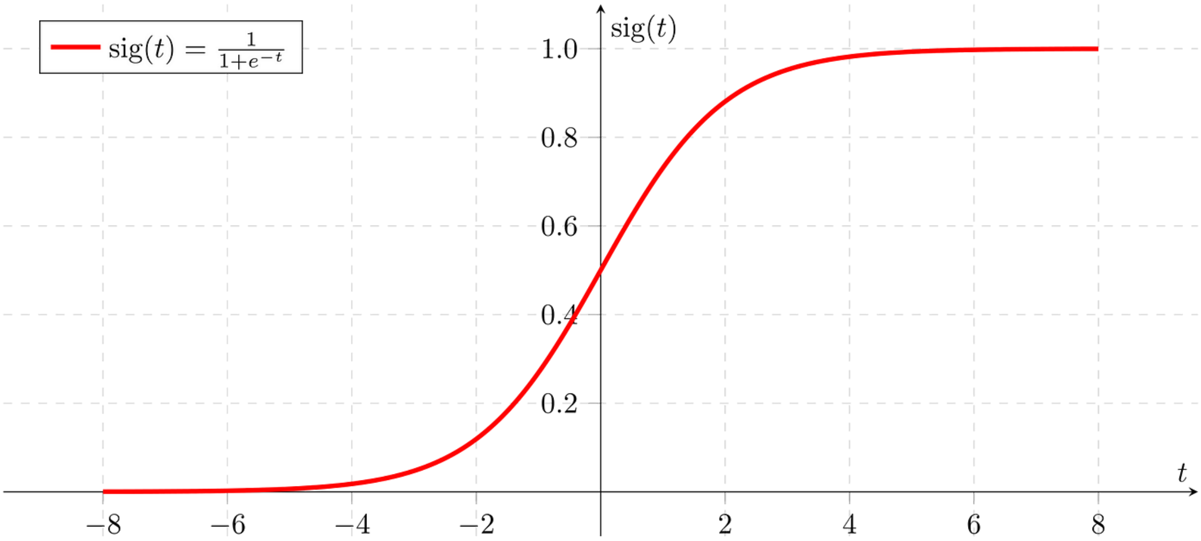
[Week4]_문가을

4. 얕은 신경망 네트워크

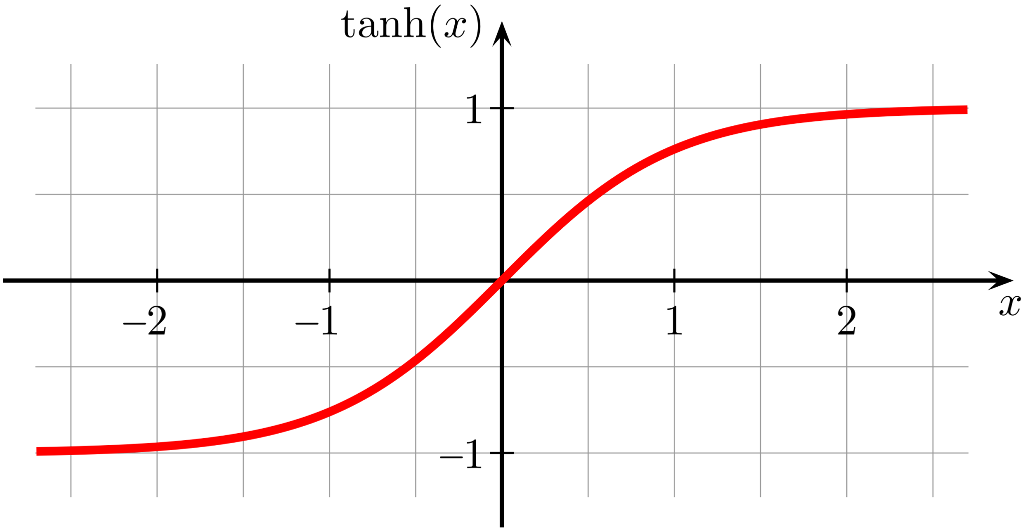
활성화 함수

- sigmoid function



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

- tanh function



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

은닉 유닛에서 시그모이드 함수보다 더 좋은 성능을 냄.

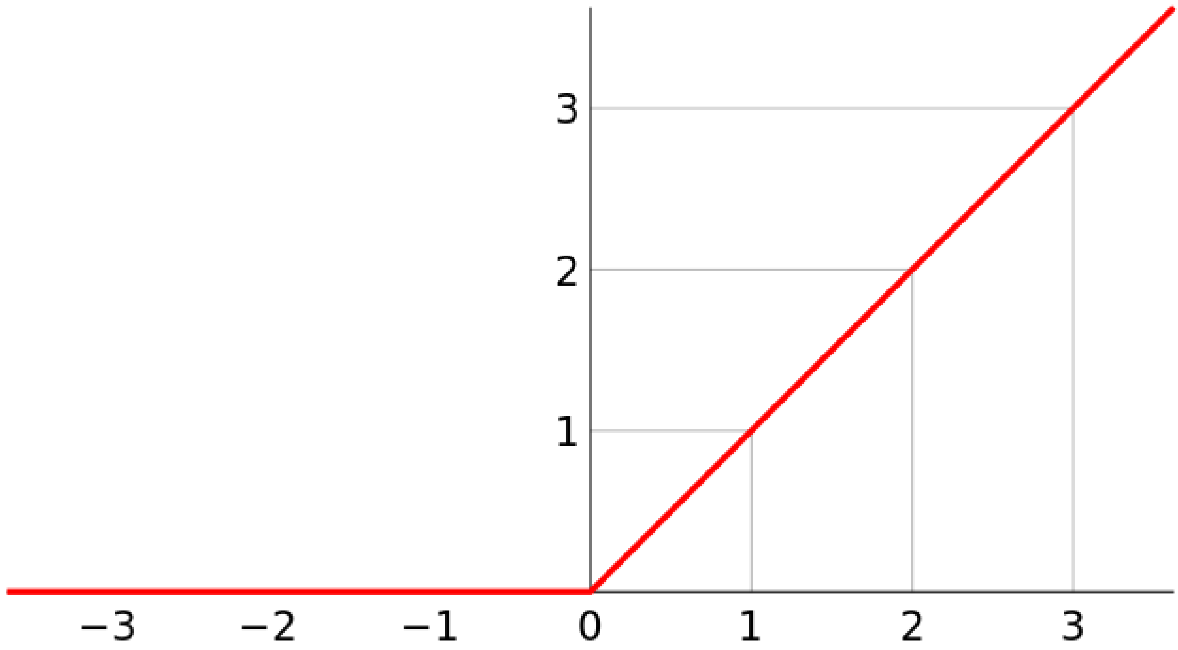
→ 값이 +1 과 -1 사이기 때문에 평균값이 0에 더 가까움. 데이터의 중심을 0.5 대신 0으로 만드는 효과가 있음.

출력 유닛에서는 시그모이드 함수가 더 좋을 수 있음.

→ y가 0이나 1이라면 y^은 -1과 1 사이 대신 0과 1 사이로 출력하는 게 더 좋기 때문임. (이진 분류)

시그모이드 함수와 tanh 함수 모드 z가 크거나 작으면 함수의 기울기가 0에 가까워져 경사하강법이 느려질 수 있다는 단점이 있음.

- ReLU function



$$ReLU(z) = max(0, z)$$

z 가 양수일 때는 도함수가 1이고, z가 음수이면 도함수가 0이됨.

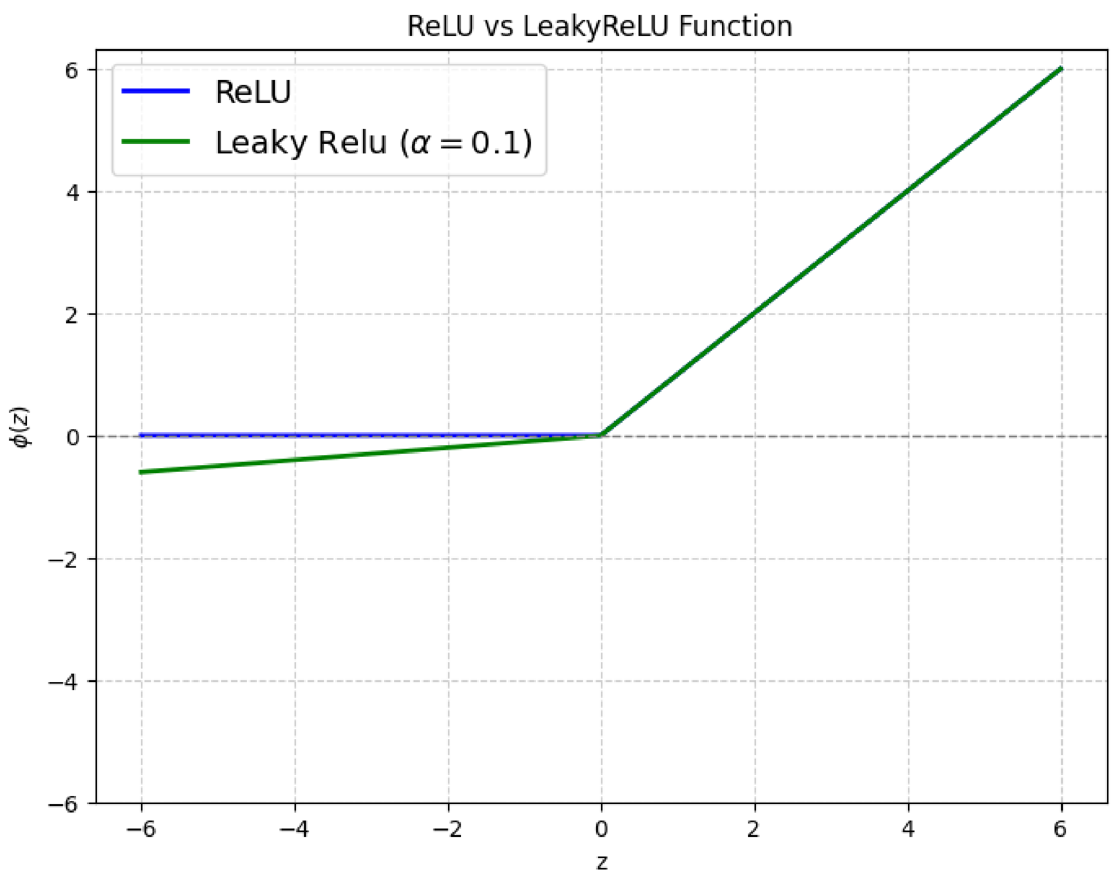
z가 0일 때 미분값은 정의 되지 않지만 z가 0이 될 확률이 매우 낮기 때문에 미분값을 0이나 1로 가정해도 잘 작동함.

장점 : 학습을 느리게 하는 원인인 함수의 기울기가 0에 가까워지는 걸 막기 때문에 훨씬 빠르게 학습 가능함.

단점 : z 값이 음수일 때 도함수 값이 0임.

→ 이를 보완한 leaky ReLU도 있음. z값이 음수일 때 기울기를 약간 줌.

- leaky ReLU



$$max(0.01z, z)$$

<**출력층**에서 활성화 함수 선택>

1) 이진 분류 → sigmoid function

2) 그 외라면 → ReLU function

<**은닉층**에서 활성화 함수 선택>

1) 대부분 ReLU function

2) 가끔 tanh function

왜 비선형 함수를 써야할까요?

선형 함수, 항등 활성화 함수를 사용할 경우 (a = g(z) = z)

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

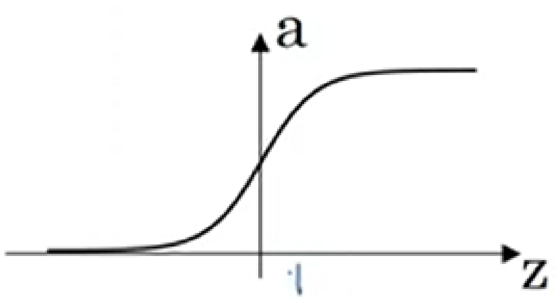
→ 여러 선형 함수의 조합은 하나의 선형 함수가 되기 때문에

→ 층이 많아도 은닉층이 없는 것과 같음.

y 가 실수값이라면 y^이 -inf 부터 inf 까지 실수값이 되도록 선형 함수를 사용해도 좋음.

활성화 함수의 미분

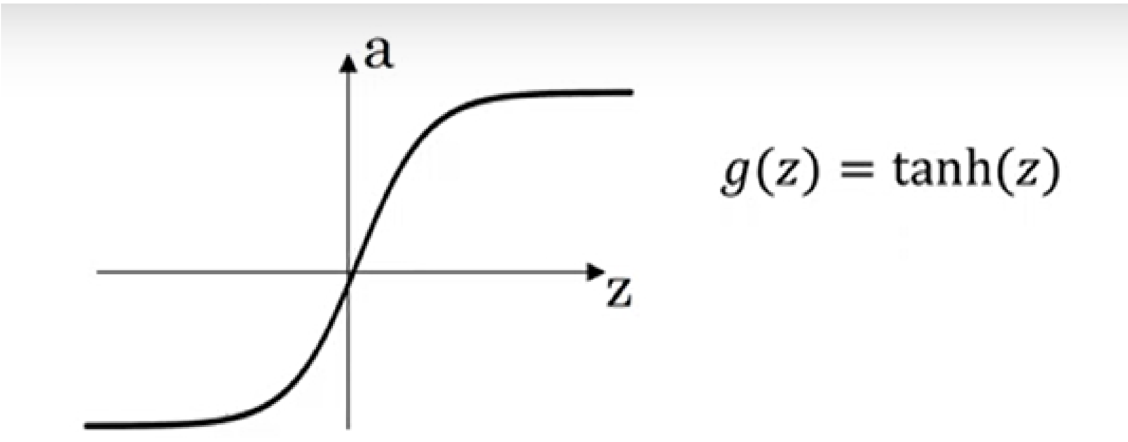
- sigmoid activation function



$$g(z) = \frac{1}{1 + e^{-z}}$$

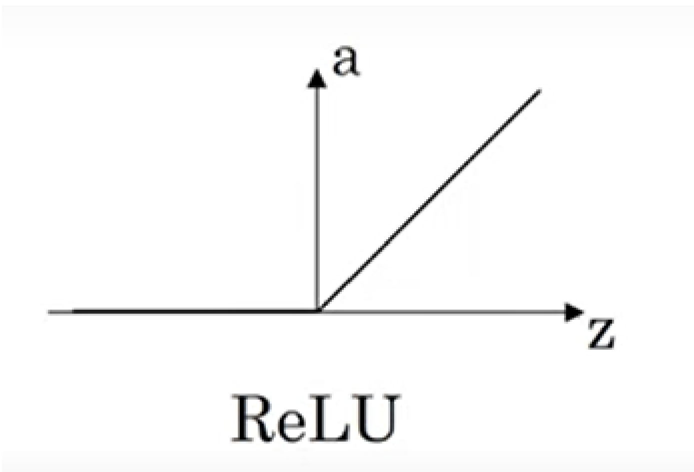
$$\begin{aligned}\frac{d}{dz}g(z) &= \frac{1}{1+e^{-z}}(1-\frac{1}{1+e^{-z}}) \\ &= g(z)(1-g(z)) \\ &= a(1-a)\end{aligned}$$

- tanh activation funtion



$$\frac{d}{dz}g(z) = 1-(tanh(z))^2$$

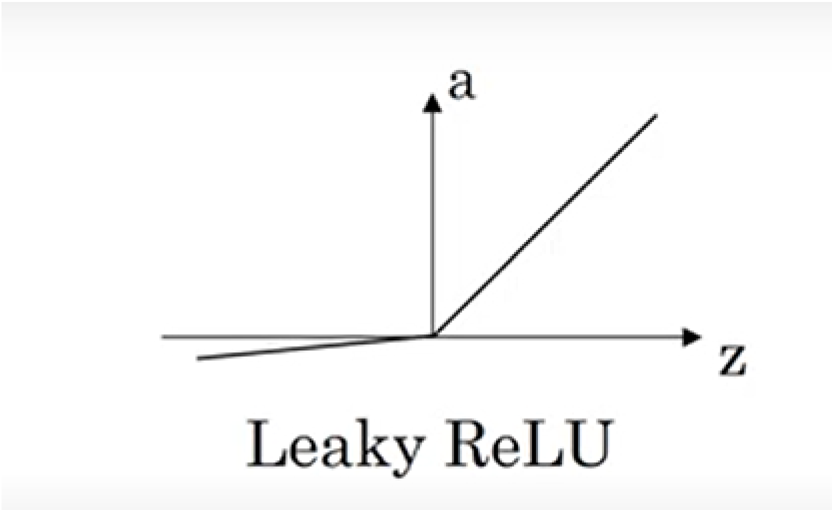
- ReLU



$$g'(x) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z >= 0 \end{cases}$$

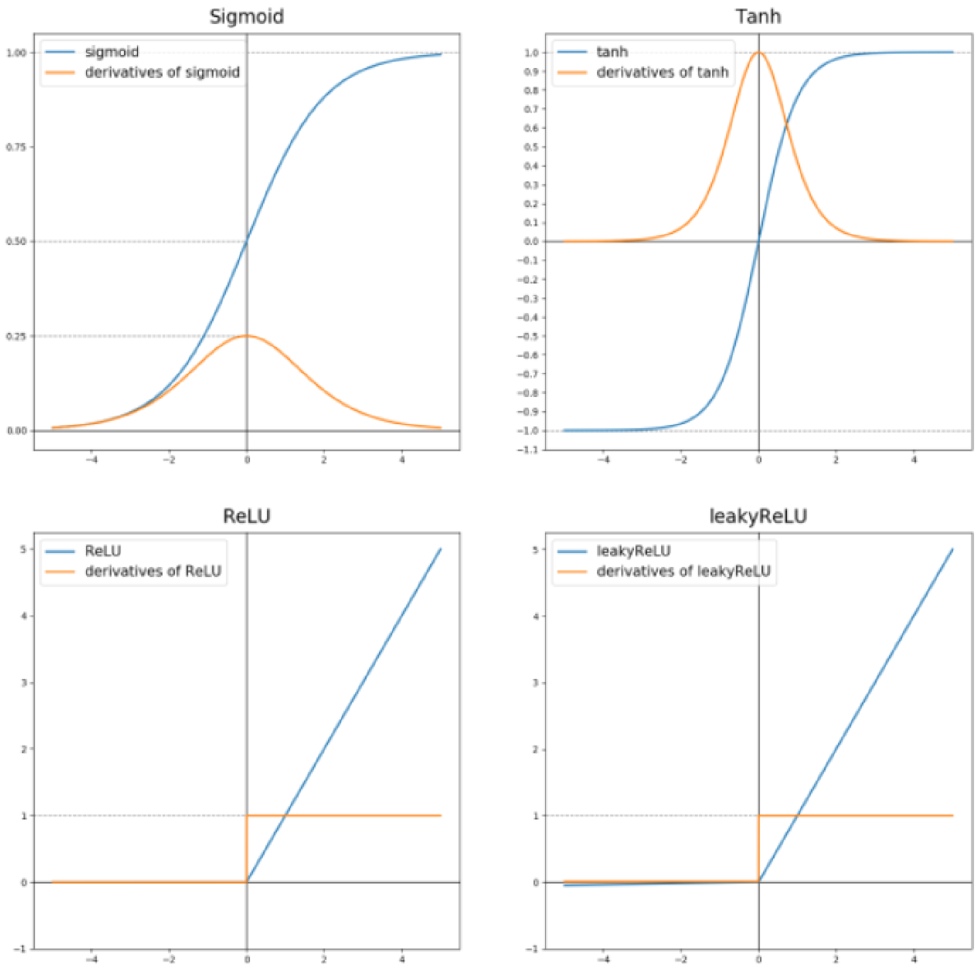
z가 0일 때 미분 불가능하지만 1 또는 0이라고 해도 문제 없음

- leaky ReLU



$$g(z) = max(0.01z, z)$$

$$g'(x) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z >= 0 \end{cases}$$



신경망 네트워크와 경사하강법

- 단일층 신경망

Forward propagation

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

Back propagation

$$
\begin{aligned}
dZ^{[2]} &= A^{[2]} - Y \\
dW^{[2]} &= \frac{1}{m} dZ^{[2]} A^{[1]T} \\
db^{[2]} &= \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True) \\
dZ^{[1]} &= W^{[2]T} dZ^{[2]} * g^{[1]}\prime(Z^{[1]}) \\
dW^{[1]} &= \frac{1}{m} dZ^{[1]} X^T \\
db^{[1]} &= \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)
\end{aligned}
$$

*np.sum

→ axis=1 : 가로로 더하기

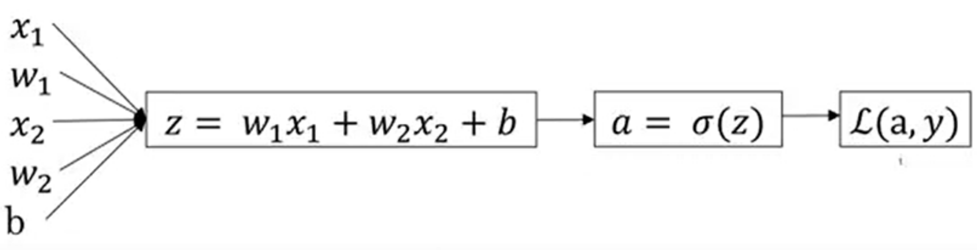
→ keepdims : 잘못된 1차원 배열을 출력하지 않게 함.

db^[2]에 대한 파이썬 출력값은 (n^[2],1) 벡터.

db^[1]에 대한 파이썬 출력값은 (n^[1],1) 벡터임.

역전파에 대한 이해

logistic regression - back propagation

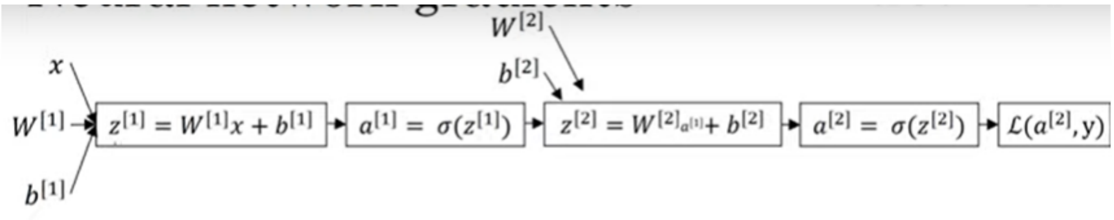


$$
\begin{aligned}
dz &= \frac{dL}{da}\frac{da}{dz} \\
&= a - y \\
&= da * g'(z)
\end{aligned}
$$

$$
\begin{aligned}
dw_1 &= \frac{dL}{dz}\frac{dz}{dw} \\
&= dzx_1
\end{aligned}
$$

$$
\begin{aligned}
db &= \frac{dL}{dz}\frac{dz}{db} \\
&= dz * 1 \\
&= dz
\end{aligned}
$$

- 2개의 층을 가진 신경망 (단일 훈련 샘플)



dx는 구하지 않아도 됨. → 지도학습에서 고정된 값이기 때문임.

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]}a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

$$dW^{[1]} = dz^{[1]}x^T$$

$$db^{[1]} = dz^{[1]}$$

a[1]을 transpose 하는 이유 (잘 이해가 가지 않음..)

→ 로지스틱 회귀는 하나의 출력을 갖는 경우 w는 행 벡터가 됨. 그렇지만 여기의 w는 열 벡터기 때문에 a를 전치해야 함.

$$
\begin{aligned}
dz^{[1]} &= \frac{dL}{da^{[1]}}\frac{da^{[1]}}{dz^{[1]}} \\
&= \frac{dL}{dz^{[1]}}\frac{dz^{[2]}}{da^{[1]}}\ \frac{da^{[1]}}{dz^{[1]}} \\
&= dz^{[2]}W^{[2]T}\ g^{[1]}\prime(z^{[1]})
\end{aligned}
$$

- 단일 훈련 샘플 / 여러 훈련 샘플

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]}a^{[1]^T}$	$dW^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$
$dW^{[1]} = dz^{[1]}x^T$	$dW^{[1]} = \frac{1}{m}dZ^{[1]}X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

랜덤 초기화

신경망을 훈련시킬 때 변수를 임의값으로 초기화시켜야 함.

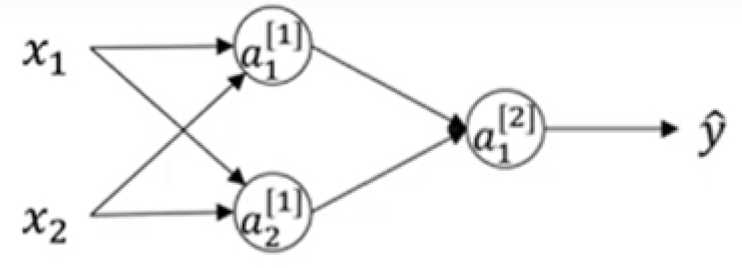
w의 값을 모두 0으로 초기화할 경우 → 대칭의 문제 발생

→ 은닉 유닛이 같은 함수를 계산

→ 은닉 유닛이 출력 유닛에 항상 같은 영향을 주게됨.

→ 첫 번째 반복 이후에 같은 상태가 계속해서 반복됨.

→ 은닉 유닛이 몇 개가 있든 한 개와 같은 상태임.



$$
\begin{aligned}
W^{[1]} &= np.random.randn((2,2)) * 0.01 \\
b^{[1]} &= np.zero((2,1)) \\
W^{[2]} &= np.random.randn((1,2)) * 0.01 \\
b^{[2]} &= 0
\end{aligned}
$$

가중치의 초기값을 매우 작은 값으로 정하는 것이 좋음.

→ 가중치가 클 경우, z가 크거나 작은 값이 나올 수 있음.

→ sigmoid 함수나 tanh 함수에서 z가 크거나 작은 부분에서 경사의 기울기가 매우 낮기 때문에, 경사 하강법이 매우 느리게 진행됨.

→ 0.01 혹은 작은 값을 곱함.

b^[1]은 대칭의 문제를 가지지 않음.

→ w를 임의의 값으로 초기화하여 다른 은닉 유닛에서 다른 결과를 만들어 낼 것이기 때문에 대칭 문제가 없기 때문임.

→ 0으로 초기화해도 괜찮음.