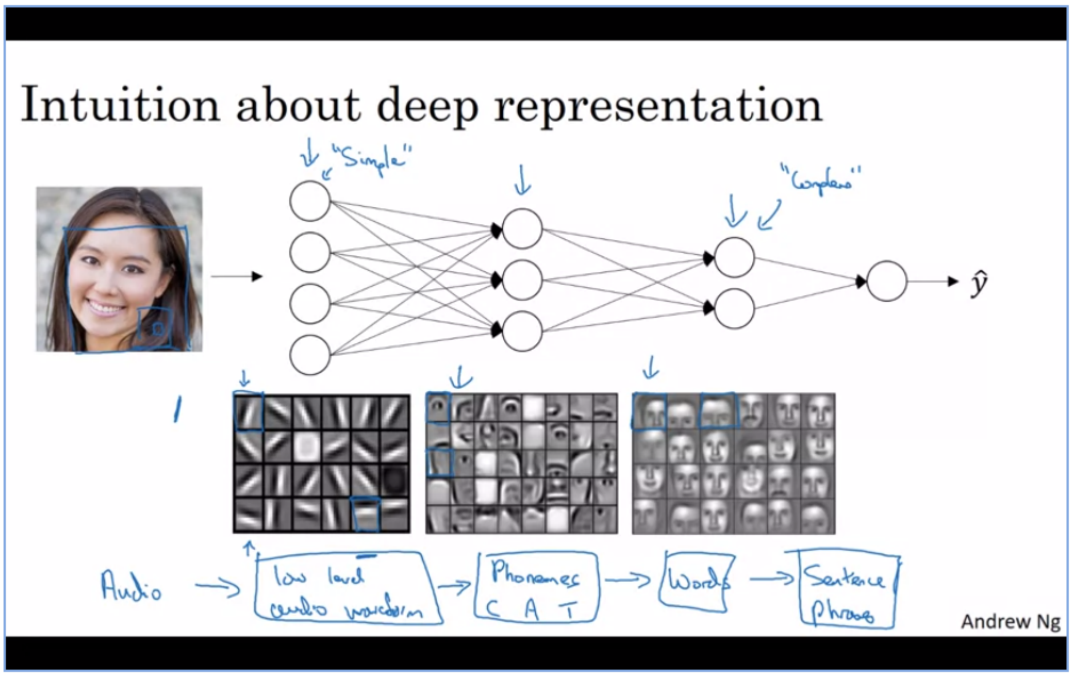
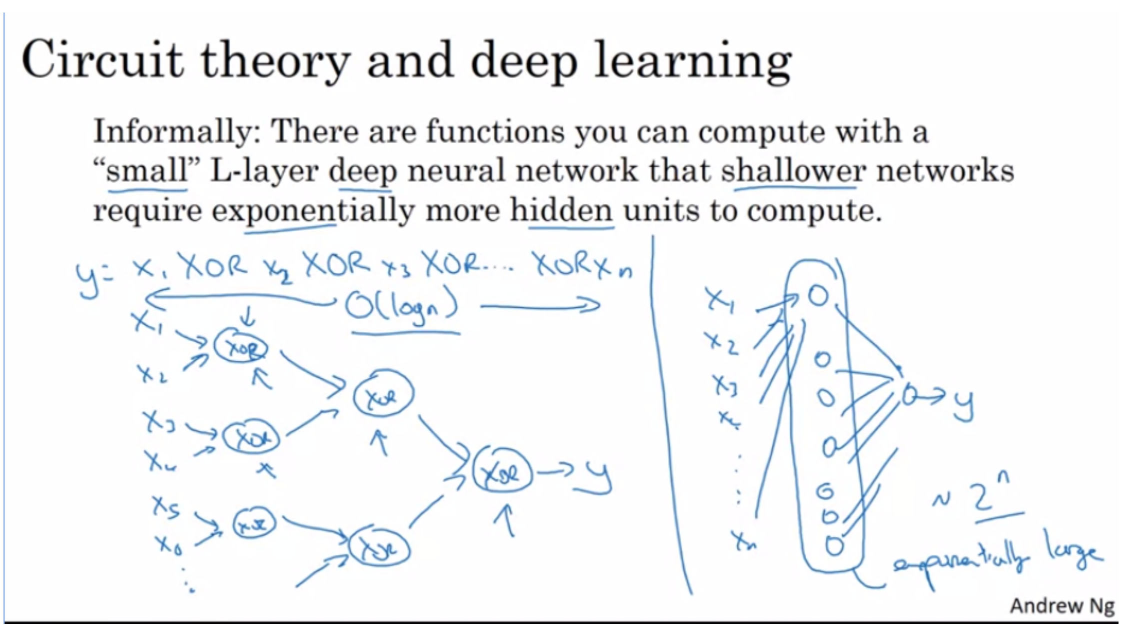
Week 4 - Deep NN

笔记本:	DL 1 - NN and DL		
创建时间:	2021/1/8 13:03	**更新时间:**	2021/1/8 14:50

Intuition

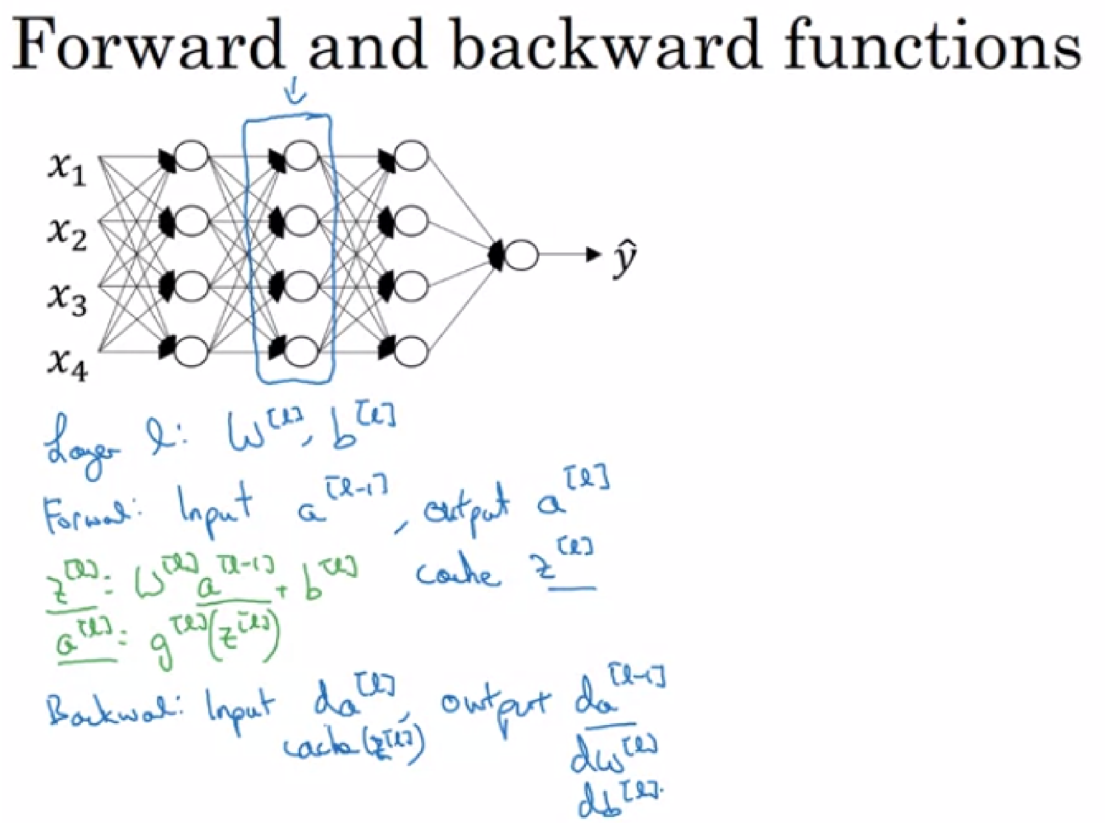
Why deep representations?



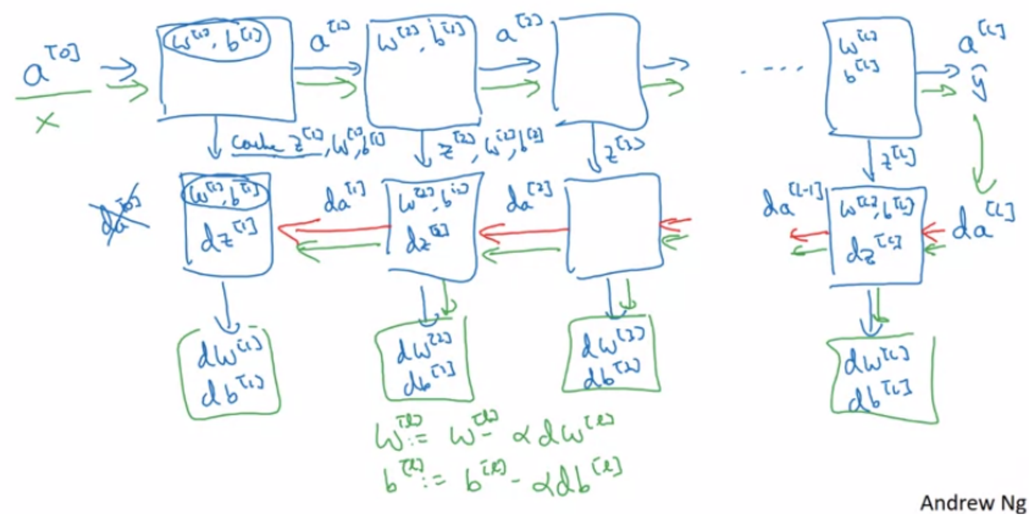


Circuit theory

Building blocks

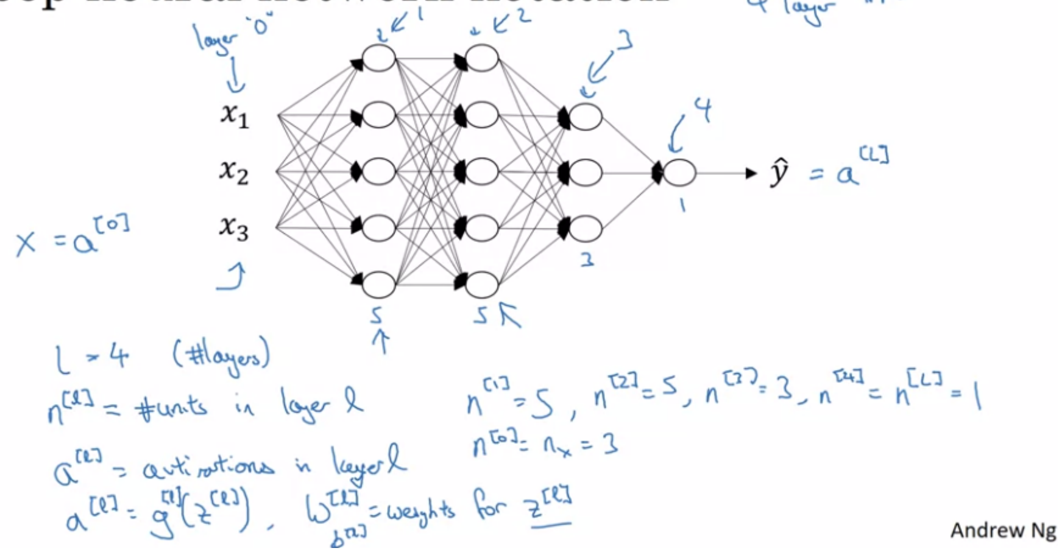


Forward and backward functions



notation

Deep neural network notation



$L = 5$

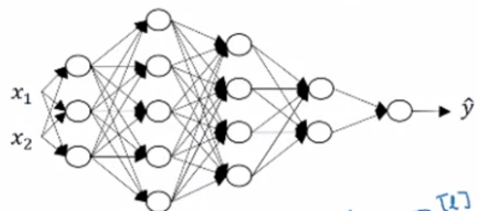
$\rightarrow W^{[l]} : (n^{[l]}, n^{[l-1]})$

$\rightarrow b^{[l]} : (n^{[l]}, 1)$

$dW^{[l]} : (n^{[l]}, n^{[l-1]})$

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

Vectorized implementation



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

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

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

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

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

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

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

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

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

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

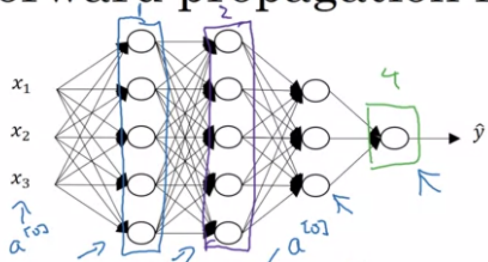
Andrew Ng

(b get broadcasted)

---

forward

Forward propagation in a deep network



x: $z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$

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

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

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

$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}$, $a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

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

$A^{[0]} = X$

$$\boxed{\begin{aligned} Z^{[l]} &= W^{[l]} A^{[l-1]} + b^{[l]} \\ A^{[l]} &= g^{[l]}(Z^{[l]}) \end{aligned}}$$

Vectorized:

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

for $l = 1..4$

$Z^{[1]} = W^{[1]} \cancel{X} A^{[0]} + b^{[1]}$

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

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

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

$\hat{Y} = g(Z^{[4]}) = A^{[4]}$

Andrew Ng

Forward propagation for layer l

- Input $a^{[l-1]}$
- Output $a^{[l]}$, cache $(z^{[l]})$ $W^{[l]}, b^{[l]}$

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

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

Vectorized:

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

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

$a^{[0]}$

$A^{[0]}$

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

Andrew Ng

Backward propagation for layer l

→ Input $da^{[l]}$

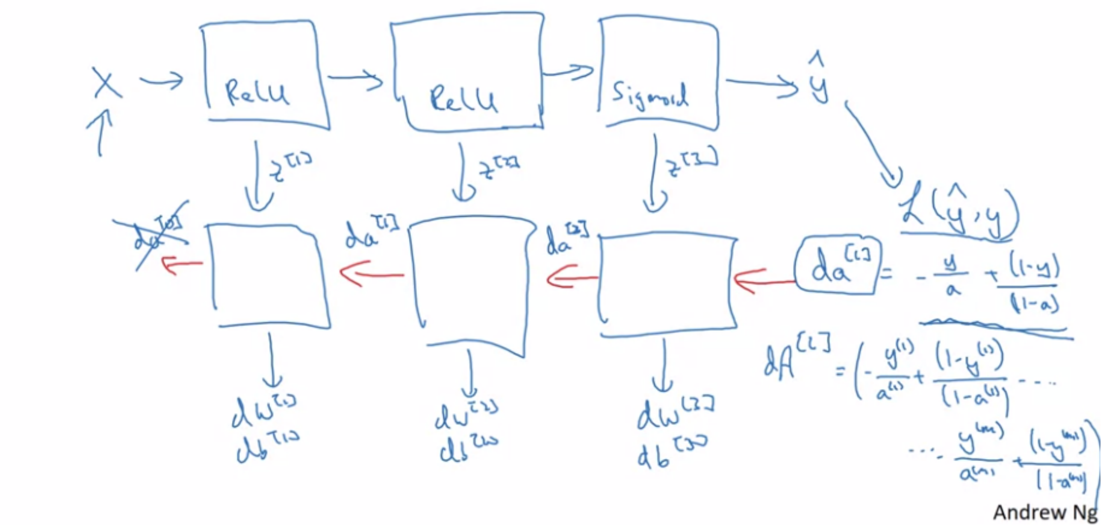
→ Output $\boxed{da^{[l-1]}}$, $dW^{[l]}$, $db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$
$$dW^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$
$$db^{[l]} = dz^{[l]}$$
$$\boxed{da^{[l-1]}} = W^{[l]T} \cdot dz^{[l]}$$
$$dz^{[l]} = W^{[l+1]T} dz^{[l+1]} * g^{[l]\prime}(z^{[l]})$$

$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$
$$dW^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]T}$$
$$db^{[l]} = \frac{1}{m}\, \text{np.sum}(dZ^{[l]}, \text{axis}=1, \text{keepdims}=\text{True})$$
$$\boxed{dA^{[l-1]}} = W^{[l]T} \cdot dZ^{[l]}$$

Andrew Ng

Summary



hyperpara, more in C2

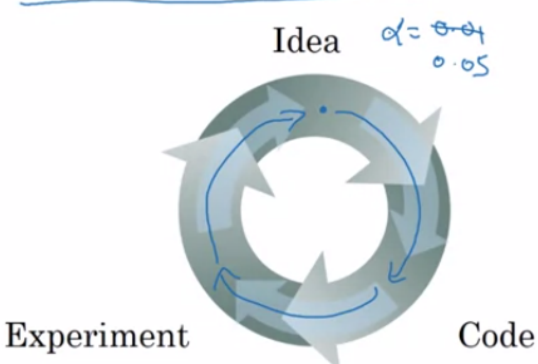
What are hyperparameters?

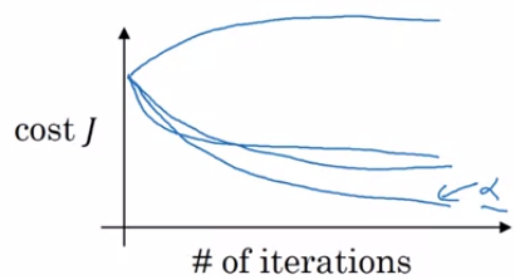
Parameters: $\underline{W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}}$ ...

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





Vision, Speech, NLP, Ad, Search, Recommendation.

deeplearning.ai

Deep Neural Networks

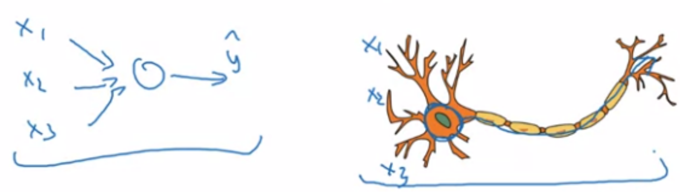
What does this have to do with the brain?

Forward and backward propagation

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

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

"It's like the brain"



Andrew Ng

actually what a single neuron does is still a mystery, NN is more like learning very flexible functions, very complex functions to learn X to Y mappings