Activation Functions in Neural Networks

What is Activation Function?

It's just a thing function that you use to get the output of node. It is also known as Transfer Function.

We can simply define it,

"An activation function is a mathematical function that transforms the weighted sum of input values at a neuron into an output signal determining whether the neuron activated based on a specific threshold or rule."

Why we use Activation functions with Neural Networks?

- It is used to determine the output of neural network like yes or no.
- It maps the resulting values between 0 to 1 or -1 to 1 etc. (depending upon the function).

Types of activation function:

The Activation Functions can be basically divided into 2 types-

1. Linear Activation Function
2. Non-linear Activation Functions

1. Linear or Identity Activation Function

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

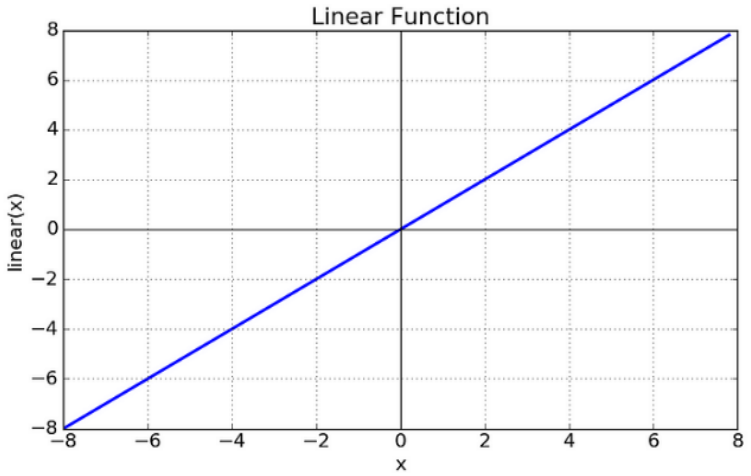


Fig: Linear Activation Function

Equation : f(x) = x **Range :** (-infinity to infinity)

2. Non-linear Activation Function

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this

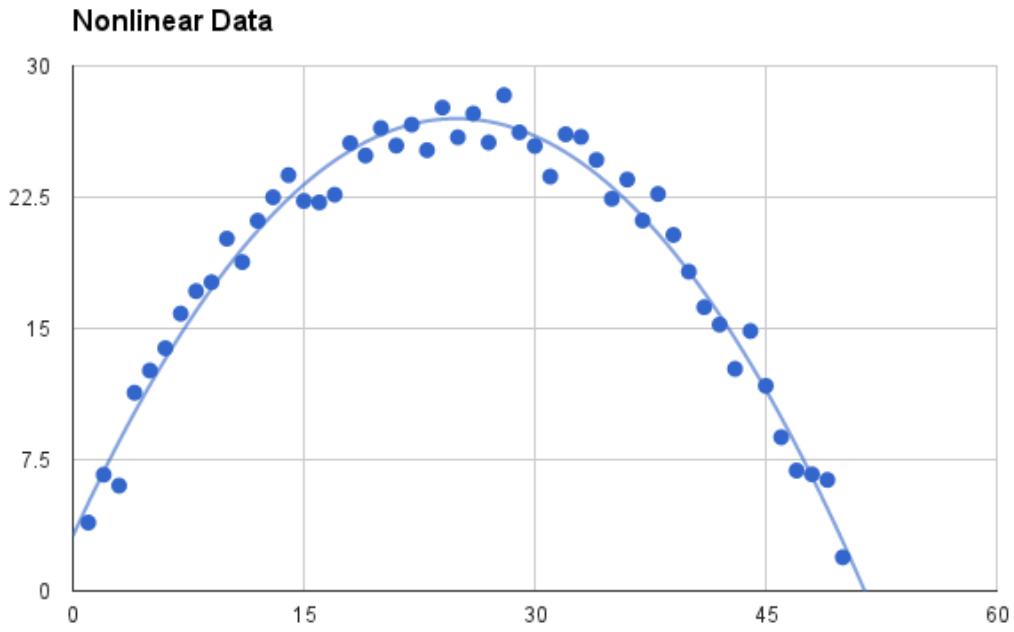


Fig: Non-linear Activation Function

It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

The main terminologies needed to understand for nonlinear functions are:

Derivative or Differential: Change in y-axis w.r.t. change in x-axis.It is also known as slope.

Monotonic function: A function which is either entirely non-increasing or non-decreasing.

The Nonlinear Activation Functions are mainly divided on the basis of their **range or curves**-

1. Sigmoid or Logistic Activation Function
2. Tanh or hyperbolic tangent Activation Function
3. ReLU (Rectified Linear Unit) Activation Function
4. Leaky ReLU
5. Softmax

2.1. Sigmoid or Logistic Activation Function

The Sigmoid Function curve looks like a S-shape.

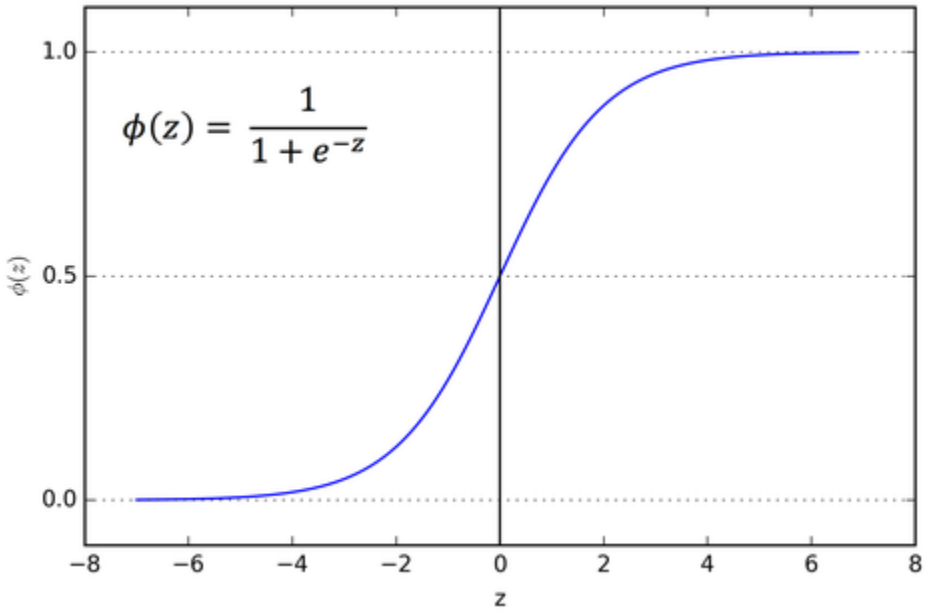


Fig: Sigmoid Function

The main reason why we use sigmoid function is because it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output.Since probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.

The function is **differentiable**. That means, we can find the slope of the sigmoid curve at any two points.

The logistic sigmoid function can cause a neural network to get stuck at the training time.

The **softmax function** is a more generalized logistic activation function which is used for multiclass classification.

2.2. Tanh or hyperbolic tangent Activation Function

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).

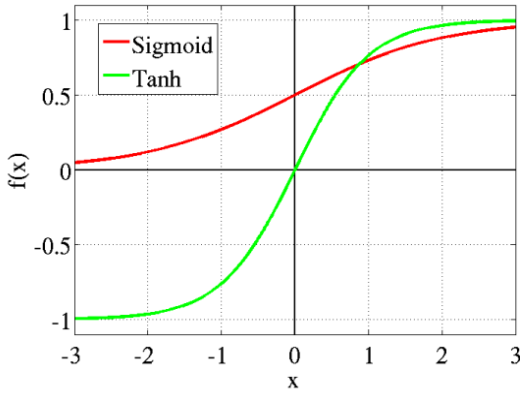


Fig: tanh v/s Logistic Sigmoid

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

The function is **differentiable**.

The tanh function is mainly used classification between two classes.

Both tanh and logistic sigmoid activation functions are used in feed-forward nets.

2.3. ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning.

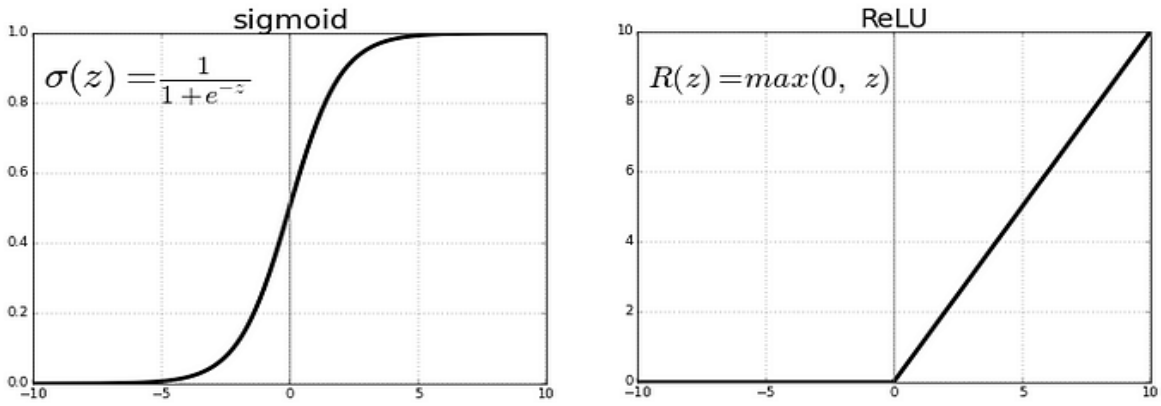


Fig: ReLU v/s Logistic Sigmoid

Range: [0 to infinity)

As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

The function and its derivative **both are monotonic**.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

2.4. Leaky ReLU

It is an attempt to solve the dying ReLU problem

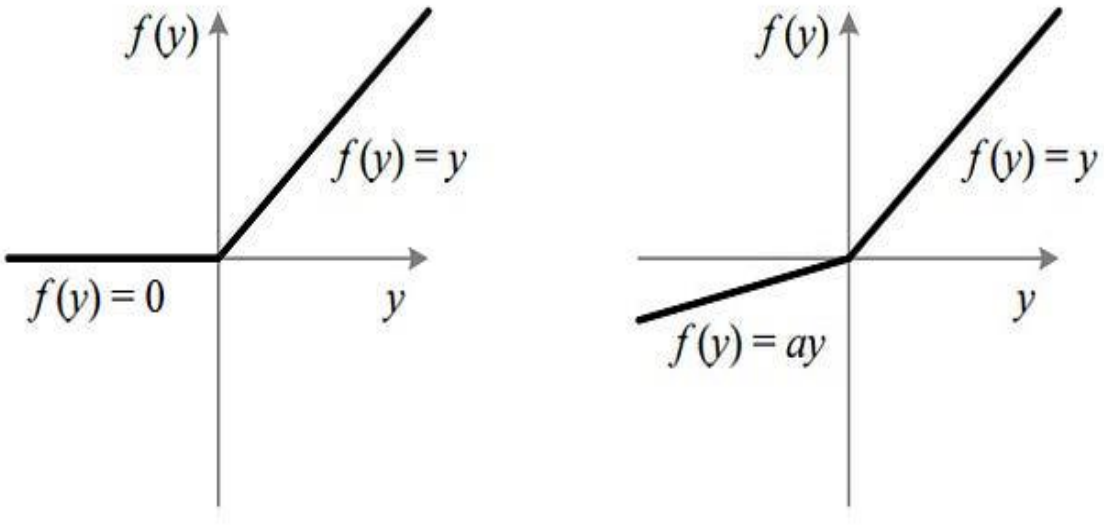


Fig : ReLU v/s Leaky ReLU

The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so.

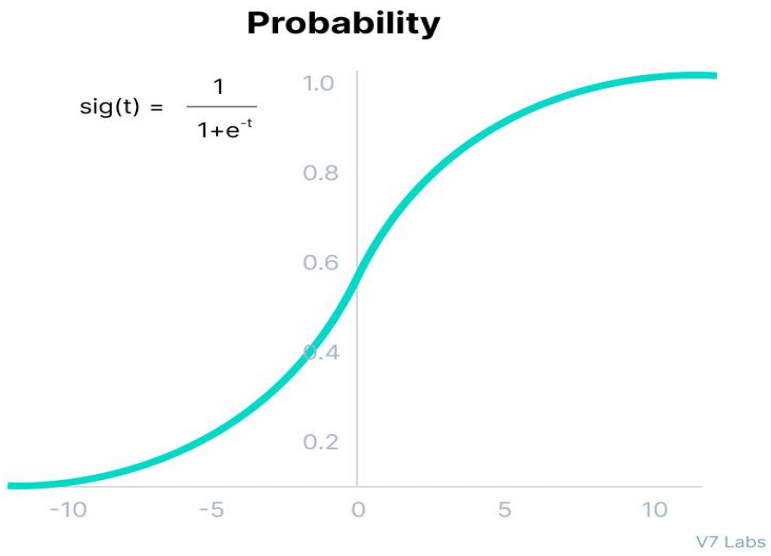
When **a is not 0.01** then it is called **Randomized ReLU**.

Therefore the **range** of the Leaky ReLU is (-infinity to infinity).

Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

2.5 Softmax Function

Before exploring the ins and outs of the Softmax activation function, we should focus on its building block—the sigmoid/logistic activation function that works on calculating probability values.



The output of the sigmoid function was in the range of 0 to 1, which can be thought of as probability.

But, This function faces certain problems.

Let's suppose we have five output values of 0.8, 0.9, 0.7, 0.8, and 0.6, respectively. How can we move forward with it?

The answer is: We can't.

The above values don't make sense as the sum of all the classes/output probabilities should be equal to 1.

You see, the Softmax function is described as a combination of multiple sigmoids.

It calculates the relative probabilities. Similar to the sigmoid/logistic activation function, the SoftMax function returns the probability of each class.

It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification.

Mathematically it can be represented as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Let’s go over a simple example together.

Assume that you have three classes, meaning that there would be three neurons in the output layer. Now, suppose that your output from the neurons is [1.8, 0.9, 0.68].

Applying the softmax function over these values to give a probabilistic view will result in the following outcome: [0.58, 0.23, 0.19].

The function returns 1 for the largest probability index while it returns 0 for the other two array indexes. Here, giving full weight to index 0 and no weight to index 1 and index 2. So the output would be the class corresponding to the 1st neuron(index 0) out of three.

You can see now how softmax activation function make things easy for multi-class classification problems.

3. Common Challenges to Choose an Activation Function and mitigation of the challenges

Choosing the right activation function is a crucial decision in designing a neural network, as it can significantly impact the model's performance. Here are some common challenges associated with choosing an activation function and ways to mitigate them:

- **Vanishing and Exploding Gradients:** In deep neural networks, gradients can become extremely small (vanishing) or large (exploding) during backpropagation, leading to slow or divergent training.
  Mitigation: Use activation functions that are less prone to vanishing/exploding gradients, such as the Rectified Linear Unit (ReLU) and its variants (Leaky ReLU, Parametric ReLU, etc.). Batch normalization can also help stabilize training.
- **Dead Neurons:** Some neurons may become inactive and stop learning during training, known as "dead neurons," especially with certain activation functions.
  Mitigation: Use activation functions like Leaky ReLU or Parametric ReLU, which allow a small gradient for negative inputs, preventing neurons from becoming entirely inactive.
- **Sensitivity to Input Scale:** Some activation functions are sensitive to the scale of input data, which can affect the convergence of the network.
  Mitigation: Standardize or normalize input data to have zero mean and unit variance. This can help mitigate the sensitivity to input scale.

- **Non-Monotonic Activation Functions:** Some activation functions are not monotonic, making it difficult for gradient-based optimization algorithms to converge.

  Mitigation: Choose activation functions that are monotonic, such as the rectified linear units (ReLU) and variants.
- **Computational Complexity:** Some activation functions, especially those involving complex mathematical operations, can increase the computational complexity of the model.

  Mitigation: Consider using simpler activation functions like ReLU, which are computationally efficient. If necessary, use approximations or piecewise linear functions to balance computational efficiency and expressive power.
- **Sigmoid/Tanh Saturation:** Sigmoid and hyperbolic tangent (tanh) functions can saturate for extreme values, causing the gradients to become very small and slow down learning.

  Mitigation: Use alternatives like the scaled exponential linear unit (SELU) or the Rectified Linear Unit (ReLU) family to mitigate the vanishing gradient problem associated with sigmoid and tanh.
- **Choice of Activation Function for Output Layer:** The choice of activation function for the output layer depends on the task (e.g., sigmoid for binary classification, SoftMax for multiclass classification).

  Mitigation: Match the activation function of the output layer to the nature of the problem. For regression tasks, a linear activation function is often used.