Vanishing Gradient and Exploding Gradient Problem

Vanishing gradient and exploding gradient are two common effects associated with training deep neural networks and their **impact is usually stronger the deeper the network.**

As you know, two fundamental operations when training neural networks are Forward-propagation and Back-propagation.

When we carry out Back-propagation, that is, moving backward in the Network to calculate gradients of the loss function with respect to the weights, gradient values tend to decrease or increase dramatically, the further we get back in the network.

This happens in cases when we have activation functions like **Sigmoid, or TanH**, whose non-linear regions below 0 (i.e. x << -5) and over 0 (i.e. x >> 5) return gradient values in the saturation regions. This means (x << -5) that neurons in earlier layers will learn at a very slow pace as compared to those layers located later in the network (vanishing gradients problem).

Exploding gradients are the other side of the coin, i.e. when activation functions are saturated (with x >> 5), gradient values tend to increase dramatically, making updates to weights unstable, thus not being able to converge.

Some possible techniques to try to prevent these problems are, in order of relevance:

- **Use ReLu - like activation functions**: ReLu activation functions keep linearity for regions where sigmoid and TanH are saturated, thus responding better to gradient vanishing / exploding. You can also use different types like Leaky-ReLu, Randomized ReLu, etc.

- **Use Batch Normalization (BN)**: this is another solution you could use in order to make your network more robust against gradient vanishing / exploding, especially if you are using sigmoid or TanH as activation functions. Actually, BN gives you more flexibility during the selection of the activation function for your network. The obtained architecture gets more robust at training, given that it is less prone to diverging due to initialization values or from higher learning rates.

- **Reduce learning rate**: if you increase your learning rate without considering using a ReLu-like activation function and/or not using BN, your network can diverge during training much more easily. By reducing your learning rate you can reduce the chance of suffering vanishing / exploding gradients problem, but your network will take longer to learn. That is why the first two options are located first in the list.

- **Change your architecture**: If you are using Convolutional Neural Networks, for example, and you are suffering from vanishing / exploding gradients, it might make sense to move to a new architecture like ResNETs. In comparison to other networks, these structures connect different layers between each other, i.e. the so-called skip connections, acting as gradient highways, allowing the gradient to flow between the different layers unhindered.

- **Use proper weight initialization**: you could use, for example, Xavier initialization Xavier et al. to reduce the chance of suffering vanishing / exploding gradients. By itself this option does not guarantee you will resolve these issues, but it makes your network more robust when combined with other methods.

- **Gradient clipping**: this can be used when having exploding gradient problems. Firsthand, we select a threshold value, and in case the value returned by the function of a gradient is greater than this threshold, we set it to a different value. You can check more info here.