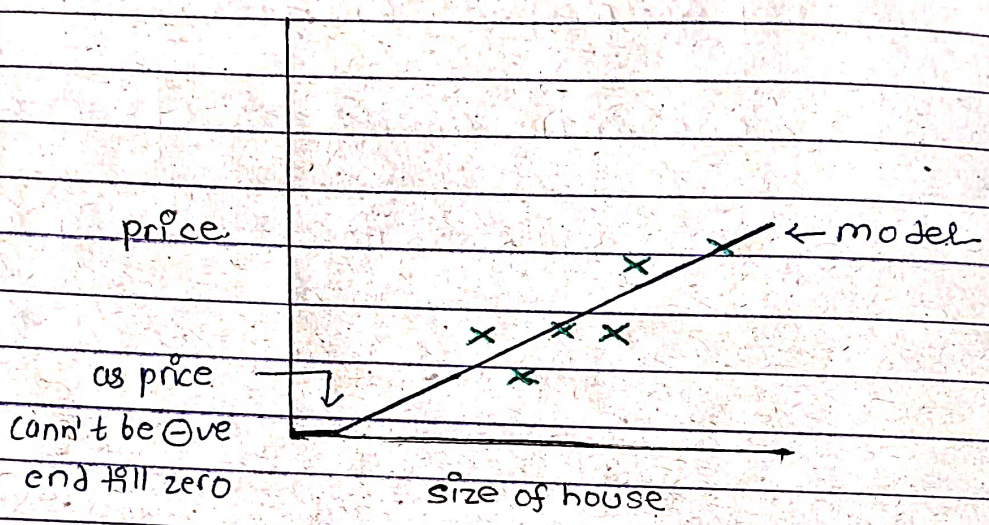
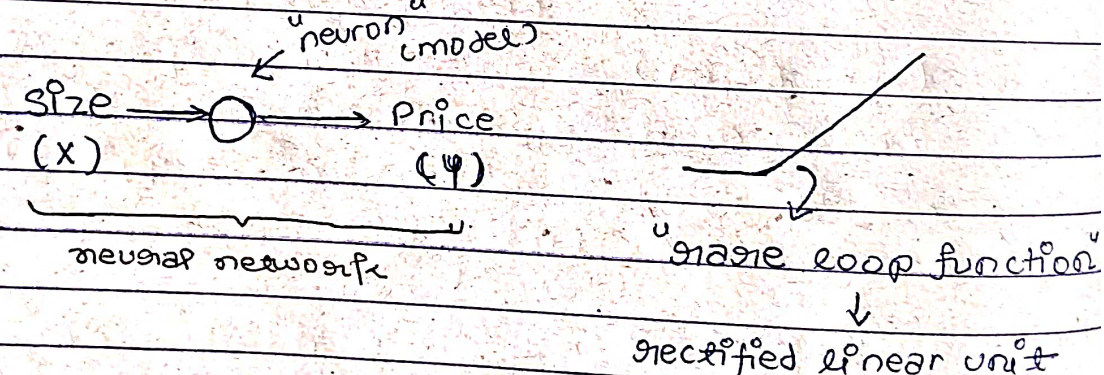
Artificial Neural Network

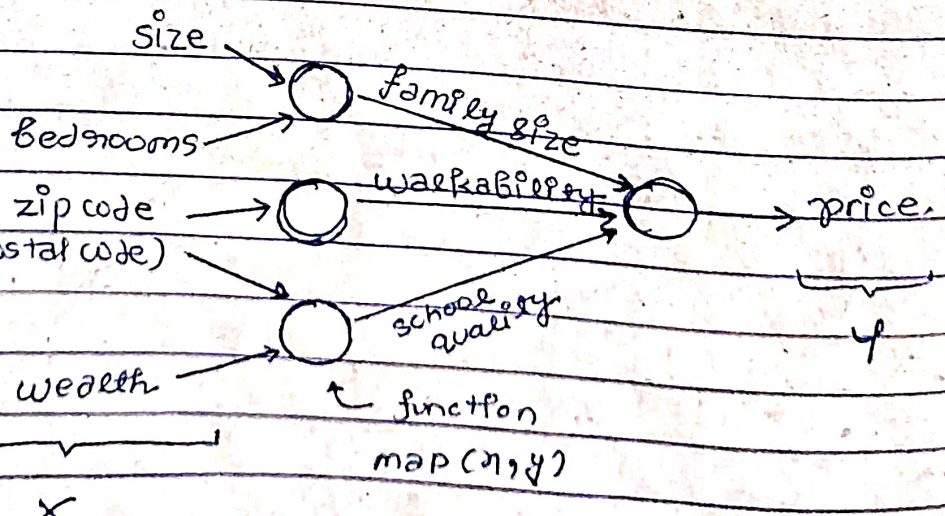
What is a neural Network?

i.e. house-price prediction



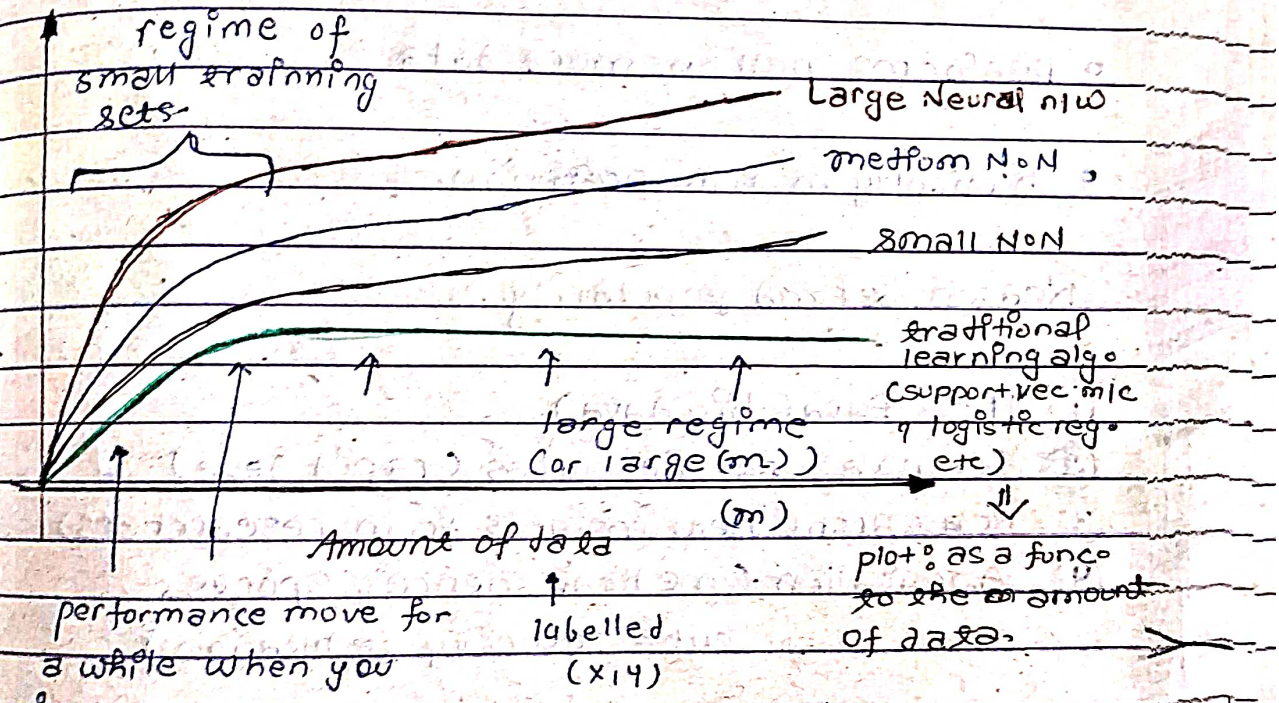


i.e. multiple-input features.



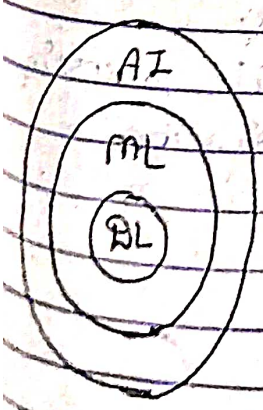
Neural networks are computational models that mimics the complex functions of the human brain.

The neural network consists of interconnected nodes or neurons that processes and learns from data enabling tasks such as pattern recognition and decision making in machine learning.



Scale drives deep learning progress.

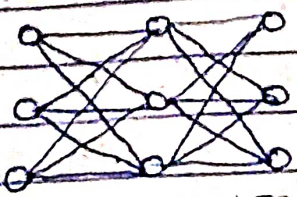
- m : size of training set in training example



ML →

i/p → feature extraction → classification → o/p

DL →



i/p → → o/p

feature extra +

- It's a particular kind of ML that achieves great power & flexibility by ML to represent

- requires more computation power.
- Automatic feature extraction
- performs well on more data
- difficulty in interpretation.

Reason behind popularity

1. Huge Data (big data)
2. Computational Resources (rapid dev.)
3. Research (new insights from researchers)
4. Application (medical science, sports, cybersecurity, education)

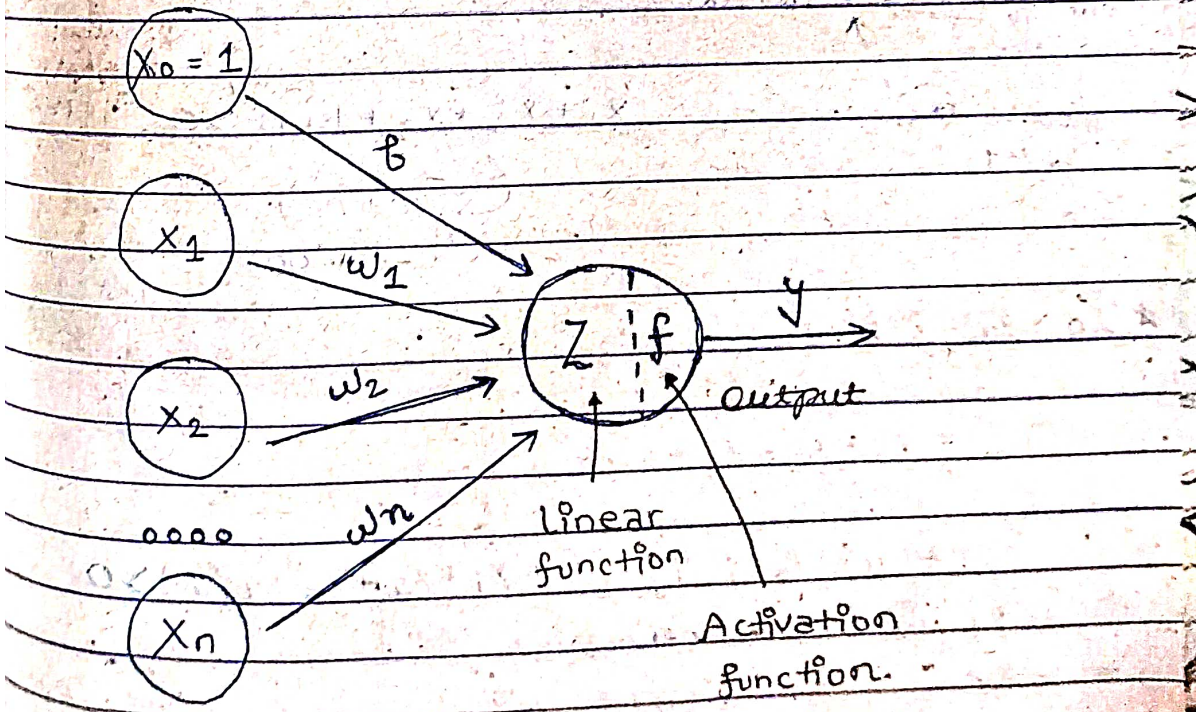
CPU vs GPU

CPU	GPU
• few complex cores	• hundreds of simpler cores
• focuses on doing one task as possible	• possible of doing various tasks parallely.
• used for general purpose tasks	• used for graphic processing or matrix multiplication.

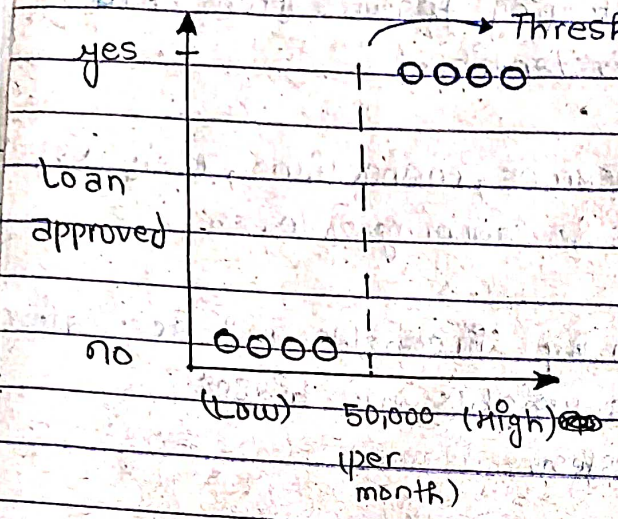
Keras
Tensorflow } DL frameworks
Pytorch
Caffe

What are Neural Networks?

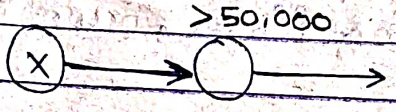
- Neural Networks extract identifying features from data, lacking pre-programmed understanding.
- Network components includes neurons, connections, weights, biases, propagation functions & learning rule.
- Neurons receive inputs govern by thresholds and activation functions. Connections involves weights and biases regulating information transfer.
- Learning, adjusting weights & biases, occurs in three stages —
  input computation, output generation and iterative refinement enhancing networks proficiency in diverse tasks.

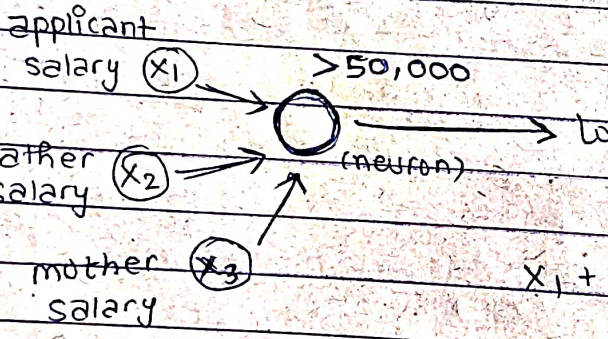


★ Intuition behind perceptron



- salary as i/p
- checks if it is 50'000 or not (threshold)
- get o/p → yes or no





$x_1 + x_2 + x_3 + bias > 0$

↳ learned (-50,000)

★ Intuition behind activation func°

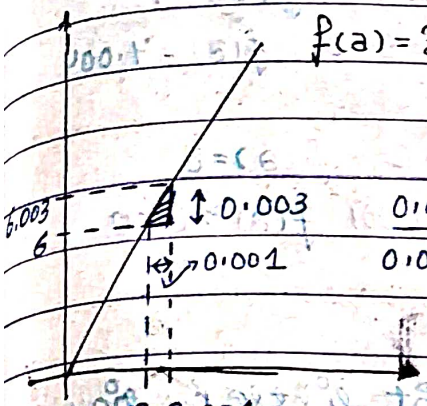
$z = x_1 + x_2 + x_3 + bias$

o/p → will be '1' if $(z = x_1 + x_2 + x_3 + bias) > 0$
it will be '0' otherwise

$$o/p \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

o/p → step fn°(z) or o/p → step(z)

Derivatives



$a = 2 \quad f(a) = 6$

$a = 2.001 \quad f(a) = 6.003$

$\frac{0.03}{0.01}$ (height / width)

slope (derivative) of $f(a)$ at $a = 2$ is 3.

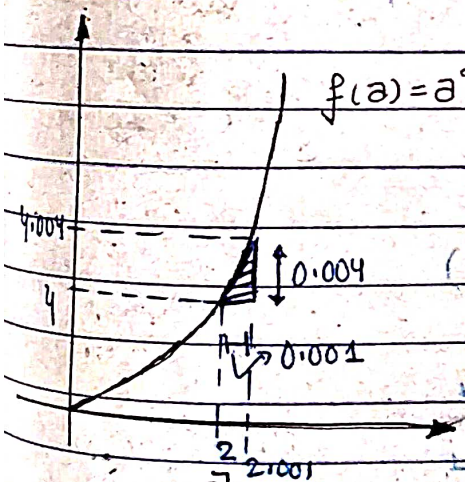
$a = 5 \quad f(a) = 15$

$a = 5.001 \quad f(a) = 15.003$

Slope at $(a = 5)$ is also 3

$$\frac{d f(a)}{d a} = 3 = \frac{d}{da} f(a)$$ if I nudge 'a' to the right a little-bit

I expect $f(a)$ to go up by three times as much as nudge the value of 'a'.



$a = 2. \quad f(a) = 4$

$a = 2.001 \quad f(a) \approx \boxed{4.004}$

(≈ 4.004001)

slope (derivation) of $f(a)$ at $a = 2$ is 4.

$\frac{d}{da} f(a) = 4$ when $a = 2$

$a = 5 \quad f(a) = 25$

$a = 5.001 \quad f(a) = 25.010$

$\frac{d}{da} f(a) = 10$ when $a = 5$

$\frac{d}{da} f(a) = \frac{d}{da} a^2 = 2a.$

* for any given value of 'a' you nudge up by 0.0001 infinitesimal small number, you would expect $f(a)$ to go up by "2a".

① $f(a) = a^2$ $\frac{d}{da} f(a) = 2a$ $a = 2$ $f(a) = 4$

$a = 2.001$ $f(a) = 4.004$

② $f(a) = a^3$ $\frac{d}{da} f(a) = 3a^2$ $a = 2$ $f(a) = 8$

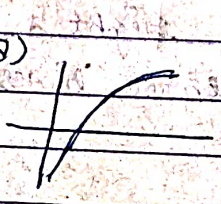
$a = 2.001$ $f(a) \approx 8.012$

$3 \times 2^2 \Rightarrow 12$

when you nudge 'a' to the right little by tiny bit f(a) should go up by 12

③ $f(a) = \log_e(a)$ or $\ln(a)$ $\Rightarrow \frac{d}{da} f(a) = \frac{1}{a}$ $a = 2, f(a) = 0.6931$

$a = 2.001, f(a) = 0.693$ gone up by 0.0005



$\frac{d}{da} f(a) = 1/2$ go up by 1/2.

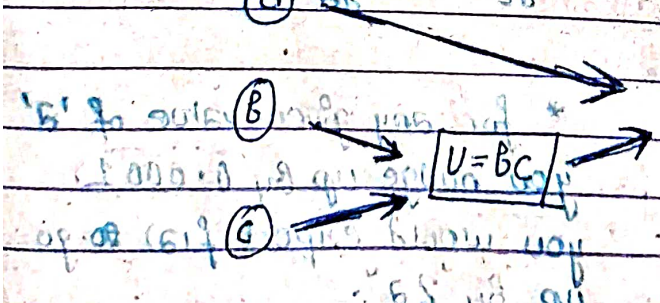
★ Computation Graph:

Let, $J(a,b,c) = 3(a + b*c)$

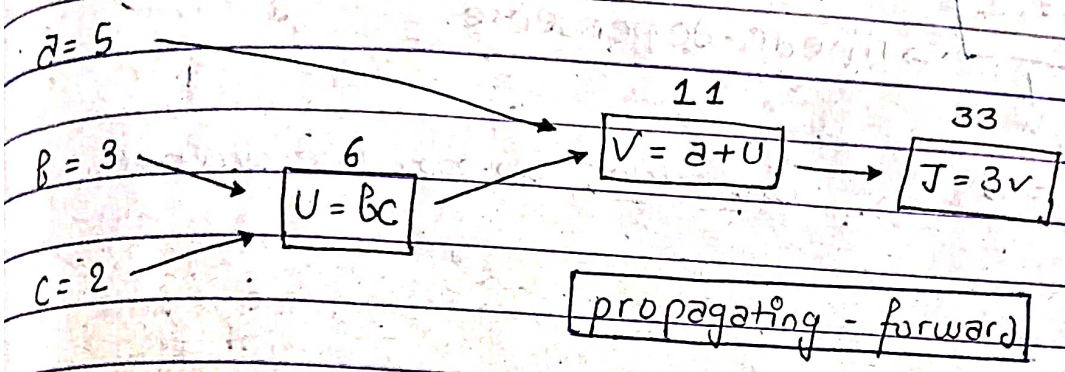
$u = bc$

$v = a + u$.

$J = 3v$



Derivatives with computation graph:



→ $\frac{dJ}{dv} = ?$ ○ if we were to take this value (V) and change it little bit. How would the value of "J" change.

$J = 3V$

$V = 11$ pump up to → 11.001 'J' bump up to → 33.003

Hence, $\frac{dJ}{dV} = 3$

→ $\frac{dJ}{da} = ?$ ○ How does 'a' affect "J".

$V = a + U$

$a = 5$ bump to → 5.001

$V = 11$ bump to → 11.001

$J = 33$ bump to → 33.003

change 'a' → 'V' changes → change 'J' as $\frac{dJ}{dv} = 3$

$\frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{dv}{da}$ {chain-rule}

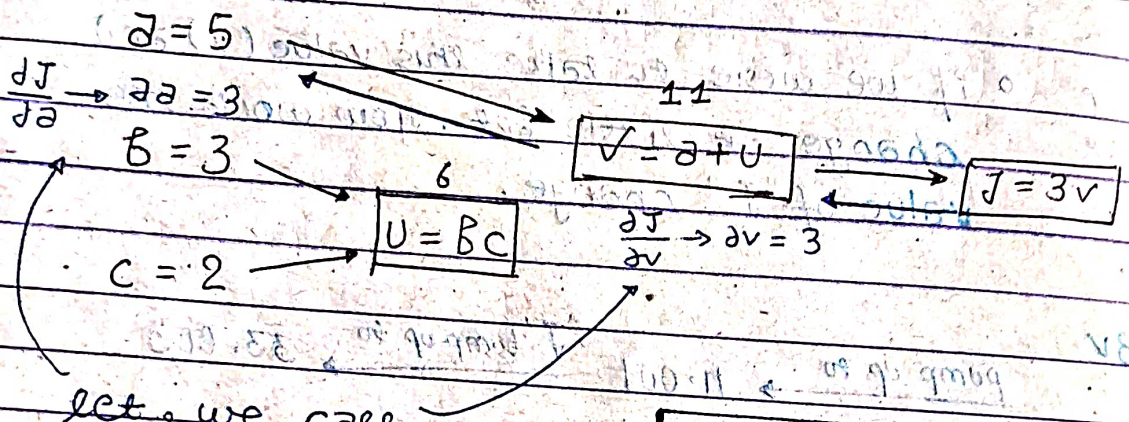
{chain-rule}

$a = 5$ (to 5.001)

$\frac{dv}{da} = 1$ as $(v = 11.001)$

↳ linear-dependence.

$\frac{dJ}{dv} = 3$ outcome to produce output



let, we call propagating-backward (for writing code)

→ $\frac{dJ}{du} = ?$ $u = 6$ bump to 6.001

$v = 11 \xrightarrow{\text{u}} 11.001$

$J = 33 \xrightarrow{\text{u}} 33.003$

$\frac{dJ}{dv} \cdot \frac{dv}{du}$ (chain-rule)

$3 \quad 1$

→ $\frac{dJ}{db} = ?$ $\frac{dJ}{du} \cdot \frac{du}{db}$ (chain-rule)

$b \rightarrow 3 \rightarrow 3.001$

$u = b \cdot c \rightarrow 6 \rightarrow 6.002$ (as $c = 2$)

'u' has gone up twice as 'b' has gone up

$\frac{dJ}{db} \Rightarrow \frac{dJ}{du} \cdot \frac{du}{db} \rightarrow 6$

$b = 3 \longrightarrow 3.001$) as $(c = 2)$

$u = b \cdot c = 6 \longrightarrow 6.002$

↳ $J = 33.006 \rightarrow V = 11.002$ $\left\{ as \ \frac{dJ}{db} = 6 \right\}$

$\rightarrow J = 3V$

outcome: which variable has impact to ↑↑ efficiency

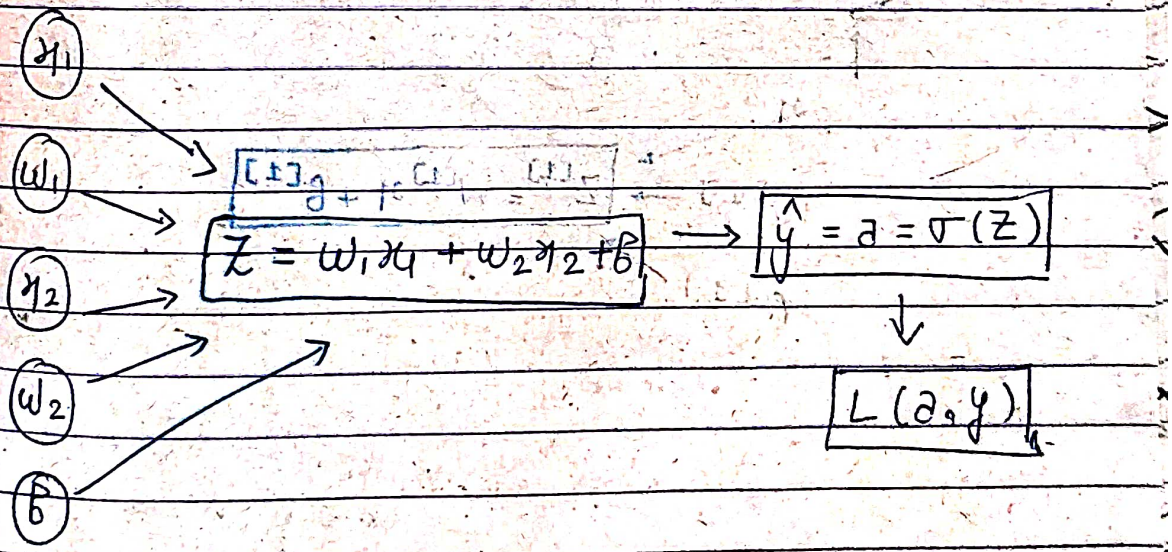
★ Logistic Regression in Neural Network

LoR recap

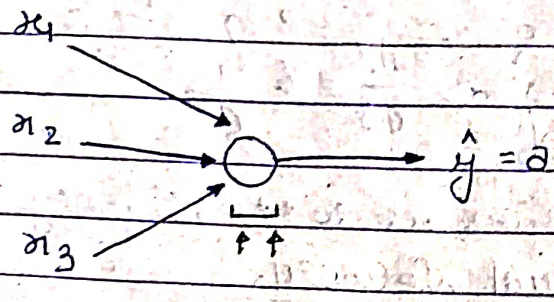
$$z = w^T x + b \qquad (h\theta(x) = \theta^T x + \epsilon)$$

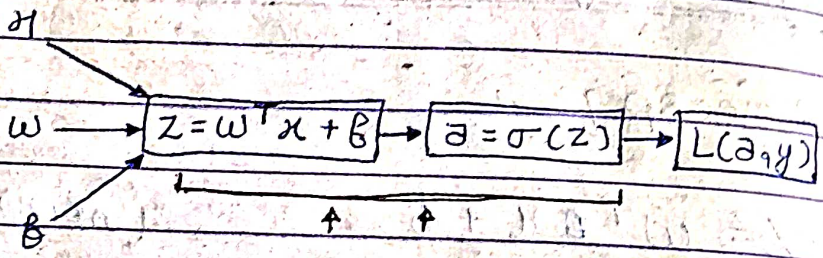
$$\hat{y} = a = \sigma(z) \qquad (h\theta(x) \rightarrow \text{predicted} \rightarrow \hat{y})$$

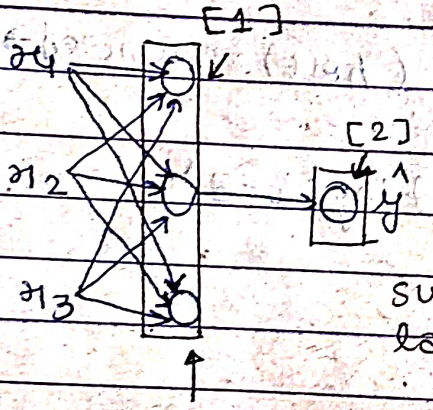
$$L(a, y) = -(y \log(a) + (1-y) \log(1-a))$$



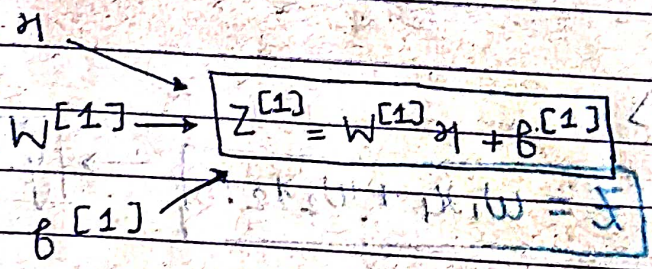
Neural Network Representation :

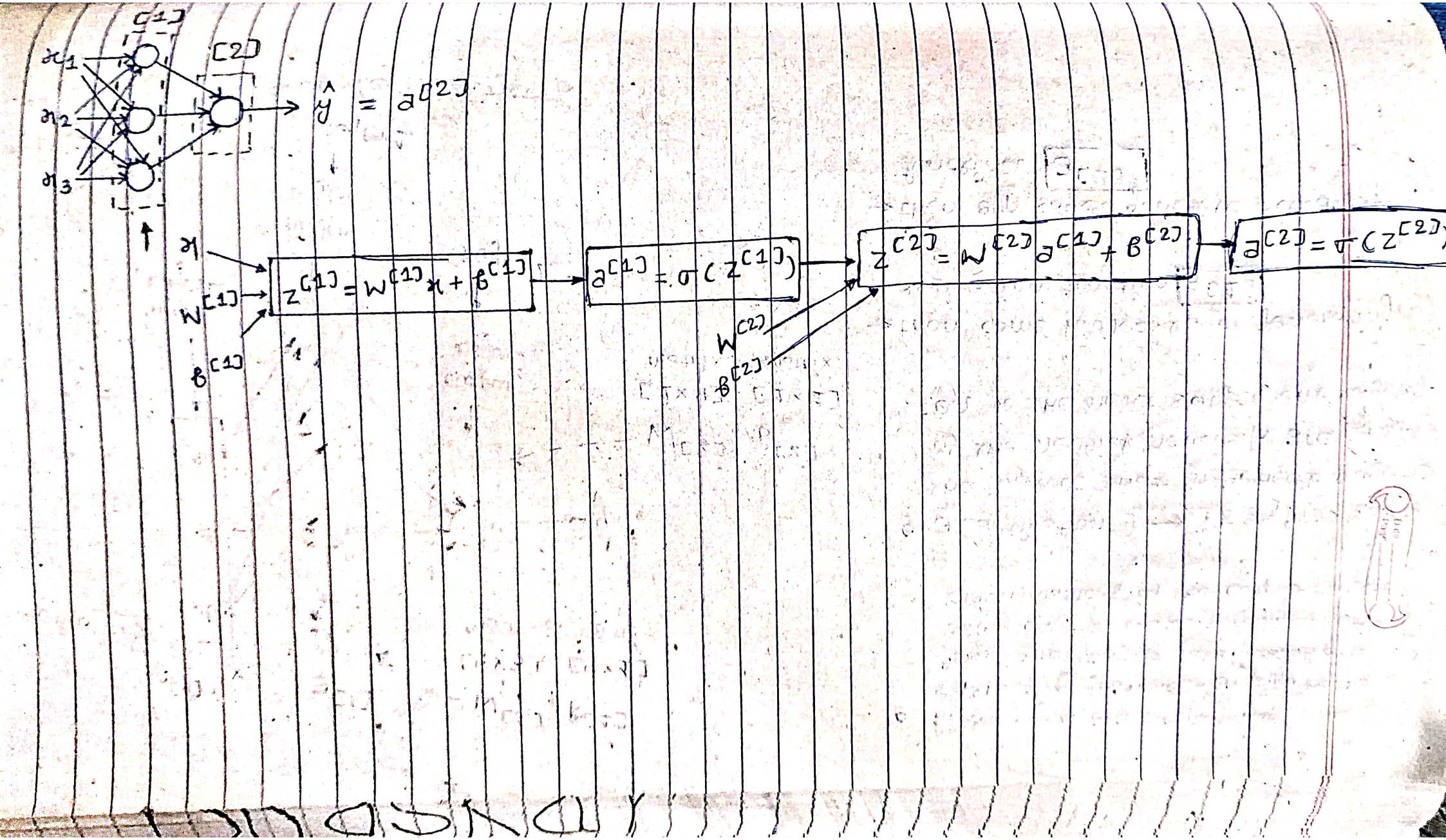


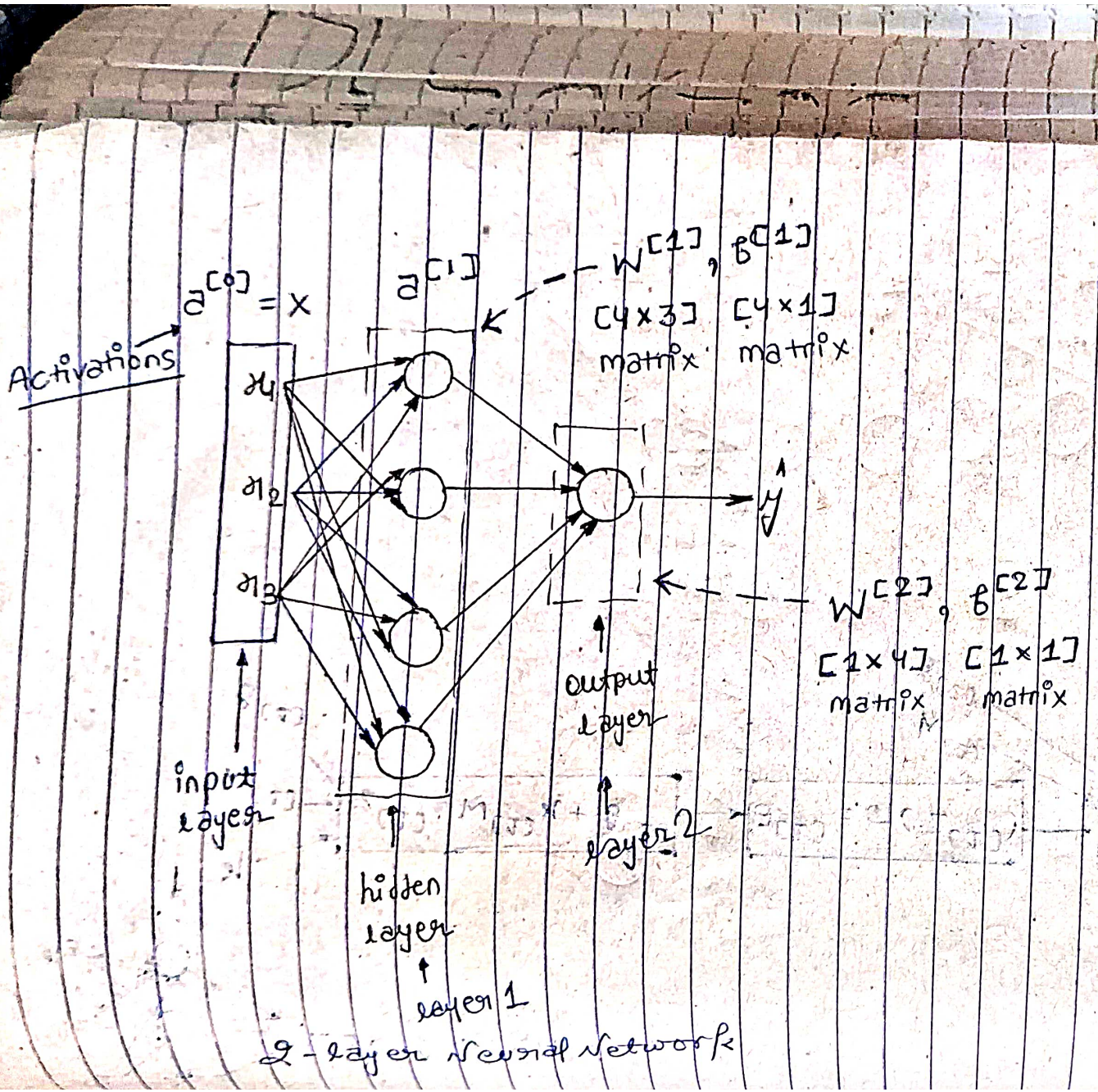




superscript refer to belong to that layer




[1]
[2]
x_1
x_2
x_3
$\hat{y} = a^{[2]}$
x
$W^{[1]}$
$b^{[1]}$
$z^{[1]} = W^{[1]} x + b^{[1]}$
$a^{[1]} = \sigma(z^{[1]})$
$W^{[2]}$
$b^{[2]}$
$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$
$a^{[2]} = \sigma(z^{[2]})$



- hidden layer are responsible for extracting features and patterns from the input data which are then used to make predictions & classifications in the output layer

- a [activations] → it refers to the values that different layers of the neural network are passing on to the next subsequent layers.

→ then come hidden layer generating activation function $a^{[1]}$

→ then opp layer generate activation function $a^{[2]}$

$a^{[1]}$ → the nodes in this activation func. $a^{[1]}$ will generate values –

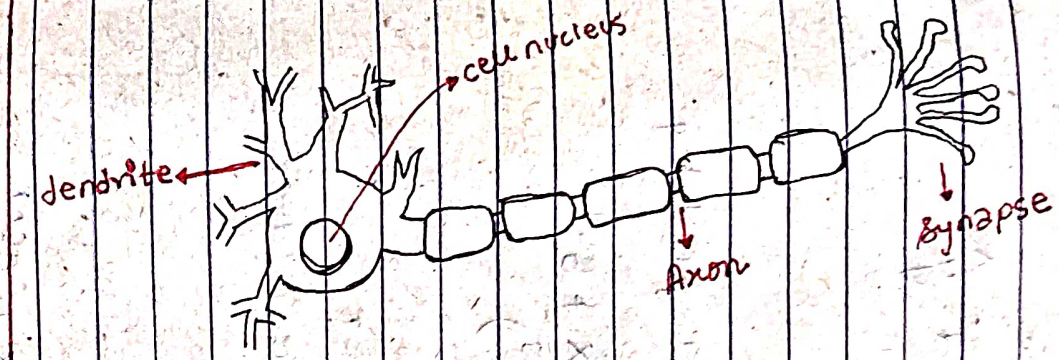
$a_1^{[1]}$, $a_2^{[1]}$, $a_3^{[1]}$, $a_4^{[1]}$

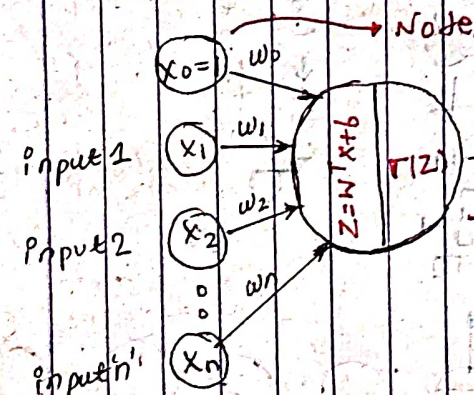
$a_1^{[1]}$ $a_2^{[1]}$ $a_3^{[1]}$ $a_4^{[1]}$

hidden layer

$$a^{[1]} \equiv \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$$

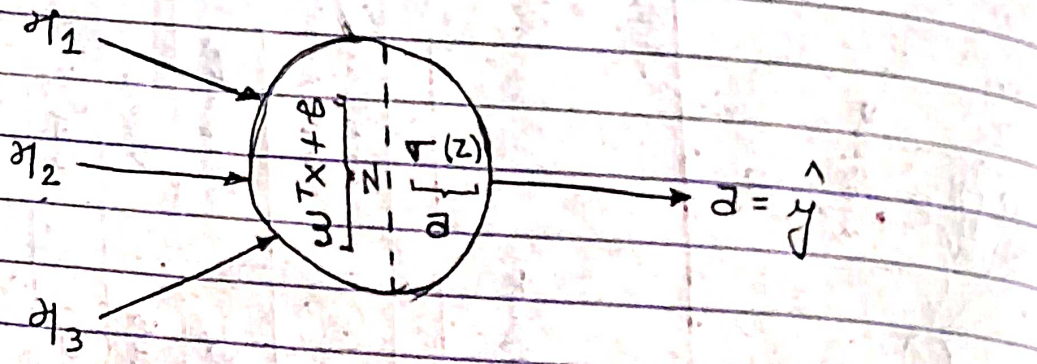
4 dimensional vector



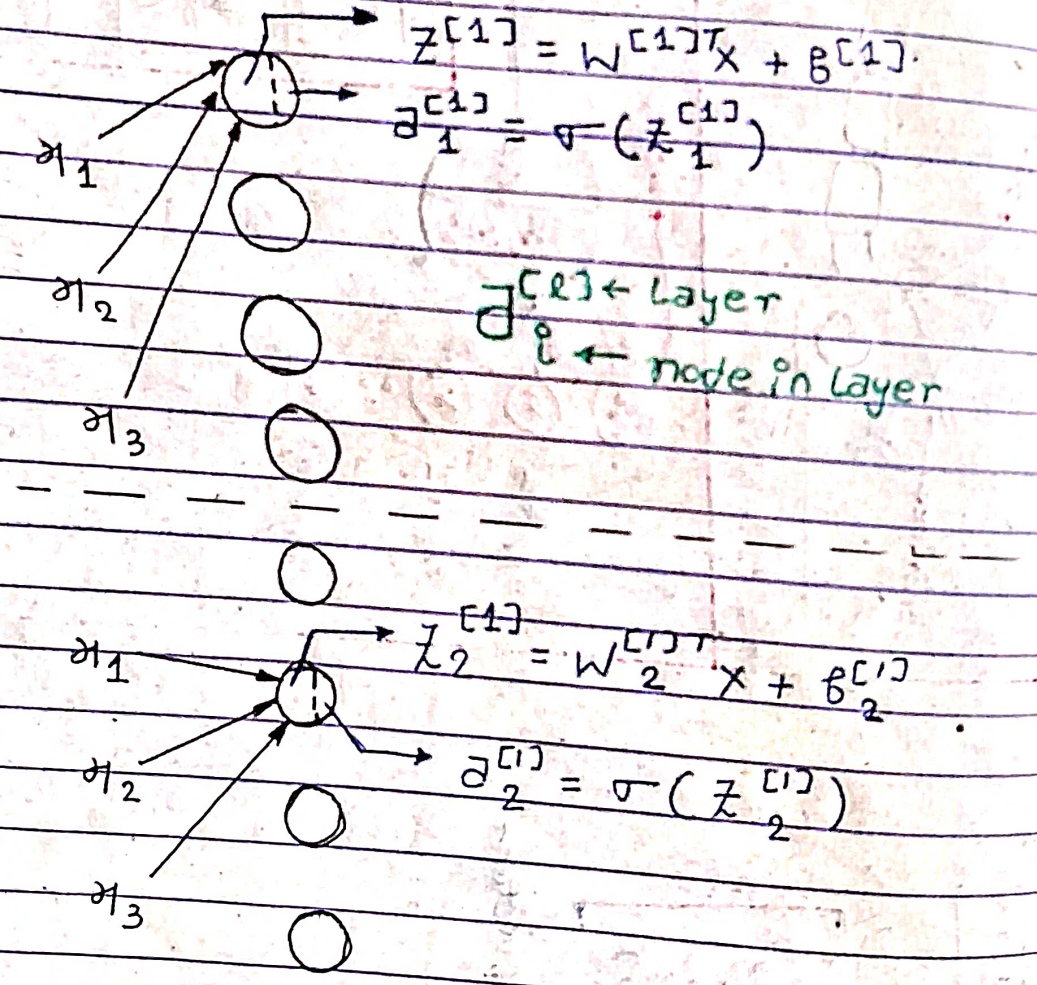


Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell Nucleus	Nodes
Synapse	weights
Axon	Output

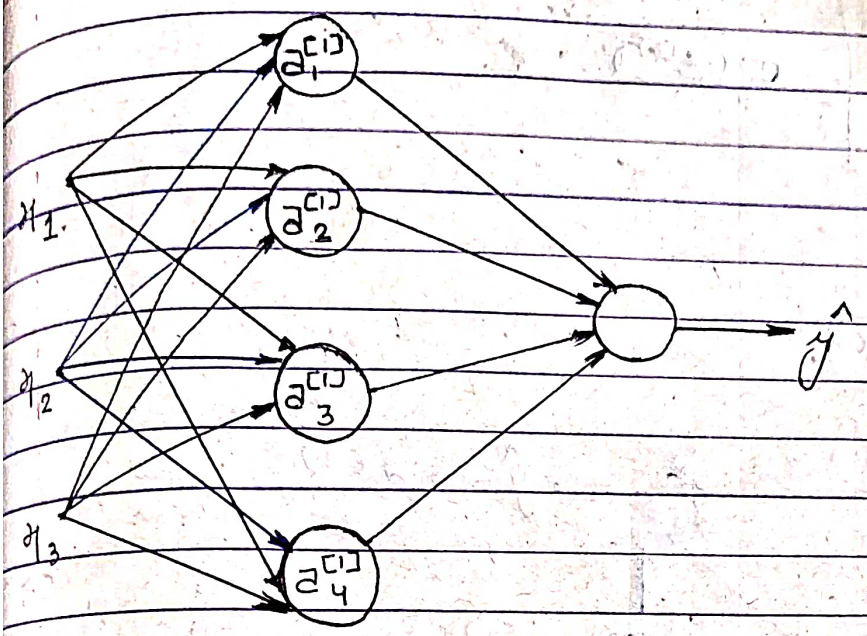
★ Computing Neural Network's Output :



Hence, for previous given 2-layer NN.



Similarly, perform with others.



$$z_1^{[1]} = W_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = W_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = W_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = W_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

★ Vectorization :

$z^{[1]} \Rightarrow$

$$\begin{bmatrix} W_1^{[1]T} \\ W_1^{[2]T} \\ W_1^{[3]T} \\ W_1^{[4]T} \end{bmatrix}$$

← stacking 4 weigh vectors

← (4×3) matrix

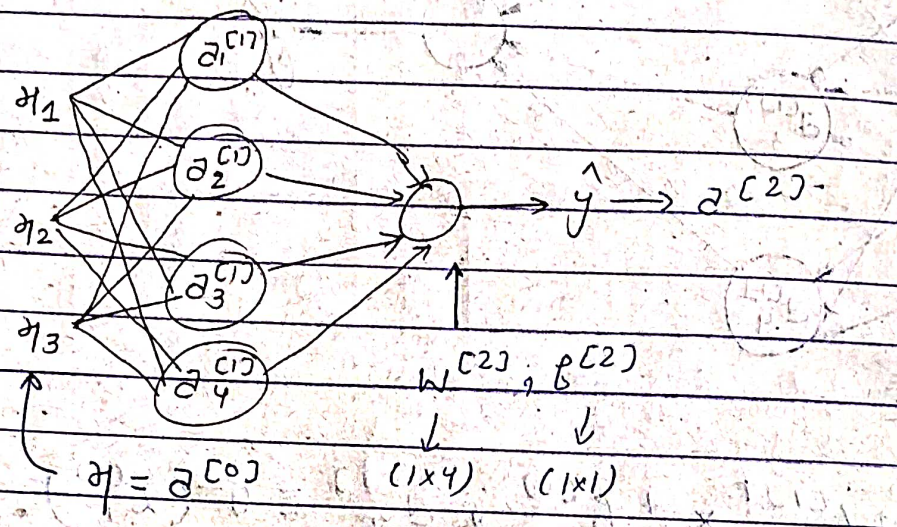
dot product ↓ *

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \end{bmatrix}$$

↑ i/p features

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

Q Outcome,



given input x

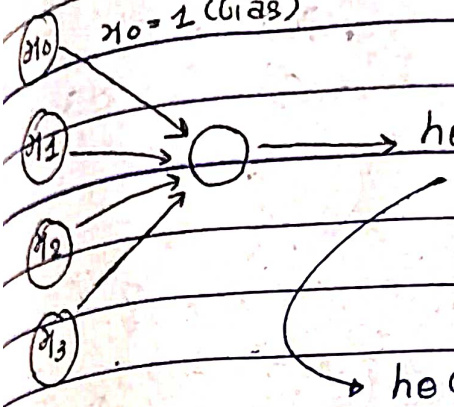
$$z^{[1]}_{(4\times1)} = W^{[1]}_{(4\times3)} a^{[0]}_{(3\times1)} + b^{[1]}_{(4\times1)}$$

$$a^{[1]}_{(4\times1)} = \sigma(z^{[1]})_{(4\times1)}$$

$$z^{[2]}_{(1\times1)} = W^{[2]}_{(1\times4)} a^{[1]}_{(4\times1)} + b^{[2]}_{(1\times1)}$$

$$a^{[2]}_{(1\times1)} = \sigma(z^{[2]})_{(1\times1)}$$

Neuron Model : Logistic Unit



$x_0 = 1$ (bias)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Sigmoid (logistic) activation func : $g(z) = 1/1+e^{-z}$

=> Activation functions !

Why do we need activation functions ?

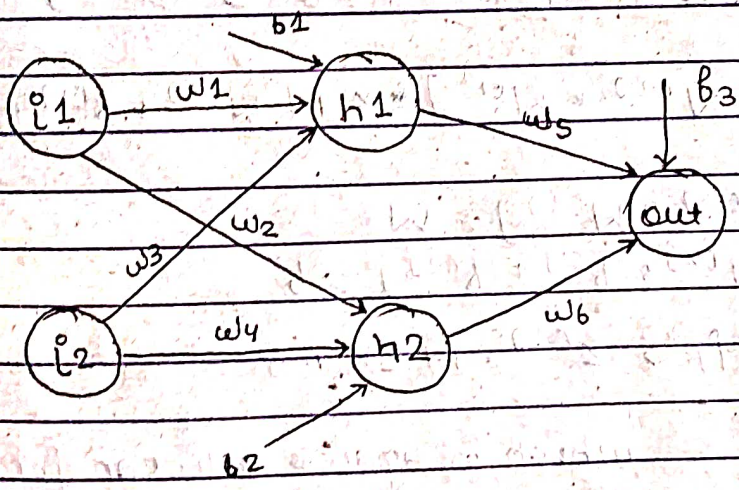
We know that neural nw has neurons that works in correspondence with weight, bias & their respective activation function.

In a neural nw we would update the weight & biases of neurons on the basis of the error at the output. This process is known as Back-propagation

- Activation functions makes the Back-propagation possible since the gradients are supplied along with the error to update the weight & biases

Q: Why do we need non-linear activation func° ?

ie,



hidden layer (layer 1)

$$Z^{[1]} = W^{[1]} X + b^{[1]} a^{[1]}$$

- $Z^{[1]}$ → vectorized op of layer 1
- $W^{[1]}$ → vectorized weights assigned to neurons of hidden layer i.e. $w_1^1, w_2^1, w_3^1, w_4^1$
  ↳ w_1, w_2, w_3, w_4

- X → vectorized i/p features (i.e.) i1 and i2
- B → vectorized bias assigned to neurons in hidden layer i.e. $B1, B2$

  ↳ $b_1^1 \quad b_2^1$

- $a^{[1]}$ → vectorized form of any linear func°

layer 2 (i.e. output) layer

i/p for layer 2 is o/p from layer 1

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

$$a^2 = \sigma(z^{[2]})$$

calculation at o/p layer

$$z^{[2]} = (W^{[2]} * [W^{[1]} X + B^{[1]}] + B^{[2]})$$

$$z^{[2]} = [W^{[2]} * W^{[1]}] * X + [W^{[2]} * b^{[1]} + b^{[2]}]$$

Let,

$$[W^{[2]} * W^{[1]}] = W$$

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

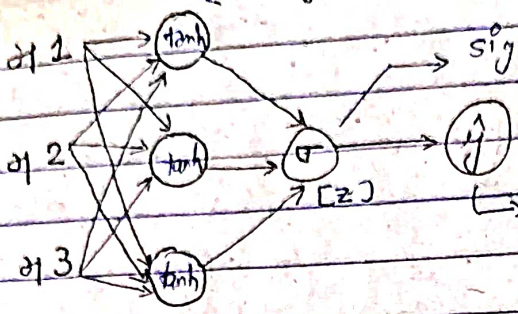
final o/p → $z^{[2]} = W * X + b$

which is again a linear func°

even after applying neural nw

Hence, we can conclude that, doesn't matter how many hidden layer we attach in neural net, all layer will Behave same way Because, the composition of two linear func° is linear itself

$g^{[1]}(z^{[1]}) = \tanh(z^{[1]})$



sigmoid $g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$

↳ ye hoga y

$0 \le \hat{y} \le 1$

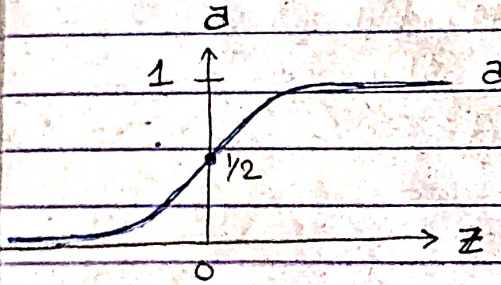
given ki:

$z^{[1]} = W^{[1]} x + B^{[1]}$

$a^{[1]} = \sigma(z^{[1]}) \rightarrow g^{[1]}(z^{[1]})$

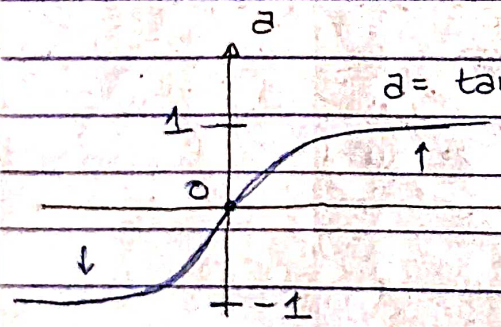
$z^{[2]} = W^{[2]} a^{[1]} + B^{[2]}$

$a^{[2]} = \sigma(z^{[2]}) \; g^{[2]}(z^{[2]})$.



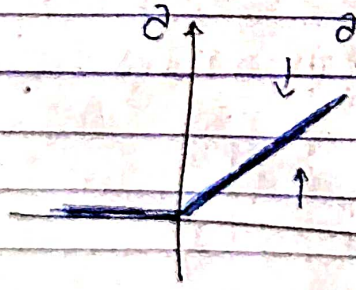
$a = \frac{1}{1+e^{-z}}$

- used when need binary classification o/p (0 or 1)
- Historically used in hidden layers ~~not preffered due to~~ vanishing gradient problem



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

- similar to sigmoid But centered at 0 which helps with vanishing gradient
- Commonly used in ~~Hidden~~ layers
- Range : $[-1, 1]$

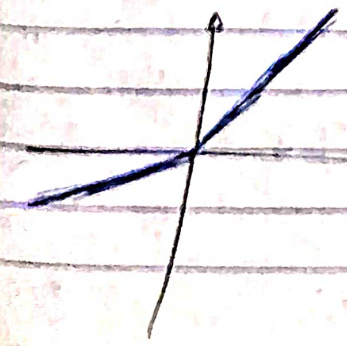


$a = \max(0, z)$

~~fast computation &~~ promote sparsity

range : $[0, +\infty]$

- A variant of RELU
- helps in dying RELU problem allow a small non-zero gradient when the o/p is 0ve.

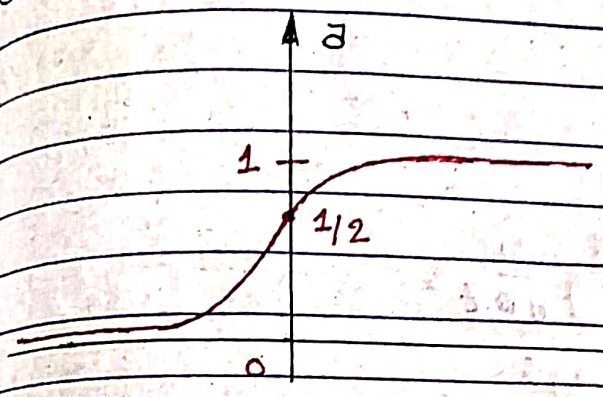


$a = \max(0.1z, z)$

- Help in trainning deeper nw
- o/p : $[-\infty, +\infty]$

*Derivatives of Activation functions :

① Sigmoid Activation function



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

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

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

$$\Rightarrow g(z)(1-g(z)) \quad \text{or} \quad a(1-a)$$

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

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

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

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

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

$z = 0 \longrightarrow g(z) = 1/2$

$\frac{d}{dz} g(z) \approx \frac{1}{2}(1-1/2) \Rightarrow 1/4$

derivation:

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

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

Apply chain rule $\Rightarrow \frac{df(u)}{dz} \Rightarrow \frac{df}{du} \cdot \frac{du}{dz}$

$u = (1+e^{-z})$, $f = u^{-1}$

$$\Rightarrow \frac{d}{du}(u^{-1}) \frac{d}{dz}((1+e^{-z}))$$

$\frac{d}{du}(u^{-1}) \rightarrow -1/u^2$

$$\Rightarrow \frac{1}{(1+e^{-z})^2} \frac{d}{dz}(1+e^{-z})$$

$$\Rightarrow \frac{1}{(1+e^{-z})^2} \cdot e^{-z} \frac{d}{dz}(-z) \Rightarrow \frac{e^{-z}}{(1+e^{-z})^2}$$

generalize:

$$\frac{1+e^{-z}-1}{(1+e^{-z})^2} \Rightarrow \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2}$$

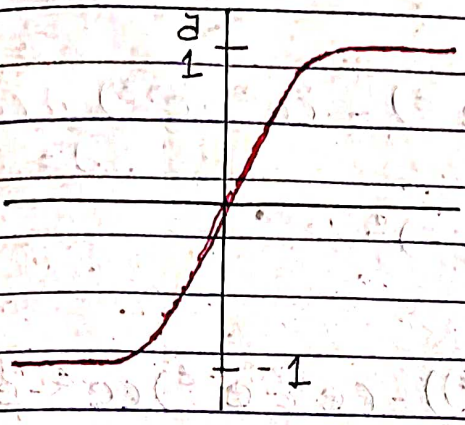
$$\Rightarrow \frac{1}{(1+e^{-z})} * \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$\Rightarrow g(z)(1-g(z))$$

or

$(\partial \in 1-\partial)$?

(2) Tanh Activation function



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

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

$$\Rightarrow 1 - (\tanh(z))^2$$

$$a = g(z) \longrightarrow \boxed{g'(z) = 1 - a^2}$$

- $z = 10 \rightarrow \tanh(z) \approx 1$, $g'(z) \approx 0$
- $z = -10 \rightarrow \tanh(z) \approx -1$, $g'(z) \approx 0$
- $z = 0 \rightarrow \tanh(z) = 0$, $g'(z) = 1$

derivation,

The quotient rule: $\frac{d}{dx}\left(\frac{u}{v}\right) \Rightarrow \frac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$

by chain rule: $\frac{d}{dx} e^{ax} \Rightarrow e^{ax} \cdot \frac{d}{dx}(ax)$

$$\Rightarrow a \cdot e^{ax}$$

As a particular case of this we have,

$$\frac{d}{dz} e^{-z} \Rightarrow -e^{-z}$$

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

$$\frac{(e^z + e^{-z})\frac{d}{dz}(e^z - e^{-z}) - (e^z - e^{-z})\frac{d}{dz}(e^z + e^{-z})}{(e^z + e^{-z})^2}$$

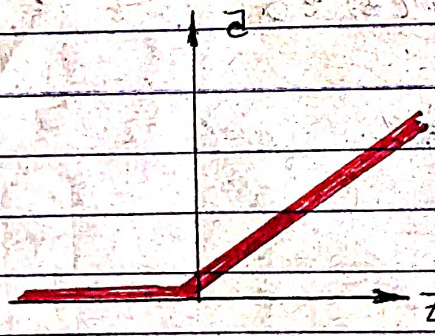
$$\frac{(e^z + e^{-z})(e^z - (-e^{-z})) - (e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2}$$

$$1 - (\tanh(z))^2$$

or

$$g'(z) = 1 - a^2$$

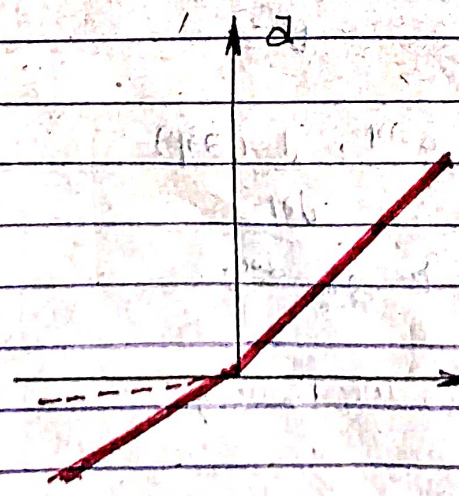
③ ReLU Activation function



$$g(z) = \max(0, z)$$

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

④ Leaky ReLU Activation function

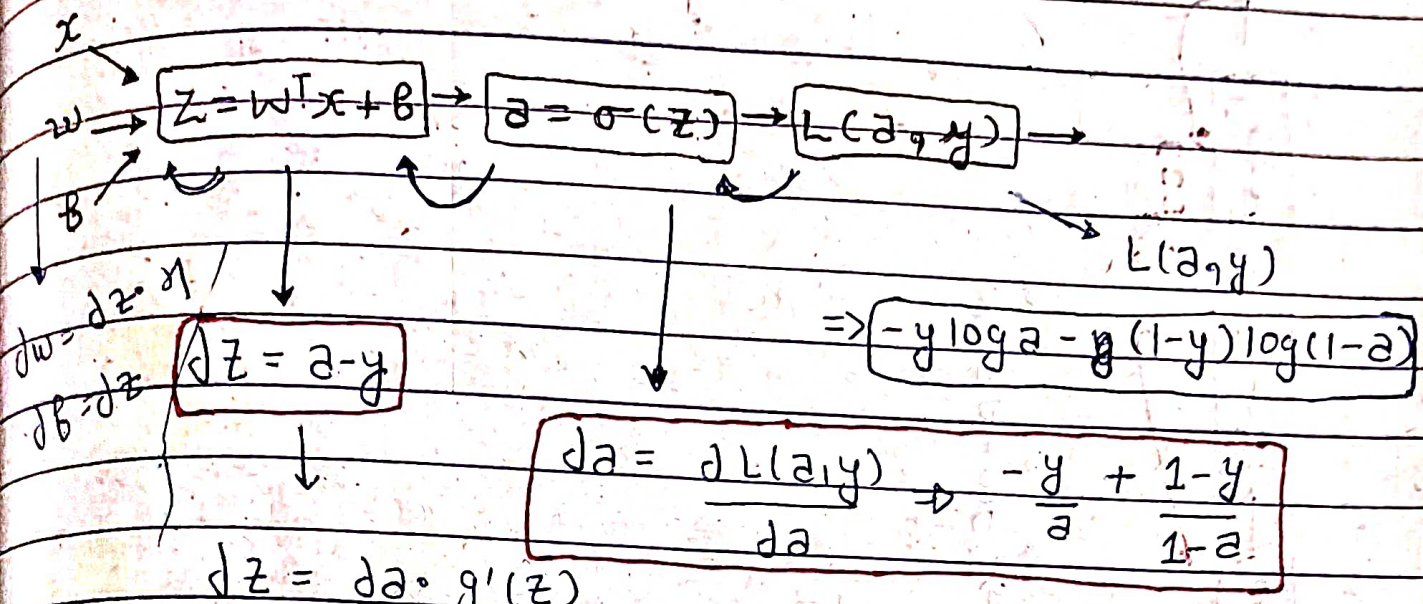


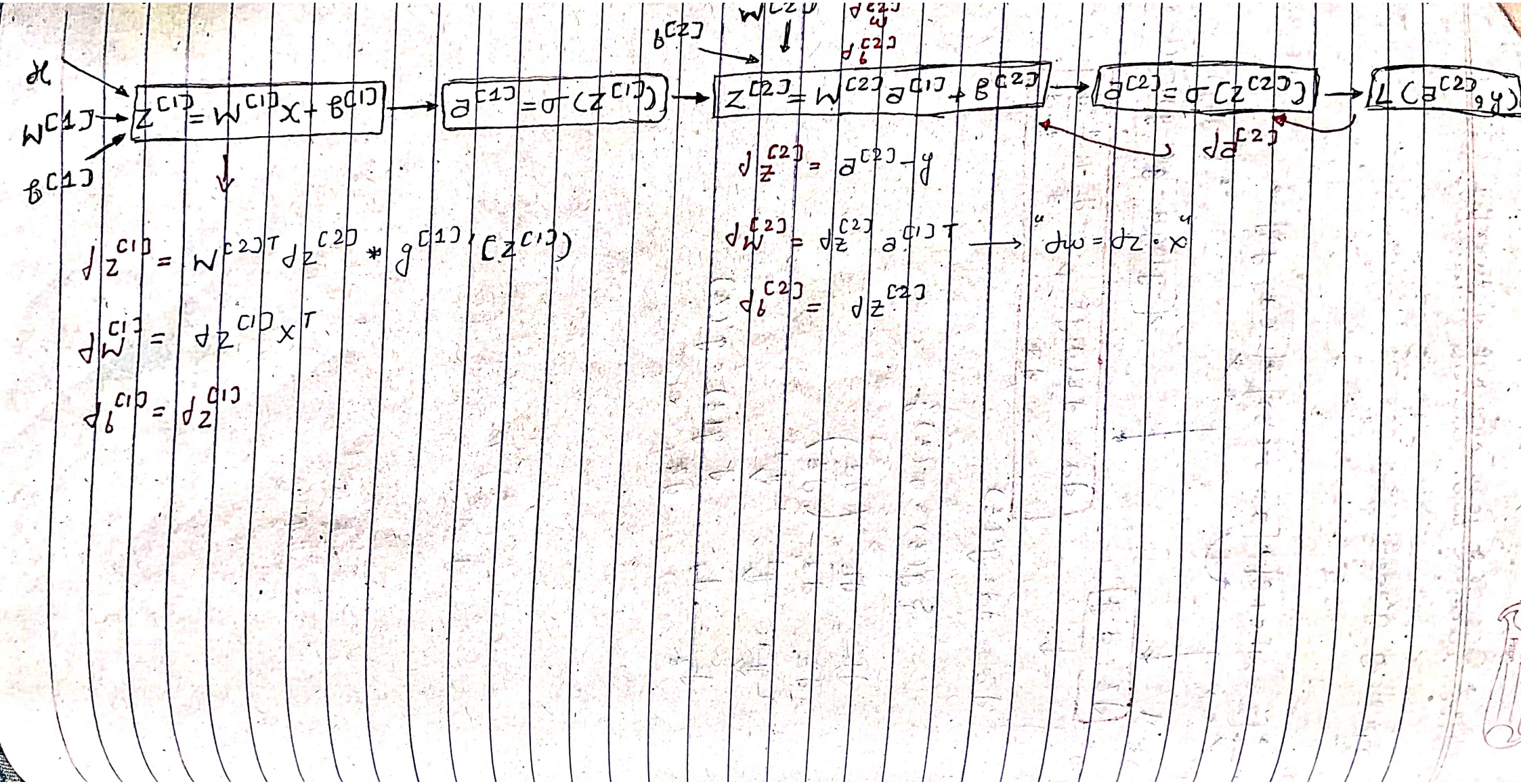
$$g(z) = \max(0.01z, z)$$

$$g'(z) \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$$

Gradient Descent for Neural Networks

Computing gradients -




x
W[1]
b[1]
Z[1] = W[1] x + b[1]
a[1] = σ(Z[1])
b[2]
W[2]
Z[2] = W[2] a[1] + b[2]
a[2] = σ(Z[2])
L(a[2], y)
dZ[2]
dZ[2] = a[2] − y
dW[2] = dZ[2] a[1]T → dw = dz · x
db[2] = dZ[2]
dZ[1] = W[2]T dZ[2] * g[1]'(Z[1])
dW[1] = dZ[1] xT
db[1] = dZ[1]

Gradient Descent :-

$$dz^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis = 1, keepdims = True)$$

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

$$dW^{[1]} = \frac{1}{m} dz^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dz^{[1]}, axis = 1, keepdims = True)$$