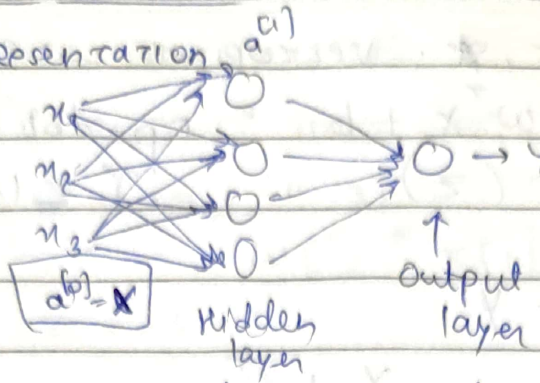
19/7/24

Page No.
Date

III) NEURAL NETWORKS

1. REPRESENTATION



$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix}$$

Activation $= \sigma(z)$ $\quad z = w^T x + b$.

$[l] \rightarrow$ layer

$a_i \leftarrow$ node in layer

$(4,1) \quad (4,3) \quad (3,1) \quad (4,1)$

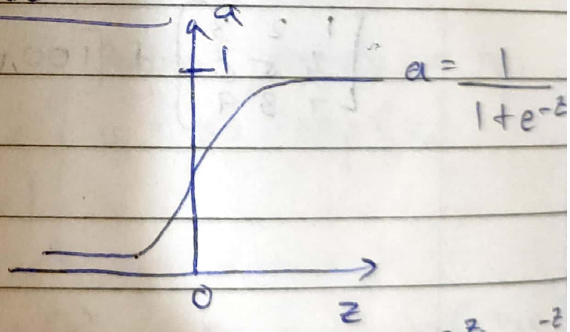
$$z^{[1]} = w^{[1]T} x + b^{[1]} \qquad a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = w^{[2]T} a^{[1]} + b^{[2]} \qquad a^{[2]} = \sigma(z^{[2]})$$

2. ACTIVATION ~~FUNCTIONS~~ FUNCTIONS

a. SIGMOID FUNCTION

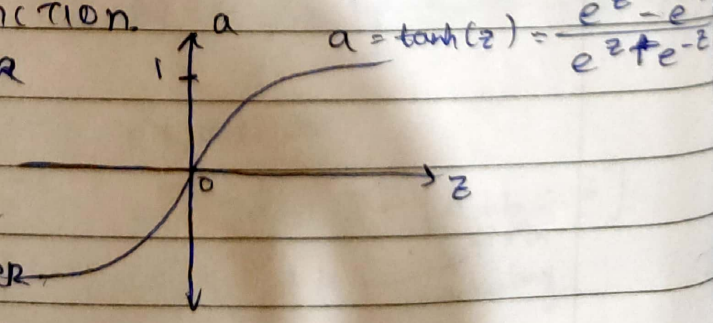
[ONLY FOR BINARY CLASSIFICATION]



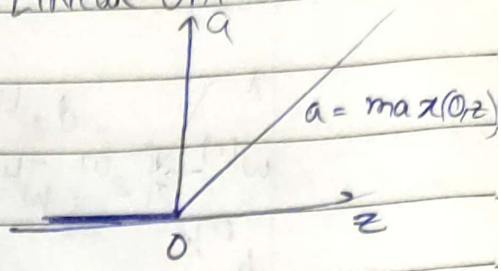
b. tanh function.

(always BETTER THAN SIGMOID)

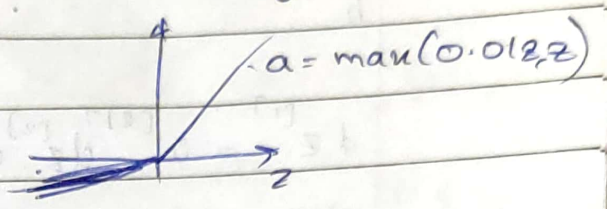
[ONLY PROBLEMATIC WHEN OUTPUT LAYER SHOULD NOT GIVE -ve VALUES]



c. ReLU Function : Rectified Linear Unit
[Almost always used by default]



⇒ Leaky ReLU



(*) we do not use linear activation functions

Why? It is useless.

If the hidden layers use a linear actvn fn, then it is as good as not using any hidden layer. Since $g(z) = z$, linear actvn fn yields the same output everytime.

The composition of 2 linear fns is a linear fn.

- If needed (say a real number output b/w $-\infty$ & ∞) only then can we use a linear actvn but only in the output layer.

20/7/24

3. Derivatives of Activation.

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

$g'(z) = g(z)(1-g(z))$

b. tanh $g(z) = \tanh$

$\therefore g'(z) = 1 - (\tanh(z))^2$

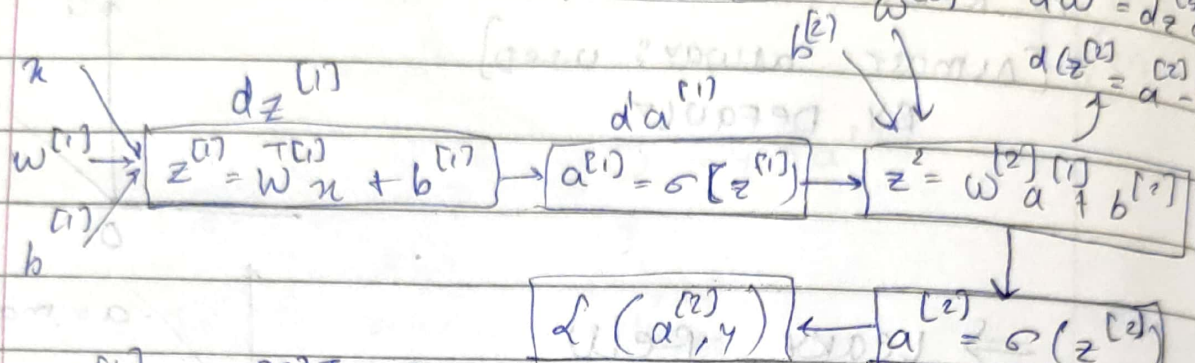
c. ReLU $g(z) = \max(0, z)$

$\therefore$ when z = +ve, $g'(z) = 1$

z = -ve $g'(z) = 0$.

Page No.
Date

4. NEURAL NETWORK GRADIENTS



$$dz^{[1]} = w^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$$

↑ element-wise product

$$dw^{[1]} = dz^{[1]} x^T \qquad db^{[1]} = dz^{[1]}$$

⇒ WHAT HAPPENS IF YOU INITIALIZE ALL WEIGHTS TO ZERO? $w^{[1]} = np.random.randn((2,2)) * 0.01$

ALL THE HIDDEN UNITS ARE SYMMETRIC & EVEN IF WE CALCULATE GRADIENT DESCENT MANY TIMES, IT WILL STILL COMPUTE THE SAME FUNCTION.

BUT BIAS $b^{[1]} = np.zeros((2,1))$ CAN BE ZERO.