Machine Learning

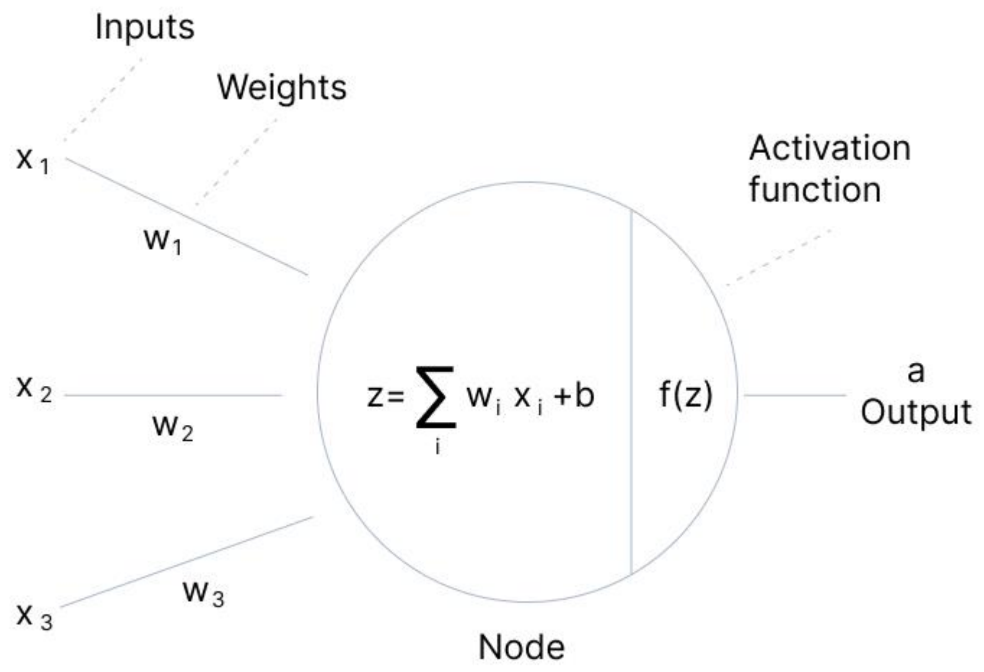
Dr Changjiang He, Dr Kuo-Ming Chao
Computer Science| School of Art
University of Roehampton

Lesson 6.1

Activation Function

Activation Function

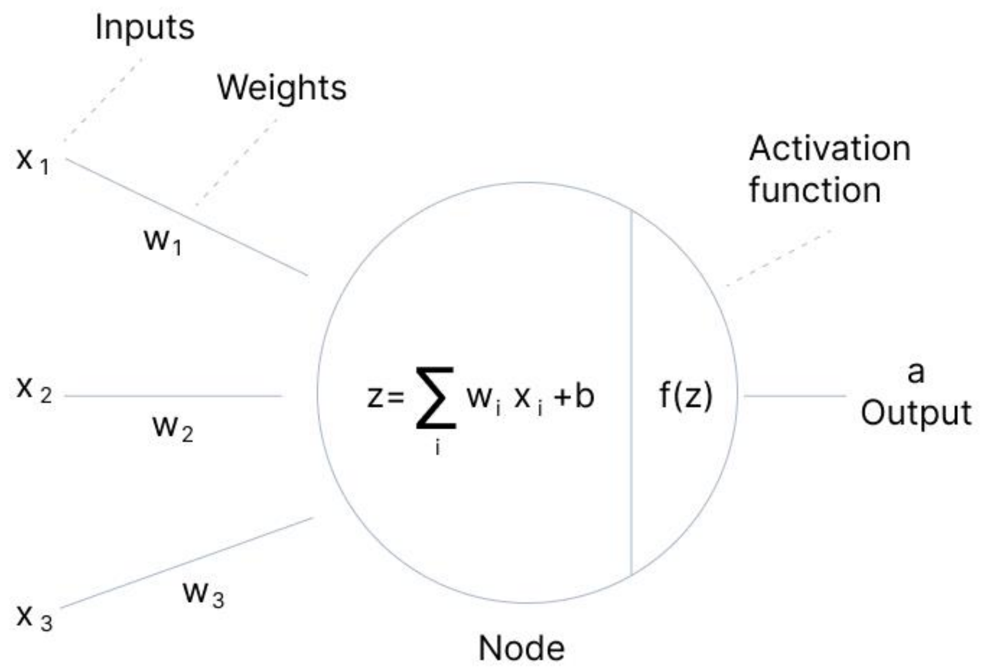
- An Activation Function decides whether a neuron should be activated or not.
- This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.



Activation Function

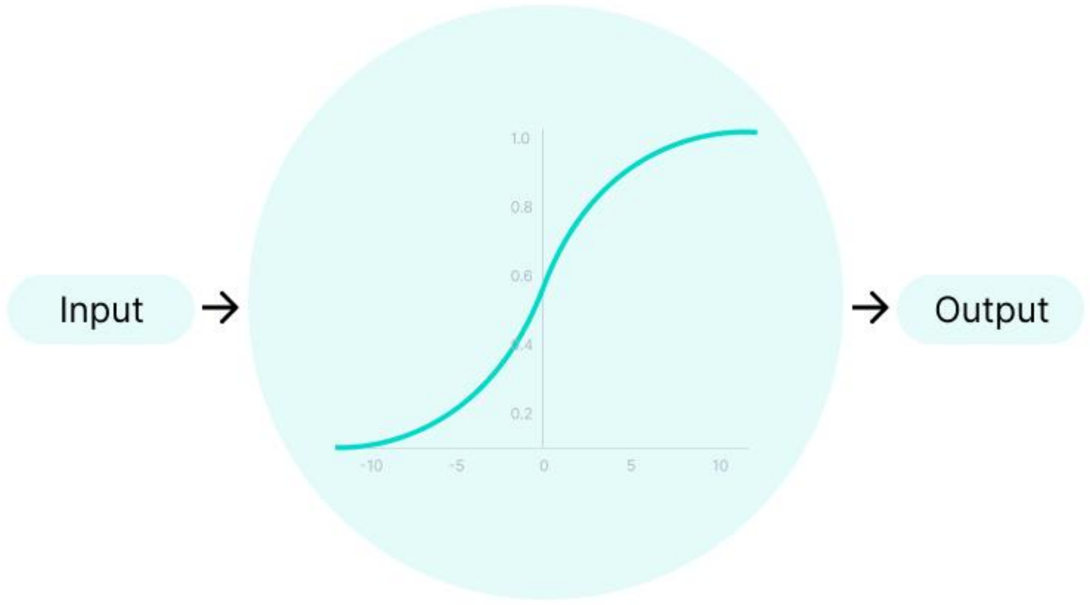
- The role of the activation function is to derive output from a set of input values fed to a node (or a layer).

- In other words, the primary role of the activation function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.



Why Activation Function

- The purpose of an activation function is to add non-linearity to the neural network.
- Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it.



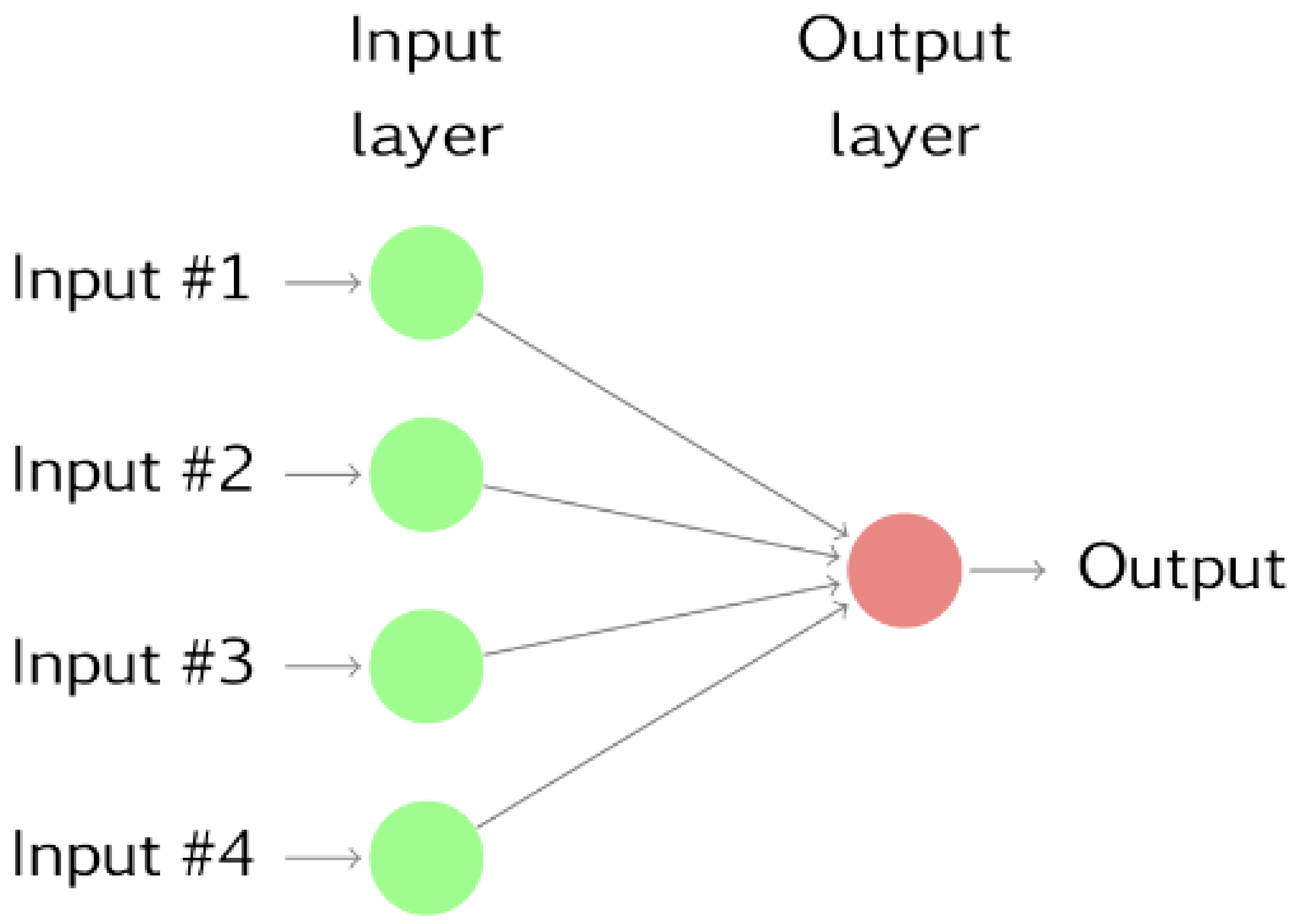
Activation Function

Activation Function

- Let’s suppose we have a neural network working without the activation functions.
- In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases.
- It’s because it doesn’t matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself.

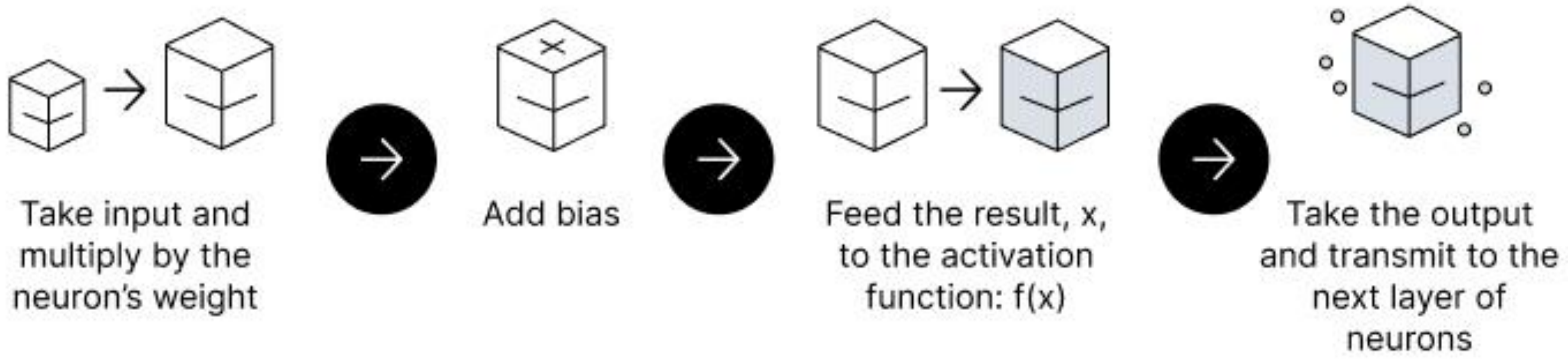
Activation Function

- Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model.



Forward Propagation

- In the feedforward propagation, the Activation Function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.


Take input and multiply by the neuron's weight
Add bias
Feed the result, x, to the activation function: f(x)
Take the output and transmit to the next layer of neurons

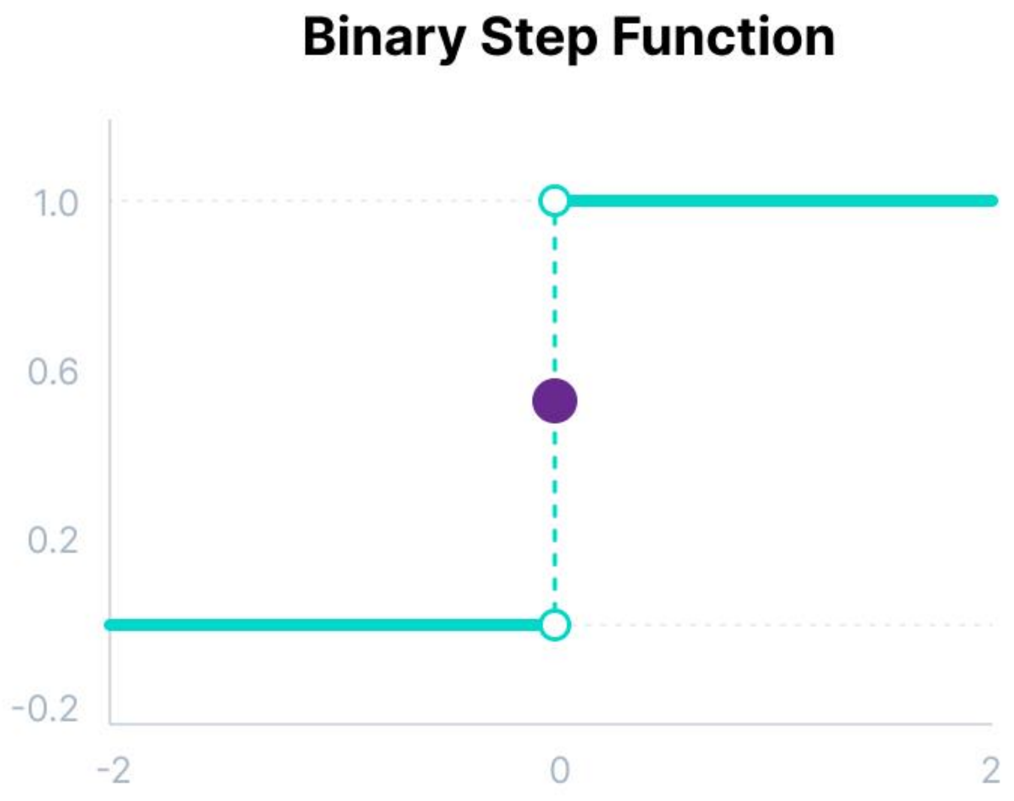
Backward Propagation

- backpropagation aims to minimize the cost function by adjusting the network’s weights and biases.

- The cost function gradients determine the level of adjustment with respect to parameters like activation function, weights, bias, etc.

Binary Step Function

- Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
- The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



Binary Step Function

- Mathematically it can be represented as: $f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$

- Here are some of the limitations of binary step function:

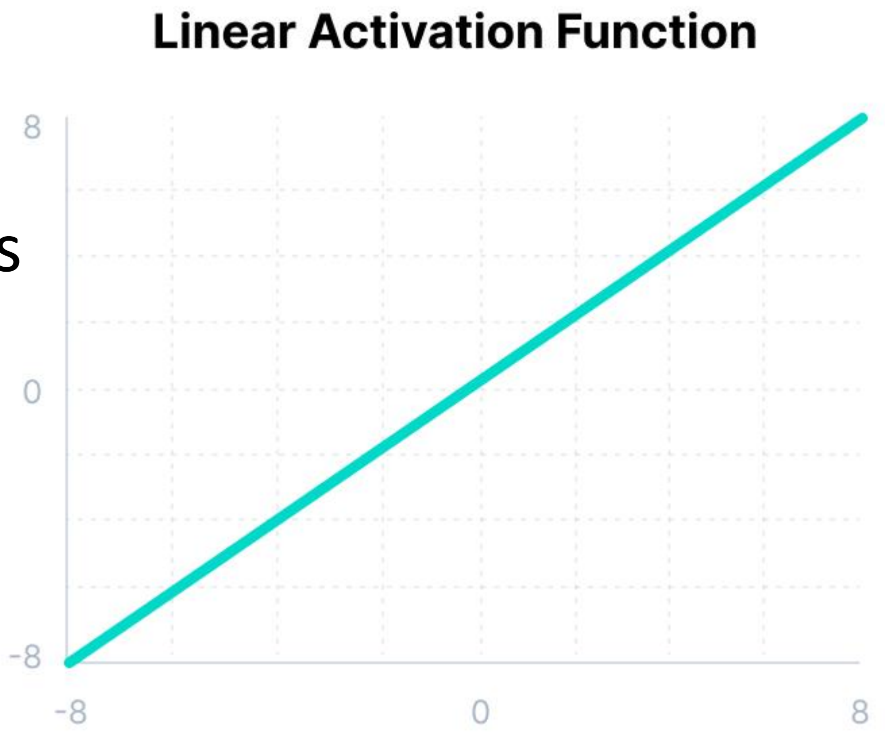
  - It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.

  - The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

Linear Activation Function

- The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input.

- The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.



Linear Activation Function

- Mathematically it can be represented as: $f(x) = x$

- However, a linear activation function has two major problems :
  - It’s not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x. All layers of the neural network will collapse into one if a linear activation function is used.
  - No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

Nonlinear Activation Function

Non-linear activation functions solve the following limitations of linear activation functions:

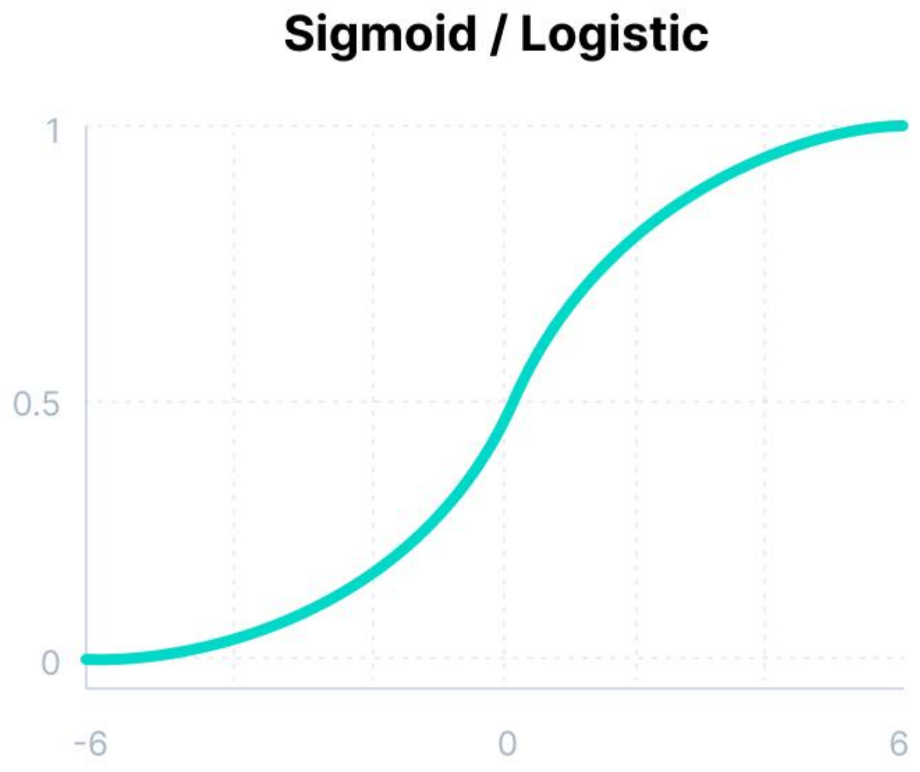
- They allow backpropagation because now the derivative function would be related to the input, and it’s possible to go back and understand which weights in the input neurons can provide a better prediction.

- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

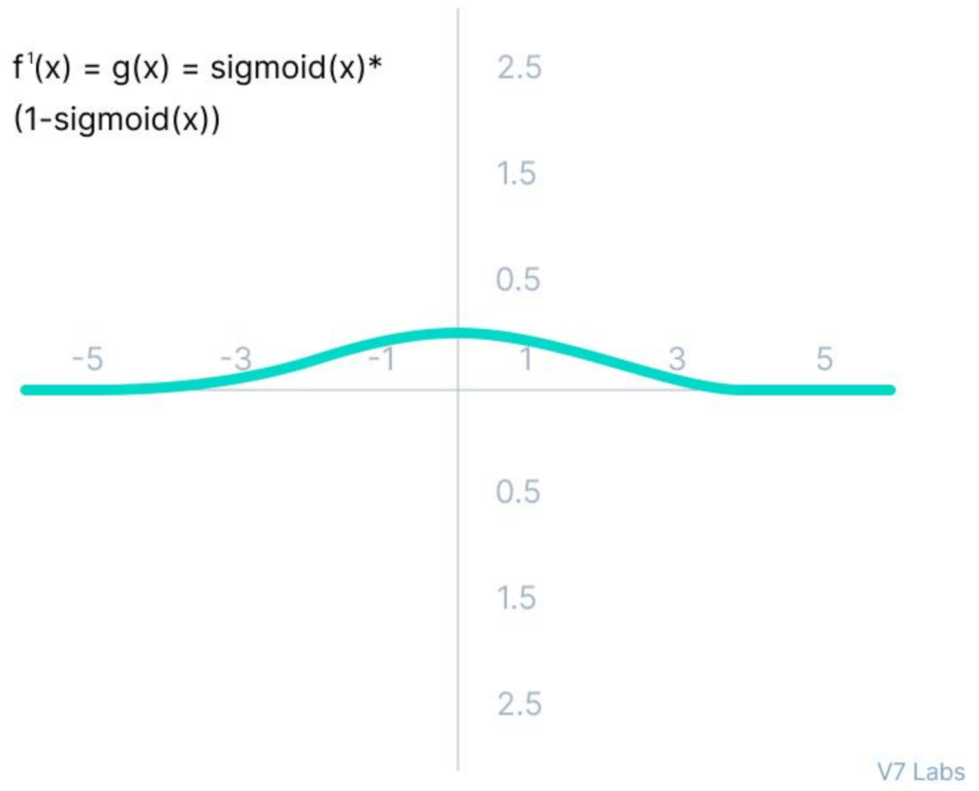
Sigmoid / Logistic Activation Function

- This function takes any real value as input and outputs values in the range of 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, as shown below.



Sigmoid / Logistic Activation Function

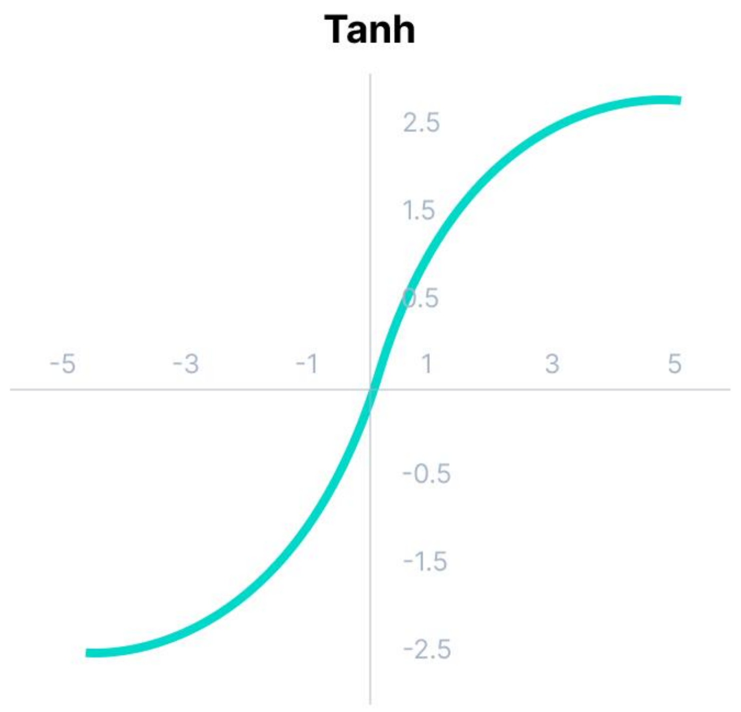


Sigmoid / Logistic Activation Function

- Mathematically it can be represented as: $$f(x) = \frac{1}{1 + e^{-x}}$$
- Here’s why sigmoid/logistic activation function is one of the most widely used functions:
  - It is commonly 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 because of its range.
  - The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

Tanh Function

- Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1.
- In Tanh, 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.

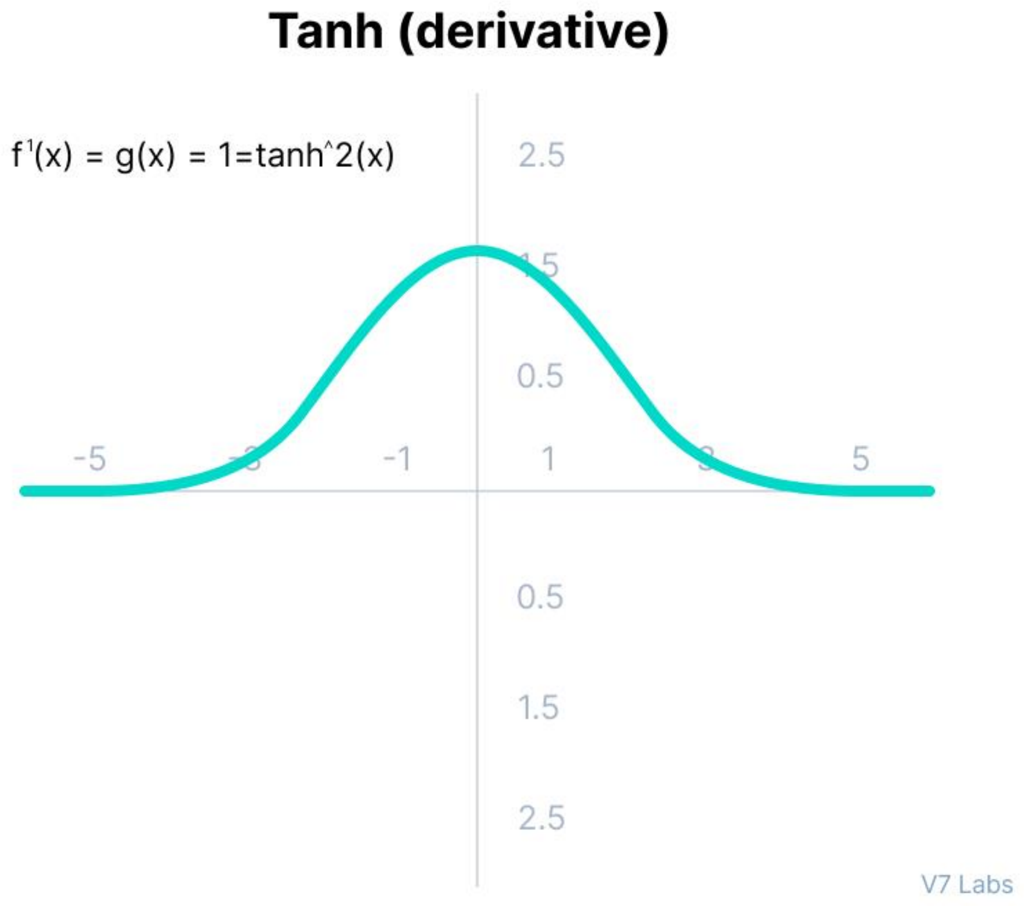


Activation Function

- Mathematically it can be represented as: $f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$

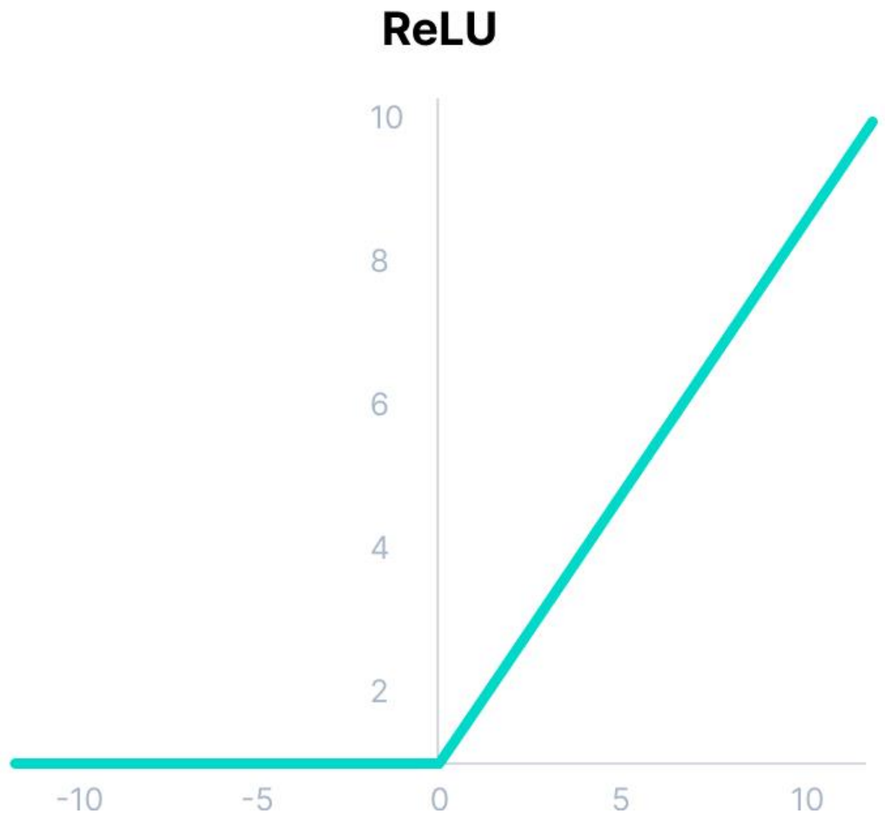
- Advantages of using this activation function are: The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive. Usually used in hidden layers of a neural network as its values lie between -1 to 1; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data and makes learning for the next layer much easier.

Activation Function



ReLU

- ReLU stands for Rectified Linear Unit.
- The main catch here is that the ReLU function does not activate all the neurons at the same time.
- The neurons will only be deactivated if the output of the linear transformation is less than 0.



ReLU

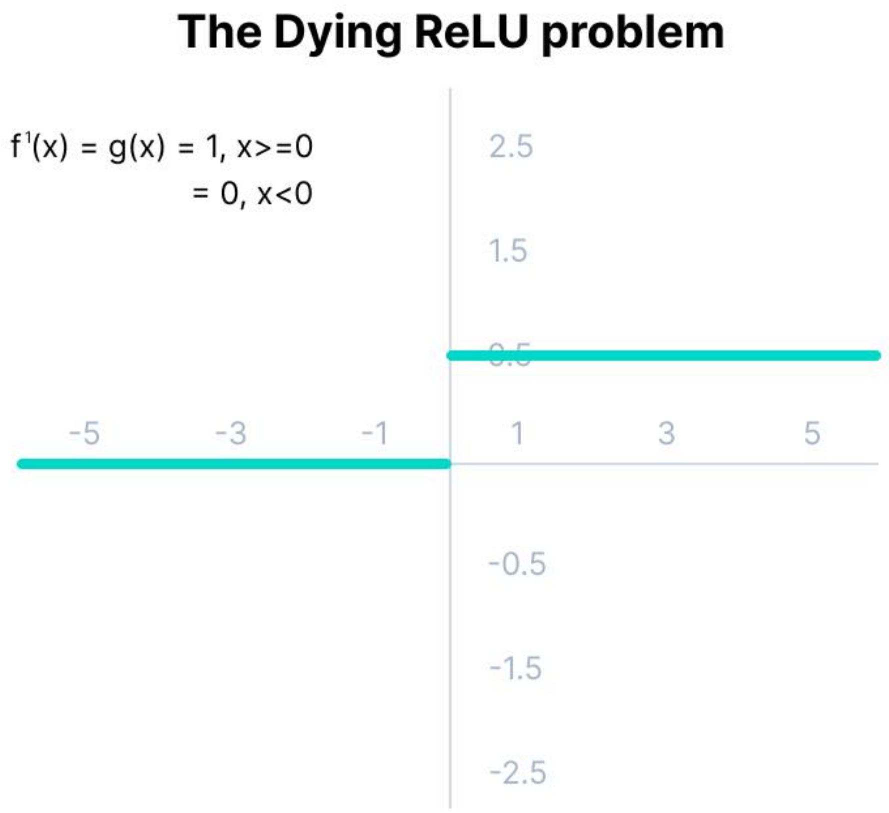
- Mathematically it can be represented as: $f(x) = max\ (0, x)$

- The advantages of using ReLU as an activation function are as follows: Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions.

- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

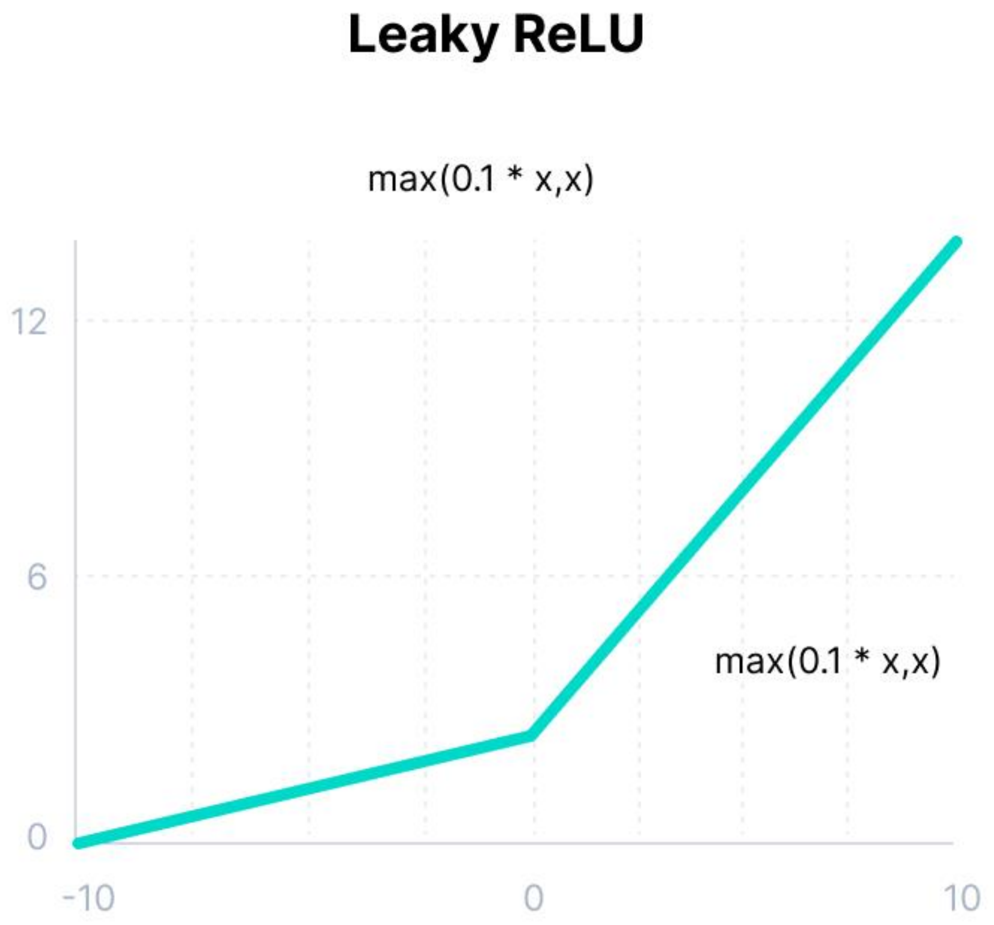
ReLU

- The limitations faced by this function are the Dying ReLU problem.
- The negative side of the graph makes the gradient value zero.
- Due to this reason, during the backpropagation process, the weights and biases for some neurons are not updated.



Leaky ReLU

- Leaky ReLU is an improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area.



Leaky ReLU

- Mathematically it can be represented as: $f(x) = max\ (0.1x, x)$

- The advantages of Leaky ReLU are same as that of ReLU, in addition to the fact that it does enable backpropagation, even for negative input values.

- By making this minor modification for negative input values, the gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would no longer encounter dead neurons in that region.

Softmax

- 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. Probability in Softmax Function Probability .

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

Softmax

- 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.

Softmax

- 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.

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

How to choose activation function

As a rule of thumb, you can begin with using the ReLU activation function and then move over to other activation functions if ReLU doesn’t provide optimum results.

And here are a few other guidelines to help you out.

1.ReLU activation function should only be used in the hidden layers.

2.Sigmoid/Logistic and Tanh functions should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).

3.Swish function is used in neural networks having a depth greater than 40 layers.

How to choose activation function

a few rules for choosing the activation function for your output layer based on the type of prediction problem that you are solving:

1. **Regression** - Linear Activation Function
2. **Binary Classification**—Sigmoid/Logistic Activation Function
3. **Multiclass Classification**—Softmax
4. **Multilabel Classification**—Sigmoid

How to choose activation function

The activation function used in hidden layers is typically chosen based on the type of neural network architecture.

5.**Convolutional Neural Network (CNN)**: ReLU activation function.
6.**Recurrent Neural Network**: Tanh and/or Sigmoid activation function.