International Journal of Computer Architecture and Mobility (ISSN 2319-9229)
Volume 7 -Issue 8, August 2019

Performance Analysis of Various Activation Function

Mohit Kaothekar
mkaothekar@gmail.com

Nachiketa Rajput
nachiketarajput100@gmail.com

Khyati Mehta
1107khyati@gmail.com

Abstract

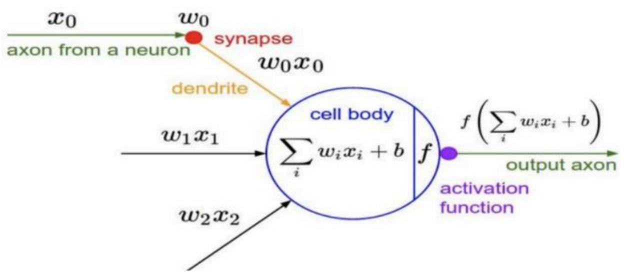
Artificial neural network consist of 3 types of layers, an Input layer, an Output layer and zero or many Hidden layers. All these layers consist of one or many Nodes called Neurons. Hidden layer and output layer neurons use a special function for processing the input provided by previous layer called Activation Function. There are various activation functions which can be used according to the problem, all activation functions have some advantages and some disadvantages. Sometimes it is difficult to choose which activation function to use while designing a neural network. This paper provides comparisons, advantages and disadvantages of many of the popular activation functions which can be used. As a result, selecting an activation functions becomes easy and time saving.

Introduction

Artificial neural network are the simplified model of biological neural network present in our brain. One of the most attractive properties of ANNs is the feasibility to adapt their behaviour to the changing characteristics. ANN is based on a collection of connected neurons and each connection, like the synapsis in a biological brain can transmit a signal to other neurons connected to it.Many researchers have investigated a variety of methods to improve ANN performance by optimizing training methods, learn parameters, network structure and comparably few work is done towards activation function.

In artificial neural network, the activation function of a node defines the output of that node given an input or set of inputs. The objective of an Activation Function is to introduce non-linearity into the network. Only non-linear activation functions are used in ANN because the purpose of the activation function is to introduce non-linearity into the network. Activation functions cannot be linear because neural networks with a linear activation function are effective only one layer deep, regardless of how complex their architecture is. Input to networks is usually linear transformation (input * weight), but real world problems are non-linear. To make the incoming data nonlinear, we use nonlinear mapping called activation function. An activation function is a decision making function that determines the presence of a particular neural feature. It is mapped between 0 and 1, where zero means absence of the feature, while on means its presence. Unfortunately, the small changes occurring in the weights cannot be reflected in the activation values differentiable between this ranges. A neural network must be able to take any input from –infinity to +infinity, but it should be able to amp it an output that ranges between {0, 1} or between {-1, 1} in some cases – thus the need for activation function. Non-linearity is needed in activation functions because its aim in a neural network is to produce nonlinear decision boundary via nonlinear combinations of the weights and inputs.

Another way to think of it: without a non-linear activation function in network, a neural network, no matter how many layers it had, would behave just like a single layer perceptron, because summing these layers would give you just another linear function.

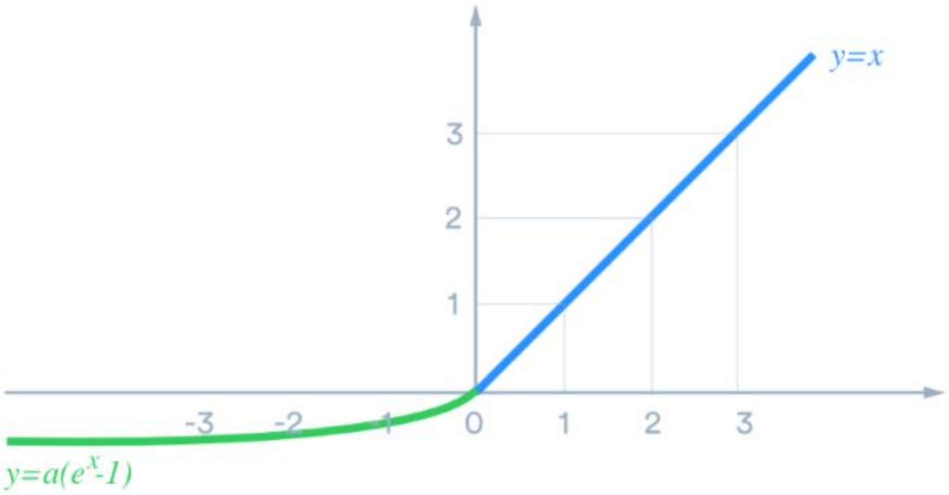


Activation Function Types

Some of the most common activation function are used to solve non-linear problems. These functions are: ELU, ReLU, Leaky ReLU, Uni-polar Sigmoid, Bipolar Sigmoid, Hyperbolic Tangent, and Softmax.

- **ELU**: Exponential Linear Unit is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has an extra alpha constant which should be positive number. It is very much similar to ReLU except negative inputs they are both in identity function from for non-negative inputs. On the other hand, ELU becomes smooth slowly until its output equal to –α whereas ReLU sharply smoothes.

$$f(x) = \begin{cases} a(e^{-x} - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$$

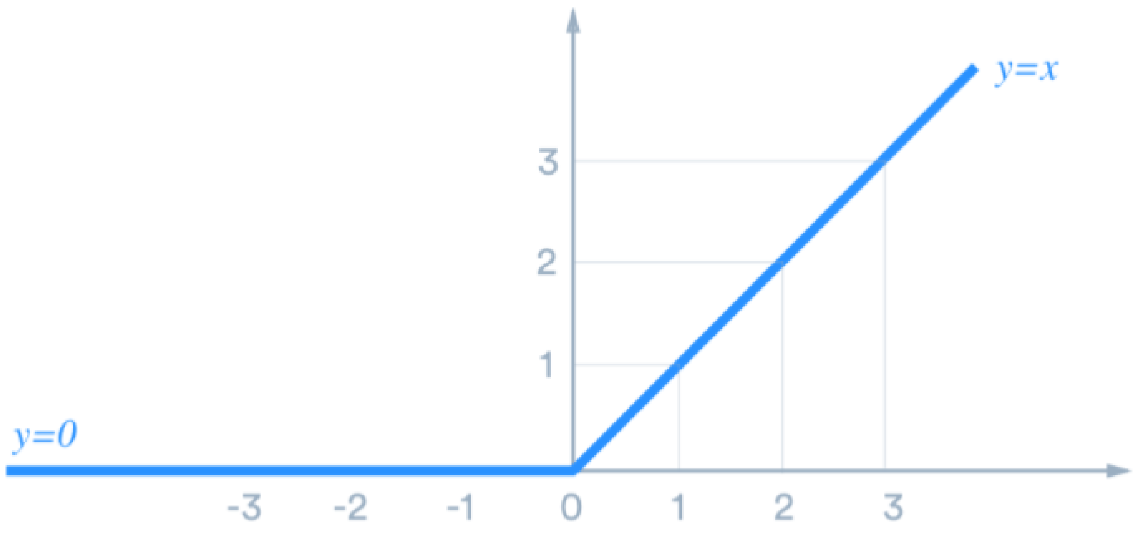


Advantages:
  - ELU becomes smooth slowly until its output equal to –α whereas ReLU sharply smoothes.
  - ELU is a strong alternative to ReLU.
  - Unlike to ReLU, ELU can produce negative outputs.

Disadvantages:
  - For x>0, it can blow up the activation with the output range of [0, ∞].

- **ReLU**: 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 mathematical form of this function we can see that it is very simple and efficient. 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.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



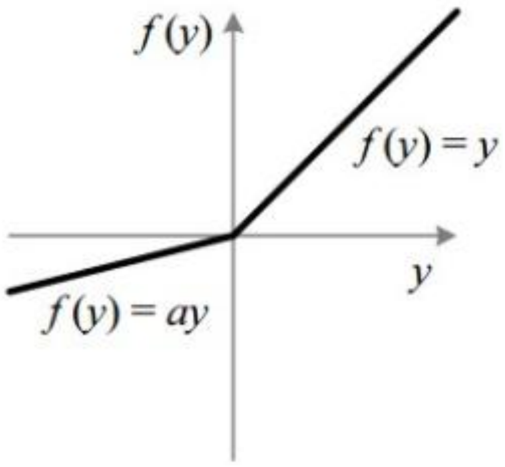
Advantages:

- It avoids and rectifies vanishing gradient problem.
- ReLU is less computationally expensive than Tanh and sigmoid because it involves simpler mathematical operations.

Disadvantages:

- One of its limitation is that it should only be used within Hidden layers of a Neural Network Model.
- 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.
- In another words, For activations in the region (x<0) of ReLU, gradient will be 0 because of which the weights will not get adjusted during descent. That means, those neurons which go into that state will stop responding to variations in error/ input (simply because gradient is 0, nothing changes). This is called dying ReLU problem.
- The range of ReLU is [0, ∞). This means it can blow up the activation.

- **Leaky ReLU**: Leaky ReLU is a variant of ReLU. Leaky ReLU has a small slope for negative values, instead of altogether zero. For example, leaky ReLU may have y=0.01x when x>0.



Advantage:

- Leaky ReLUs are one attempt to fix the "dying ReLU" problem by having a small negative slope (of 0.01, or so).

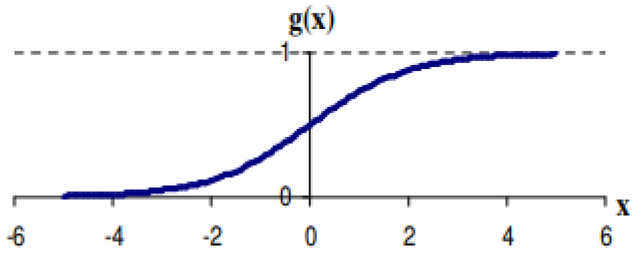
Disadvantage:

- As it possess linearity, it can't be used for the complex Classification. It lags behind the Sigmoid and Tanh for some of the use cases.

➢ **Uni-polar Sigmoid**: Activation function of Uni-polar sigmoid function is given as follows:

$$g(x) = \frac{1}{1 + e^{-x}}$$

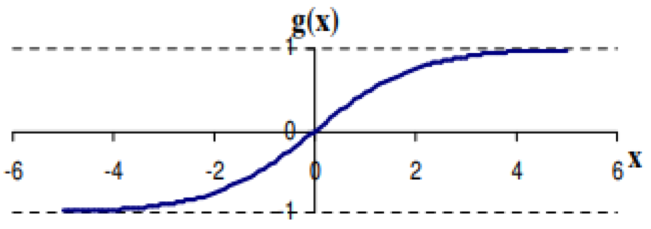
This function is especially advantageous to use in neural network trained by back-propagation algorithms. Because it is easy to distinguish, and this can interestingly minimize the computation capacity for training the term sigmoid means S-shaped, and logistic form the sigmoid maps the interval (-∞, ∞) onto (0, 1).



➢ **Bipolar Sigmoid**: Activation function of Bi-polar sigmoid function is given by:

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

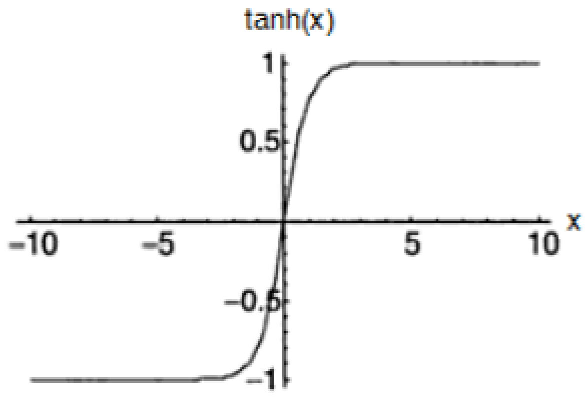
This function is similar to the sigmoid function. For this type of activation function described in below figure, it goes well for applications that produce output values in the range of [-1, 1].



➢ **Hyperbolic Tangent**: This function is easily defined as the ratio between the hyperbolic sine and the cosine functions or expanded as the ratio of the half-difference and the half-sum of two exponential functions in the points x and –x as follows:

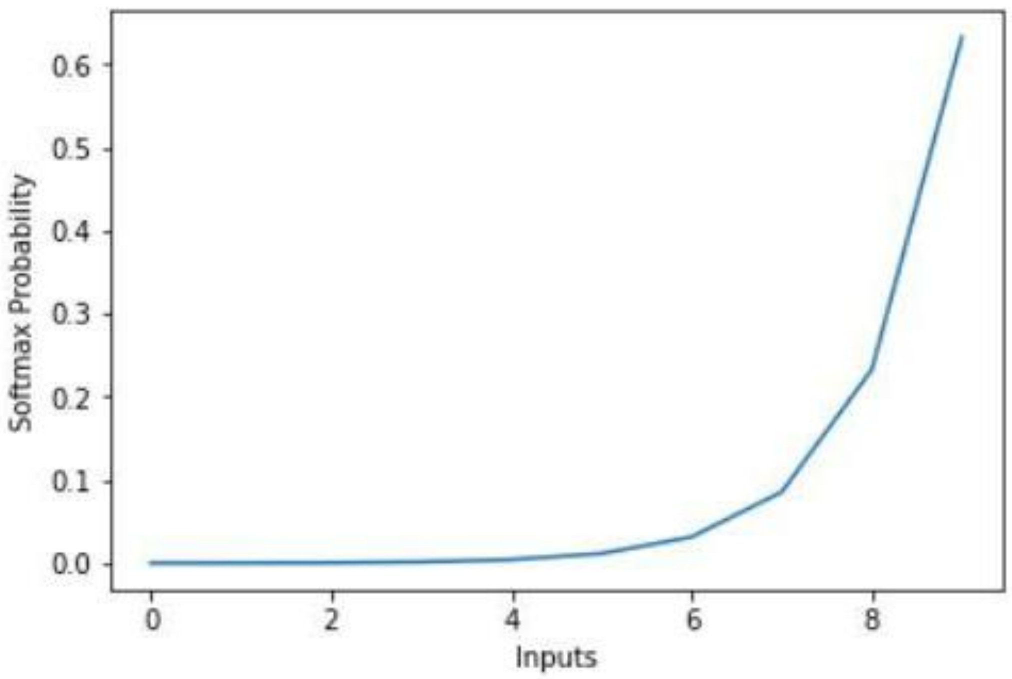
$$tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Hyperbolic tangent function is similar to sigmoid function. Its range outputs between -1 and 1 as seen in below figure. The following is a graph of the hyperbolic tangent function for real values of its argument x:



➢ **Softmax Function**: Softmax function calculates the probability distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{For j=1… K.}$$



Conclusion: Choosing the right Activation Function

Now that we have seen so many activation functions, we need some logic / heuristics to know which activation function should be used in which situation. Good or bad – there is no rule of thumb.

However depending upon the properties of the problem we might be able to make a better choice for easy and quicker convergence of the network.

- Sigmoid functions and their combinations generally work better in the case of classifiers

- Sigmoid and Tanh functions are sometimes avoided due to the vanishing gradient problem
- ReLU function is a general activation function and is used in most cases these days
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice
- Always keep in mind that ReLU function should only be used in the hidden layers
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn't provide with optimum results

References:

1. Hinkelmann, Knut. "Neural Networks, p. 7" (PDF). University of Applied Science Northwestern Switzerland.
2. Hodgkin, A. L.; Huxley, A. F. (1952-08-28). "A quantitative description of membrane current and its application to conduction and excitation in nerve". The Journal of Physiology. **117** (4): 500–544. doi:10.1113/jphysiol.1952.sp004764. PMC 1392413. PMID 12991237.
3. Wuraola, Adedamola; Patel, Nitish (2018), "Computationally Efficient Radial Basis Function", 2018 International Conference on Neural Information Processing (ICONIP), Siem reap Cambodia: Springer, pp. 103–112
4. Haykin, Simon S. (1999). Neural Networks: A Comprehensive Foundation. Prentice Hall. ISBN 978-0-13-273350-2.
5. Cybenko, G.V. (2006). "Approximation by Superpositions of a Sigmoidal function". In van Schuppen, Jan H. (ed.). Mathematics of Control, Signals, and Systems. Springer International. pp. 303–314.
6. Snyman, Jan (3 March 2005). Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer Science & Business Media. ISBN 978-0-387-24348-1.
7. Wu, Huaiqin (2009). "Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions". Information Sciences. **179** (19): 3432–3441. doi:10.1016/j.ins.2009.06.006.