Chapter 4: Activation Functions

We use different activation functions for different cases.

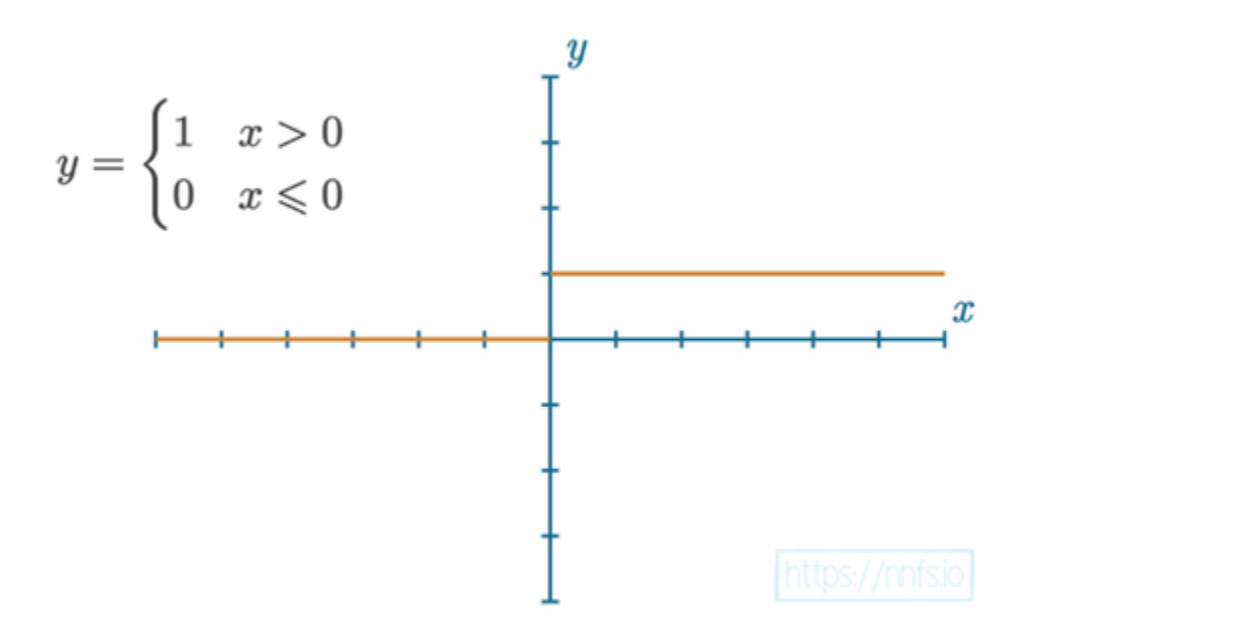
The activation function is applied to the output of a neuron (or layer of neurons), which modifies outputs.

Neural network will have **two types of activation functions**:

- Activation function **used in hidden layers**(will be the same for all of them, but it doesn't have to.)
- Activation function **used in the output layer**.

The Step Activation Function

- This activation function serves is to mimic a neuron "firing" or "not firing" based on input information.
- If the weights · inputs + bias results in a value greater than 0, the neuron will fire and output a 1; otherwise, it will output a 0.
- Has been used historically in hidden layers, but nowadays, it is rarely a choice.



The Linear Activation Function

- Simply the equation of a line where y=x and the output value equals the input.
- Usually applied to the last layer's output in the case of a regression model — a model that outputs a scalar value instead of a classification.

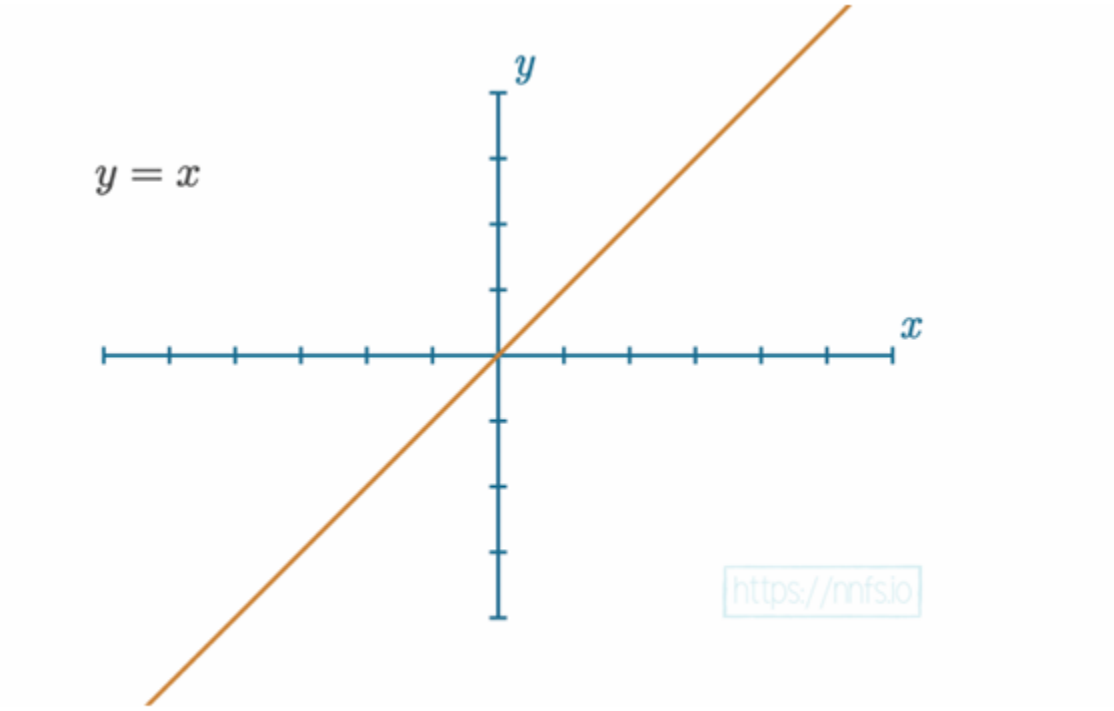


Fig 4.02: Linear function graph.

The Sigmoid Activation Function

- The step function isn't very helpful for training because it only gives outputs of 0 or 1, offering no info on how close it was to switching. This makes it hard for optimizers to adjust weights and biases effectively. More detailed activation functions work better for learning.
- Original, more granular, activation function used for neural networks.

- Returns the value:
  - In the range of 0 for negative infinity,
  - 0.5 for the input of 0,
  - 1 for positive infinity.

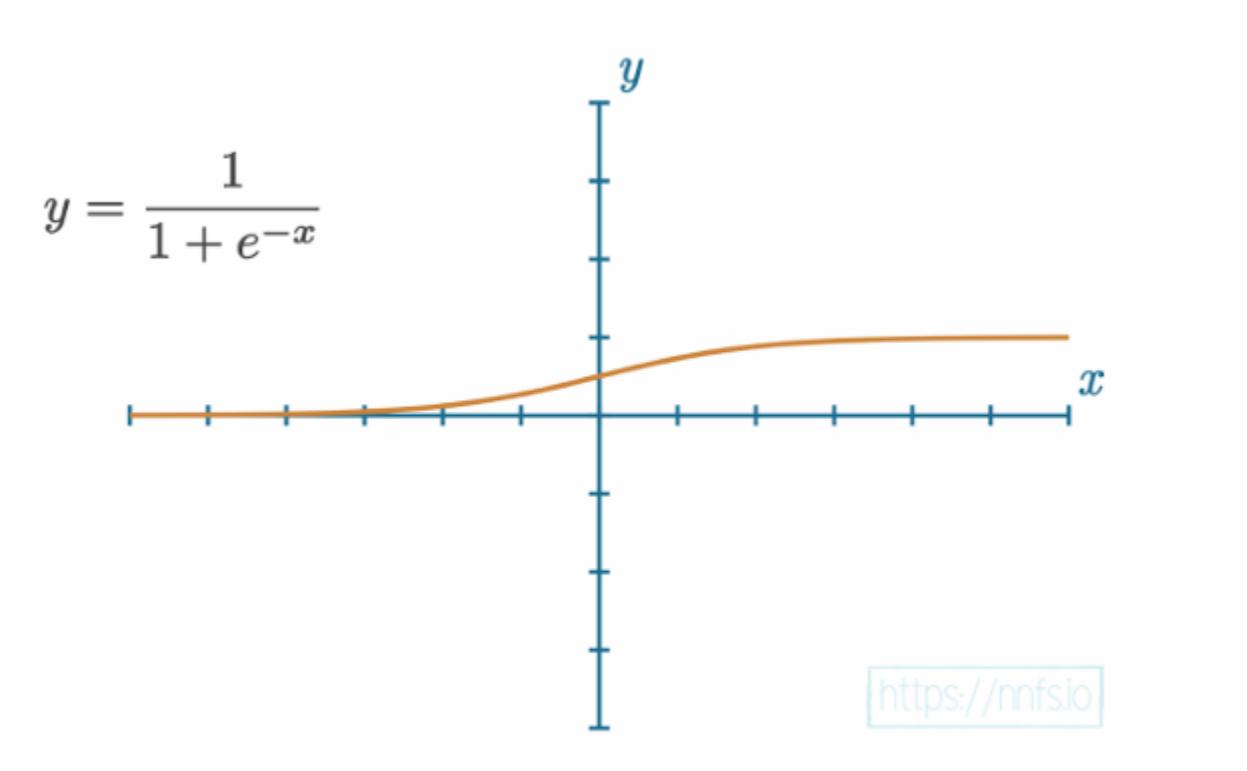


Fig 4.03: Sigmoid function graph.

The Rectified Linear (ReLU) Activation Function

- y=x, clipped at 0 from the negative side.
- If x is less than or equal to 0, then y is 0 — otherwise, y is equal to x.
- Simple, powerful, most widely use – mainly speed and efficiency.
- Extreme ly close to being a linear activation but non linear due to bend after 0

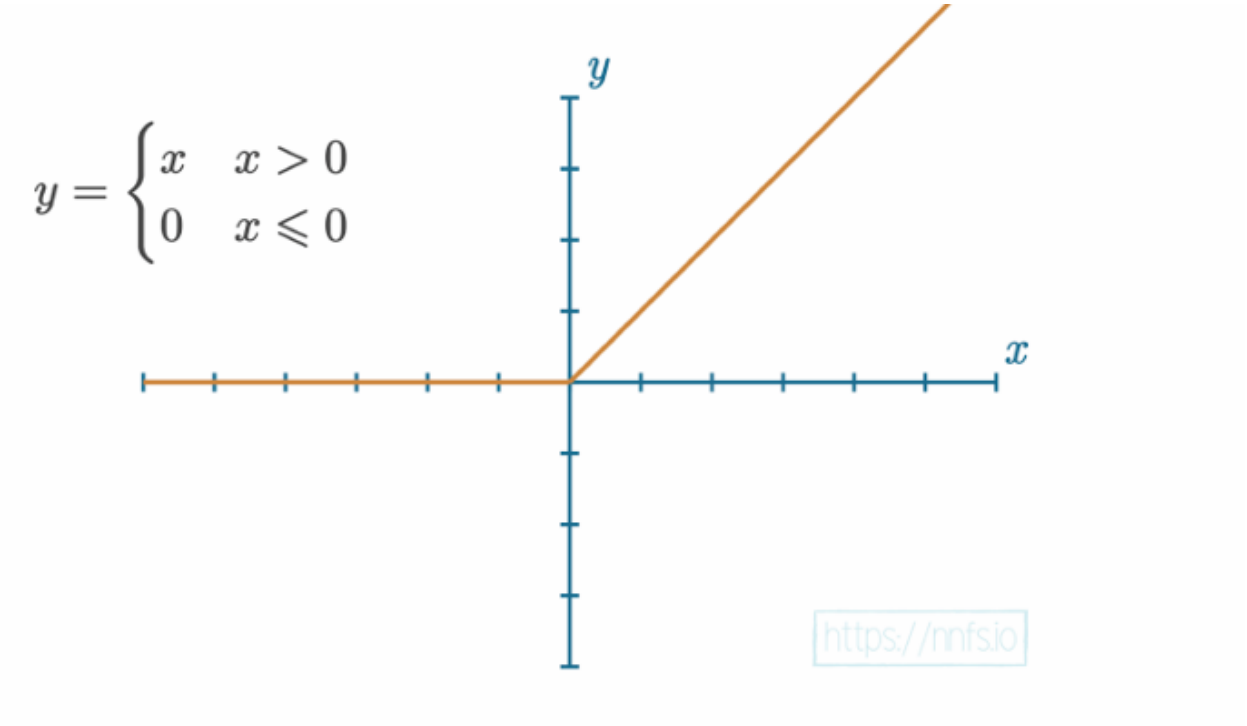


Fig 4.04: Graph of the ReLU activation function.

Why is non-linear function essential to the neural network?

To introduce non-linearity into the model, which helps to learn complex patterns and approximate any functions unlike the linear function. Thus, helps to learn complex, real-world data with any flexibility.

So, for a neural network to fit a nonlinear function, it needs two or more hidden layers, each with nonlinear functions.

The Softmax Activation Function

Why are we bothering with another activation function? It just depends on what our overall goals are.

The rectified linear unit is unbounded, not normalized with other units, and exclusive.

> "Not normalized" implies the values can be anything, an output of [12, 99, 318] is without context
>
> "exclusive" means each output is independent of the others

SoftMax activation on the output data can take in non-normalized, or uncalibrated, inputs and produce a normalized distribution of probabilities for our classes.

This distribution returned by the softmax activation function represents confidence scores for each class and will add up to 1. The predicted class is associated with the output neuron that returned the largest confidence score.

Here's the function for the **Softmax**:

$$S_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{L} e^{z_{i,l}}}$$

Exponentiation in Softmax:

Exponentiation serves multiple purposes. To calculate the probabilities, we need non-negative values. Imagine the output as [4.8, 1.21, -2.385] — even after normalization, the last value will still be negative since we'll just divide all of them by their sum. A negative probability (or confidence) does not make much sense. An exponential value of any number is always non-negative — it returns 0 for negative infinity, 1 for the input of 0, and increases positive values.

We also included a subtraction of the largest of the inputs before we did the exponentiation to prevent the exploding values.

Let's see some examples of how and why this can easily happen:

It doesn't take a very large number, in this case, a mere 1,000, to cause an overflow error. So, subtracting the highest values prevents this explosion/ overflow error.

NOTE:

Axis 0 refers to the rows.

Axis 1 refers to the columns.