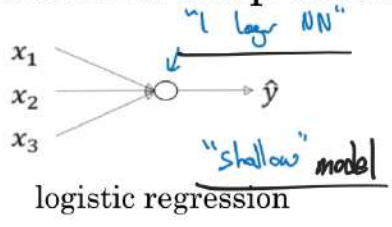
deeplearning.ai

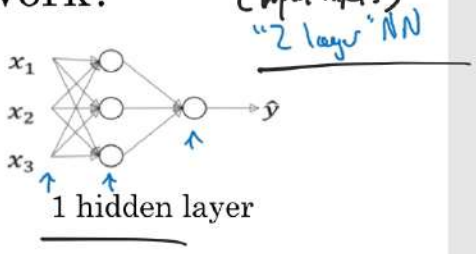
Deep Neural Networks

Deep L-layer Neural network

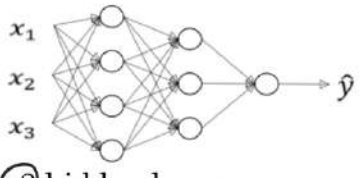
What is a deep neural network?



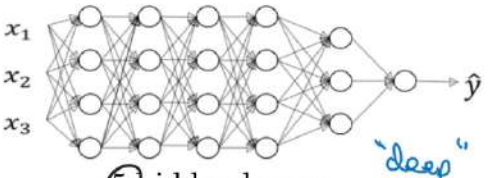
logistic regression



1 hidden layer



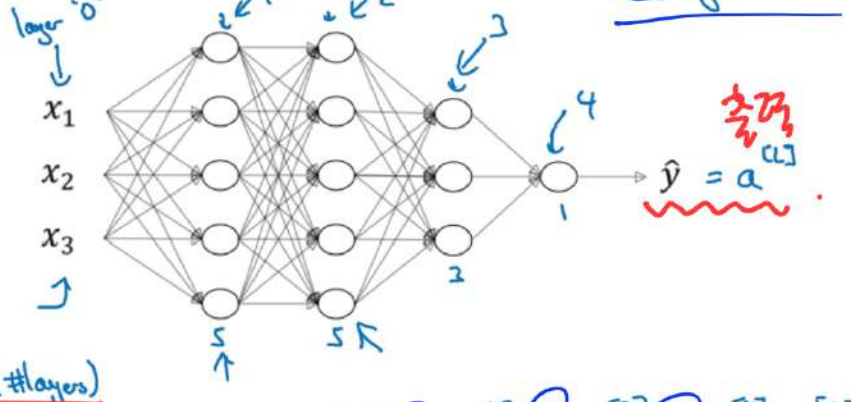
2 hidden layers



5 hidden layers

은닉층의 개수가 또 다른 하이퍼파라미터가 된다.

Deep neural network notation



L = 4 (#layers)

$n^{[l]}$ = #units in layer l

$n^{[1]} = 5$, $n^{[2]} = 5$, $n^{[3]} = 3$, $n^{[4]} = n^{[L]} = 1$

$n^{[0]} = n_x = 3$

활성값 $a^{[l]}$ = activations in layer l

$a^{[l]} = g^{[l]}(z^{[l]})$, $w^{[l]}$ = weights for $z^{[l]}$

$b^{[l]}$

Andrew

deeplearning.ai

Deep Neural Networks

Forward and backward propagation

Forward propagation for layer l

→ Input $a^{[l-1]}$ ← $W^{[l]}, b^{[l]}$

→ Output $a^{[l]}$, cache ($z^{[l]}$)

$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

Vectorized:

$Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$

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

$a^{[0]}$ $A^{[0]}$ $X = A^{[0]}$ → □ → □ → □ →

Backward propagation for layer l

→ Input $da^{[l]}$

→ Output $da^{[l-1]}, dW^{[l]}, db^{[l]}$

element-wise

$dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$

$dW^{[l]} = dz^{[l]} \cdot a^{[l-1]}$ ~~~ cache에 저장

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

$da^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$

$dz^{[l]} = W^{[l+1]T} dz^{[l+1]} * g^{[l]'}(z^{[l]})$

$dz^{[l]} = dA^{[l]} * g^{[l]'}(Z^{[l]})$

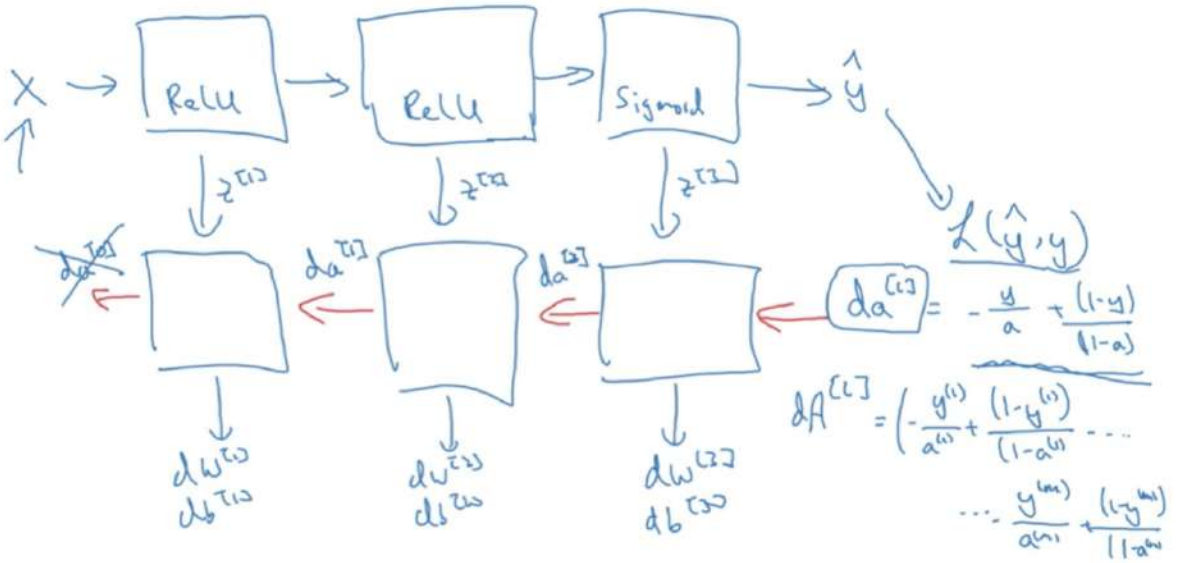
$dW^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$

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

$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$

연결해보기!!

Summary



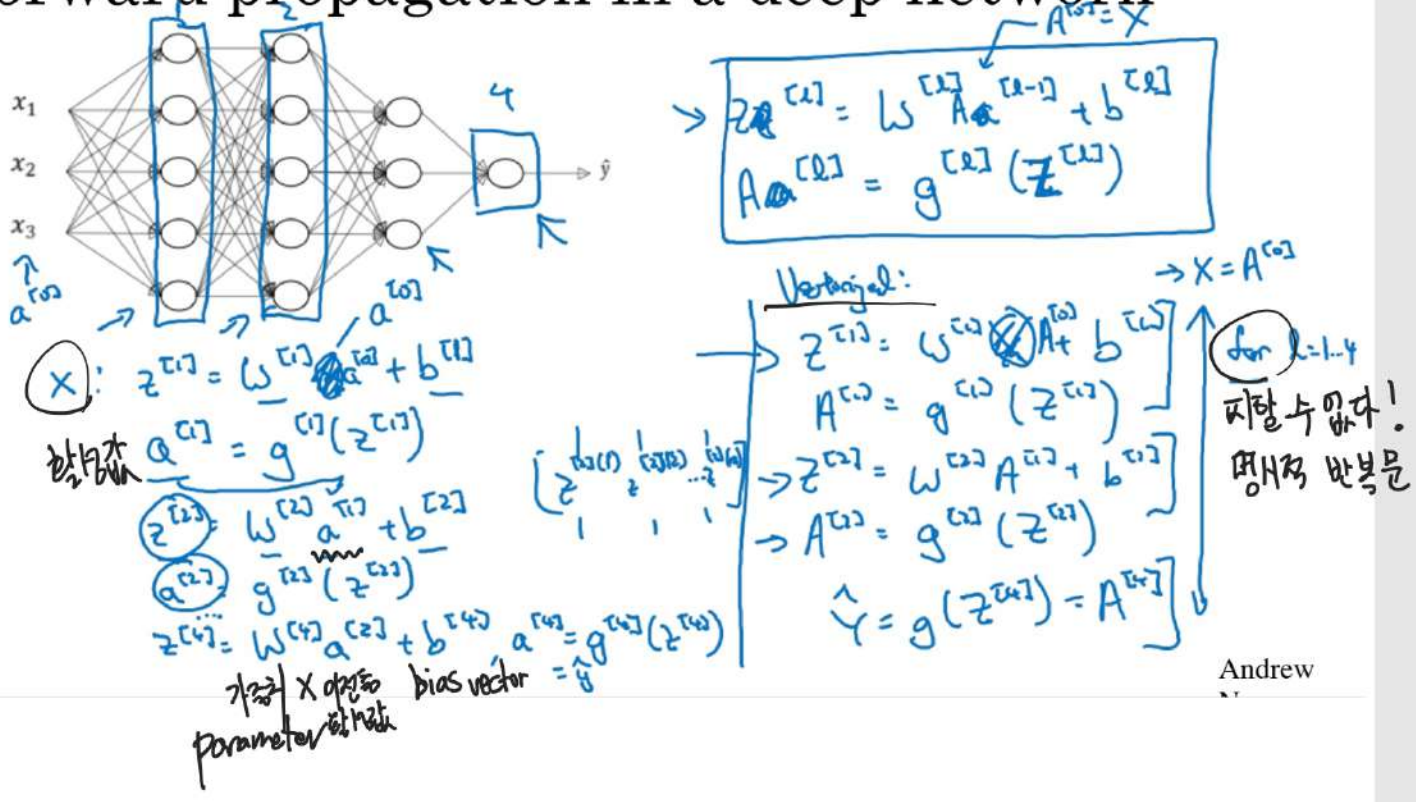
deeplearning.ai

Deep Neural Networks

Forward Propagation in a Deep Network

수식을 이용해 W 와 b 벡터가 올바른 차원을 가지는지 확인하고, 역전파를 구현하는 경우에도 dW[l] 의 차원은 W[l] 의 차원과 같은지 확인

Forward propagation in a deep network

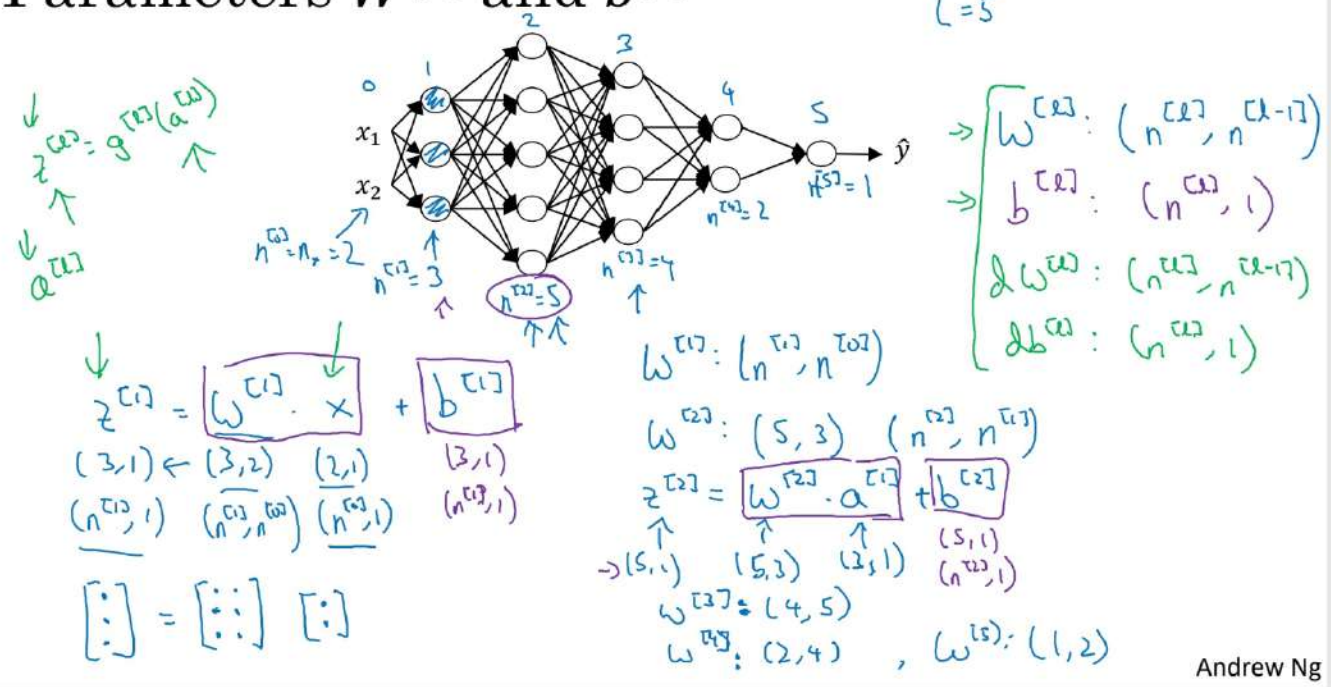


deeplearning.ai

Deep Neural Networks

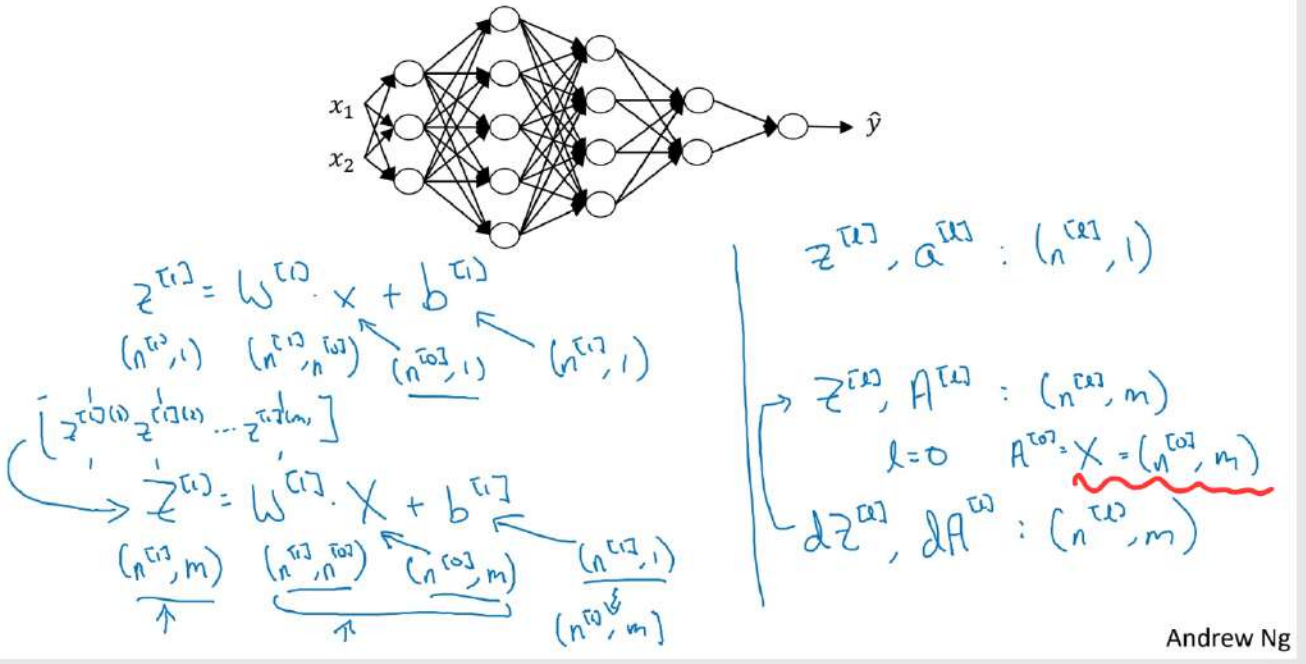
Getting your matrix dimensions right

Parameters $W^{[l]}$ and $b^{[l]}$



Andrew Ng

Vectorized implementation



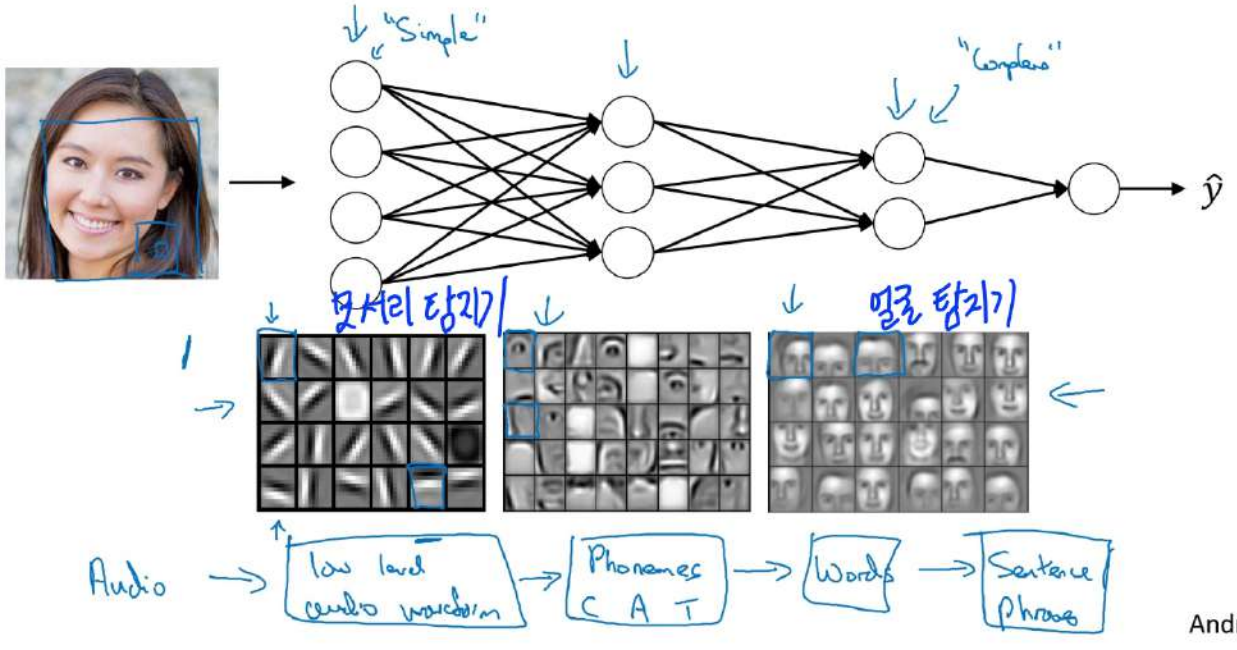
Andrew Ng

deeplearning.ai

Deep Neural Networks

Why deep representations?

Intuition about deep representation



Andrew Ng

Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute.

Andrew Ng

deeplearning.ai

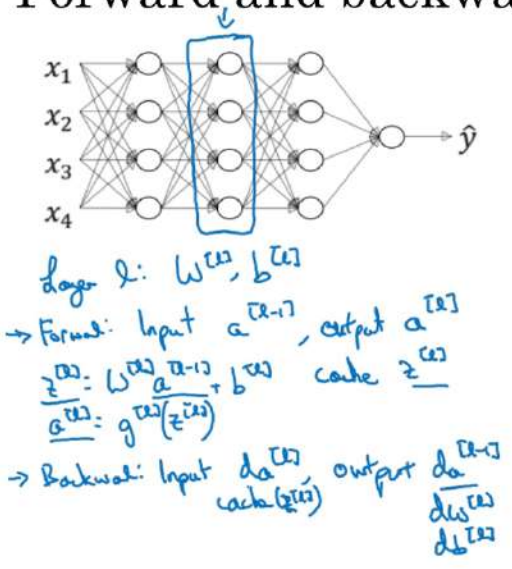
Deep Neural Networks

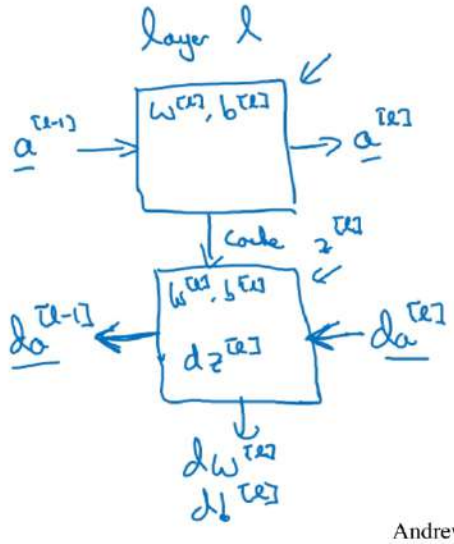
Building blocks of deep neural networks

l 번째 층에서 정방향 함수는 이전 층의 활성화 값인 a[l-1] 을 입력으로 받고, 다음 층으로 a[l] 값을 출력으로 나오게 함. 이때 선형결합된 값인 z[l] 와 변수 W[l] ,b[l] 값도 캐시로 저장

l 번째 층에서 역방향 함수는 da[l] 을 입력으로 받고, da[l] 를 출력. 이때 업데이트를 위한 dW[l] 와 db[l] 도 함께 출력. 이들을 계산하기 위해서 전방향 함수때 저장해두었던 캐시를 .

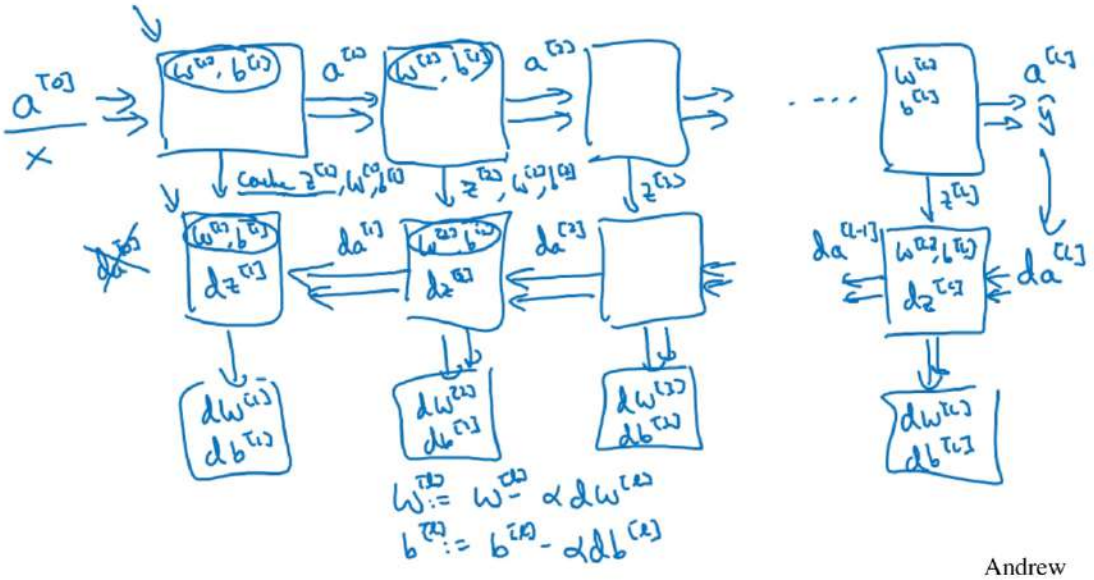
Forward and backward functions



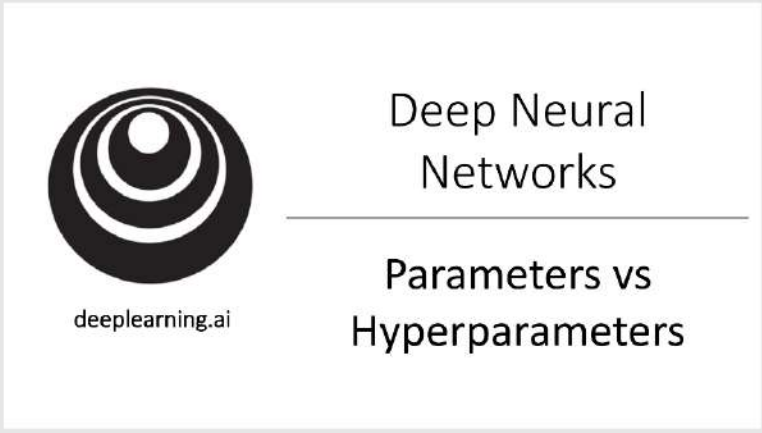


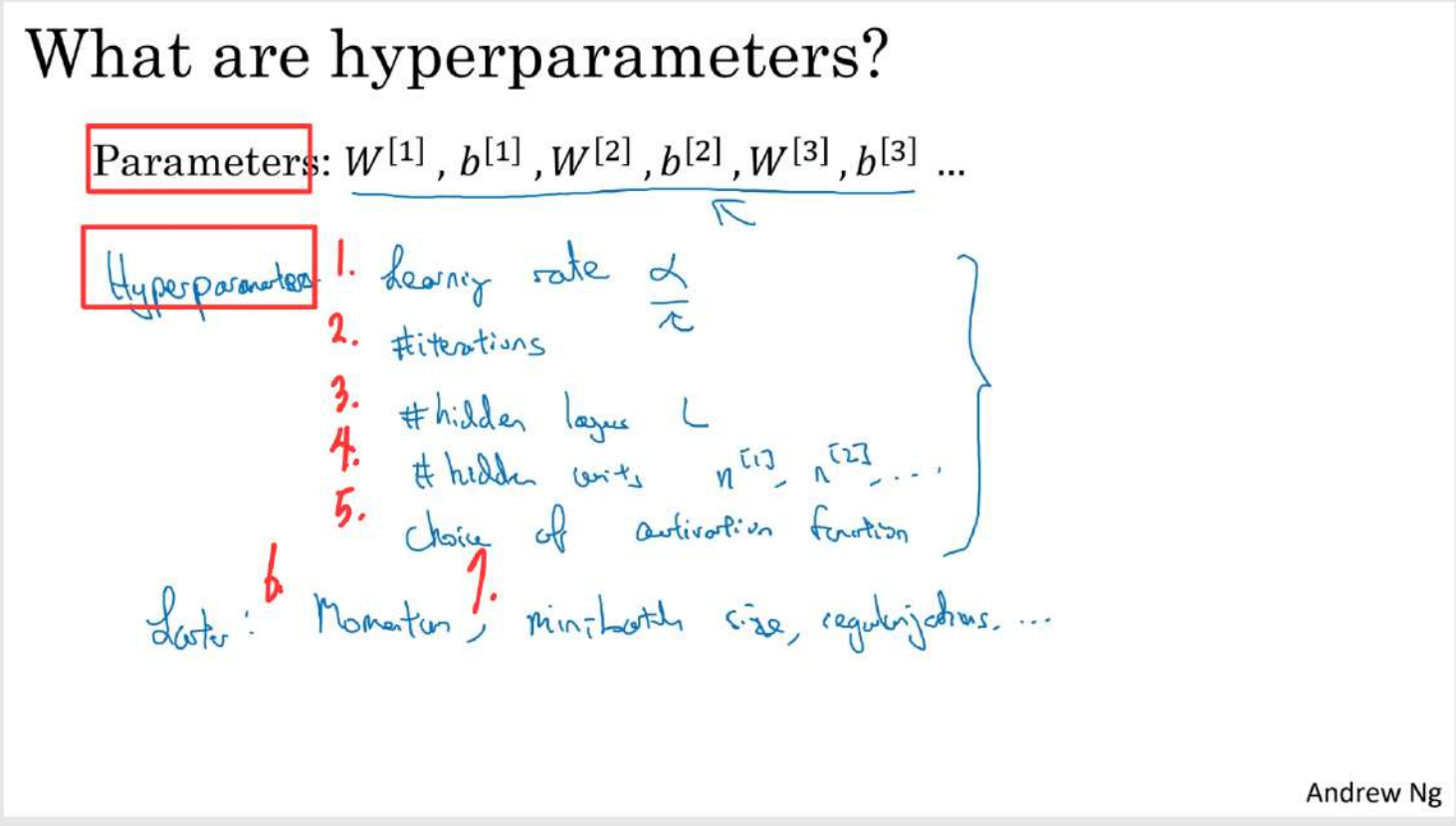
Andrew

Forward and backward functions



Andrew





매개변수인 하이퍼파라미터를 결정함으로서 최종 모델의 변수를 통제할 수 있습니다.
하이퍼파라미터는 결정된것 없, 여러번의 시도를 통해 적합한 하이퍼파라미터 를 찾아야

deeplearning.ai

Deep Neural Networks

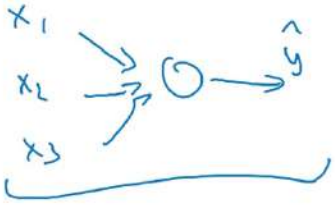
What does this have to do with the brain?

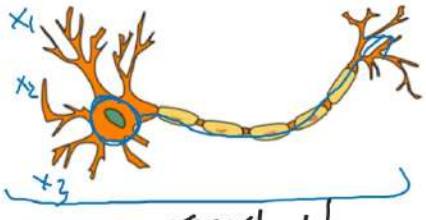
Forward and backward propagation

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

"It's like the brain."

$$
\begin{aligned}
dZ^{[L]} &= A^{[L]} - Y \\
dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\
db^{[L]} &= \frac{1}{m} np.\text{sum}(\text{d}Z^{[L]}, axis = 1, keepdims = True) \\
dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\
&\vdots \\
dZ^{[1]} &= dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\
dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\
db^{[1]} &= \frac{1}{m} np.\text{sum}(\text{d}Z^{[1]}, axis = 1, keepdims = True)
\end{aligned}
$$





뉴런은 아니다 ㅈ!

X → y

Andrew Ng