The challenges with vanishing gradients

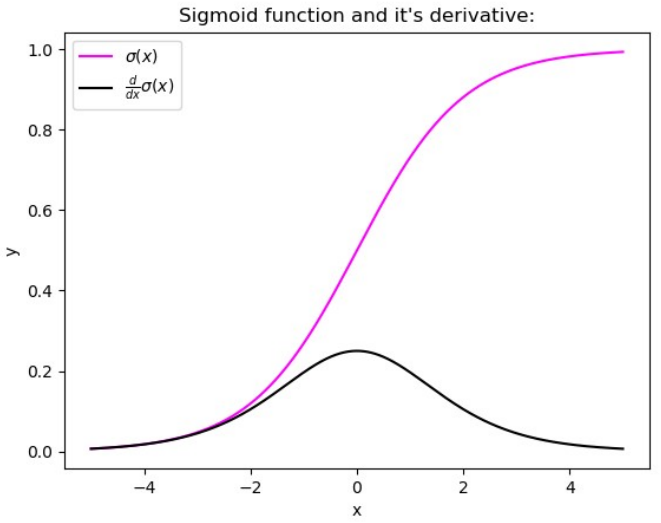
Binabh Devkota
Roll 05

What is vanishing gradients?

- During backpropagation updated weights of nodes in the network depend on the gradients of the activation functions of each node
- Gradients of the weights in the network become very small, making it difficult to update the weights during training
- Problem becomes worse as the number of layers in the architecture increases
- Certain activation functions, like the sigmoid function suffer this issiue

But why this problem?

- Sigmoid function, squishes a large input space into a small input space between 0 and 1
- Large change in the input of the sigmoid function will cause a small change in the output



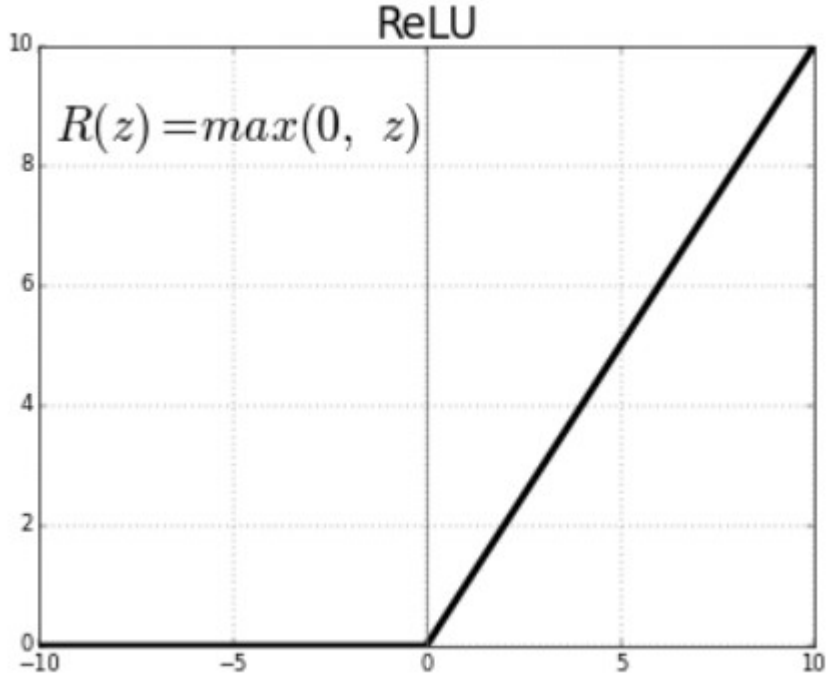
How to identify a vanishing gradient problem?

- The parameters of the higher layers change significantly whereas the parameters of lower layers would not change much (or not at all).
- The gradient may become 0 during training.
- The model learns very slowly and perhaps the training stagnates at a very early stage just after a few iterations.

Solutions? 

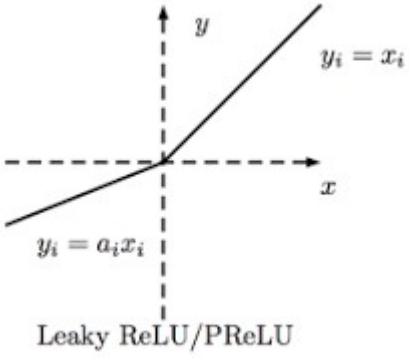
Using Non-saturating Activation Functions

- ReLU
- ReLu function is also not a perfect pick for the intermediate layers
- Problem known as dying ReLus where in some neurons just die out



Using Non-saturating Activation Functions

- Leaky ReLU
- The smaller slope for the leak ensures that the neurons powered by leaky Relu never die



Proper Weight Initialization

- The variance of outputs of each layer should be equal to the variance of its inputs
- The gradients should have equal variance before and after flowing through a layer in the reverse direction
- More on this paper titled Deep Sparse Rectifier Neural Networks

Batch Normalization

- Normalization along with any variant of the ReLU activation function can significantly reduce the chances of vanishing/exploding problems at the beginning
- Does not guarantee that the problem won’t reappear during training

Batch Normalization

- It consists of adding an operation in the model just before or after the activation function of each hidden layer.
- This operation simply zero-centers and normalizes each input, then scales and shifts the result using two new parameter vectors per layer: one for scaling, the other for shifting.
- In other words, the operation lets the model learn the optimal scale and mean of each of the layer’s inputs.
- To zero-center and normalize the inputs, the algorithm needs to estimate each input’s mean and standard deviation.
- It does so by evaluating the mean and standard deviation of the input over the current mini-batch (hence the name “Batch Normalization”).

Thank You! 🙏