Deep L-layer Neural Network

Total layers = hidden layers + output layers

$L = 4$ (No. of layers)

$n^{(l)}$ = No. of units in layer l

$n^{(1)} = 5 \quad n^{(2)} = 5 \quad n^{(3)} = 3 \quad n^{(L)} = 1$

$n^{(0)} = n_x = 3$

$a^{(l)} = g^{(l)}(z^{(l)})$ → activation in layer l

$w^{(l)}$ = weights for $z^{(l)}$

Forward Propogation in Deep Net.

General equation:

$$z^{(l)} = w^{(l)} a^{(l-1)} + b^{(l)}$$

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

Vectorized:

$$Z^{(1)} = W^{(1)} A^{(0)} + b^{(1)}$$
$$A^{(1)} = g^{(1)}(Z^{(1)})$$

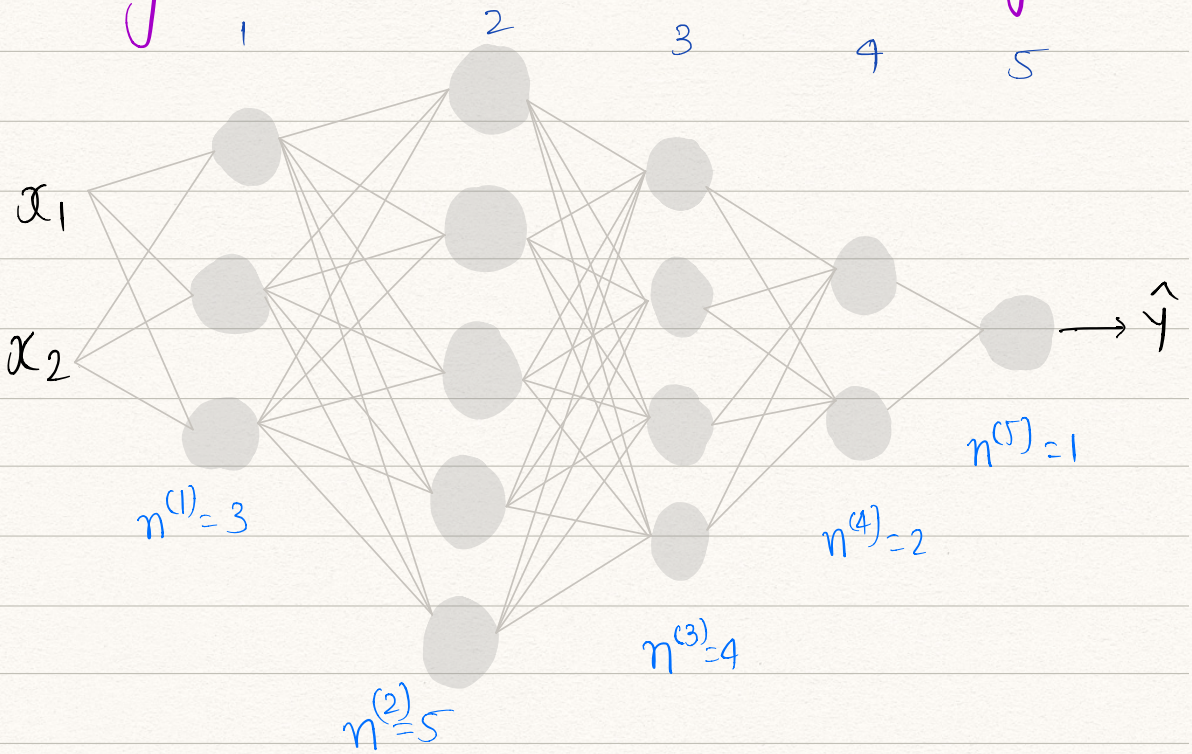
$$Z^{(2)} = W^{(2)} A^{(1)} + b^{(2)}$$
$$A^{(2)} = g^{(2)}(Z^{(2)})$$

$$\hat{y} = g^{(4)}(Z^{(4)}) = A^{(4)}$$

for loop

$0 \rightarrow l$

Getting Your Matrix dimension right:



$$z^{(1)} = W^{(1)} \cdot x + b^{(1)}$$

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

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

$$W^{(l)} = (n^{(l)}, n^{(l-1)})$$
$$b^{(l)} = (n^{(l)}, 1)$$

$W^{(1)} = (n^{(1)}, n^{(0)}) = (3,2)$ $\quad b^{(1)} = (3,1)$

$W^{(2)} = (5,3)$ $\quad b^{(2)} = (5,1)$

$W^{(3)} = (4,5)$ $\quad b^{(3)} = (4,1)$

$W^{(4)} = (2,4)$ $\quad b^{(4)} = (2,1)$

$W^{(5)} = (1,2)$ $\quad b^{(5)} = (1,1)$

$dw^{(l)} = (n^{(l)}, n^{(l-1)})$, $db^{(l)} = (n^{(l)}, 1)$

Vectorized Implimentation:

$$Z^{(1)} = W^{(1)} \cdot X + b^{(1)}$$

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

$$Z^{(1)}, A^{(1)} = (n^{(1)}, m) = dZ^{(1)}, dA^{(1)}$$

Why deep Representation?

→ As we proceed deep into layers, they get more complex.

→ Initial hidden layers compute simple functions & as they move deeper they combine previous results & get more complex.

e.g. Audio → low levels audio waveform —compose→ phonems —compose→ words —compose→ Sentences

Circuit Theory & Deep learning:

Informally:

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

Building Blocks of deep Neural Nets:

Forward & Backward functions:

layer l : $W^{(l)}, b^{(l)}$

Forward :

input : $a^{(l-1)}$ o/p : $a^{(l)}$

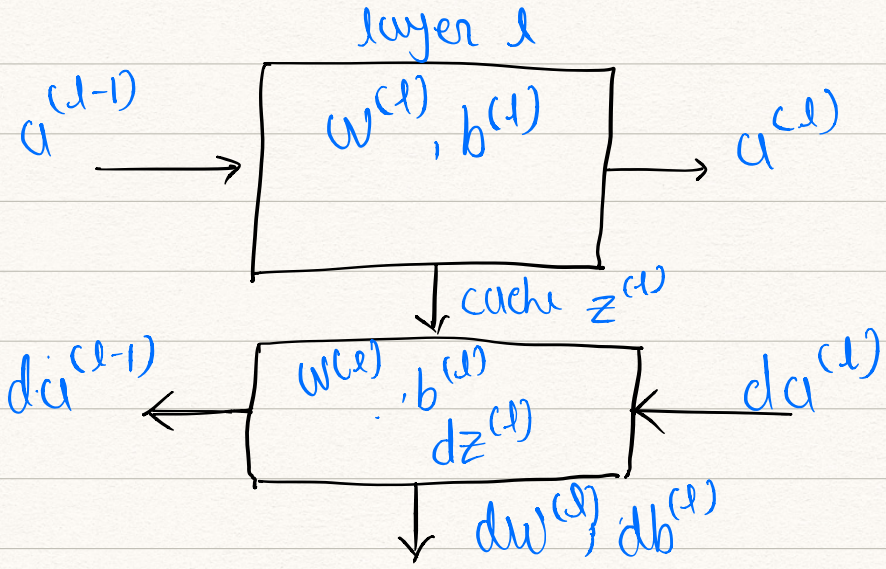
$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$ cache : $z^{(l)}$

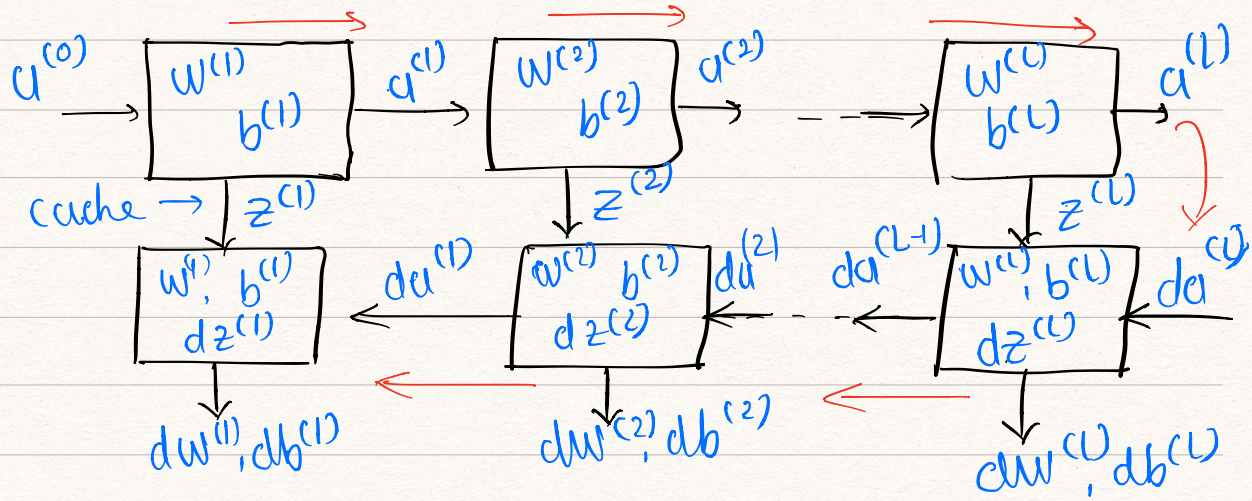
$a^{(l)} = g^{(l)}(z^{(l)})$

Backward:

i/p : $da^{(l)}$ o/p : $da^{(l-1)}, dW^{(l)}, db^{(l)}$

cache : $z^{(l)}$





Backward Propagation for layer l:

Input : $da^{(l)}$

Output : $da^{(l-1)}, dW^{(l)}, db^{(l)}$

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

$$dW^{(l)} = dz^{(l)} \cdot a^{(l-1)}$$
$$db^{(l)} = dz^{(l)}$$

$$da^{(l-1)} = W^{(l)T} \cdot dz^{(l)}$$

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

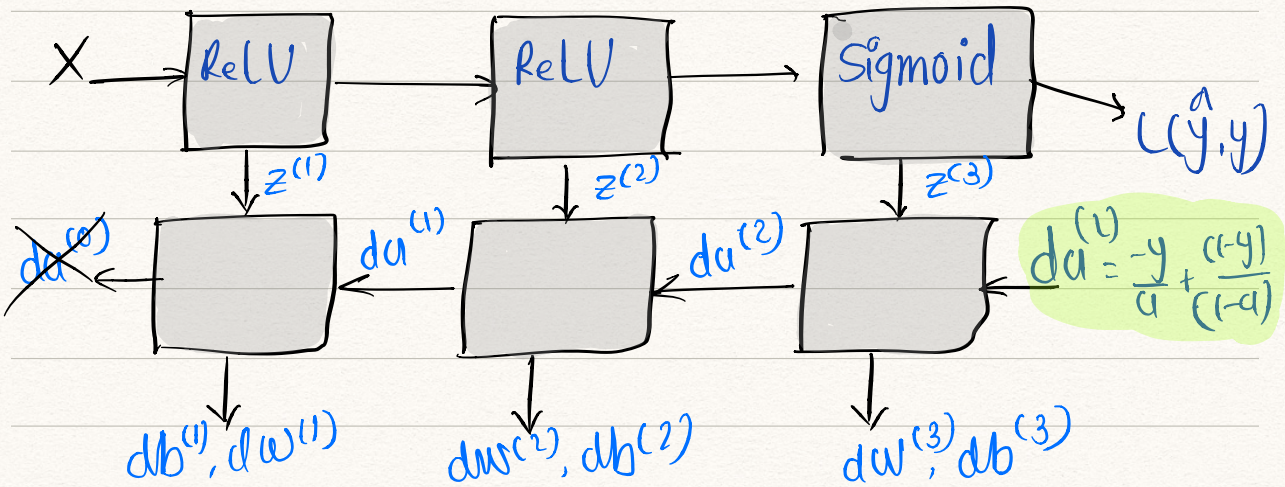
Vectorized:

$$dz^{(l)} = dA^{(l)} * g^{(l)'}(z^{(l)})$$

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

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

$$dA^{(l-1)} = W^{(l)T} \cdot dz^{(l)}$$



Parameters Vs. Hyperparameters:

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots$

Hyperparameters: Learning Rate (α)

\# of iterations of gradient decent to carry out.

\# of hidden layer

\# of hidden units

Activation function choice

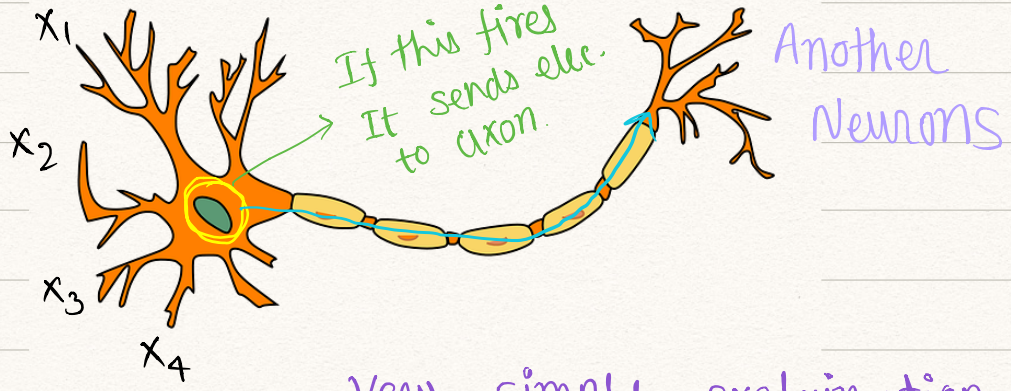
Other: Momentum, mini batch size, Regularization

Applied deep learning is a very empirical process.

idea → code → experiment

What does deep learning have to do with brain :

→ Not a lot...



Very simple explaination.

Quiz

Q:1 ✗ cache intermediate value forward to backward

Q:2 all but w^l, b^l, a^l

Q:3 Deeper layer compute more Complex

Q:4 False

Q:5 len (layer_dims)

W: (i, i-1)

Q:6 Layer : 4

hidden : 3

Q:7 False ✗ True

Q:8 True

Q:9 $W_1 = (4 \times 4)$ $W^2 = (3,4)$ $W^3 = (1,3)$

$b^1 = (4,1)$ $b^2 = (3,1)$ $b^3 = (1,1)$

Q:10 (n^l, n^{l-1})