DATA SCIENCE INFINITY

COMMON ACTIVATION FUNCTIONS

In Deep Learning - an Activation Function is essentially some logic that determines whether a neuron will be "activated" ...

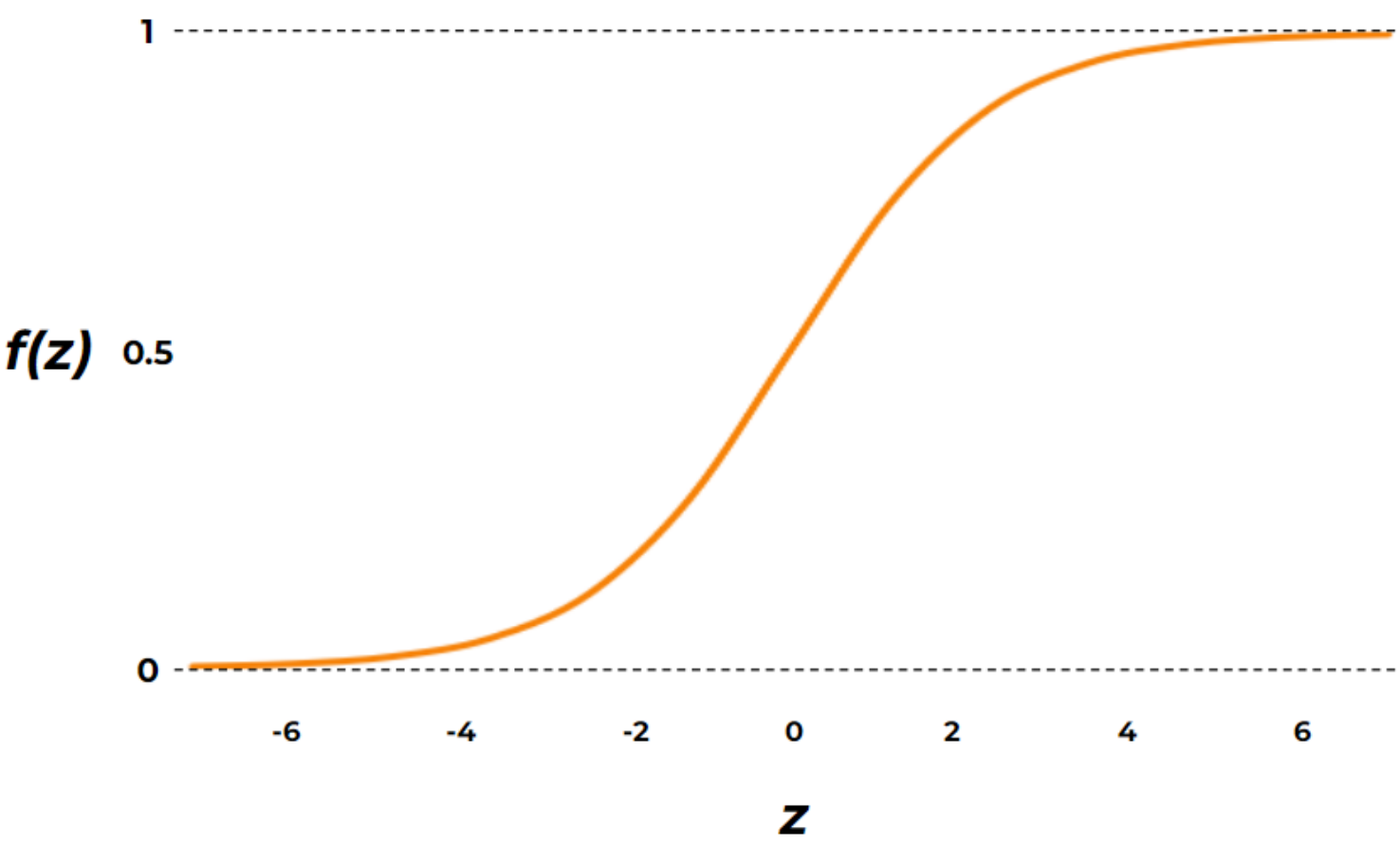
...or if it will not.

In more mathematical terms - it determines what value a neuron will pass onward to neurons in a subsequent layer.

There are many different Activation Functions that can be used - let's take a look!

SIGMOID

The Sigmoid Activation Function squashes an input (z) to an output value somewhere between 0 and 1

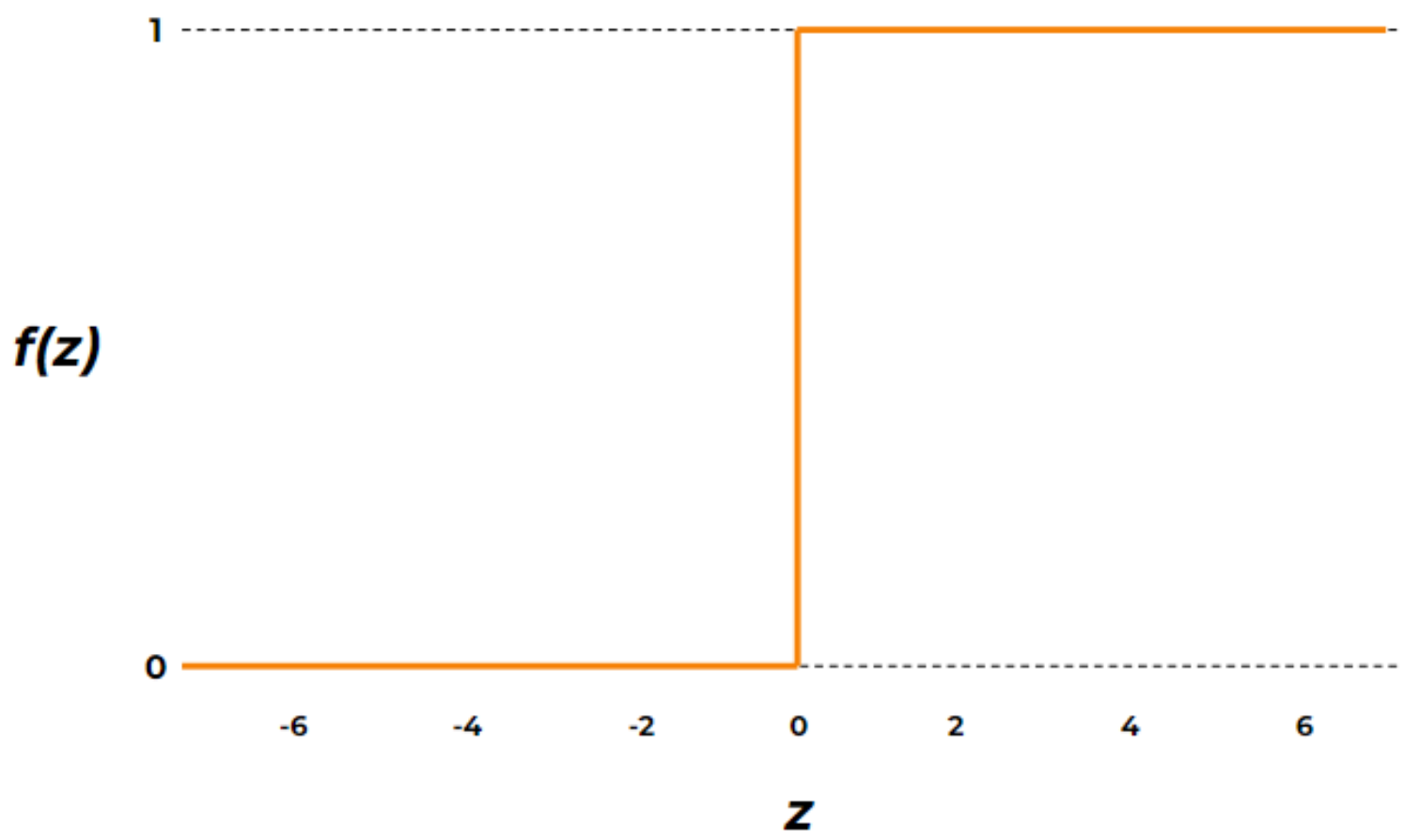


$$f(z) = \frac{1}{1 + e^{-z}}$$

This function is often used in the output layer of Binary Classification and Multi-Label Classification problems

STEP

The Heaviside Step Function provides only two possible output values, 1 or 0. If the input value is >= 0 then the output is 1, otherwise the output is 0

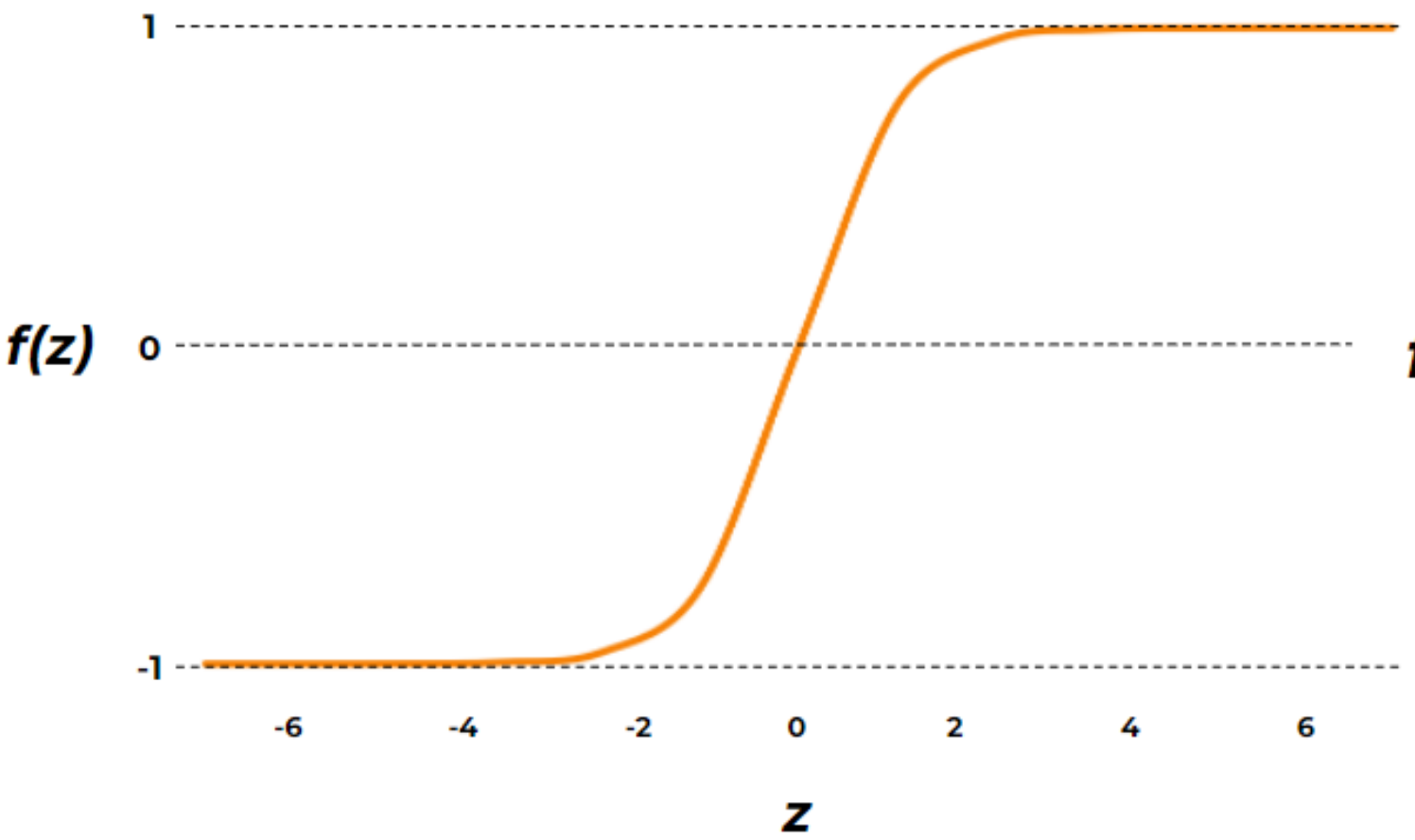


$$f(z) = \begin{cases} 1 \text{ if } z >= 0 \\ 0 \text{ if } z < 0 \end{cases}$$

Used in early perceptrons. This function is non-differentiable at z = 0 and it's derivative is 0 everywhere else meaning Gradient Descent can't make any progress in optimising the weight values of the network

TANH

The Hyperbolic Tangent (Tanh) Function has a similar "S shape" to the Sigmoid Function...

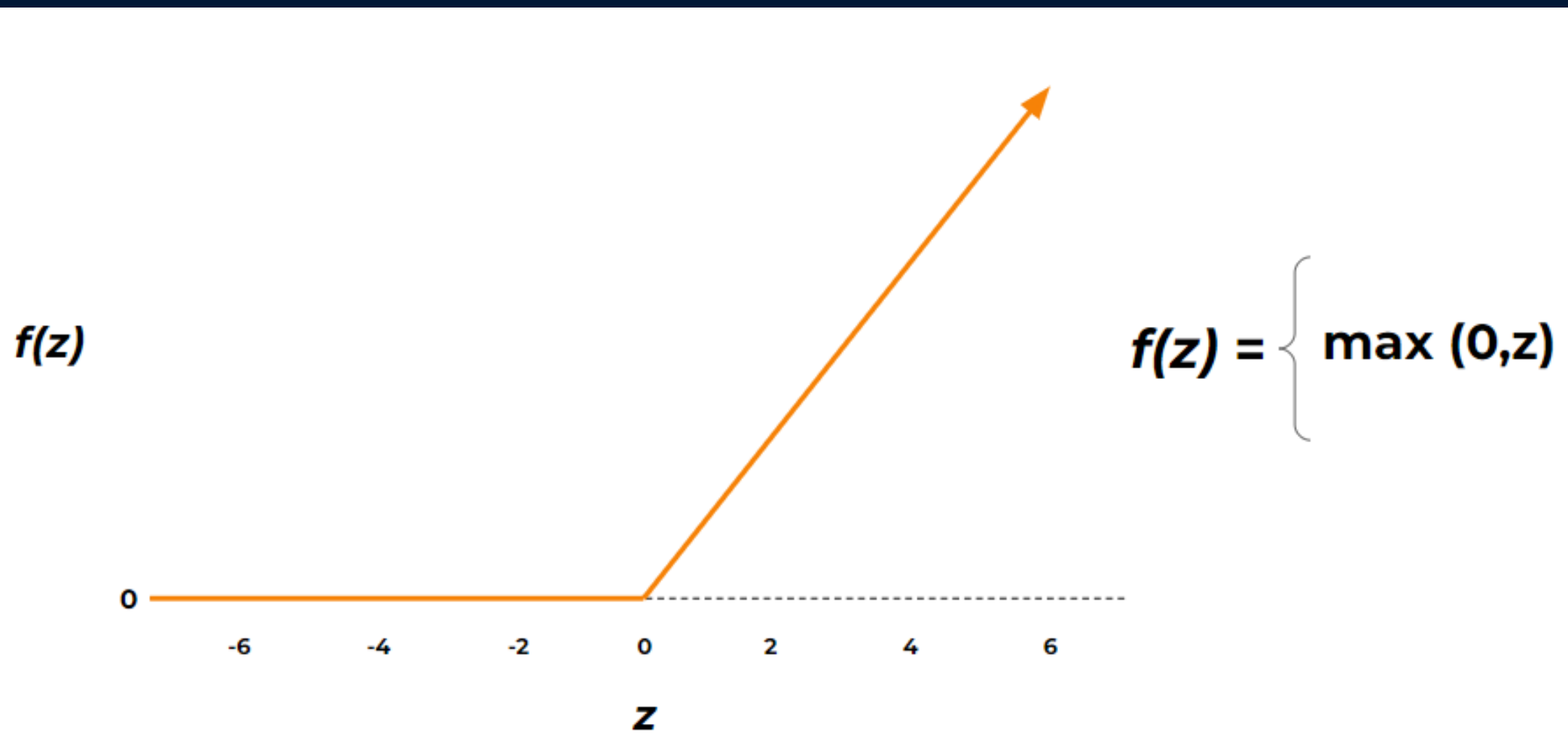


$$f(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

...however Tanh instead squashes an input (z) to an output value somewhere between -1 and +1 rather than 0 and 1

RELU

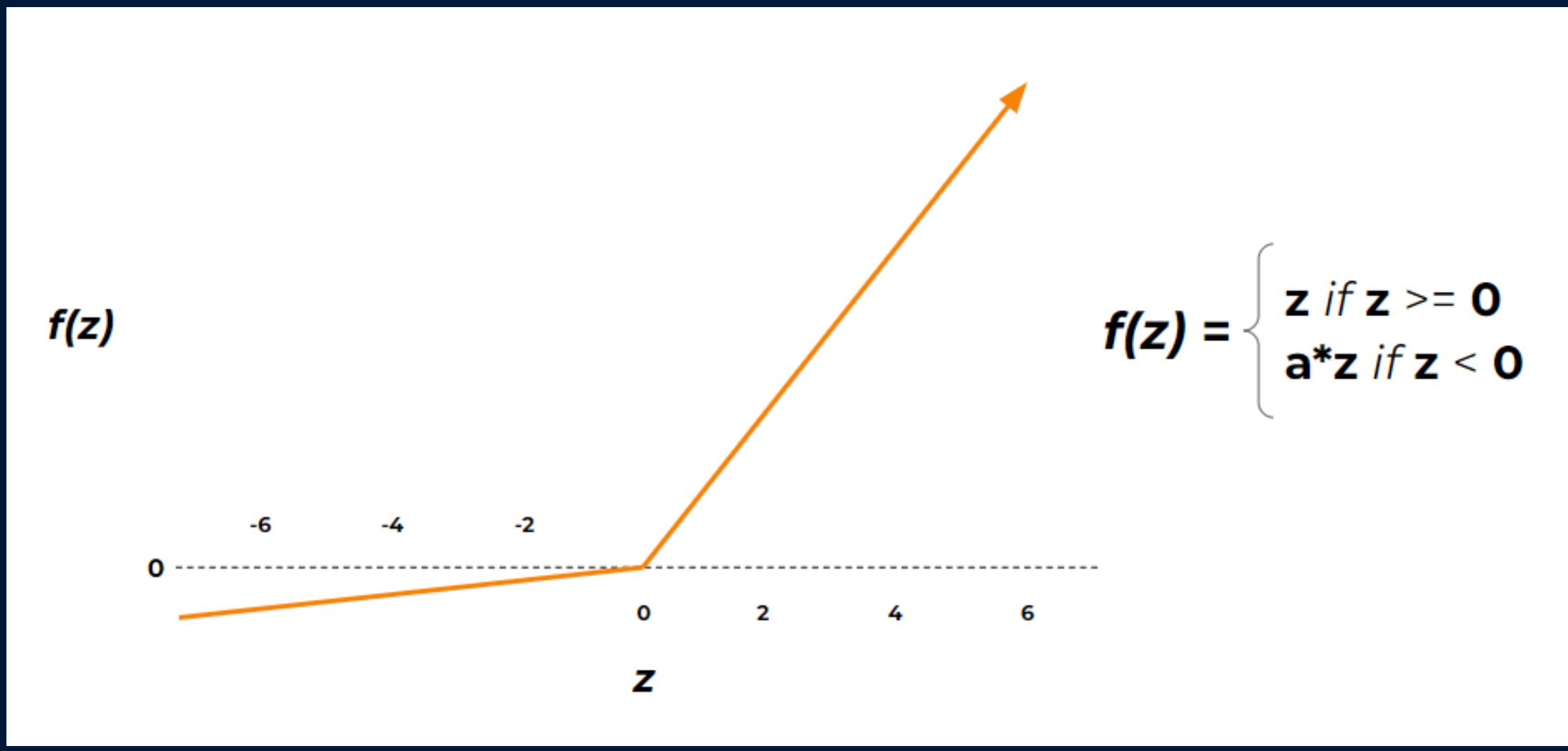
The Rectified Linear Unit (ReLU) Function is very popular for hidden layer neurons due to speed and performance



Negative input values are output as 0 while positive input values remain as they are.

LEAKY RELU

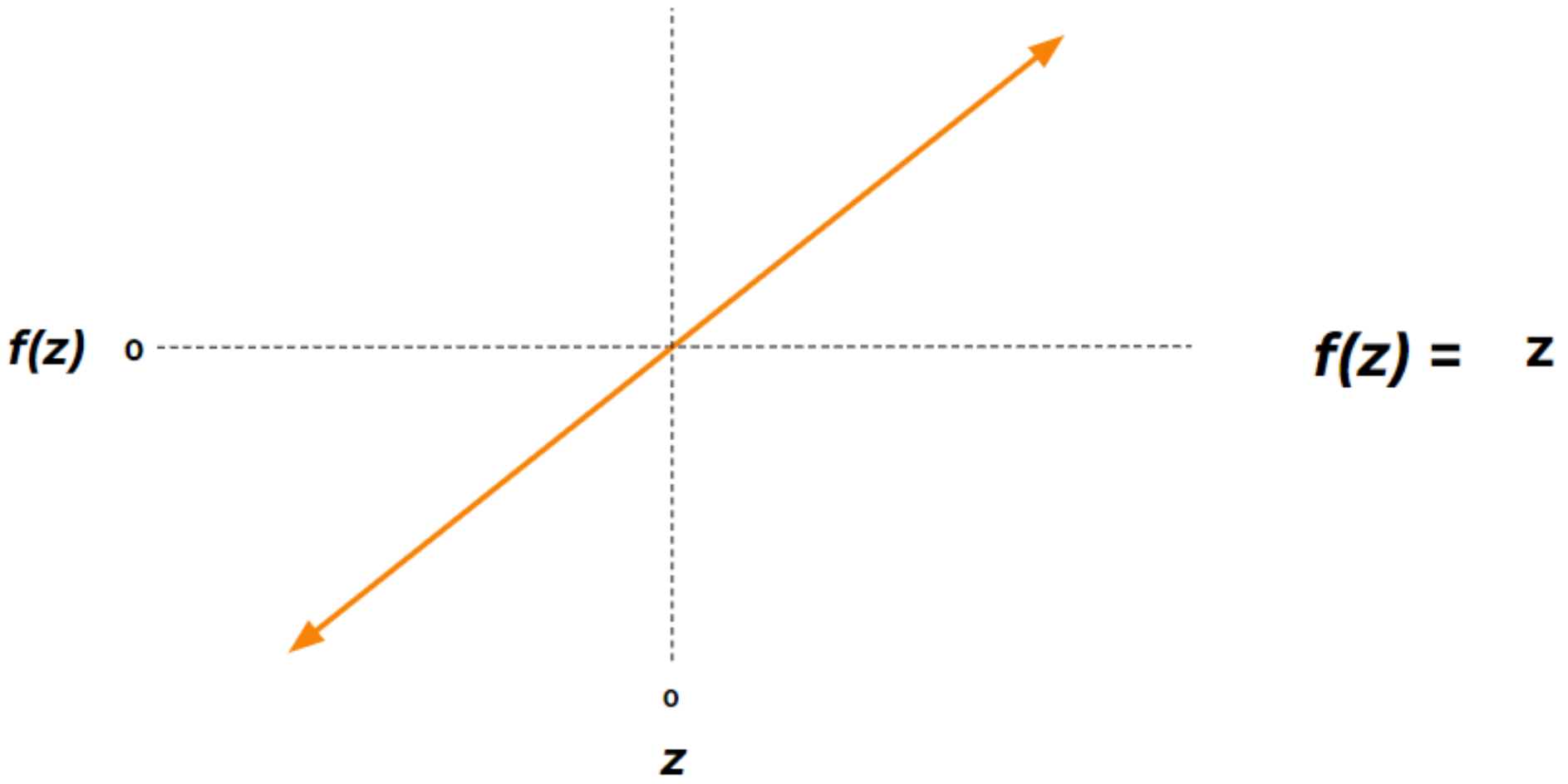
Leaky ReLU is very similar to ReLU but allows some small negative slope for input values that are negative



Leaky ReLU was created to deal with the "dying ReLU problem" which comes from standard ReLU forcing all negative inputs to be output with a value of 0

LINEAR

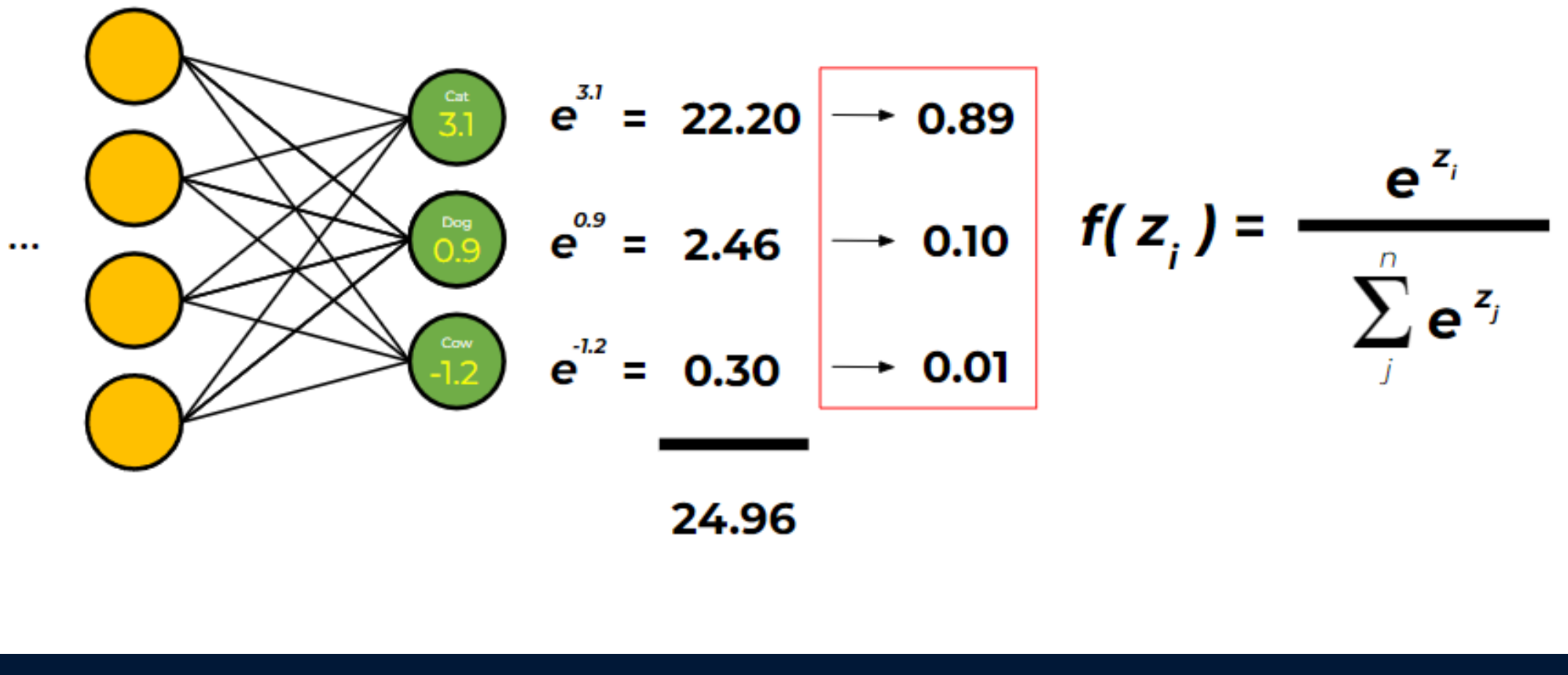
With a Linear Activation Function the output equals the input - there is no capping or transforming



This is often applied in the output layer of a Regression task where we require an unbounded numeric value

SOFTMAX

The Softmax Function turns a vector of numbers into a vector of probabilities (that add up to a total of 1)



This is often applied in the output layer of a Multi-Class Classification task where we would like a probability of an obversation belonging to each class

COMMON USAGE

Below is a guide for commonly utilised Activation Functions for different problem types...

Problem Type	**Hidden Layer** Activation Function	**Output Layer** Activation Function
Regression	RELU	Linear
Binary Classification	RELU	Sigmoid
Multi-Class Classification	RELU	Softmax
Multi-Label Classification	RELU	Sigmoid

Important Note: These are very simply commonly used approaches - others can, and will be applied!