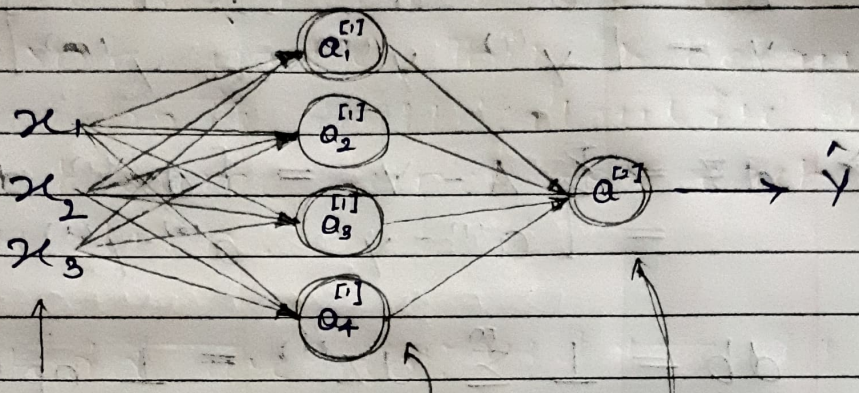
* ~~Shallow~~

## Shallow Neural Network




- Notation :

$[L]$ → layer no.

$a_i$ → node in layer

* ## Neural Network Representation

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]} = a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]} = a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]} = a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_3^{[1]} = a_3^{[1]} = \sigma(z_4^{[1]})$$

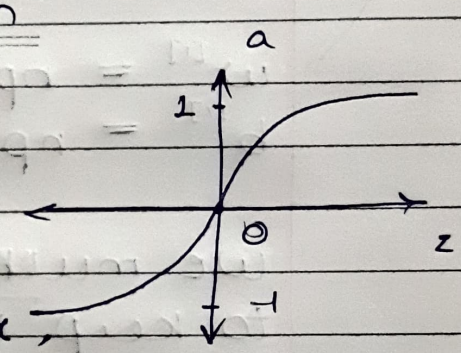
This is represented in vectorized form →

$$z^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

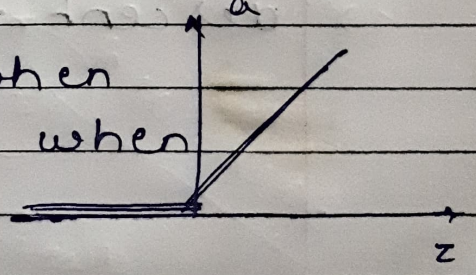
## * Activation Function

- $\tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$




tanh > sigmoid,
it converges faster,
mean is closer to zero
But sigmoid is pref in o/p layer

- Relu = max(0, z)
Derivative is 1 when z is positive & 0 when it is negetive.




$$\begin{cases} 0 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{cases}$$

## * Random Initialization

Initializing parameters $(w, b)$ in neural networks to 0 doesn't work. This is because every node will calculate the exact same funct for each iteration.

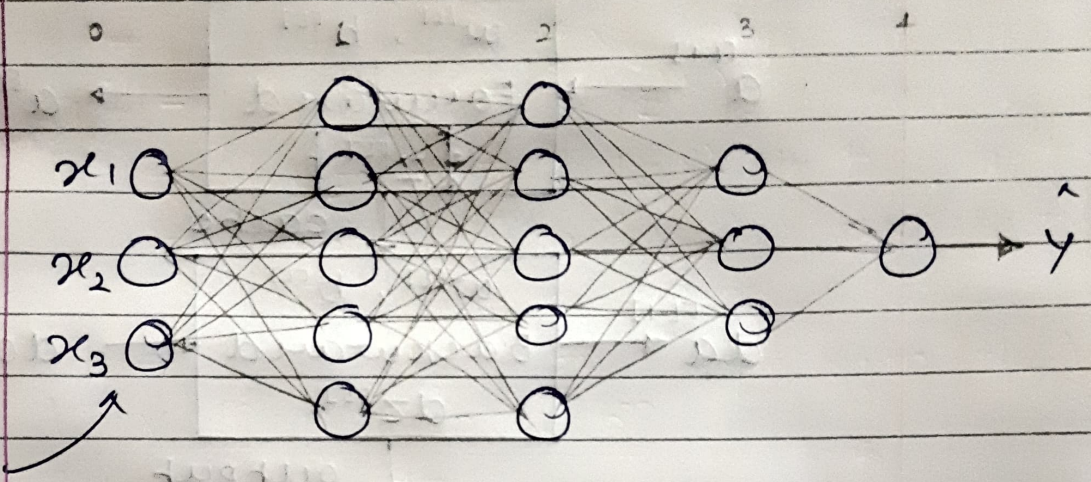
Instead we do,

$w^{[1]} = np.random.randn((2,2)) * 0.0$
$b^{[1]} = np.zeros((2,1))$

we multiply smthing like this to keep the wts small as large wts will result in larger value in the activ. $f^N$ which causes the gradient descent to become slower

# Deep Neural Network

## * Forward Propogation




For a single example $x$ :

- $z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$
  $a^{[1]} = g^{[1]}(z^{[1]})$

$\therefore z^{[l]} = w^{[l]} a^{[l-1]} + b^{[l]}$
$a^{[l]} = g^{[l]}(z^{[l]})$

$\hat{y} = g(z^{[4]}) = A^{[4]}$

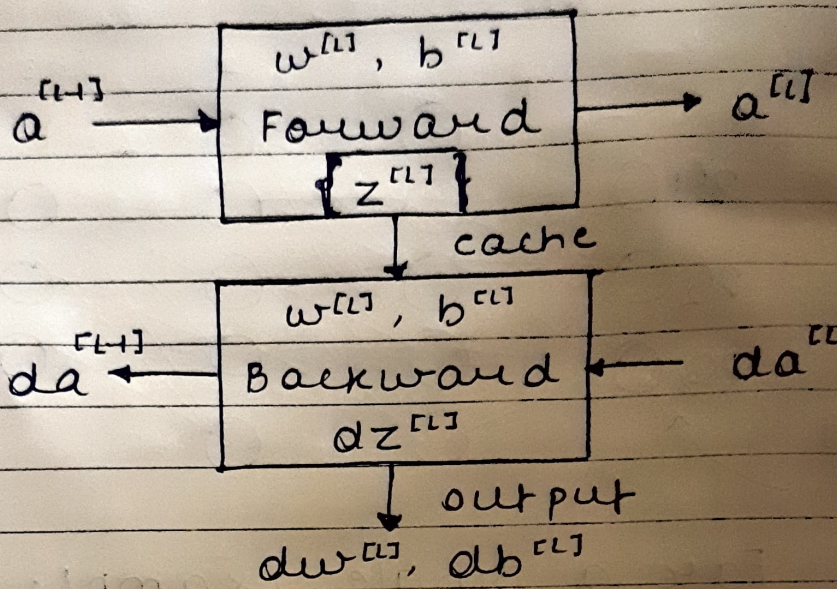
- Dimensions

$w^{[l]} = (n^{[l]}, n^{[l-1]})$ $\quad z^{[l]} = (a, c)$ $\quad w\ (a,b)\ (b,c)$

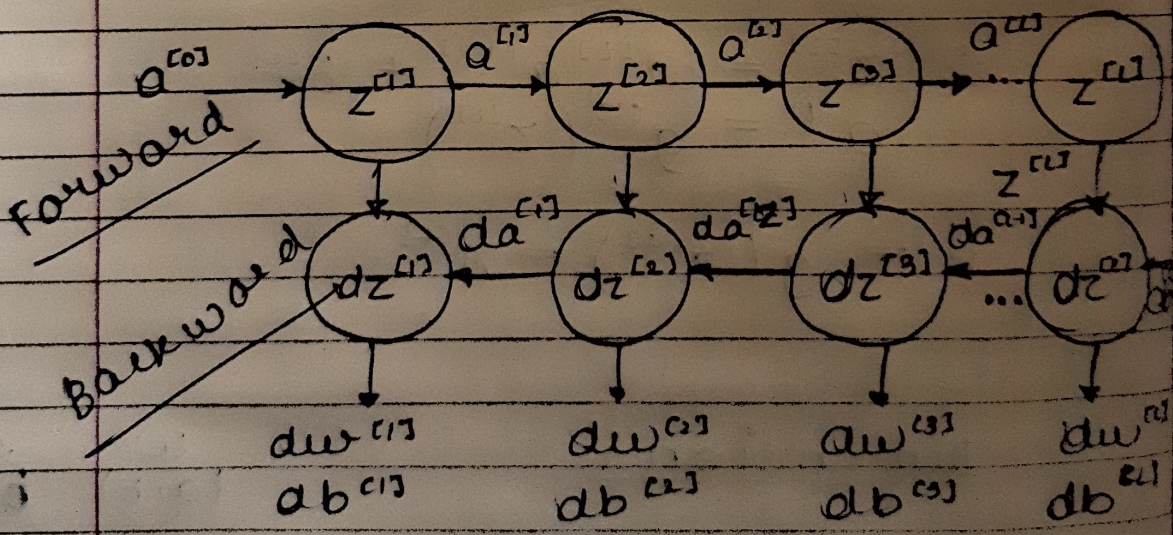
$b^{[l]} = (n^{[l]}, 1)$

# * Forward and Backward Functions

## single layer




## Multiple layers



* Params and Hyperparams

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

Hyperparameters:

$\alpha$, iterations, layers, neurons, activation ~~layers~~ functions, etc