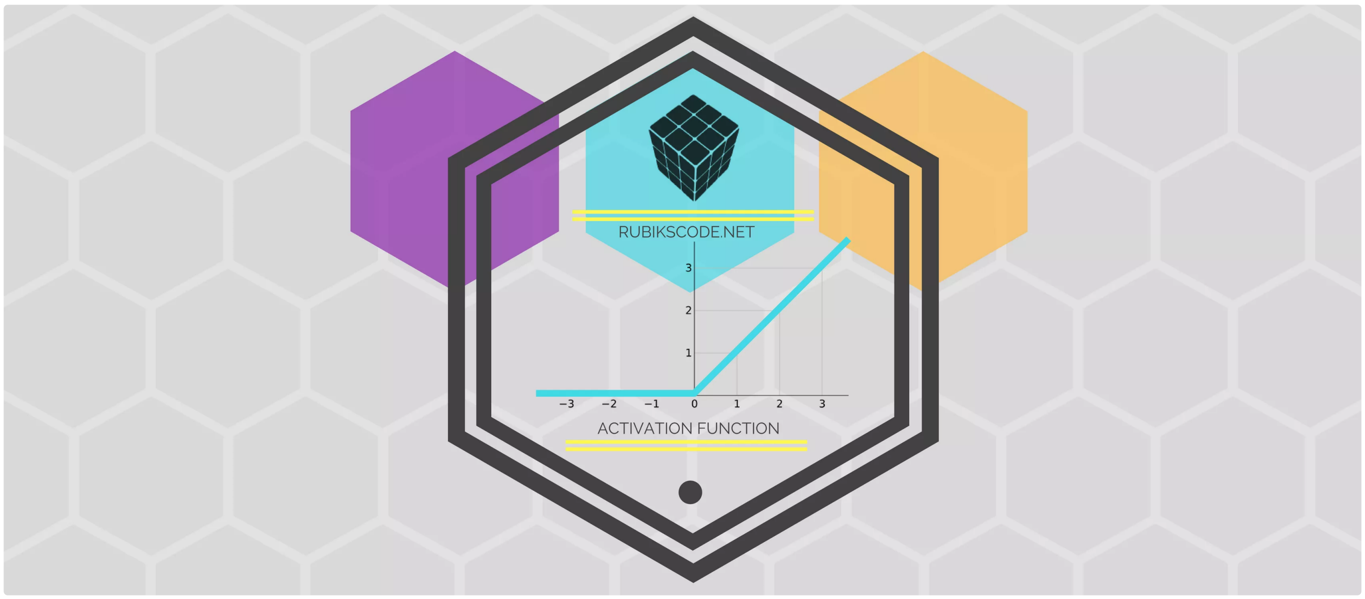
RUBIK'S CODE Freedom. Wisdom. Excellence.

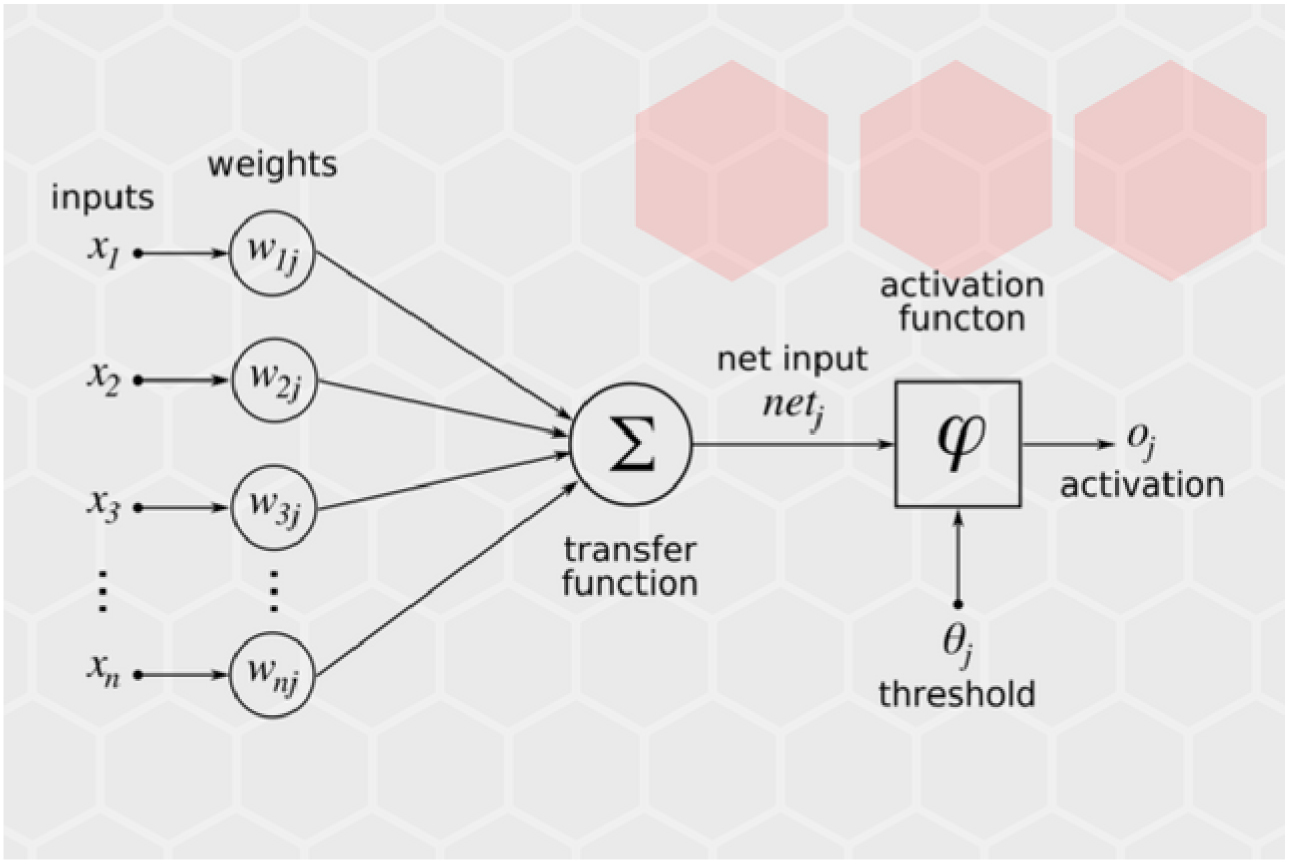
Common Neural Network Activation Functions

NOVEMBER 20, 2017 — 5 COMMENTS



In the previous article, I was talking about what Neural Networks are and how they are trying to imitate biological neural system. Also, the structure of the neuron, smallest building unit of these networks, was presented. Neurons have this simple structure, and one might say that they alone are useless. Nevertheless, when they are connected with each other, they become a powerful tool. Something like a bee and a hive. Bee cannot produce that much honey alone, but thanks to the effort of many bees working together, you have honey in your kitchen.

As a little reminder, let's run through neuron's structure. Basically, they have these connections, which simulate synapses of a biological neuron. Connections are the input of every neuron, and they are weighted, meaning that some connections are more important than others. In a nutshell, neurons are some kind of input collectors. Also, they have an output, which is activated when certain criteria are met. Usually, input from weighted connections is summarized, which is done by so-called *weighted sum* function. This value is then passed to the *activation function,* which calculates the value of the output. You can read more about this in previous article.



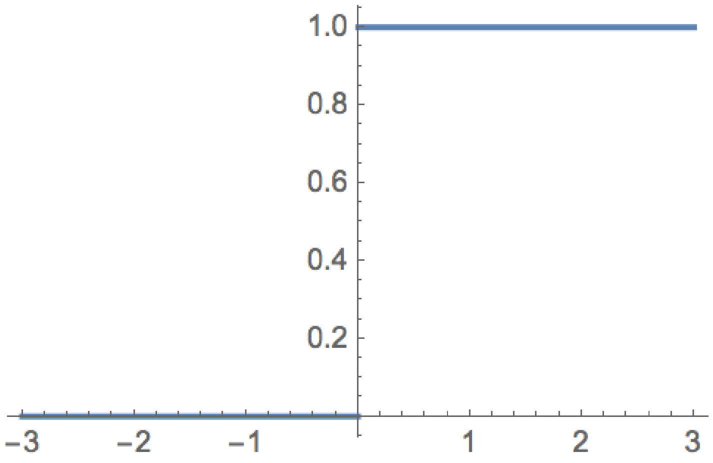
Now, I would like to dive a little bit more into one aspect of an artificial neuron. That aspect is activation function itself, and what are the predominant activation functions out there. Why? Well, activation function is a very important factor in this game. Because, by using the right functions, Neural Networks can compute nontrivial problems by using only a small number of nodes. So, let's dive into it.

Activation Function

As mentioned before, this function will define the output value of our artificial neuron, i.e., is that neuron *"activated"* or not. Biologically speaking, this function can be modeled as the expected firing rate of the total input currently arising out of incoming signals at synapses. Apart from that, this function in global will define how *"smart"* our Neural Network is, and how hard it will be to train it. Since these networks are biologically inspired, one of the first activation functions that was ever used was the step function, also known as the perceptron.

Perceptron

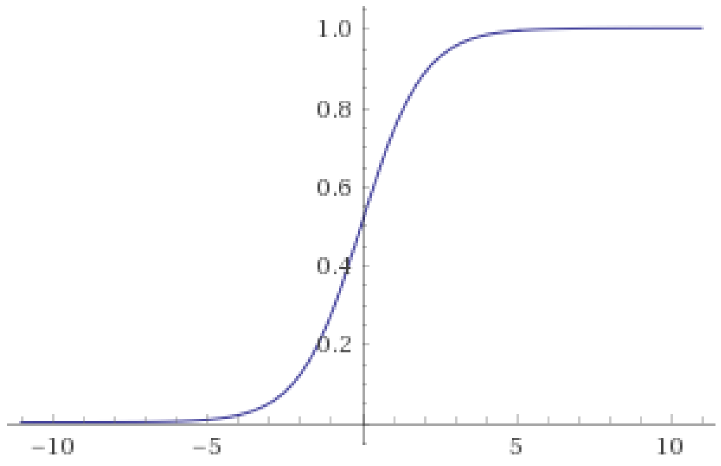
This activation function has an interesting piece of history attached to it. This algorithm was first used back in 1957 in the custom-made computer called *Mark 1 Perceptron,* and was used for image recognition. It was considered the future of artificial intelligence during the first expansion of the field. Even the author of the algorithm – Frank Rosenblatt said that perceptron is "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." Although it seemed promising in the begging, it was proved that perceptron was not able to recognize many classes of patterns. This caused the field of neural networks to stagnate for a while.



In its essence, perceptron is a step function, that maps its real-valued vector input to a single binary output value. So, if the summed value of the input reaches the certain threshold, the value on the neuron's output will be – `1`, otherwise will be – `0`. Very straightforward and especially useful in the last layer of a network.

Sigmoid function

This function is smoother and more biologically plausible than a simple step function. Consider that one neuron gets *fired*, meaning that it starts sending neurotransmitters through the synapses to another neuron. This will happen gradually over time and it will take some time to pass all neurotransmitters. This is, of course, a very simplified description of that scenario. The sigmoid function looks like:



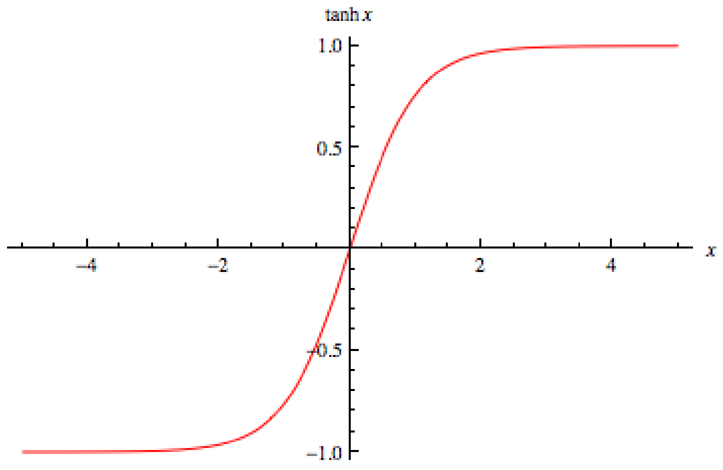
When using this function, we are calculating neuron's output value using the formula:

$$\frac{1}{1+\exp(-x)}$$

where x is *network input* of the neuron. Basically, the weighted input is multiplied by a slope parameter. This function is heavily used for linear regression – one of the most well-known algorithms in statistics and machine learning. This function is also heavily used for the output layer of the neural network, especially for probability calculations.

Hyperbolic Tangent Function

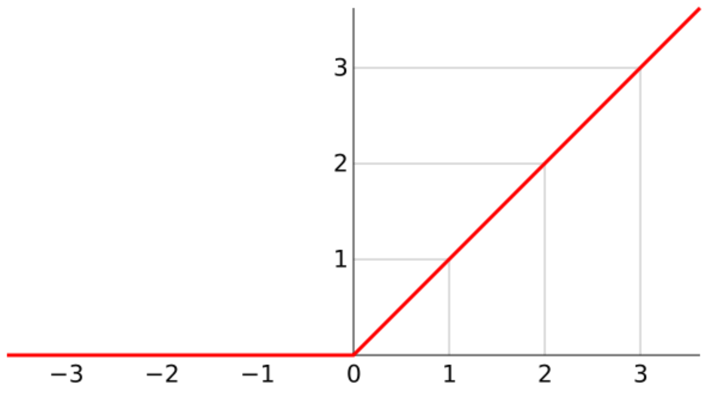
This function is similar to the sigmoid function. Output values of this function can variate from `-1` to `1`, indifference to the sigmoid function which covers values from `0` to `1`. Although this is not what happens in the neurons, biologically wise, this function gives better results when it comes to training neural networks. Sometimes, neural networks get "stuck" during training when the sigmoid function, meaning that when provided with input that is strongly-negative, the output of these networks very near zero. This in turns messes up the learning process, that will be more covered in next posts. Nevertheless, this is how Hyperbolic tangent function looks like:



The formula that we are using is *tahn(x),* where x is *network input* of the neuron. It is used for the same purposes as the sigmoid function, but in networks that have negative inputs. What is interesting to notice here is how this function gives better results in some cases even though it doesn't have strong biological interpretation.

Rectifier Function

Rectifier Function is probably the most popular activation function in the world of neural networks. It is heavily used to solve all kind of problems out there and for a good reason. This function is most biologically plausible of all functions described so far, and the most efficient function when it comes to training neural networks. Here is how it looks like:



The formula that we are using is *max(0, x),* where x is *network input* of the neuron. This is one of the reasons this function is so efficient because they don't require normalization or other complicated calculations.

Conclusion

In this article, we reviewed a few most popular activation functions in neural networks. Note that there are also many other options for activation functions not covered here, that might be the right choice for your specific problem. However, these basic activation functions covered here can be used to solve a majority of the problems one will likely face. Do you have a favorite activation function?

This article is a part of Artificial Neural Networks Series, which you can check out **here**.

Read more posts from the author at **Rubik's Code**.
