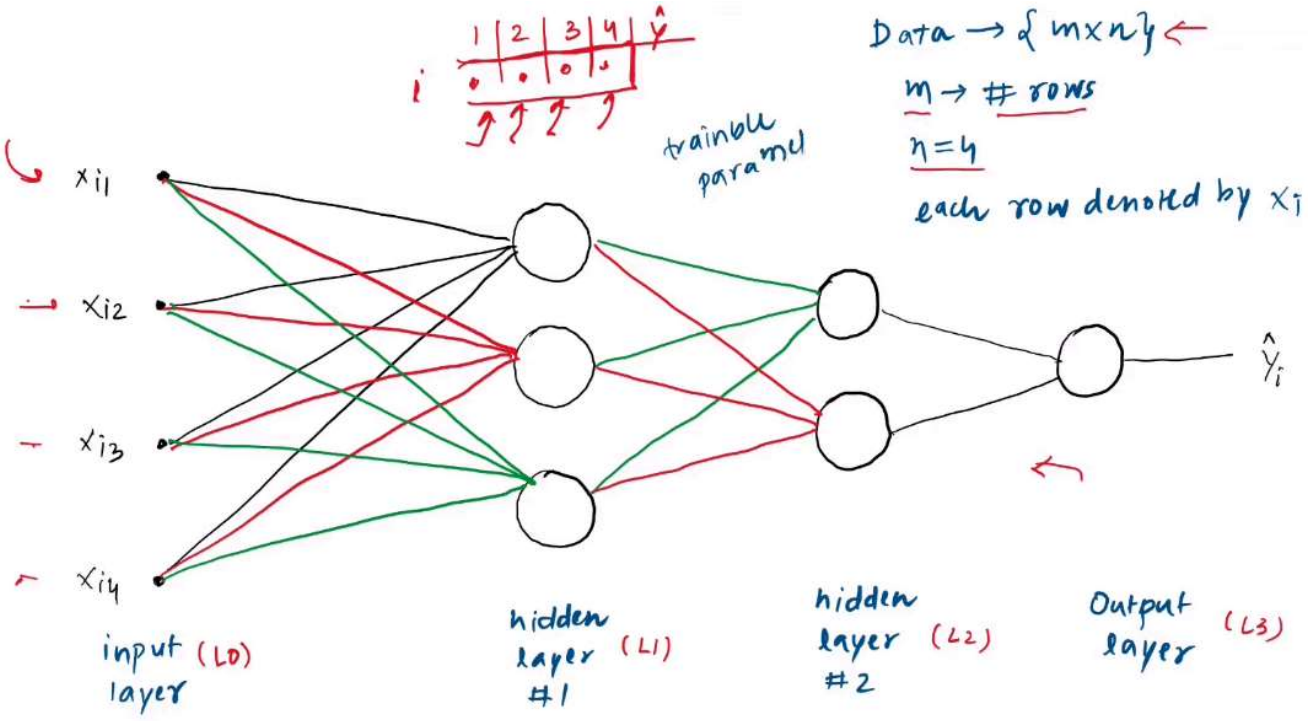
Multi Layer Perceptron

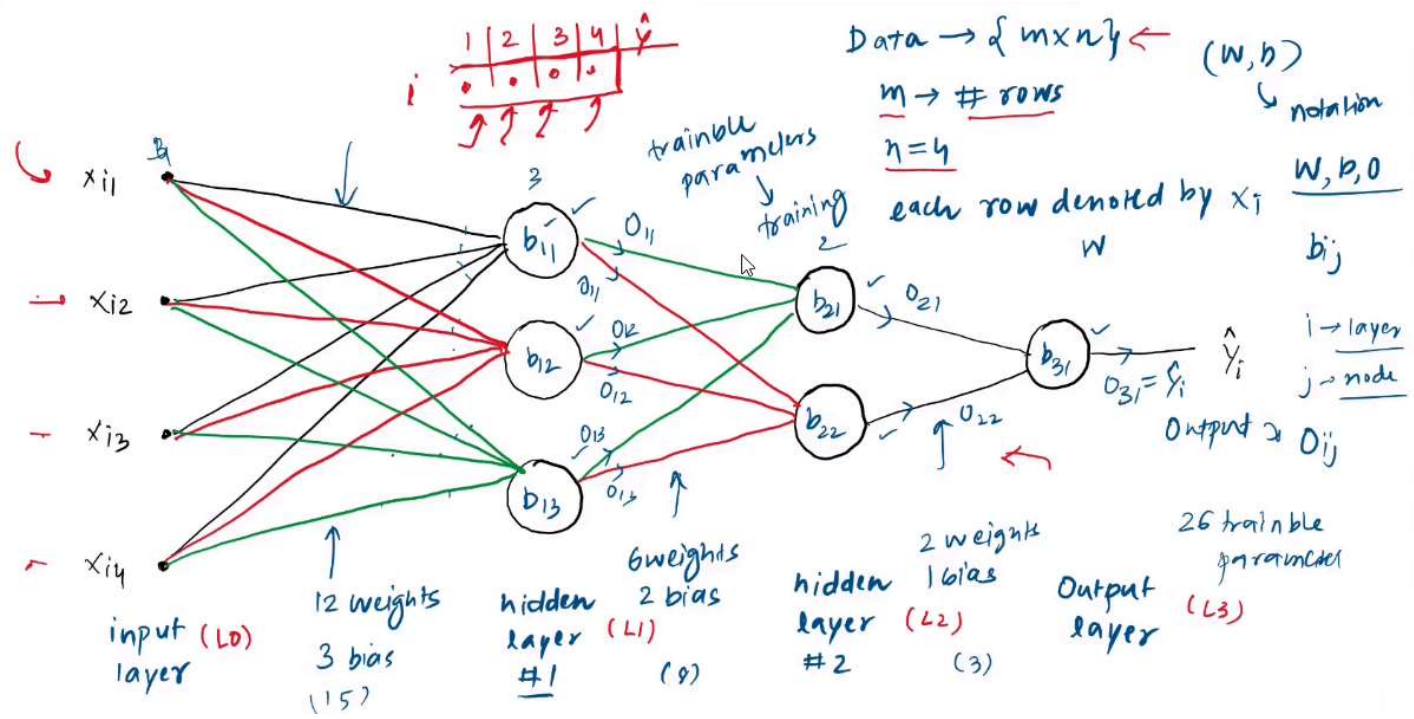
A Perceptron that contains multiple hidden layers is known as Multi Layer Perceptron

- Here **i** is number of Rows.
- **1,2,3,4** are the features, i.e. the Columns.



Trainable Parameters

In **Neural** networks, we have to identify the **Trainable Parameters**.



Total Trainable Parameters = **Total Number of Weights** + **Total Number of Biases**

- They participate in Building and Compilation of our Deep Learning Model.
- 26 in our case.

Bias

Role of Bias: A value that is added to Neurons to reduce the magnitude of error in the Network.

Notation:

- b_{ij}
  Where
  - i > Which Layer
  - j > Which Node

Output

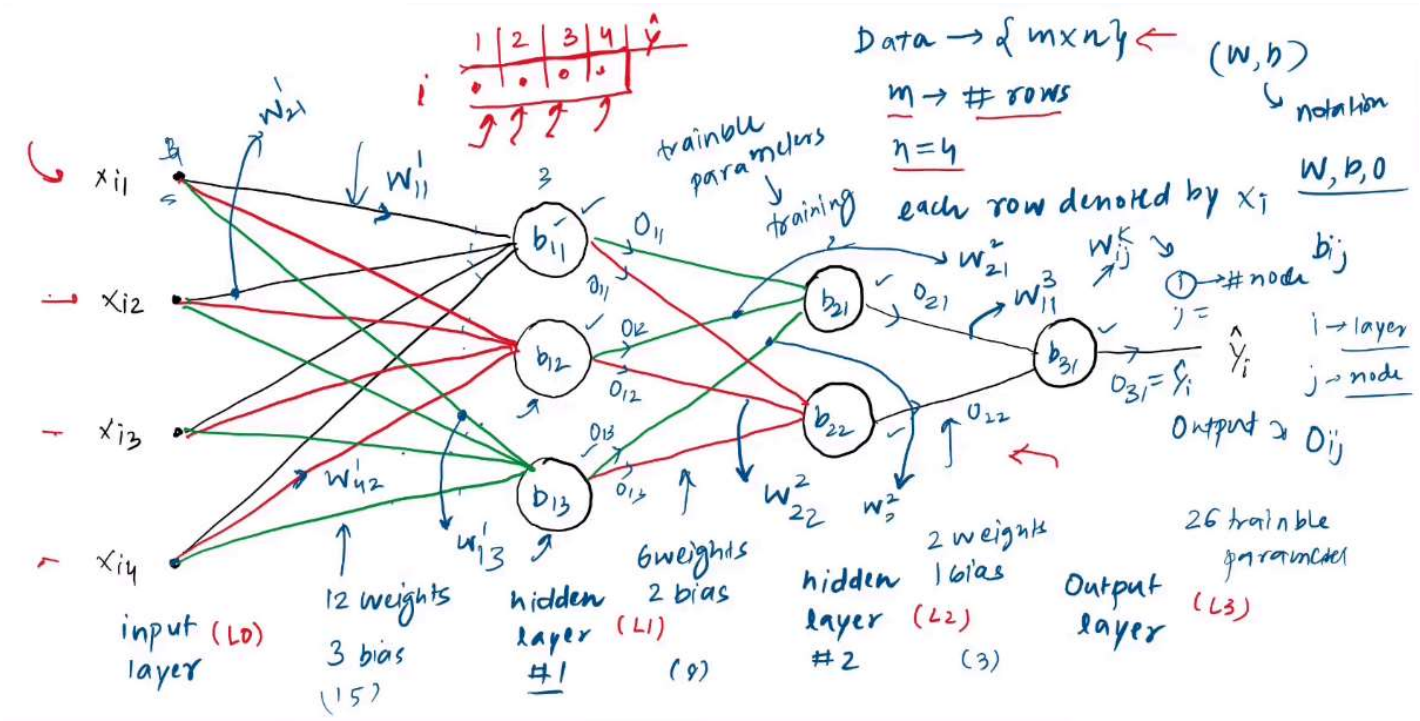
Notation:

- O_{ij}
  Where
  - i > Which Layer
  - j > Which Node

Weights

Notation:

- W_{ij}^{k}
  Where
  - k > layer Number
  - i > Coming from Which Node
  - j > Going to Which Node



Loss

It helps the model to understand how far away it is from the right answer. mean_squared_error is generally used for Regression Problems.

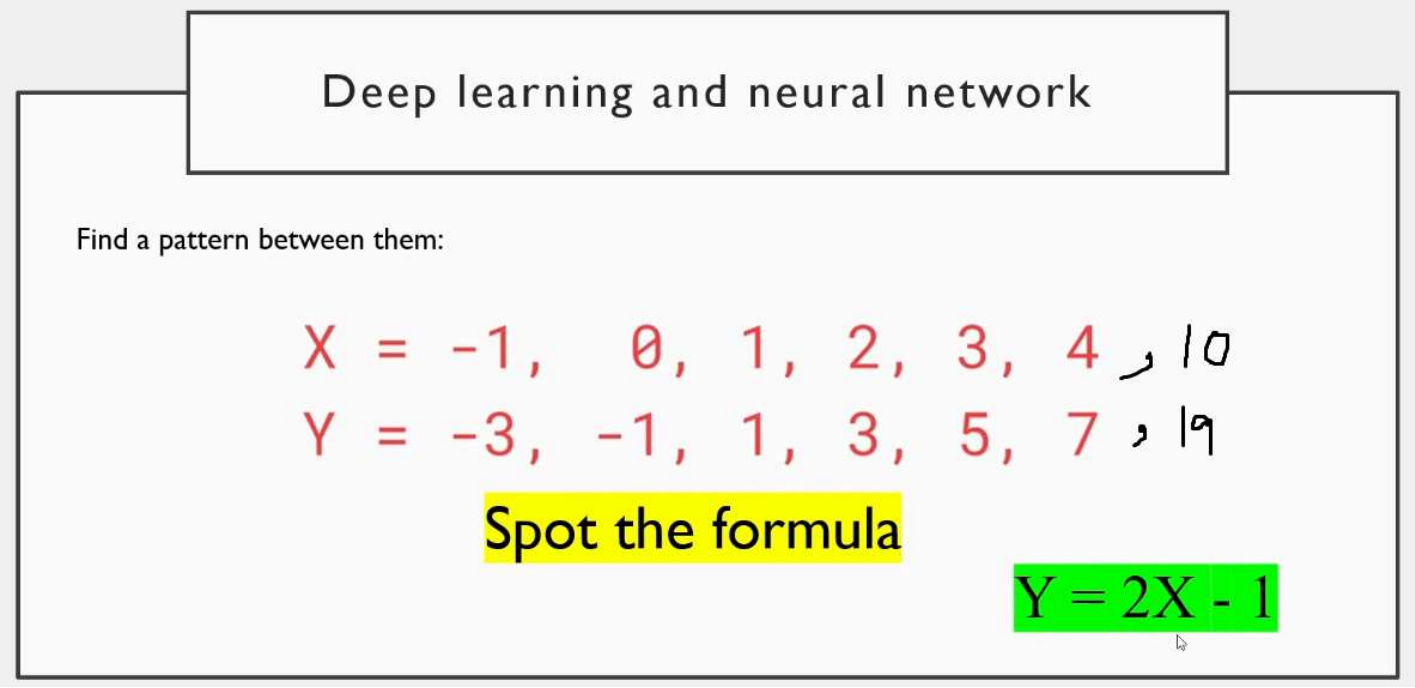
Optimizer

There are two types of solution

- Feasible Solution
  - All solutions are feasible, if they are getting the job done.
- Optimal Solution
  - Solution that gets the job done while either producing the **Best** results or taking **Bare Minimum** resources.

Stochastic Gradient Descent (SGD):

- **Definition:** A popular optimization algorithm used in machine learning and deep learning to minimize the loss function.
- **How it Works:** It updates the model's parameters iteratively using a randomly selected subset of data (mini-batch) rather than the entire dataset, which makes it faster and suitable for large datasets.
- **Applications:** Used in training neural networks and other models in machine learning.
-



Types of Categorical Data

Ordinal

Ordinal data is a type of categorical data with a set order or ranking among the categories. While the data is categorized, the categories have a meaningful order, but the differences between them are not quantitatively meaningful.

Person	Education Level	Satisfaction Rating
1	High School	Satisfied
2	Bachelor's	Neutral
3	Master's	Unsatisfied

Nominal

Nominal data is a type of categorical data where the categories are labels or names without any inherent order or ranking. The primary purpose of nominal data is to categorize data without implying any quantitative value.

Example Table:

Person	Gender	Favorite Fruit
1	Male	Apple
2	Female	Banana
3	Non-binary	Cherry

StandardScaler

StandardScaler is part of the `sklearn.preprocessing` module and is used to transform features by removing the mean and scaling to unit variance.

Formula:

The standard score of a sample xxx is calculated as:

$z = \frac{x - \mu}{\sigma}$

Where:

- z is the standardized value.
- x is the original value.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.

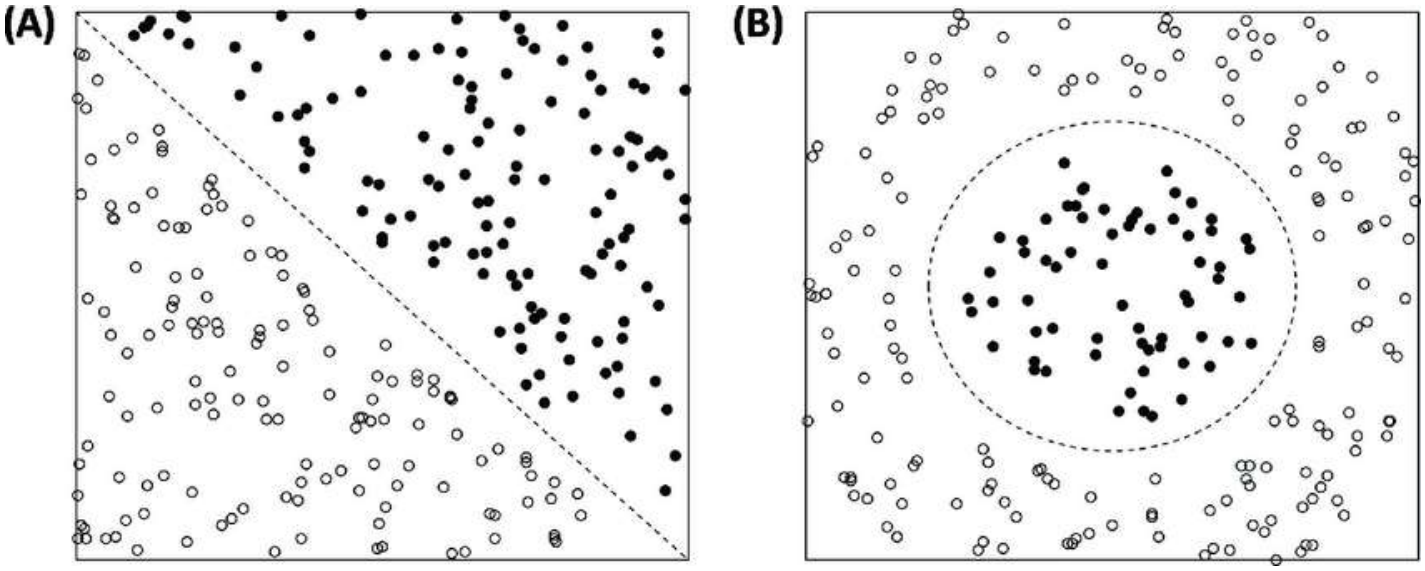
Why Use StandardScaler?

- **Feature Scaling:** Ensures that each feature contributes equally to the distance calculations, preventing features with larger magnitudes from dominating.
- **Normalization:** Helps in achieving faster convergence for gradient descent optimization algorithms.
- **Model Performance:** Improves the performance of models that assume normally distributed data or are sensitive to the scale of the features.

When to Use:

- **Linear Models:** Linear Regression, Logistic Regression.
- **Distance-based Models:** KNN, K-Means Clustering.
- **Neural Networks:** Generally benefit from normalized input features.
- **SVM:** Support Vector Machines can be sensitive to unscaled data.

Linear and Non-Linear Separability



A: A straight line can separate two classes of Data making the data Linearly Separable.

B: No straight line can separate two classes of Data making the data Non-Linearly Separable.

Deep Learning Algorithms (Multi Layer Perceptron) are very good at processing Non-Linearly Separable Data

Learning Rate (α)

It indicates how to adjust the weights of your model during Back Propagation.

We start with a very small learning rate i.e. 0.01 or 0.001 etc.

Shallow vs Deep Neural Networks

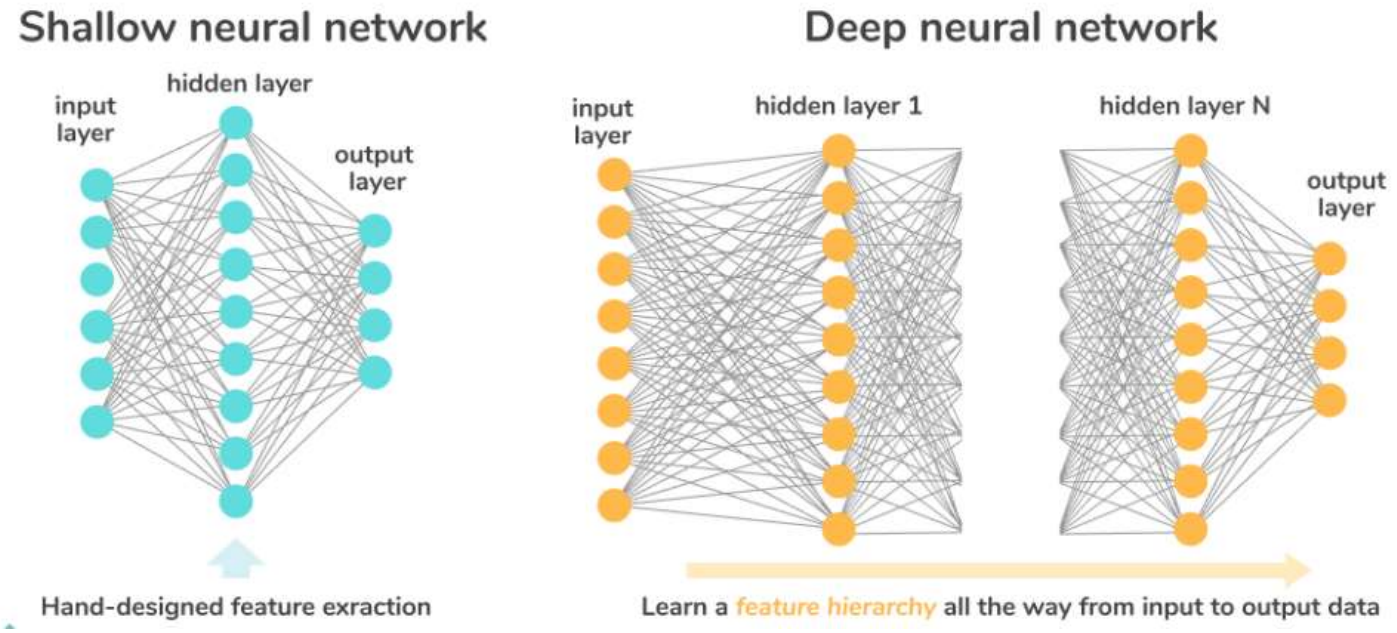
Shallow Neural Networks:

- **Definition**: A shallow neural network typically has one or two hidden layers between the input and output layers.
- **Complexity**: The network's architecture is simple, which makes it easier to understand, train, and debug.
- **Learning Capacity**: Due to the limited number of layers, shallow networks have a reduced capacity to learn complex patterns and representations from data.

- **Training Time**: Generally, shallow networks require less computational power and time to train, especially on smaller datasets.
- **Use Cases**: Shallow networks are often sufficient for simpler tasks, such as linear regression, basic classification tasks, or when the data is relatively small and the relationships within the data are not highly complex.

Deep Neural Networks:

- **Definition**: A deep neural network has multiple hidden layers, often with a much higher number of layers than a shallow network. Typically, if a network has more than two hidden layers, it is considered "deep."
- **Complexity**: The architecture is more complex, with many layers and neurons, enabling the network to capture and learn hierarchical patterns in the data.
- **Learning Capacity**: Deep networks have a higher capacity to model complex, non-linear relationships and can learn intricate features from large and complex datasets.
- **Training Time**: Deep networks require more computational resources, longer training times, and often large datasets to reach optimal performance. They are also prone to issues like overfitting and vanishing/exploding gradients, although modern techniques like batch normalization, dropout, and advanced optimizers help mitigate these problems.
- **Use Cases**: Deep neural networks are used in more complex tasks such as image and speech recognition, natural language processing, autonomous driving, and other applications where data has complex structures and relationships.



Activation Functions

Linear

- A.k.a the Identity Activation function
- Use in regression models where the predicted value needs to be a continuous range, such as predicting prices, temperatures, etc.

Sigmoid

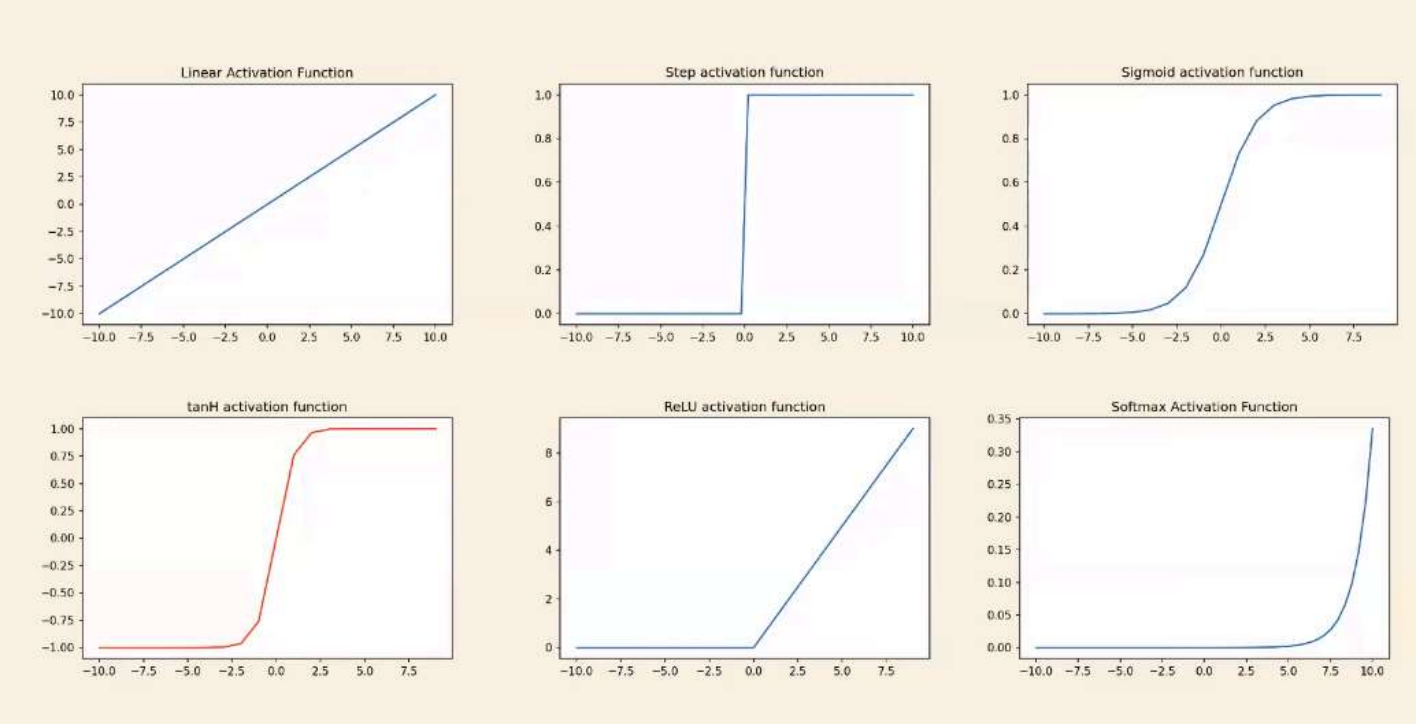
- Used in Binary Classification.
- Better performance in Binary Classification as compared to Step Activation Function
- Less common now a days due to *Vanishing Gradient* issue.

ReLU

- Used in Modern Deep Learning tasks, i.e. Complex Neural Networks in various fields.

Softmax

- Used in Multi Classification



Activation Functions in Hidden Layers

- Previously, **tanH** was used as activation function for Hidden Layers.
- Currently, **sigmoid** and **ReLU** are preferred.

Common Rules in MLP

- **Neurons** should never be less than your Given Training Shape i.e. Your Number of Features.
- **Batch Size** should not be very large or small
  - If Large > The learning of the model will reduce
  - If Small > Number of Computations will increase resulting in more time.
  - Common Values are **32** or **64**.
- **Epochs** are the loops of Training and Testing
  - More Epochs results in improved results but significantly increase execution time.

Types of Error

Mean Squared Error

*We take **square** of Error Value to remove the -ve sign.*