

1.Neural Network:

Neuron: The fundamental unit in a neural network that processes inputs using weights and bias to produce an output.

2.(perceptron)

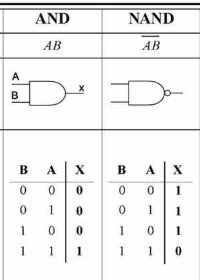
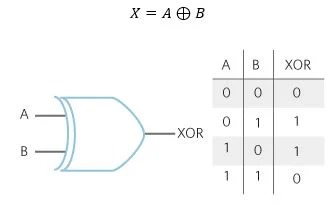
A perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line. Input is typically a feature vector ***x*** multiplied by weights ***w*** and added to a bias ***b*** :

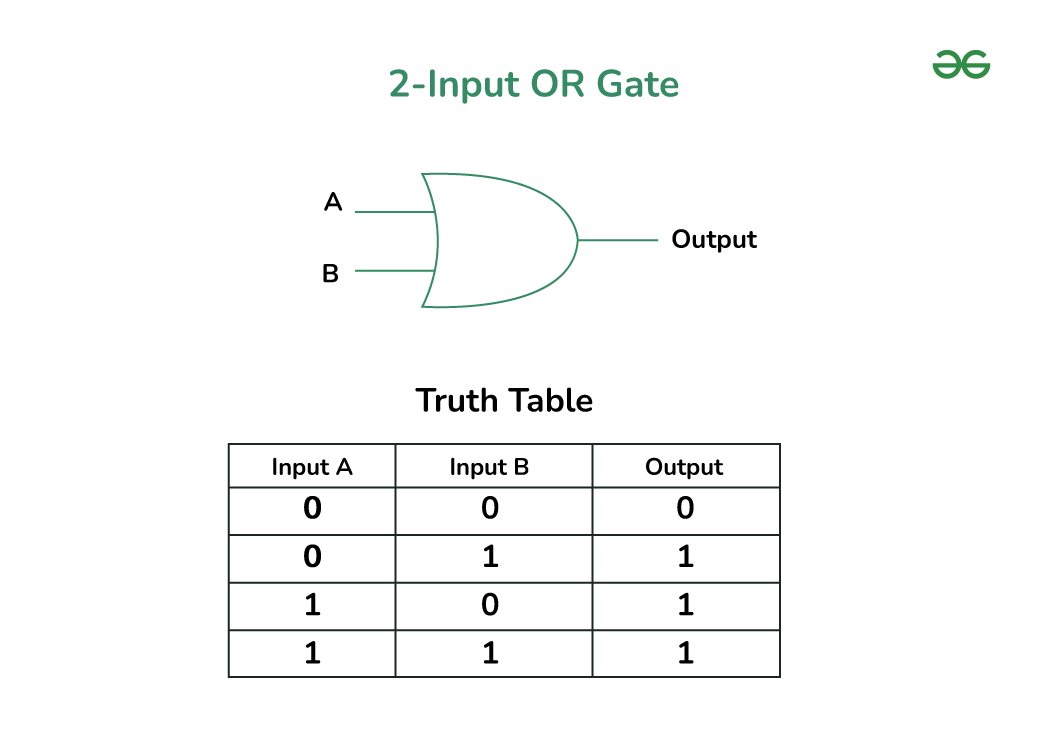
straight line

A single-layer perceptron does not include hidden layers, which allow neural networks to model a feature hierarchy. It is, therefore, a shallow neural network, which prevents it from performing non-linear classification

<https://medium.com/codex/single-layer-perceptron-and-activation-function-b6b74b4aae66>

2. Logic gates:

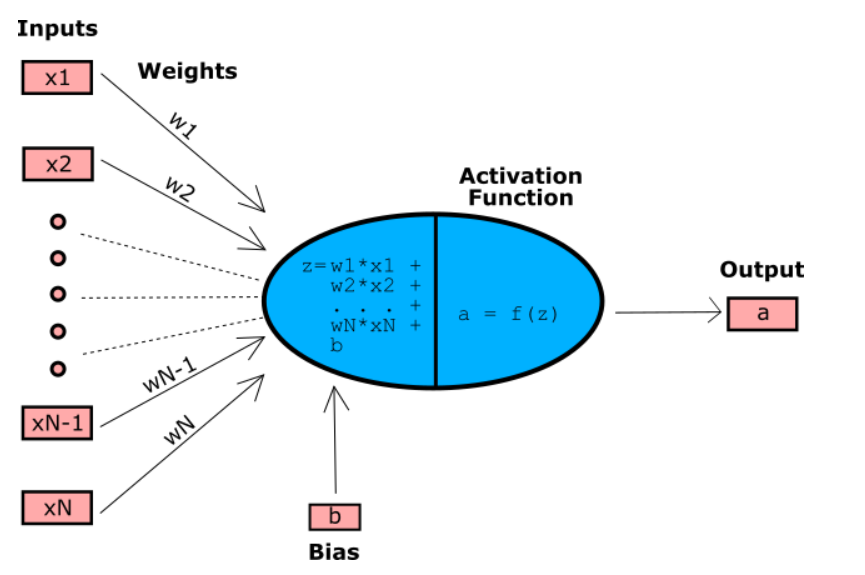
1.  2. 

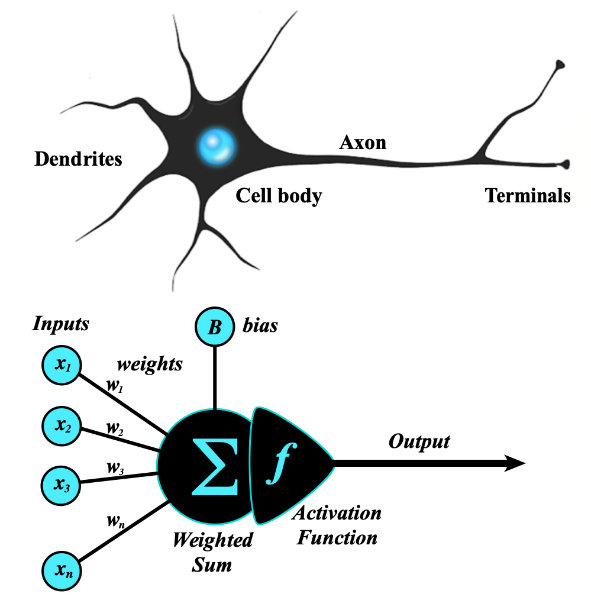
3. 

.

**Brief Documentation (1 paragraph):**

* Describe neuron architecture (inputs, weights, bias).
* Learning rule used (Perceptron update rule).
* Summary of results from logic gates (success on AND/OR/NAND, failure on XOR).

A demo showing the training process and final accuracy for the AND , OR , and NAND gates. A. 



Weight:A parameter that scales the input to a neuron, determining its influence on the neuron's output.

Bias: A parameter added to the weighted sum of inputs in a neuron to shift the output, allowing more flexible modeling

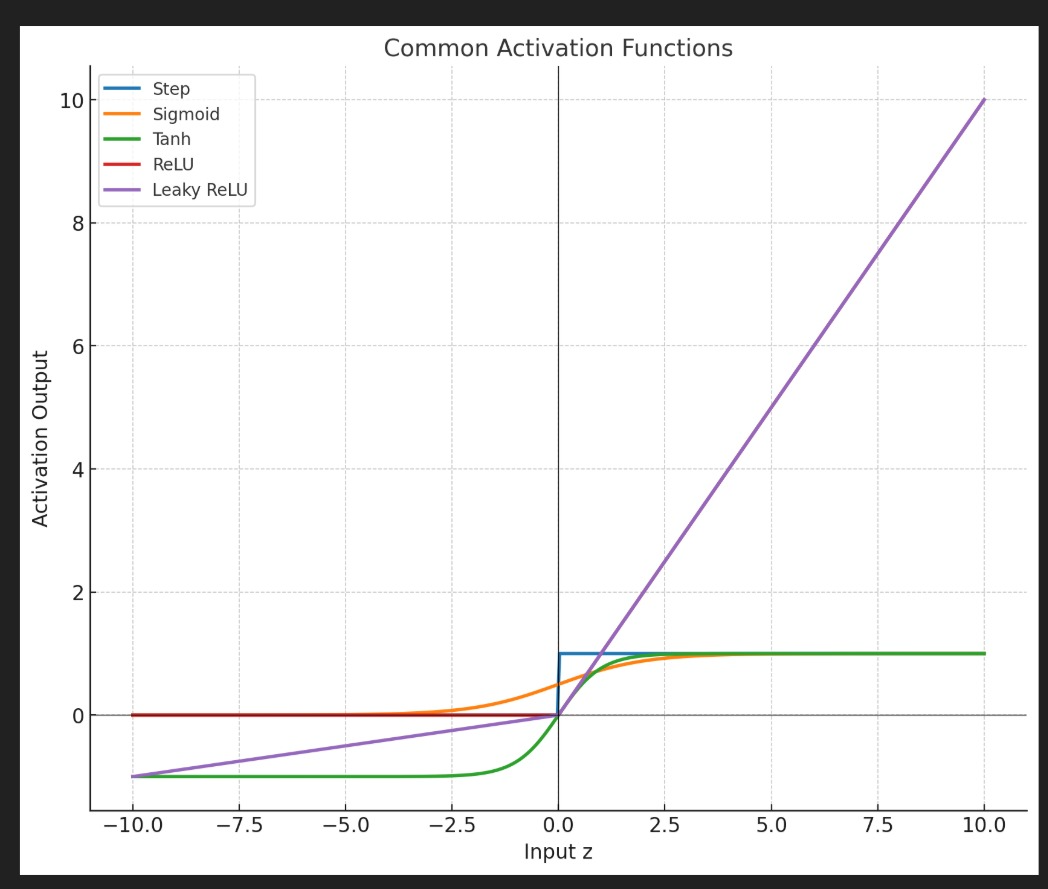
Activation Function:

The activation function is a mathematical function that is used within [neural networks](https://databasecamp.de/en/ml/artificial-neural-networks) and decides whether a neuron is activated or not. It processes the weighted sum of the neuron’s inputs and calculates a new value to determine how strongly the signal is passed on to the next layer in the network.

Z=weight\*input + bias

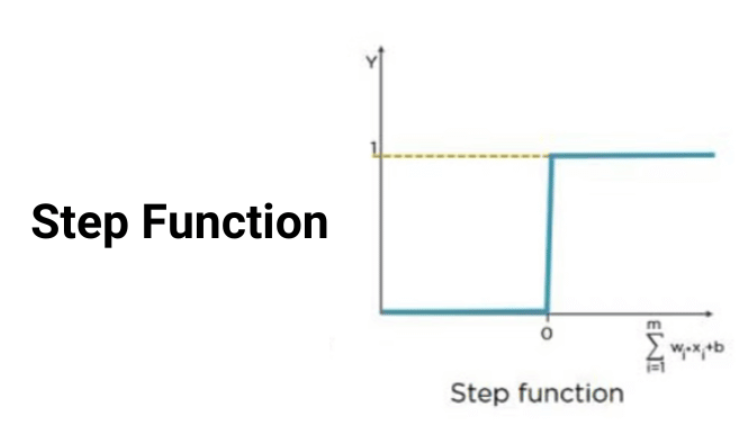
z>=0:1

z<0: 0



Types of Methods in activation functions

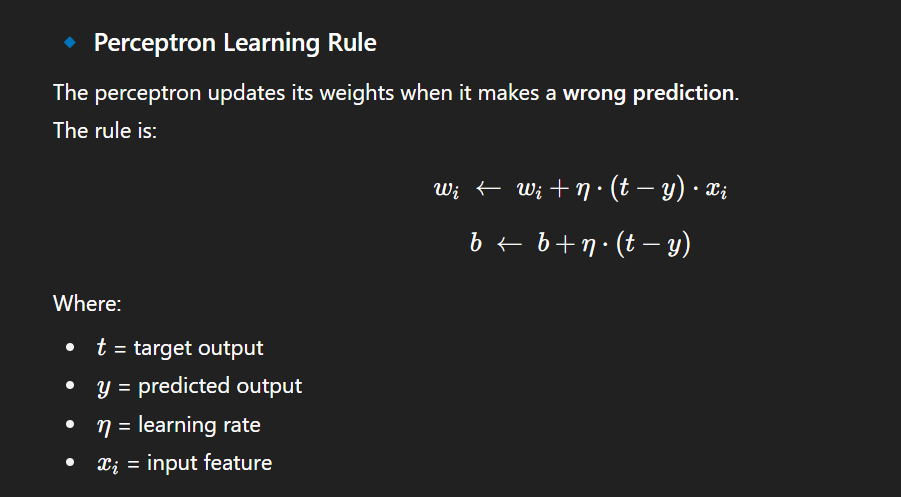
Step Function:



Learning Rule: Perceptron

Learning Rate:

**Perceptron Learning Rule**

****

**Meaning:**

* If prediction is correct (t=y), no update.
* If prediction is wrong, adjust weights in the direction of the error.

This guarantees convergence **if the data is linearly separable** (e.g., AND, OR, NAND).

**⚖️ Learning Rate (η)**

The **learning rate** controls the **step size** of weight updates.

* **Small η (e.g., 0.01, 0.1):**
  + Tiny weight updates → slow but steady learning.
  + More epochs needed.
  + Good stability.
* **Large η(e.g., 0.5, 1.0):**
  + Big updates → faster initial learning.
  + Risk of overshooting the correct weights.
  + Can cause oscillations (no convergence).

This guarantees convergence **if the data is linearly separable** (e.g., AND, OR, NAND).

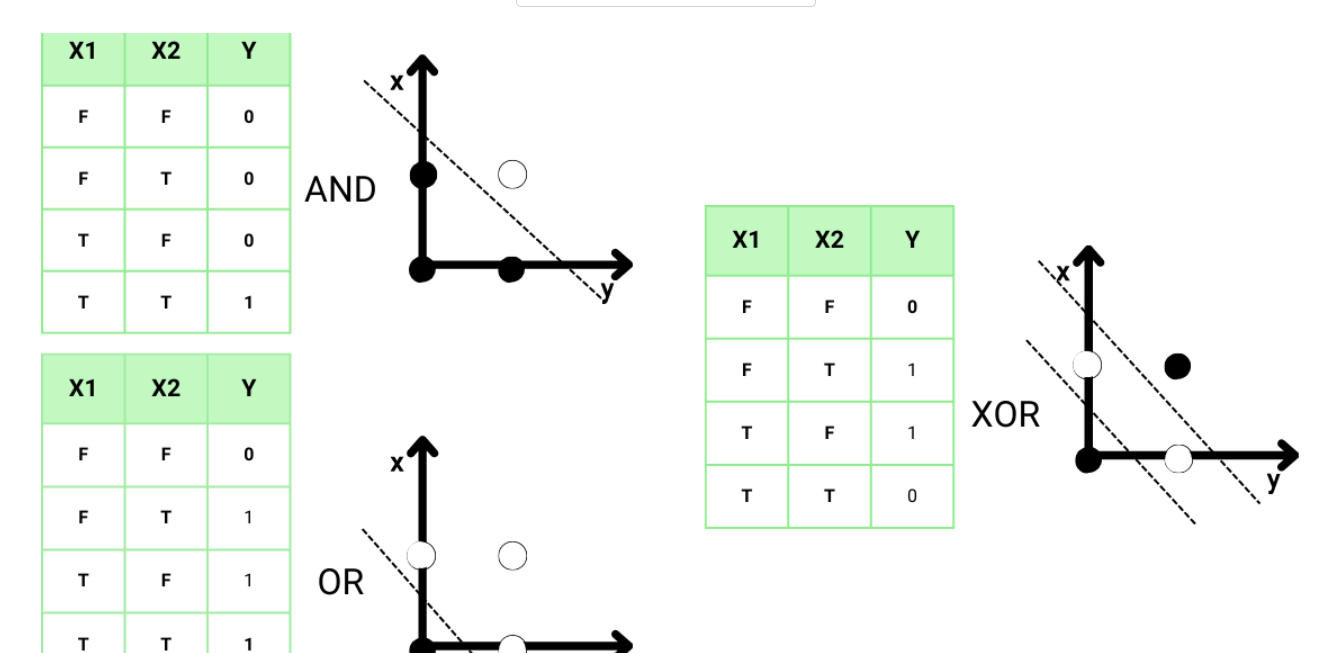
The statement refers to the **Perceptron Convergence Theorem**, which guarantees that if a dataset is linearly separable, the Perceptron learning algorithm will find a separating hyperplane (a solution) in a finite number of steps. The provided examples, AND, OR, and NAND, are all logically equivalent to tasks that can be solved with a single Perceptron, making them linearly separable.

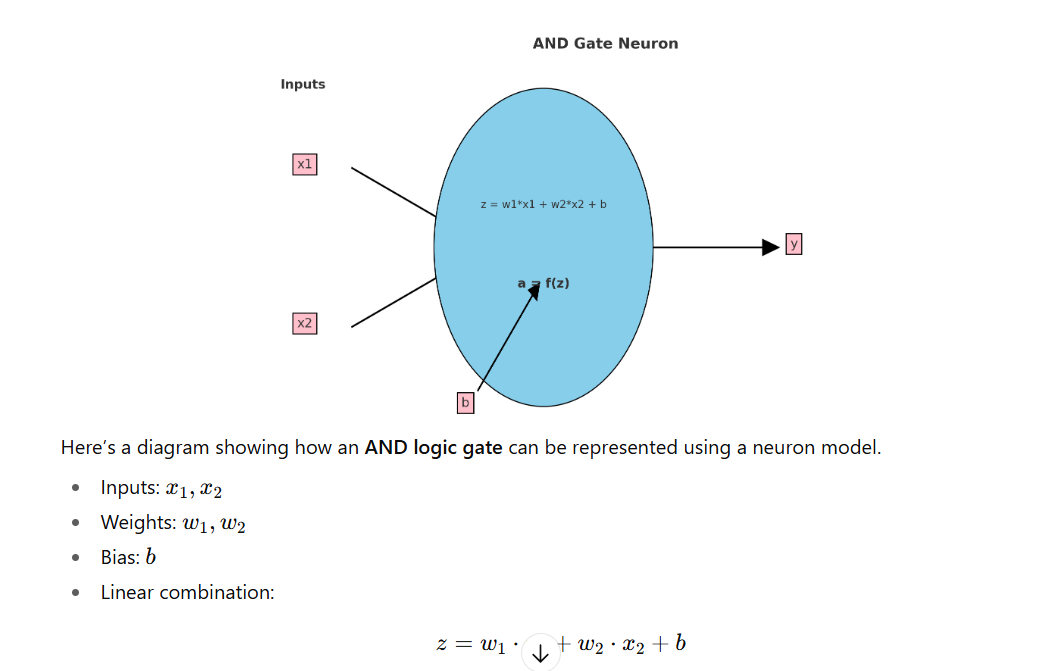
For Xor gate : Limitation and we can resolve it

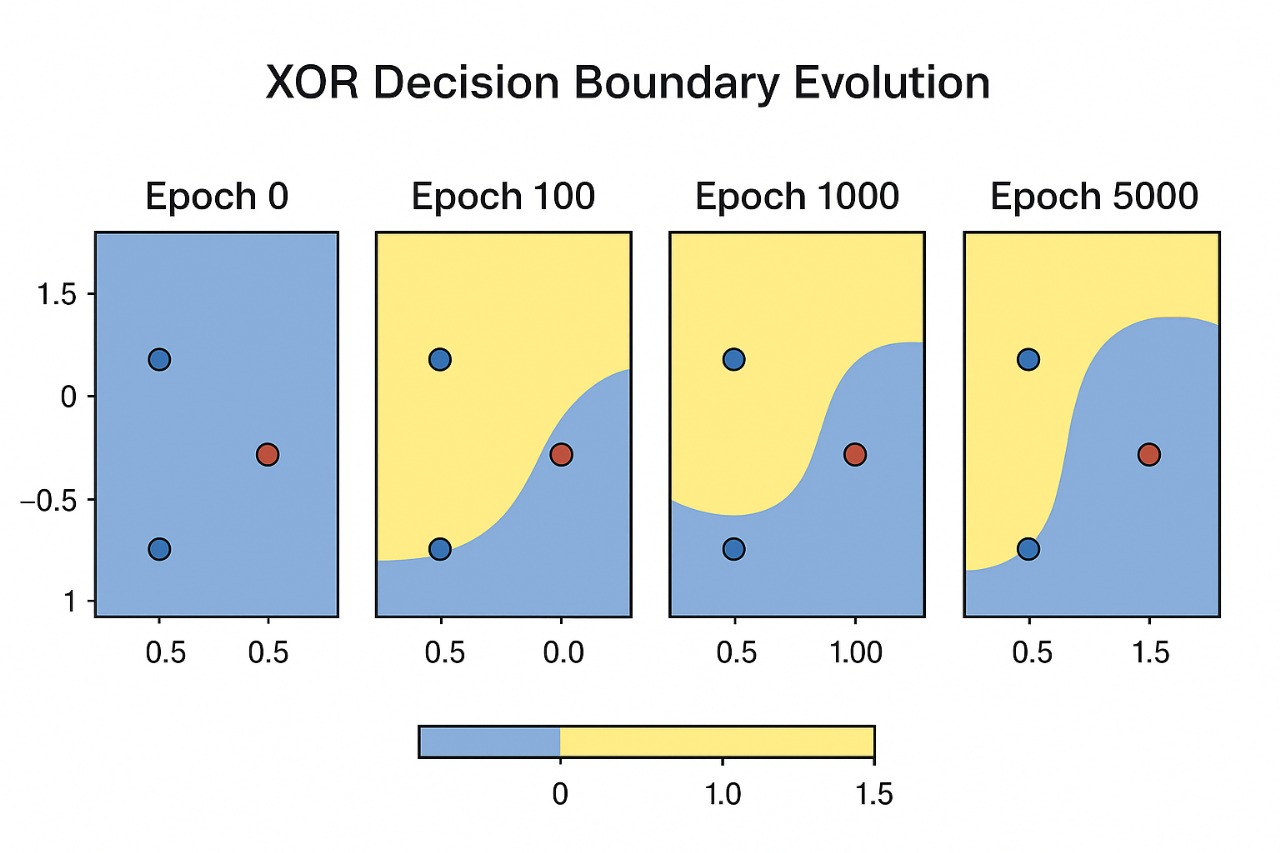
1. **. Introduce a hidden layer:**

The key is to use a multi-layer neural network (MLP).

We can resolve using adding one hidden layer with non-linear activation function







Challenges we face : learning different function to train the model for the desired output