Part 2 - Understanding Forward Propagation

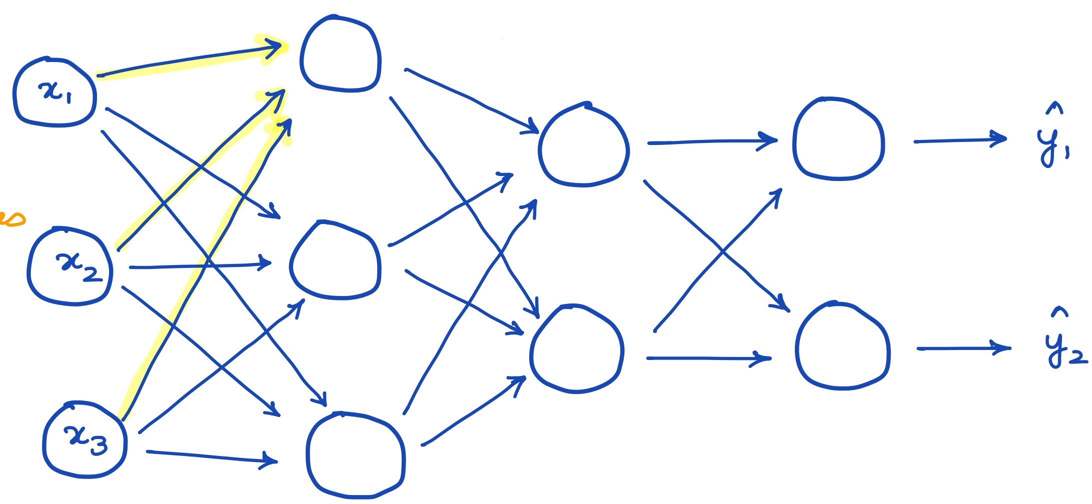
- How to pass one example?
- How to pass multiple examples (Vectorization)
- Writing the Code

Forward Propagation is most important step in neural network and if you are working with Tensor Flow or PyTorch then you just need to write forward Propagation step and Back Propagation is done automatically.

Here we will write everything from scratch:

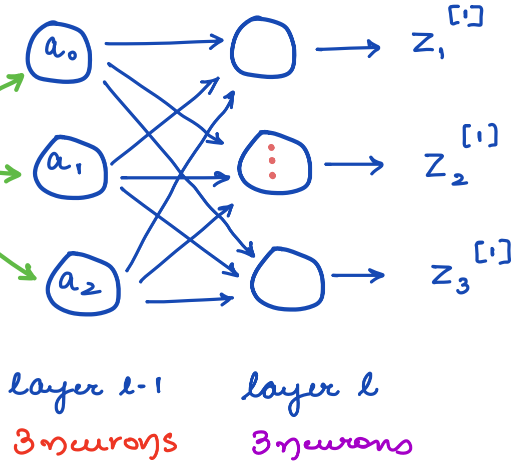
one example with 3 features

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{3\times 1}$$



$$Z = \sum w_i x_i + b$$

activations coming from previous layer



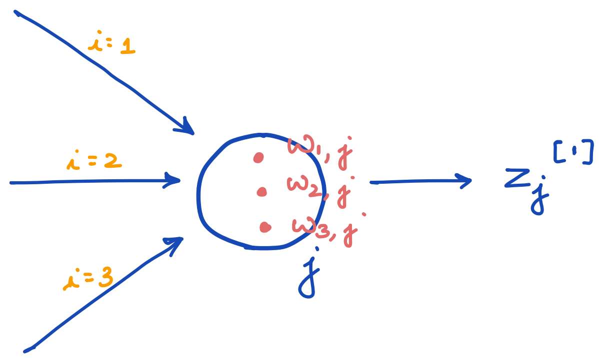
$z_j^{[\ell]} \rightarrow$ jth neuron in ℓth layer

$$\boxed{z_j^{[\ell]} = \sum_i w_{i,j}^{[\ell]} a_i^{[\ell-1]} + b}$$

$w_{i,j}$ = weight connecting the ith neuron from the previous layer to jth neuron in current layer.

i goes for all the neurons in previous layer.

j is basically for current neuron in present layer.

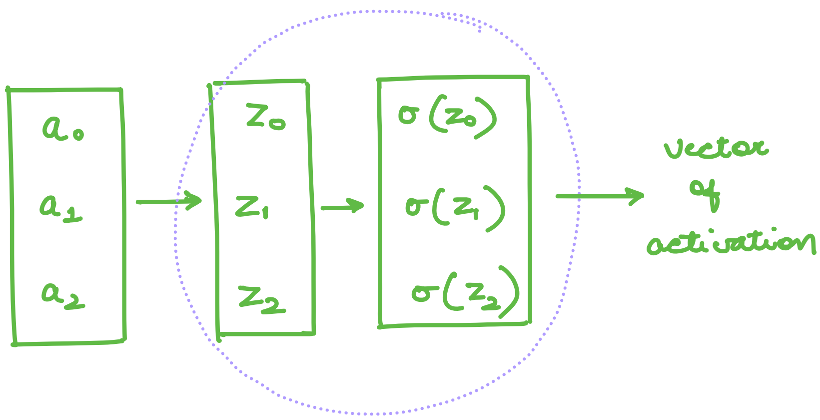


$$W^L = \begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \end{bmatrix}_{3 \times 3}$$

(w of layer L)

for 1st neuron in lth layer | for 2nd neuron in lth layer | for 3rd neuron in lth layer

for 1 example you will get 3 inputs a_0, a_1, a_2 (if you have 3 features)



this will come out of a single layer.

$\downarrow$

$Z^{[l]}$

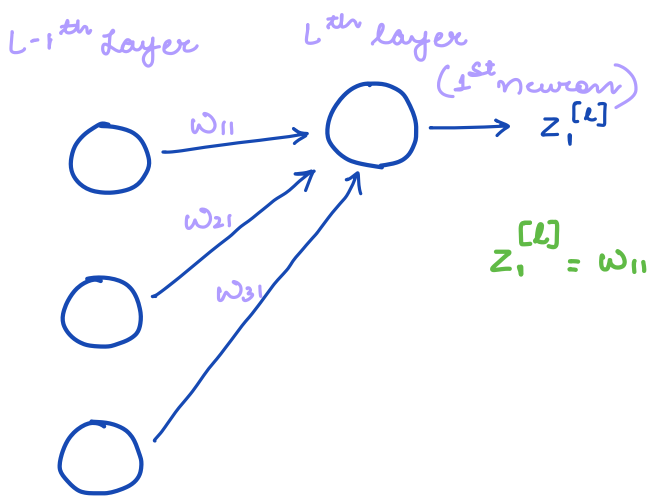
Using Vector Notation:

$$z^{[l]} = \underbrace{\underbrace{\underbrace{[w^l]^T}_{(l,\,l-1)} \underbrace{a^{l-1}}_{(l-1,\,1)}}_{(l,\,1)} + \underbrace{b^l}_{(l,\,1)}}_{l \text{ outputs}}$$

$w^l \rightarrow l-1,\ l$

$(w^l)^T \rightarrow l,\ l-1$

$$(w^L)^T = \begin{bmatrix} w_{11}^L & w_{21}^L & w_{31}^L \\ w_{12}^L & w_{22}^L & w_{32}^L \\ w_{13}^L & w_{23}^L & w_{33}^L \end{bmatrix} \begin{bmatrix} a_1^{L-1} \\ a_2^{L-1} \\ a_3^{L-1} \end{bmatrix} = \begin{bmatrix} z_1^{[l]} \\ z_2^{[l]} \\ z_3^{[l]} \end{bmatrix} = Z^{[l]}$$



$$z_1^{[l]} = w_{11} a_1 + w_{21} a_2 + w_{31} a_3$$

$$\sigma(z^{[l]}) = \begin{bmatrix} \sigma(z_1^{[l]}) \\ \sigma(z_2^{[l]}) \\ \sigma(z_3^{[l]}) \end{bmatrix} = \text{output after 1st hidden layer}$$

$$z_2 = \left(w^{[2]}\right)^T \cdot a^{[1]} + b^{[2]}$$

$$a_2 = \sigma(z_2)$$

$$z_3 = \left(w^{[3]}\right)^T \cdot a^{[2]} + b^{[3]}$$

$$\hat{y} = \text{softmax}(z_3)$$

We are not applying activation fx^n, we are taking softmax so that we can get probability distribution for each class.

Softmax can be treated like sigmoid, sigmoid works with binary classification (it reduces outputs to 0 and 1). Softmax reduces these nos into a probability distribution and sum of probability is going to be 1.

In output layer we won't use any activation.

We will get some nos:

$$z_3 = [10, 10, 20]$$

this class has higher probability

$$\xrightarrow{\text{softmax}}$$

$$[0.1, 0.1, 0.8]$$

Using softmax we reduced these nos to some probability distribution.