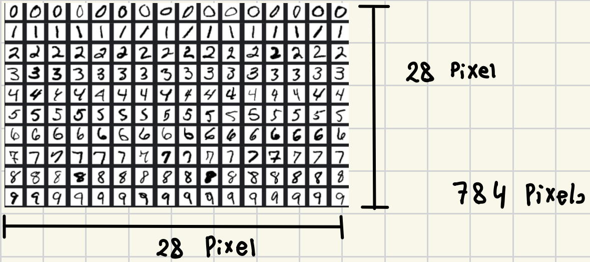
MNIST Newral Network

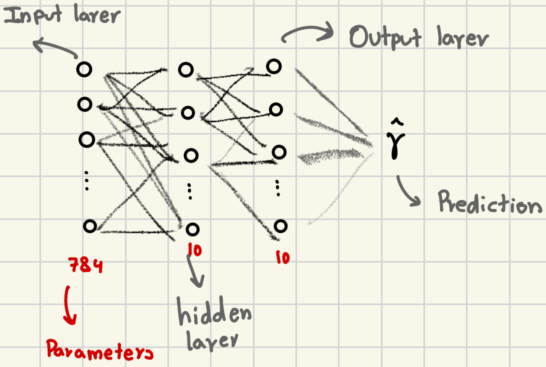


$$X = \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(n)} & — \end{bmatrix}^T = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ | & | & & | \end{bmatrix}$$

Each row is an example, and each row is 784 Colums long each Corresponding to a pixel in the Image

After transposing we have 784 rows Corresponding to each pixel

We will build a newral network to predict a digit from 0 to 9 given an Image



Our Newral Net will have 3 layers

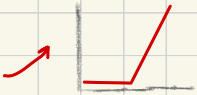
Forward Prop

1. $A^{[0]} = X \quad (784 \times m)$ This Is our Input layer

2. $Z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$

   ↳ Unactivated first layer obtained by multiplying (dot product) of our Input layer and a Weight and adding a bias term (Constant)

3. $A^{[1]} = g(Z^{[1]}) = \text{ReLu}(Z^{[1]})$



ReLu $= \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$

   ↳ In this step we apply an activation function (ReLu) So the layers are not linear combinations of each other.

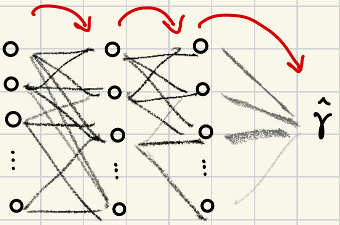
4. $Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$

   ↳ $Z^{[2]}$ Is our Unactivated second layer Is equal to a second weight parameter times our (activated first layer) $A^{[1]}$ plus a bias term

5. $A^{[2]} = \text{SoftMax}(Z^{[2]})$

   ↳ Finaly we use softmax to Convert the output layer to probability

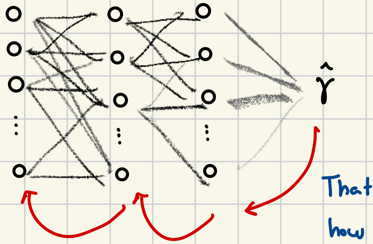
$$\begin{array}{ccccc} \text{Output layer} & & \text{Soft Max} & & \text{Probabilities} \\ \begin{bmatrix}1.3\\5.1\\2.2\\0.7\end{bmatrix} & \rightarrow & \dfrac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} & \rightarrow & \begin{bmatrix}0.02\\0.9\\0.05\\0.01\end{bmatrix} \end{array}$$



So In forward Propagation we navegate through our neural net so we Can get to our Predictor.

Back Prop

We use back Propagation to adjust the weights and biases to make good Predictions.



In this case we will start at the Prediction and we will see by how much did it deviated from the actual label

That will give us an error. Then we will see how much the previous weights and biases Contribute to the error. So we can adjust the weights.

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

So $dz^{[2]}$ Represents the error of the second layer. So its our predictions ($A^{[2]}$) and subtract the actual errors (Y).

↳ This subtraction is made by 1-hot encoding

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

$dw^{[2]}$ is the derivative of the loss function with respect to the weights In layer 2

3. $db^{[2]} = \frac{1}{m} \sum dz^{[2]}$

This is the averedge of the absolut error. this is how much the second layer was off by from the Prediction

4. $dz^{[1]} = W^{[2]T} dz^{[2]} \cdot g'(z)$

Here we take the error from the second layer and apply the Weights In reverse to get to the errors in the first layer. And multiply the derivatie of g to Undo the activation function

5. $\begin{cases} dw^{[1]} = \frac{1}{m} dz^{[1]} X^{T} \\ db^{[1]} = \frac{1}{m} \sum dz^{[1]} \end{cases}$

And we do the same thing that we did with the second layer to calculate how much $W^{[1]}$ and $b^{[1]}$ Contributed to the error in the first layer.

After Finding how much each weight term and each bias term Contributed to the error we Update our parameters

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

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

α is the learning rate

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

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

↳ After our Parameters are updated we go through Forward-Prop again