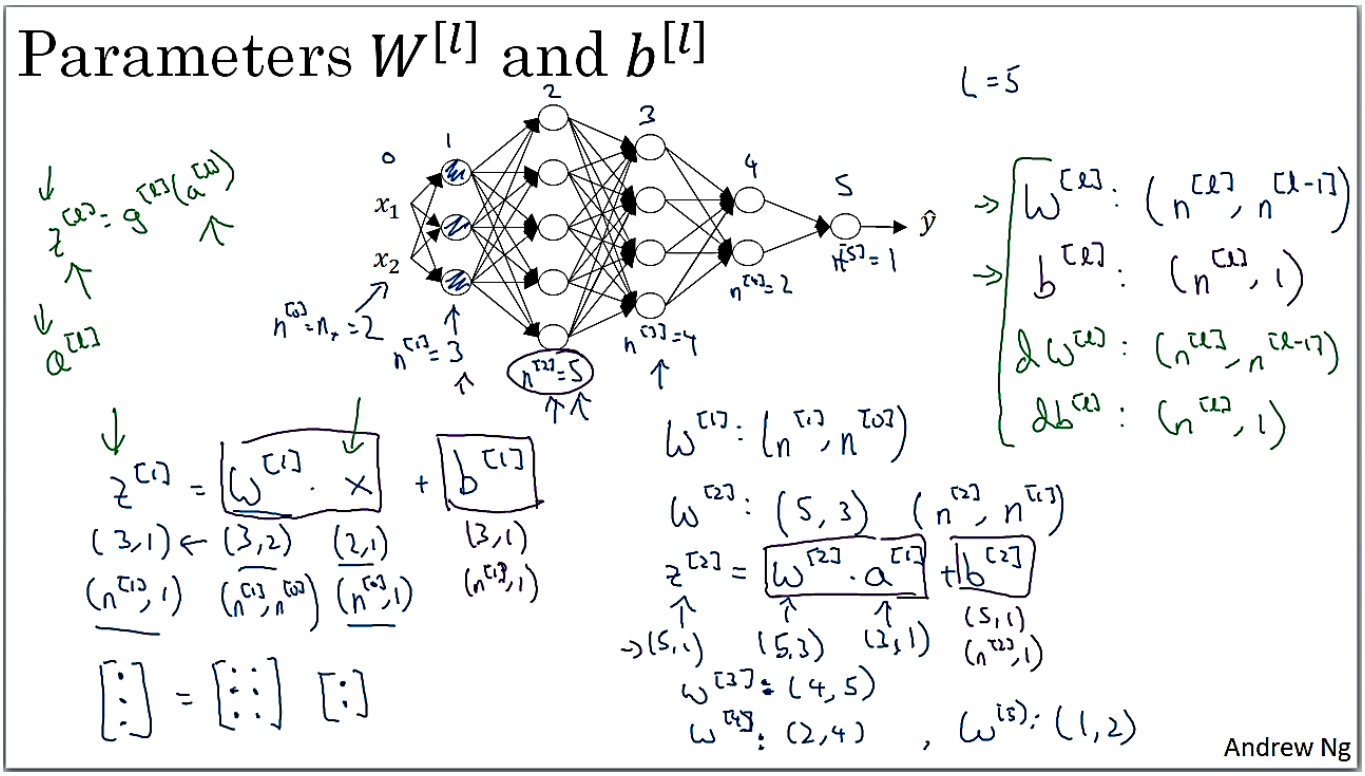
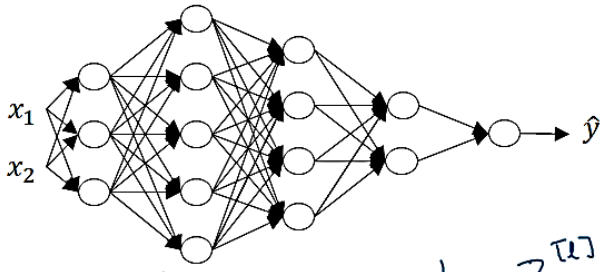
deeplearning.ai

Deep Neural Networks

Getting your matrix dimensions right



Vectorized implementation



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

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

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

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

$(n^{[1]}, m) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, m) \quad (n^{[1]}, 1) \rightarrow (n^{[1]}, m)$

$z^{[l]}, a^{[l]} : (n^{[l]}, 1)$

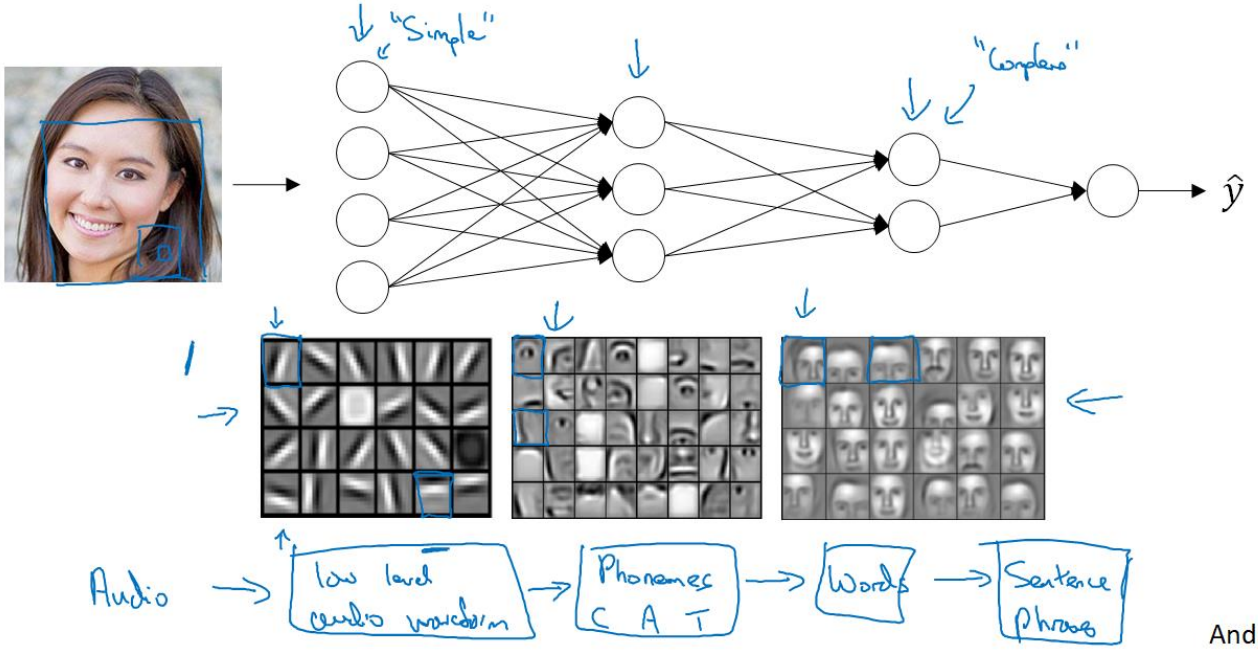
$Z^{[l]}, A^{[l]} : (n^{[l]}, m)$

$l = 0 \quad A^{[0]} = X = (n^{[0]}, m)$

$dZ^{[l]}, dA^{[l]} : (n^{[l]}, m)$

Andrew Ng

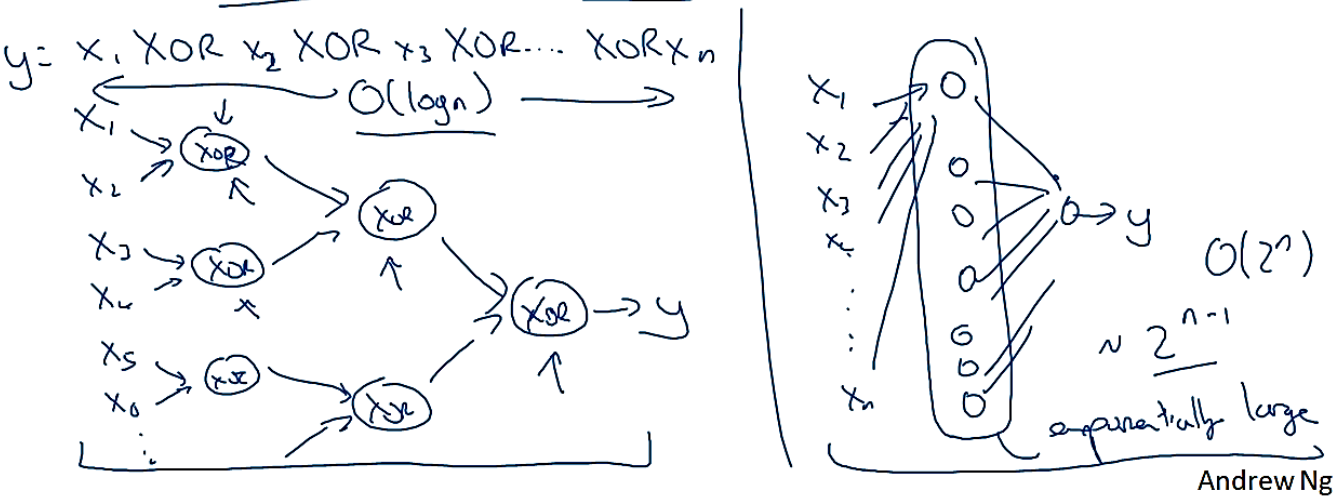
Intuition about deep representation



Andrew Ng

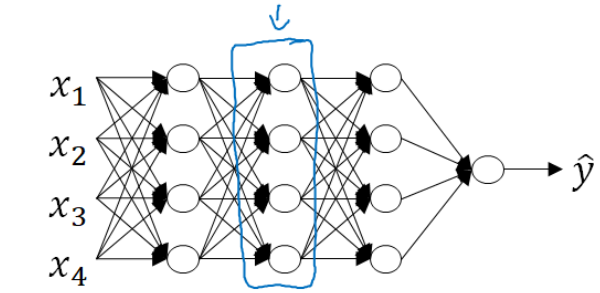
Circuit theory and deep learning

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

Forward and backward functions



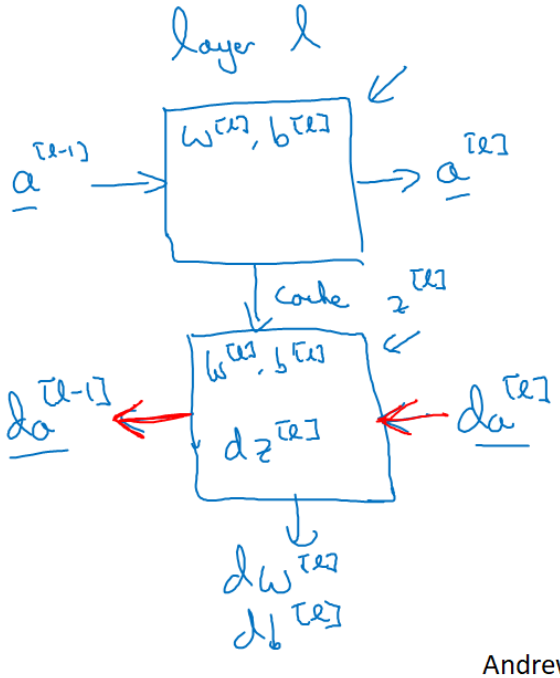
Layer l: $W^{[l]}, b^{[l]}$

→ Forward: Input $a^{[l-1]}$, output $a^{[l]}$

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

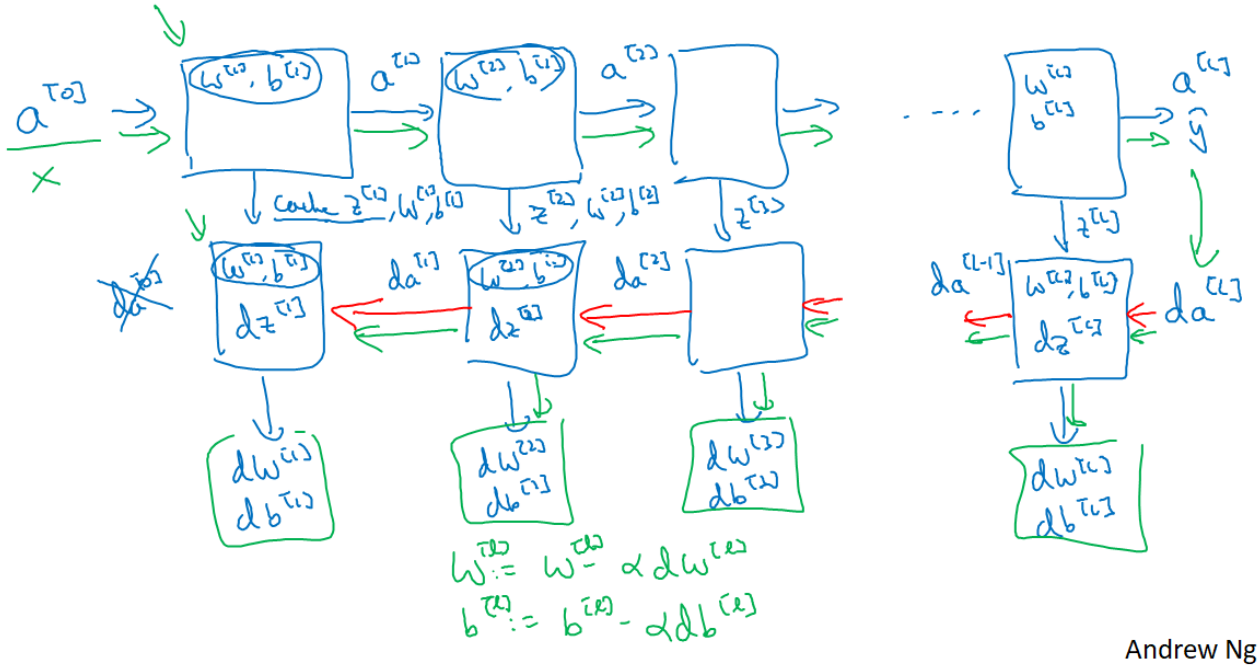
$a^{[l]} = g^{[l]}(z^{[l]})$

→ Backward: Input $da^{[l]}$, cache($z^{[l]}$), output $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$



Andrew Ng

Forward and backward functions



Andrew Ng

Forward propagation for layer l

→ Input $a^{[l-1]}$

→ Output $a^{[l]}$, cache $(z^{[l]})$

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

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

Vectorized:

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

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

$a^{[0]}$

$A^{[0]}$

$X = A^{[0]} \rightarrow [\,] \rightarrow [\,] \rightarrow [\,] \rightarrow$

Andrew Ng

Backward propagation for layer l

→ Input $da^{[l]}$

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

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

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

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

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

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

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

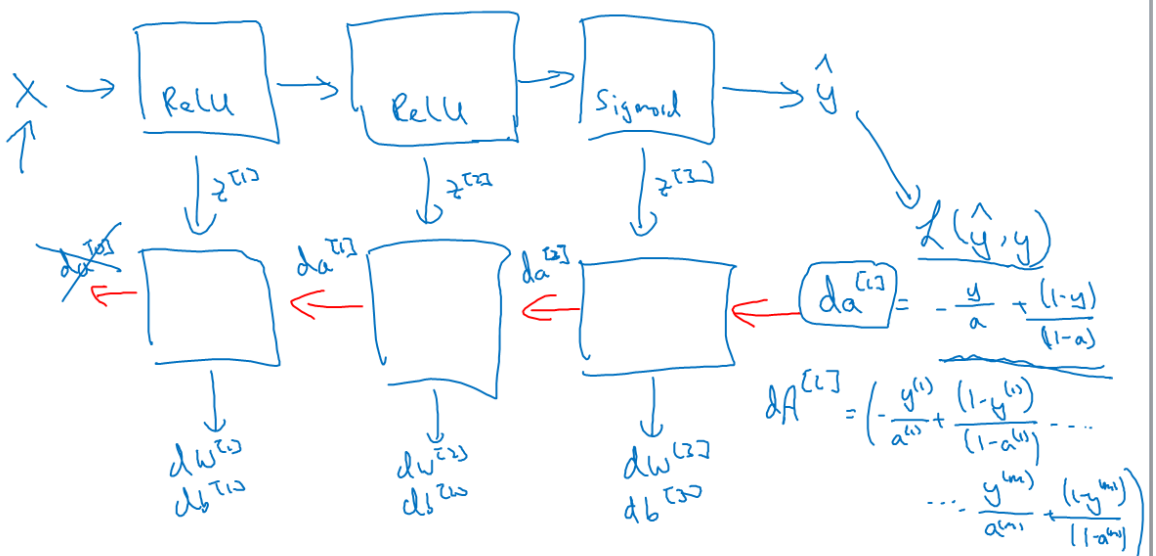
$dW^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$

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

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

Andrew Ng

Summary



Andrew Ng

What are hyperparameters?

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Hyperparameters: learning rate α

#iterations

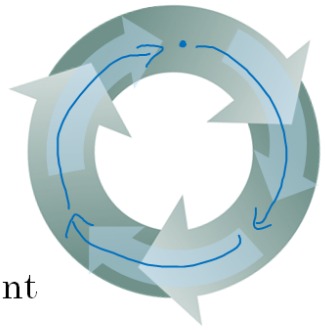
hidden layers L

hidden units $n^{[1]}, n^{[2]}, \ldots$

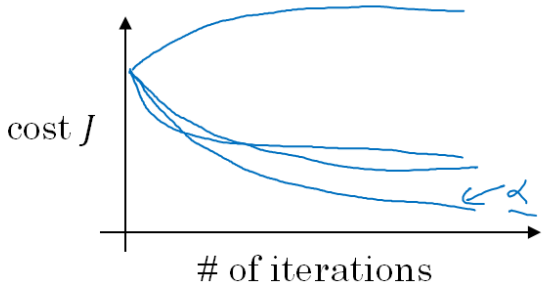
Choice of activation function

Later: Momentum, mini-batch size, regularizations, ...

Applied deep learning is a very empirical process



$\alpha = 0.04$
0.05



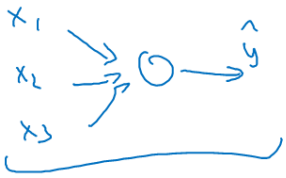
Vision, Speech, NLP, Ads, Search, Recommendations.

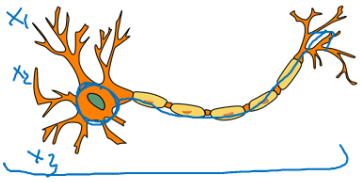
Forward and backward propagation

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

"It's like the brain"

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





X ⟶ y