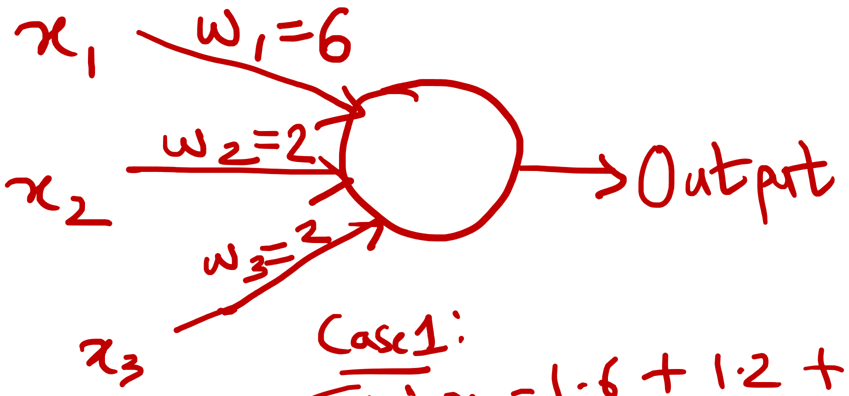
Basic Structure of NN - History

PERCEPTRON



$$\text{Output} = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} w_i \cdot x_i \leq \text{threshold} \\ 1 & \text{if } \sum_{i=1}^{n} w_i \cdot x_i > \text{threshold} \end{cases}$$

Case1:

$\sum w_i \cdot x_i = 1\cdot 6 + 1\cdot 2 + 1\cdot 2 = 6+2+2 = 10$ ∴ Neuron will fire.

Case2: $= 0\cdot 6 + 1\cdot 2 + 1\cdot 2 = 0+2+2 = 4$ ∴ Neuron will NOT fire.

Note: Weights & Threshold both are Real Numbers.

	x_1	x_2	x_3
Case 1:	1	1	1
Case2:	0	1	1

Threshold = 5

Suppose,
① You absolutely adore BBQ food, so much that even if ur friend is uninterested & it is hard to get to, BUT You really hate Bad Weather.

Basic Structure of NN - History

Suppose the weekend is coming up & U hv heard that there is going to be a Barbeque Festival in your city, & you are trying to decide whether or not to go to the festival.

Based on 3 factors:

1. Is the weather good? (Not too hot or too rainy)
2. Does your GF/BF/Wife/Husband want to accompany you?
3. Is the festival near some railway stn or Metro? (Assume: U dont hv a CAR)

Basic Structure of NN - History

Simpler way to represent weights & inputs

$$\boxed{\sum_{i=1}^{n} w_i \cdot x_i > \text{Threshold}}$$

Dot product of

$$\boxed{\sum_{i=1}^{n} w_i \cdot x_i \equiv w \cdot x}$$

$$\text{Output} = \begin{cases} 0 & \text{if } w \cdot x \leq Th. \\ 1 & \text{if } w \cdot x > Th. \end{cases}$$

$$w \cdot x \leq Th$$

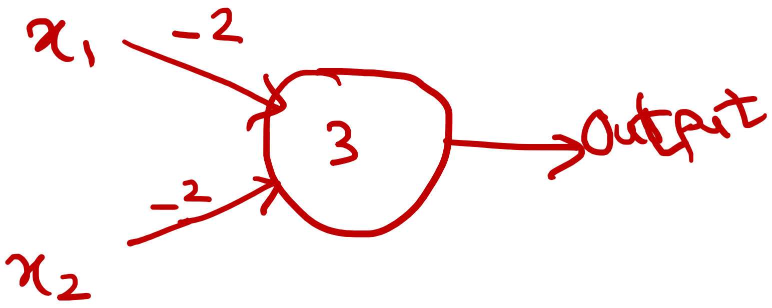
$$\therefore \; w \cdot x - Th. \leq 0$$

Let bias $\boxed{b \equiv -\text{Threshold}}$

$$\boxed{\text{Output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}}$$

Simple example on using weights

eg: We hv a Perceptron with two inputs, each with weight = -2 & an overall bias of 3.



Input

	x_1	x_2	o/p of Neuron
1	0	0	FIRE
2	0	1	FIRE
3	1	0	FIRE
4	1	1	NOT FIRE

$w \cdot x + b$

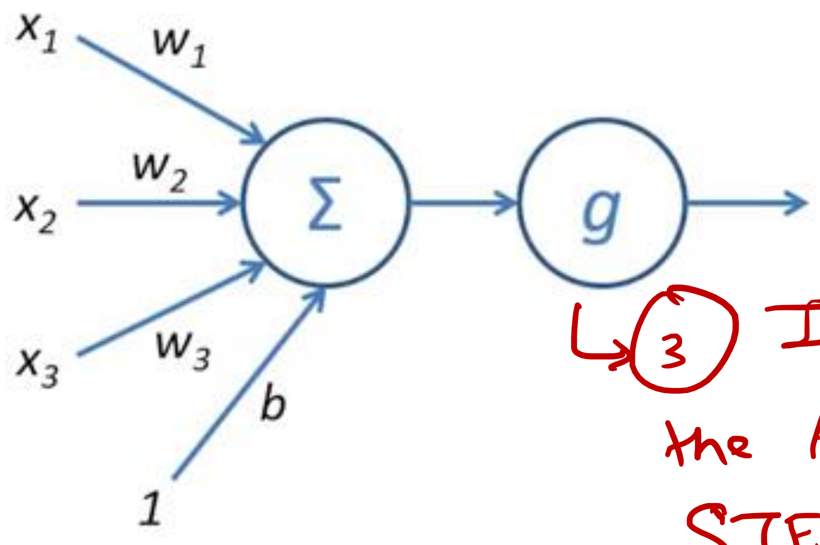
1. $0 \cdot (-2) + 0(-2) + 3 = 0 + 3 = 3$ (+ve)
2. $0(-2) + 1(-2) + 3 = 0 + (-2) + 3 = 1$ (+ve)
3. $1(-2) + 0(-2) + 3 = 1$ (+ve)
4. $1(-2) + 1(-2) + 3 = -2 + (-2) + 3 = -1$ (-ve)

Activation Functions → Secret sauce in a food's recipe

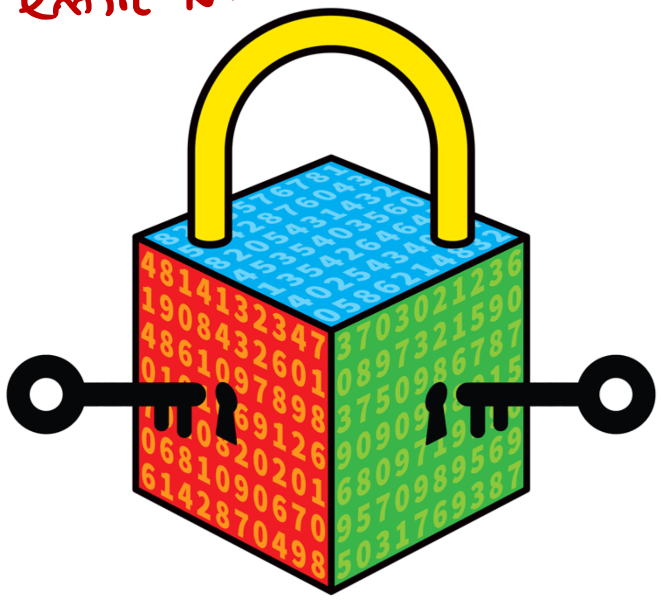
- This is the secret of the Neural Network.

① N.N. Training is all about tuning weights & biases

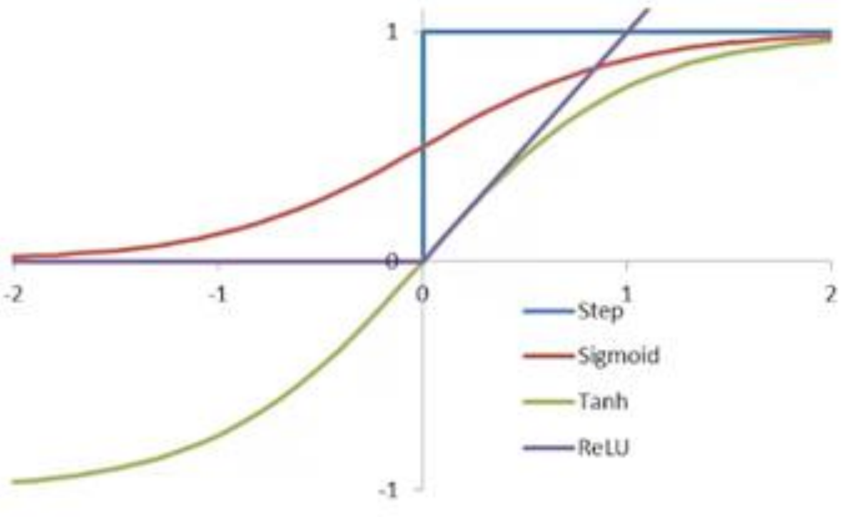
② If there was No activation function 'f', the o/p of entire NN would hv been Linear.

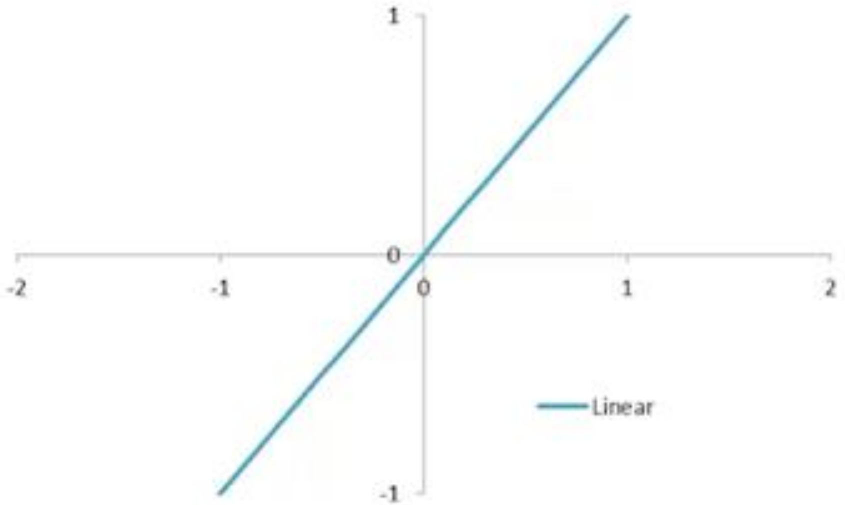


↳ ③ In Perceptron, the A.F. used is STEP FUNCTION.



Types of Activation Functions





Types of Activation Functions

Step : ⓐ Used

- **Step**: original concept behind classification and region bifurcation. Not used anymore

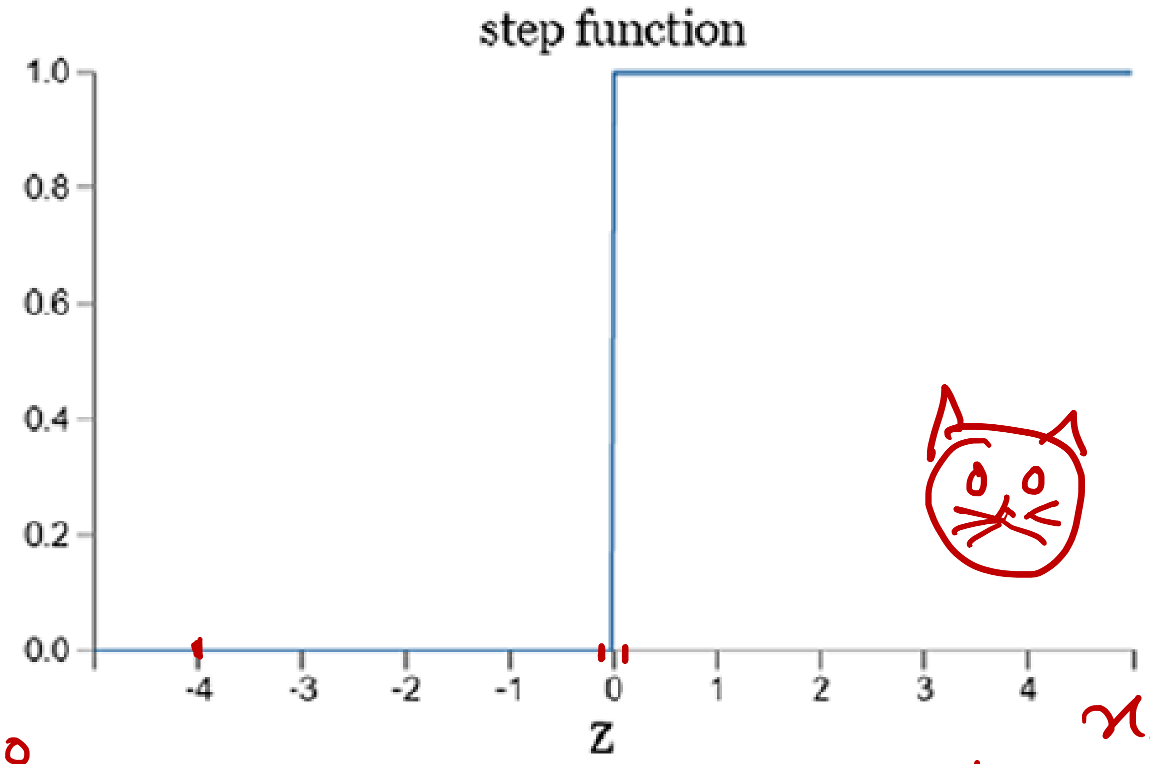
① Classification
② Region Bifurcation.

ⓑ Not Used anymore

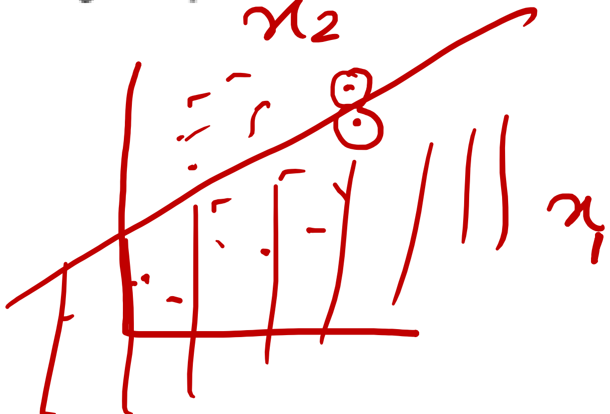
Output $= \sum w_i \cdot x_i + \text{Bias}$
$\approx$ Real No.

$-4 = 0$
$-3.3 = 0$
$-1 = 0$
$-0.5 = 0$

$-0.3 = 0$
$-0.2 = 0$
$-0.1 = 0$
$0 = 0$
$0.1 = 1$



Step function



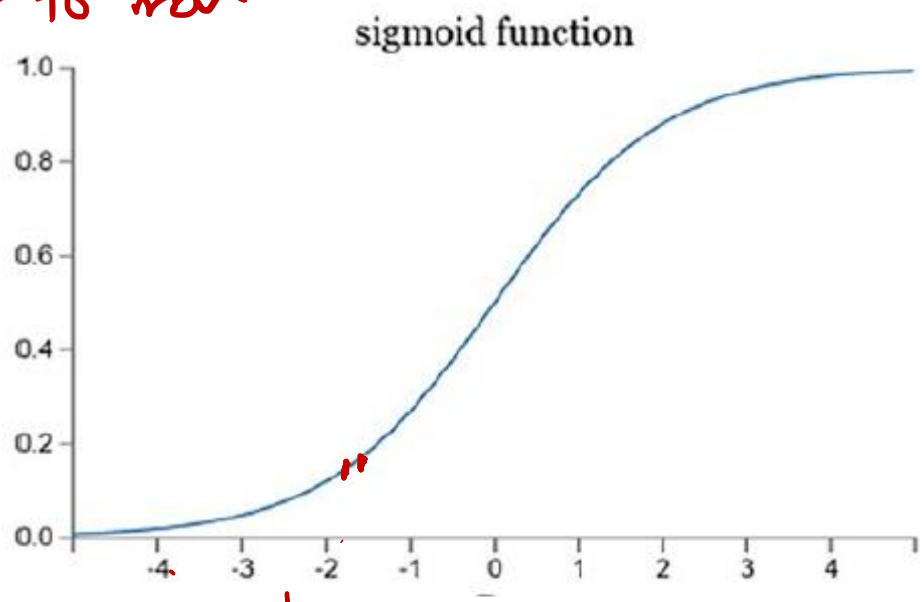
Types of Activation Functions

Problem with Perceptron?

A small change in the weights or biases of any single perceptron in the network can sometimes cause the Output of that perceptron to completely flip from 0 to 1.

Types of Activation Functions

SIGMOID : Direction in which I hv to move



$w \cdot x + b$

$= -3$

$= -2.9$

$= +2.5$

$= -2$

$= -1.7$

→ Used for Smooth output Transitions $= \dfrac{1}{1 + \exp^{-(\Sigma w \cdot x + b)}}$

$z = w \cdot x + b$

$$\boxed{= \frac{1}{1 + e^{-z}}}$$

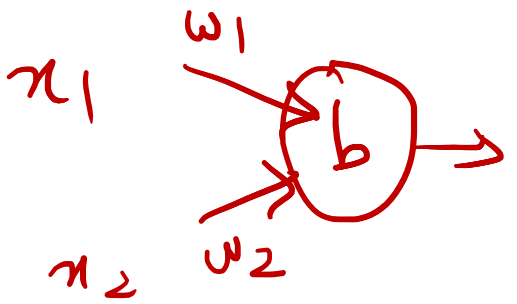
$$= \frac{1}{1 + \exp^{-w \cdot x - b}}$$

Sigmoid are similar to Perceptrons

Just like a perceptron,

the sigmoid neuron has i/ps $x_1, x_2, \dots$

but the inputs here can be any value b/w 0 & 1.

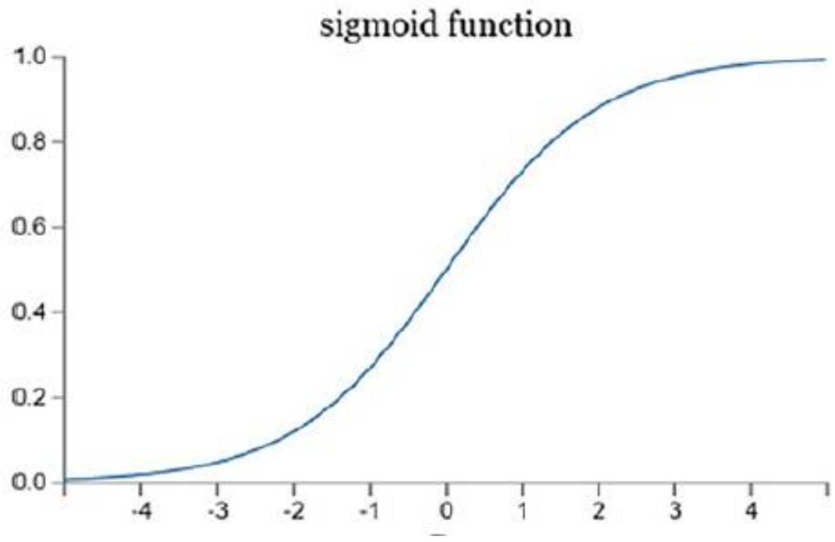


Valid

$= 0.638$ (Yes)

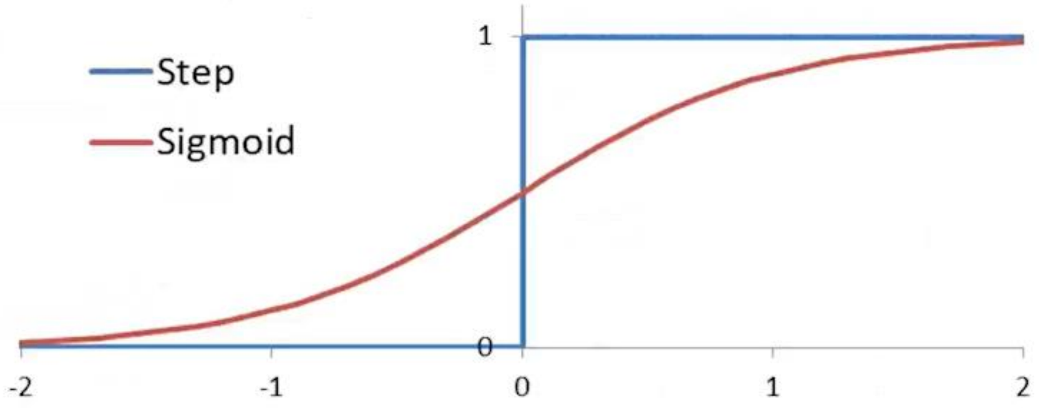
$x_1 = 0/1 \approx$ b/w 0 to 1

$x_2 = 0/1 \approx$ — " —



Types of Activation Functions

- The sigmoid function is a smoother step function.
- Smoothness ensures that there is more information about the direction in which to change the weights if there are any errors.
- Sigmoid function is also mathematically linked to Logistic Regression, which is theoretically well-backed linear classifier.



Drawback of SIGMOID:

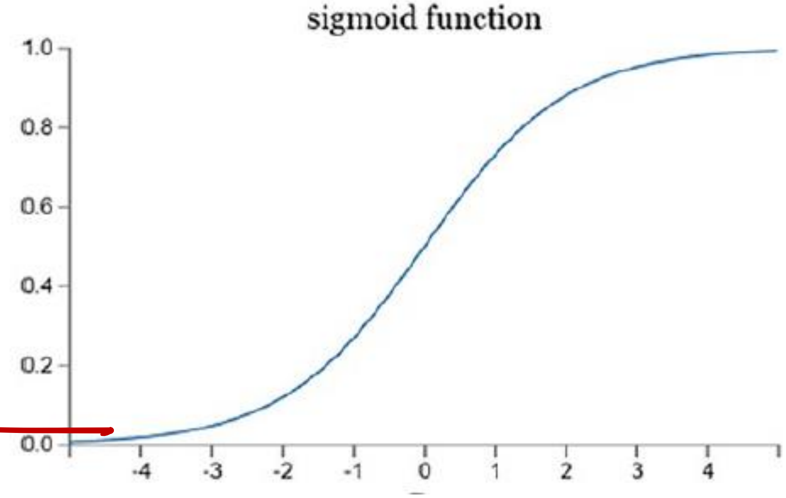
To understand the similarity to perceptron model

① Assume

$z = w \cdot x + b$ is a large +ve No

then $e^{-z} \approx 0$, $\therefore \sigma(z) \approx 1$

$\sigma = \dfrac{1}{1+e^{-z}}$ whr $z = w \cdot x + b$



In other words, when $z = w \cdot x + b$ is large & +ve, o/p from Sigmoid is Approx 1.

② Assume

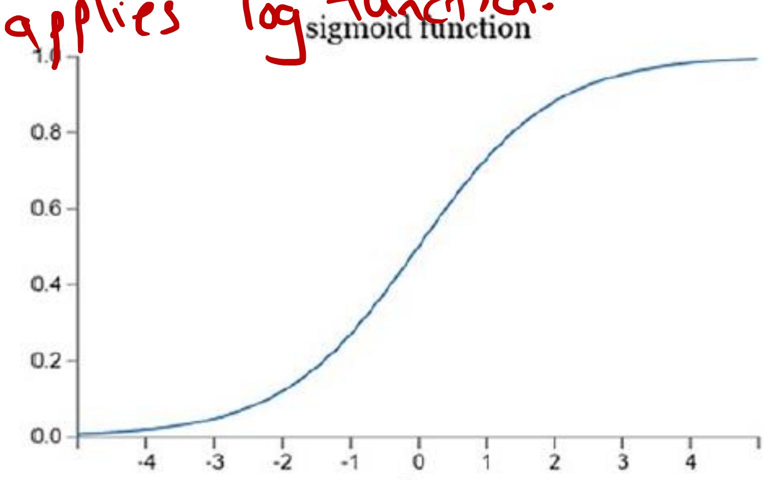
$z = w \cdot x + b$ is a very -ve No

then $e^{-z} \rightarrow \infty$ $\therefore \sigma(z) \approx 0$

Conclusion: When z is either very large or very -ve, the SIGMOID ~~behaves~~ approximates the perceptron.

Problem with Sigmoid / Logit Activation Function.

↳ bcz it applies log function.



Vanishing Gradient Problem:

1. Sigmoid squashes i/p to a v. small range [0,1]
2. ∴ It has very steep Gradient (Slope).
3. ∴ Large changes in i/p cause very small changes in the o/p.

This is referred to as Van. Gr. Prob.

4. V.G.P. ↑ with ↑ # layers

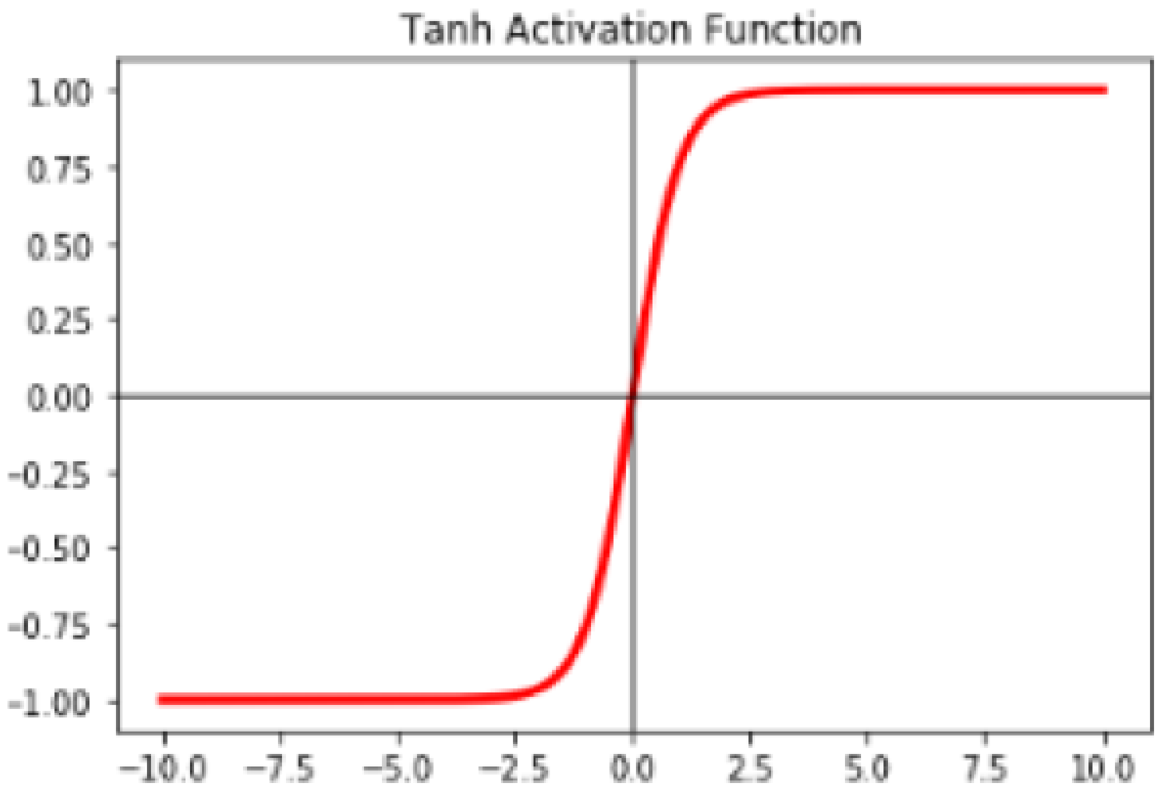
TANH Activation Function

① Sigmoid & Tanh → Qualitatively Same A.F.

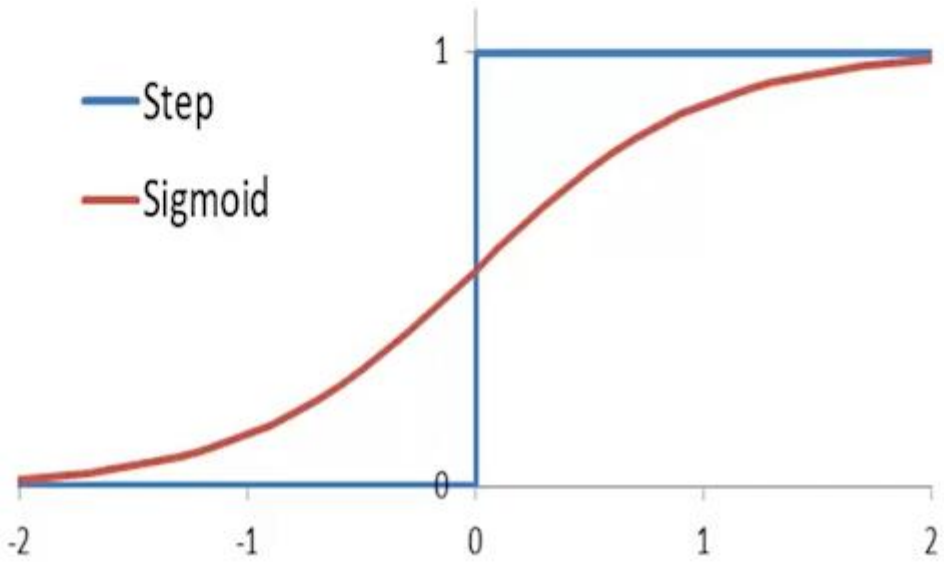
Sigmoid ⇒ 0 to 1

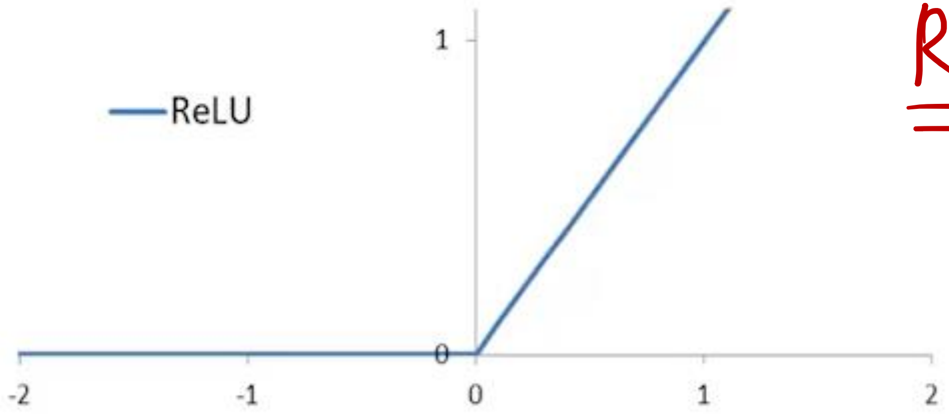
Tanh ⇒ −1 to +1

The tanh(z) function is a rescaled version of the sigmoid, and its output range is [− 1,1] instead of [0,1].



Problem with Sigmoid : Near zero gradient on both extremes.





RELU: Has a constant gradient for half of its inputs

* RELU cannot give meaningful final o/p. ∴ RELU is used only for HIDDEN layer.

ReLU Activation Function

$$Z = \max(0, x_j)$$

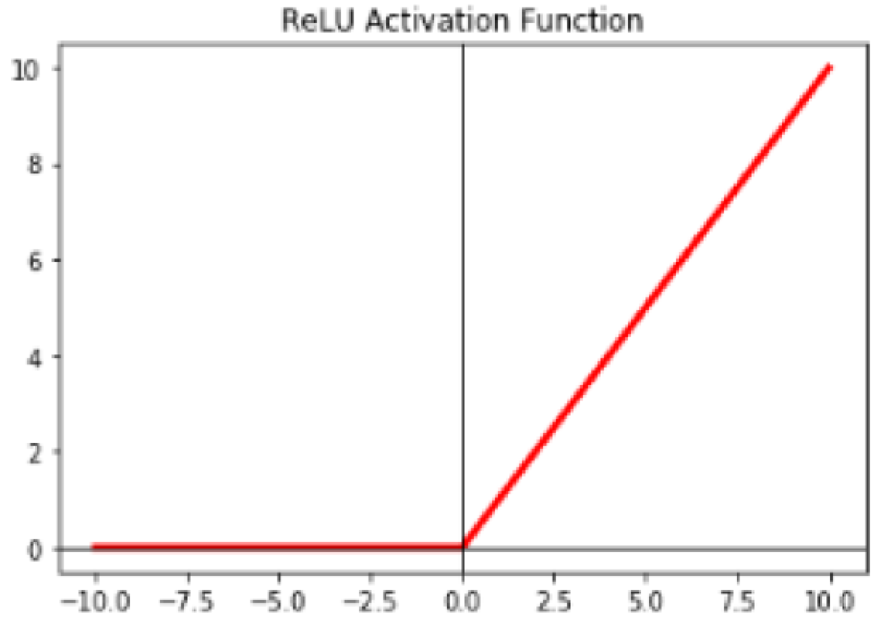
eg: $\max(0, 50) \Rightarrow 50$

50

71.3 $\max(0, 71.3) \Rightarrow 71.3$

-51 $\max(0, -51) = 0$

-1.3 $\max(0, -1.3) \Rightarrow 0$

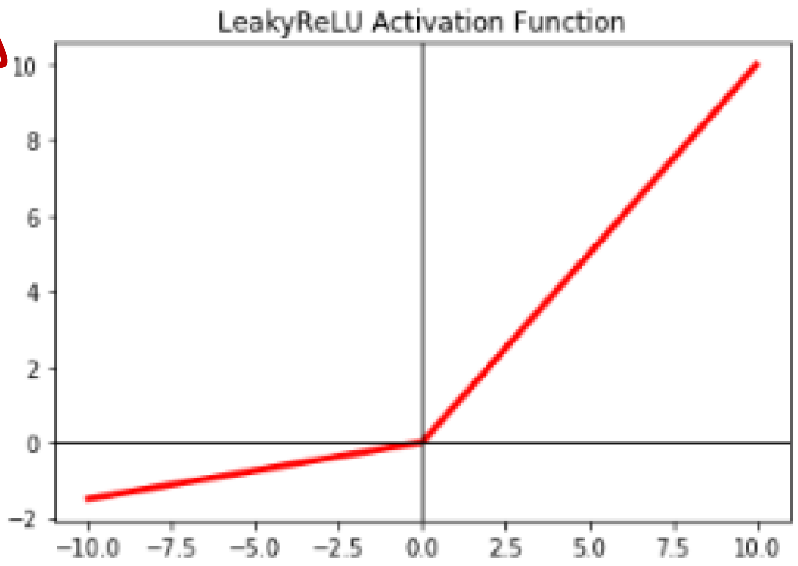


ReLU function graph

1. RELU is never used at o/p Node
2. Now-a-days RELU are v. popular

Rectified Linear Unit viz ReLU

Problem with ReLU: Issue of Dying



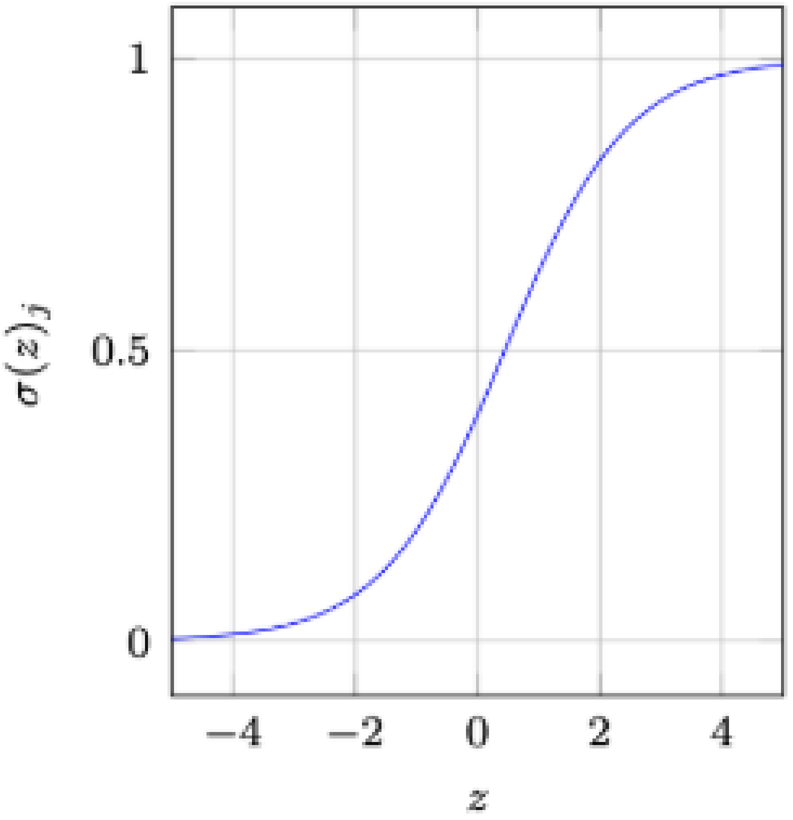
Leaky ReLU

→ introduces a marginally reduced slope (~0.01) for all values of x less than 0.

or Normalized Exponential Function

Softmax Activation Function

$$\sigma(z)_j = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1 \ldots K$$



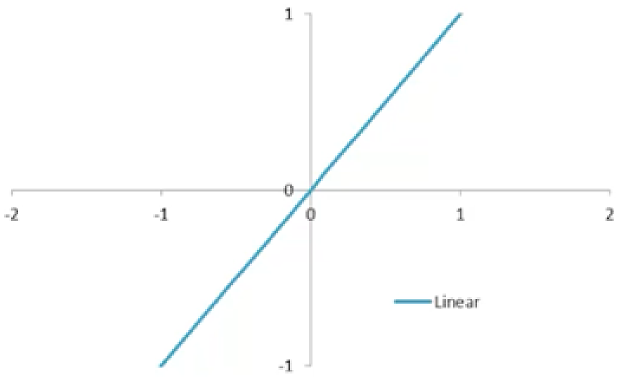
① Softmax is similar to Sigmoid.

② Use both at output Nodes

③ Sigmoid : Binary Cl. Pb.
Softmax : Multi-class Cl. Pb.

④ Softmax is generalized version of sigmoid

Types of Activation Functions

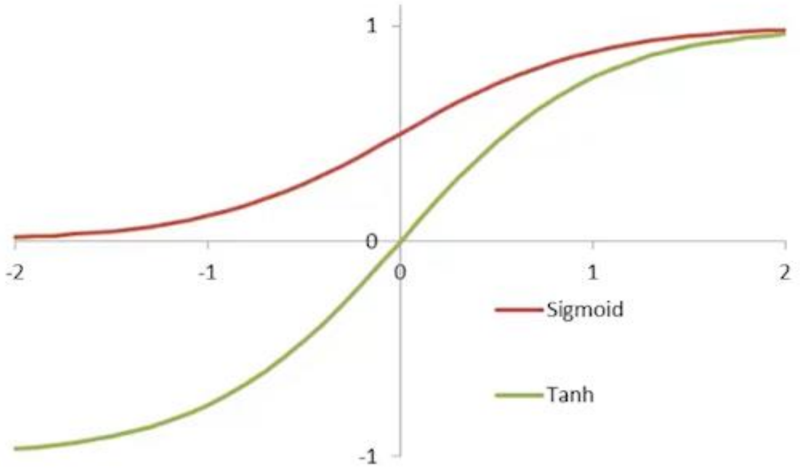


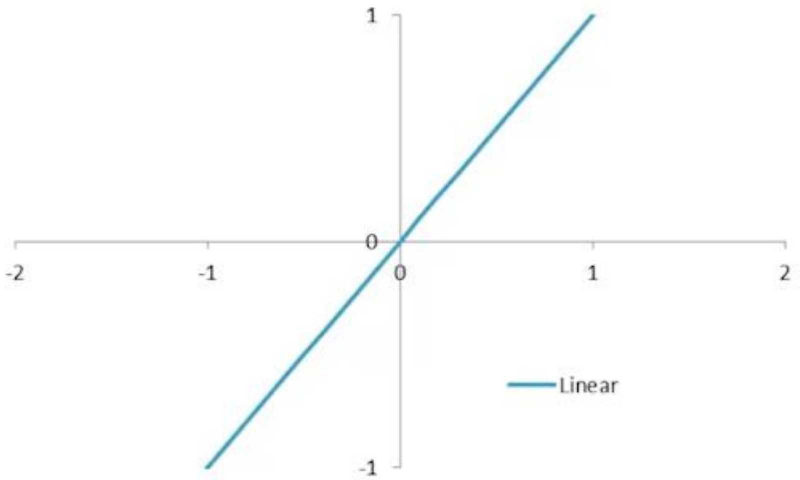
- Linear is nothing but whatever input you have,

  your output is the same as the input.
- Among all Activation Functions we have seen so far, this is the only linear one, rest all are non-linear Activation functions.
- The other thing is that the o/p of a linear function can be a large +ve value or a large –ve value, whereas for other Activations functions, the o/p was restricted.
- So basically the Linear function is useful when you want your output to have any value which happens a lot when you have regression problem.

Linear AF: o/p b/w – infinity to + infinity

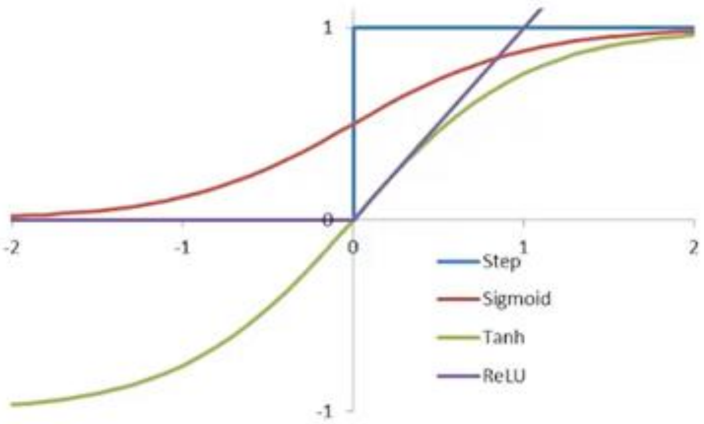
Output activation functions can only be of the following kinds

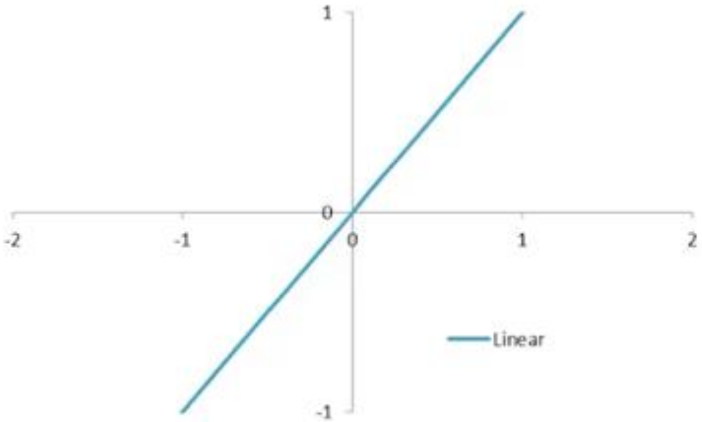




1. Sigmoid – Binary Cl. Pb.
2. Tanh – Used when the desired o/p is in range {-1, +1}
3. Softmax – Multi-class
4. Linear – Regression Pb.
5. ReLU – Used for internal nodes (Non-o/p Nodes)

Types of Activation Functions





- **Step**: $g(x) = \frac{\text{sign}(x)+1}{2}$
- **Sigmoid**: $g(x) = \frac{1}{1+e^{-x}}$
- **Tanh**: $g(x) = \tanh(x)$
- **ReLU**: $g(x) = \max(0, x)$
- **Softmax**: $g(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}}$
- **Linear**: $g(x) = x$

Refer 1 “Linear_Classification.ipynb”