Activation Functions

What is an activation function?

While Building a Neural Network we mostly have three layers: the input layer, the hidden layer, and the output layer. 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 based on the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

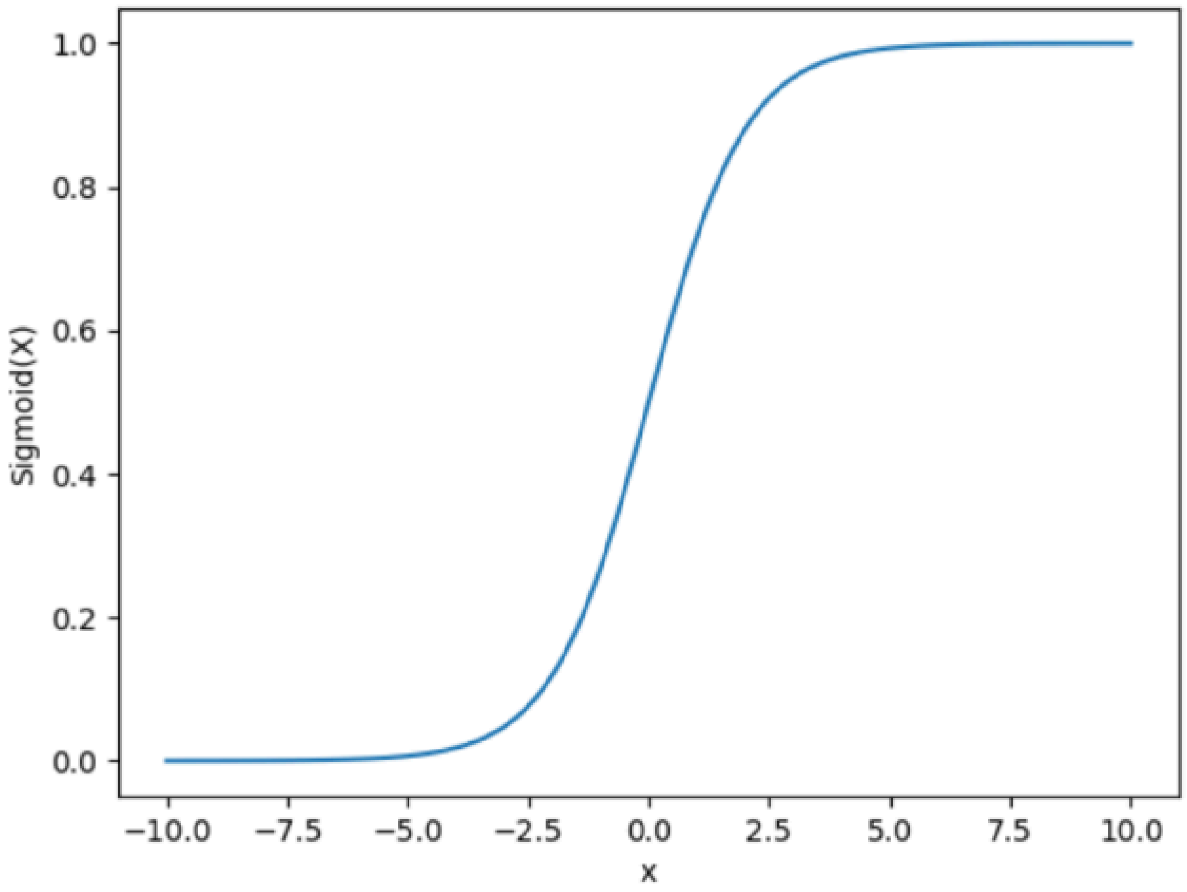
The activation function decides whether a neuron should be activated by calculating the weighted sum and adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Types of Activation Function:

We use different types of activation functions while building a neural network. Some of the common activation functions are:

1. Sigmoid Function:

- It is a function plotted as an '**S**' shaped graph.
- **Equation: A = 1/(1 + e-x)**
- **Graph:**



- **Pros:**

  Smoothness: The sigmoid function is smooth and differentiable across its entire range. This property is beneficial for optimization algorithms like gradient descent, making it easier to find the minimum or maximum of the function.

  Output Range: The sigmoid function outputs values in the range (0, 1), which is particularly useful for binary classification problems where the goal is to produce probabilities. The output can be interpreted as the probability of belonging to a particular class.

  Logistic Interpretation: The sigmoid function is closely related to the logistic regression model, and its output can be interpreted as the log-odds ratio. This makes it convenient for modeling binary outcomes in statistics.

- **Cons:**

Vanishing Gradient Problem: One significant drawback of the sigmoid function is the vanishing gradient problem, especially in deep neural networks. As the input moves away from the origin (0), the gradient of the sigmoid becomes very close to zero. This can lead to slow or stalled learning during backpropagation, particularly in deep networks.

Output Not Centered Around Zero: The sigmoid function's output is not centered around zero, which may not be ideal for certain optimization scenarios. Functions like the hyperbolic tangent (tanh) address this issue by having an output range centered around zero.

Output Saturation: The sigmoid function can saturate for extreme input values, leading to a flat gradient. This can hinder the learning process, as the model becomes less responsive to changes in the input.

Not Zero-Centered: The sigmoid function is not zero-centered, which might be a disadvantage when used as an activation function in certain network architectures. This can cause issues like biased weight updates during training.

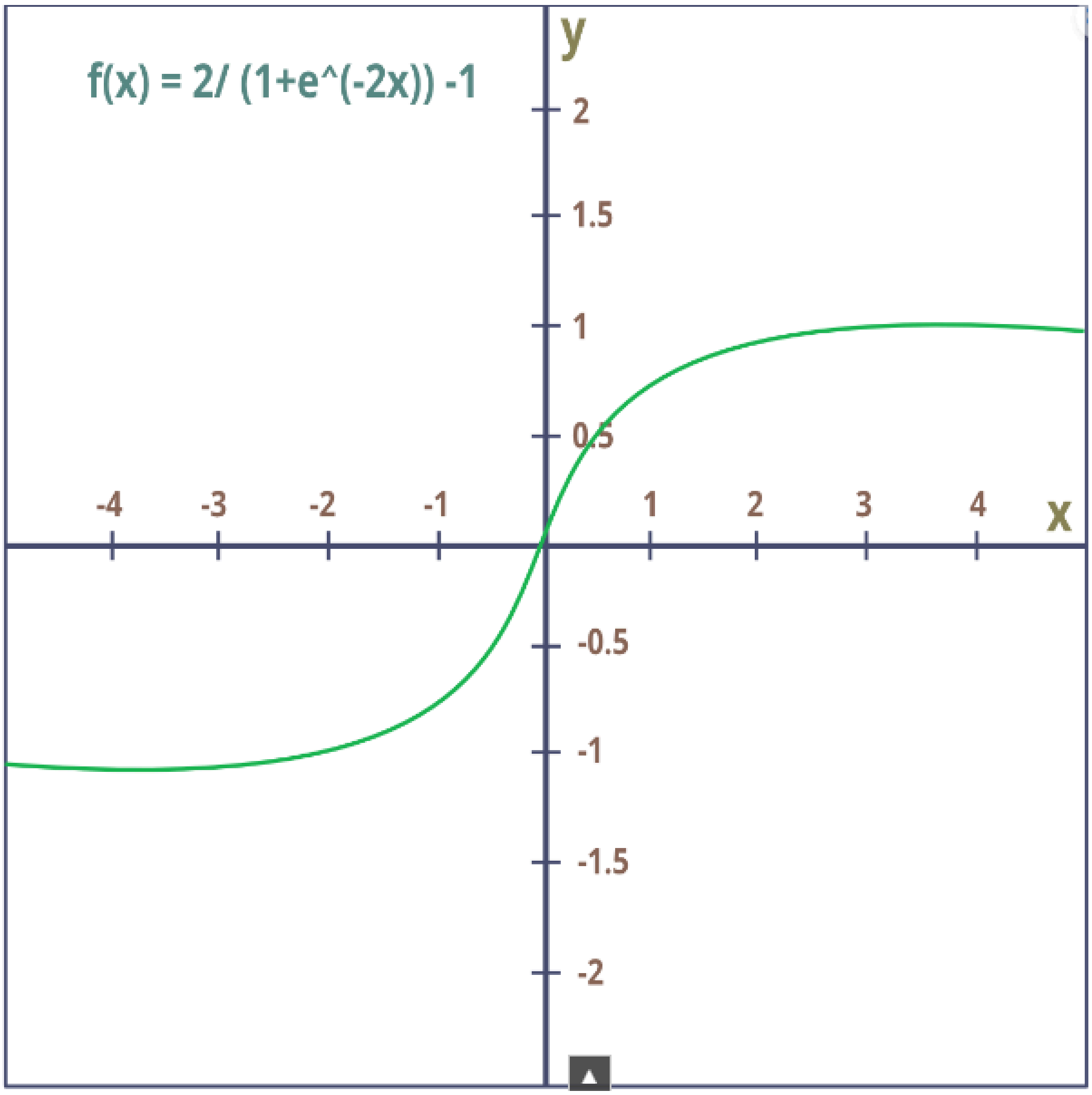
2. Tanh Function :

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

- **Equation :**

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

- **Graph :**



- **Pros:**

  Zero-Centered: The tanh function is zero-centered, meaning that its output range spans from -1 to 1. This zero-centered property can be beneficial for optimization algorithms, as it helps mitigate issues such as the vanishing gradient problem associated with functions like the sigmoid.

Smooth and Differentiable: Similar to the sigmoid function, tanh is smooth and differentiable across its entire range. This smoothness facilitates the use of gradient-based optimization algorithms like backpropagation.

Output Range: The tanh function outputs values in the range (-1, 1), making it suitable for situations where data might have negative and positive components. This can be advantageous in certain network architectures.

Less Likely to Saturate: While tanh can still saturate for extreme input values, it tends to saturate less quickly than the sigmoid function. This can help in mitigating the vanishing gradient problem to some extent.

- **Cons:**

  Vanishing Gradient Problem: Despite being less prone to the vanishing gradient problem compared to the sigmoid function, tanh can still suffer from this issue, especially in deep neural networks. The gradient can become very small for inputs far from zero, affecting the learning process.

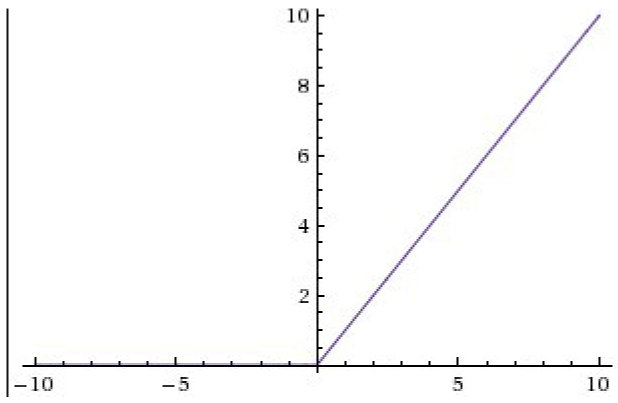
  Non-Zero-Centered in Practice: While tanh is theoretically zero-centered, in practice, during the training of deep networks, the mean of the activations may shift away from zero. This can still introduce challenges related to optimization and convergence.

  Computational Cost: The tanh function involves exponentiation, which can be computationally more expensive compared to functions like the rectified linear unit (ReLU) which only involve simple thresholding operations.

  Not Always Preferable for All Layers: Tanh is often used in hidden layers, but it might not be the best choice for all layers in a neural network. In some cases, the choice of activation functions may vary based on the specific architecture and the nature of the problem being solved.

3. RELU Function:

- It Stands for Rectified linear unit. It is the most widely used activation function. Chiefly implemented in hidden layers of the Neural network.
- **Equation :- A(x) = max(0,x)**
- **Graph :**



- **Pros:**

  Simple and Efficient: ReLU is a simple and computationally efficient activation function. It only involves a threshold operation, setting negative values to zero and leaving positive values unchanged.

  Promotes Sparsity: ReLU tends to sparsely activate neurons, meaning that only a subset of neurons are activated for any given input. This can lead to more efficient representations and reduce the likelihood of overfitting.

  Mitigates Vanishing Gradient Problem: Unlike the sigmoid and tanh functions, ReLU does not suffer from the vanishing gradient problem to the same extent. The derivative of ReLU is either 0 or 1, which helps propagate gradients more effectively during backpropagation.

  Accelerates Training: The simplicity of the ReLU function and its non-saturating nature can lead to faster training of deep neural networks. It allows for more straightforward weight updates during backpropagation.

  Zero-Centered for Positive Values: While ReLU itself is not zero-centered, it is zero-centered for positive values. This property can be beneficial in certain optimization scenarios.

- **Cons:**

  Dead Neurons: One common issue with ReLU is the "dead neuron" problem. If a large gradient flows through a ReLU unit and updates the weights in such a way that the unit will always output zero, the unit becomes inactive (dead), and it stops learning altogether.

Not Suitable for All Data: ReLU may not be suitable for all types of data, especially when dealing with sequences or data with both positive and negative values. It tends to zero out negative values, potentially discarding useful information.

Not Smooth Everywhere: The ReLU function is not differentiable at zero, which can be problematic for certain optimization algorithms that rely on derivatives. However, this issue is often addressed in practice by using subgradients or variants like Leaky ReLU.

Not Always Zero-Centered: While ReLU is zero-centered for positive values, it is not zero-centered for negative values. This lack of zero-centeredness can affect the convergence and learning dynamics in some cases.

References:

- Geeks-for-geeks & Google
- Picture Collected from Geeks-for-geeks