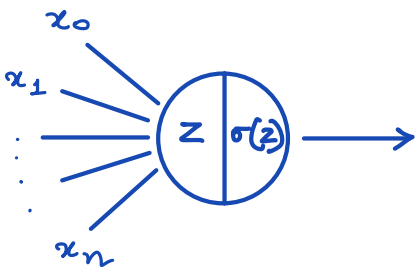
Vanishing Gradient in Backpropagation :

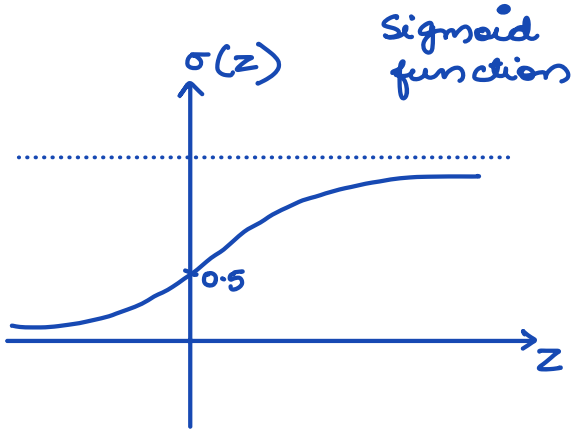
We will learn about Vanishing Gradient problem, how it happens and how you can overcome this problem. This problem arises particularly due to usage of activation function like sigmoid.

PROBLEM:

Lets say you have a neural network and there are some inputs coming in (z) and you apply sigmoid (σ) function and produce some output.



$$Z = \sum_{i=0}^{n} w_i x_i$$

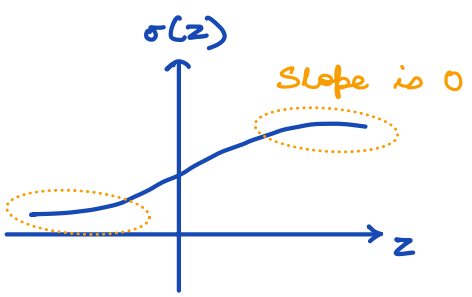


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

If $z = +\infty$ function approaches 1
If $z = -\infty$ function approaches 0
If $z = 0$ function has value 0.5

Problem is due to following reasons:

- If z is very large then slope is 0
- If z is very small then gradient of this function becomes 0



$$\sigma'(z) = \sigma(z)\left[1-\sigma(z)\right]$$

- If z is large then $\sigma(z) = 1$ $\quad z \to \infty \quad \sigma(z) = 1$

$$\sigma'(z) = \sigma(z)\underbrace{[1 - \sigma(z)]}_{\text{becomes } 0}$$

$$\sigma'(z) = 0$$

- If z is small then $\sigma(z) = 0$ $\quad z \to -\infty \quad \sigma(z) = 0$

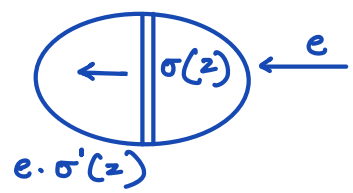
$$\sigma'(z) = \underbrace{\sigma(z)}_{\text{becomes } 0}[1 - \sigma(z)]$$

$$\sigma'(z) = 0$$

Why do we need slope ?

In a multilayer perceptron when you propagate errors back, error is multiplied with derivative of activation function.

$$\delta^l = \left(\delta^{l+1}, w^{l+1^T}\right) \odot \sigma'(z^l)$$



$e \cdot \sigma'(z)$

$e \cdot \sigma(z)(1 - \sigma(z))$

if z is large $\sigma(z) = 1$ $\quad \underbrace{(1 - \sigma(z))}_{0}$

$$\underbrace{\qquad\qquad}_{0}$$

$e \cdot \sigma(z)(1 - \sigma(z))$

if z is small $\sigma(z) = 0$ $\quad \underbrace{\sigma(z)}_{0}$

$$\underbrace{\qquad\qquad}_{0}$$

Eventually the gradient that you are propagating back becomes zero.

This is why we don't use sigmoid function. To avoid this one thing that we generally do is when you are initialising weights you make sure all the weights are small and they should be in range of [-0.5 to 0.5]

$$\sum w_i x_i$$

$$\downarrow$$

small

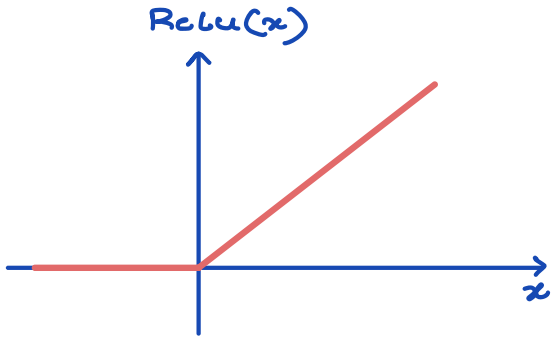
$[-0.5, 0.5]$

You can't have weights in range [2,5] because:

- all of them are positive.
  Some weights should be positive and some negative.
- they will make value of z very large.

To avoid this, people use another activation function called as ReLu.

$$\text{ReLu}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$



ReLu helps because gradient becomes 1 if x is +ve and 0 if x is -ve.

$$\text{ReLu}'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Relu will activate some neurons and deactivate some neurons while back propagating.

Instead of using Sigmoid as activation function you should use Relu.

Sigmoid doesnot work well when you have a deep network, during back propagation it will create trouble.