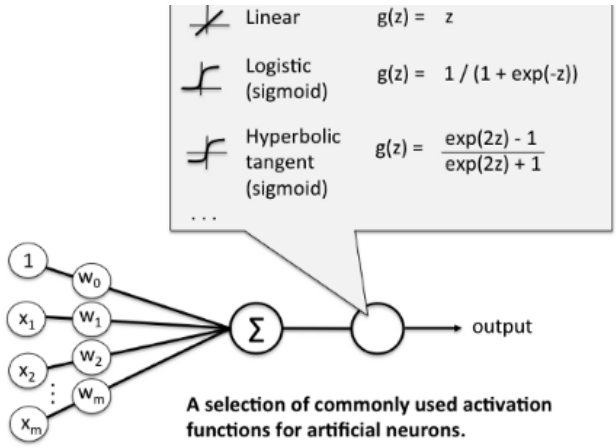


Activation functions and it's types-Which is better?

Anish Singh Walia Follow
May 29, 2017 · 8 min read

What are Artificial Neural Networks ?

Now, I know we all are familiar with what A-NN is but still let me define it for my own satisfaction -It is a very powerful , strong as well as a very complicated Machine Learning technique which mimics a human brain and how it functions.

Like our human brain has millions of neurons in a hierarchy and Network of **neurons** which are interconnected with each other via **Axons** and passes Electrical signals from one layer to another called **synapses.** This is how we humans learn things. Whenever we see, hear,feel and think something a **synapse(electrical impulse)** is fired from one neuron to another in the hierarchy which enables us to learn , remember and memorize things in our daily life since the day we were born.

Okay , now let's not get into Biology.

SIDE NOTE

> *If you don't know about how to implement deep learning or make deep neural networks, I would really suggest 2 of my favorite courses and insist to complete this amazing course by* ***DataCamp*** *on* ***Deep learning in***

> ***Python using Keras*** *or you can try out with this new course on* ***CNN for image procesing using Keras****. Both of these are amazing and everyone should try these out ,all who are new to Deep learning.*

> *If you want to learn building data products and focused more on application side then* ***Building chatbots in Python*** *or* ***NLP fundamentals in Python using NLTK*** *are the courses to try out.These course teaches you about how to implement deep learning in python and the fundamentals of deep learning from basics of neural networks to fine tuning and optimizing a deep neural network.*

> *Or, if you want to first get clear with the basics of machine learning then these courses on* ***Pre-processing for Machine learning in python*** *,* ***Supervised learning in Python using Scikit-learn*** *,* ***Python for Data science*** *are the best choices for you .All these courses will teach you the basics very nicely and properly. I started learning python from these 3 courses. So python lovers can start with these courses based on their interest.*

> *For the R lovers I am also adding the links to my favorite* ***machine learning*** *and* ***statistics*** *courses in R called* ***Machine learning Toolbox*** and ***Statistical Modelling in R*** *,* ***Foundations of Probability theory in R****. Knowledge of* ***statistics*** *and* ***probability theory*** *is a must and pre-requisites for both Machine learning and Deep learning.*

So the above courses and resources mentioned are some of my personal picks and my personal favorites. Best part is that not only they help you grab and learn the conceptual and core under the hood mathematical things but also help you to practically learn to implement them and get your hands dirty using the tool of your choice. ***This technique of learning subjects related to data science is the best*****. So make sure to give them a try on the basis of your topic of interest. I am sure you will love it and be more productive.** ***These are the courses which have helped me a lot to be what I am today*****. So just wanted to share it with the audience and the readers. Happy learning!.**

What are Activation functions and what are it uses in a Neural Network Model?

Activation** functions* are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.*They introduce non-linear properties to our Network*.Their main purpose is to convert a input signal of a node in a A-NN to an output signal.*** That output signal now is used as a input in the next layer in the stack.

Specifically in A-NN we do the sum of products of inputs(**X**) and their corresponding Weights(**W**) and apply a Activation function **f(x)** to it to get the output of that layer and feed it as an input to the next layer.

The question arises that why can't we do it without activating the input signal?

If we do not apply a Activation function then the output signal would simply be a simple ***linear function***.A *linear function* is just a polynomial of **one degree.** Now, a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. A Neural Network without Activation function would simply be a **Linear regression Model,** which has limited power and does not performs good most of the times. We want our Neural Network to not just learn and compute a linear function but something more complicated than that. *Also without activation function our Neural network would not be able to learn and model other complicated kinds of data such as images, videos , audio , speech etc. That is why we use Artificial Neural network techniques such as* ***Deep learning to make sense of something complicated ,high dimensional,non-linear -big datasets, where the model has lots and lots of hidden layers in between and has a very complicated architecture which helps us to make sense and extract knowledge form such complicated big datasets.***

So why do we need Non-Linearities?

Non-linear functions are those which have degree more than one and they have a curvature when we plot a Non-Linear function. Now we need a Neural Network Model to learn and represent almost anything and any arbitrary complex function which maps inputs to outputs. Neural-Networks are considered ***Universal Function Approximators***. *It means that they can compute and learn any function at all*. Almost any process we can think of can be represented as a functional computation in Neural Networks.

Hence it all comes down to this, we need to apply a Activation function f(x) so as to make the network more powerfull and add ability to it to learn something complex and complicated form data and represent non-linear complex arbitrary functional mappings between inputs and outputs. Hence using a non linear Activation we are able to generate non-linear mappings from inputs to outputs.

Also another important feature of a Activation function is that it should be differentiable. We need it to be this way so as to perform backpropogation optimization strategy while propogating backwards in the network to compute gradients of Error(loss) with respect to Weights and then accordingly optimize weights using Gradient descend or any other Optimization technique to reduce Error.

Just always remember to do :

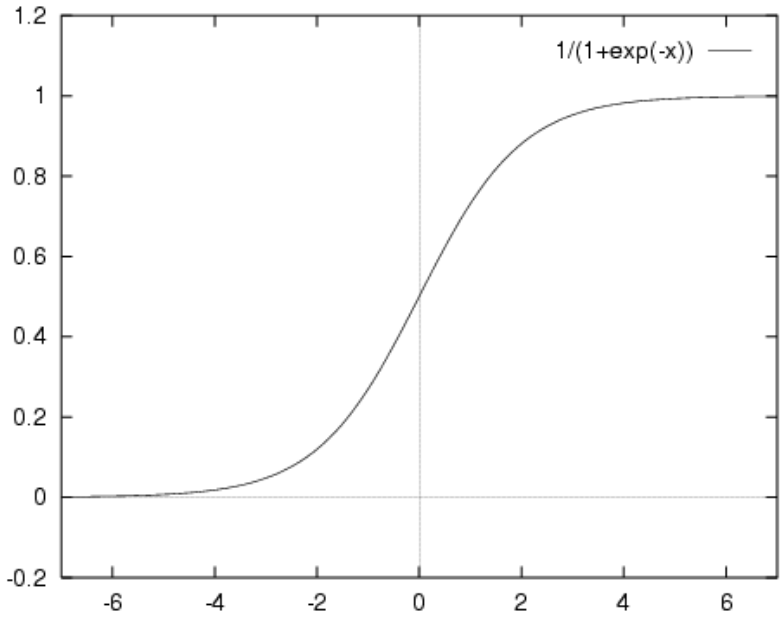
> ***"Input times weights , add Bias and Activate"***

Most popular types of Activation functions -

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

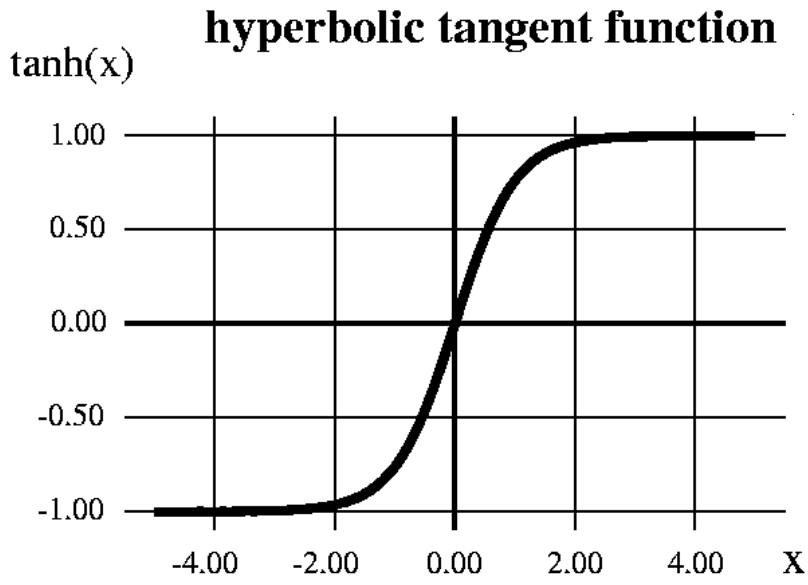
Sigmoid Activation function: It is a activation function of form **f(x) = 1 / 1 + exp(-x) .** Its Range is between 0 and 1. It is a S—shaped curve. It is easy to understand and apply but it has major reasons which have made it fall out of popularity -

- Vanishing gradient problem
- Secondly , its output isn't zero centered. It makes the gradient updates go too far in different directions. **0 < output < 1, and it makes optimization harder.**
- Sigmoids saturate and kill gradients.
- Sigmoids have slow convergence.



Now how do we solve the above problems ?

Hyperbolic Tangent function- Tanh : It's mathamatical formula is **f(x) = 1—exp(-2x) / 1 + exp(-2x).** Now it's output is zero centered because its range in between -1 to 1 i.e -1 < output < 1 . Hence optimization is *easier* in this method hence in practice it is always preferred over Sigmoid function . *But still it suffers from Vanishing gradient problem.*



Then how do we deal and rectify the vanishing gradient problem ?

ReLu- Rectified Linear units : It has become very popular in the past couple of years. It was recently proved that it had *6 times improvement*

in convergence from Tanh function. It's just *R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x. Hence as seeing the mathamatical form of this function we can see that it is very simple and efficinent . A lot of times in Machine learning and computer science we notice that most simple and consistent techniques and methods are only preferred and are best.* Hence it avoids and rectifies **vanishing gradient** problem . Almost all deep learning Models use **ReLu** nowadays.

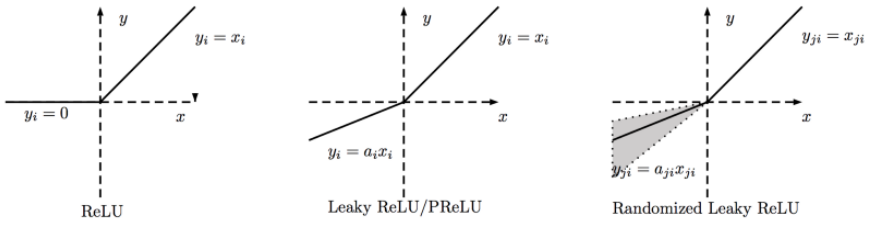
But its limitation is that it should only be used within Hidden layers of a Neural Network Model.

Hence for output layers we should use a **Softmax** function for a Classification problem to compute the probabilites for the classes , and for a regression problem it should simply use a **linear** function.

Another problem with ReLu is that some gradients can be fragile during training and can die. It can cause a weight update which will makes it never activate on any data point again. Simply saying that ReLu could result in Dead Neurons.

To fix this problem another modification was introduced called ***Leaky ReLu*** to fix the problem of dying neurons. It introduces a small slope to keep the updates alive.

We then have another variant made form both ReLu and Leaky ReLu called **Maxout** function **.**



Enough theory right? , then why not go and compare the different activation functions and their performance yourself. Pick up a simple dataset and implement deep learning on it and try different activation function at various times. You will see the difference yourself.

And for all the R lovers if you want to implement deep learning in R you can visit my ***github repository*** on making a simple *digit recognizer on MNIST dataset* and use it as a reference to make deep learning models in R using *keras* package for R.

Conclusion

The question was which one is better to use ?

Answer to this question is that nowadays we should use **ReLu** which should only be applied to the hidden layers. And if your model suffers form dead neurons during training we should use **leaky ReLu** or **Maxout** function.

It's just that *Sigmoid and Tanh* should not be used nowadays due to the **vanishing Gradient Problem** which causes a lots of problems to train,degrades the accuracy and performance of a **deep Neural Network Model.**