Activation Functions

Harshit Gaur

What’s an Activation Function?

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to **introduce non-linearity** into the output of a neuron.

Why non-linearity is required?

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

The Exact Requirement of Activation Function

This observation results again in a linear function even after applying a hidden layer, hence we can conclude that, doesn’t matter how many hidden layer we attach in neural net, all layers will behave same way because ***the composition of two linear function is a linear function itself***. Neuron can not learn with just a linear function attached to it. A non-linear activation function will let it learn as per the difference w.r.t error. **Hence we need activation function**

How to choose what Activation Function to use?

The choice of activation function has a large impact on the capability and performance of the neural network, and different activation functions may be used in different parts of the model.

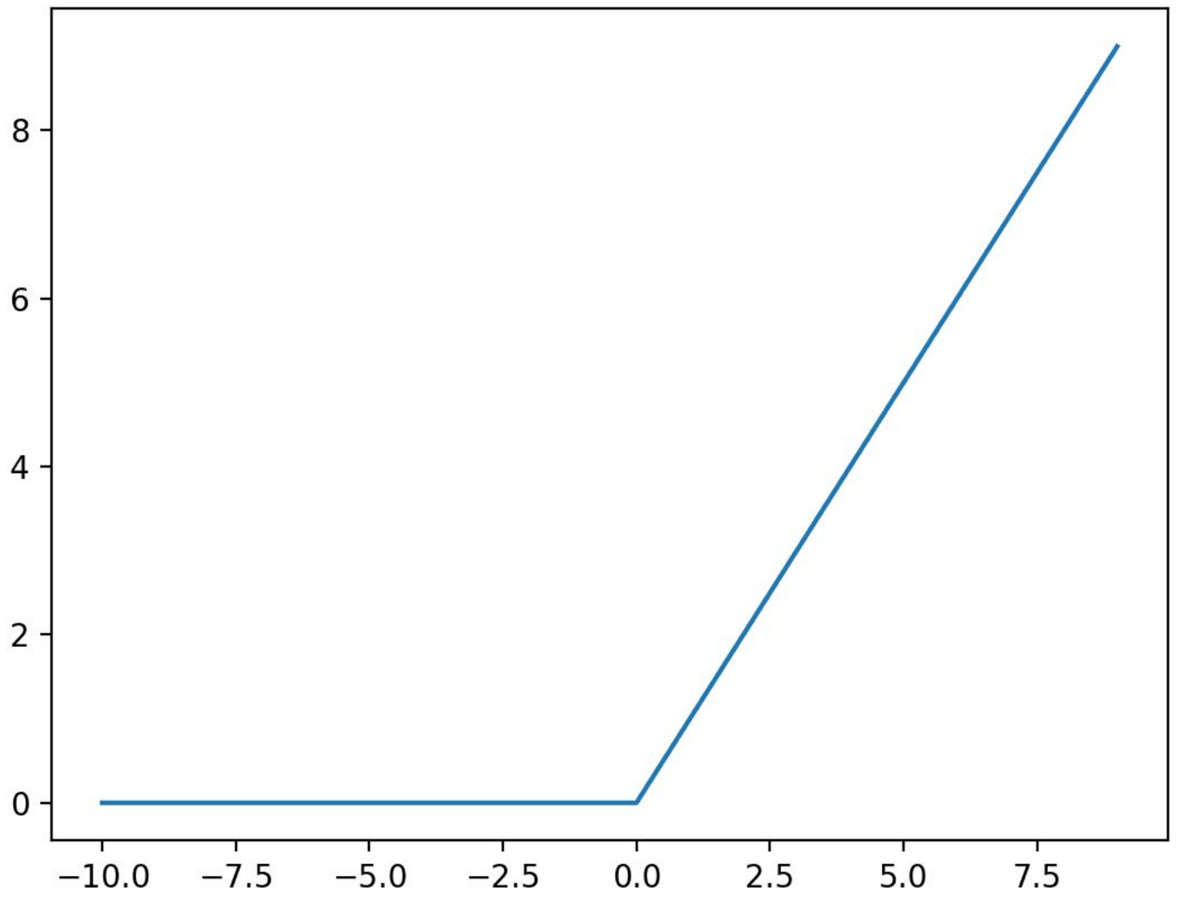
Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.

Activation Functions for Hidden Layers

- Rectified Linear Activation(ReLU)
- Logistic (Sigmoid)
- Hyperbolic Tangent (Tanh)

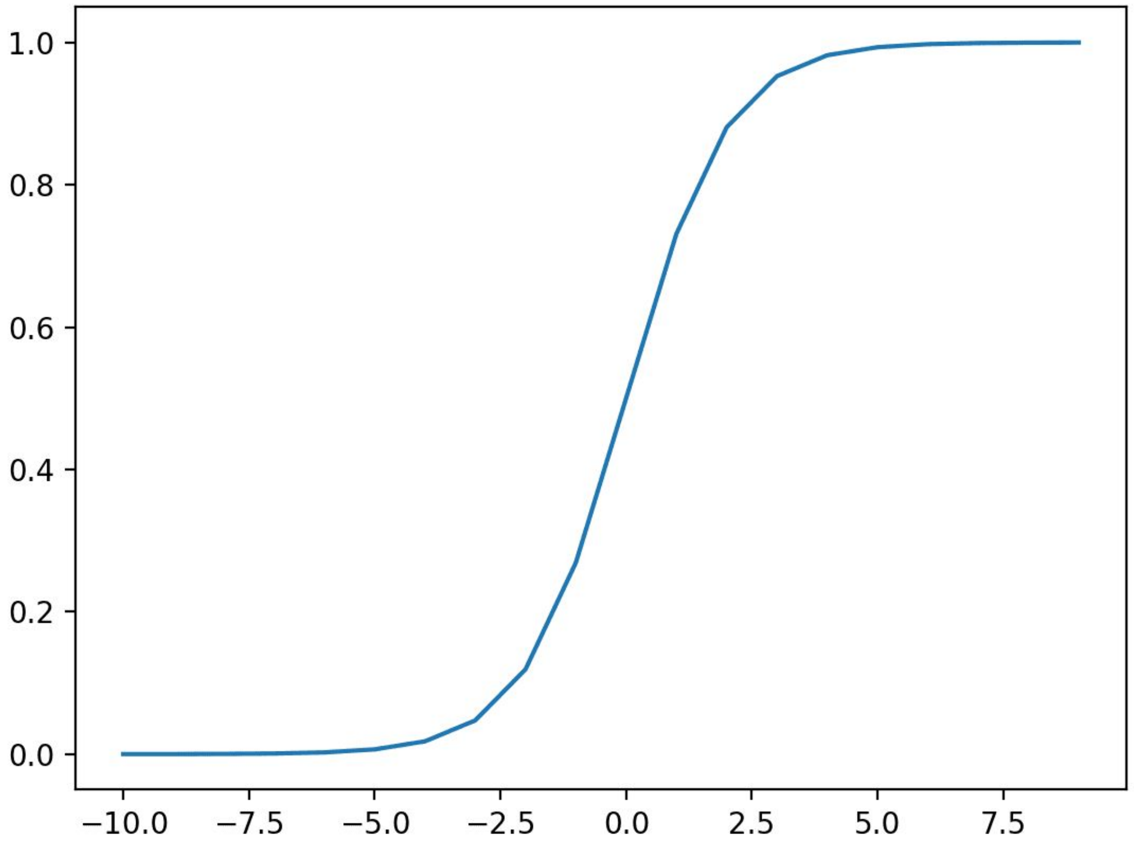
ReLU or Rectified Activation Function

The rectified linear activation function, or ReLU activation function, is perhaps the most common function used for hidden layers. It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or “dead” units. The ReLU function is calculated as follows: **max(0.0, x)** This means that if the input value (x) is negative, then a value 0.0 is returned, otherwise, the value is returned


8
6
4
2
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

Sigmoid or Logistic Function

The sigmoid activation function is also called the logistic function. It is the same function used in the logistic regression classification algorithm. The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0. The sigmoid activation function is calculated as follows: 1.0 / (1.0 + e^-x) Where e is a mathematical constant, which is the base of the natural logarithm.


1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

Tanh Hidden Layer Activation Function

The hyperbolic tangent activation function is also referred to simply as the Tanh (also “*tanh*” and “*TanH*“) function.

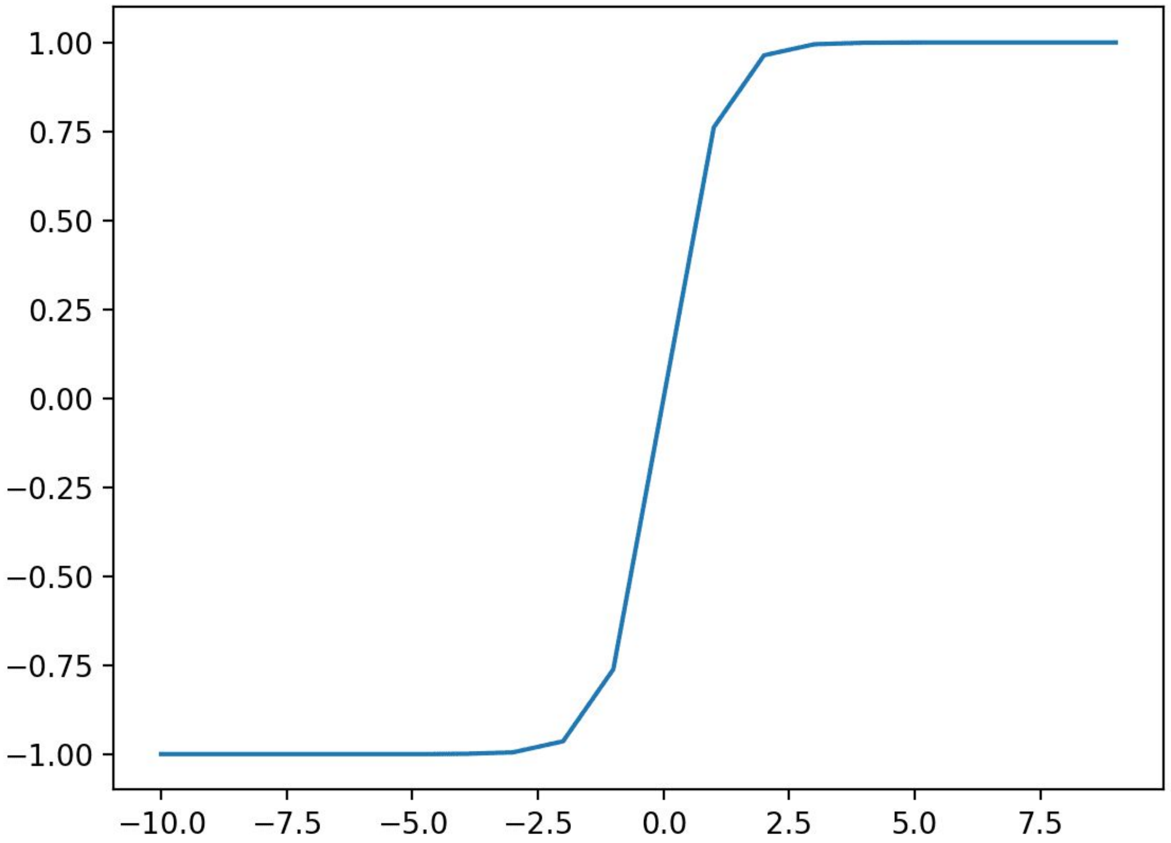
It is very similar to the sigmoid activation function and even has the same S-shape.

The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

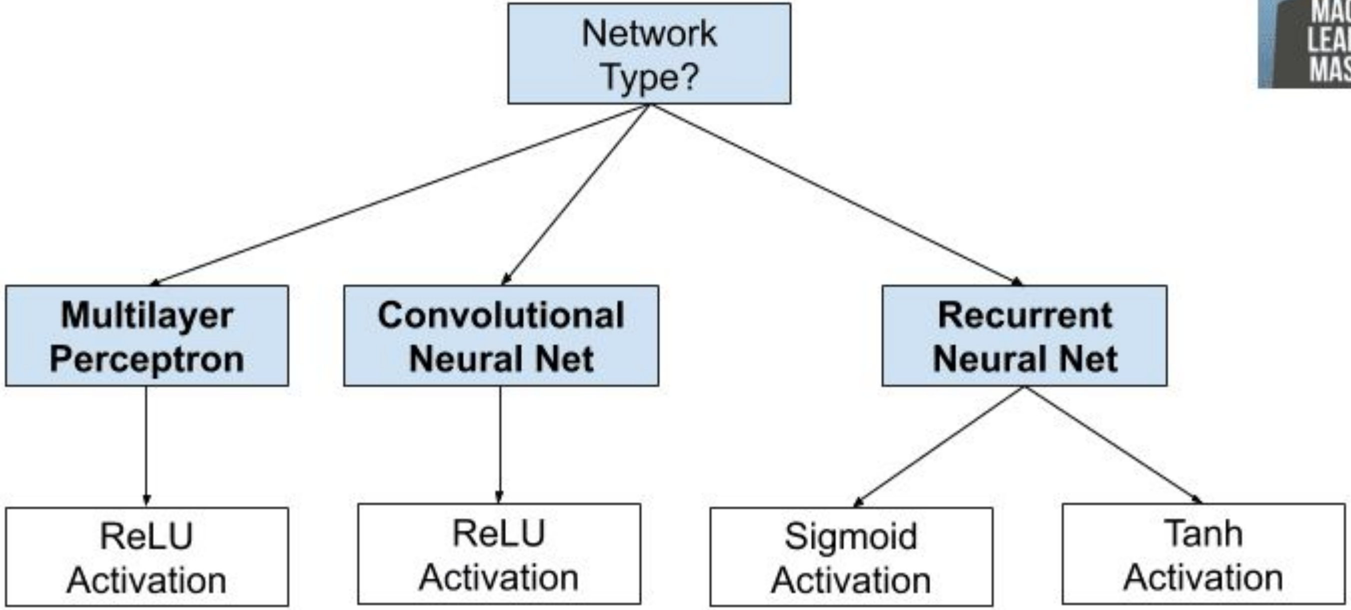
The Tanh activation function is calculated as follows:

- **(e^x – e^-x) / (e^x + e^-x)**

Where e is a mathematical constant that is the base of the natural logarithm


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

How to Choose an Hidden Layer Activation Function



MachineLearningMastery.com

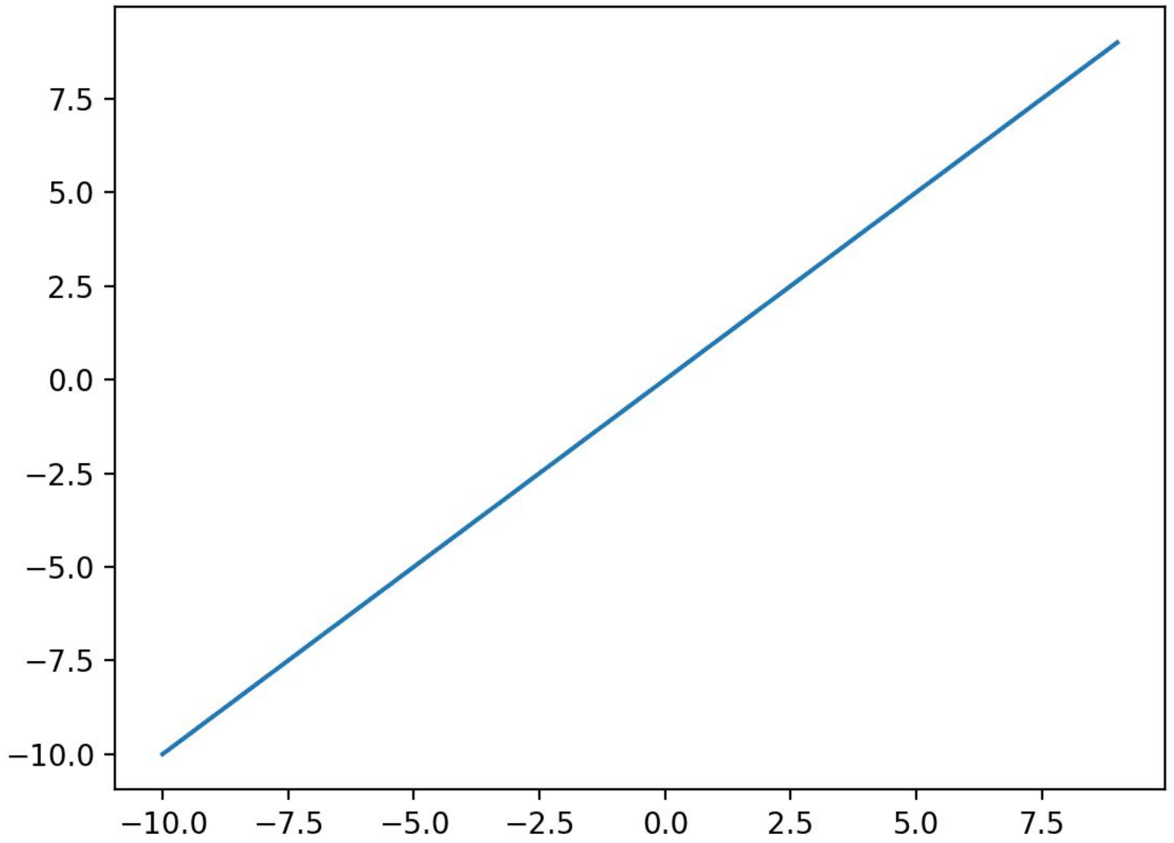
Activation Function for Output Layers

- Linear
- Logistic (Sigmoid)
- Softmax

Linear Function

The linear activation function is also called “*identity*” (multiplied by 1.0) or “*no activation*.”

This is because the linear activation function does not change the weighted sum of the input in any way and instead returns the value directly.


7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

The softmax function outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership.

It is related to the argmax function that outputs a 0 for all options and 1 for the chosen option. Softmax is a "*softer*" version of argmax that allows a probability-like output of a winner-take-all function.

As such, the input to the function is a vector of real values and the output is a vector of the same length with values that sum to 1.0 like probabilities.

The softmax function is calculated as follows:

- **e^x / sum(e^x)**

Where *x* is a vector of outputs and e is a mathematical constant that is the base of the natural logarithm.

- **Binary Classification**: One node, sigmoid activation.
- **Multiclass Classification**: One node per class, softmax activation.
- **Multilabel Classification**: One node per class, sigmoid activation.