deeplearning.ai

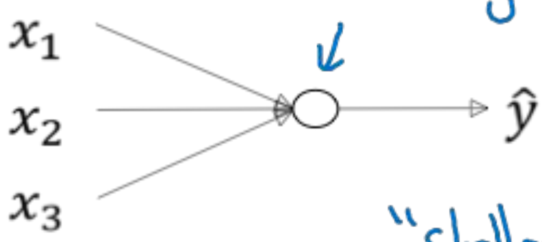
Deep Neural Networks

Deep L-layer Neural network

(NN with L layers)

What is a deep neural network?

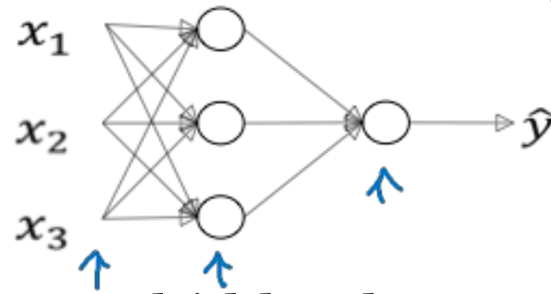
"1 layer NN"



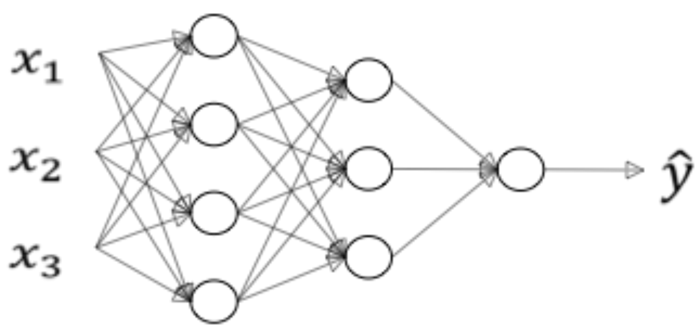
"shallow"

logistic regression

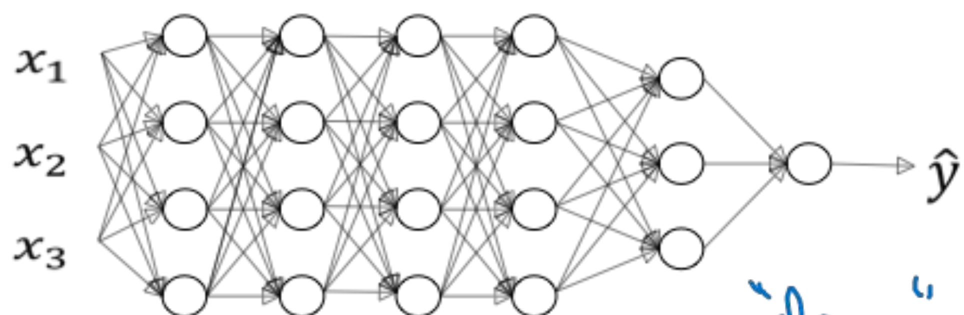
"2 layer" NN



1 hidden layer



2 hidden layers

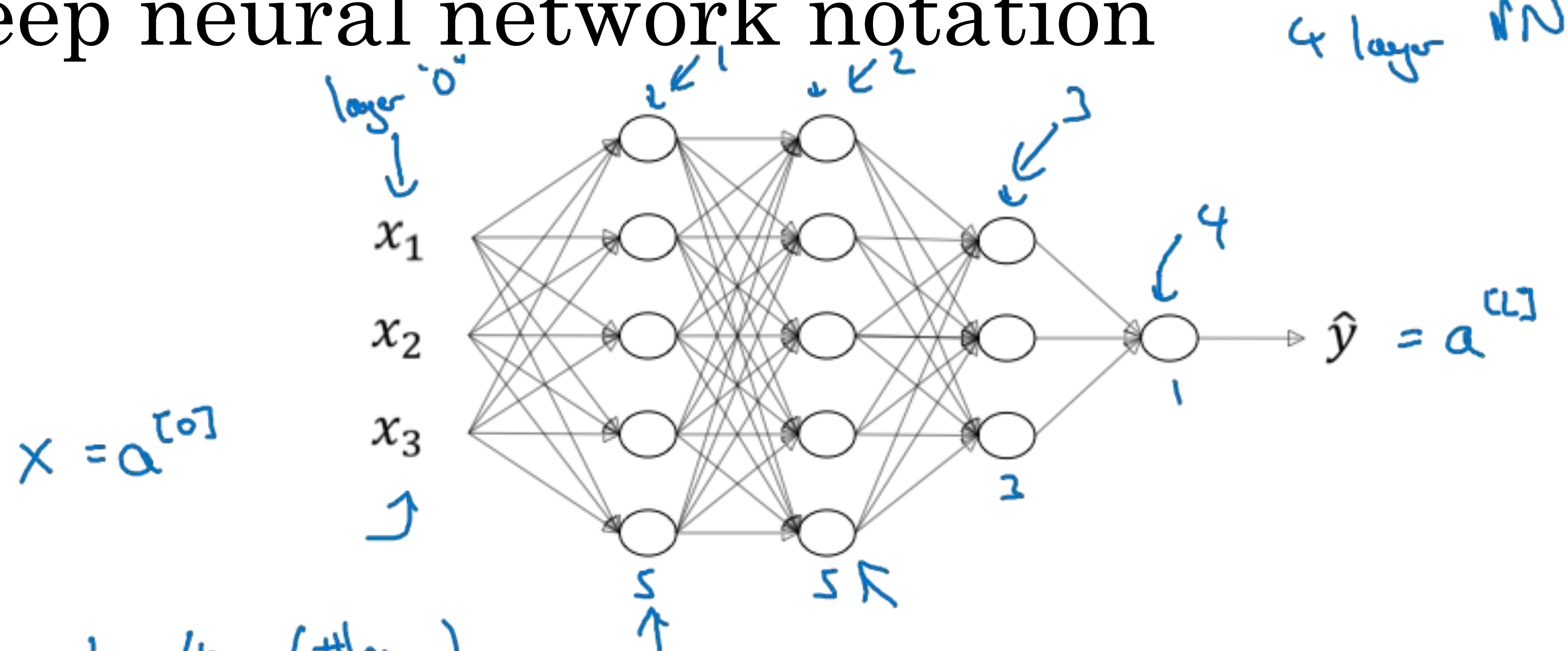


"deep"

5 hidden layers

Deep neural network notation

4 layer NN



$L = 4$ (#layers)

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

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

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

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

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

$b^{[l]}$

activation type for layer l

Andrew N

deeplearning.ai

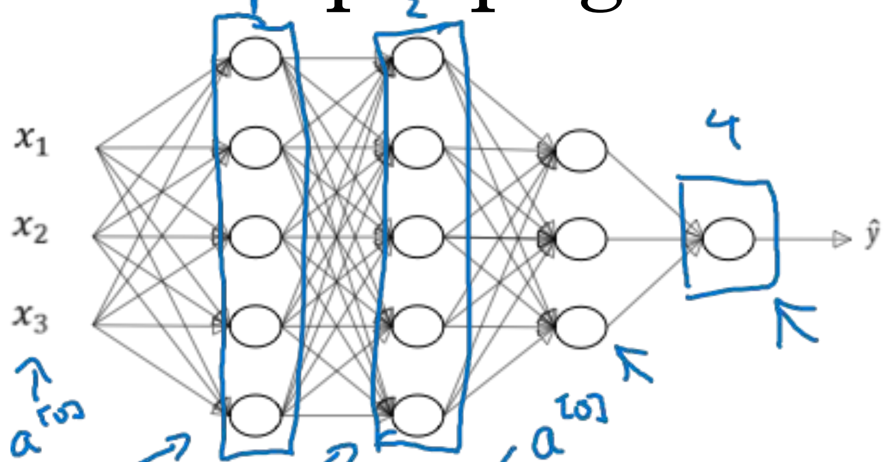
Deep Neural Networks

Forward Propagation in a Deep Network

In general the forward propagation for a deep NN can be written like this.

- Note that when computing F.P. the activations of consecutive layers are computed one after the other. There is no way of eliminating these iterations (i.e. vectorizing the whole network).

Forward propagation in a deep network



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

$A^{[0]} = X$

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

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

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

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

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

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

Vectorized:

$X = A^{[0]}$

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

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

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

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

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

for $l = 1..4$

Andrew Ng

deeplearning.ai

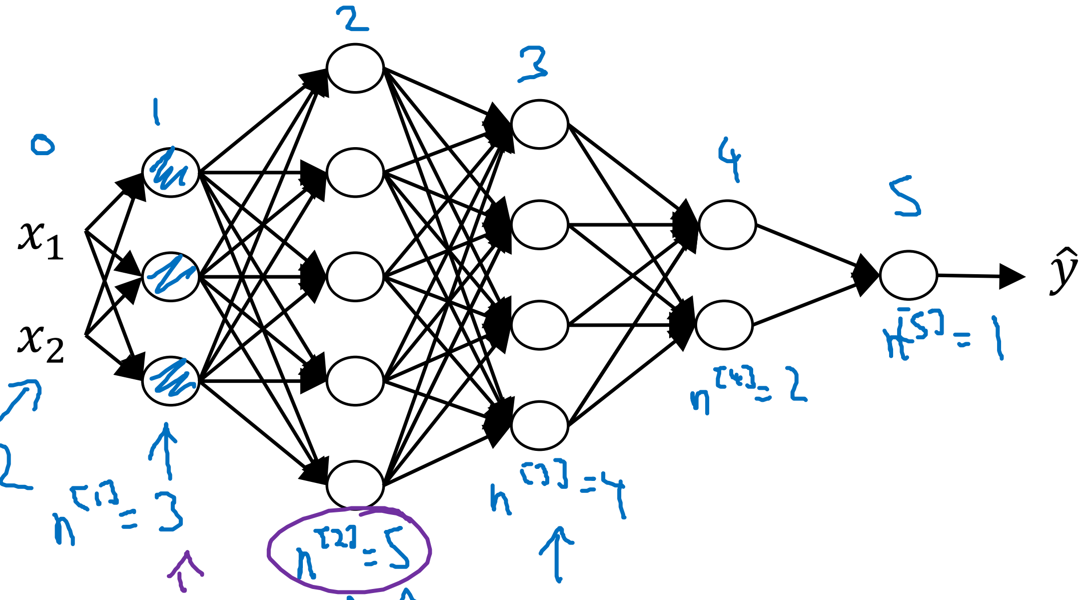
Deep Neural Networks

Getting your matrix dimensions right

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

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

$a^{[l]}$



$L = 5$

THE DIMENSION OF THE PARAMETERS AT THE LAYER L ARE

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

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

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

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

... and their gradients always have the SAME dimensions

$\dim(W) = \dim(dW)$

$\dim(b) = \dim(db)$

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

$(3,1) \leftarrow (3,2) \quad (2,1) \quad (3,1)$

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

$$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$$

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

$W^{[2]} : (5, 3) \quad (n^{[2]}, n^{[1]})$

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

$(5,1) \quad (5,3) \quad (3,1) \quad (5,1)$

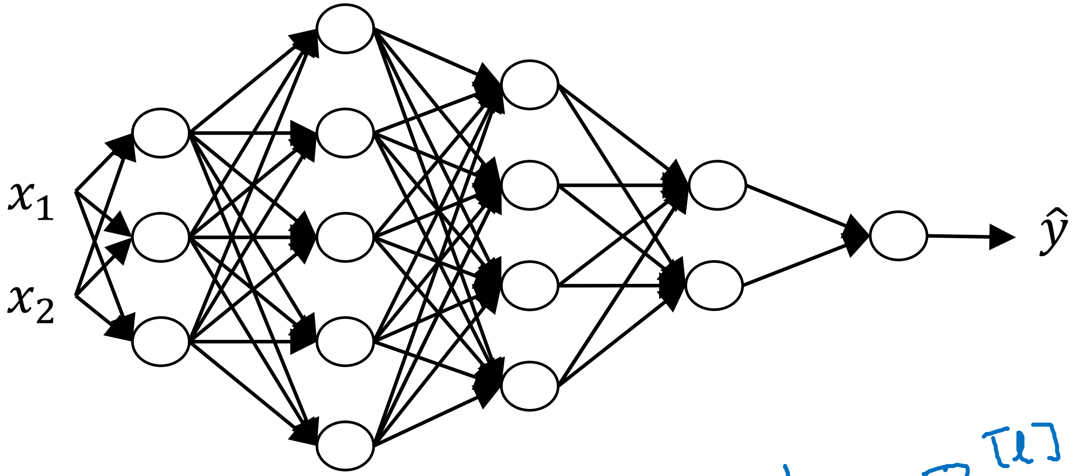
$(n^{[2]}, 1)$

$W^{[3]} : (4, 5)$

$W^{[4]} : (2, 4) \quad , \quad W^{[5]} : (1, 2)$

Andrew Ng

Vectorized implementation



THE DIMENSIONS OF THE ACTIVATIONS ARE

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

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

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

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

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

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

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

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

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

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

↑ as usual, their gradients have the same dimension.

Andrew Ng

deeplearning.ai

Deep Neural Networks

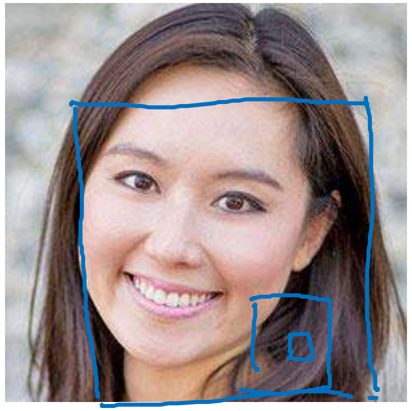
Why deep representations?

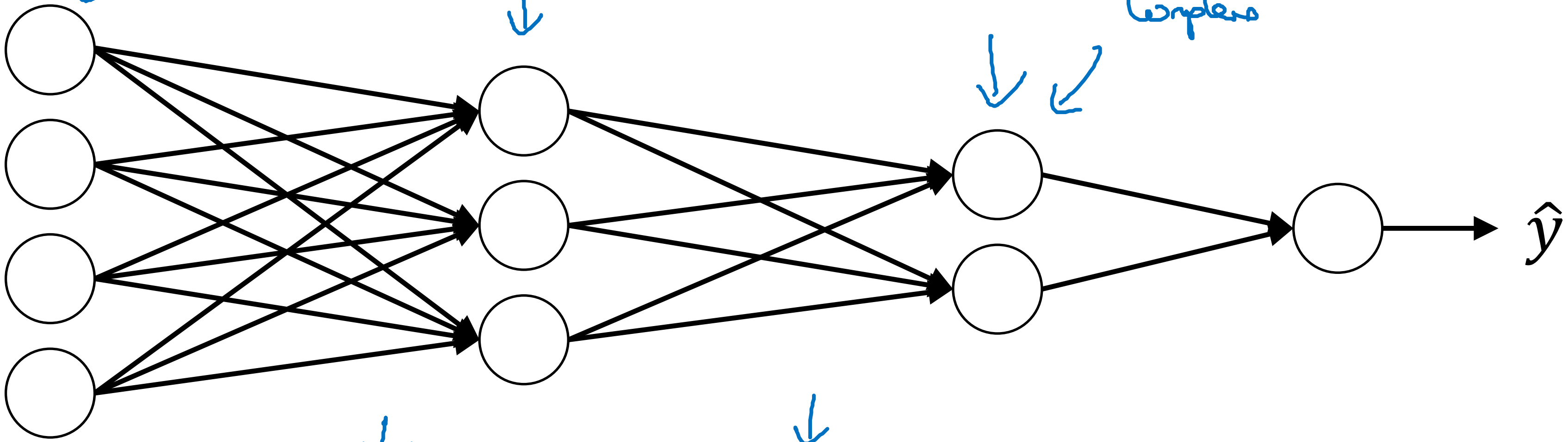
Intuition about deep representation

- EARLIER LAYERS' WEIGHTS LEARN TO DETECT SIMPLER (LOW LEVEL) FEATURES FROM THE INPUT.
- LATER LAYERS DETECT MORE COMPLEX (HIGH LEVEL) FEATURES FROM THE INPUT.

"Simple"

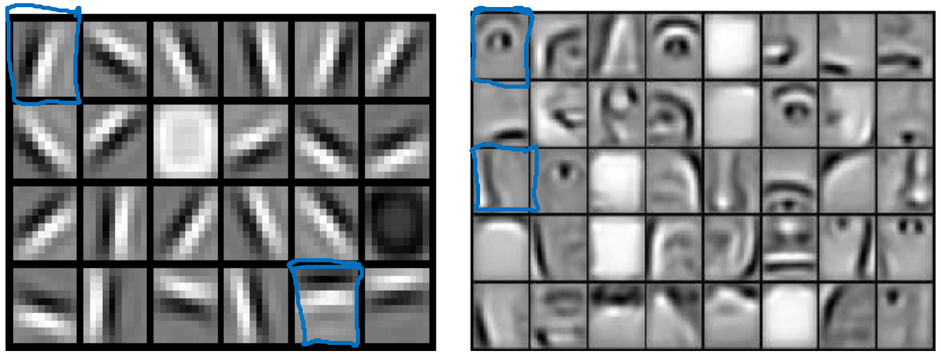
"Complex"





example applications:

face recognition



speech recognition:

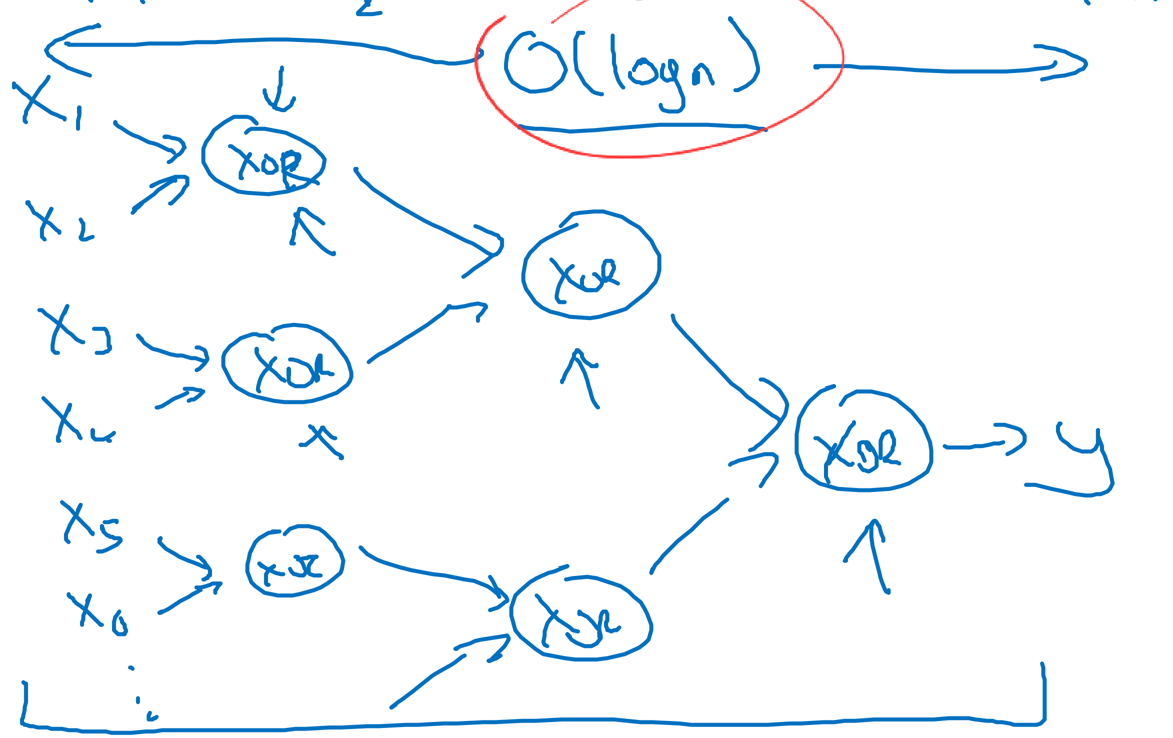
Audio → low level audio waveforms → Phonemes C A T → Words → Sentence phrase

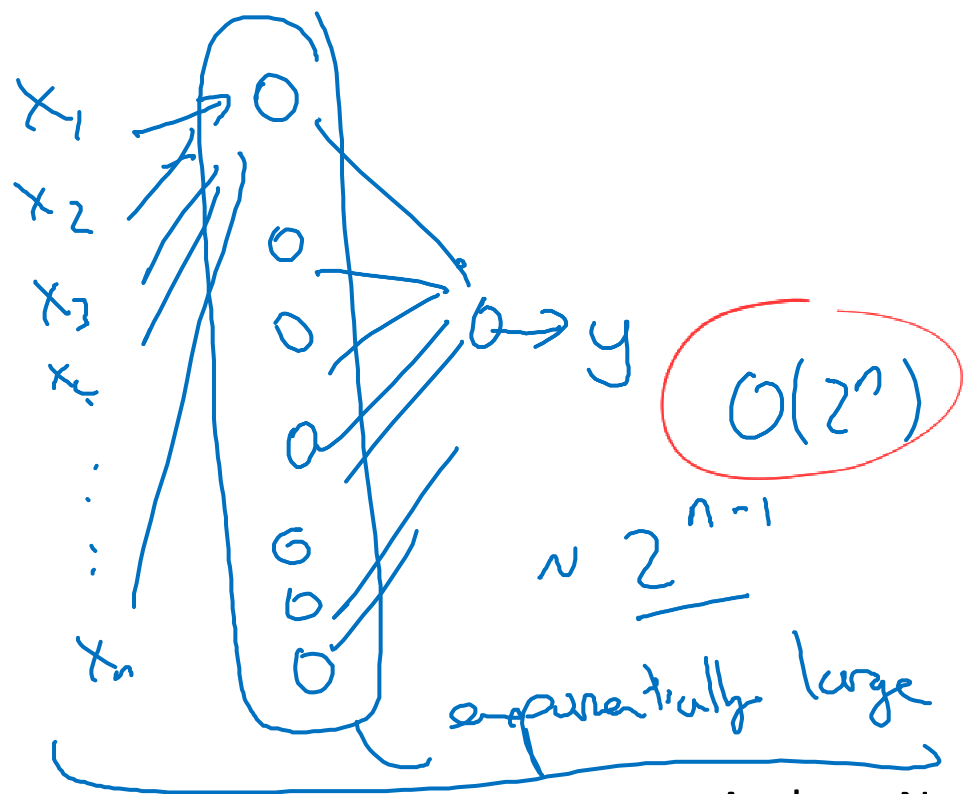
Circuit theory and deep learning

- The complexity of a network increases much more with depth rather than with width.
- To simulate a "narrow but deep" network with a "wide but shallow" NN we might need exponentially more neurons.

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.

$y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \text{ XOR} \ldots \text{ XOR } x_n$





- How to write FP & BP for a general layer l and being efficient by caching useful values.

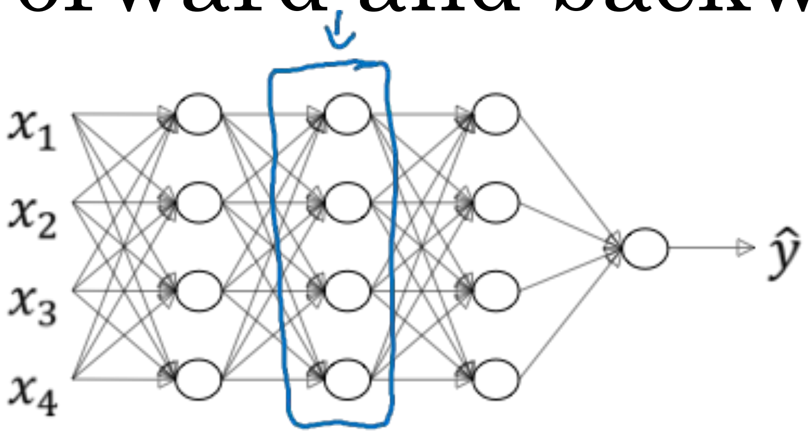
deeplearning.ai

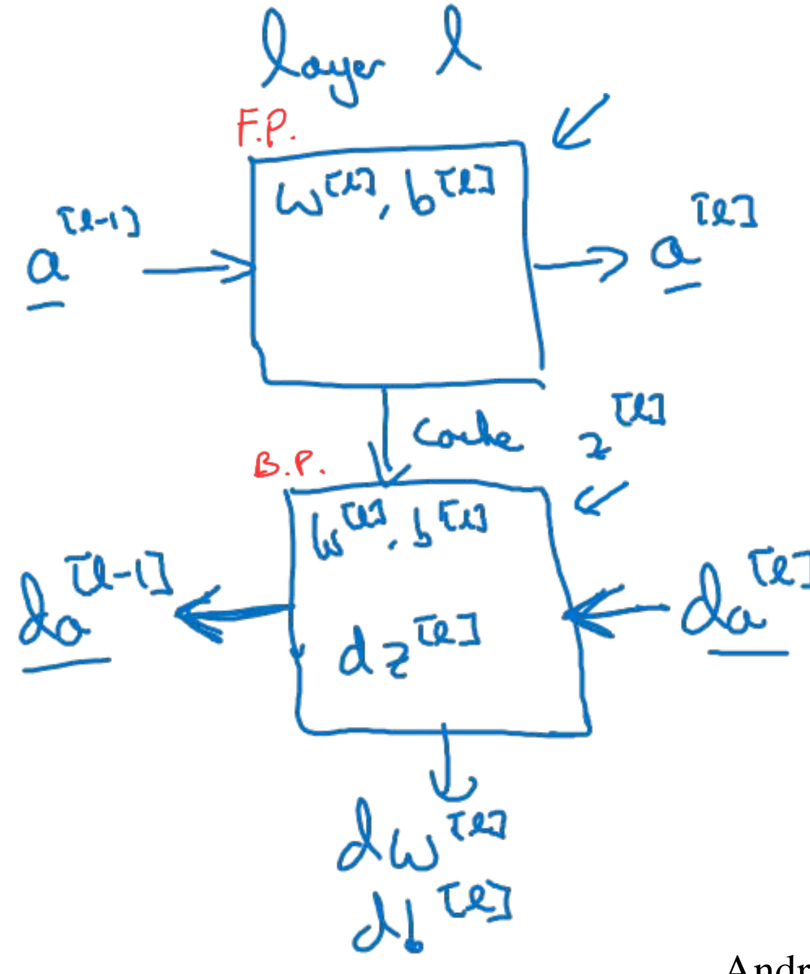
Deep Neural Networks

Building blocks of deep neural networks

→ For efficient computation of the BP step for a certain layer l, it is useful to cache $z^{[l]}$ during the FP step for that layer.

Forward and backward functions





ignore, just watch next slide

1 Layer l: $W^{[l]}, b^{[l]}$ F.P.

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

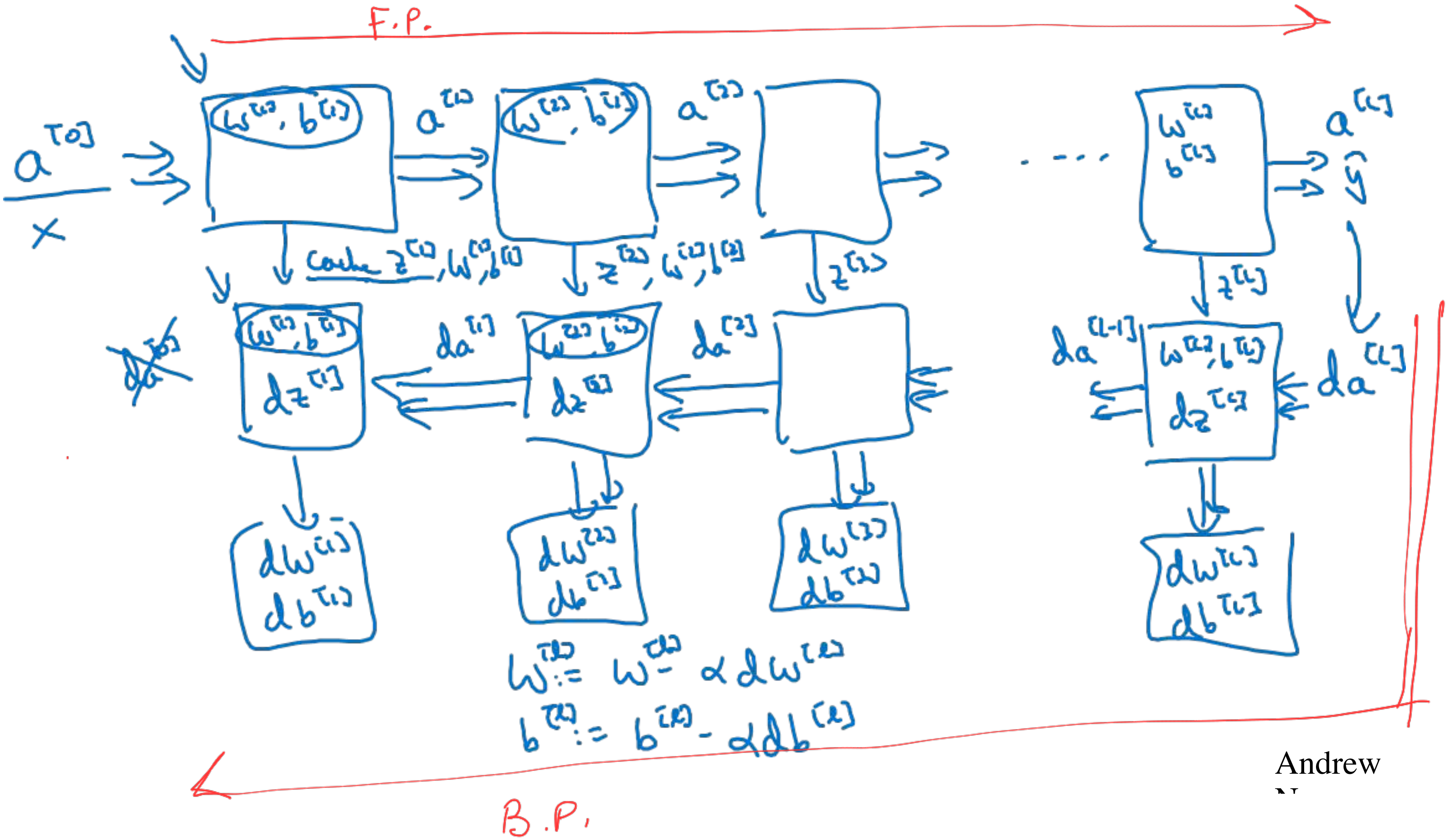
$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ 2 cache $z^{[l]}$

useful later for BP. of layer l

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

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

Forward and backward functions



Andrew Ng

deeplearning.ai

Deep Neural Networks

Forward and backward propagation

Backward propagation for layer l

→ Input $da^{[l]}$

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

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

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

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

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

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

B.P.

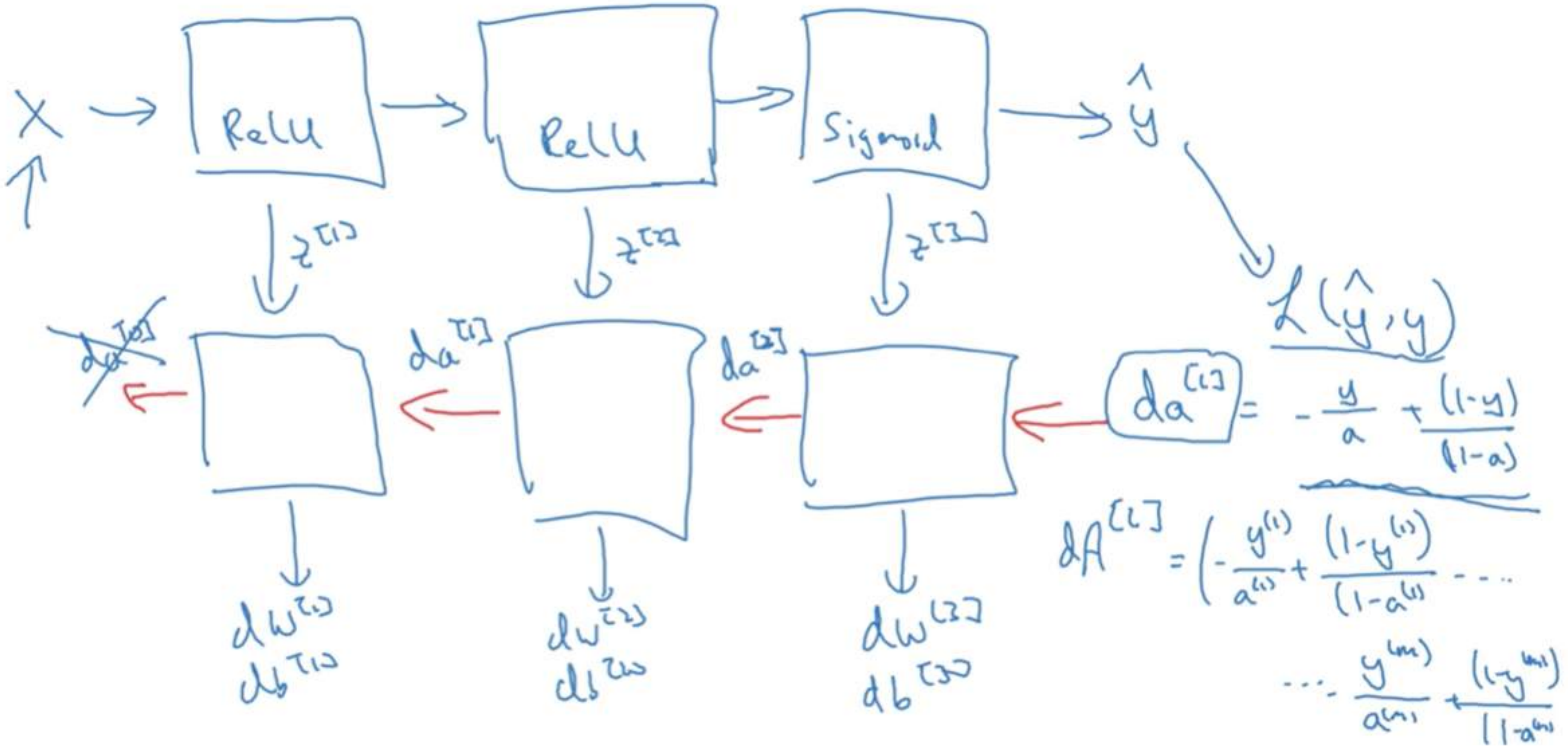
$$dZ^{[l]} = dA^{[l]} * g^{[l]'}(Z^{[l]})$$

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

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

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

Summary



X
ReLU
ReLU
Sigmoid
$\hat{y}$
$z^{[1]}$
$z^{[2]}$
$z^{[3]}$
$\mathcal{L}(\hat{y}, y)$
$\cancel{da^{[0]}}$
$da^{[1]}$
$da^{[2]}$
$da^{[l]} = -\frac{y}{a} + \frac{(1-y)}{(1-a)}$
$dA^{[l]} = \left(-\frac{y^{(1)}}{a^{(1)}} + \frac{(1-y^{(1)})}{(1-a^{(1)})} \cdots \right.$
$\cdots -\frac{y^{(m)}}{a^{(m)}} + \frac{(1-y^{(m)})}{(1-a^{(m)})}$
$dW^{[1]}$
$db^{[1]}$
$dW^{[2]}$
$db^{[2]}$
$dW^{[3]}$
$db^{[3]}$

deeplearning.ai

Deep Neural Networks

Parameters vs Hyperparameters

What are hyperparameters?

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

Hyperparameters: learning rate α

#iterations

#hidden layers L

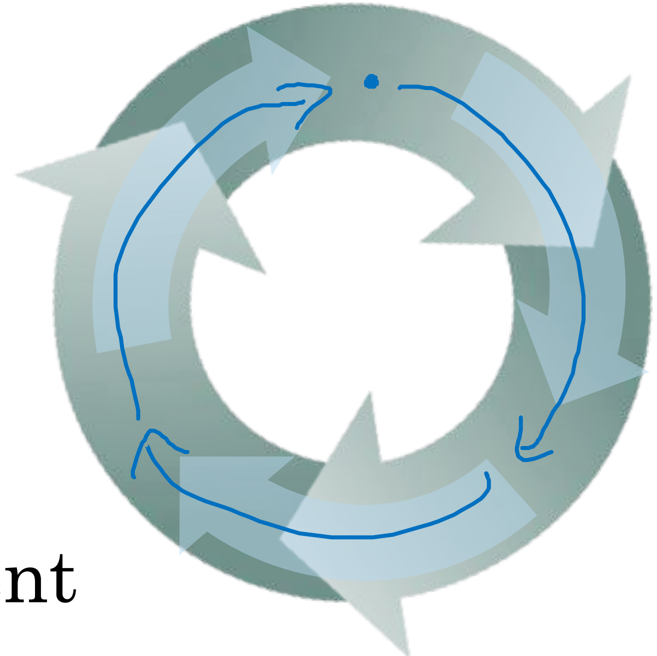
hidden units $n^{[1]}, n^{[2]}, \ldots$

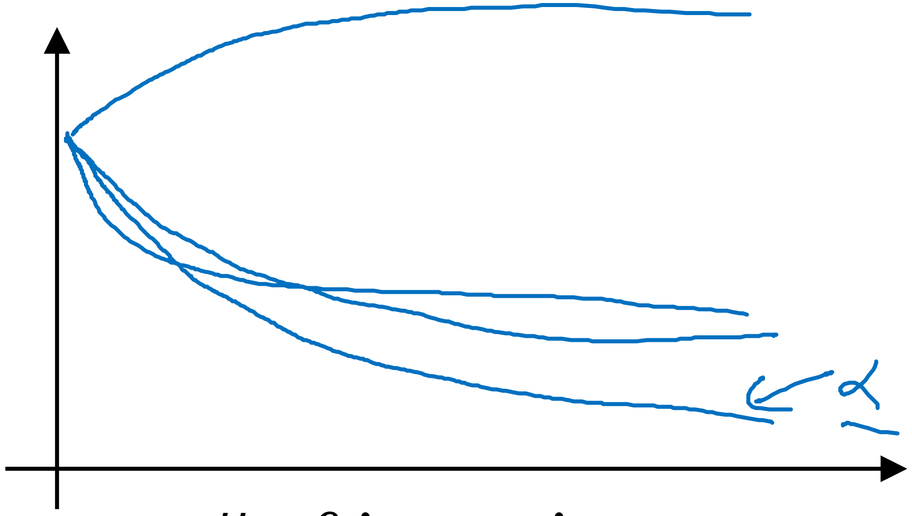
choice of activation function

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

Andrew Ng

Applied deep learning is a very empirical process





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

deeplearning.ai

Deep Neural Networks

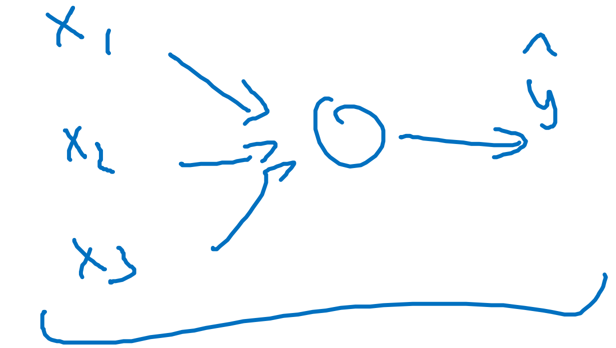
What does this have to do with the brain?

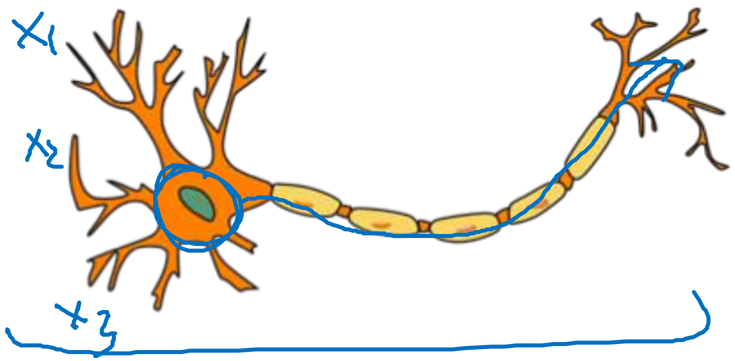
Forward and backward propagation

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}\left(Z^{[L]}\right) = \hat{Y}$$

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

"It's like the brain"





$X \longrightarrow y$

Andrew Ng