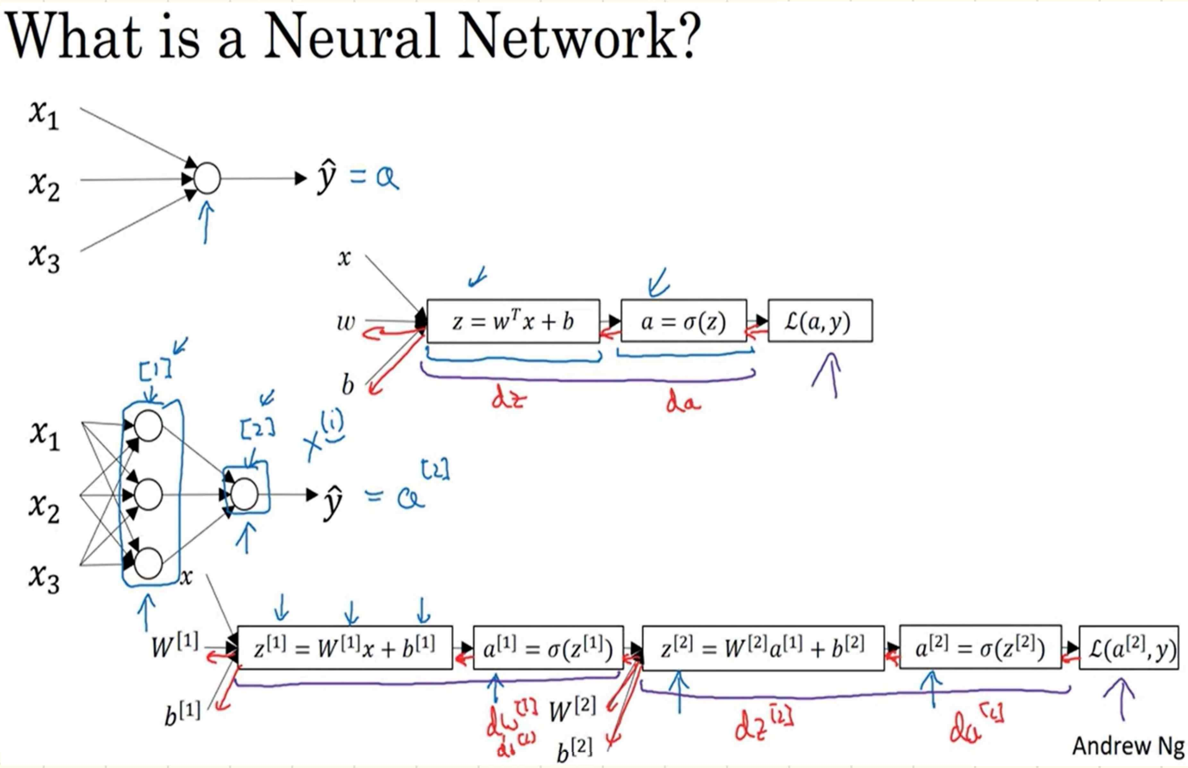
[3-1. Shallow Neural Network]

<Neural Networks Overview>

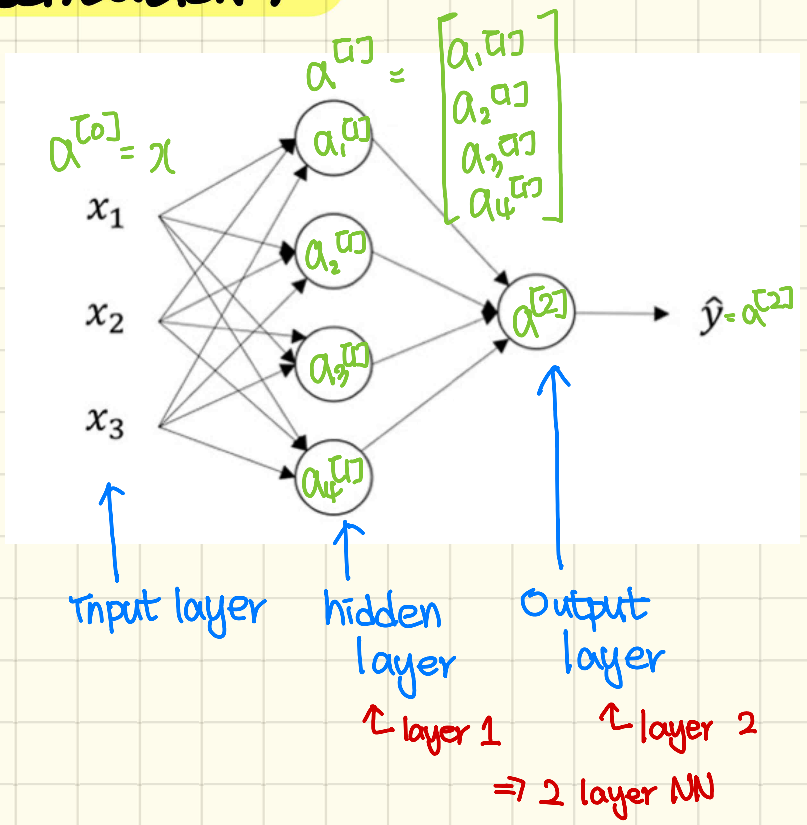


<Neural Network Representation>

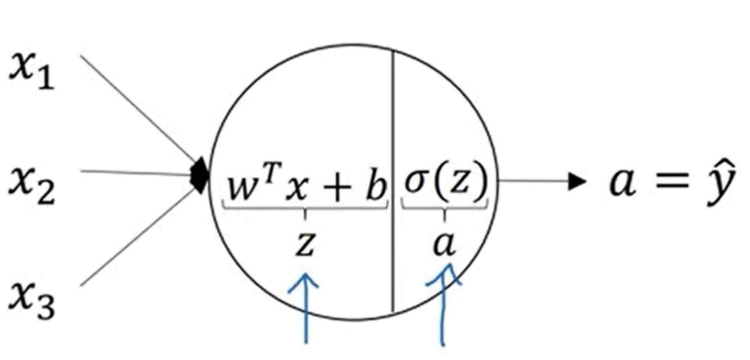
* Input layer(입력층) : $a^{[0]}$
  Hidden layer (은닉층) : $a_n^{[1]}$
  Output layer (출력층)

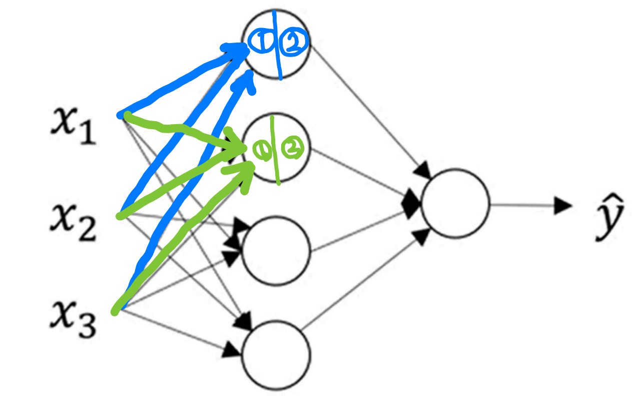
*신경망의 층의 개수를 셀 때는 입력층은 제외한다

∴ 오른쪽 사진은 2-layer Neural Network



<Computing a Neural Network's Output>





$z = w^T x + b$

$a = \sigma(z)$

* 파란색 표시

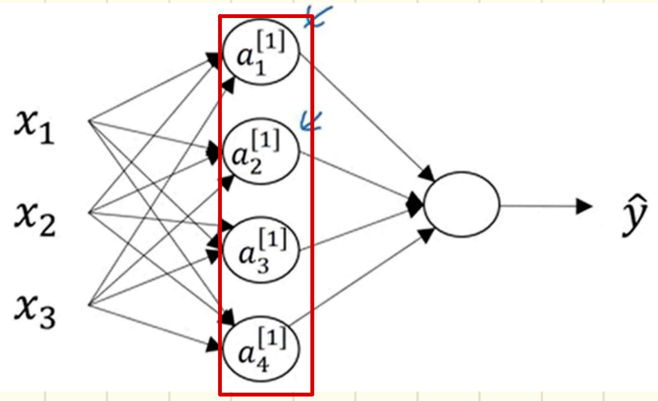
① $z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$

② $a_1^{[1]} = \sigma(z_1^{[1]})$

* 연두색 표시

① $z_2^{[1]} = w_2^{[1]T} + b_2^{[1]}$

② $a_2^{[1]} = \sigma(z_2^{[1]})$

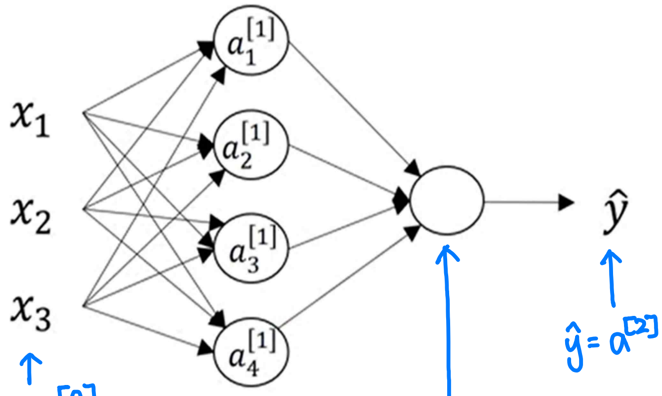


$a^{[1]} = \sigma(z^{[1]})$

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \underbrace{\underbrace{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}_{(4,3)}}_{W^{[1]}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{(3,1)} + \underbrace{\underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}_{(4,1)}}_{b^{[1]}} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix}$$

Made with Goodnotes



$x = a^{[0]}$

$\hat{y} = a^{[2]}$

$w^{[2]}$, $b^{[2]}$
(1,4) (1,1)

※ $a_i^{[l]}$
- l : 몇 번째 층인지
- i : 해당 층에서 몇번째 노드인지

Given input x:

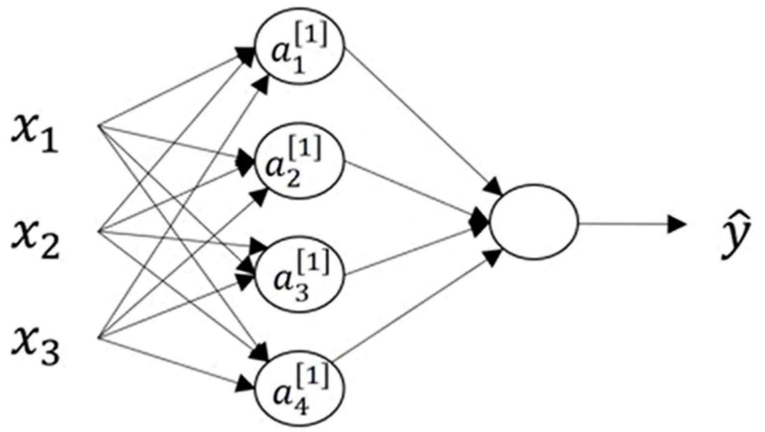
$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{\cancel{x}^{\,a^{[0]}}} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \sigma(\underset{(4,1)}{z^{[1]}})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,4)}{W^{[2]}} \underset{(4,1)}{a^{[1]}} + \underset{(1,1)}{b^{[2]}}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma(\underset{(1,1)}{z^{[2]}})$$

<Vectorizing Across Multiple Examples>



$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

$x \longrightarrow a^{[2]} = \hat{y}$

m개의 training set 으로 확장

$\Rightarrow x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$

...

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$

∴ $a^{[i](j)}$
- i : 몇 번째 층인지
- j : 몇 번째 훈련샘플인지

$\Rightarrow$ for $i=1$ to m,

$z^{[1](i)} = w^{[1]}x^{(i)} + b^{[1]}$

$a^{[1](i)} = \sigma(z^{[1](i)})$

$z^{[2](i)} = w^{[2]}x^{(i)} + b^{[2]}$

$a^{[2](i)} = \sigma(z^{[2](i)})$

Made with Goodnotes

for i = 1 to m:

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$\xrightarrow{\text{Vectorizing}}$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{(n_x, m)}, \quad Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \underbrace{\begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}}_{\text{trainig unit}} \Big\} \text{hidden units}$$

<Explanation for Vectorized Implementation>

$$z^{[1](1)} = w^{[1]}x^{(1)} + \cancel{b^{[1]}}, \quad z^{[1](2)} = w^{[1]}x^{(2)} + \cancel{b^{[1]}}, \quad z^{[1](3)} = w^{[1]}x^{(3)} + \cancel{b^{[1]}}$$

0이라고 가정

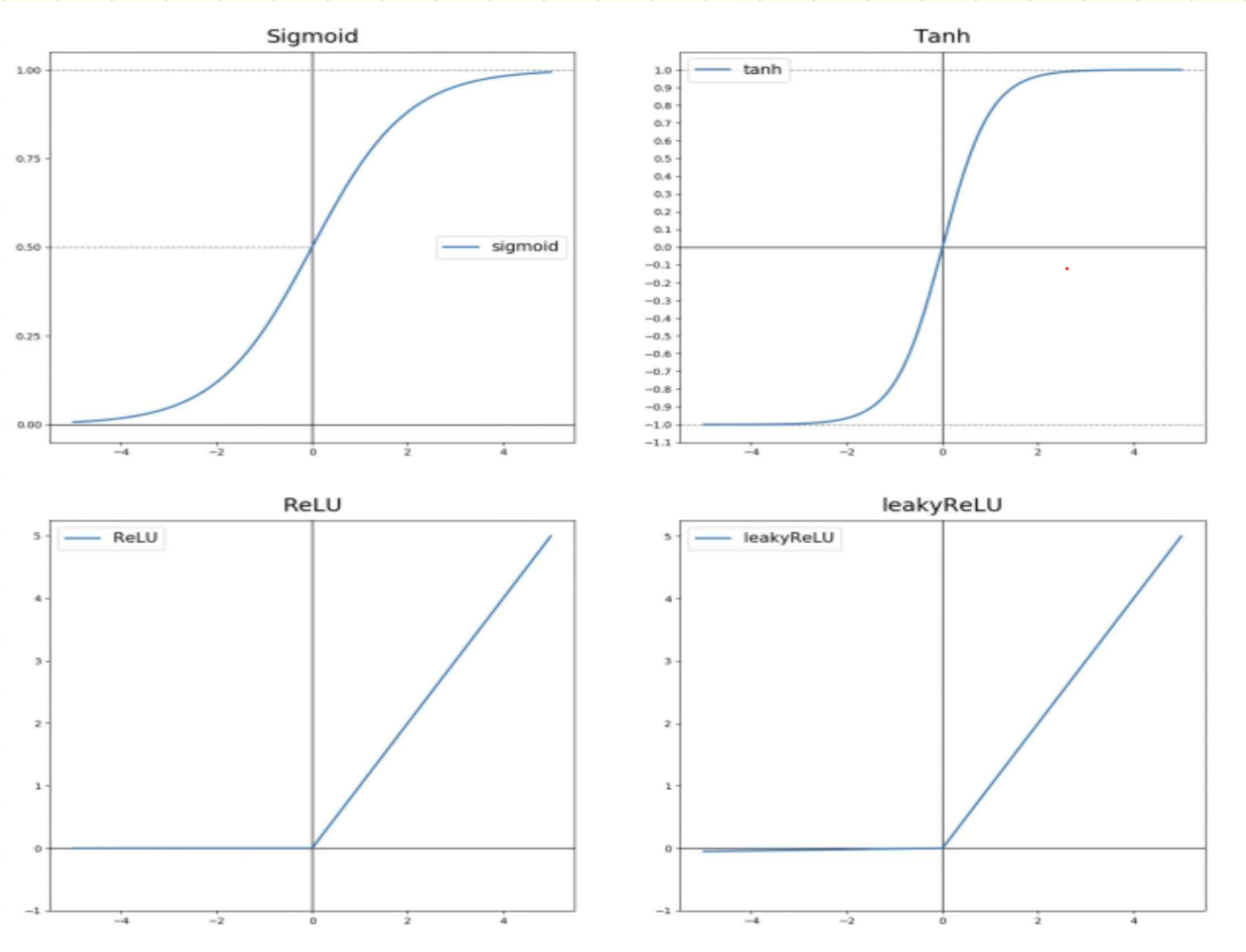
$$w^{[1]} = \begin{bmatrix} \sim\sim \\ \sim\sim \\ \sim\sim \\ \sim\sim \end{bmatrix} \quad w^{[1]}x^{(1)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \quad w^{[1]}x^{(2)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \quad w^{[1]}x^{3} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$w^{[1]}\begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \end{bmatrix} = \begin{bmatrix} | & | & | \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \\ | & | & | \end{bmatrix} = Z^{[1]}$$

$$\therefore \underset{\sim\sim\sim\sim\sim\sim\sim\sim\sim\sim}{w^{[1]}x^{(i)} = z^{[1](i)}}$$

Made with Goodnotes

<Activation Functions>



- sigmoid : $a = \frac{1}{1+e^{-z}}$
- ☆☆ tanh : $a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$
- ☆☆☆ ReLU : $a = \max(0, x)$
- leaky ReLU : $a = \max(0.01x, x)$

tanh의 장점

· 값이 [-1, 1] 사이에 있고 평균이 0 이므로, 데이터를 이동하는 효과가 있으며, 평균이 0.5인 sigmoid 보다 더 효율적이다.

ReLU의 장점

: 0보다 큰 활성함수의 미분값이 다른 활성 함수의 값보다 크기 때문에 더 효율적이다.

Made with Goodnotes

<Why do you need Non-Linear Activation Function?>

$z^{[1]} = w^{[1]}x + b^{[1]}$

$a^{[1]} = \cancel{g^{[1]}(z^{[1]})} \rightarrow z^{[1]}$ 선형 활성화 함수 : $g(z)=z$를 사용한다면,

$z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$

$a^{[2]} = \cancel{g^{[2]}(z^{[2]})} \rightarrow z^{[2]}$

$$a^{[1]} = z^{[1]} = w^{[1]}x + b^{[1]}$$

$$\begin{aligned} a^{[2]} = z^{[2]} &= w^{[2]}a^{[1]} + b^{[2]} \\ &= w^{[2]}\{w^{[1]}x + b^{[1]}\} + b^{[2]} \\ &= \underbrace{(w^{[2]}w^{[1]})}_{w'}x + \underbrace{(w^{[2]}b^{[1]} + b^{[2]})}_{b'} \\ &= w'x + b' \end{aligned}$$

∴ 선형 활성화 함수를 사용한다면, $g(g(g(z))) = z$ 로 세개의 은닉층을 쌓아도 효과를 얻지 못한다.

<Derivatives of Activation Functions>

* 시그모이드

- $g(z) = \dfrac{1}{1+e^{-z}}$
- $g'(z) = \boxed{\dfrac{d}{dz}g(z)} = \dfrac{1}{1+e^{-z}}\left(1 - \dfrac{1}{1+e^{-z}}\right) = g(z)(1-g(z))$

slope of $g(z)$ at z

* Tanh

- $g(z) = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$
- $g'(z) = 1 - (g(z))^2$

$\rightarrow$ $z=10$, $\tanh(z) \approx 1 \rightarrow g'(z) \approx 0$

$z=-10$, $\tanh(z) \approx -1 \rightarrow g'(z) \approx 0$

$z=0$, $\tanh(z) = 0 \rightarrow g'(z) = 1$

* ReLU

- $g(z) = \max(0, z)$
- $g'(z) = \begin{cases} 0 & (\text{if } z<0) \\ 1 & (\text{if } z \geq 0) \end{cases}$

실제로는 정의하기 어려움

* Leaky ReLU

- $g(z) = \max(0.01z, z)$
- $g'(z) = \begin{cases} 0.01 & (\text{if } z<0) \\ 1 & (\text{if } z \geq 0) \end{cases}$

Made with Goodnotes

< Gradient Descent for Neural Networks >

- Parameters : $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$
  $(n^{[1]}, n^{[0]})$ $(n^{[1]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[2]}, 1)$ $n_x = n^{[0]}$, $n^{[2]} = 1$
- Cost Function : $J(w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} L(\underbrace{\hat{y}}_{=a^{[2]}}, y)$
- Gradient Descent :

(반복 계산) $dw^{[1]} = \frac{\partial J}{\partial w^{[1]}}$, $db^{[1]} = \frac{\partial J}{\partial b^{[1]}}$, ...

$$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$$
$$b^{[1]} = b^{[1]} - \alpha db^{[1]}$$

< Forward propagation >

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]}) = \sigma(Z^{[2]})$$

< Back Propagation >

$dZ^{[2]} = A^{[2]} - Y$, $Y = [y^{(1)} \; y^{(2)} \; \cdots \; y^{(m)}]$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$db^{[2]} = \frac{1}{m}$ np.sum($dZ^{[2]}$, axis=1, keepdims=True)

: $(n^{[1]},)$로 계산 X
$(n^{[1]}, 1)$로 계산 O

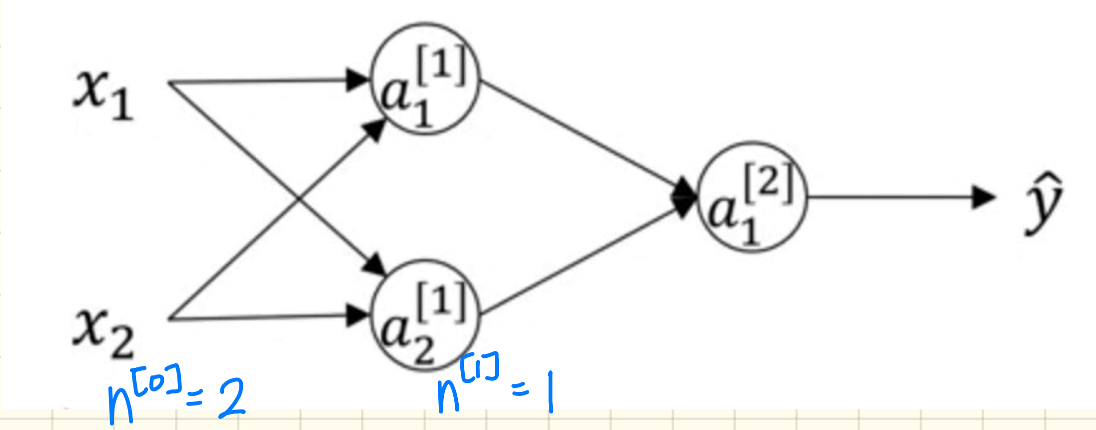
$$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{[1]}, m)} * \underbrace{g^{[1]'}(Z^{[1]})}_{(n^{[1]}, m)}$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$db^{[1]} = \frac{1}{m}$ np.sum($dZ^{[1]}$, axis=1, keepdims=True)

Made with Goodnotes

<Random Initialization>



$n^{[0]} = 2$ $n^{[1]} = 1$

If) w의 초기값을 '0'으로 초기화 한다면?

$w^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ → $a_1^{[1]} = a_2^{[1]}$
$dz_1^{[1]} = dz_2^{[1]}$

⇒ dw를 계산했을 때, 모든 층이 같은 값을 갖게된다.

∴ 0이 아니라 np.random.rand()를 이용하여 랜덤값을 부여해줘야 함.

⇒ $w^{[1]}$ = np.random.randn((2,2)) $\underbrace{*0.01}$ 매우 작은 값으로 만들어주기 위함.

$b^{[1]}$ = np.zero((2,1))

$w^{[2]}$ = np.random.randn((1,2)) * 0.01

$b^{[2]} = 0$

Made with Goodnotes