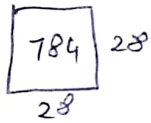
Building a Neural Network using only Maths

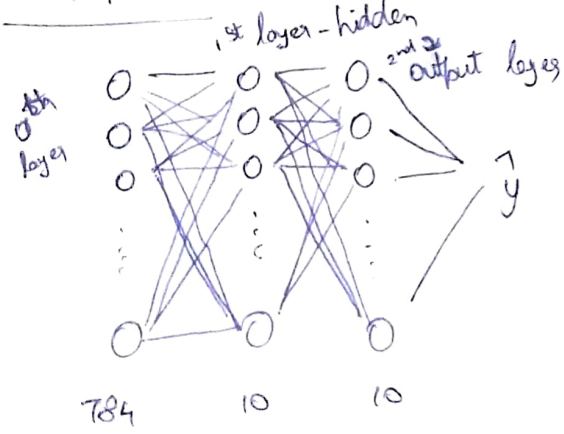
- **MNIST Dataset:**
  - we are going to be using 28x28 Pixels



  - m training images

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

- **Neural Network:**



2 layers

$w \rightarrow$ weight
$b \rightarrow$ bias
$A \rightarrow$ Matrix
$Z^{[1]} \rightarrow$ unactivated first layer
$Z^{[2]} \rightarrow$ " second "

Forward Propagation:

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

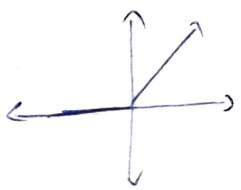
$$Z^{[1]} = w^{[1]} \cdot A^{[0]} + b^{[1]}$$

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

$$Z^{[2]} = w^{[2]} \cdot A^{[1]} + b^{[2]}$$

$$A^{[2]} = \text{Softmax}(Z^{[2]})$$

output $\Rightarrow$ Softmax $\Rightarrow$ Probabilities



$$\Rightarrow \begin{cases} x \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases}$$

Softmax fn. $\Rightarrow \dfrac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Back Propagation:

For optimizing the weights

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

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

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

$$dZ^{[1]} = w^{[2]T} \cdot dZ^{[2]} * g'(Z^{[1]})$$

$$dw^{[1]} = \frac{1}{m} \sum dZ^{[1]} \cdot x^{T}$$

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

$dZ^{[2]}$ -> error of the 2nd layer
(how much the output layer is off from the actual)

Y -> one-hot-encode

g' -> derivative of the activation function

updating:

$$w^{[1]} = w^{[1]} - \alpha\, dw^{[1]}$$

$$w^{[2]} = w^{[2]} - \alpha\, dw^{[2]}$$

$$b^{[1]} = b^{[1]} - \alpha\, db^{[1]}$$

$$b^{[2]} = b^{[2]} - \alpha\, db^{[2]}$$

α -> learning rate