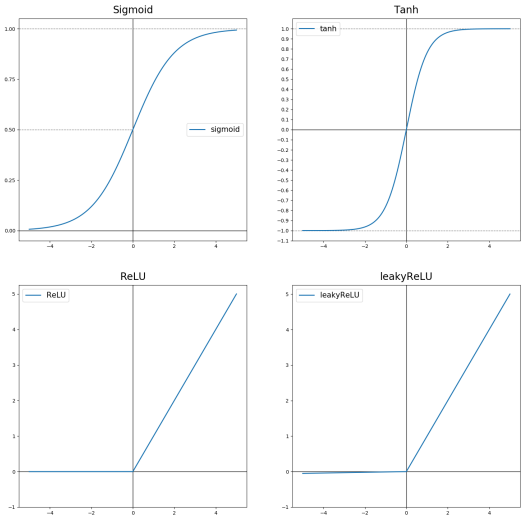
4주차

활성화 함수

- 활성화 함수로 σ보다 사용하기에 적합한 비선형 함수가 있다.
- $tanh$의 장점: $a = tanh(z)$이 $a = \sigma(z)$보다 좋다. $tanh$는 값이 -1부터 1 사이이기 때문에 평균값이 0에 더 가까워 **데이터의 중심을 원점으로 이동하게 하는 효과**가 있다. 이는 다음 층에서의 학습을 더 용이하게 한다. 따라서 은닉층에서는 σ보다 $tanh$를 사용한다. 다만 $y \in 0, 1$ 이진 분류의 출력층에서는 σ를 사용한다.
- $tanh$와 σ의 단점: z가 굉장히 크거나 작으면 함수의 도함수가 굉장히 작아져 경사 하강법이 매우 느려진다.
- $ReLU$의 장점: $tanh$와 σ의 단점을 극복할 수 있다. 요즘에는 $ReLU$가 가장 많이 쓰인다. **신경망을 빠르게 학습**시킬 수 있다. 학습을 느리게 하는 원인인, 함수의 기울기가 0에 수렴하는 걸 막기 때문이다. z=0일 확률은 매우 작지만, z=0이라도 상관 없다.
- $ReLU$의 단점: z가 음수일 때 도함수가 0이다. 그러나 은닉 노드의 z는 0보다 크기 때문에 실제로는 잘 동작한다. 예를 들어 집값 예측에서, $\hat{y}$는 가격으로 항상 양수이기 때문에 $ReLU$가 잘 동작한다.
- $leakyReLU$: z가 음수할 때 도함수가 0이 아니게 되도록 해준다. 밑의 그래프를 자세히 보면 z<0일 때 계수가 0.01이기 때문에 그래프가 살짝 꺾이는 것을 볼 수 있다. 실제로는 많이 쓰이지 않지만 쓰인다면 $ReLU$보다 좋은 결과를 보여준다.

- Sigmoid
  - $a = \frac{1}{1 + e^{-z}}$
- Tanh
  - $a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$
- ReLU
  - $a = max(0, z)$
- leaky ReLU
  - $a = max(0.01z, z)$



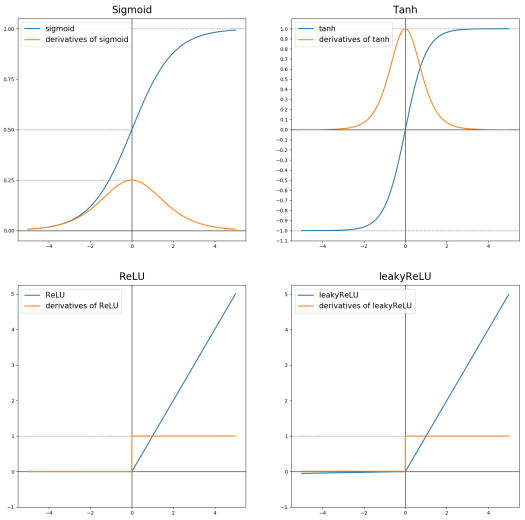
x축은 z, y축은 a=g(z), g는 활성화 함수

왜 비선형 활성화 함수를 써야 할까요?

- $g(z) = z$라는 선형 활성화 함수가 있다고 한다면 아무리 신경망을 거쳐도 $g(g(g(z))) = z$로, 은닉층이 없는 것과 다름 없다. 선형 은닉층은 쓸모 없다. **따라서 은닉층에서 선형식을 사용한다면(비선형식을 쓰지 않는다면) 신경망이 깊어져도 의미 있는 결과를 출력하기 어렵다.**
- 선형 활성화 함수를 쓰는 곳은 출력층이다. 은닉층에서는 사용하지 않는다.
- 비선형 활성화 함수에는 $\sigma, ReLU, tanh, leakyReLU$ 등이 있다.

활성화 함수의 미분

- 시그모이드
  - $g(z) = \dfrac{1}{1+e^{-z}}$
  - $g'(z) = \dfrac{d}{dz}g(z) = g(z)(1-g(z))$
- Tanh
  - $g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
  - $g'(z) = 1-(g(z))^2$
- ReLU
  - $g(z) = max(0, z)$
  - $g'(z) = 0$ (z < 0인 경우)
  - $g'(z) = 1$ (z >= 0인 경우)
- Leaky ReLU
  - $g(z) = max(0.01z, z)$
  - $g'(z) = 0.01$ (z < 0인 경우)
  - $g'(z) = 1$ (z >= 0인 경우)



- 시그모이드에서, $a = g(z)$이기 때문에 a의 값을 안다면 쉽게 $g'(z)$를 구할 수 있다는 이점이 있다. ($\because g'(z) = a(1-a)$)
- $tanh$에서, $a = g(z)$이기 때문에 a의 값을 안다면 쉽게 $g'(z)$를 구할 수 있다는 이점이 있다. ($\because g'(z) = 1-a^2$)

신경망 네트워크와 경사 하강법

* neural network for 1 hidden layer

nodes (n_x): $n^{[0]}$, $n^{[1]}$, $n^{[2]}$

· parameters: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$

$(n^{[1]}, n^{[0]})$ $(n^{[1]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[2]}, 1)$

· Cost function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}, y)$, $\hat{y} = a^{[2]}$

· Gradient descent

Repeat with $(x^{(i)}, y^{(i)})$ where $i = 1 \cdots k$

{ $dW^{[1]} = \frac{dJ}{dW^{[1]}}$, $db^{[1]} = \frac{dJ}{db^{[1]}}$

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

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

단일층이 아닐 경우에는 [1] 뿐 아니라

[2], [3] ··· [m]도 [1]과 똑같은 방법으로 업데이트 }

X ⋱

$W^{[1]}$ ⋯ $z^{[1]} = W^{[1]}X + b^{[1]}$ ⋯ $A^{[1]} = g^{[1]}(z^{[1]})$ ⋱

$b^{[1]}$ ⋮ $W^{[2]}$ ⋯ $z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$ ⋯ $A^{[2]} = g^{[2]}(z^{[2]})$ ⋯ $L(A^{[2]}, Y)$

$b^{[2]}$ ⋮

역전파에 대한 이해

- 로지스틱 회귀의 역전파를 구하면 다음과 같다.

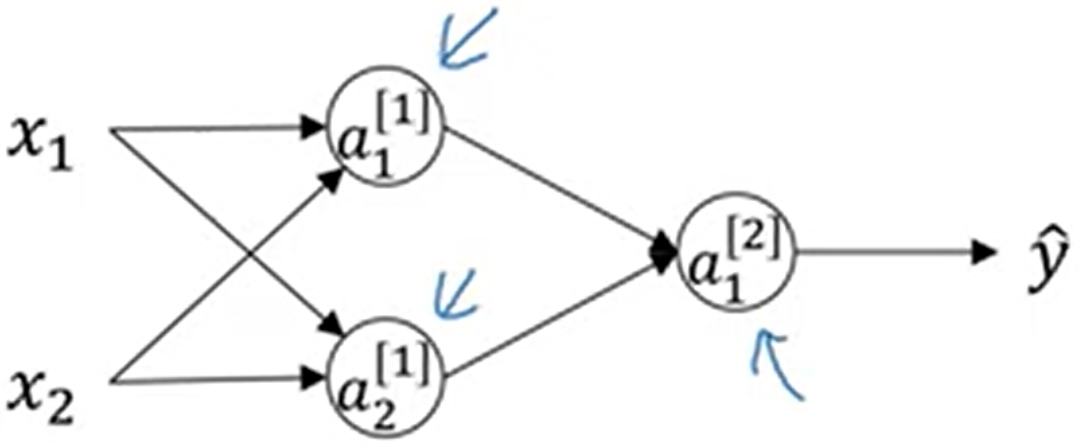
$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]}a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]}A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T}dZ^{[2]} *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

- $n^{[0]}, n^{[1]}, n^{[2]}$...는 각 층 별 노드의 개수, m은 $X = [x^{(1)}, x^{(2)} \dots x^{(m)}]$로, 훈련 데이터 x의 개수
- 벡터화를 하면 shape가 (#node, m)이다.

랜덤 초기화

- 신경망에서 w의 초기값을 0으로 설정하고 경사 하강법을 적용할 경우 올바르게 작동하지 않는다.
  - dw를 계산했을 때 모든 층이 대칭이 되어 같은 값을 가지게 되기 때문이다→쓸모없음
  - 그러나 b는 w와 같은 문제가 발생하지 않아 영행렬로 초기화해줘도 문제 없다.
- 따라서 np.random.rand()를 통해 0이 아닌 랜덤값을 부여해줘야 한다.

> 📌 **random 함수**
>
> np.random.randint(100)-0~99까지의 정수 하나를 랜덤으로 반환
>
> np.random.rand(100)-0~1사이의 수 100개를 랜덤으로 반환
>
> np.random.randn(100)-가우시안 정규분포를 따르는 수 100개를 랜덤으로 반환
>
> np.random.rand()-0~1사이의 수 하나를 랜덤으로 반환




```
w^{[1]}=np.random.randn((2,2))*0.01 #(2,2) 행렬
b_{[1]}=np.zeros((2,1))
w^{[2]}=np.random.randn((1,2))*0.01
b_{[2]}=np.zeros((1,1))
```

💡w에 0.01을 곱해 w를 매우 작게 만들어주는 이유는, 시그모이드/tanh를 이용했을 때 z가 너무 큰 값이 되면 기울기 이슈로 학습의 속도가 매우 느려진다는 것에 있다. w가 너무 크면 이에 따라 z도 커지고, 활성화 함수를 거친 a도 커지기 때문에 애초에 초기의 w를 작은 값으로 설정해야 할 필요가 있다. 0.01 외의 다른 작은 수를 곱해줘도 상관 없다!

QUIZ

✕ 2. 로지스틱 회귀의 가중치 w를 모두 0으로 설정하면 "break symmetry"할 수 없어 적절한 decision boundary를 학습할 수 없기 때문에 랜덤하게 초기화해야한다. *0/1

- (●) True ✕
- ( ) False

정답

- (●) False

✕ 4. 다음 중 옳은 번호를 **모두** 골라주세요. *

0/1

1. $a^{[2](6)}$는 6번째 훈련 샘플에 대한 두번째 레이어의 활성화 벡터(activation vector)를 뜻한다.
2. X는 각 컬럼이 하나의 훈련 샘플인 행렬이다.
3. $a^{[2](6)}$는 두번째 훈련 샘플에 대한 6번째 레이어의 활성화 벡터(activation vector)를 뜻한다.
4. $a^{[2]}$는 두번째 레이어의 활성화 벡터(activation vector)를 뜻한다.
5. X는 각 열이 하나의 훈련 샘플인 행렬이다.
6. $a_4^{[2]}$는 두번째 레이어의 네번째 뉴런의 활성화 함수의 출력(activation output)을 뜻한다.
7. $a_4^{[2]}$는 네번째 훈련 샘플에 대한 두번째 레이어의 활성화 벡터(activation vector)를 뜻한다.

- [x] 1 ✓
- [x] 2 ✓
- [ ] 3
- [x] 4 ✓
- [x] 5 ✕
- [x] 6 ✓
- [ ] 7