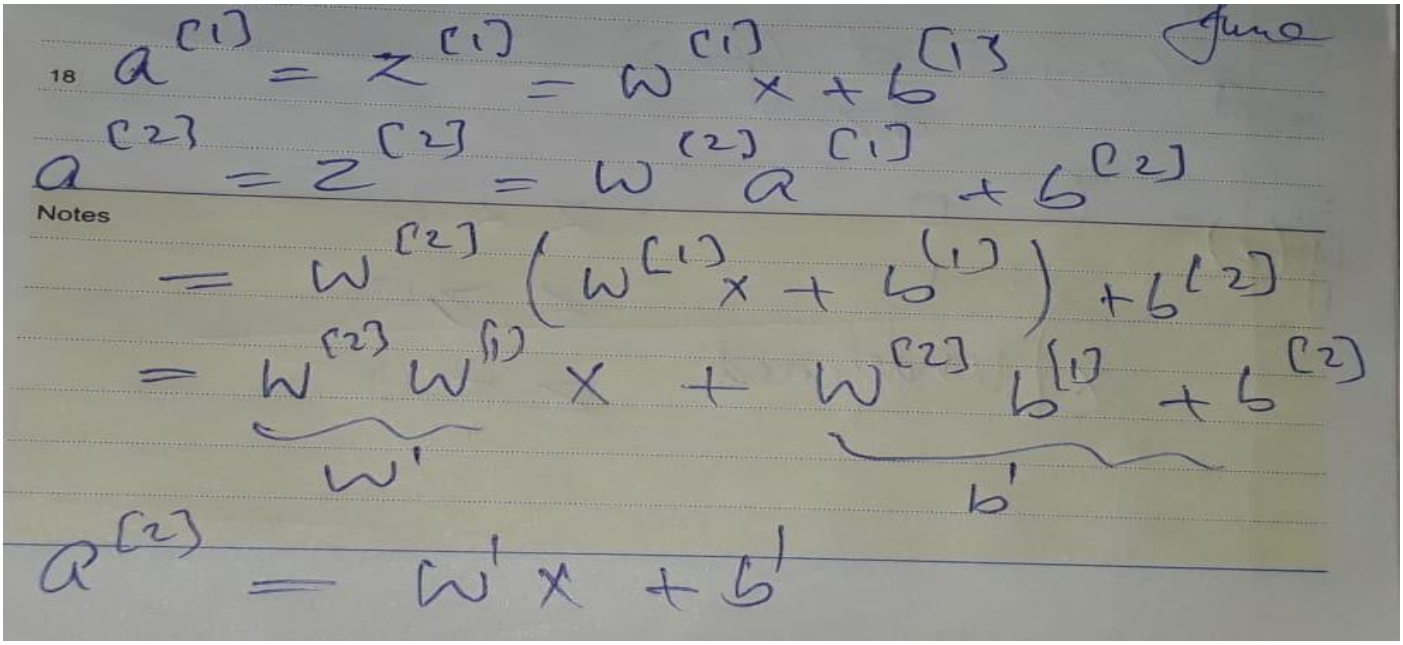
AUTHOR: SYEDA DARAQSHAN

WEEK 3

NEURAL NETWORKS AND DEEP LEARNING

- Why Neural network needs non-linear activation function?

The purpose of the **activation function** is to introduce **non-linearity** into the output of a **neuron**. ... **Neuron** cannot learn with just a **linear function** attached to it, it **requires** a **non-linear activation function** to learn as per the difference w.r.t error.



- Neural Network is an outputting a linear function.
- If the input is linear the output is also linear.
- LINEAR + LINEAR = LINEAR
- Linear activation function to be used in Output Layer .

- It is advised to not use Linear Activation Function for Hidden Layers!

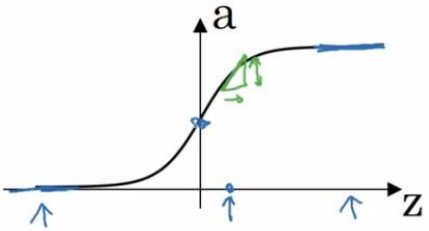
FORWARD AND BACKWARD PROPAGATION

Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]} a^{[1]T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$
$dW^{[1]} = dz^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

- G(z) is a sigmoid function, then the slope of the function is

Sigmoid activation function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$\frac{d}{dz} g(z)$ = slope of $g(z)$ at z

$= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right)$

$= g(z)\left(1 - g(z)\right)$ ←

$z = 10$. $g(z) \approx 1$

$\frac{d}{dz} g(z) \approx 1\,(1-1) \approx 0$

$z = -10$ $g(z) \approx 0$

$\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$

$z = 0$ $g(z) = \frac{1}{2}$

→ $g(z) = \max(0, z)$

$g'(z) = \frac{d}{dz}(g(z))$

Notes

$$g'(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \\ \text{undefined} & z = 0 \end{cases}$$

STEPS TO BIULD A NEURAL NETWORK

To build a Neural Network :-

1) Define Neural Network structure (No. of input, hidden units)
2) Initialize model param.
3) Loop
   → Implement forward propagation
   → Compute Loss
   → Implement backward propagation to get gradients
   → Update param.

Notes

☆ build helper func to compute 1–3. and merge them into 1 func = nn_model, after