BUILDING A NEURAL MODEL FROM SCRATCH

NO TENSORFLOW & PYTORCH

JUST MATHS 

- ASHISH KISHORE

28 X 28
grayscale Images

784 Pixels — m training images

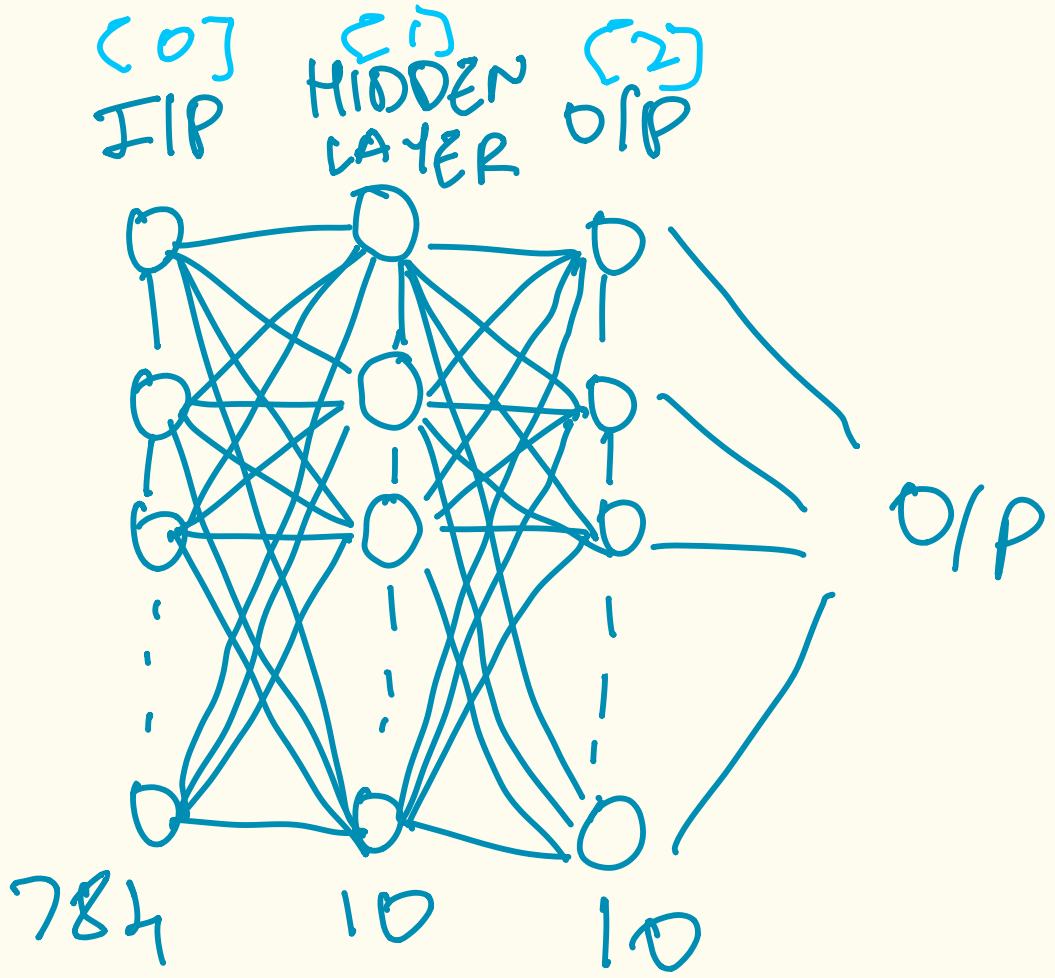
ranging from
0 - 255
0 = Black & 255 = white

$$X = \begin{bmatrix} \text{---} & x^{(1)} & \text{---} \\ \text{---} & x^{(2)} & \text{---} \\ & \vdots & \\ \text{---} & x^{(m)} & \text{---} \end{bmatrix}^T = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

→ Each row is an example
→ Each row is going to 784 columns long

→ 784 rows each column corresponding to pixel value
→ m columns [Example]

784 ⟹ 0, 1, 2 9
28 X 28
10 classes



TRAINING

A) Forward Propagation :- Take the image & run it through the NN

$$A^{[0]} = [X] \quad (784 \times m)$$

$$Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]} \rightarrow \text{Unactivated layer}$$

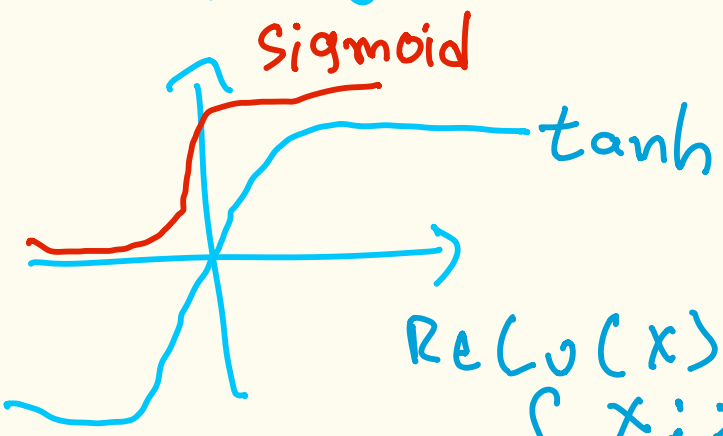
$$10 \times m = 10 \times 784 \quad 784 \times m \quad 10 \times 1 \Rightarrow 10 \times m$$

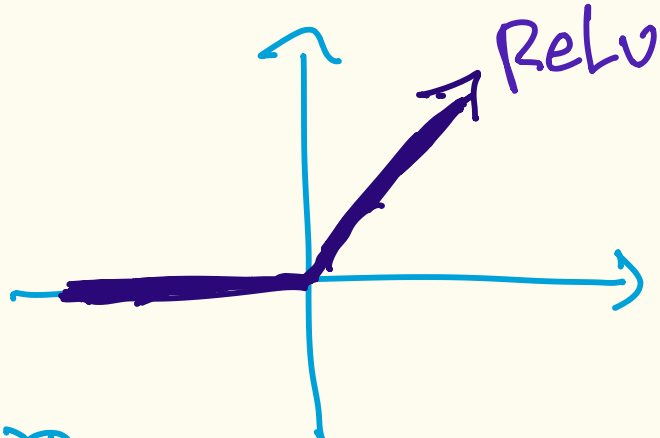
Apply activation function

→ If there is no activation, each neuron/node will be a linear combination of previous nodes with bias term.

→ The 2nd layer is gonna be the combination of nodes in 1st layer but 1st layer is just linear combination of I/P layer. If there is only linear combination, with weights & biases there will be no interesting function but a complex linear regression.

So use of ACTIVATION FUNCTION is required.





$$ReLu(x) = \begin{cases} x; & x > 0 \\ 0; & x < 0 \end{cases}$$

→ If any activation function is added layers turn into complex from linearity, layers become complex with preceding & becomes Non linear.

$$A^{[1]} = g(Z^{[1]}) = ReLU(Z^{[1]})$$

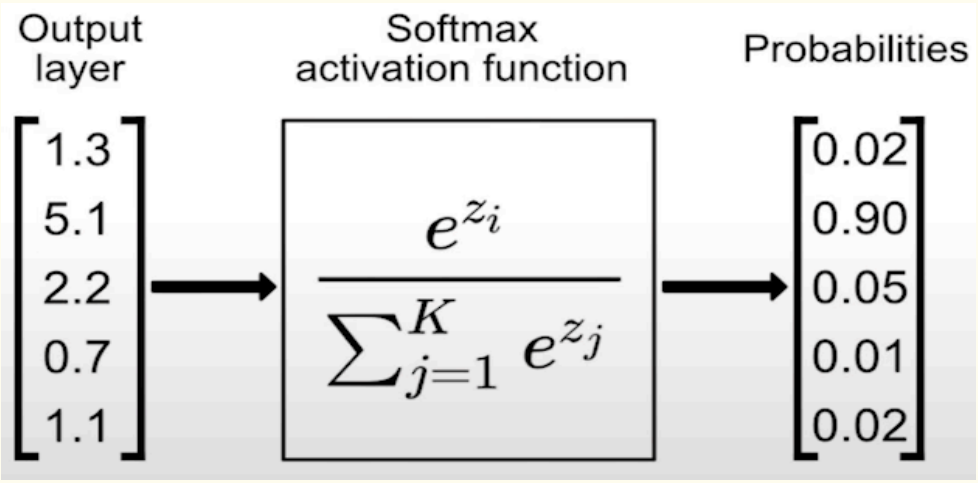
$$Z^{[1]} = W^{[2]} A^{[1]} + b^{[2]}$$

$10 \times m \quad 10 \times 10 \quad 10 \times m \quad 10 \times 1 \Rightarrow 10 \times m$

↓

Activated 1st Layer

Softmax Activation function :- It is applied to O/P layer, each 10 neurons corresponds to each of 10 digits, there must be probability for each digit b/w 0 & 1 with 1 absolute certainty & 0 no chance at all.



$\frac{e^{\text{that node}}}{\text{sum of } e^{\text{all nodes}}}$

O/P is b/w 0-1

B) Back Propagation:-

To get good weights & biases to make better predictions we use Back Propagation.

→ We go in the opposite way,

Prediction → Find out by how much prediction got deviated by actual o/p or label → This gives us loss

How much each of these weights & biases contributed to that error → Adjust weights & biases

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

$dZ^{[2]}$: $10 \times m$; $A^{[2]}$: $10 \times m$; Y: $10 \times m$

$A^{[2]}$: Probabilities of all digits

Y: One hot Encoding of digits

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

$dW^{[2]}$: 10×10; $dZ^{[2]}$: $10 \times m$; $A^{[1]T}$: $m \times 10$

$$dB^{[2]} = \frac{1}{m} \sum dZ^{[2]}$$

$dB^{[2]}$: 10×1; $\sum dZ^{[2]}$: 10×1

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} .* g'(Z^{[1]})$$

$dZ^{[1]}$: $10 \times m$; $W^{[2]T}$: 10×10; $dZ^{[2]}$: $10 \times m$; $g'(Z^{[1]})$: $10 \times m$

g' = Derivative of activation func...

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

10×784 $\quad$ $10 \times m$ $\quad$ $m \times 784$

$$dB^{[1]} = \frac{1}{m} \sum dZ^{[1]}$$

10×1 $\quad$ 10×1

c) Update the weights

$$W^{[1]} = W^{[1]} - \alpha dW^{[1]}$$
$$b^{[1]} = b^{[1]} - \alpha db^{[1]}$$
$$W^{[2]} = W^{[2]} - \alpha dW^{[2]}$$
$$b^{[2]} = b^{[2]} - \alpha db^{[2]}$$

α = learning rate, hyperparameter

Cycle of Model