Problem statement:-

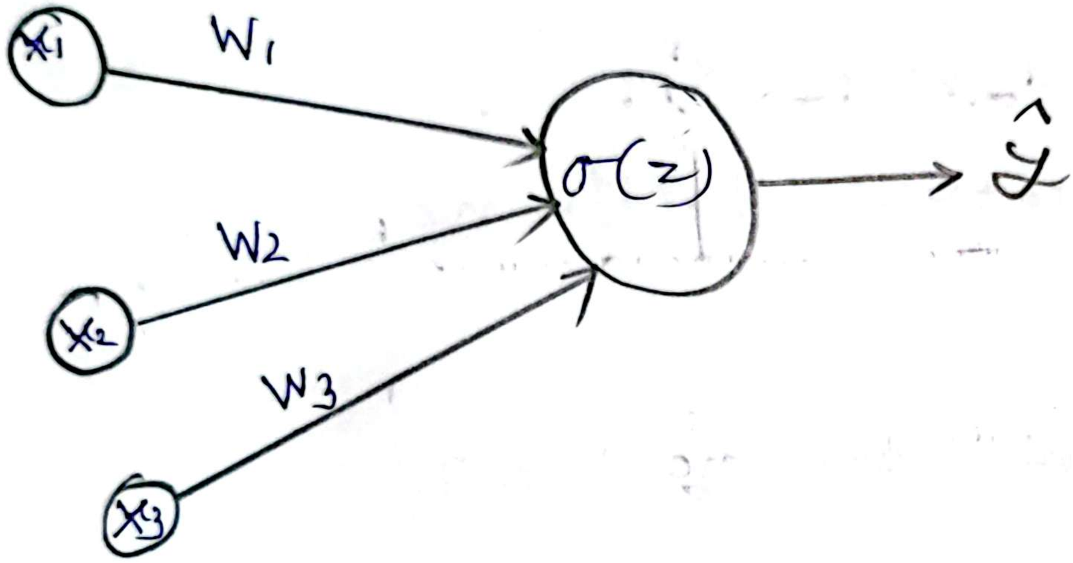
For training:

exp No.	Input			Output
	x_1	x_2	x_3	
exp-1	0	0	1	0
example-2	1	1	1	1
example-3	1	0	1	1
example-4	0	1	1	0

For testing.

build a simple neural network, with no hidden layer to train with the examples given above.

The neural network with no hidden layer.



Here, $z = \sum W_i x_i = W_1 x_1 + W_2 x_2 + W_3 X_3 + b$

[b = bias]

and $\sigma(z)$ = sigmoid function

where $\sigma(x) = \dfrac{1}{1+e^{-x}}$

or, $\sigma(z) = \dfrac{1}{1+e^{-z}}$

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

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

$$= \sigma(z)(1-\sigma(z))$$

For forward propagation:- the code is:-

```
import numpy as np.

def sigmoid(x):
    return 1/(1+np.exp(-x))


# Define training inputs

training_inputs = np.array([[0,0,1],
                            [1,1,1],
                            [1,0,1],
                            [0,1,1]])


input_layer_size = 3
output_layer_size = 1

weights = np.random.random((input_layer_
          size,output_layer_size))*0.01
```

bias = np.zeros((1, output_layer_size, 1))

z = np.dot(training_inputs, weights) + bias

A = sigmoid(z)

why z = np.dot(train input, weights)

np.dot(training_input, weights)

matrix multiplication

= training_input * weights

example-1 (inputs)

$$
= \begin{array}{r} \\ \text{exp-2} \\ \text{exp-3} \\ \text{exp-4} \end{array}
\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}
*
\begin{bmatrix} -.1659 \\ -0.440649 \\ -0.99771 \end{bmatrix}
$$

1×3   3×1

$$= \begin{bmatrix} -0.997 \\ -1.609 \\ -1.63 \\ -1.938 \end{bmatrix} = \begin{bmatrix} \Sigma w_i x_i \text{ (example-1)} \\ \Sigma w_i x_i \text{ (example-2)} \\ \Sigma w_i x_i \text{ (example-3)} \\ \Sigma w_i x_i \text{ (example-4)} \end{bmatrix}$$

So, by matrix multiplication ~~or np.dot(inputs, weights)~~

or np.dot(training-inputs, weights)

We can get the value of

(Z) more efficiently. that's why we

np.dot(training inputs, weights).

$\boxed{\text{like:}}$

$\Sigma w_i x_i$ (example-1), Here, $x_1 = 0$, $x_2 = 0$, $x_3 = 1$

so,

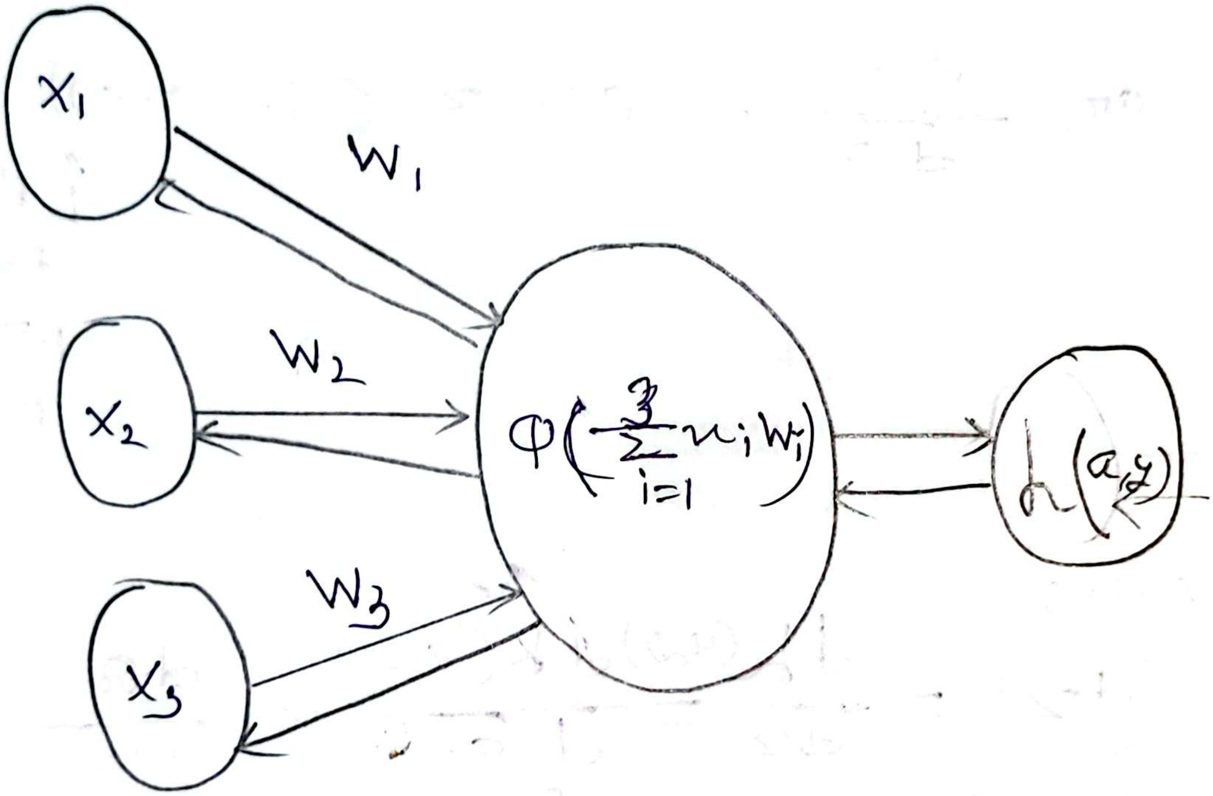
$$w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$= (-0.1659 \times 0 + 0 \times w_2 + (-0.9971) \times 1)$$

$$\Sigma w_i x_i = -0.9971 \text{ (example-1)}$$

Similarly for example-2, 3 and 4 same can be said.

For backpropagation:-



Here $h(\hat{y}, y)$ is the cost function.

It is function which measures how well the model is performing. or, it can measure the difference between the pridicted output ($\hat{y}$) to the actual output.

and It is given by,

$$\mathcal{L}(a, y) = -\left(y \log(a) + (1-y) \cdot \log(1-a)\right)$$

Here, $a = \sigma(z) = \frac{1}{1+e^{-z}}$

or, $\frac{da}{dz} = \sigma(z)(1 - \sigma(z))$

[See forward propagation part].

Again,

$$'dz' = \frac{d\mathcal{L}(a,y)}{dz} = \frac{d\mathcal{L}(a,y)}{da} \times \frac{da}{dz}$$

(in the code)

$$'dz' = \underbrace{\frac{(a-y)}{a(1-a)}}_{\frac{d\mathcal{L}}{da}} \left(\sigma(z)(1 - \sigma(z))\right)$$

$$= \frac{a-y}{a\cancel{(1-a)}} \cancel{(a)} \cdot \cancel{(1-a)}$$

$$= a - y$$

$\therefore$ 'dz' = error

similarly,

$$\frac{dL}{dw_1} = 'dw_1' = \frac{dL}{dz} \times \frac{dz}{dw_1}$$

$$= 'dz' \times \left(\frac{d}{dw_1} (w_1 x_1 + w_2 x_2 + w_3 x_3 + b) \right)$$

$$= ('dz') * (x_1)$$

$$\boxed{\therefore 'dw_1' = (x_1)('dz')}$$

similarly $\boxed{'dw_2' = (x_2)\,'dz'}$

$$\boxed{'dw_3' = (x_3)('dz')}$$

and in the same way,

$$\boxed{\frac{\partial L(a,y)}{\partial b} = 'dz'}$$

So, For m training examples the backpropagation code is described in the code:-

}

$\hat{y}$ = predicted sygmoid output

error = A - training_outputs → actual outputs

dz = error

dw = (1/m) * np.dot(training_inputs.T, dz)

db = (1/m) * np.sum(dz, axis=0, keepdims=True)

weighth and bias update:

weights = weights - learning_rate * dw

biases = biases - learning_rate * db.

why dw = ($\frac{1}{m}$)(np.dot(training_inputs.T, dz) ?

[Ans:] np.dot(training_inputs.T, dz) finds the ~~sum~~ of all the weight updates, in the form a matrix, and Here's How:-

$$\text{Training\_inputs.T} = \begin{array}{c c} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} & \begin{array}{l} x_1 \\ x_2 \\ x_3 \end{array} \end{array}$$

3×4

and let $dz = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$ 4×1

matrix multiplication

so, $(\text{Training\_inputs.T}) * (dz)$

$$= \begin{bmatrix} 0.5 \\ 0.6 \\ 0.1 \end{bmatrix} = \begin{bmatrix} x_1\, dz \\ x_2\, dz \\ x_3\, dz \end{bmatrix}$$

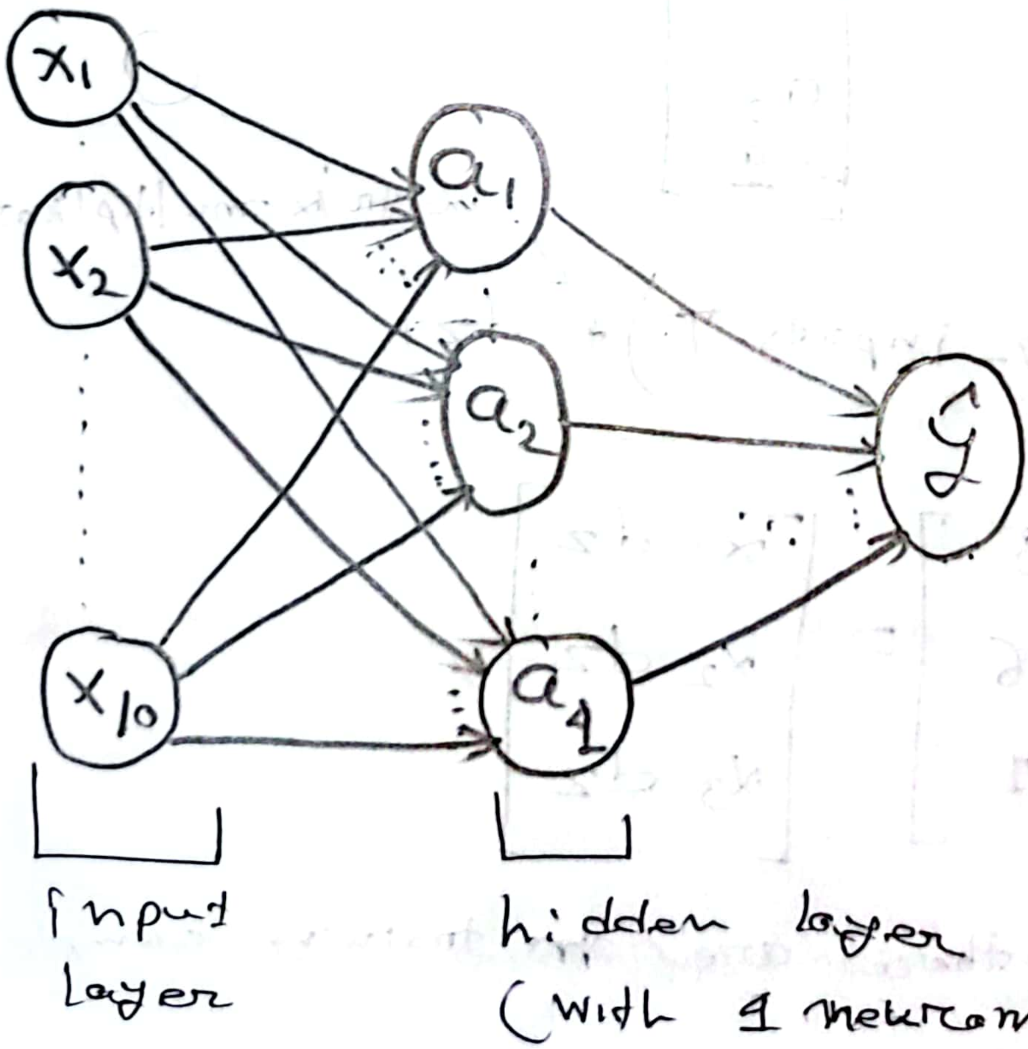
lastly as there are m training example,

so we $\left(\frac{1}{m}\right)$ is introduced.

building a simple neural network (with one hidden layer) from ~~sercoch~~ scratch

problem statement is given in the code.

The proposed neural network structure is given below:-



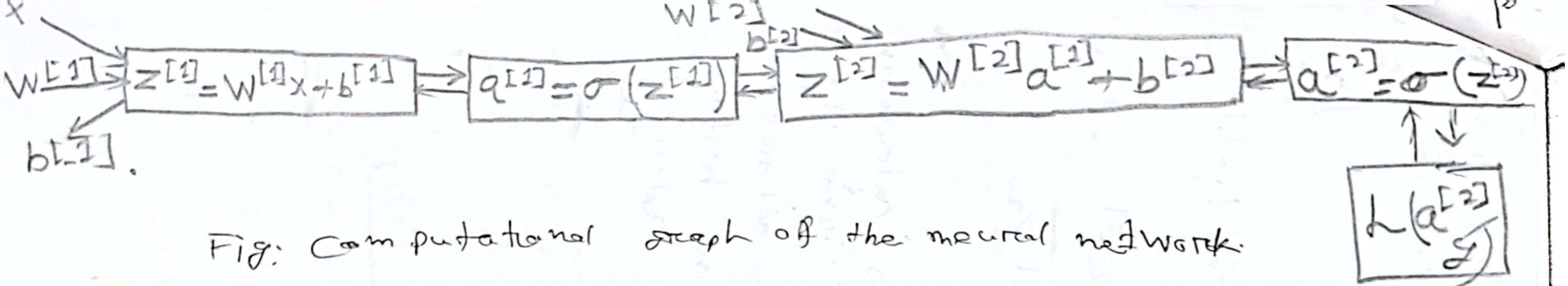


Fig: Computational graph of the neural network.

$\longrightarrow$ Forward propagation

$\longleftarrow$ backward propagation

Forward propagations

the equations are:-

$$z_1^{[1]} = W^{[1]} x + b_1^{[1]}$$

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

$$z^{[2]} = W^{[2]} a^{[1]} + b_2^{[2]}$$

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

$$\frac{d\mathcal{L}(a_2, y)}{dz_1} = \frac{d\mathcal{L}}{dz_2} \cdot \frac{dz_2}{da_1} \cdot \frac{da_1}{dz_1}$$

$$\text{"}dz_1\text{"} = \text{"}dz_2\text{"} \cdot W_2 \, (a_1)(1 - a_1)$$

$$\therefore \frac{d\mathcal{L}}{dW_1} = \frac{d\mathcal{L}}{dz_1} \times \frac{dz_1}{dW_1}$$

$$= \text{"}dz_1\text{"} \; \frac{d(W_1 x + b_1)}{dW_1}$$

$$\boxed{\text{"}dW_1\text{"} = \text{"}dz_1\text{"} \; x}$$

similarly, $\boxed{db_1 = dz_2}$

Now corresponding coding for implementing the backpropagation equations, with (m) examples

```
dZ2 = A2 - Y
dW2 = (1/m) * np.dot(A1.T, dZ2)
db2 = (1/m) * np.sum(dZ2, axis=0,
                     keepdims=True)
```

$$dZ_1 = np.dot(dZ_2, W2.T) * sigmoid\_derivative(Z_1)$$

$$dW_1 = (1/m) * np.dot(X.T, dZ_1)$$

$$db_1 = (1/m) * np.dot(dZ_1, axis=0, keepdims=True)$$