Course 1 Week 1: General Notes

# What algoritm/function do we use for the neural network?

## There is a good forum post about functions here

<https://www.coursera.org/learn/neural-networks-deep-learning/discussions/weeks/1/threads/OC2Bh8ahEeetfwrt7EZTMA>

### Akhilesh Pandey

· [5 months ago](https://www.coursera.org/learn/neural-networks-deep-learning/discussions/weeks/1/threads/OC2Bh8ahEeetfwrt7EZTMA/replies/g4jsBT5ZEeiKbA458YilXg/comments/-H-vwKqFEeihqhJIzHfuBA)

**Sigmoid activation** suffers from vanishing gradient. This happens when the z=wx+b fall in the saturation region. At saturation point the gradient is close to zero so when propagated backward smaller two smaller numbers are multiplied resulting in much smaller number. By the time the gradient of loss w.r.t parameters, the value becomes very close to zero. When we to the parameter update, we are adding very small amount to the parameters. Hence, it takes longer time for the parameters to reach the optimum point. Therefore, the training is slow with Sigmoid. This is not the case with Relu because the gradient of Relu is gradient of max(0,x) and most of the time its 1.

This is sorta easy to see, the function is very “flat” between some values and the slope actually gets less steep the more the values of x approach -infinition or +infinite. In maths terms we have a“*Vanishing gradient problem”* so for every new learning value in that domain of the function, nothing much happens in the value of y.

A nice sharp slope on the other hand...now we are getting somewhere!

However… like with all functions we must be aware of our domain.

reLU has a slope of 0 for x<=0. That is pretty bad, we don’t have a vanishing gradient, we have none at all!

So, they made some tweaks to fix this. See below for an outline

## An outline of common functions

<https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>

### **Most popular types of Activation functions -**

1. *Sigmoid or Logistic*
2. *Tanh — Hyperbolic tangent*
3. *ReLu -Rectified linear units*

# Week 2

For further reading, but it seems we don’t need it

<https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy>

I think the below is good actually

<http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>

The Google Class that uses a linear classifier. There is also an explanation of weights about 5 minutes in and examples on SciKit visualisation of the weights values.

<https://www.youtube.com/watch?v=Gj0iyo265bc&index=9&list=PLT6elRN3Aer7ncFlaCz8Zz-4B5cnsrOMt>

Backpropagation and Gradient Descent in this excellent maths channel

<https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>

oiv