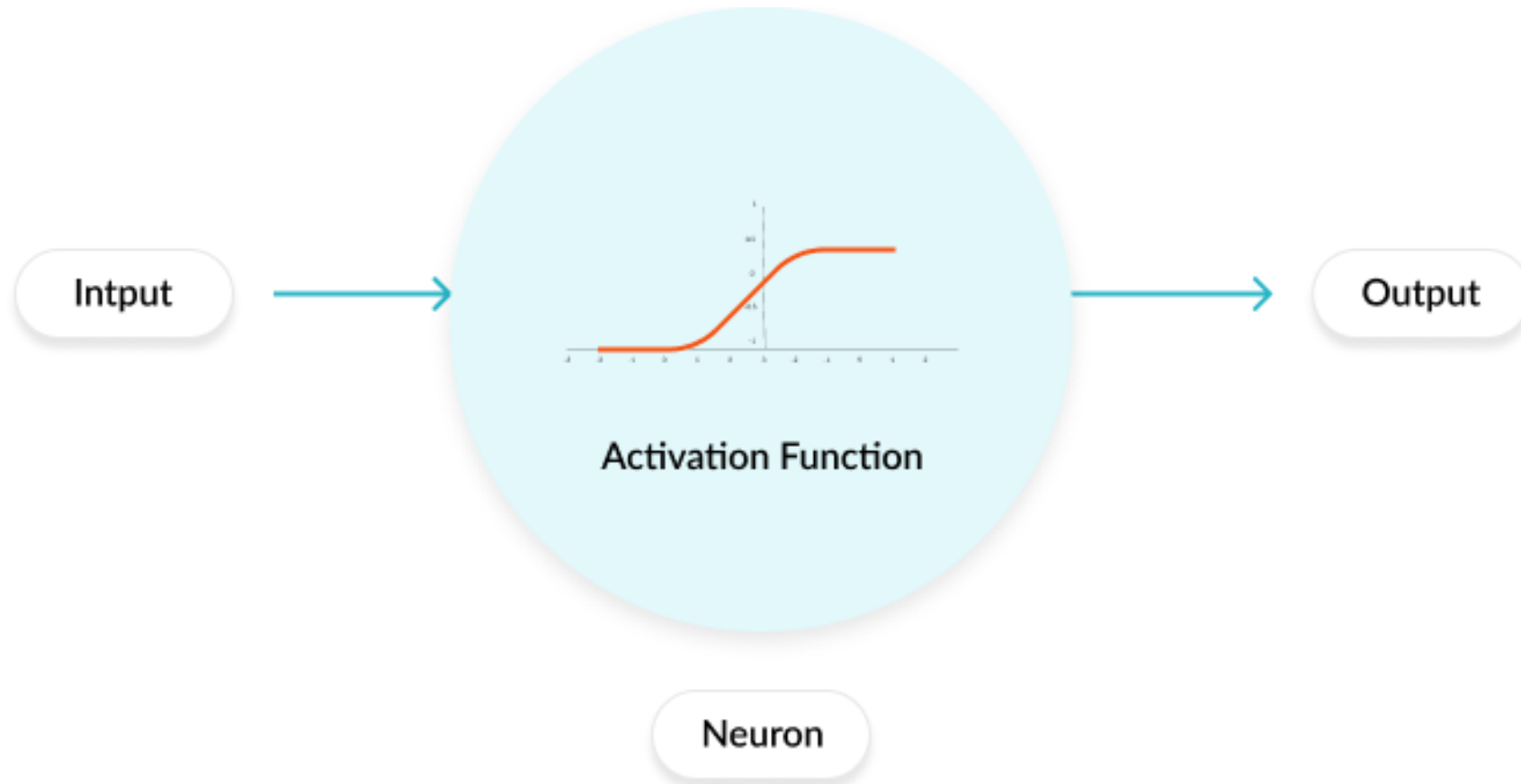


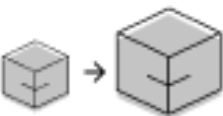
Activation Function

Alok



- <https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba>
- <https://medium.com/analytics-vidhya/demystifying-neural-networks-a-mathematical-approach-part-1-4e10bed61400>
- <https://medium.com/analytics-vidhya/demystifying-neural-networks-a-mathematical-approach-part-2-e48bb8611661>
- <https://medium.com/deep-math-machine-learning-ai/chapter-7-artificial-neural-networks-with-math-bb711169481b>

- The basic process carried out by a neuron in a neural network is:



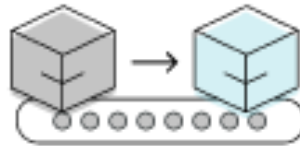
Take input and multiply by the neuron's weight.

The diagram shows a small gray cube with a plus sign on its front face, followed by an arrow pointing to a larger gray cube with a plus sign on its front face.




Add bias*

The diagram shows a gray cube with a plus sign on its top face and a minus sign on its front face.



Feed the result, x , to the activation function: $f(x)$

The diagram shows a gray cube with a plus sign on its front face, followed by an arrow pointing to a light blue cube with a plus sign on its front face. Below the cubes is a horizontal row of eight small gray circles.



Take the output and transmit to the next layer of neurons.

The diagram shows a light blue cube with a plus sign on its front face, surrounded by several small gray circles.

Each neuron's output is the input of the neurons in the next layer of the network, and so the inputs cascade through multiple activation functions until eventually, the output layer generates a prediction.

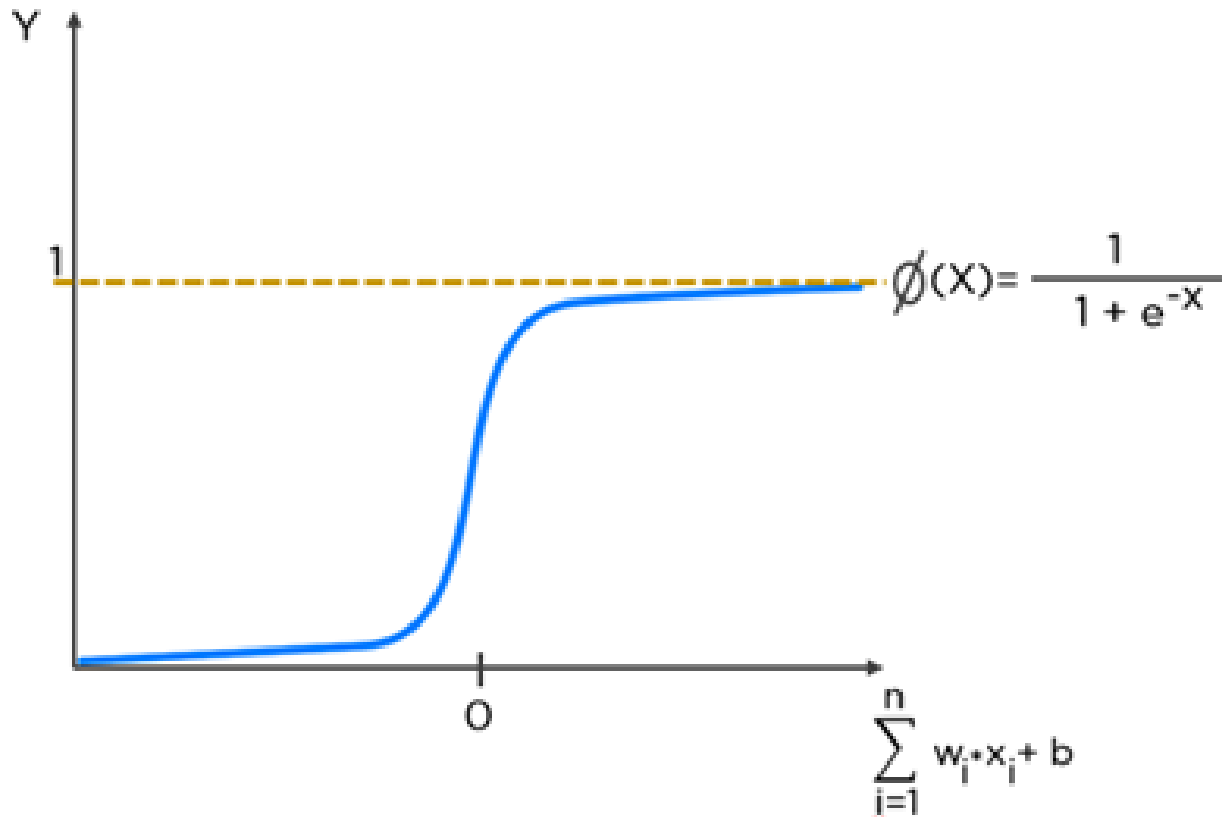
Neural networks rely on nonlinear activation functions—the derivative of the activation function helps the network learn through the backpropagation process

7 Common Activation Functions

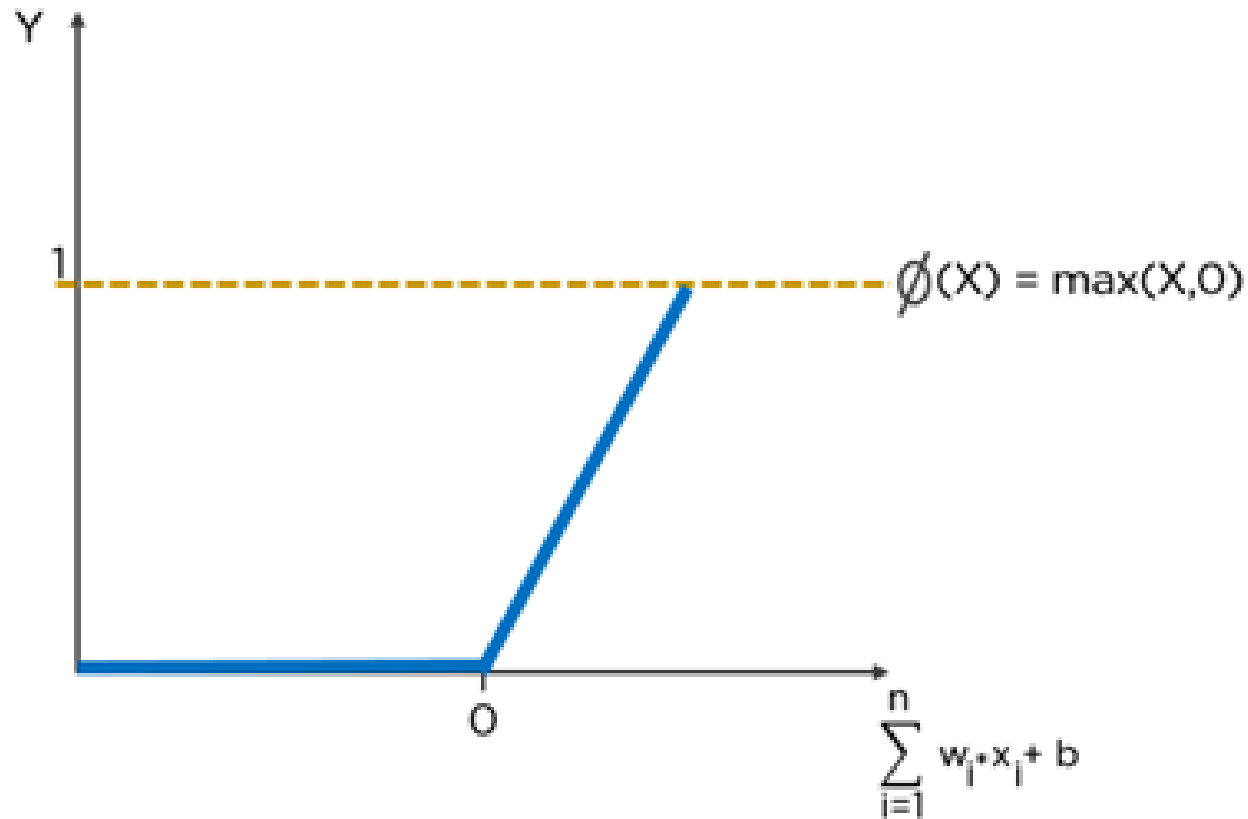
- **The sigmoid function** has a smooth gradient and outputs values between zero and one. For very high or low values of the input parameters, the network can be very slow to reach a prediction, called the *vanishing gradient* problem.
- **The TanH function** is zero-centered making it easier to model inputs that are strongly negative strongly positive or neutral.
- **The ReLu function** is highly computationally efficient but is not able to process inputs that approach zero or negative.
- **The Leaky ReLu** function has a small positive slope in its negative area, enabling it to process zero or negative values.
- **The Parametric ReLu** function allows the negative slope to be learned, performing backpropagation to learn the most effective slope for zero and negative input values.
- **Softmax** is a special activation function use for output neurons. It normalizes outputs for each class between 0 and 1, and returns the probability that the input belongs to a specific class.
- **Swish** is a new activation function discovered by Google researchers. It performs better than ReLu with a similar level of computational efficiency.

- There is no rule in the field of neural networks.
- It all depends on your data and in what form you want the data to be transformed after passing through the activation function.
- Activation functions decide whether a node should be fired or not. The node which gets fired depends on the Y value.
- Generally speaking, there are 3 activation functions that are regularly used to construct a neural network.
- These are: **Sigmoid function**, **ReLU** (Rectified Linear Unit) function and **Softmax function**.

- A Sigmoid function is often used in the hidden layers where you are trying to predict the probability as an output (0 or 1).
- This is how the graph of a sigmoid function looks like:

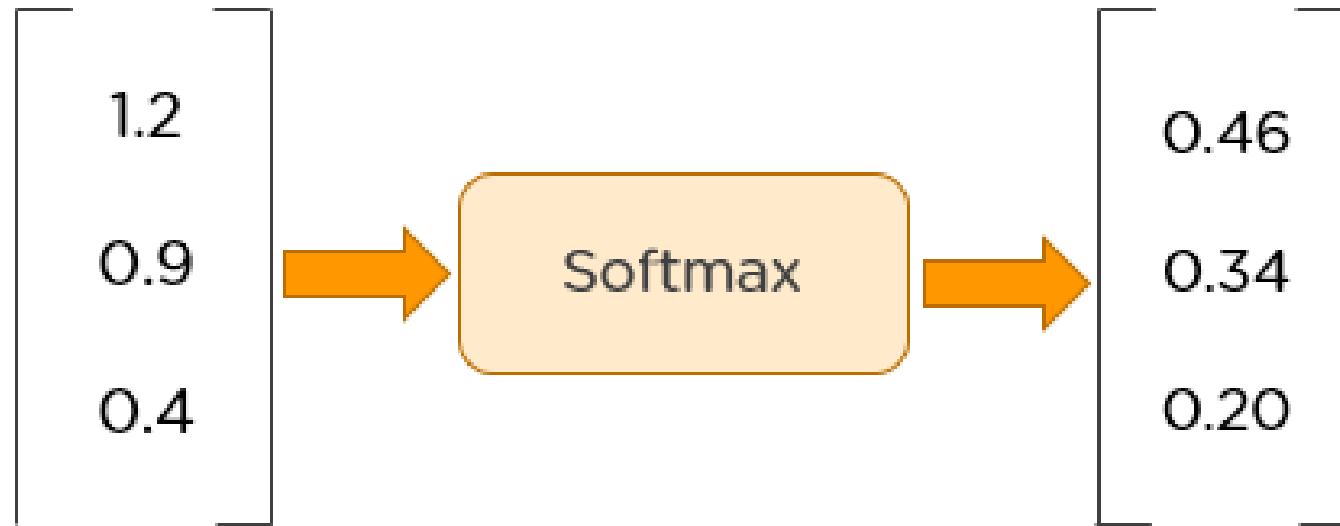


- ReLU is one of the most widely used Activation function and gives an output of X if X is positive and 0 otherwise. It is also used in the hidden layers. The looks like the following:

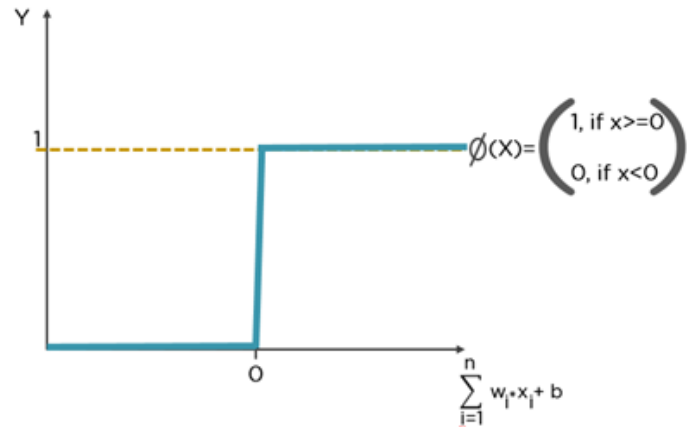


- Softmax function is always used in the output layers that generates the output between 0 and 1.
- It divides each output such that the total sum of the outputs is equal to 1.

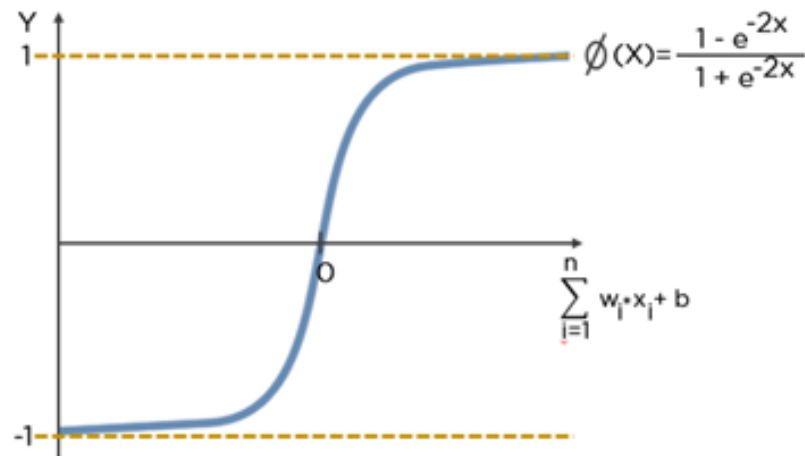
$$\text{Softmax}(L_n) = \frac{e^{L_n}}{\|e^L\|}$$



- There are several other activation function that are used for general regression and classification problems like Step function and Tanh function.
- A Step function is a threshold based activation function. If the Y value is greater than a certain value (threshold), the function is activated and fired else not.



A Tanh function is used in both hidden as well as in the output layers. This function is similar to Sigmoid function and is bound to range $(-1, 1)$.



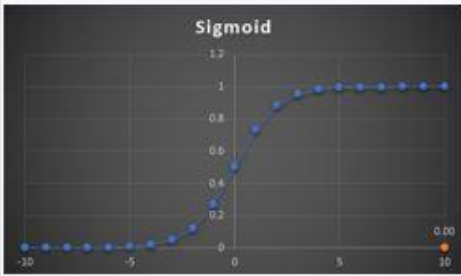
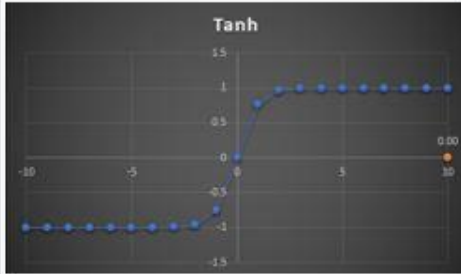
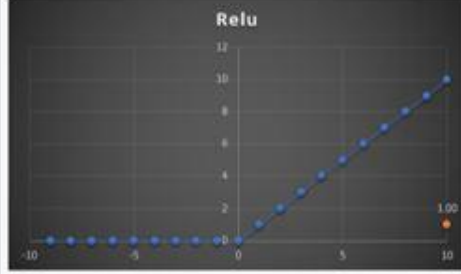

- The activation function of the last layer is often tied to your kind of problem: is it regression, classification? Often softmax are used for classifying between classes having an idea of the probabilities of each class.
- On the other hand, the activation functions of hidden layers is an interesting and evolving topic. Non-linearities allow the network to learn an highly non linear function, and in theory every one could be good for this goal.

- Which is the right Activation Function?
- We have seen many activation functions, we need some domain knowledge to know which is the right activation function for our model.
- Choosing the right activation function depends on the problem that we are facing, there is no activation function which yields perfect results in all the models.

- Sigmoid functions and their combinations usually work better for classification techniques ex. Binary Classification 0s and 1s.
- • Tanh functions are not advised or implemented because of the dead neuron problem.
- • ReLU is a widely used activation function and yields better results compared to Sigmoid and Tanh.
- • Leaky ReLU is a solution for a dead neuron problem during the ReLU function in the hidden layers.
- There are other activation functions like softmax, selu, linear, identity, soft-plus, hard sigmoid etc which can be implemented based on your model.

- Sigmoid functions and their combinations generally work better in the case of classifiers
- Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- ReLU function is a general activation function and is used in most cases these days
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice
- Always keep in mind that ReLU function should only be used in the hidden layers
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn't provide with optimum results

- <https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f>
- <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

| Name | Plot | Equation | Derivative |
|--|---|--|--|
| Sigmoid |  | $f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ | $f'(x) = f(x)(1 - f(x))$ |
| Tanh |  | $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ | $f'(x) = 1 - f(x)^2$ |
| Rectified Linear Unit (relu) |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Leaky Rectified Linear Unit (Leaky relu) |  | $f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |