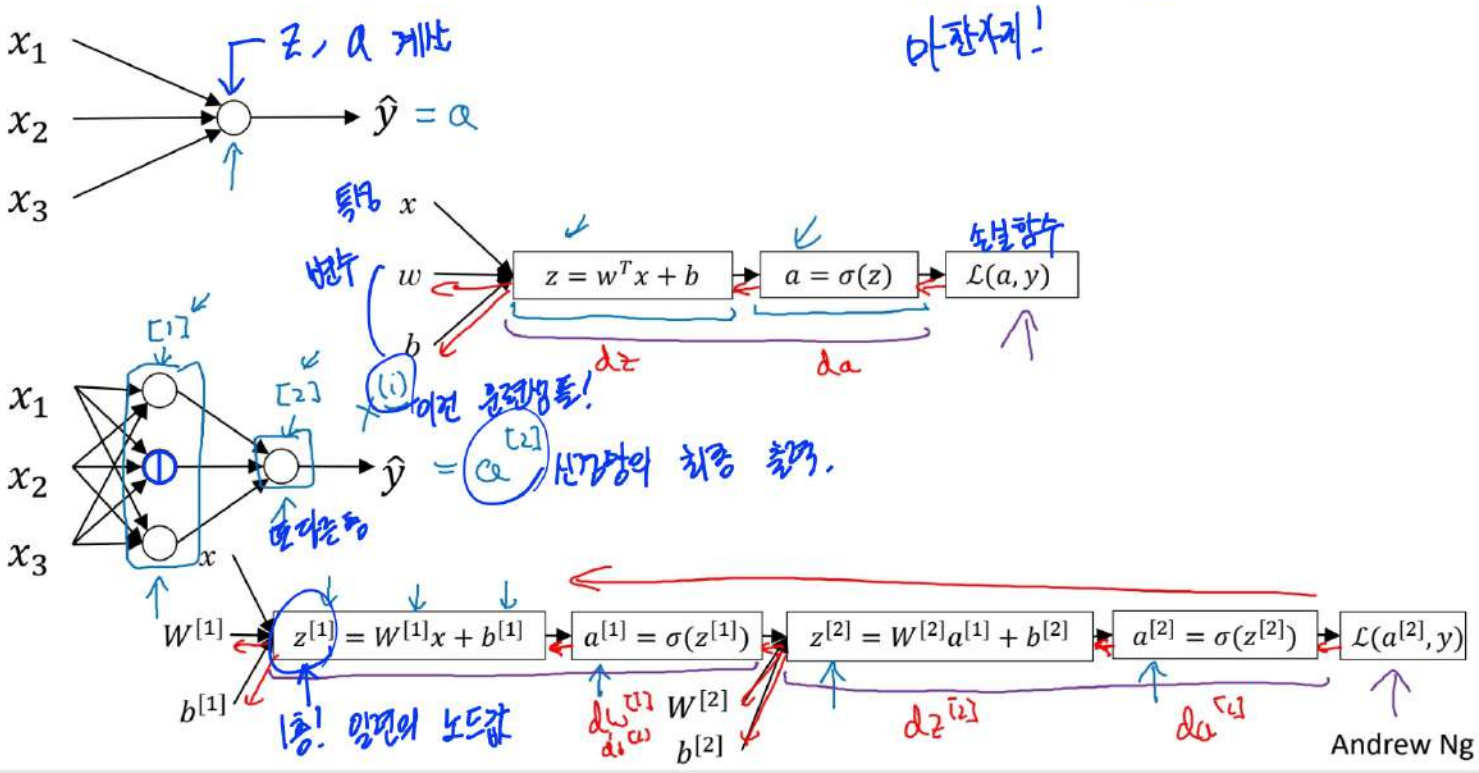
deeplearning.ai

One hidden layer Neural Network

Neural Networks Overview

로지스틱 회귀에서는 도함수 계산을 위해 역방향 계산을 했었다.
마찬가지!

What is a Neural Network?



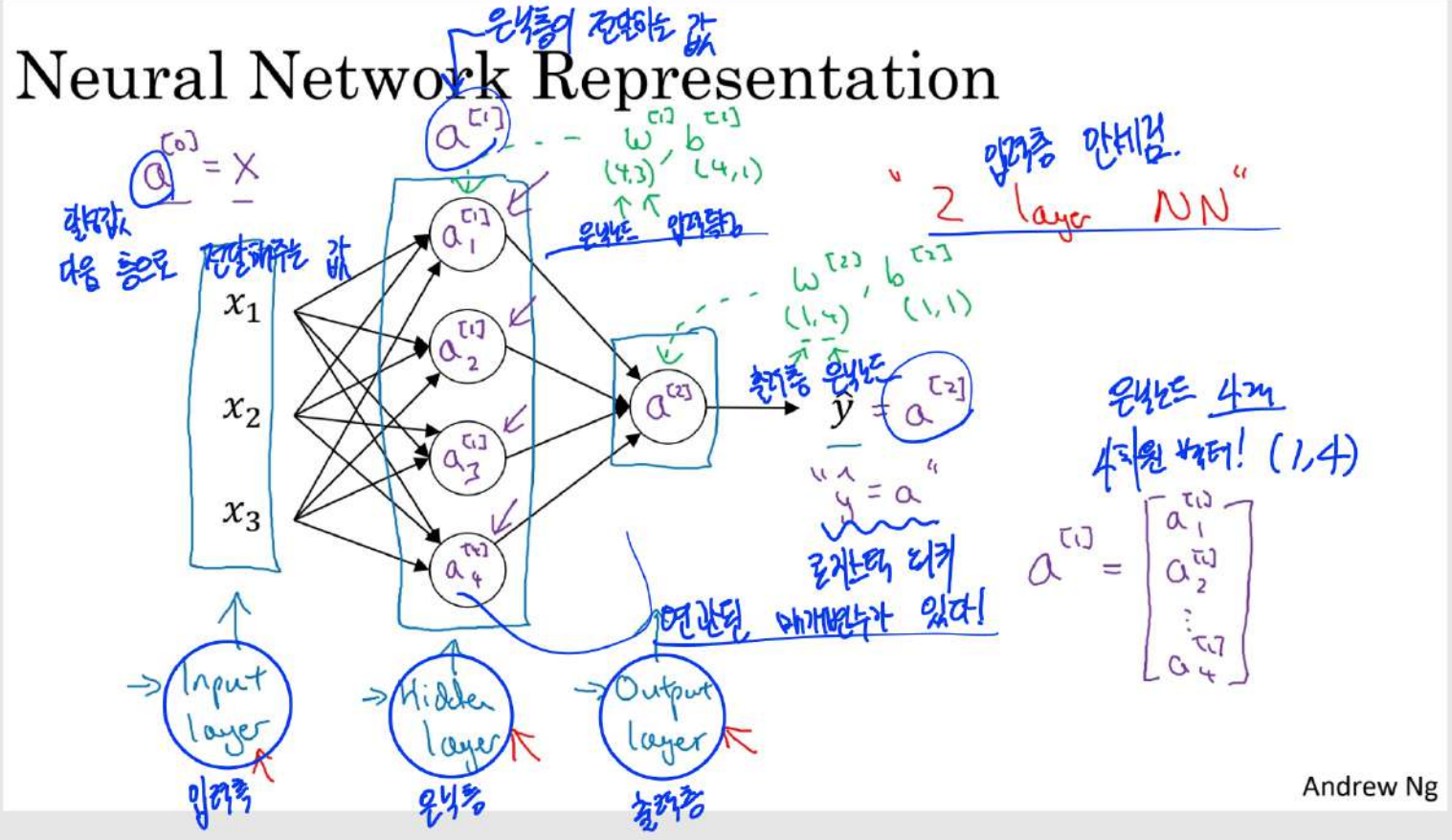
deeplearning.ai

One hidden layer Neural Network

Neural Network Representation

지도학습
: 훈련 세트가 입력값 X와 출력값 Y.
(은닉층의 실제 값은 훈련세트에 X)

은닉층이 하나인 신경망!

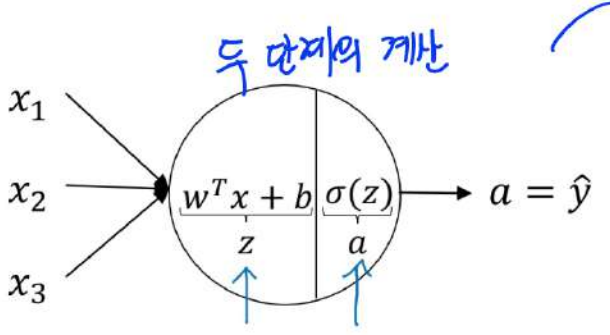


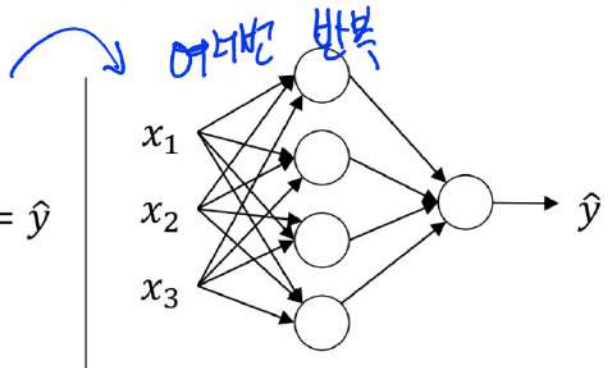
deeplearning.ai

One hidden layer Neural Network

Computing a Neural Network's Output

Neural Network Representation



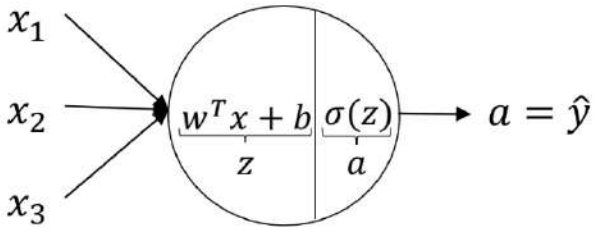


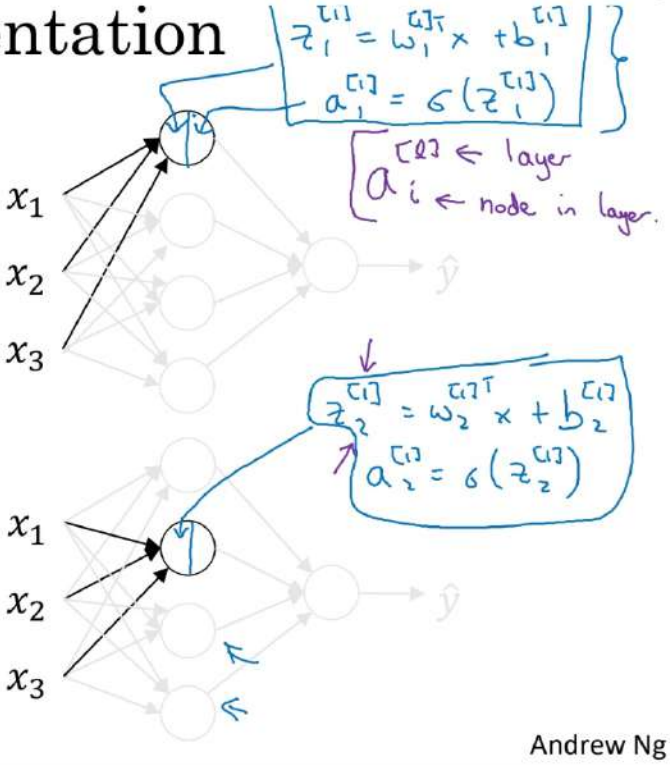
$$z = w^T x + b$$

$$a = \sigma(z)$$

Andrew Ng

Neural Network Representation



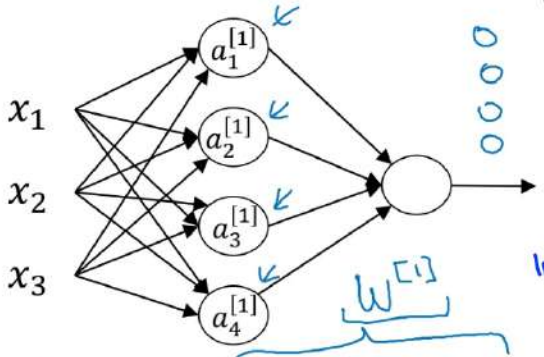


$$z = w^T x + b$$

$$a = \sigma(z)$$

Andrew Ng

Neural Network Representation



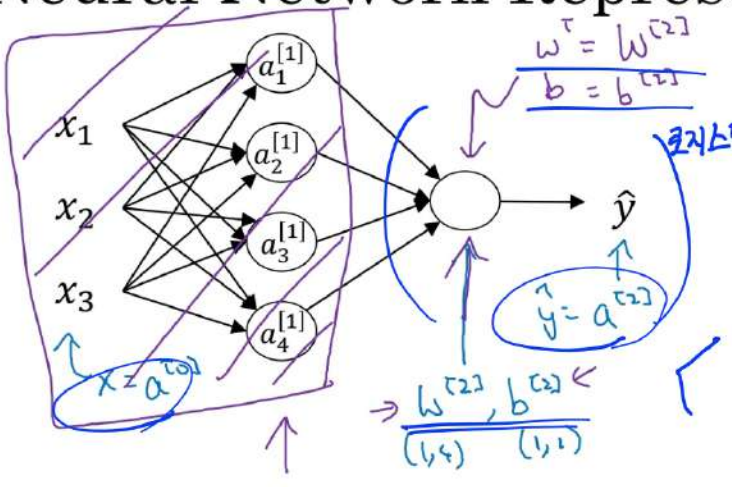
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$z^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

Andrew Ng

Neural Network Representation learning



Given input x:

$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]} \quad (4,1) \quad (4,3) \quad (3,1) \quad (4,1)$$
$$a^{[1]} = \sigma(z^{[1]}) \quad (4,1) \quad (4,1)$$
$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]} \quad (1,1) \quad (1,4) \quad (4,1) \quad (1,1)$$
$$a^{[2]} = \sigma(z^{[2]}) \quad (1,1) \quad (1,1)$$

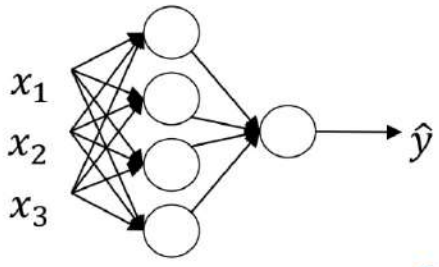
Andrew Ng

deeplearning.ai

One hidden layer Neural Network

Vectorizing across multiple examples

Vectorizing across multiple examples



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

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

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

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

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

$a^{[2](i)}$ ← example i 번째 출력값들

layer 2

for $i = 1$ to m:

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

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

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

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

$Z^{[1]} = W^{[1]}X + b^{[1]}$

$A^{[1]} = \sigma(Z^{[1]})$

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

$A^{[2]} = \sigma(Z^{[2]})$

$X = [x^{(1)}\ x^{(2)}\ \cdots\ x^{(m)}]$ (대문자 입력값들)

n_x, m (입력 개수)

$Z^{[1]} = [z^{[1](1)}\ z^{[1](2)}\ \cdots\ z^{[1](m)}]$

$A^{[1]} = [a^{[1](1)}\ a^{[1](2)}\ \cdots\ a^{[1](m)}]$

training examples

hidden units

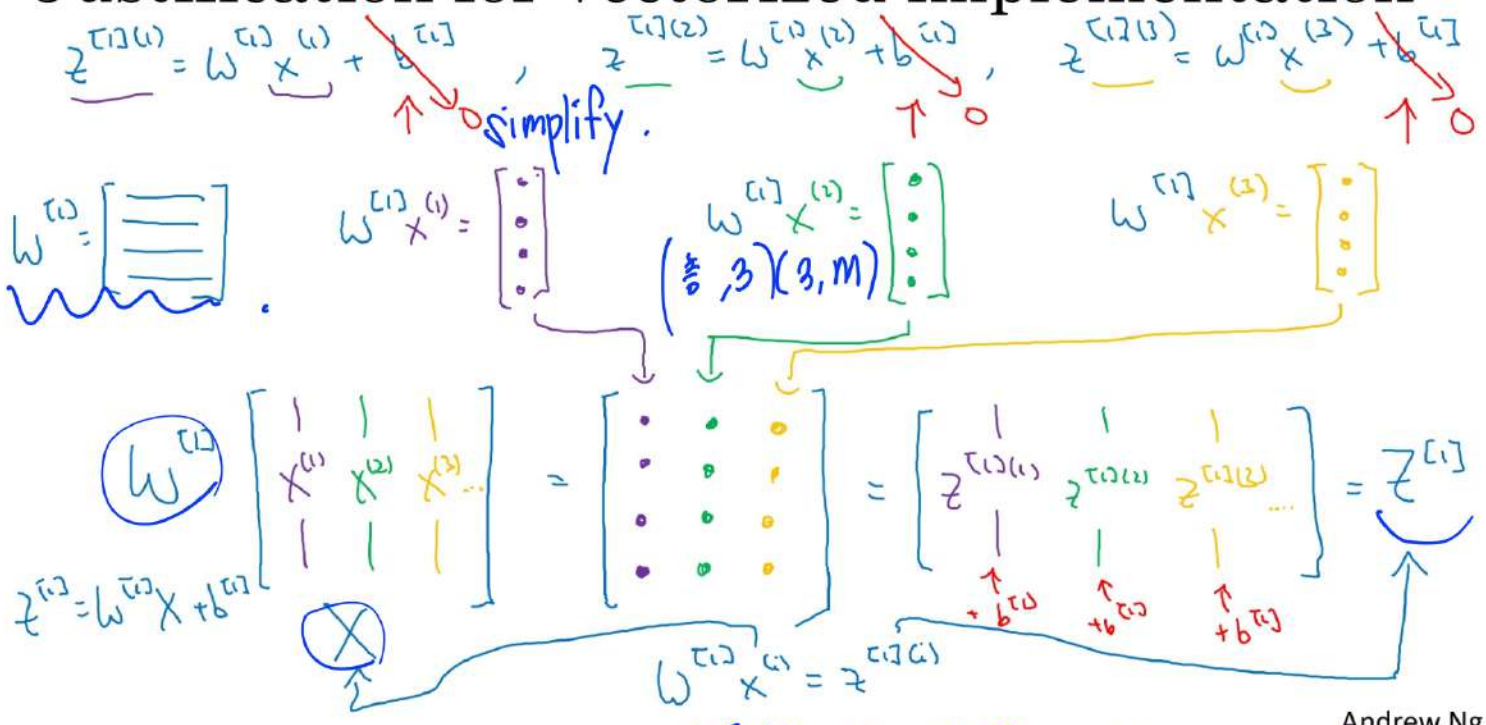
Andrew Ng

deeplearning.ai

One hidden layer Neural Network

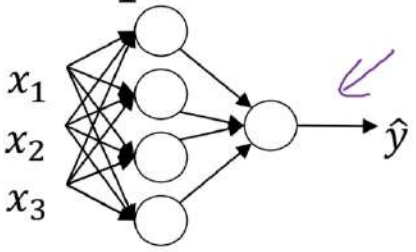
Explanation for vectorized implementation

Justification for vectorized implementation



Andrew Ng

Recap of vectorizing across multiple examples



$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

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

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

Andrew Ng

One hidden layer Neural Network

Activation functions

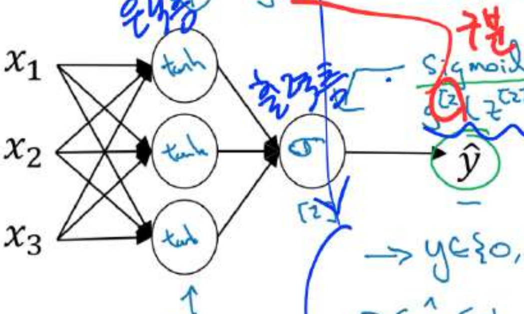
deeplearning.ai

→ 시그모이드 & tanh: z ↑↓ 이면 도함수 굉장히 작아짐 → 0.

Activation functions

시그모이드 활성화 함수를 쓰는 한 가지 예외! 이진 분류할 때 출력층의 σ

은닉 유닛에 대해! 시그모이드보다 대부분의 경우 좋다.



→ $y \in \{0, 1\}$

$0 \le \hat{y} \le 1$

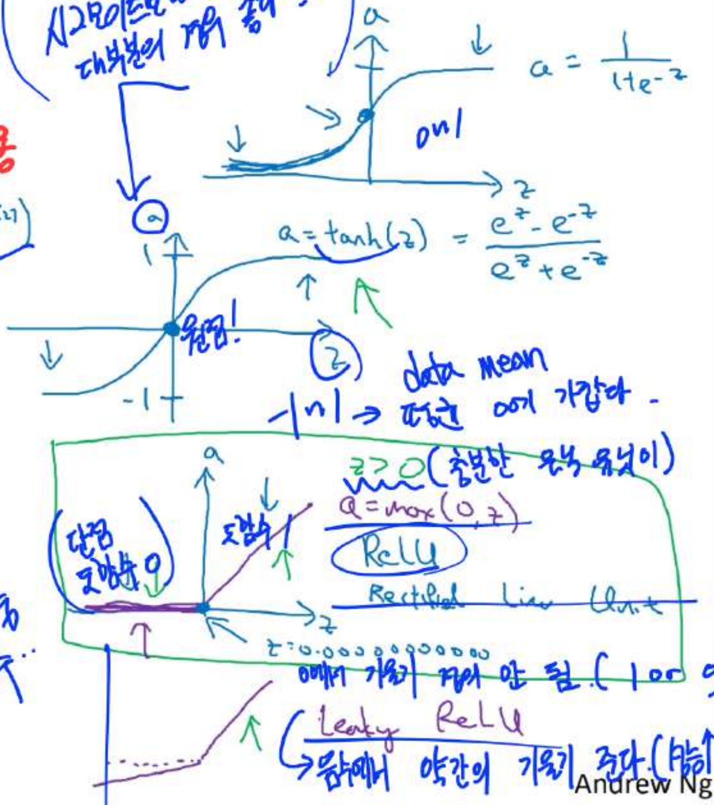
Given x:

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

$$a^{[1]} = \sigma(z^{[1]}) \quad g^{[1]}(z^{[1]})$$ 비선형 함수…

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

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

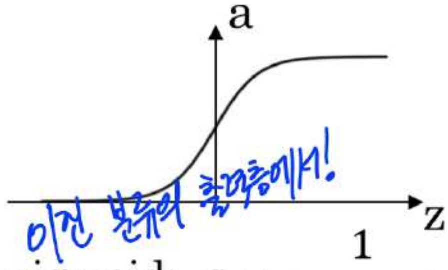


Andrew Ng

선택… (은닉층 수, 활성화 함수의 종류, 가중치 초기화)

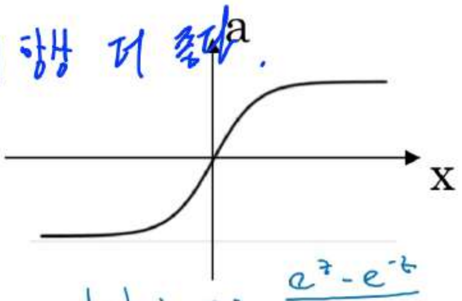
→ 활성화 함수의 기본값! 장점: 대부분의 z에 대해 기울기가 0과 매우 다름 (느린 학습 원인) → 학습 빠름 (tanh, σ 보다)

Pros and cons of activation functions

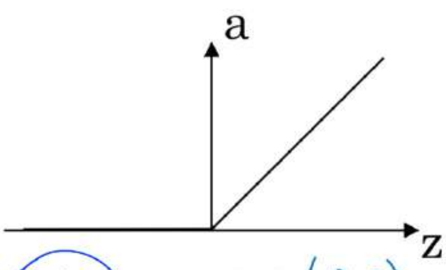


sigmoid: $a = \frac{1}{1 + e^{-z}}$

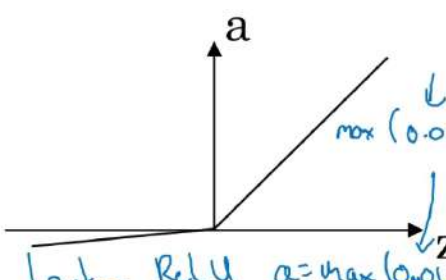
거의 항상 더 좋다.



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



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



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

Andrew Ng

deeplearning.ai

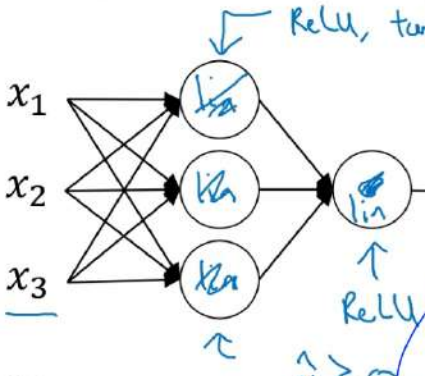
One hidden layer Neural Network

Why do you need non-linear activation functions?

$in \to \sigma$ 가 로지스틱 regression보다 나아지지 않는다!

Activation function

(선형 은닉층은 쓸모가 없다)
선형함수들의 조합은 또 다른 선형함수



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

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

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

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

$$= W'x + b'$$

Given x:

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

$g(z) = z$
"linear activation function"
항등함수

$g(z) = z$

선형함수 입력이 선형해석만을 출력!

Andrew Ng

$\hat{y}$: x에 대한 선형 함수

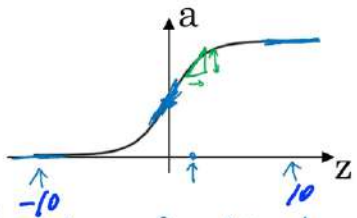
타겟 문제에 대한 머신러닝을 할 때 y가 실수값이라면 출력층에서 (앞층…?) 선형 activation 함수를 써도 괜찮다!

deeplearning.ai

One hidden layer Neural Network

Derivatives of activation functions

Sigmoid activation function



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

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

$g'(z) = \frac{d}{dz} g(z)$ = slope of $g(x)$ at z

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

$= g(z)\left(1 - g(z)\right)$ ←

$= a(1-a)$

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

$z = 10$. $g(z) \approx 1$

$\frac{d}{dz} g(z) \approx 1\,(1-1) \approx 0$

$z = -10$ $g(z) \approx 0$

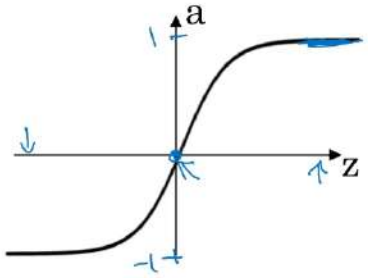
$\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$

$z = 0$ $g(z) = \frac{1}{2}$

$\frac{d}{dz} g(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4}$

Andrew Ng

Tanh activation function



$$g(z) = \tanh(z)$$

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

$g'(z) = \frac{d}{dz} g(z)$ = slope of $g(z)$ at z

$= 1 - (\tanh(z))^2$ ←

$a = g(z)$, $g'(z) = 1 - a^2$

$z = 10$ $\tanh(z) \approx 1$

$g'(z) \approx 0$

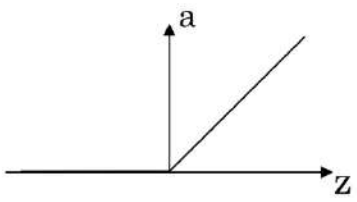
$z = -10$ $\tanh(z) \approx -1$

$g'(z) \approx 0$

$z = 0$ $\tanh(z) = 0$

$g'(z) = 1$

ReLU and Leaky ReLU



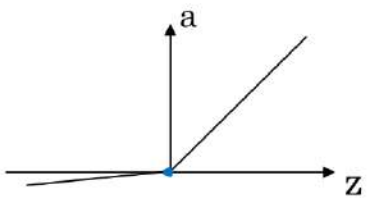
ReLU

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

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

~~undef if $z = 0$~~

$z = 0.000\ldots0$



Leaky ReLU

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

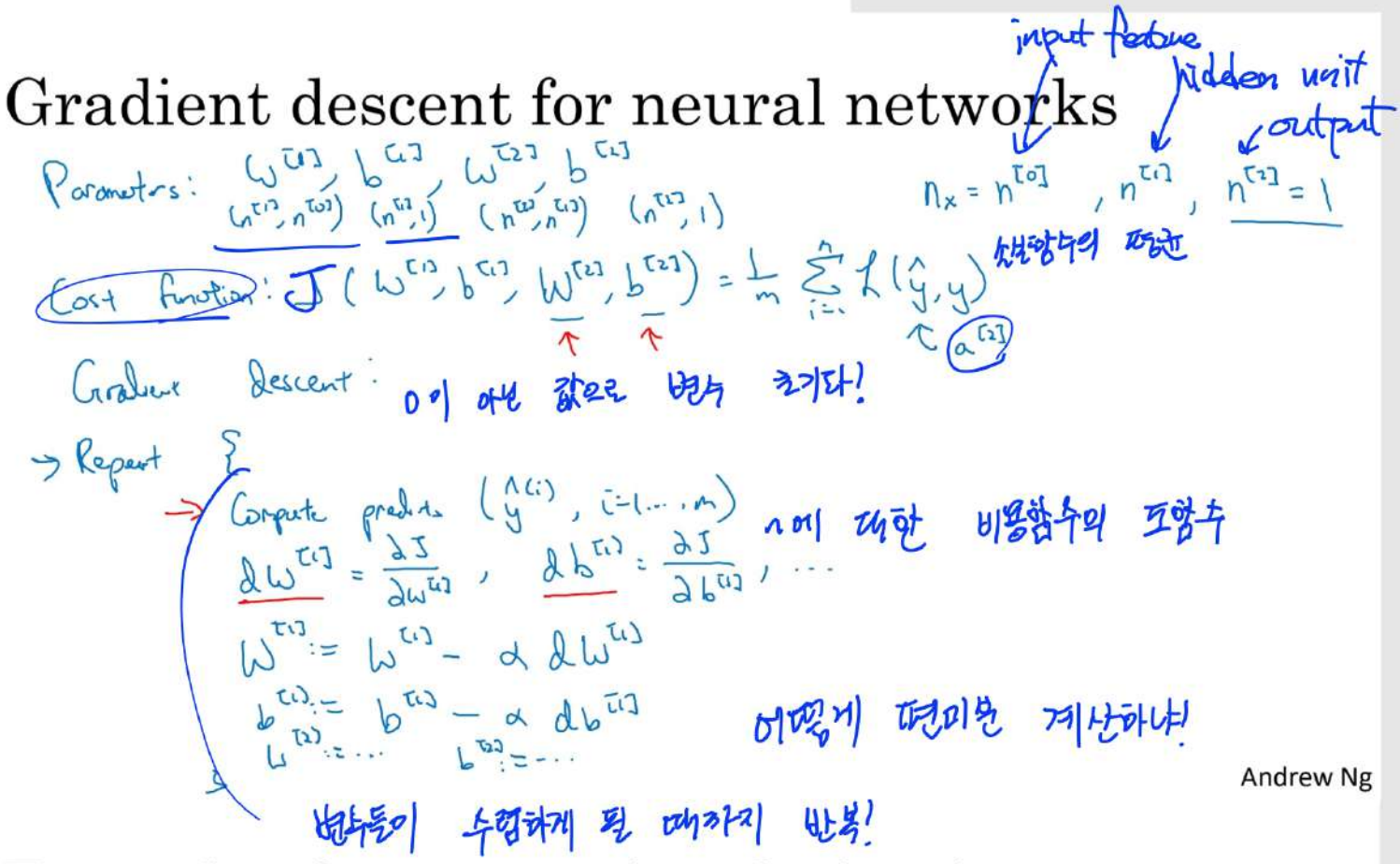
$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$

Andrew Ng

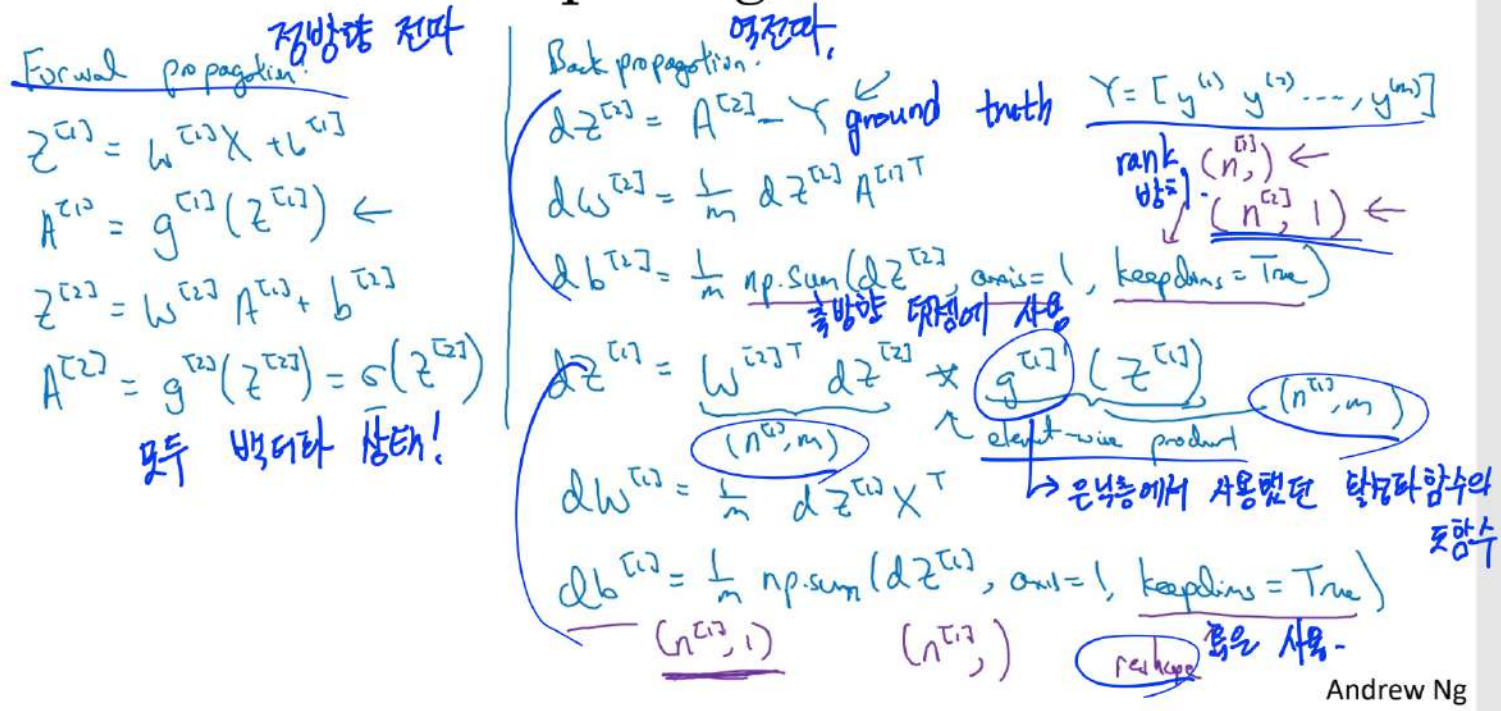
deeplearning.ai

One hidden layer Neural Network

Gradient descent for neural networks



Formulas for computing derivatives



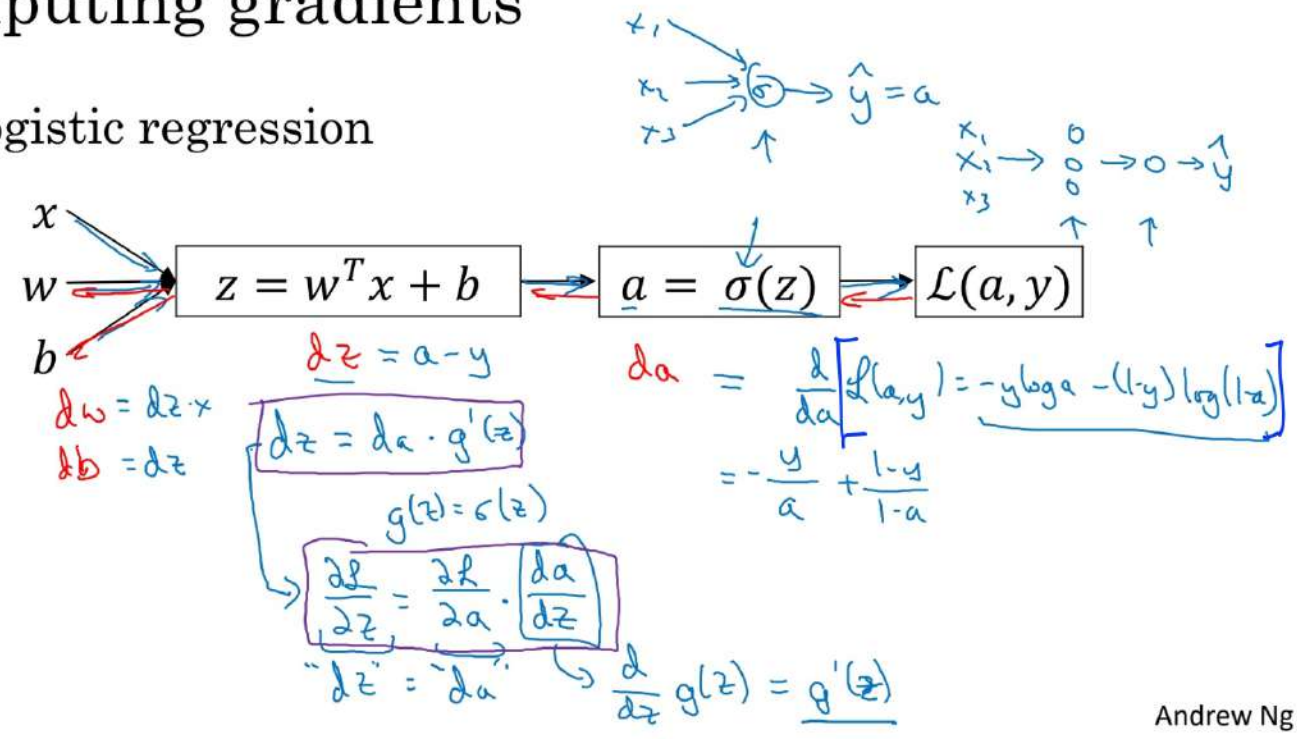
deeplearning.ai

One hidden layer Neural Network

Backpropagation intuition (Optional)

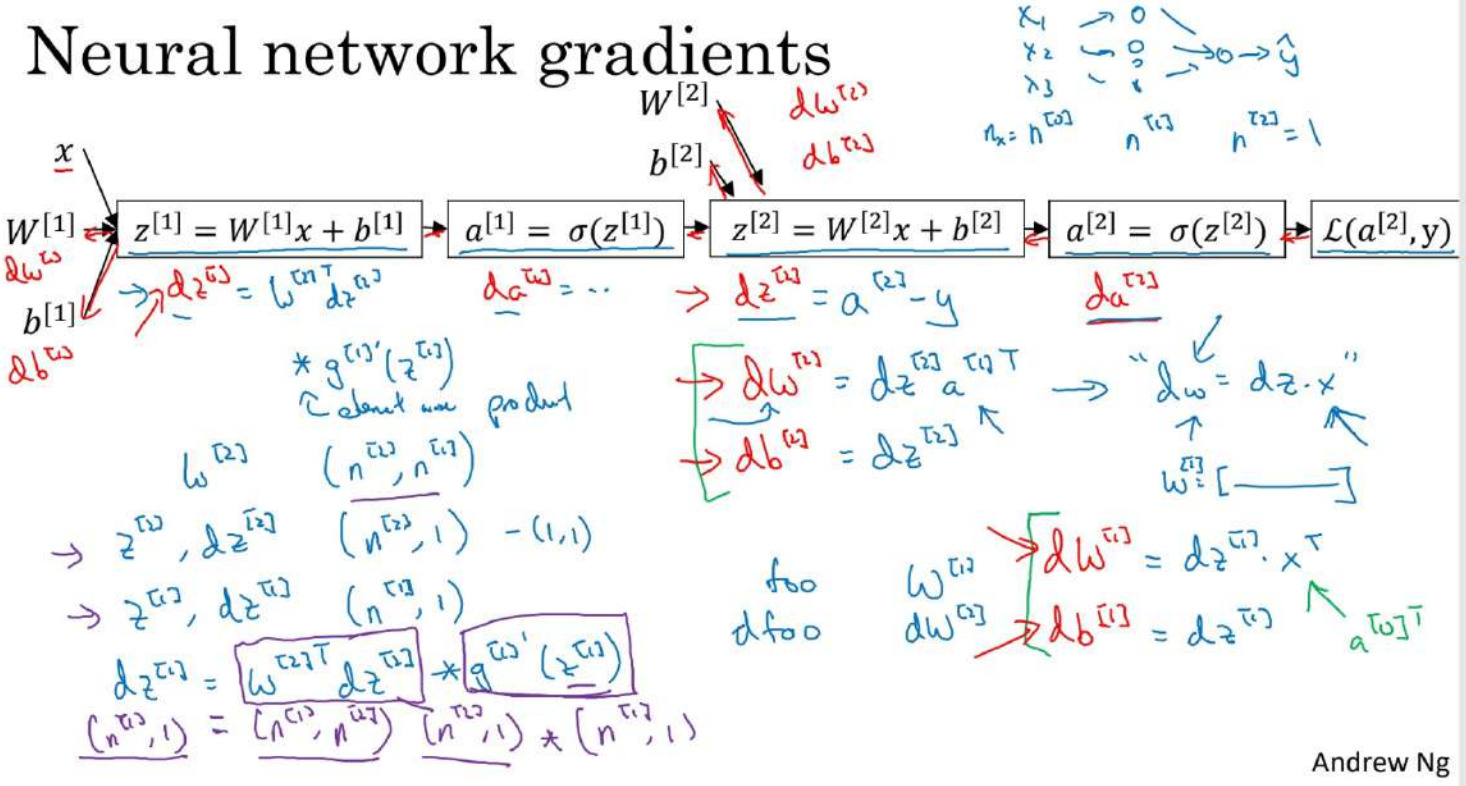
Computing gradients

Logistic regression



Andrew Ng

Neural network gradients



Andrew Ng

Summary of gradient descent

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

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

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

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

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

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

Vectorized Implementation:

$Z^{[1]} = W^{[1]}X + b^{[1]}$

$A^{[1]} = g^{[1]}(Z^{[1]})$

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

$Z^{[1]} = W^{[1]}X + b^{[1]}$

$A^{[1]} = g^{[1]}(Z^{[1]})$

$da = -\frac{y}{a} + \frac{1-y}{1-a}$

$dz = a - y$

$dw = dz\,x$

$db = dz$

x 고정 → dx 계산 완료

Andrew Ng

Summary of gradient descent

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

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

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

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

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

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

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

$dZ^{[2]} = A^{[2]} - Y$

$J(\cdot) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}, y)$

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

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

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

$(n^{[1]}, m)$ $(n^{[1]}, m)$ $(n^{[1]}, m)$

elementwise product

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

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

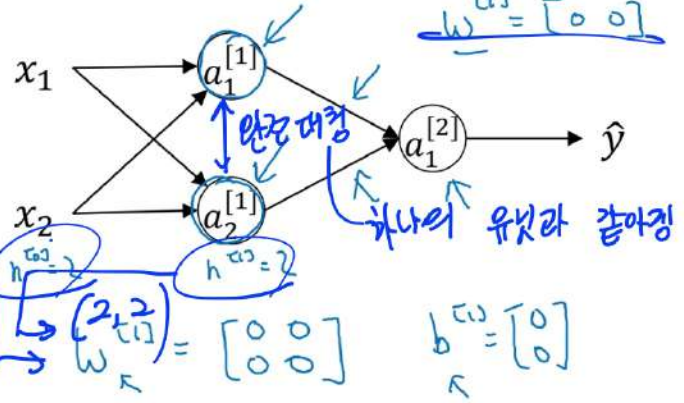
Andrew Ng

deeplearning.ai

One hidden layer Neural Network

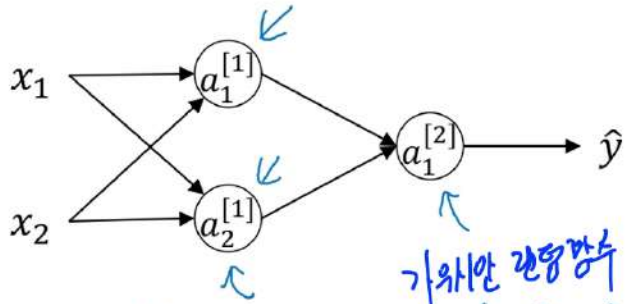
Random Initialization

What happens if you initialize weights to zero?



Andrew Ng

Random initialization



Andrew Ng