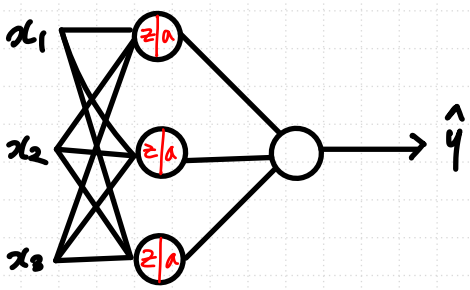
활성화 함수



Given x) 해당 layer neuron개수 $\begin{bmatrix} w_{01} w_{11} w_{21} \\ \vdots \end{bmatrix} = W^{[1]}$

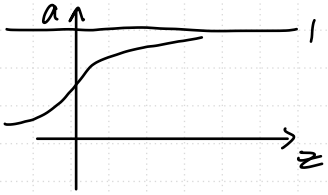
$z^{[1]} = W^{[1]} x + b^{[1]}$

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

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

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

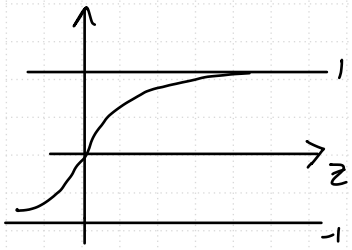
Sigmoid function



$a = \frac{1}{1+e^{-z}}$

원점을 지나고 바울이 달라짐

tanh function



$a = \tanh(z)$

$= \frac{e^z - e^{-z}}{e^z + e^{-z}}$

∇ → tanh in hidden layer?

→ 평균이에 더 가깝기 때문에 더 좋다

→ 데이터 중심 이동 to 0 : 다음 층의 학습이 더 쉬워진다

but 최적함수는 아님

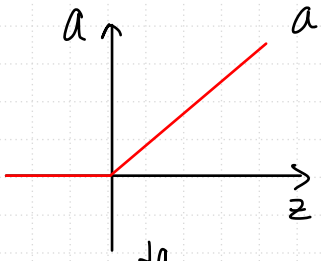
→ 이진 분류에서는 0과 1 사이의 값이 더 좋기 때문에 시그모이드 함수가 더 좋다.

Sigmoid와 tanh의 단점

→ z가 굉장히 크거나 작아질 도함수의 값이 거의 0임

→ 경사하강법이 느려짐

ReLu ? (rectified linear unit)



$a = \max(0, z)$

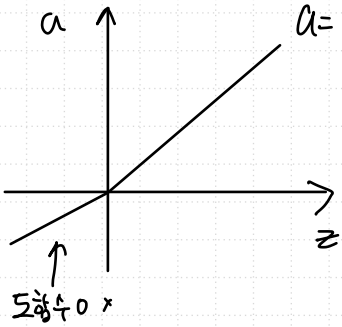
$z > 0$ $\frac{da}{dz} = 1$

$z < 0$ $\frac{da}{dz} = 0$

$z = 0$? 엄밀하게 정의 X but 컴퓨터에서 0.000…0이 나온다는 거의 희박함

∴ 미분불가능해도 ㄱㅊ

Leaky ReLu?



$a = \max(0.01z, z)$

ReLU와 Leaky ReLu의 장점

→ 대부분의 z에 대해 기울기가 0과 매우 다르다

→ 신경망이 훨씬 빠르게 학습함

※ 각 문제에 따라 어떤 함수가 좋을지는 직접 다 해봐야 안다.

No. Date. Content.

왜 비선형 함수를 써야하는가?

if) 활성화 함수 없다면

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

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

$= W^{[2]}(W^{[1]}x + b^{[1]}) + b^{[2]}$

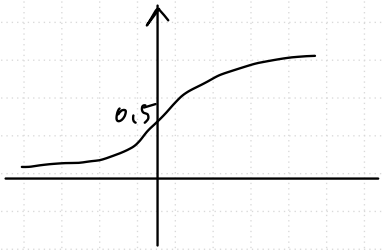
$= \underbrace{W^{[2]}W^{[1]}}_{W'}x + \underbrace{W^{[2]}b^{[1]} + b^{[2]}}_{b'}$

$= W'x + b'$

층이 얼마나 많든간에, 은닉층이 없는 것과 다름없는 결과가 나온다.

활성화 함수의 미분

1) Sigmoid



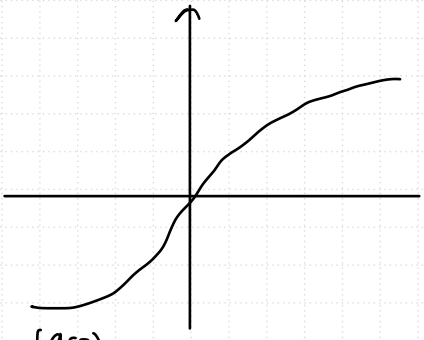
$$\frac{dg(z)}{dz} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z)(1 - g(z)) = a(1-a)$$

⊕ $z = 0$) $g'(z) = \frac{1}{4}$

∴ 이때 $g(z)$의 값을 구해놨다면 $g'(z)$는 빠르게 구할 수 있다.

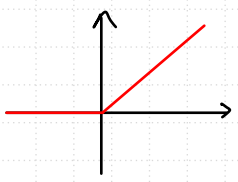
2) tanh



$$\frac{dg(z)}{dz} = 1 - (\tanh(z))^2 = 1 - a^2$$

∴ tanh 함수도 미리 구한 $\tanh(z)$가 있다면 빠르게 구할 수 있다.

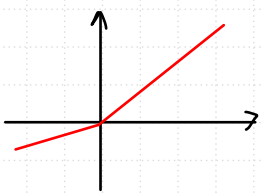
3) ReLU



$$g(z) = \max(0, z)$$

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

4) Leaky-ReLU



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

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

신경망 네트워크와 경사하강법

Params: $W^{[1]}$ $b^{[1]}$ $W^{[2]}$ $b^{[2]}$
$(n^{[1]}, n^{[0]})$ $(n^{[1]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[2]}, 1)$

$n_x = n^{[0]}$, $n^{[1]}$ 은닉층, $n^{[2]} = 1$ 출력유닛

Cost func (이진 분류): $J(\text{params}) = \frac{1}{m}\sum_{i=1}^{n} \mathcal{L}(\hat{y}, y)$, $\hat{y} = a^{[2]}$

Gradient descent: 0 이 아닌 수를 초기화하는게 중요

Repeat { Until convergence
Compute predicts ($\hat{y}^{(i)}$, $i = 1 \cdots m$)
1 iter {
$dw^{[1]} = \frac{dJ}{dw^{[1]}}$, $db^{[1]} = \frac{dJ}{db^{[1]}}$, ..
$w^{[1]} := w^{[1]} - \alpha\, dw^{[1]}$
$b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$
$w^{[2]} := w^{[2]} - \alpha\, dw^{[2]}$
$b^{[2]} := b^{[2]} - \alpha\, db^{[2]}$

Forward propagation

$z^{[1]} = w^{[1]}x + b^{[1]}$
$A^{[1]} = g^{[1]}(z^{[1]})$
$z^{[2]} = w^{[2]}A^{[1]} + b^{[2]}$
$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$
이진분류에서 마지막 활성함수는 sigmoid

Back propagation

$dz^{[2]} = A^{[2]} - Y$ $\quad (Y = [y^{(1)} \cdots y^{(m)}])$

$dw^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$ $\quad (n^{[2]}, n^{[1]})$ 유지

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

$dz^{[1]} = W^{[2]T} dz^{[2]} \times g^{[1]\prime}(z^{[1]})$
$(n^{[1]}, m)$ element wise product $(n^{[1]}, m)$

$dw^{[1]} = \frac{1}{m} dz^{[1]} x^{T}$ $\quad (n^{[1]}, n^{[0]})$ 유지

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

$$\frac{d\mathcal{L}}{da} = \frac{1}{da}\left(-y \log a - (1-y)\log(1-a)\right)$$
$$= -\frac{y}{a} + \frac{1-y}{1-a}$$
$$\frac{d\mathcal{L}}{dz} = a - y$$
$$\frac{d\mathcal{L}}{dw} = dz \cdot x$$
$$\frac{d\mathcal{L}}{db} = dz$$

랜덤초기화

$$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

⇒ $a_1^{[1]} = a_2^{[1]}$

⇒ $dz_1^{[1]} = dz_2^{[1]}$

) 가중치의 영향을 받지 않아 서로 다른 유닛이 같이 움직인다.

: Symmetric

⇒ 같은 값을 계산하기 때문에 은닉층에 unit이 많이 있을 필요가 없다

$$W^{[2]} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

해결 책

W을 랜덤초기화하는 법

↳ 은닉층의 갯수에 따라 이 숫자가 달라진다

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

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

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

$b^{[2]}$ = 0

가중치가 작을수록 좋다