Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

UNIT- 3
Supervised Learning-II

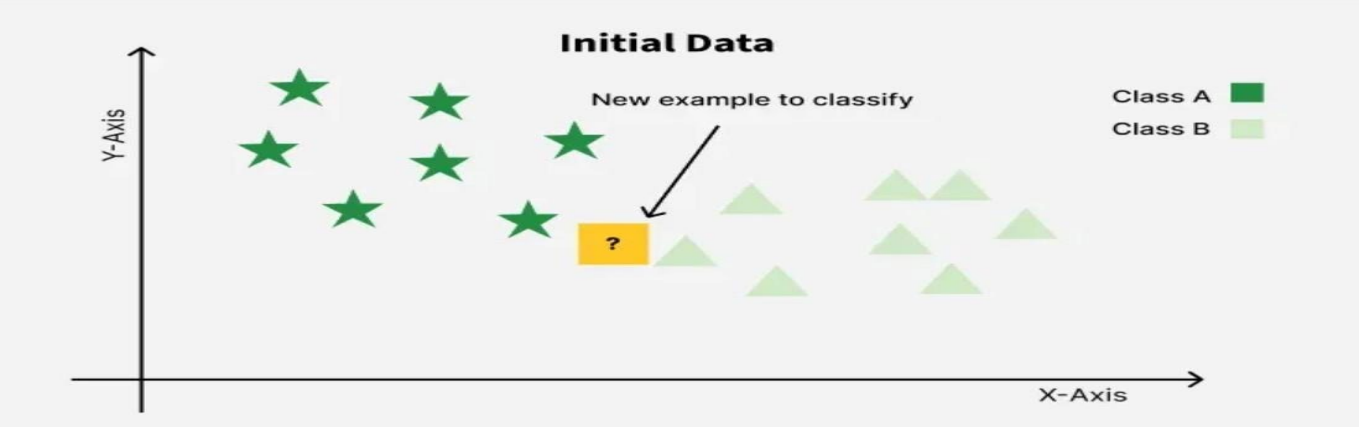
Study Guide

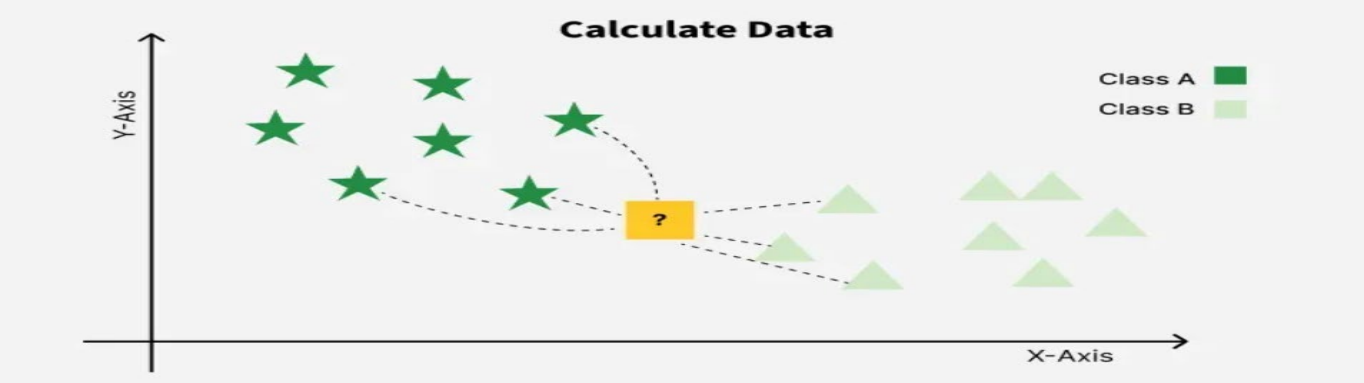
Dr. Vinod Patidar
Associate Professor
CSE Department, PIT
Parul University

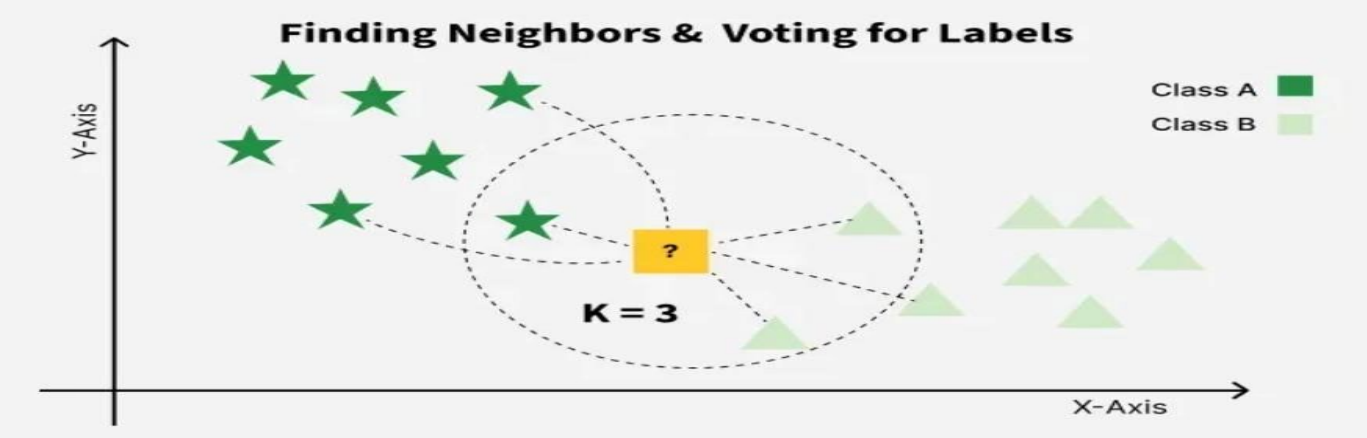
1. K-NN classifier
2. Logistic regression
3. Perceptron
4. Single layer & multi-layer
5. Support Vector Machines
6. Linear & Non-linear
7. Semi Supervised Learning

K-Nearest Neighbour(KNN) Algorithm

K-Nearest Neighbours (KNN) is a supervised machine learning algorithm generally used for classification but can also be used for regression tasks. It works by finding the "k" closest data points (neighbours) to a given input and makes a prediction based on the majority class (for classification) or the average value (for regression). Since KNN makes no assumptions about the underlying data distribution it makes it a non-parametric and instance-based learning method.

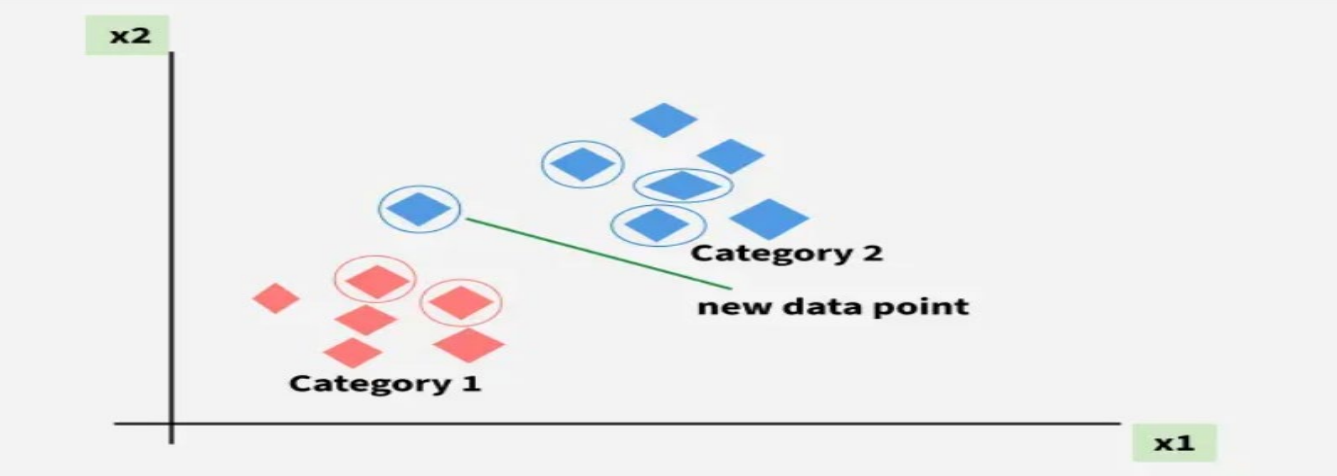






K-Nearest Neighbours is also called as a lazy learner algorithm because it does not learn from the training set immediately instead it stores the entire dataset and performs computations only at the time of classification.

For example, consider the following table of data points containing two features:



The new point is classified as Category 2 because most of its closest neighbors are blue squares. KNN assigns the category based on the majority of nearby points. The image shows how KNN predicts the category of a new data point based on its closest neighbours.

- The red diamonds represent Category 1 and the blue squares represent Category 2.
- The new data point checks its closest neighbours (circled points).
- Since the majority of its closest neighbours are blue squares (Category 2) KNN predicts the new data point belongs to Category 2.

KNN works by using proximity and majority voting to make predictions.

What is 'K' in K Nearest Neighbour?

In the k-Nearest Neighbours algorithm k is just a number that tells the algorithm how many nearby points or neighbours to look at when it makes a decision.

Example: Imagine you are deciding which fruit it is based on its shape and size. You compare it to fruits you already know.

- If k = 3, the algorithm looks at the 3 closest fruits to the new one.
- If 2 of those 3 fruits are apples and 1 is a banana, the algorithm says the new fruit is an apple because most of its neighbours are apples.

Distance Metrics Used in KNN Algorithm

KNN uses distance metrics to identify nearest neighbour, these neighbours are used for classification and regression task. To identify nearest neighbour we use below distance metrics:

1. Euclidean Distance

Euclidean distance is defined as the straight-line distance between two points in a plane or space. You can think of it like the shortest path you would walk if you were to go directly from one point to another.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

2. Manhattan Distance

This is the total distance you would travel if you could only move along horizontal and vertical lines like a grid or city streets. It is also called "taxicab distance" because a taxi can only drive along the grid-like streets of a city.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

3. Minkowski Distance

Minkowski distance is like a family of distances, which includes both Euclidean and Manhattan distances as special cases.

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}$$

From the formula above, when p=2, it becomes the same as the Euclidean distance formula and when p=1, it turns into the Manhattan distance formula. Minkowski distance is essentially a flexible formula that can represent either Euclidean or Manhattan distance depending on the value of p.

Applications of KNN

- **Recommendation Systems**: Suggests items like movies or products by finding users with similar preferences.
- **Spam Detection**: Identifies spam emails by comparing new emails to known spam and non-spam examples.
- **Customer Segmentation**: Groups customers by comparing their shopping behaviour to others.
- **Speech Recognition**: Matches spoken words to known patterns to convert them into text.

Advantages of KNN

- **Simple to use**: Easy to understand and implement.
- **No training step**: No need to train as it just stores the data and uses it during prediction.
- **Few parameters**: Only needs to set the number of neighbors (k) and a distance method.
- **Versatile**: Works for both classification and regression problems.

Disadvantages of KNN

- **Slow with large data**: Needs to compare every point during prediction.
- **Struggles with many features**: Accuracy drops when data has too many features.
- **Can Overfit**: It can overfit especially when the data is high-dimensional or not clean.

Let's dive deeper into an example of KNN to make the concept clearer. Below is a data that includes **age**, **gender** and the **class of sports** people play.

NAME	AGE	GENDER	CLASS OF SPORTS
Ajay	32	0	Football
Mark	40	0	Neither

NAME	AGE	GENDER	CLASS OF SPORTS
Sara	16	1	Cricket
Zaira	34	1	Cricket
Sachin	55	0	Neither
Rahul	40	0	Cricket
Pooja	20	1	Neither
Smith	15	0	Cricket
Laxmi	55	1	Football
Michael	15	0	Football

Here male is denoted with numeric value 0 and female with 1. Let's find in which class of people Angelina will lie whose k factor is 3 and age is 5. So we have to find out the distance using **Euclidean distance formula** to find the distance between any two points.

To calculate the distance between **Angelina** and other individuals in the dataset:

$$d = \sqrt{(age_2 - age_1)^2 + (gender_2 - gender_1)^2}$$

Here, **Angelina** has:

- **Age = 5**
- **Gender = 1** (female)

istance between Angelina and	Distance
Ajay	27.02
Mark	35.01
Sara	11.00
Zaira	29.00
Sachin	50.01
Rahul	35.01
Pooja	15.00
Smith	10.05
Laxmi	50.00
Michael	10.05

K-Nearest Neighbors (K = 3): 3 nearest neighbors to Angelina are:

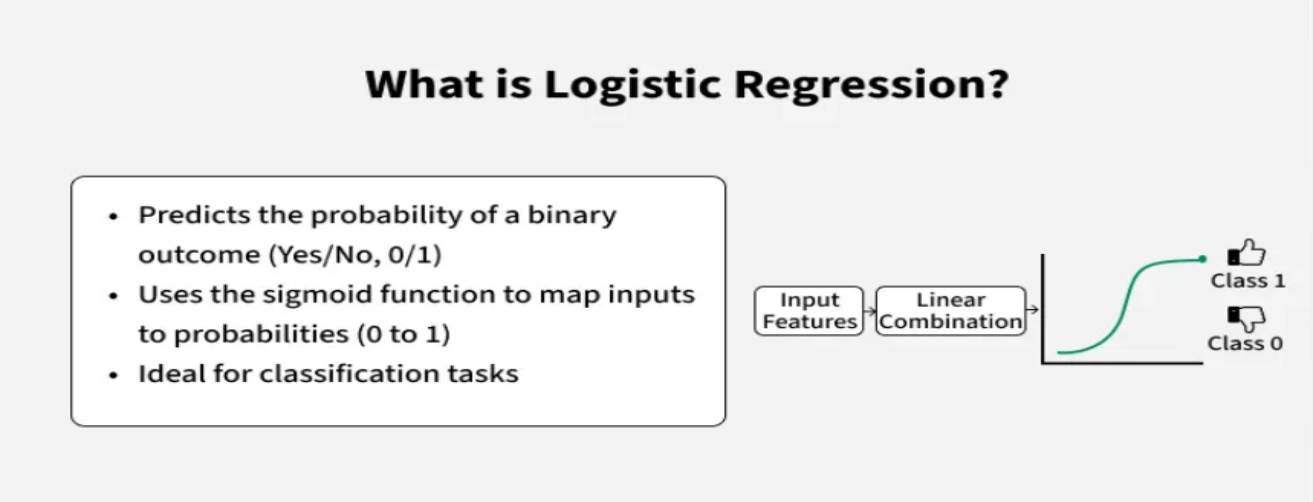
1. **Smith** (Cricket)- 10.5
2. **Michael** (Football)- 10.05
3. **Sara** (Cricket)- 11

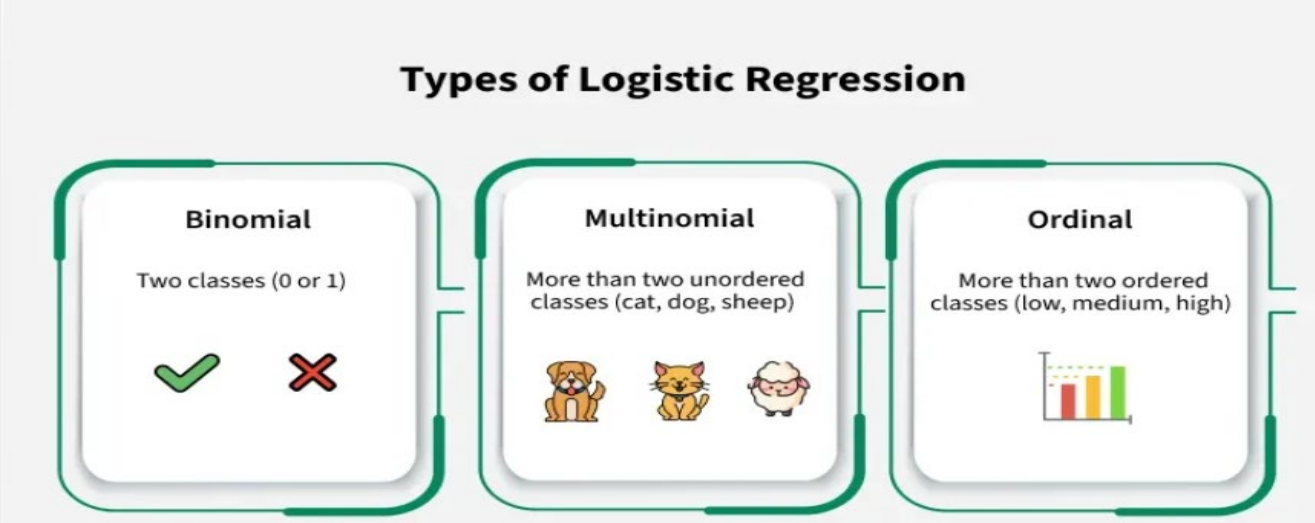
So according to KNN algorithm classifying based on Majority Vote, Angelina will be in the class of people who like cricket.
This example illustrates the working of KNN and how it classifies data based on the majority class of its nearest neighbours. By calculating the distance between data points, KNN helps in making predictions about new data. The importance of selecting the right value of **K** and handling categorical data appropriately (such as converting

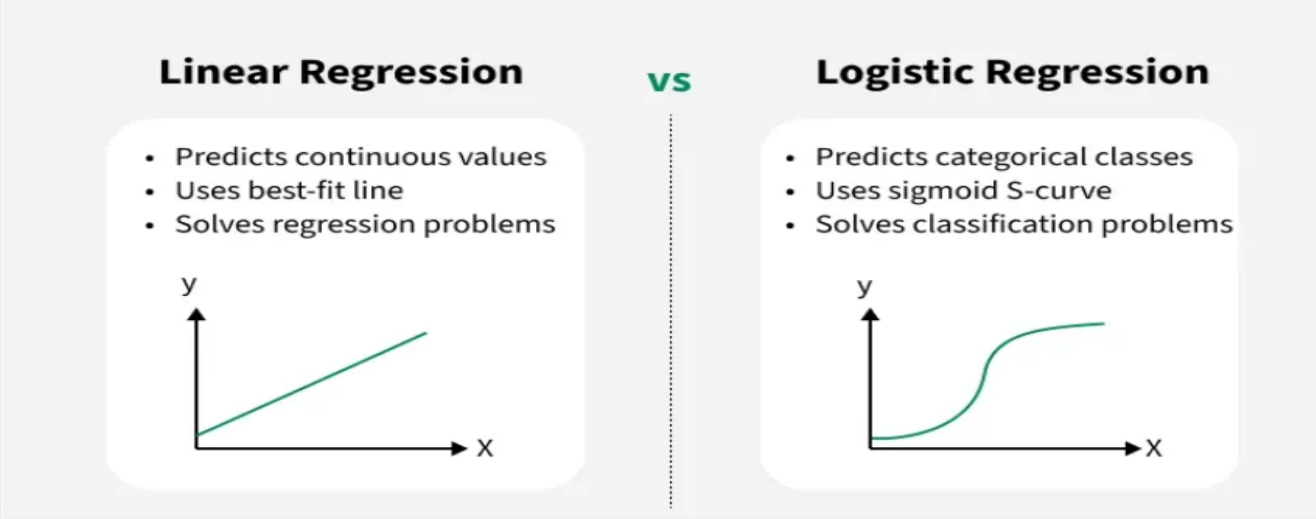
gender to numeric values) cannot be overstated in ensuring accurate classification results.

Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1. In this article, we will see the basics of logistic regression and its core concepts.







Types of Logistic Regression

Logistic regression can be classified into three main types based on the nature of the dependent variable:

1. **Binomial Logistic Regression**: This type is used when the dependent variable has only two possible categories. Examples include Yes/No, Pass/Fail or 0/1. It is the most common form of logistic regression and is used for binary classification problems.
2. **Multinomial Logistic Regression**: This is used when the dependent variable has three or more possible categories that are not ordered. For example, classifying animals into categories like "cat," "dog" or "sheep." It extends the binary logistic regression to handle multiple classes.

3. **Ordinal Logistic Regression**: This type applies when the dependent variable has three or more categories with a natural order or ranking. Examples include ratings like "low," "medium" and "high." It takes the order of the categories into account when modelling.

Assumptions of Logistic Regression

Understanding the assumptions behind logistic regression is important to ensure the model is applied correctly, main assumptions are:

1. **Independent observations**: Each data point is assumed to be independent of the others means there should be no correlation or dependence between the input samples.
2. **Binary dependent variables**: It takes the assumption that the dependent variable must be binary, means it can take only two values. For more than two categories SoftMax functions are used.
3. **Linearity relationship between independent variables and log odds**: The model assumes a linear relationship between the independent variables and the log odds of the dependent variable which means the predictors affect the log odds in a linear way.
4. **No outliers**: The dataset should not contain extreme outliers as they can distort the estimation of the logistic regression coefficients.
5. **Large sample size**: It requires a sufficiently large sample size to produce reliable and stable results.

Perceptron

Perceptron is a type of neural network that performs binary classification that maps input features to an output decision, usually classifying data into one of two categories, such as 0 or 1.

Perceptron consists of a single layer of input nodes that are fully connected to a layer of output nodes. It is particularly good at learning **linearly separable patterns**. It utilizes a variation of artificial neurons called **Threshold Logic Units (TLU)**, which were first introduced by McCulloch and Walter Pitts in the 1940s. This foundational

model has played a crucial role in the development of more advanced neural networks and machine learning algorithms.

Types of Perceptron

1. **Single-Layer Perceptron** is a type of perceptron is limited to learning linearly separable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. While powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.
2. **Multi-Layer Perceptron** possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

Basic Components of Perceptron

A Perceptron is composed of key components that work together to process information and make predictions.

- **Input Features:** The perceptron takes multiple input features, each representing a characteristic of the input data.
- **Weights:** Each input feature is assigned a weight that determines its influence on the output. These weights are adjusted during training to find the optimal values.
- **Summation Function:** The perceptron calculates the weighted sum of its inputs, combining them with their respective weights.
- **Activation Function:** The weighted sum is passed through the **Heaviside step function**, comparing it to a threshold to produce a binary output (0 or 1).
- **Output:** The final output is determined by the activation function, often used for **binary classification** tasks.
- **Bias:** The bias term helps the perceptron make adjustments independent of the input, improving its flexibility in learning.
- **Learning Algorithm:** The perceptron adjusts its weights and bias using a learning algorithm, such as the Perceptron Learning Rule, to minimize prediction errors.

These components enable the perceptron to learn from data and make predictions. While a single perceptron can handle simple binary classification, complex tasks require multiple perceptron's organized into layers, forming a neural network.

How does Perceptron work?

A weight is assigned to each input node of a perceptron, indicating the importance of that input in determining the output. The Perceptron's output is calculated as a weighted sum of the inputs, which is then passed through an activation function to decide whether the Perceptron will fire.

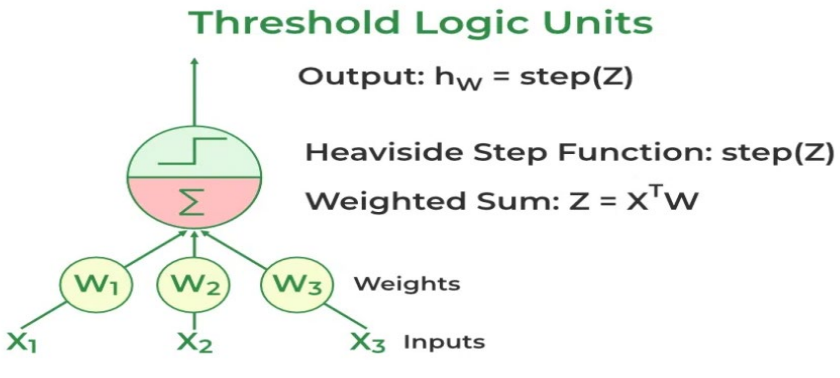
The weighted sum is computed as:

$z=w_1x_1+w_2x_2+...+w_nx_n=X^TW$

The step function compares this weighted sum to a threshold. If the input is larger than the threshold value, the output is 1; otherwise, it's 0. This is the most common activation function used in Perceptron's are represented by the Heaviside step function:

$$h(z) = \begin{cases} 0 & \text{if } z < \text{Threshold} \\ 1 & \text{if } z \geq \text{Threshold} \end{cases}$$

A perceptron consists of a single layer of Threshold Logic Units (TLU), with each TLU fully connected to all input nodes.



Threshold Logic units

Example: Perceptron in Action

Let's take a simple example of classifying whether a given fruit is an apple or not based on two inputs: its weight (in grams) and its color (on a scale of 0 to 1, where 1 means red). The perceptron receives these inputs, multiplies them by their weights, adds a bias, and applies the activation function to decide whether the fruit is an apple or not.

- Input 1 (Weight): 150 grams
- Input 2 (Color): 0.9 (since the fruit is mostly red)
- Weights: [0.5, 1.0]
- Bias: 1.5

The perceptron's weighted sum would be:

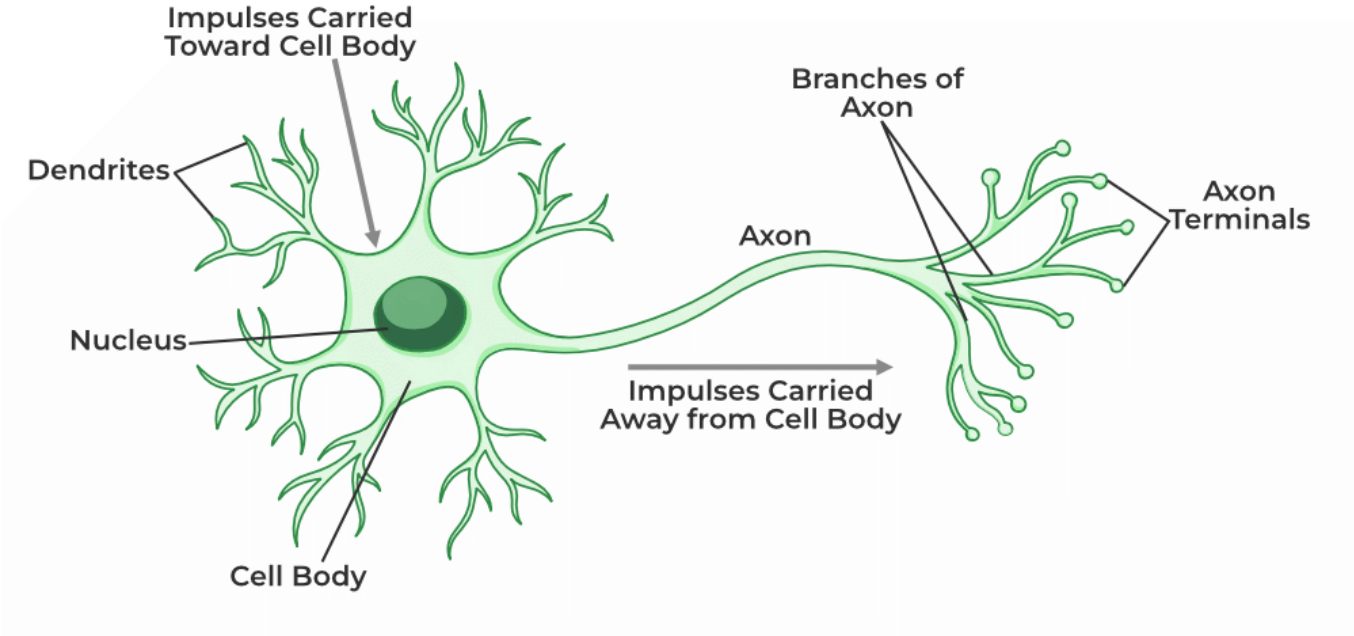
$(150*0.5)+(0.9*1.0)+1.5=76.4(150*0.5)+(0.9*1.0)+1.5=76.4$

Let's assume the activation function uses a threshold of 75. Since 76.4 > 75, the perceptron classifies the fruit as an apple (output = 1).

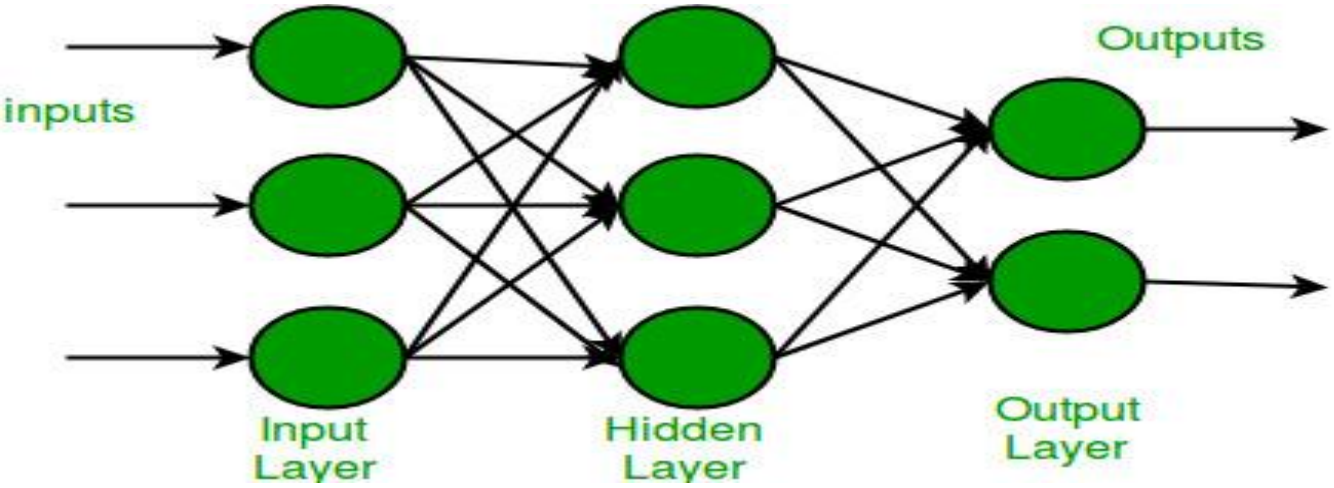
Single Layer Perceptron in TensorFlow

Single Layer Perceptron *is inspired by biological neurons and their ability to process information. To understand the SLP we first need to break down the workings of a single artificial neuron which is the fundamental building block of neural networks. An* ***artificial neuron*** *is a simplified computational model that mimics the behaviour of a biological neuron. It takes inputs, processes them and produces an output. Here's how it works step by step:*

- *Receive signal from outside.*
- *Process the signal and decide whether we need to send information or not.*
- *Communicate the signal to the target cell, which can be another neuron or gland.*



Similarly, neural networks also function in a similar manner.

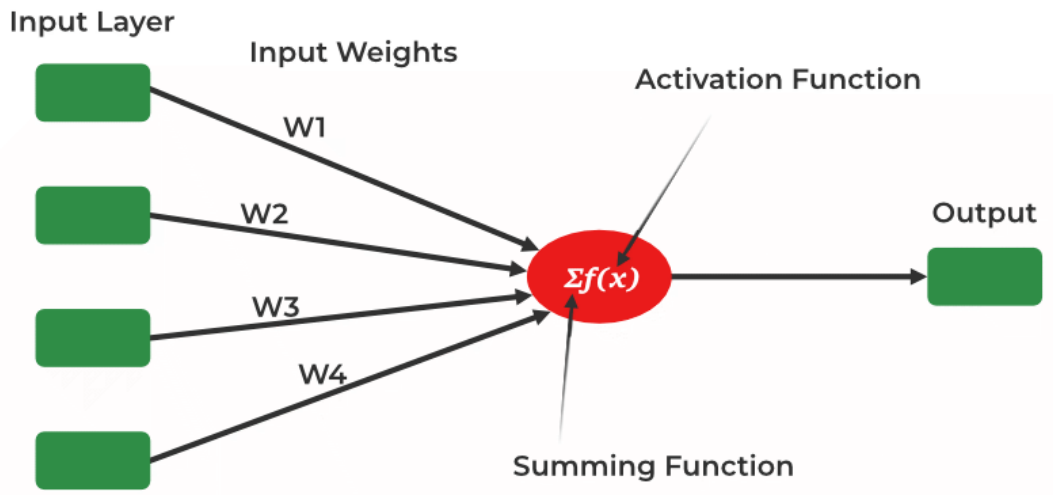


Single Layer Perceptron

It is one of the oldest and first introduced neural networks. It was proposed by **Frank Rosenblatt** in **1958**. Perceptron is also known as an artificial neural network. Perceptron is mainly used to compute the logical gate like **AND, OR and NOR** which has binary input and binary output.

The main functionality of the perceptron is:-

- Takes input from the input layer
- Weight them up and sum it up.
- Pass the sum to the nonlinear function to produce the output.

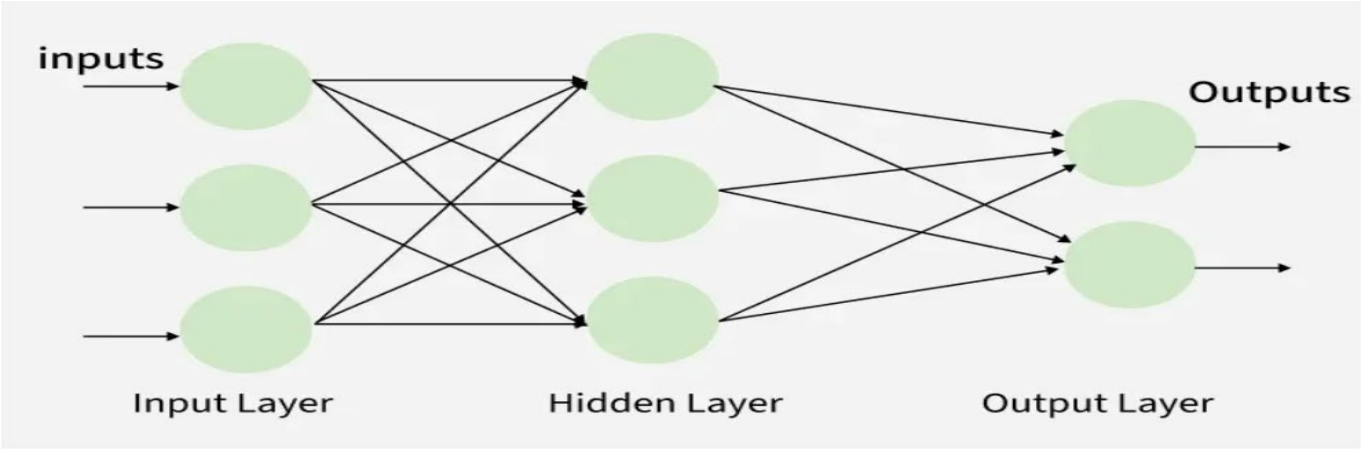


Here activation functions can be anything like **sigmoid, tanh, relu** based on the requirement we will be choosing the most appropriate nonlinear activation function to produce the better result. Now let us implement a single-layer perceptron.

Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) consists of fully connected dense layers that transform input data from one dimension to another. It is called multi-layer because it contains an input layer, one or more hidden layers and an output layer. The purpose of an MLP is to model complex relationships between inputs and outputs.

Components of Multi-Layer Perceptron (MLP)



- **Input Layer**: Each neuron or node in this layer corresponds to an input feature. For instance, if you have three input features the input layer will have three neurons.
- **Hidden Layers**: MLP can have any number of hidden layers with each layer containing any number of nodes. These layers process the information received from the input layer.
- **Output Layer**: The output layer generates the final prediction or result. If there are multiple outputs, the output layer will have a corresponding number of neurons.

Every connection in the diagram is a representation of the fully connected nature of an MLP. This means that every node in one layer connects to every node in the next layer. As the data moves through the network each layer transforms it until the final output is generated in the output layer.

Working of Multi-Layer Perceptron

Let's see working of the multi-layer perceptron. The key mechanisms such as forward propagation, loss function, backpropagation and optimization.

1. Forward Propagation

In forward propagation the data flows from the input layer to the output layer, passing through any hidden layers. Each neuron in the hidden layers processes the input as follows:

1. Weighted Sum: The neuron computes the weighted sum of the inputs:

$$z = \sum_i w_i x_i + b$$

Where:

- x_i is the input feature.
- w_i is the corresponding weight.
- b is the bias term.

2. Activation Function: The weighted sum z is passed through an activation function to introduce non-linearity. Common activation functions include:

- **Sigmoid**: $\sigma(z) = \frac{1}{1+e^{-z}}$
- **ReLU (Rectified Linear Unit)**: $f(z) = \max(0, z)$
- **Tanh (Hyperbolic Tangent)**: $\tanh(z) = \frac{2}{1+e^{-2z}} - 1$

2. Loss Function

Once the network generates an output the next step is to calculate the loss using a loss function. In supervised learning this compares the predicted output to the actual label.

For a classification problem the commonly used binary cross-entropy loss function is:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

Where:

- y_i is the actual label.
- $\hat{y}_i$ is the predicted label.
- N is the number of samples.

For regression problems the mean squared error (MSE) is often used:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

3. Backpropagation

The goal of training an MLP is to minimize the loss function by adjusting the network's weights and biases. This is achieved through backpropagation:

1. **Gradient Calculation**: The gradients of the loss function with respect to each weight and bias are calculated using the chain rule of calculus.
2. **Error Propagation**: The error is propagated back through the network, layer by layer.
3. **Gradient Descent**: The network updates the weights and biases by moving in the opposite direction of the gradient to reduce the loss: $w = w - \eta \cdot \frac{\partial L}{\partial w}$

Where:

- w is the weight.
- η is the learning rate.
- $\frac{\partial L}{\partial w}$ is the gradient of the loss function with respect to the weight.

4. Optimization

MLPs rely on optimization algorithms to iteratively refine the weights and biases during training. Popular optimization methods include:

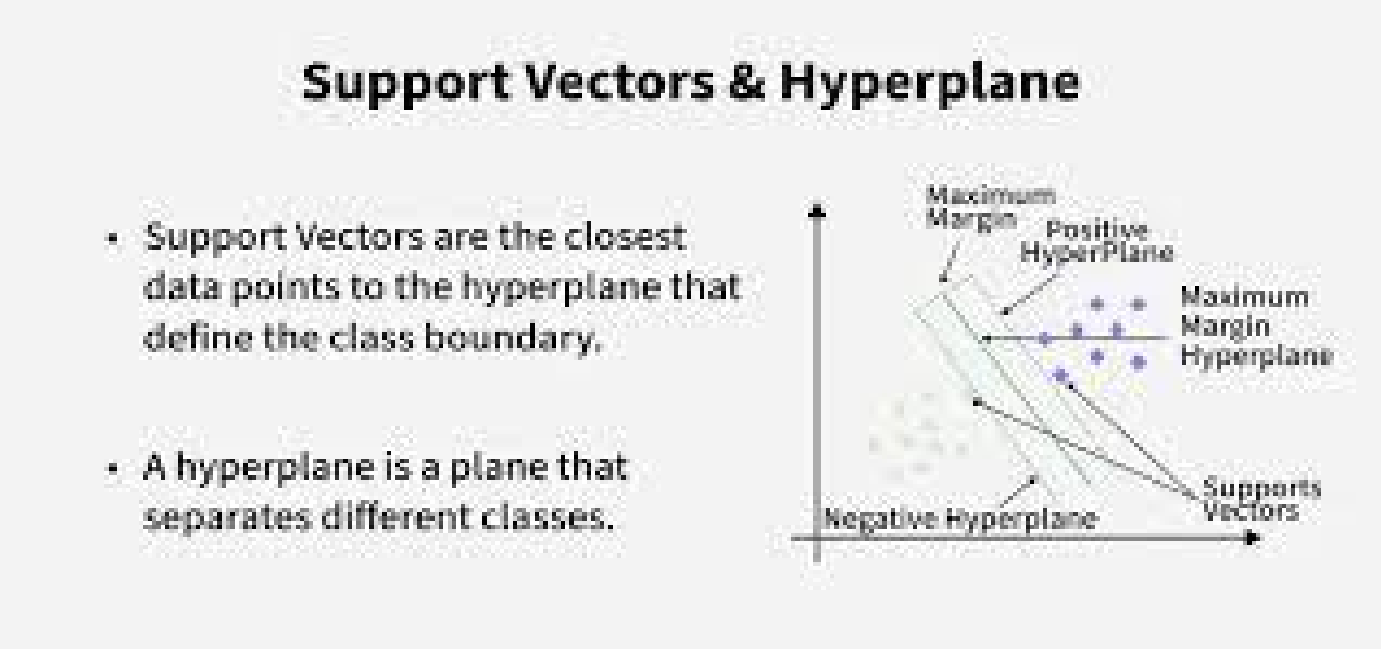
- Stochastic Gradient Descent (SGD): Updates the weights based on a single sample or a small batch of data: $w = w - \eta \cdot \frac{\partial L}{\partial w}$
- Adam Optimizer: An extension of SGD that incorporates momentum and adaptive learning rates for more efficient training:

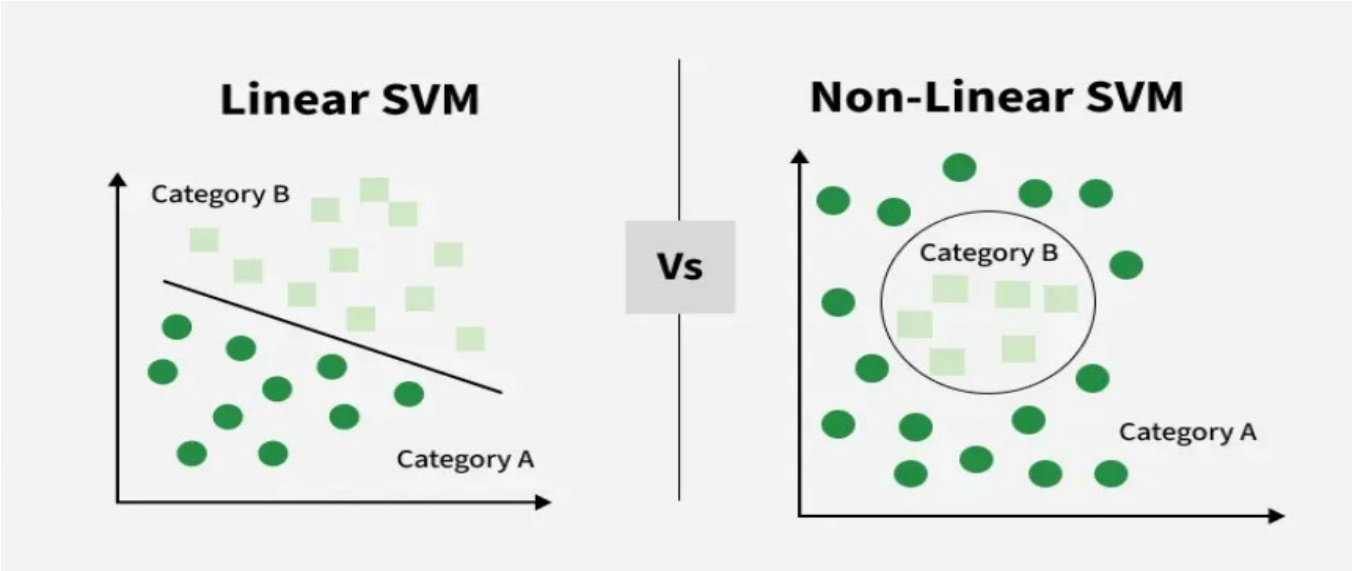
  - $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t$
  - $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2$

Here g_t represents the gradient at time t and β_1, β_2 are decay rates.

Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It tries to find the best boundary known as hyperplane that separates different classes in the data. It is useful when you want to do binary classification like spam vs. not spam or cat vs. dog.
The main goal of SVM is to maximize the margin between the two classes. The larger the margin the better the model performs on new and unseen data.



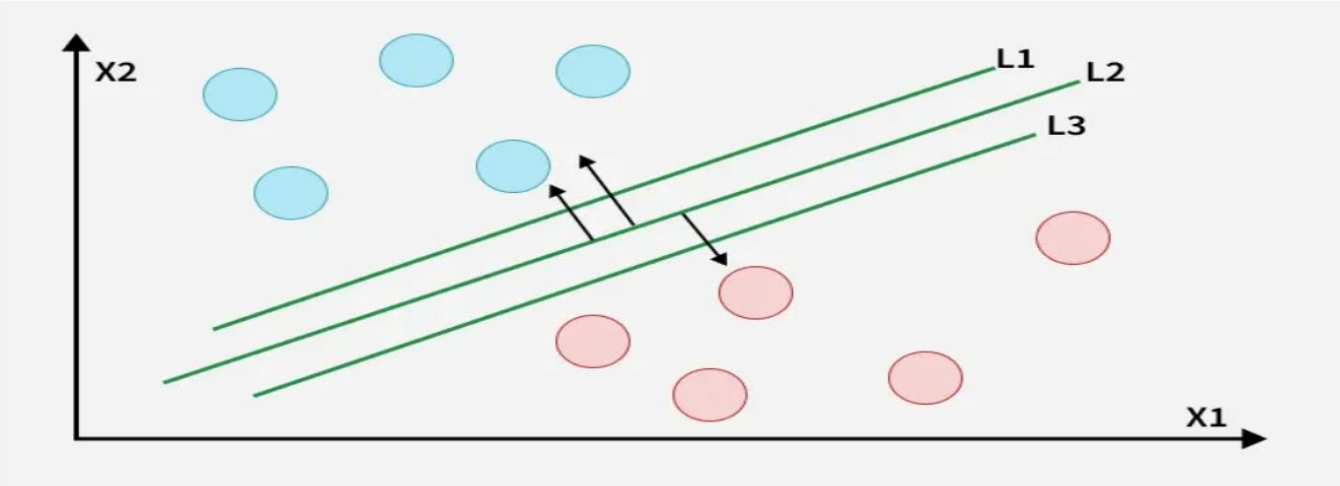


Key Concepts of Support Vector Machine

- **Hyperplane**: A decision boundary separating different classes in feature space and is represented by the equation wx + b = 0 in linear classification.
- **Support Vectors**: The closest data points to the hyperplane, crucial for determining the hyperplane and margin in SVM.
- **Margin**: The distance between the hyperplane and the support vectors. SVM aims to maximize this margin for better classification performance.
- **Kernel**: A function that maps data to a higher-dimensional space enabling SVM to handle non-linearly separable data.
- **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.
- **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.
- **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value forces stricter penalty for misclassifications.
- **Hinge Loss**: A loss function penalizing misclassified points or margin violations and is combined with regularization in SVM.
- **Dual Problem**: Involves solving for Lagrange multipliers associated with support vectors, facilitating the kernel trick and efficient computation.

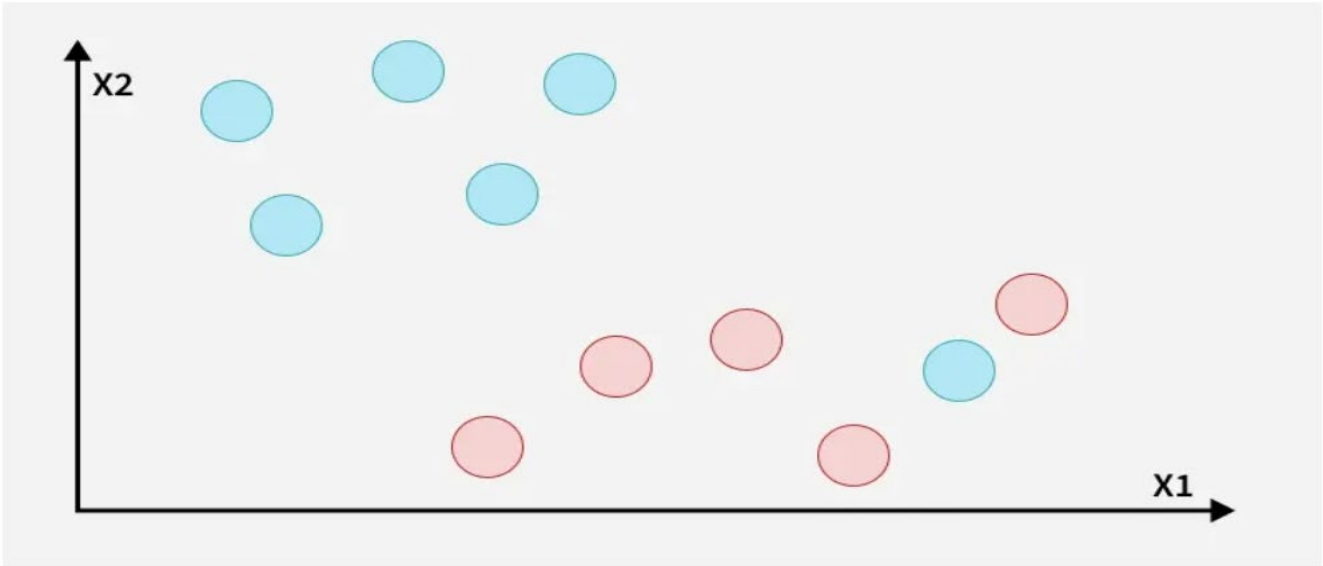
How does Support Vector Machine Algorithm Work?

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (support vectors) on each side.



Multiple hyperplanes separate the data from two classes

The best hyperplane also known as the **"hard margin"** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So from the above figure, we choose L2 as hard margin. Let's consider a scenario like shown below:

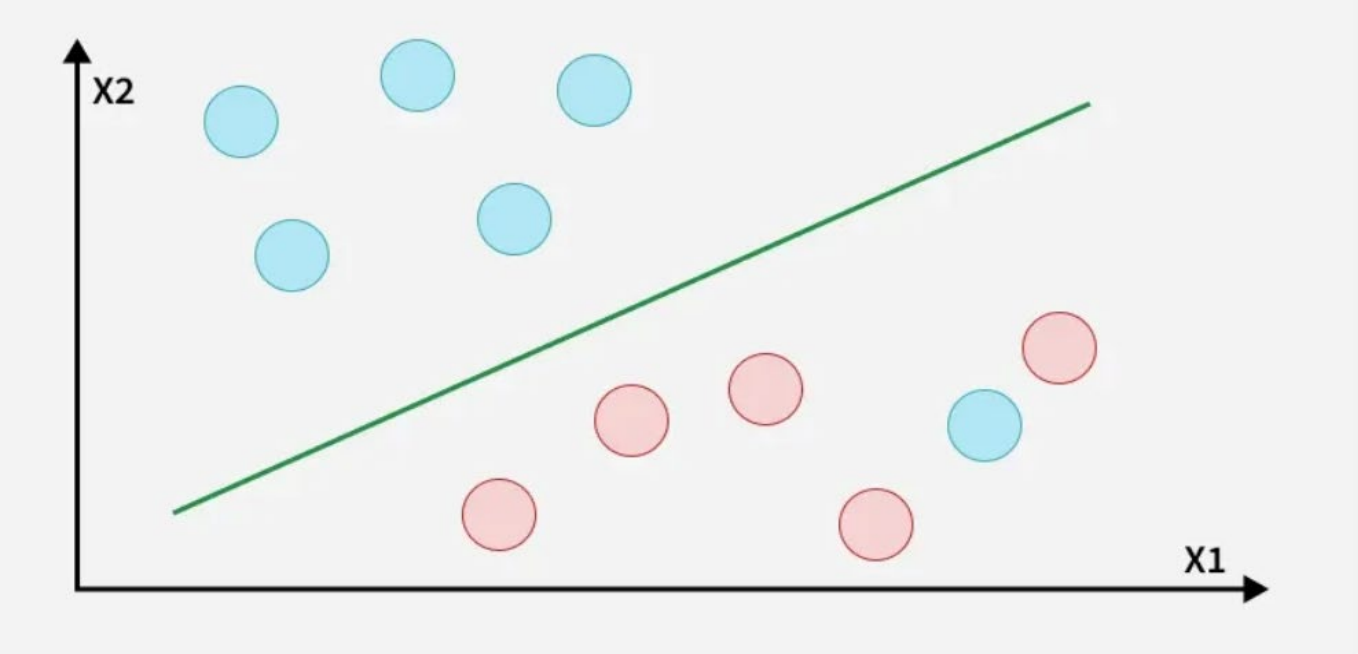


Selecting hyperplane for data with outlier

Here, we have one blue ball in the boundary of the red ball.

How does SVM classify the data?

The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



Hyperplane which is the most optimized one

A soft margin allows for some misclassifications or violations of the margin to improve generalization. The SVM optimizes the following equation to balance margin maximization and penalty minimization:

$$\text{Objective Function} = \left(\frac{1}{\text{margin}}\right) + \lambda \sum \text{penalty}$$

The penalty used for violations is often hinge loss which has the following behavior:

- If a data point is correctly classified and within the margin there is no penalty (loss = 0).
- If a point is incorrectly classified or violates the margin the hinge loss increases proportionally to the distance of the violation.

Types of Support Vector Machine

Based on the nature of the decision boundary, Support Vector Machines (SVM) can be divided into two main parts:

- **Linear SVM:** Linear SVMs use a linear decision boundary to separate the data points of different classes. When the data can be precisely linearly separated,

linear SVMs are very suitable. This means that a single straight line (in 2D) or a hyperplane (in higher dimensions) can entirely divide the data points into their respective classes. A hyperplane that maximizes the margin between the classes is the decision boundary.

- **Non-Linear SVM:** Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.

Advantages of Support Vector Machine (SVM)

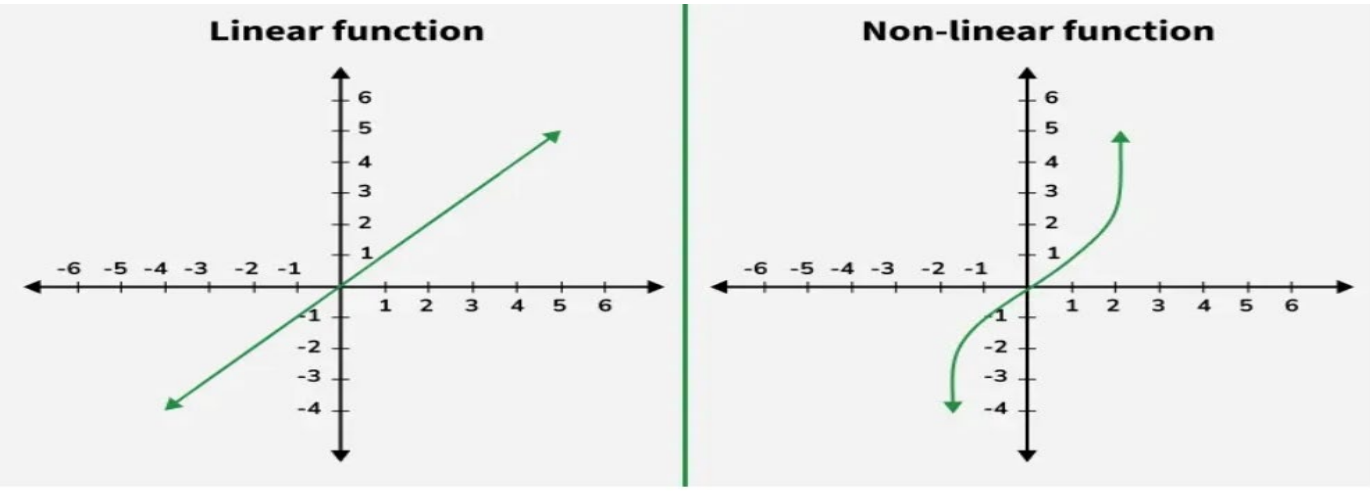
1. **High-Dimensional Performance**: SVM excels in high-dimensional spaces, making it suitable for image classification and gene expression analysis.
2. **Nonlinear Capability**: Utilizing kernel functions like RBF and polynomial SVM effectively handles nonlinear relationships.
3. **Outlier Resilience**: The soft margin feature allows SVM to ignore outliers, enhancing robustness in spam detection and anomaly detection.
4. **Binary and Multiclass Support**: SVM is effective for both binary classification and multiclass classification suitable for applications in text classification.
5. **Memory Efficiency**: It focuses on support vectors making it memory efficient compared to other algorithms.

Disadvantages of Support Vector Machine (SVM)

1. **Slow Training**: SVM can be slow for large datasets, affecting performance in SVM in data mining tasks.
2. **Parameter Tuning Difficulty**: Selecting the right kernel and adjusting parameters like C requires careful tuning, impacting SVM algorithms.
3. **Noise Sensitivity**: SVM struggles with noisy datasets and overlapping classes, limiting effectiveness in real-world scenarios.
4. **Limited Interpretability**: The complexity of the hyperplane in higher dimensions makes SVM less interpretable than other models.
5. **Feature Scaling Sensitivity**: Proper feature scaling is essential, otherwise SVM models may perform poorly.

Linear vs Non-Linear Equations

Equations are mathematical statements that show the relationship between variables and constants. They describe completely different kinds of relationships. While linear equations represent straight-line behavior and proportional changes, nonlinear equations capture more complex, curved relationships that occur in nature and real-world systems.



Function

Understanding Linear Equations

Linear equations are equations where the highest power of the variable is one. They show a constant rate of change between variables. It is used in budgeting, speed-distance-time calculations, simple data trends, unit conversions and planning linear growth scenarios.

General Form: $y=mx+c$

Here:

- **m = slope** i.e. rate of change
- **c = y-intercept** where the line crosses the y-axis

Characteristics:

- Represent straight lines on a graph.
- Show a direct proportional relationship.
- Easy to solve and interpret.
- The power of the variable is 1.

Understanding Non-Linear Equations

A non-linear equation is an equation where the power of at least one variable is greater than 1 or variables are multiplied together. It represents a curve when

plotted. It is used in physics, biology, finance, engineering and environmental modeling which involves complex relationships.

Example Forms:

- y=ax2+bx+c,
- y=ex or xy=k.

Characteristics:

- Represent curved relationships.
- The rate of change is not constant.
- Often describe natural or complex systems like population growth or motion.
- The power of the variable more than 1.

Linear vs Non-Linear Equations

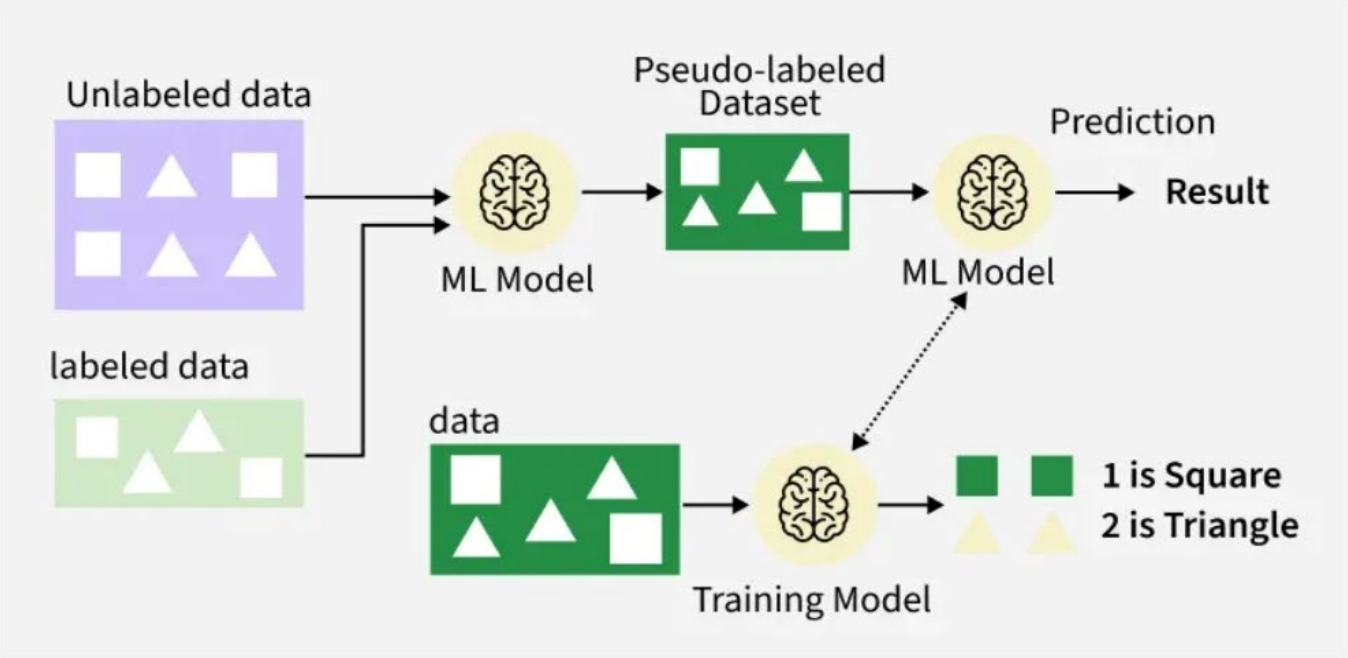
Difference between linear and nonlinear equations:

Basis	Linear Equation	Non-Linear Equation
Definition	Equation with degree 1	Equation with degree > 1 or variable products
Simplicity	Linear Equations are much simpler to solve.	Non-linear Equations are tricky in nature.
Graph Shape	Straight line	Curve (parabola, circle, etc.)
Degree of Variable	1	2 or higher
Rate of Change	Constant	Variable
Example	2x+3=02x+3=0	x2+3x+2=0x2+3x+2=0
Number of Solutions	One	One or more depending on degree

Basis	Linear Equation	Non-Linear Equation
Representation	y=mx+c	$y=ax^2+bx+c$, $x^2+y^2=r^2$ etc.

Semi-Supervised Learning in ML

Semi-supervised learning is a hybrid machine learning approach which uses both supervised and unsupervised learning. It uses a small amount of labelled data combined with a large amount of unlabeled data to train models. The goal is to learn a function that accurately predicts outputs based on inputs, similar to supervised learning, but with much less labelled data.



Semi-Supervised Learning

Semi-supervised learning is particularly valuable when acquiring labelled data is expensive or time-consuming, yet unlabelled data is plentiful and easy to collect.

- **Supervised learning**: Similar to a student being taught concepts by a teacher both in class and at home.
- **Unsupervised learning**: Like a student figuring out concepts independently without instruction like a math problem.

- **Semi-supervised learning**: A mix where the teacher provides some concepts in class and the student practices with homework assignments based on those concepts.

Working of Sem-Supervised Learning,

Several techniques fall under semi-supervised learning including:

- **Self-Training**: The model is first trained on labeled data. It then predicts labels for unlabeled data, adding high-confidence predictions to the labeled set iteratively to refine the model.
- **Co-Training:** Two models are trained on different feature subsets of the data. Each model labels unlabeled data for the other, enabling them to learn from complementary views.
- **Multi-View Training**: A variation of co-training where models train on different data representations (e.g., images and text) to predict the same output.
- **Graph-Based Models**: Data is represented as a graph with nodes (data points) and edges (similarities). Labels are propagated from labeled nodes to unlabeled ones based on graph connectivity.

When to Use Semi-Supervised Learning

- When labeled data is scarce or costly, such as medical imaging requiring expert annotation.
- When large volumes of unlabeled data exist, like social media or web content.
- For unstructured data types (text, images, audio) where labeling is difficult.
- When classes are rare and labeled examples few, improving class recognition.
- When purely supervised or unsupervised methods are insufficient.

Applications

Let's see the applications,

- **Face Recognition**: Enhancing accuracy by learning from limited labeled face images plus many unlabeled ones using graph-based methods.
- **Handwritten Text Recognition**: Adapting models to diverse handwriting styles through generative models.
- **Speech Recognition**: Improving transcription quality by using unlabeled speech data with CNNs and other techniques.

- **Security**: Google uses semi-supervised learning for anomaly detection in network traffic and malware detection.
- **Finance**: PayPal applies it for fraud detection and creditworthiness assessment using transaction data.

Advantages

- **Better Generalization**: Utilizes both labeled and unlabeled data to capture the whole data structure, improving prediction robustness.
- **Cost Efficient**: Reduces dependency on costly manual labeling by exploiting unlabeled data.
- **Flexible and Robust**: Handles different data types and sources, adapting well to changing data distributions.
- **Improved Clustering**: Refines clusters by leveraging unlabeled data, yielding better class separation.
- **Handling Rare Classes**: Enhances learning for underrepresented classes where labeled examples are minimal.

Limitations

- **Model Complexity**: Requires careful choice of architecture and hyperparameters, which may require extensive tuning.
- **Noisy Data**: Unlabeled data may contain errors or irrelevant information, risking degraded model performance.
- **Assumption Sensitivity**: Relies on assumptions such as data consistency and clusterability, which may not hold in all cases.
- **Evaluation Challenge**: Assessing performance is difficult due to limited labeled data and varied quality of unlabeled data.

Parul® University
Vadodara, Gujarat

NAAC GRADE A++

https://paruluniversity.ac.in/