NNs MATH

NOTE : I am trying to teach myself a NN from scratch using just maths, and deepen my understanding of it from fundamental layer and this is what my basic understanding is, which might be (and I am sure in ways is) wrong, but feel free to use this if it helps

1) *Defining a mathematical Perceptron :*

Let our inputs be $x1,\ x2,\ x3,\ x4, \ldots xn$

And our output be y

Now our Perceptron's value will be :

$$\sum_{i=1}^{n} xi * wi \ + \ bi$$

Then we apply the non linearity to it, we choose ReLu here

$$Let\ f(x) \ = \ Max(0, x) \ \ = \ ReLu(x)$$

After applying non Linearity :

$$ReLu\left(\sum_{i=1}^{n} xi * wi \ + \ bi\right)$$

Why do we need a non linear activation function?

Because the data is complex! And not necessarily always linear, we use the linear functions as legos to build a complex non linear function

2) *A Neural network is just a multi Layer Perceptron* $\left(NNs \ \rightarrow \ MLP\right)$

Where $n(x)$ *represent the number of input layers* *and then we can basically have arbitrary number of layers and depth to it*

any layer that is not an input layer, or an output layer is a hidden layer

The more hidden layers the more Activation functions we can have and basically more legos to make a more complex curve.

Basically

$$Number\ of\ Layers \propto Complexity\ of\ curve$$

Now for each perceptron we have the calculations refrenced in (1)

therefore we represent them using matrices, cause that helps us ease the calculations and visualize them simply

for each perceptron in a layer we calclate zi (*where i is in range* (1, *total perceptrons in layer*))

$$\begin{pmatrix} x1 & & & & & \\ x2 & & & & & \\ x3 & * \ w1 & w2 & w3 & .. & wn \\ .. & & & & & \\ xn & & & & & \end{pmatrix} + \begin{pmatrix} b1 \\ b2 \\ b3 \\ .. \\ bn \end{pmatrix} = zi$$

Then we apply the activation function

$$ReLu(zi)$$

and we repeat this process for each perceptron of each layer until we reach the output nodes

Also a good way to visualize how different ReLus form a complex curve it to think of each ReLu output from previous layer as the input for next Layer and when we finally add them, we basically get the curve using the principle of super position!

The key idea is super position of inputs from previous layers

watch StatQuest's video on activation function to visualize this better

3) *Complexity of a n dimentional curve* :

now for a simplified example lets say we are construting a linear function to determine the price of a house (y)

here $x1$ *is out input say the area of the house*

so we take data points and plot them on a graph

and try to minimize the loss :

the loss is equivalent to :

$$Average\ Loss\ =\ \left(\sum_{i=1}^{n}(yi\ -\ p(xi))^2\right)/\,n\ =\ AL$$

here $p(xi)$ *represents the predicted value on the linear line*

and now we want to minimize the Average Loss function

TO BE CONTINUED....