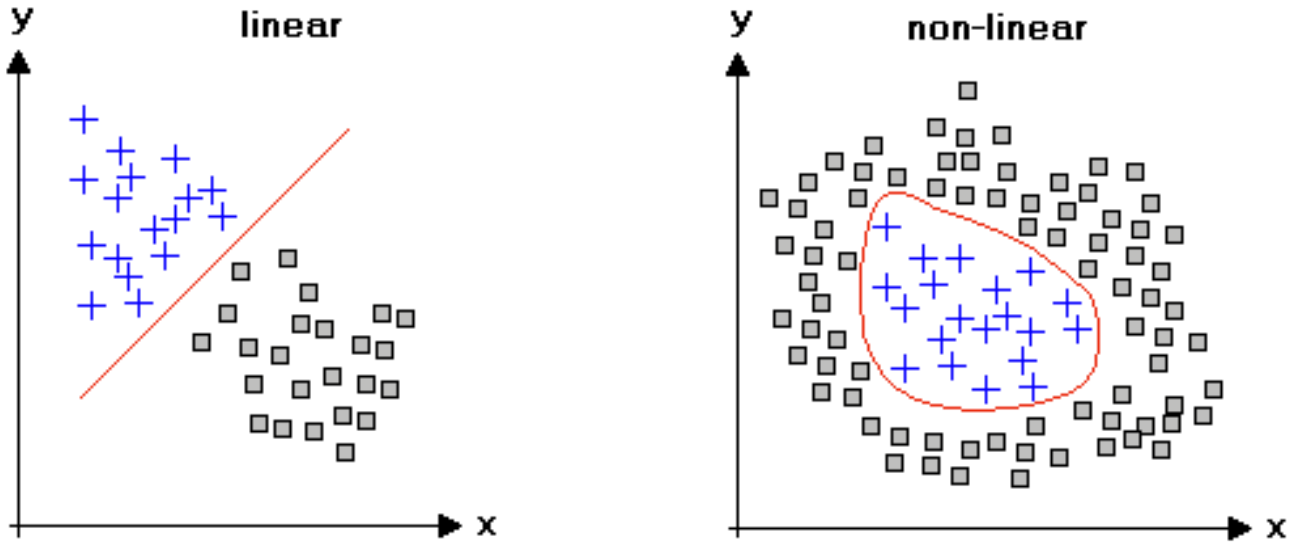
Definition

Activation functions are necessary to introduce non-linearity into an ANN. This is important when our goal is to learn complex pattern in data. Without them,the model only behaves as a single linear model $y = W * X + B$. ***Thus, a deep ANN without activation function is still equivalent to a shallow one.***



To understand this better: Suppose we have the following ANN:

- 10 input features.
- 2 hidden layers with 20 neurons each.
- An output layer with 2 neurons.

A ***forward pass*** without an activation function would look like this:

$L_1 = W1 * X + B_1$ <=> (20,1) = (20,10) *(10,1)* + *(20,1)*
$L_2 = W2 L1 + B_2$ <=> *(20,1)* = *(20,20)* * *(20, 1)* + *(20, 1)*
$y*{prediction} = W3 *L_2* + *B_3$ <=> *(2,1)* = *(2,20)* (20,1) + (2,1)

Each transformation above is linear, stacking them only results in another linear transformation.

Universal Approximation Theorem

A mathematical theorem stating that a neural network with a single hidden layer and a finite number of neurons can approximate any continuous function, given an appropriate activation function.

$|f(x) - \hat{f}(x)| < \epsilon$ for all x where $\hat{f}(x)$ is the neural network (the approximation) and $f(x)$ is the real function we are trying to approximate.

A Side Note

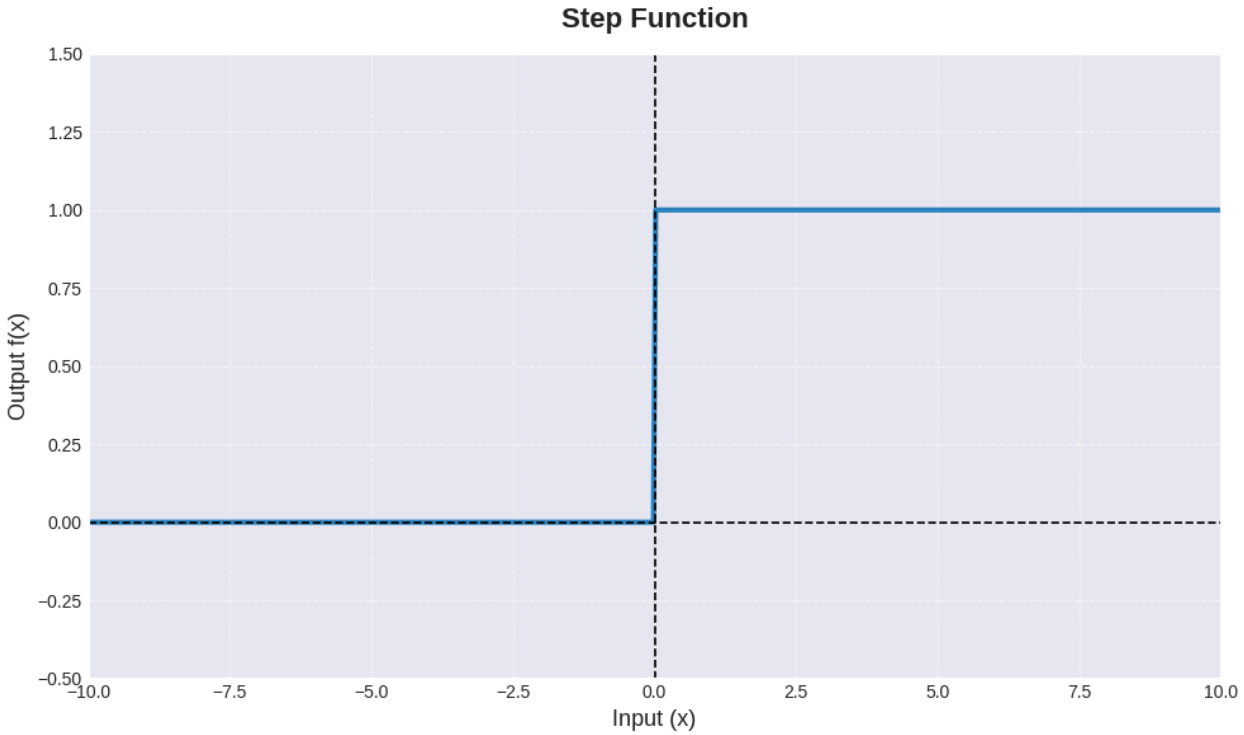
An ANN can have multiple activation functions and not only one. Typically, one used in the hidden layers, and another one for the output layer.

Types of Activation Functions

We will discuss several types of activation functions:

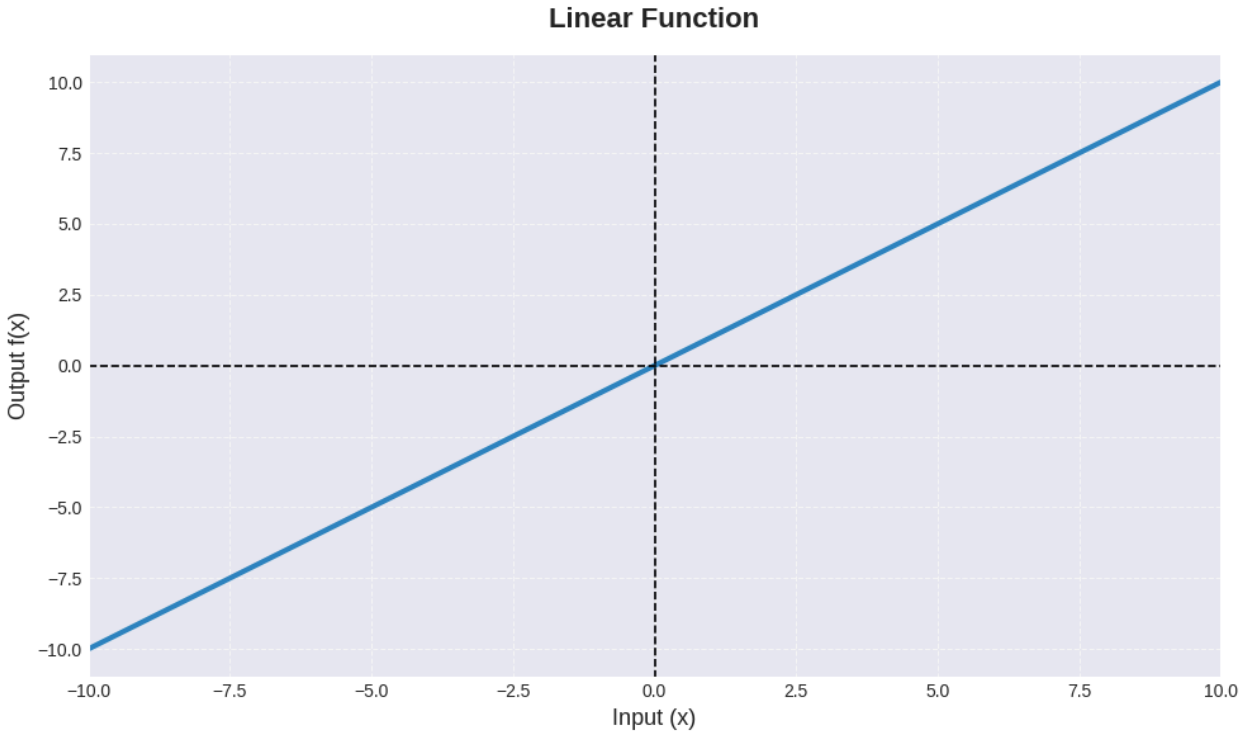
- **Step Activation Function**
  - This is the basic type of an activation function. It tries to mimic the behavior of a neuron (activate, does not activate).
    - If the result of $W * X + B > 0$, the neuron is activated and outputs 1
    - If the result of $W * X + B <= 0$, the neuron is deactivated (outputs zero).
  - This step function was one of the earliest, it is rarely used nowadays.

    - Not differentiable at zero and has zero derivative everywhere, does not work with gradient-based optimization methods.



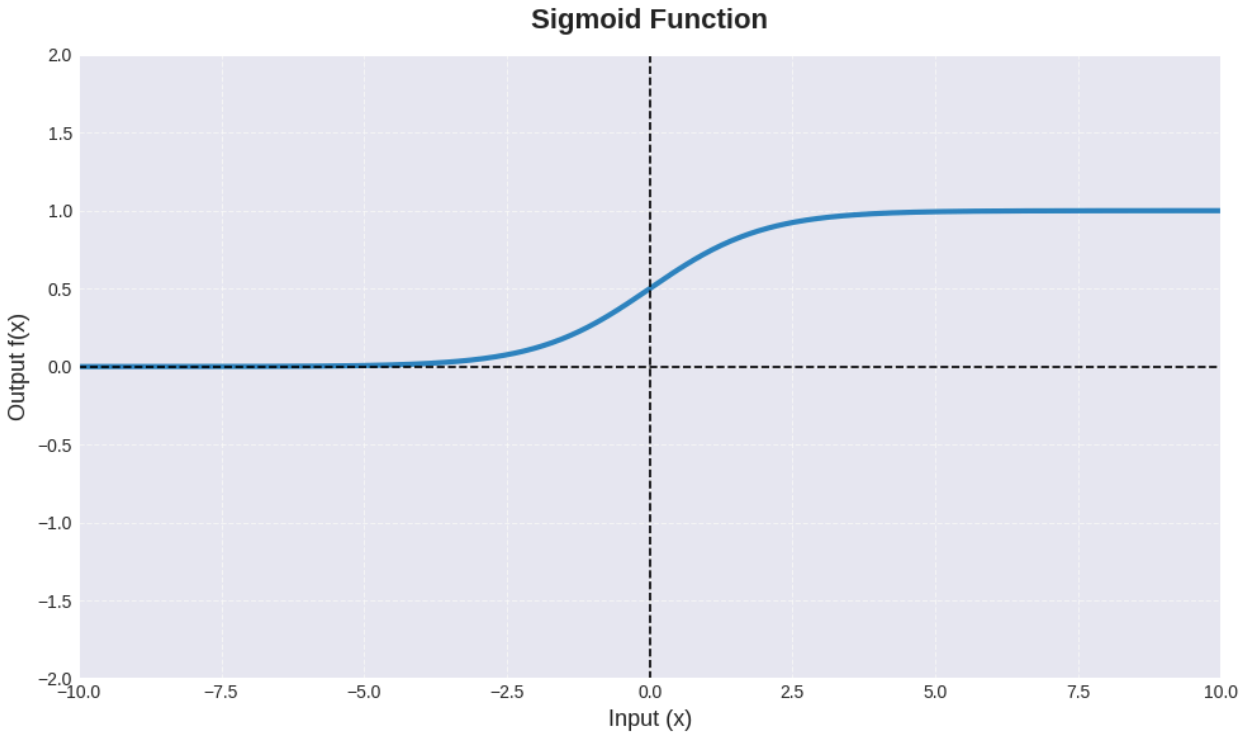
- **Linear Activation Function**
  - A linear function is simply the equation of a line.

    $y = x$
  - A linear activation function is usually used in the output layer in the case of a regression model (which outputs a scalar value instead of classification).
  - It must not be used in the hidden layers of the ANN (makes it similar to a single layer linear model).

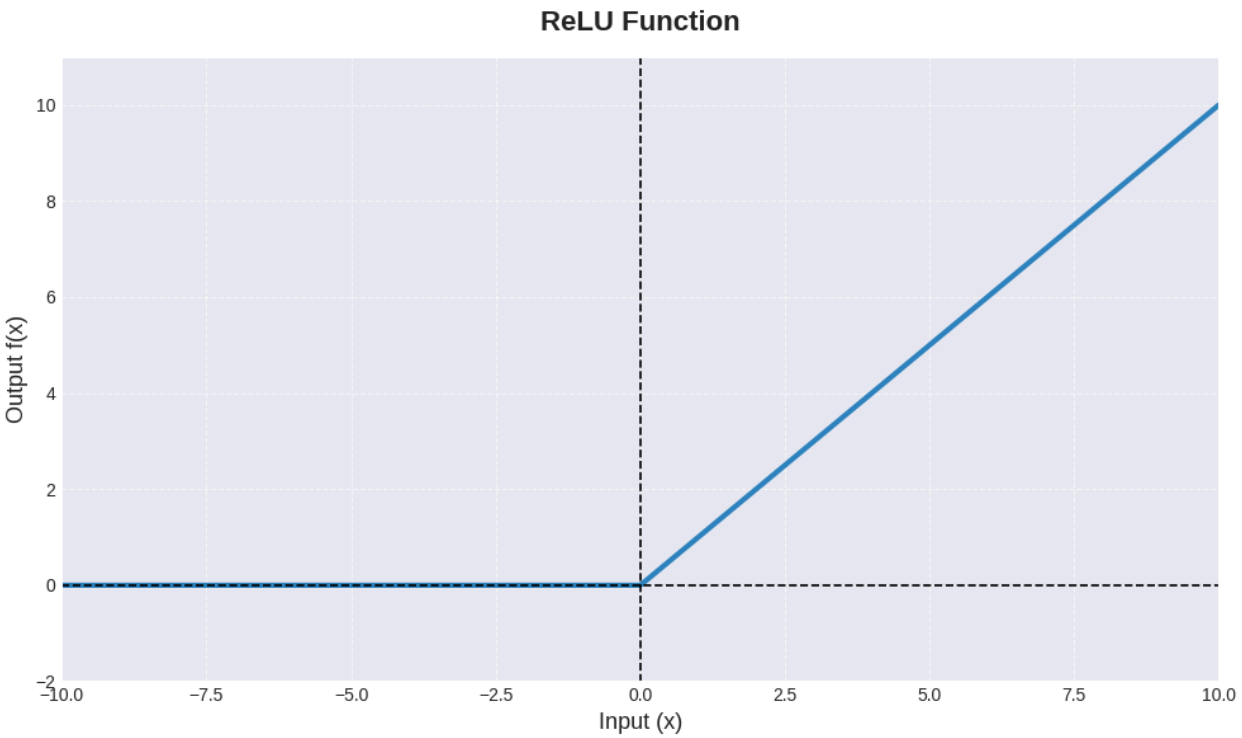


- **Sigmoid Activation Function**
  - The problem with the step function is that it is not very informative. When training a neural network (as we'll see later) we have to asses the impact of weights and biases on the network's output. The step function erases this information by outputting only 1 or zero.

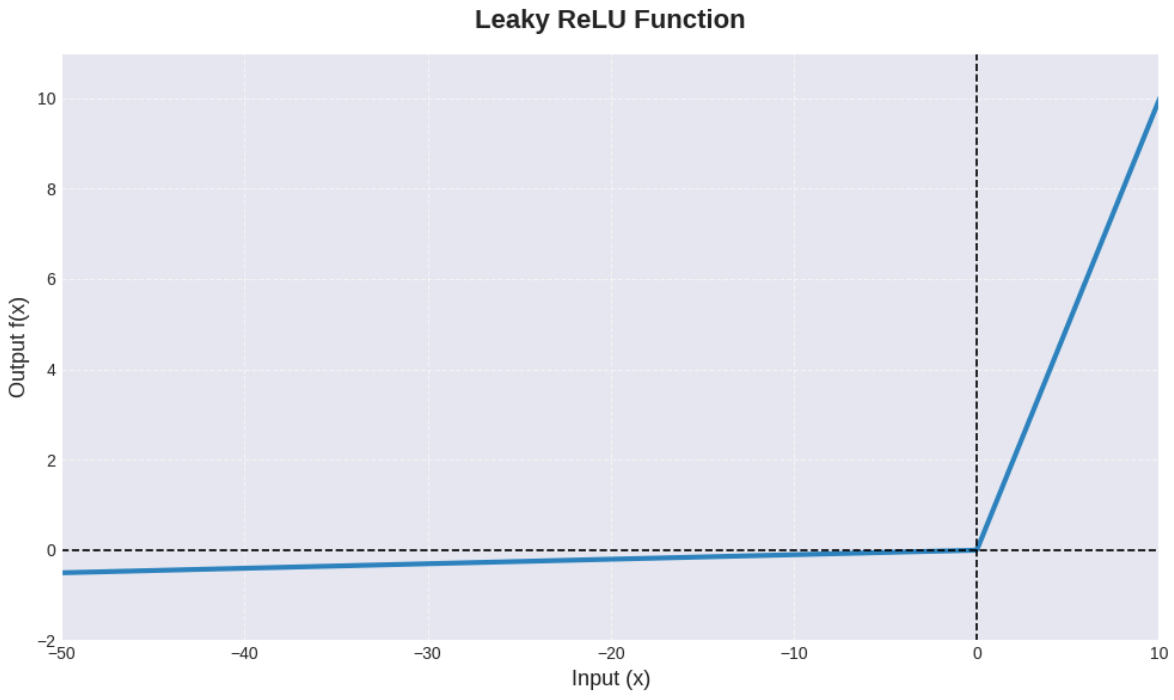
- **The step function makes it hard to guess how close the function was from activating or deactivating the neuron.** Thus we have to have a more informative activation function.
- The original activation function used for ANNs is called the sigmoid function. $y = \frac{1}{1+e^{-x}}$



  - This function returns a value of 0 when x is -infinity, 0.5 when x = 0, and 1 for +infinity.
  - Used in the output layers of **binary classifiers** as it squishes the output to values between zero and one.
  - Used in **probabilistic models** where we need the output to reflect some probability value.
  - For large (positive/negative) the sigmoid function acts as a step function, which does not help in learning (slow, or no learning at all). (**vanishing gradient**)
- **Rectified Linear Activation Function**
  - ***ReLU*** is simpler than sigmoid, it is literally $y = x$ clipped from zero. If x <=0 then y is 0.

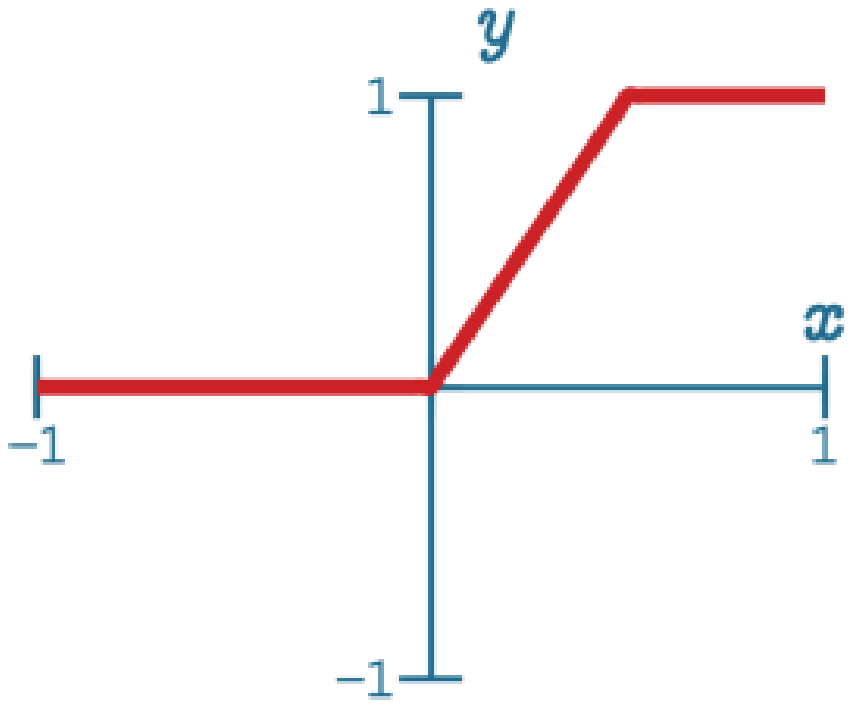


- Pros
  - Solves the vanishing gradient problem for x > 0
  - Faster learning than sigmoid as it only calculates $max(0, x)$ unlike sigmoid where we need to calculate the exponential.
- Cons
  - Does not limit/constraint the input value x, (it grows with x unlike sigmoid).
  - Can cause Dying ReLU when most of the values are (wx + b) are negative. This can be avoided by doing proper initialization or by using Leaking ReLU.



- Example:
  - Consider two consecutive neurons from different layers

- The weight connecting x to the first neuron has a value of 1, and the bias is 0.5. A ReLU is used as activation.
- The weight connecting the output (after activation) to the second neuron is -2 and the bias is 1. A ReLU is used as activation.
- Draw the resulting function y w.r.t x
- Solution:



- **Softmax Activation Function**
  - Softmax is usually used on the output layer.
  - Best Suitable for classification problems where we have N classes.
  - ReLU does not have a bound on the numbers it outputs, so it cannot be used here.
  - Sigmoid does not work when we have multiple output neurons (classes) since each output is independent.
  - Sigmoid is usually preferred over softmax in the case of multi-label classification.
  - **Softmax** converts raw outputs from the neural network into probabilities, where each probability corresponds to the likelihood of each class.
    - The output values are between 0 and 1 and sum up to 1, making them interpretable as a probability distribution.
    - Softmax is particularly useful in tasks like image recognition, natural language processing, and other applications that require selecting one class out of several possible choices.
      $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$
    - z_i is the output for the $i-th$ class

- N is the number of classes