분

$$D_{\vec{v}} f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\vec{v}) - f(\mathbf{x})}{h}$$

방향미분



take a small step along u

$$D_{\mathbf{u}}f(\mathbf{x}_0) := \lim_{\varepsilon \to 0} \frac{f(\mathbf{x}_0 + \varepsilon \mathbf{u}) - f(\mathbf{x}_0)}{\varepsilon}$$

편미분

각 축에 대한 방향 미분을 편미분이라 한다.

$$D_{(1,0\cdots,0)}f(\mathbf{x}) = \lim_{h\to 0}\frac{f(\mathbf{x}+h(1,0\cdots,0))-f(\mathbf{x})}{h}$$

$$\vdots$$

$$D_{(0,0\cdots,1)}f(\mathbf{x}) = \lim_{h\to 0}\frac{f(\mathbf{x}+h(0,0\cdots,1))-f(\mathbf{x})}{h}$$

방향미분 축에 해당하는 변수로 나머지 변수는 상수처럼 취급하고 일변수 미분을 한 계산 결과와 동일하다.

편도함수를 $\frac{\partial f}{\partial \mathbf{x}_i}$. 쓴다.

Gradient

편도함수를 변수 순서로 묶어 놓은 벡터를 gradient라고 한다.

$$\nabla f = (\frac{\partial f}{\partial \mathbf{x}_1}, \frac{\partial f}{\partial \mathbf{x}_2}, \cdots, \frac{\partial f}{\partial \mathbf{x}_n})$$

모든 방향 미분계수를 gradient와 방향을 내적하여 구할 수 있다.

$$D_{\vec{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \circ \vec{v}$$

접평면의 방정식 :

$$z = \nabla f(x_0, y_0) \circ ((x, y) - (x_0, y_0)) + f(x_0, y_0)$$

방향미분계수의 최대,최소

함수 f와 점 $\mathbf{x}$는 고정되어 있고 방향의 크기는 1이라 하자.

$$D_{\vec{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \circ \vec{v} = \|\nabla f(\mathbf{x})\| \|\vec{v}\| \cos\theta = \|\nabla f(\mathbf{x})\| \cos\theta$$

(i) 방향 미분이 가장 커지는 방향은 gradient 방향이고 값은 gradient의 크기이다.

(ii) 방향 미분이 가장 작아지는 방향은 gradient 반대 방향이고 값은 마이너스 gradient의 크기이다

(iii) 방향 미분이 0이 되는 방향은 gradient와 수직인 방향이다.

Gradient

편도함수를 변수 순서로 묶어 놓은 벡터를 gradient라고 한다.

$$\nabla f = \left(\frac{\partial f}{\partial \mathbf{x}_1}, \frac{\partial f}{\partial \mathbf{x}_2}, \cdots, \frac{\partial f}{\partial \mathbf{x}_n}\right)$$

모든 방향 미분계수를 gradient와 방향을 내적하여 구할 수 있다.

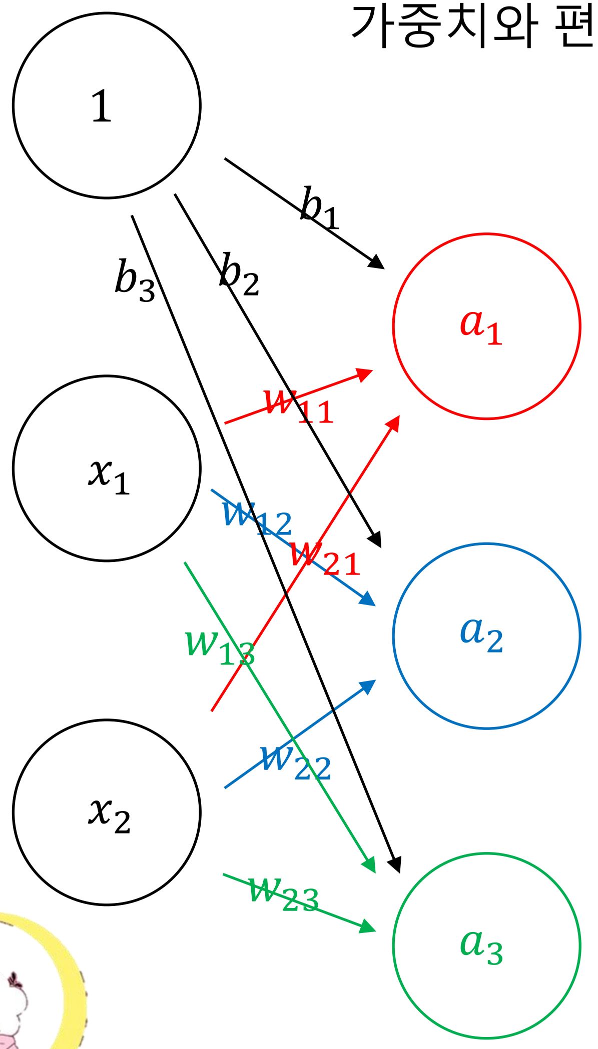
$$D_{\vec{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \circ \vec{v}$$

접평면의 방정식 :

$$z = \nabla f(x_0, y_0) \circ ((x, y) - (x_0, y_0)) + f(x_0, y_0)$$

손실함수의 변수

가중치와 편향이 손실함수의 변수, 입력 데이터는 손실함수의 계수



라벨이 (0,0,1)이라면 손실함수는

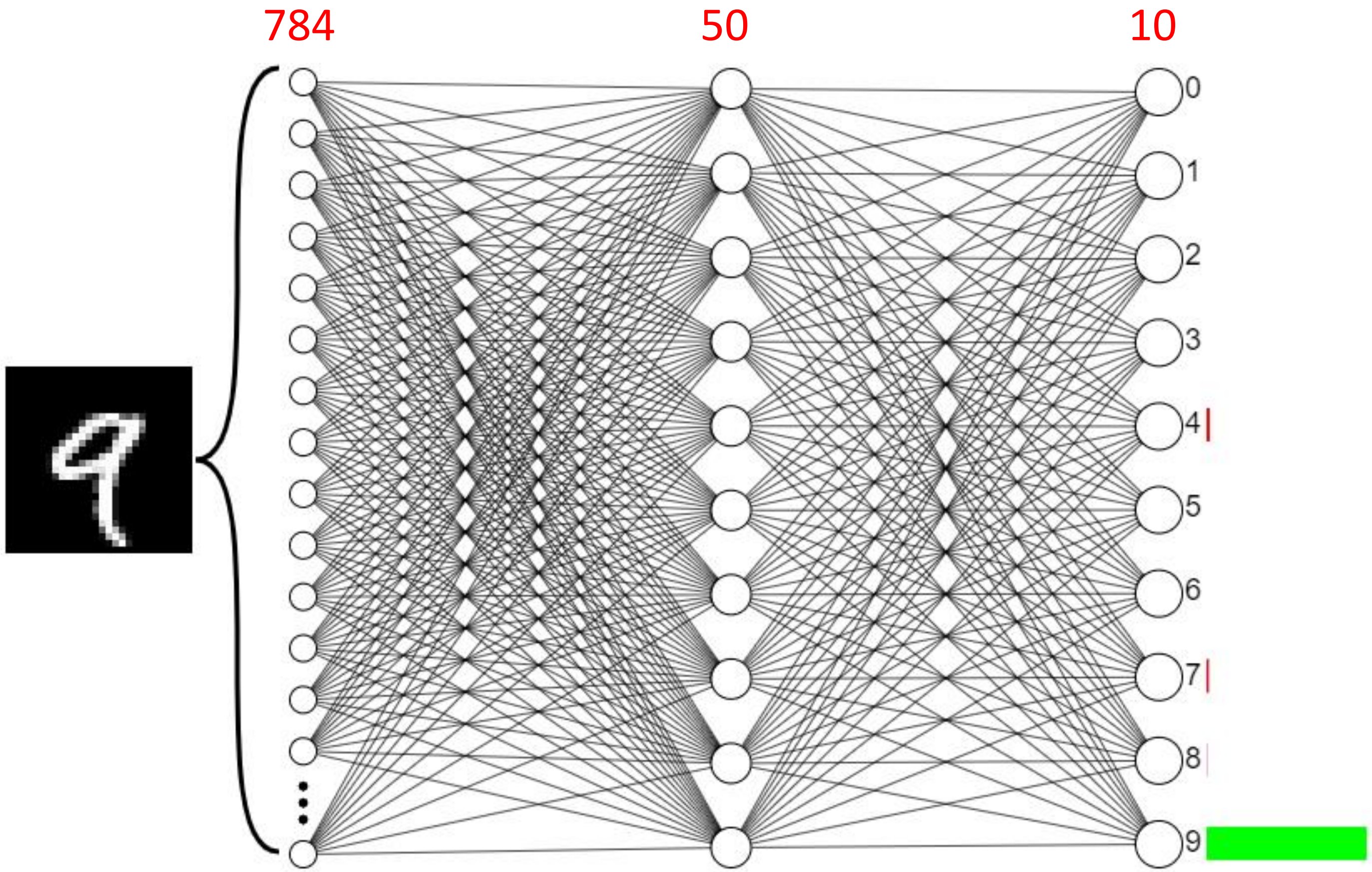
$$L\left(\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}, \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}\right)$$

$$= -\log \frac{e^{w_{13}x_1 + w_{23}x_2 + b_3}}{e^{w_{11}x_1 + w_{21}x_2 + b_1} + e^{w_{12}x_1 + w_{22}x_2 + b_2} + e^{w_{13}x_1 + w_{23}x_2 + b_3}}$$

$w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}$와 b_1, b_2, b_3가 변수고 x_1, x_2는 계수

손실함수의 변수

가중치와 편향이 손실함수의 변수, 입력 데이터는 손실함수의 계수



손실함수 변수개수는

$$784 \times 50 + 50 + 50 \times 10 + 10 = 39{,}760$$

경사하강법 (Gradient Descent)

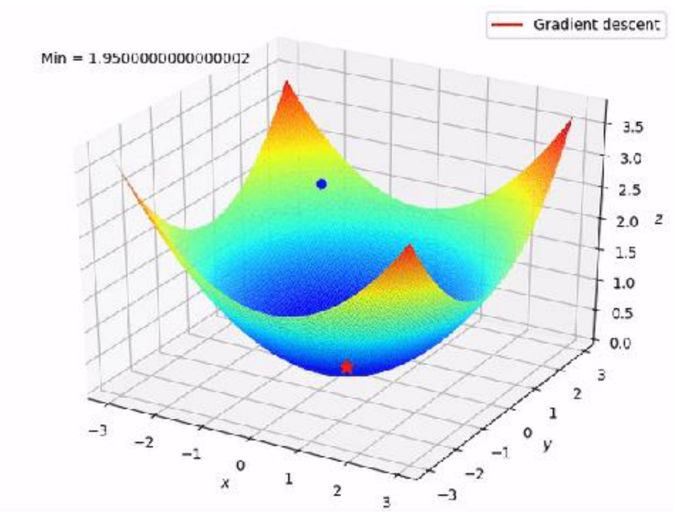
다변수 함수 $f : \mathbb{R}^n \to \mathbb{R}$가 점 $\mathbf{x}$에서 가장 감소하는 방향은 gradient의 반대 방향 $-\nabla f(\mathbf{x})$이다.

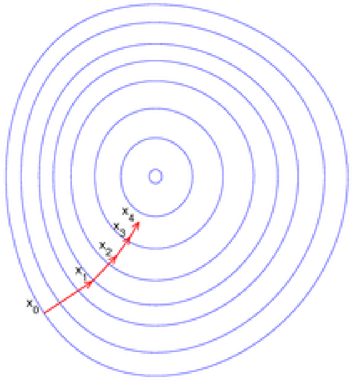
경사하강법은 gradient의 반대방향으로 한 발자국씩 내딛으며 함수값을 낮추는 방법이다.

점화식

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \nabla f(\mathbf{x}_n)$$

으로 주어진다. 보폭은 $\eta \|\nabla f(\mathbf{x}_n)\|$.



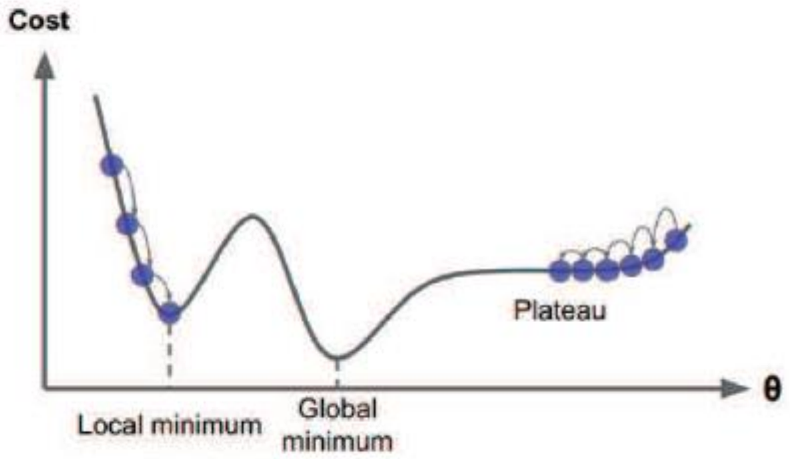


경사하강법의 문제점

극소점이나 안장점 근처에서는 $\nabla f(\mathbf{x}_n)$크기가 작아져서 보폭도 작아진다.

따라서, 점 $\mathbf{x}_n$이 최소점이 아니라 극소점 또는 안장점에 안착할 수도 있는 문제점이 있다.

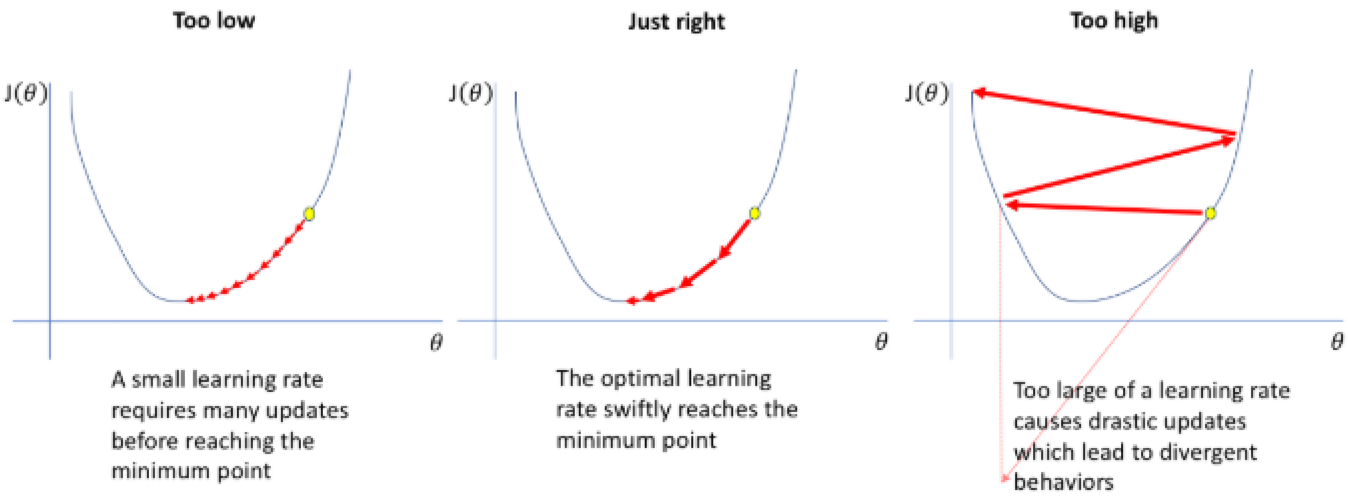
초기위치에 따라 결과가 달라진다.



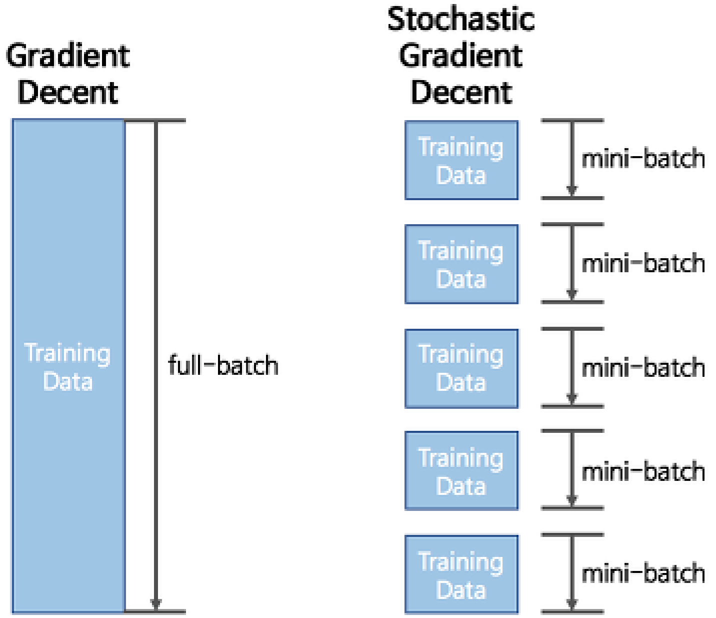
Learning Rate

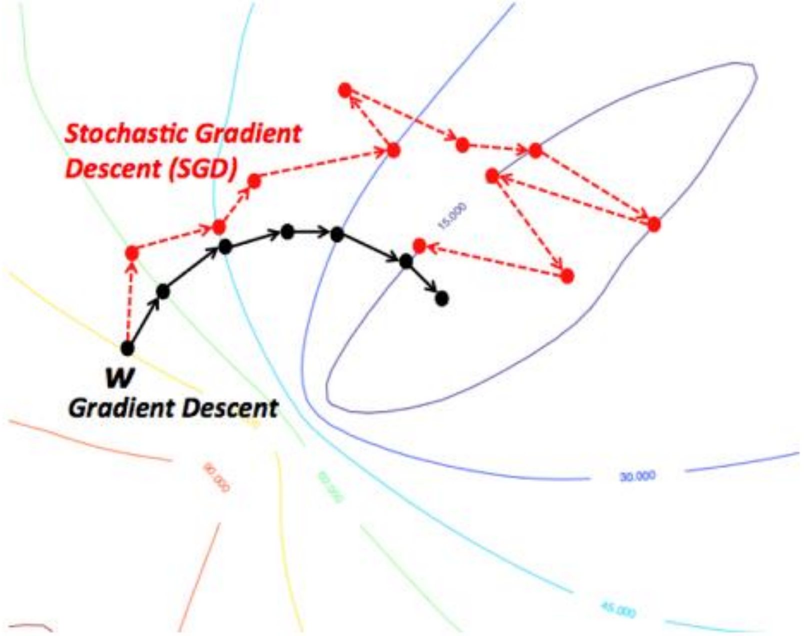
learning rate가 작으면 목적지 근처까지 못 가거나 많은 걸음을 걸어야 목적지 근처까지 간다.

learning rate가 크면 무질서하게 움직이며 발산한다.

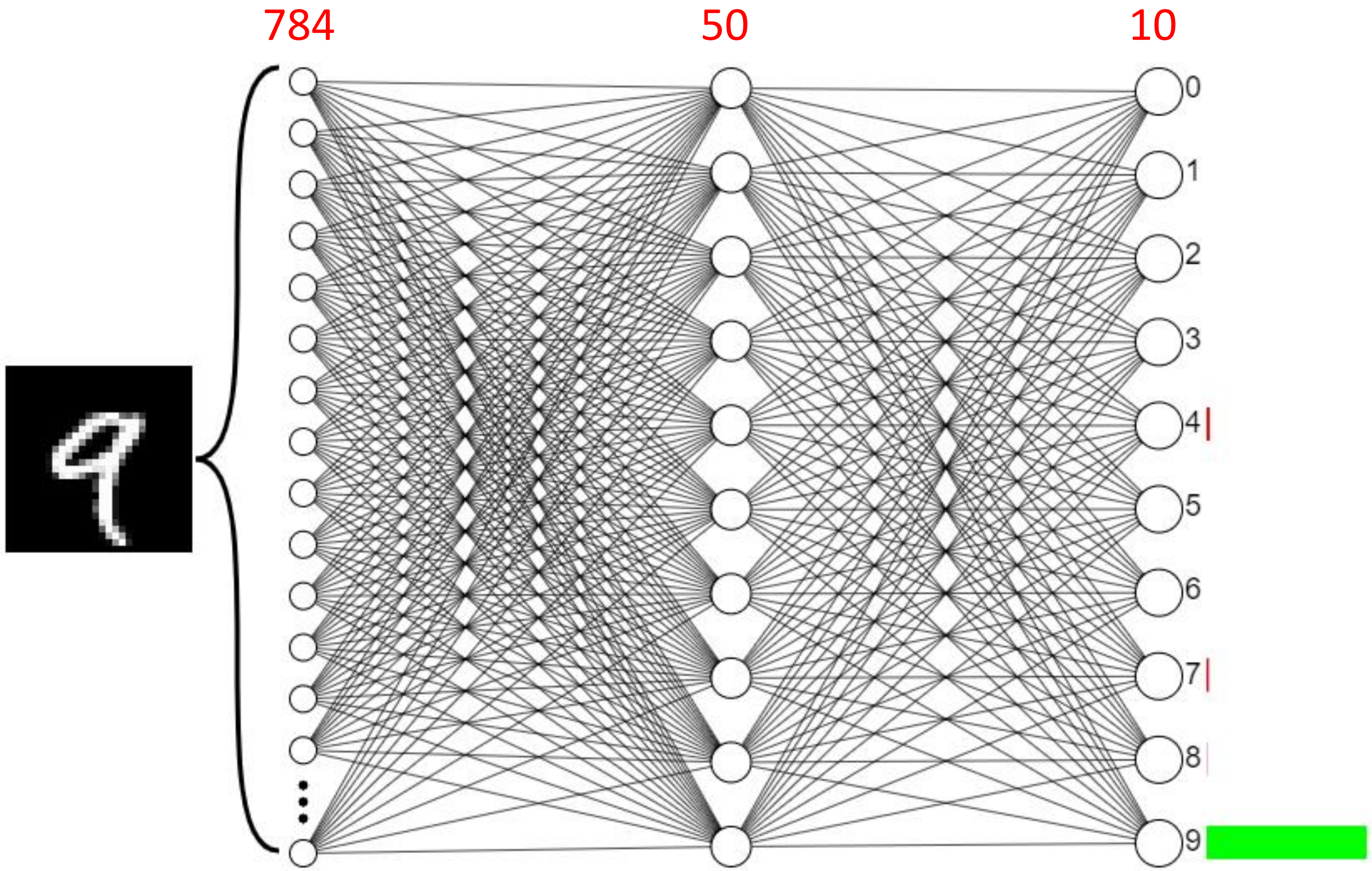


Stochastic Gradient Descent





수치미분을 통한 손글씨 학습



1. 2층 신경망, 뉴런의 개수는 784, 50, 10

2. 손실 함수는 교차 엔트로피, 변수개수는
(변수개수 : $784 \times 50 + 50 + 50 \times 10 + 10 = 39{,}760$)

3. learning rate $l = 0.1$,학습회수=10,000

4. 학습회수만큼 반복하여 $-l \cdot \nabla f$을 더하여 weight와 bias를 업데이트

5. 실행시키고 며칠 기다림

수치미분 vs 역전파

수치미분을 이용한 학습에는 $39,760 \times 10,000$번의 수치미분 계산이 필요
수치미분하는 함수는 affine함수, sigmoid함수, softmax, 교차 엔트로피를 합성한 함수
수치미분을 통한 학습은 계산 비용이 너무 크다.

역전파는 사람이 Affine층, Sigmoid층, SoftMax층, 손실함수층에서 각각 미분공식을 이용하여 직접 미분한 후 chain rule을 써서 전체 미분을 구해 코딩하는 알고리즘
컴퓨터가 가장 빨리 할 수 있는 계산은 덧셈과 곱셈인데 각 층을 공식으로 미분해보면 덧셈과 곱셈으로 표현된다.