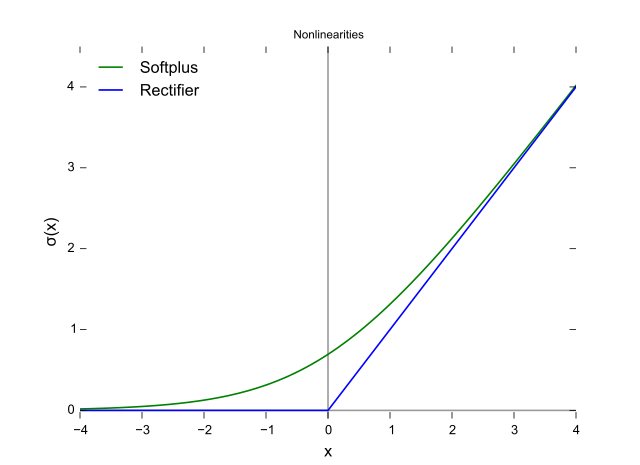
**Linear & Non Linear Models**

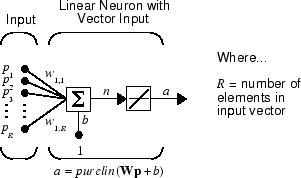
Linearity refers to the property of a system or model where the output is directly proportional to the input, while nonlinearity implies that the relationship between input and output is more complex and cannot be expressed as a simple linear function

A Rectified Linear Unit is a form of activation function used commonly in deep learning models. In essence, the function returns 0 if it receives a negative input, and if it receives a positive value, the function will return back the same positive value.



Linear Classification refers to categorizing a set of data points into a discrete class based on a linear combination of its explanatory variables. Non-Linear Classification refers to categorizing those instances that are not linearly separable. It is possible to classify data with a straight line.

The linear transfer function calculates the neuron's output by simply returning the value passed to it. This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. A linear network cannot, of course, be made to perform a nonlinear computation.



A nonlinear neural network is a neural network that uses nonlinear transformations in its layers, such as activation functions, convolution, or pooling. An activation function is a function that adds nonlinearity to the output of a neuron, such as a sigmoid, tanh, or relu function.

A nonlinear model describes nonlinear relationships in experimental data. Nonlinear regression models are generally assumed to be parametric, where the model is described as a nonlinear equation. Typically machine learning methods are used for non-parametric nonlinear regression.

**Activation functions in Neural Networks**

It is recommended to understand [Neural Networks](https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/) before reading this article.

In the process of building a neural network, one of the choices you get to make is what [Activation Function](https://www.geeksforgeeks.org/activation-functions-neural-networks/) to use in the hidden layer as well as at the output layer of the network. This article discusses some of the choices.

**Elements of a Neural Network**

**Input Layer:**This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.

**Hidden Layer*:***Nodes of this layer are not exposed to the outer world, they are part of the abstraction provided by any neural network. The hidden layer performs all sorts of computation on the features entered through the input layer and transfers the result to the output layer.

**Output Layer:**This layer bring up the information learned by the network to the outer world.

**What is an activation function and why use them?**

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

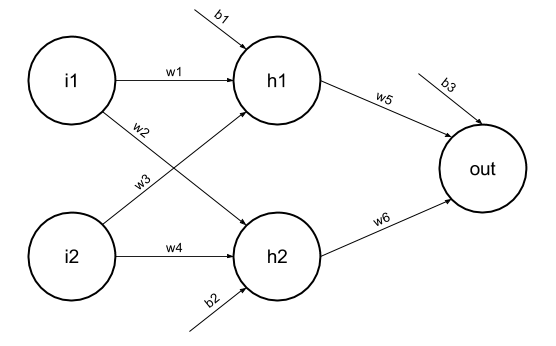
**Explanation:** We know, the neural network has neurons that work in correspondence with *weight, bias,* and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as [***back-propagation***](https://www.geeksforgeeks.org/backpropagation-in-data-mining/). Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

**Why do we need Non-linear activation function?**

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.

**Mathematical proof**

*Suppose we have a Neural net like this :-*



Elements of the diagram are as follows:

**Hidden layer i.e. layer 1:**

*z(1) = W(1)X + b(1) a(1)*

*Here,*

* *z(1) is the vectorized output of layer 1*
* *W(1) be the vectorized weights assigned to neurons of hidden layer i.e. w1, w2, w3 and w4*
* *X be the vectorized input features i.e. i1 and i2*
* *b is the vectorized bias assigned to neurons in hidden layer i.e. b1 and b2*
* *a(1) is the vectorized form of any linear function.*

*(****Note:****We are not considering activation function here)*

**Layer 2 i.e. output layer :-**

***Note :****Input for layer 2 is output from layer 1*

*z(2) = W(2)a(1) + b(2)*

*a(2) = z(2)*

**Calculation at Output layer**

*z(2) = (W(2) \* [W(1)X + b(1)]) + b(2)*

*z(2) = [W(2) \* W(1)] \* X + [W(2)\*b(1) + b(2)]*

*Let,*

*[W(2) \* W(1)] = W*

*[W(2)\*b(1) + b(2)] = b*

*Final output : z(2) = W\*X + b*

*which is again a linear 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 an activation function.**

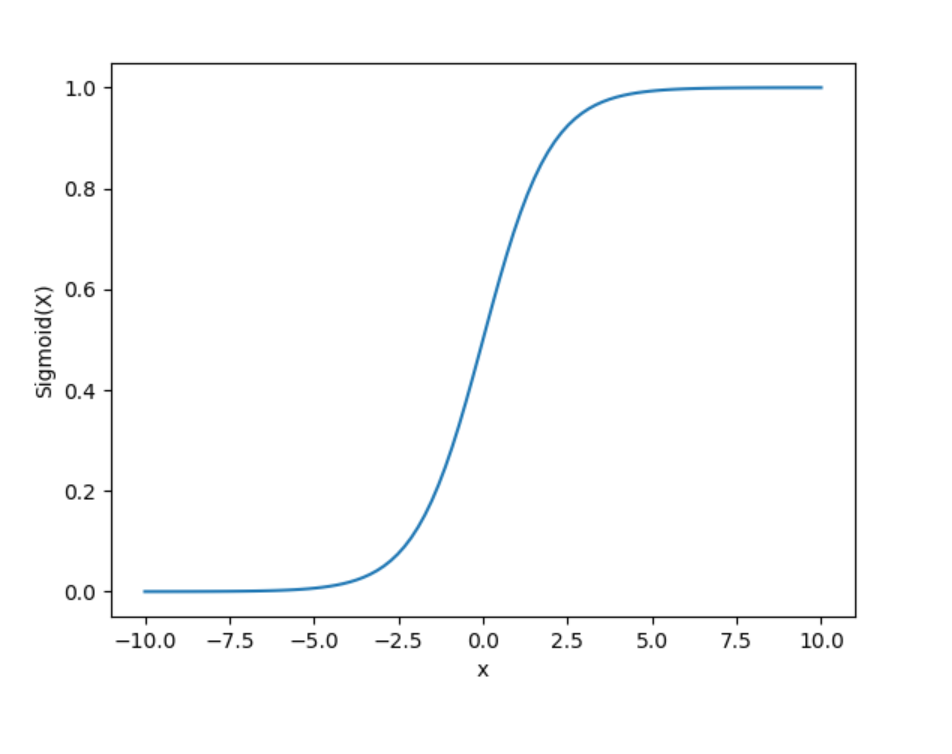
**Variants of Activation Function**

**Linear Function**

* **Equation :**Linear function has the equation similar to as of a straight line i.e. **y = x**
* No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
* **Range :** -inf to +inf
* **Uses : Linear activation function** is used at just one place i.e. output layer.
* **Issues :**If we will differentiate linear function to bring non-linearity, result will no more depend on *input “x”* and function will become constant, it won’t introduce any ground-breaking behavior to our algorithm.

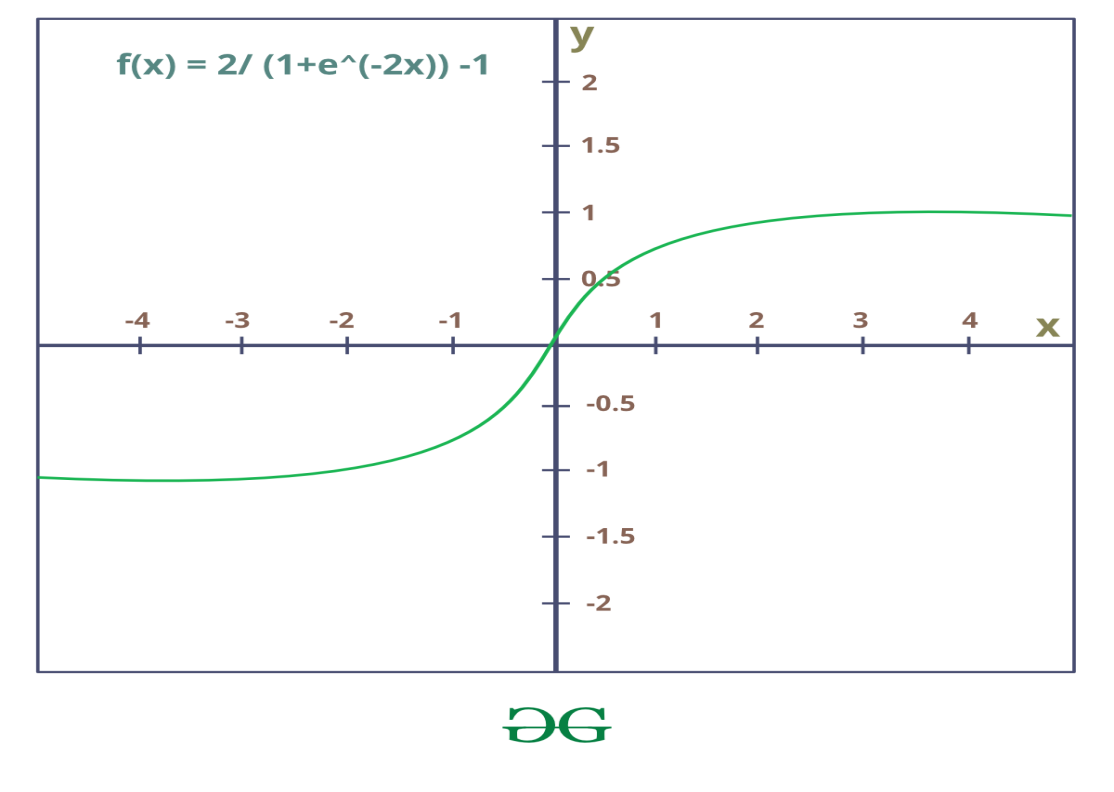
**For example :** Calculation of price of a house is a regression problem. House price may have any big/small value, so we can apply linear activation at output layer. Even in this case neural net must have any non-linear function at hidden layers.

**Sigmoid Function**

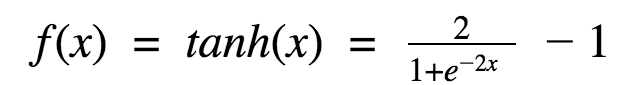


* It is a function which is plotted as **‘S’** shaped graph.
* **Equation :**A = 1/(1 + e-x)
* **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
* **Value Range :**0 to 1
* **Uses :**Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

**Tanh Function**



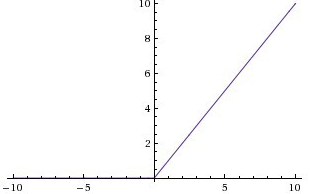
* The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It’s actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.
* **Equation :-**



**Value Range :-**-1 to +1

* **Nature :-**non-linear
* **Uses :-**Usually used in hidden layers of a neural network as it’s values lies between **-1 to 1**hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

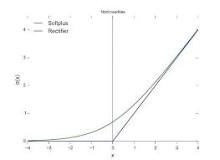
**RELU Function**



* It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.
* **Equation :- *A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
* **Value Range :-**[0, inf)
* **Nature :-**non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
* **Uses :-**ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

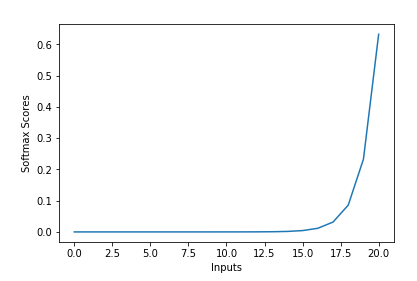
In simple words, RELU learns *much faster* than sigmoid and Tanh function.

Is ReLU linear or non-linear?



A rectified linear unit (ReLU) is an activation function that introduces the property of non-linearity to a deep learning model and solves the vanishing gradients issue. "It interprets the positive part of its argument. It is one of the most popular activation functions in deep learning.

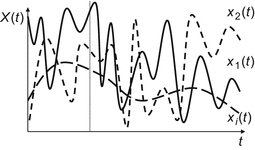
**Softmax Function**



The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi- class classification problems.

* **Nature :-**non-linear
* **Uses :-**Usually used when trying to handle multiple classes. the softmax function was commonly found in the output layer of image classification problems.The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
* **Output:-**The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.
* The basic rule of thumb is if you really don’t know what activation function to use, then simply use *RELU* as it is a general activation function in hidden layers and is used in most cases these days.
* If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.
* If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

A stochastic (random) function X(t) is a many-valued numerical function of an independent argument t, whose value for any fixed value t ∈ T (where T is the domain of the argument) is a random variable, called a cut set .



stochastic is well-described by a random probability distribution.

In stochastic neural networks, the algorithm instead of providing deterministic values to each neurons it assigns probabilities to each neuron. If each neuron passes the threshold values then only the neurons will fire. It is built by introducing random variation into the network and by giving stochastic weights.

Using Stochastic Processes is a key tool in real-time mathematical model of systems which has a continuous random varying nature. It has a wide beach of applications ranging from Image Processing, Neuroscience, Bio Informatics, Financial Management, Statistics, etc.

A variable or process is stochastic if there is uncertainty or randomness involved in the outcomes. Stochastic is a synonym for random and probabilistic, although is different from non-deterministic. Many machine learning algorithms are stochastic because they explicitly use randomness during optimization or learning.

Stochastic modeling forecasts the probability of various outcomes under different conditions, using random variables. Stochastic modeling presents data and predicts outcomes that account for certain levels of unpredictability or randomness.

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable).

Quora

then the main difference is that with fixed input the output of stochastic neural net is likely to be different (stochastic, or random to certain extent) for multiple evaluations, in contrast to deterministic neural networks, where for fixed input the output is also unique (deterministic).

Such neural networks are useful if you want to model behavior of partially random systems. Imagine that you set up an experiment where you show a picture and ask to name one thing human see on the image (and there are many different things on the image). You can anticipate a set of answers that human will give for a certain image, but you cannot say precisely what specific answer will be given by a human. Therefore if you would like to model such human behavior, you would prefer stochastic neural network to do so.

[[Profile photo for Andrew Sheehy](https://www.quora.com/profile/Andrew-Sheehy-9)](https://www.quora.com/profile/Andrew-Sheehy-9" \t "_blank)

[Related](https://www.quora.com/Are-neural-networks-stochastic-or-deterministic" \t "_blank)

**[Are neural networks stochastic or deterministic?](https://www.quora.com/Are-neural-networks-stochastic-or-deterministic" \t "_blank)**

After training has been completed, then the internal workings of a neural network are deterministic, not stochastic.

A neural network is essentially a mathematical structure that transforms one data object applied to the input end into another data object which appears at the output end.

If we are thinking about determinism, then a neural network is no different to this completely made-up function: y(x) = [3x^3 - 1.8x^2 + sin(3x/4)] / 6.5exp(4x + 3).

y(x) will always return the same result when x=0.3447 which will a real number.

If you wrote out the equation for a neural network like this then it would be extremely complex, but it would produce deterministic results just the same: you would only need to apply a given data structure to the input end once. You do have to apply the same input again and again and analyse the distribution of results. You only get one result.

However, the training algorithm is not deterministic, which means that the parameter values you get after one training process are very likely to be different to those that you will get after another training process - even when the training data is the same. This is actually why the training of a complex neural network is a bit of an art and can often involve a fair bit of trail and error as some training journeys either lead to poor results or fail to converge.

But let’s return to how the network works after it has been trained.

The picture becomes more subtle when we have deliberately *designed*y(x) to return a statistical parameter. So we might want y(x) to represent the confidence level that a given input (e.g. an image) contains a ‘stop’ sign an so the value of y(x) might in this case range from 0 to 1.

So a more complete answer to your question would be that after it has been trained, a neural network is *intrinsically* deterministic - but we might *interpret*the output it generates scholastically.

Deterministic update: If the activation value exceeds the threshold, the node/neuron fires.

Stochastic update: If the activation value exceeds the threshold, there is a probability associated with firing. That is there is a probability of the neuron not firing even if it exceeds the threshold.

If the probability is one then that update is Deterministic.

**What is the capacity of a perceptron?**

From an information theory point of view, a single perceptron with K inputs has a capacity of 2K bits of information.

**What is the capacity of a neural network?**

Neural networks are defined at various levels of abstraction, and thus it models different aspects of neural networks. Therefore the network capacity is nothing but the levels of abstraction or the number of fundamental memories or the number of patterns that can be stored and recalled in a network.

What are the limitations of simple perceptron?

**The following are the limitation of a Perceptron model:**

* The output of a perceptron can only be a binary number (0 or 1) due to the hard-edge transfer function.
* It can only be used to classify the linearly separable sets of input vectors. If the input vectors are non-linear, it is not easy to classify them correctly.
* **What is the perceptron convergence procedure?**
* Perceptron Convergence Theorem: For any finite set of linearly separable labeled examples, the Perceptron Learning Algorithm will halt after a finite number of iterations. In other words, after a finite number of iterations, the algorithm yields a vector w that classifies perfectly all the examples