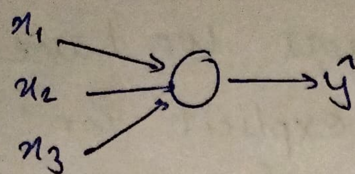
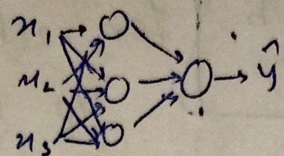
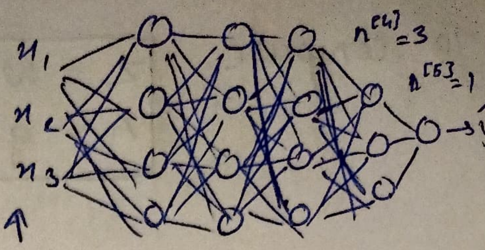
★ Deep Neural Networks.



Logistic Regression.
(Shallow NN)



"2 layer NN"
1 hidden layer



layer 0

"deep" NN
4 hidden layers.
(5-layer NN)
$\parallel$
L

'L' → no. of layers

$n^{[l]}$ = # no. of nodes/units in layer l.

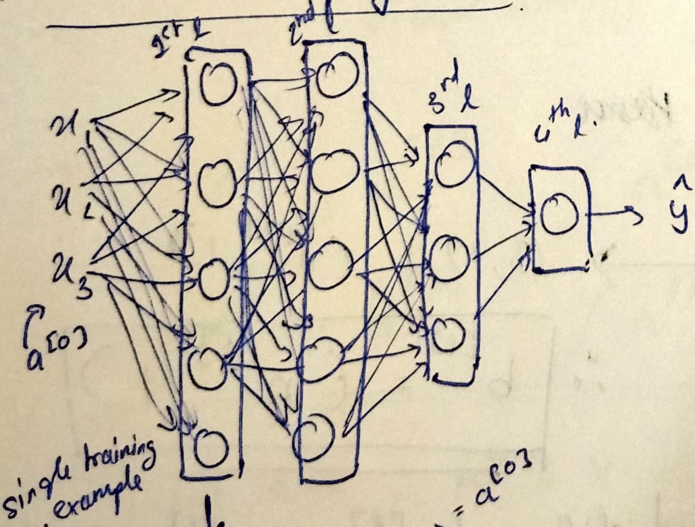
$a^{[l]}$ = activations in layer l.

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

$a^{[0]} = x$ (input layer = 0)

$a^{[L]} = \hat{y}$ (output layer = L)

★ Forward Propagation in a deep network.



$x \Rightarrow \begin{cases} z^{[1]} = W^{[1]} x + b^{[1]} \quad (x = a^{[0]}) \\ a^{[1]} = g^{[1]}(z^{[1]}) \end{cases}$

$\begin{cases} z^{[2]} = W^{[2]} a^{[1]} + b^{[2]} \\ a^{[2]} = g^{[2]}(z^{[2]}) \end{cases}$

vectorized:

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

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

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

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

$\left[z^{[2](1)} \; z^{[2](2)} \; \dots \; z^{[2](m)} \right]$ → for m training examples

$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}$

$a^{[4]} = g^{[4]}(z^{[4]})$

$$\boxed{\text{general: } z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}, \quad a^{[l]} = g^{[l]}(z^{[l]})}$$

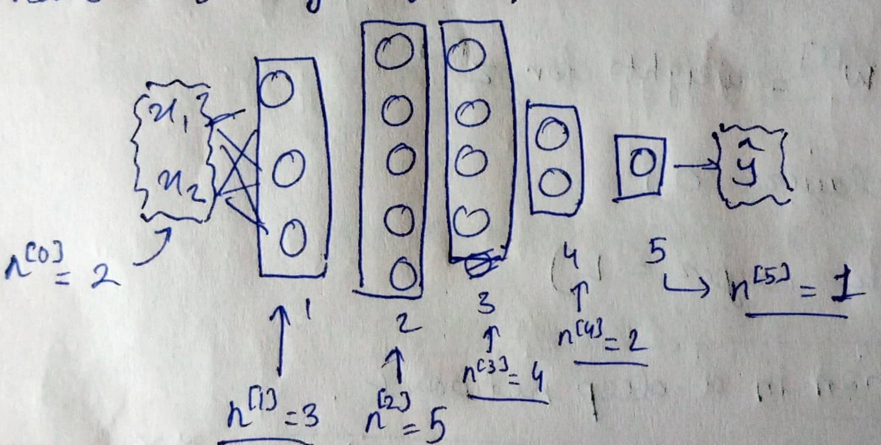
Here, $\boxed{\hat{Y} = g(Z^{[4]}) = A^{[4]}}$

To compute $\boxed{\begin{aligned} Z &= WX + b, \\ A &= g(Z) \end{aligned}}$ for all layers we have to use explicit 'for' loop. No other way to do it.

One of the bugs in your neural network could ~~be~~ occur because of matrix dimensions.

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

We have a 5-layer NN.



$$\boxed{\underset{(3,1)}{Z^{[1]}} = \underset{(3,2)}{W^{[1]}} \cdot \underset{(2,1)}{X} + b^{[1]}}$$

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

$$\underset{3\times 1}{\begin{bmatrix} \circ \\ \circ \\ \circ \end{bmatrix}} = \underset{3\times 2}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} \underset{2\times 1}{\begin{bmatrix} \circ \\ \circ \end{bmatrix}}$$

~~Hence~~

$(3,1) = (n^{[1]}, 1)$

$\therefore \boxed{b^{[l]} = (n^{[l]}, 1)}$

$\therefore \boxed{W^{[l]} = (n^{[l]}, n^{[l-1]})}$

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

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

$W^{[4]} = (2, 4)$

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

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

- $Z^{[l]}$, X, $A^{[l]}$.

for 1 training example:- $\underset{(n^{[1]},1)}{Z^{[1]}} = \underset{(n^{[1]},n^{[0]})}{W^{[1]}} \cdot \underset{(n^{[0]},1)}{X} + \underset{(n^{[1]},1)}{b^{[1]}}$

for m training examples:- $Z^{[1]} = \underbrace{\begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[2](2)} & \cdots & z^{[2](m)} \\ | & | & & | \end{bmatrix}}_{\text{m training examples}}$

$$\boxed{\underset{(n^{[1]},m)}{Z^{[1]}} = \underset{(n^{[1]},n^{[0]})}{W^{[1]}} \cdot \underset{(n^{[0]},m)}{X} + \underset{(n^{[1]},1)}{b}}$$

remains same

$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{\text{no. of examples}} \Big\}$ no. of features (x_1, x_2)

$\underbrace{W \cdot X}_{(n^{[1]},m)} + \underset{(n^{[1]},1)}{b}$

↓

by python broadcasting → $(n^{[1]},m)$

$\therefore$ $Z^{[l]}, a^{[l]} = (n^{[l]},1)$ → for 1 training example

$Z^{[l]}, A^{[l]} = (n^{[l]},m)$ → for m training examples.

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

in back propagation $dZ^{[l]}, dA^{[l]} = (n^{[l]},m)$

WHY Deep Representation?

What does a deep representation do?

eg:- In case of audio recognition,

audio clip (Input) → 1st layer → 2nd layer → 3rd layer → 4th layer

1st layer ↓ might detect/classify low pitch / High pitch or low tone / high tone etc. (simple function).

2nd layer ↓ might try to identify and group 'phonic' sounds eg:- 'C' 'A' 'T' (slightly more complicated) →

3rd layer ↓ might form words

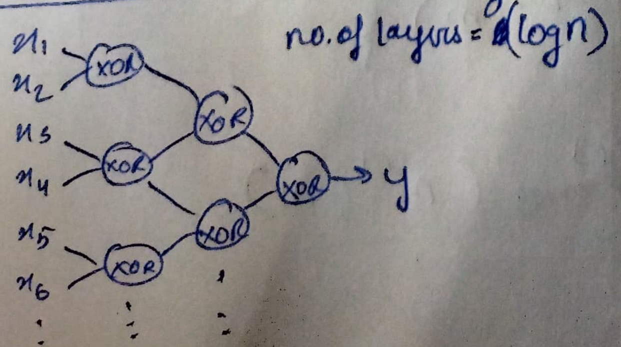
4th layer ↓ Sentences or phrases.

Circuit theory:

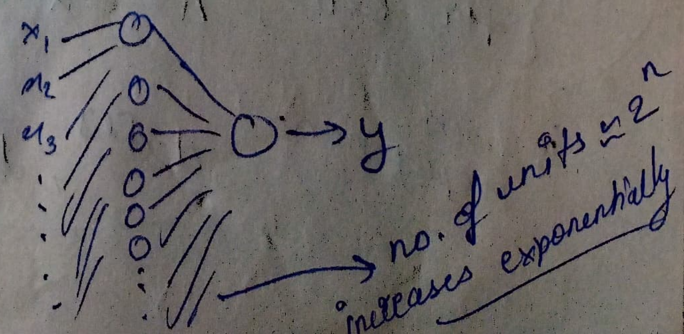
There are some functions which we can compute with a "small" L-layer deep neural network, that shallower and "bigger" networks require exponentially more hidden units to compute.

eg:- $y = x_1 \text{ XOR } x_2 \text{ XOR } x_3 \dots \text{ XOR } x_n$.

small but deep NN



big but shallow NN



* forward and backward functions.

for a layer l: $W^{[l]}, b^{[l]}$

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

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

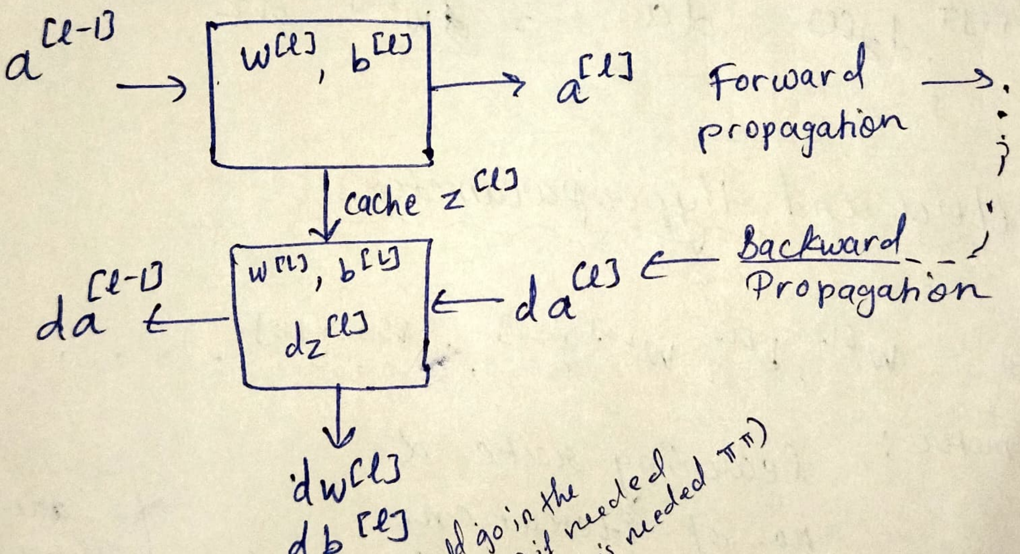
$$a^{[l]} = g^{[l]}(z^{[l]})$$

→ **Backward**: $\boxed{\text{input}}\ da^{[l]}$, $\boxed{\text{output}}\ da^{[l-1]}$

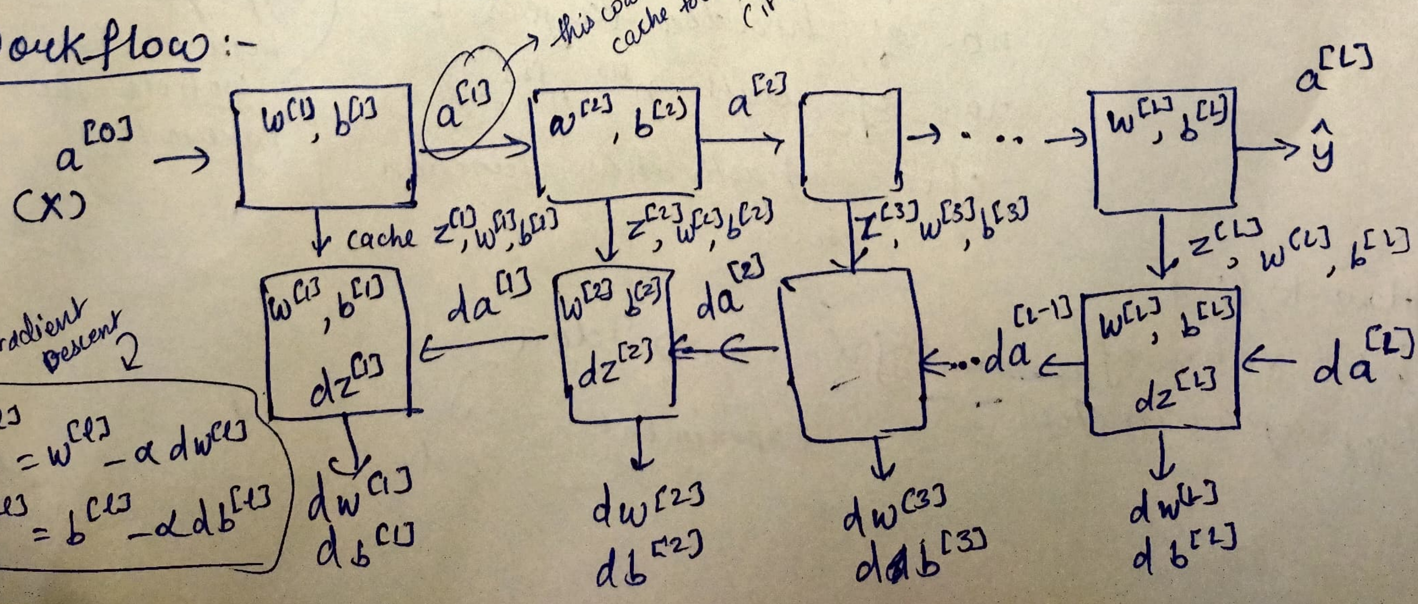
cache $(z^{[l]})$ $\qquad dW^{[l]}$

$db^{[l]}$

layer l.



Workflow:-



Gradient Descent

$$W^{[l]} = W^{[l]} - \alpha\, dW^{[l]}$$

$$b^{[l]} = b^{[l]} - \alpha\, db^{[l]}$$

Forward Propagation :- Input $a^{[l-1]}$ $\quad w^{[l]}, b^{[l]}$ ~~$b^{[l+1]}$~~

Output $a^{[l]}$, cache $(z^{[l]})$

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

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

Backward Propagation :- Input $da^{[l]}$

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

vectorized

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

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

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

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

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

$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$ $\rightarrow$ obtained from cache.

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

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

---

* Parameters and Hyperparameters.

Parameters :- $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \ldots$

Hyperparameters :-

$$\left.\begin{array}{l} \text{learning rate } \alpha \\ \text{no. of iterations} \\ \text{no. of hidden layers } L \\ \text{no. of units } n^{[1]}, n^{[2]}, \ldots \\ \text{choice of activation function} \end{array}\right\}$$ are called hyperparameters because they control the parameters

etc.

How to find best value of hyperparameter?

eg] α. Idea $\rightarrow$ code $\rightarrow$ experiment $\rightarrow$ Idea