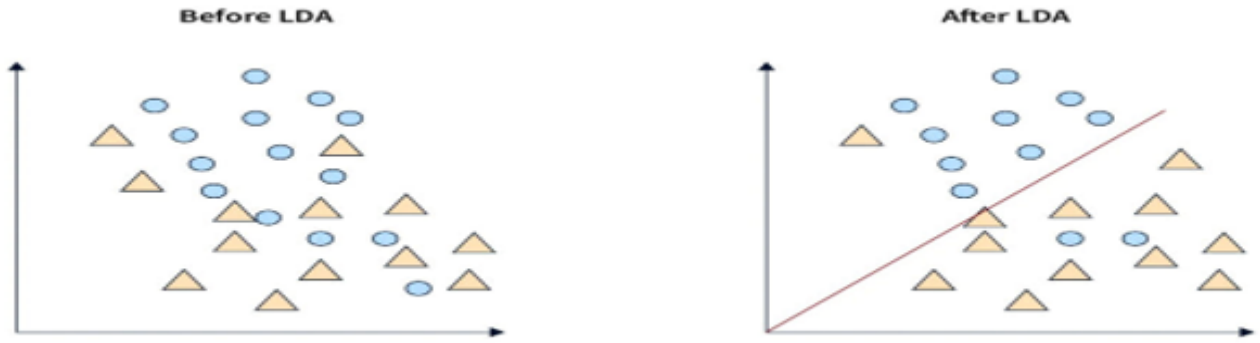
Unit - IV

UNIT-IV: Linear Discriminants for Machine Learning: Introduction to Linear Discriminants, Linear Discriminants for Classification, Perceptron Classifier, Perceptron Learning Algorithm, Support Vector Machines, Linearly Non-Separable Case, Non-linear SVM, Kernel Trick, Logistic Regression, Linear Regression, Multi-Layer Perceptrons (MLPs), Backpropagation for Training an MLP.

Introduction to Linear Discriminants:

Machine learning models are often used to solve supervised learning tasks, particularly **classification problems**, where the goal is to assign data points to specific categories or classes. However, as datasets grow larger with more features, it becomes challenging for models to process the data effectively. This is where **dimensionality reduction techniques** like Linear Discriminant Analysis (LDA) come into play.

LDA not only helps to reduce the number of features but also ensures that the important class-related information is retained, making it easier for models to differentiate between classes. This article explores LDA, its working principle, and its practical applications in various fields.



Linear Discriminant Analysis (LDA) is a supervised learning technique used for classification tasks. It helps distinguish between different classes by projecting data points onto a lower-dimensional space, maximizing the separation between those classes.

LDA performs two key roles:

1. **Classification**: It finds a linear combination of features that best separates multiple classes.
2. **Dimensionality Reduction**: It reduces the number of input features while preserving the information necessary for classification.

For example, in a dataset where each data point belongs to one of three classes, LDA transforms the data into a space where the classes are well-separated, making it easier for models to classify them correctly.

Properties and Assumptions of LDA

To perform effectively, LDA makes several key assumptions about the data:

Key Assumptions:

1. **Gaussian Distribution**: LDA assumes that the features within each class follow a normal (Gaussian) distribution.
2. **Equal Covariance Matrices**: It assumes that the variance (or spread) of data points is the same across all classes.
3. **Linearity**: The relationship between the features and the target variable is assumed to be linear.
4. **Independence**: The features used should ideally be independent of each other.

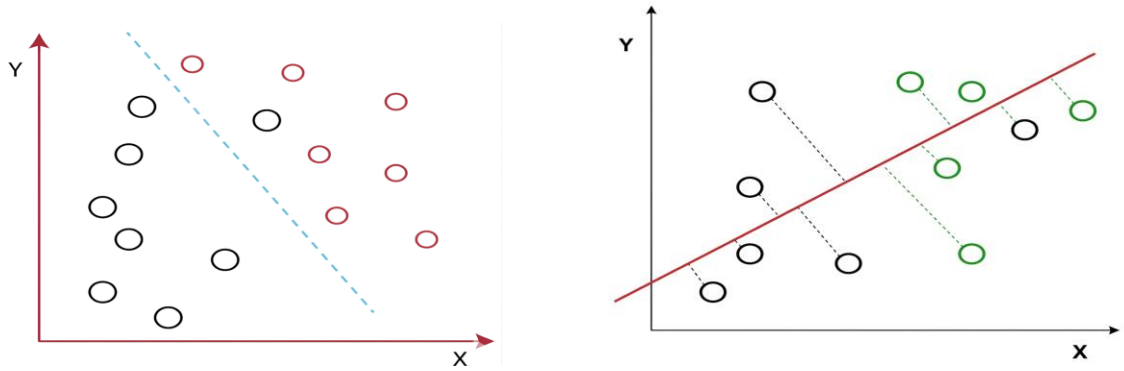
Here, the two features or classes are plotted on a 2-dimensional (2D) plane with an X-Y axis. If we tried to classify approvals using just one feature, we might observe overlap. By applying LDA, we can draw a straight line that completely separates these two class data points. LDA achieves this by using the X–Y axis to create a new axis, separating the different classes with a straight line and projecting data onto the new axis.

To create this new axis and reduce dimensionality, LDA follows these criteria:

- Maximize the distance between the means of two classes.
- Minimize the variance within individual classes.

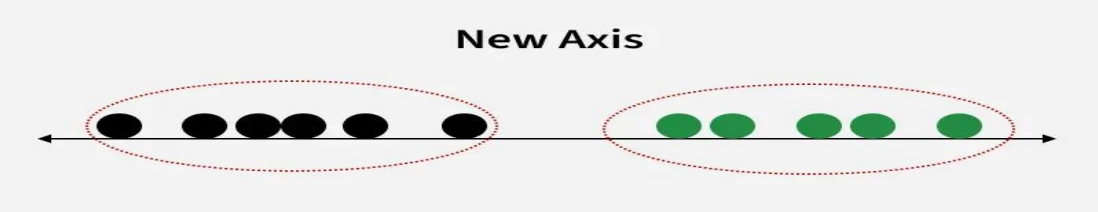
Linear Discriminants for Classification:

- Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems.
- It is used to project the features in higher dimension space into a lower dimension space.
- Suppose we have two sets of data points belonging to two different classes that we want to classify.
- When the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely.



Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.

- Two criteria are used by LDA to create a new axis:
  - Maximize the distance between means of the two classes.
  - Minimize the variation within each class.



Linear Discriminant Analysis – Steps

1. Compute the class means of dependent variable
2. Derive the covariance matrix of the class variable
3. Compute the within class — scatter matrix (S1+S2)
4. Compute the between class scatter matrix
5. Compute the Eigen values and eigen vectors from the within class and between class scatter matrix
6. Sort the values of eigen values and select the top k values
7. Find the eigen vectors corresponds to the top k eigen vectors
8. Obtain the LDA by taking the dot product of eigen vectors and original data

$$\mu_1 = \frac{1}{N_1}\sum_{x\in\omega_1} x$$

$$S_1 = \sum_{x\in\omega_1}(x-\mu_1)(x-\mu_1)^T$$

$$S_W = S_1 + S_2$$

$$S_B = (\mu_1-\mu_2)(\mu_1-\mu_2)^T$$

$$S_W^{-1}S_B w = \lambda w$$

$$\left(S_W^{-1}S_B - \lambda I\right)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = 0$$

Dataset

Step 1: Given Dataset

Two classes with two data points each.

Class C_1

$$(1, 2), (2, 3)$$

Class C_2

$$(3, 3), (4, 5)$$

Step 2: Compute Mean of Each Class

Mean formula

$$\mu = \frac{1}{n} \sum x$$

Mean of Class C_1

$$\mu_1 = \left(\frac{1+2}{2}, \frac{2+3}{2} \right)$$

$$\mu_1 = (1.5,\ 2.5)$$

Mean of Class C_2

$$\mu_2 = \left(\frac{3+4}{2}, \frac{3+5}{2} \right)$$

$$\mu_2 = (3.5,\ 4)$$

Step 3: Compute Scatter Matrix for Class C_1

Formula

$$S_1 = \sum (x - \mu_1)(x - \mu_1)^T$$

For point $(1, 2)$

$$x - \mu_1 = \begin{bmatrix} 1 - 1.5 \\ 2 - 2.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$$

$$(x - \mu_1)(x - \mu_1)^T = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

For point $(2, 3)$

$$x - \mu_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$(x - \mu_1)(x - \mu_1)^T = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

Scatter Matrix S_1

$$S_1 = \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

$$S_1 = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Step 4: Compute Scatter Matrix for Class C_2

Formula

$$S_2 = \sum (x - \mu_2)(x - \mu_2)^T$$

For point $(3, 3)$

$$x - \mu_2 = \begin{bmatrix} 3 - 3.5 \\ 3 - 4 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \end{bmatrix}$$

$$(x - \mu_2)(x - \mu_2)^T = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

For point $(4, 5)$

$$x - \mu_2 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

$$(x - \mu_2)(x - \mu_2)^T = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

Scatter Matrix S_2

$$S_2 = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 0.5 & 1 \\ 1 & 2 \end{bmatrix}$$

Scatter Matrix S_2

$$S_2 = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 0.5 & 1 \\ 1 & 2 \end{bmatrix}$$

Step 5: Compute Within-Class Scatter Matrix

$$S_W = S_1 + S_2$$

$$S_W = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} + \begin{bmatrix} 0.5 & 1 \\ 1 & 2 \end{bmatrix}$$

$$S_W = \begin{bmatrix} 1 & 1.5 \\ 1.5 & 2.5 \end{bmatrix}$$

Step 6: Compute Mean Difference

$$\mu_1 - \mu_2$$

$$= \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix} - \begin{bmatrix} 3.5 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} -2 \\ -1.5 \end{bmatrix}$$

Step 7: Compute Inverse of S_W

For matrix

$$\begin{bmatrix} 1 & 1.5 \\ 1.5 & 2.5 \end{bmatrix}$$

Determinant

$$|S_W| = (1)(2.5) - (1.5)(1.5)$$

$$= 2.5 - 2.25$$

$$= 0.25$$

Inverse formula

$$S_W^{-1} = \frac{1}{0.25} \begin{bmatrix} 2.5 & -1.5 \\ -1.5 & 1 \end{bmatrix}$$

$$S_W^{-1} = \begin{bmatrix} 10 & -6 \\ -6 & 4 \end{bmatrix}$$

Step 8: Compute LDA Projection Direction

Formula

$$w = S_W^{-1}(\mu_1 - \mu_2)$$

Substitute values

$$w = \begin{bmatrix} 10 & -6 \\ -6 & 4 \end{bmatrix} \begin{bmatrix} -2 \\ -1.5 \end{bmatrix}$$

Multiply

First value

$$10(-2) + (-6)(-1.5)$$
$$= -20 + 9$$
$$= -11$$

Second value

$$-6(-2) + 4(-1.5)$$
$$= 12 - 6$$
$$= 6$$

Step 9: Final LDA Projection Vector

$$w = \begin{bmatrix} -11 \\ 6 \end{bmatrix}$$

The **optimal LDA projection direction** is

$$w = (-11,\ 6)$$

This direction maximizes the **separation between the two classes.**

Step 10: LDA Projection Function

The projection of any point x onto the LDA direction is

$$y = w^T x$$

For 2-D data:

$$y = w_1 x_1 + w_2 x_2$$

Substitute $w_1 = -11$ and $w_2 = 6$

$$y = -11x_1 + 6x_2$$

This is the **LDA projection equation.**

Project class x1:

Point (1,2)

$$y = -11(1) + 6(2)$$

$$y = -11 + 12$$

$$y = 1$$

Point (2,3)

$$y = -11(2) + 6(3)$$

$$y = -22 + 18$$

$$y = -4$$

Class 1: (1, -4)

Point (3,3)

$$y = -11(3) + 6(3)$$

$$y = -33 + 18$$

$$y = -15$$

Point (4,5)

$$y = -11(4) + 6(5)$$

$$y = -44 + 30$$

$$y = -14$$

Class 2: (-15, -14)

LDA separates classes using a **threshold between class projections.**

Average projections:

Class C_1

$$\frac{1 + (-4)}{2} = -1.5$$

Class C_2

$$\frac{-15 + (-14)}{2} = -14.5$$

Threshold

$$\theta = \frac{-1.5 + (-14.5)}{2}$$

$$\theta = -8$$

Final LDA Classifier

Decision rule:

$$y = -11x_1 + 6x_2$$

Classification rule:

- If $y > -8 \rightarrow$ **Class C_1**
- If $y < -8 \rightarrow$ **Class C_2**

Test Point (2,2)

Substitute $x_1 = 2, x_2 = 2$

$$y = -11(2) + 6(2)$$

$$y = -22 + 12$$

$$y = -10$$

Now compare with threshold

$$-10 < -8$$

Therefore

(2,2) → Class C2

Test Point (1,3)

Substitute $x_1 = 1, x_2 = 3$

$$y = -11(1) + 6(3)$$

$$y = -11 + 18$$

$$y = 7$$

Now compare

$$7 > -8$$

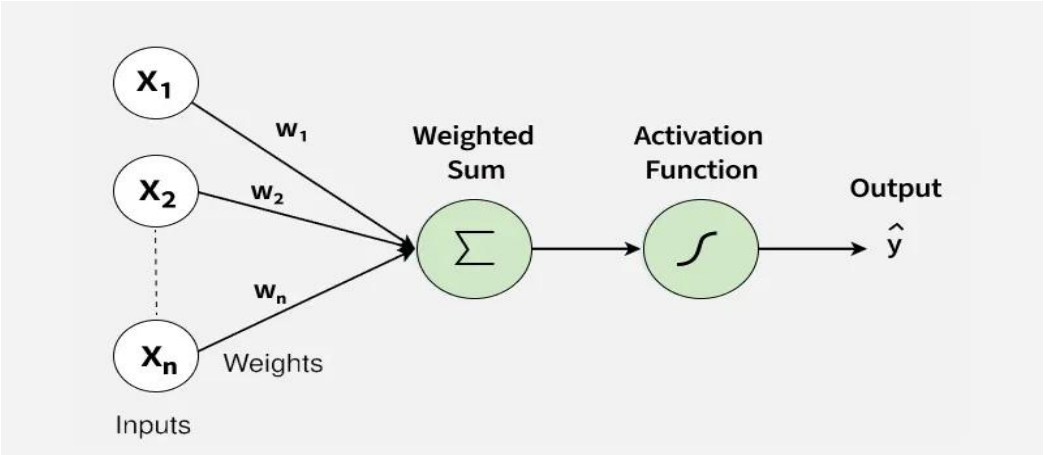
Therefore

(1,3) → Class C1

Perceptron Learning Algorithm:

Perceptron:

A simple binary linear classifier called a perceptron generates predictions based on the weighted average of the input data. Based on whether the weighted total exceeds a predetermined threshold, a threshold function determines whether to output a 0 or a 1. One of the earliest and most basic machine learning methods used for binary classification is the perceptron. Frank Rosenblatt created it in the late 1950s, and it is a key component of more intricate neural network topologies.



Perceptron Algorithm

Step 1: Initialize weights and bias

$$w_1 = 0, \quad w_2 = 0, \quad b = 0$$

Step 2: For each training example

Compute

$$f(x) = w_1 x_1 + w_2 x_2 + b$$

Step 3: Check classification

- If correctly classified → no change
- If misclassified → update weights

Update rule

$$w_1 = w_1 + \eta y x_1$$

$$w_2 = w_2 + \eta y x_2$$

$$b = b + \eta y$$

Step 4: Repeat until all points are correctly classified.

Perceptron Classifier:

The Perceptron is a supervised learning algorithm used for binary classification.

It finds a linear decision boundary that separates two classes.

Model Equation:

The perceptron computes a **linear combination of inputs**.

$$y = w_1 x_1 + w_2 x_2 + b$$

Where

- x_1, x_2 = input features
- w_1, w_2 = weights
- b = bias
- y = output value

Decision Rule

The predicted class is determined using the **sign function**.

$$\text{Class} = \begin{cases} +1 & \text{if } w_1 x_1 + w_2 x_2 + b > 0 \\ -1 & \text{if } w_1 x_1 + w_2 x_2 + b < 0 \end{cases}$$

Decision Boundary

The **decision boundary** occurs when

$$w_1 x_1 + w_2 x_2 + b = 0$$

Perceptron Learning Rule

Weights are updated when a point is **misclassified**.

Update formulas

$$w_1 = w_1 + \eta y x_1$$

$$w_2 = w_2 + \eta y x_2$$

$$b = b + \eta y$$

Where

- η = learning rate
- y = true class label (+1 or −1)
- x_1, x_2 = input features

Example Dataset:

Point	x_1	x_2	Class
A	1	1	+1
B	2	1	+1
C	1	3	-1
D	2	3	-1

Step 1:

Assume

Learning rate

$$\eta = 1$$

Initial values

$$w_1 = 0, \quad w_2 = 0, \quad b = 0$$

Decision function

$$f(x) = w_1 x_1 + w_2 x_2 + b$$

Update rule

$$w_1 = w_1 + \eta y x_1$$

$$w_2 = w_2 + \eta y x_2$$

$$b = b + \eta y$$

Step 2:

Iteration 1

Point A (1,1), $y = +1$

Substitute values

$$f(x) = (0)(1) + (0)(1) + 0$$

$$f(x) = 0$$

Misclassified → update

$$w_1 = 0 + (1)(1)(1) = 1$$

$$w_2 = 0 + (1)(1)(1) = 1$$

$$b = 0 + (1)(1) = 1$$

New values

$$w_1 = 1,\ w_2 = 1,\ b = 1$$

Iteration 2

Point B (2,1), $y = +1$

$$f(x) = 1(2) + 1(1) + 1$$

$$f(x) = 4$$

Correct classification → **no update**

Iteration 3

Point C (1,3), $y = -1$

$$f(x) = 1(1) + 1(3) + 1$$

$$f(x) = 5$$

Misclassified → update

$$w_1 = 1 + (1)(-1)(1)$$

$$w_1 = 0$$

$$w_2 = 1 + (1)(-1)(3)$$

$$w_2 = -2$$

$$b = 1 + (1)(-1)$$

$$b = 0$$

New values

$$w_1 = 0,\ w_2 = -2,\ b = 0$$

Iteration 4

Point D (2,3), $y = -1$

$$f(x) = 0(2) + (-2)(3) + 0$$
$$f(x) = -6$$

Correct classification → **no update**

Iteration 5

Point A (1,1), $y = +1$

$$f(x) = 0(1) + (-2)(1)$$
$$f(x) = -2$$

Misclassified → update

$$w_1 = 0 + (1)(1)(1)$$
$$w_1 = 1$$
$$w_2 = -2 + (1)(1)(1)$$
$$w_2 = -1$$
$$b = 0 + (1)(1)$$
$$b = 1$$

New values

$$w_1 = 1,\ w_2 = -1,\ b = 1$$

Remaining Points

Check all points again

B

$$f(x) = 1(2) + (-1)(1) + 1 = 2$$

Correct

C

$$f(x) = 1(1) + (-1)(3) + 1 = -1$$

Correct

D

$$f(x) = 1(2) + (-1)(3) + 1 = 0$$

Correct

Training stops.

Final Parameters

$$w_1 = 1$$

$$w_2 = -1$$

$$b = 1$$

Decision Boundary

Decision boundary occurs when

$$w_1 x_1 + w_2 x_2 + b = 0$$

Substitute values

$$x_1 - x_2 + 1 = 0$$

Solve for x_2

$$x_2 = x_1 + 1$$

Final Result

Perceptron classifier

$$f(x) = x_1 - x_2 + 1$$

Decision rule

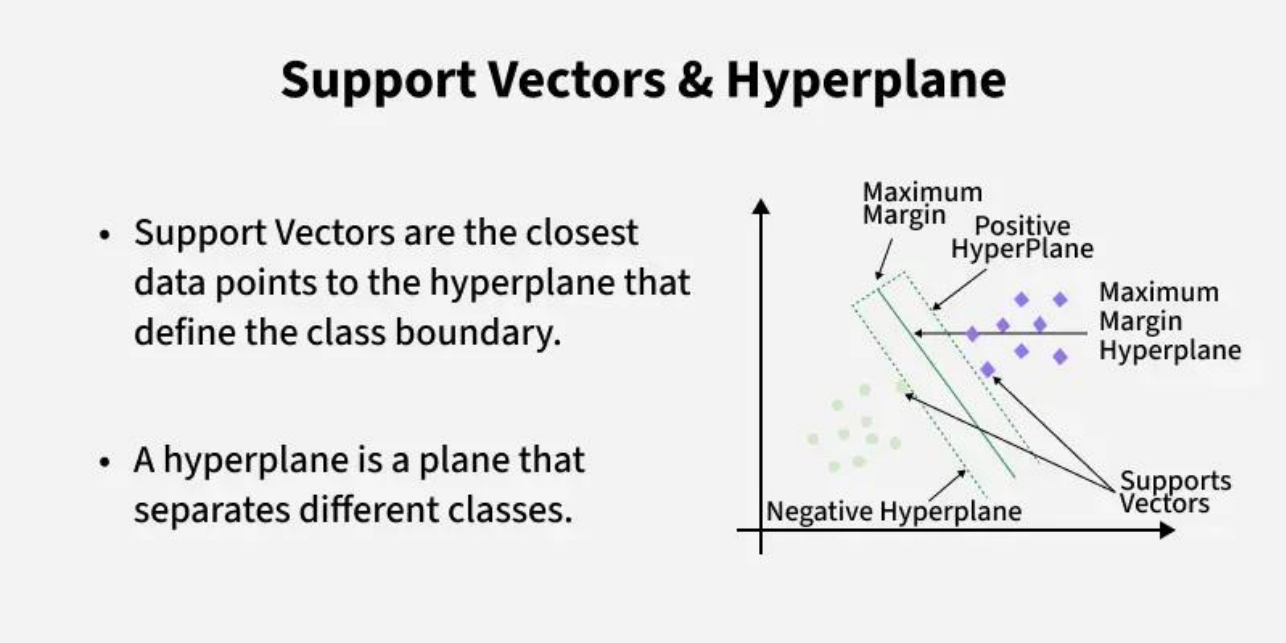
- $f(x) > 0 \rightarrow$ Class +1
- $f(x) < 0 \rightarrow$ Class -1

Support Vector Machines:

Support Vector Machines or SVMs are supervised machine learning models i.e. they use labeled datasets to train the algorithms. SVM can work out for both linear and nonlinear problems, and by the notion of margin, it classifies between various classes. However, in essence, it is used for Classification problems in Machine Learning. The objective of the algorithm is to find the finest line or decision boundary that can separate n-dimensional space into classes such that one can

put the new data points in the right class in the future. This decision boundary is called a **hyperplane**. In most of the cases, SVMs have a cut above precision than Decision Trees, KNNs, Naive Bayes Classifiers, logistic regressions, etc. In addition to this SVMs have been well known to outmatch neural networks on a few occasions. SVMs are highly recommended due to their easier implementation, and higher accuracy with less computation.

The SVM algorithm is widely used in machine learning as it can handle both linear and nonlinear classification tasks. However, when the data is not linearly separable, kernel functions are used to transform the data higher-dimensional space to enable linear separation. This application of kernel functions can be known as the —kernel trick, and the choice of kernel function, such as linear kernels, polynomial kernels, radial basis function (RBF) kernels, or sigmoid kernels, depends on data characteristics and the specific use case.



Based on the training sets, SVM are basically of two types-

1. **Linear SVM** – *Used for data that is **linearly separable**, meaning a single straight line (or flat hyperplane in higher dimensions) can effectively divide the classes.*

   Hard Margin SVM: A strict approach used when the data is perfectly separable without any outliers, aiming for a hyperplane that makes no classification errors.

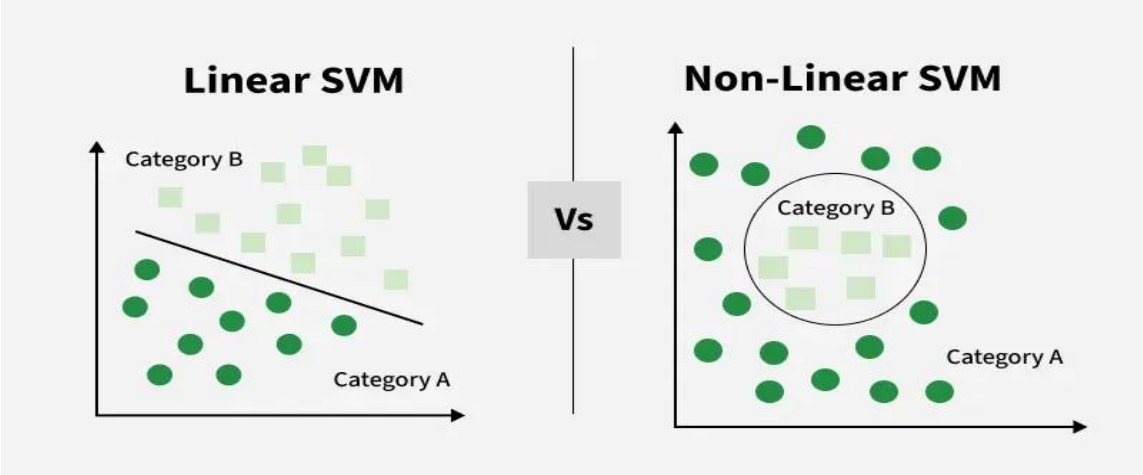
   Soft Margin SVM: A more flexible approach that allows for some misclassifications or points within the margin (by using slack variables) to handle noisy or overlapping data, balancing margin maximization and error penalties.

2. **Non-Linear SVM** – Used when the data cannot be separated by a straight line in its original form. It uses the **kernel trick** to implicitly map the data into a higher-dimensional feature space where a linear separation becomes possible. Common kernel functions include:

   Polynomial Kernel: Suitable for capturing mild curvature in data patterns.

   Radial Basis Function (RBF) Kernel (also known as Gaussian kernel): A common default choice that performs well with complex, non-linear data patterns.

   Sigmoid Kernel: Uses a hyperbolic tangent function and can be useful for modeling data with neural network-like distributions.



Before going into the working of SVM, let us quickly understand the following terms.

- **Margin –** *Margin is the gap between the hyperplane and the support vectors.*
- **Hyperplane** – *Hyperplanes are decision boundaries that aid in classifying the data points.*
- **Support Vectors** – *Support Vectors are the data points that are on or nearest to the hyperplane and influence the position of the hyperplane.*
- **Kernel function** – *These are the functions used to determine the shape of the hyperplane and decision boundary.*

Applications of SVM

- Classification of text/hypertext
- Image classification
- Classification of satellite data such as Synthetic-Aperture Radar
- Classifying biological substances like proteins

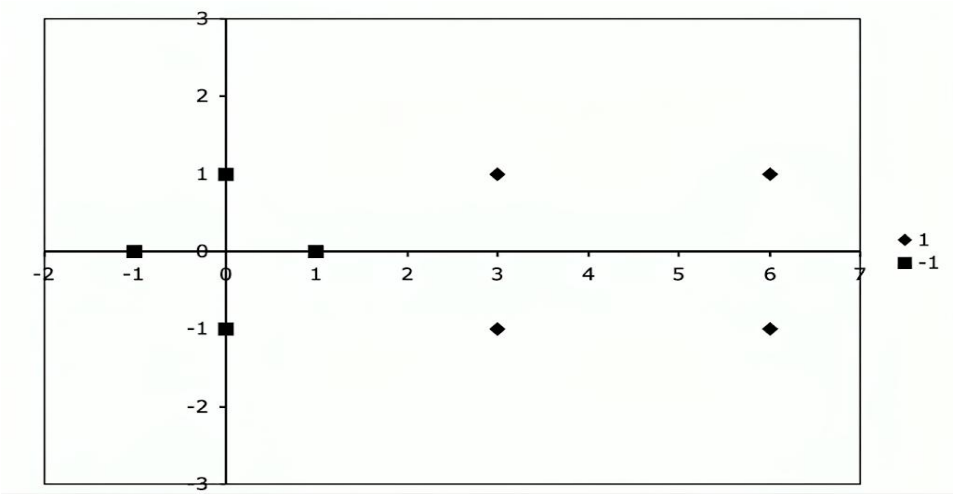
Linear SVM Example:

Suppose we are given the following positively labeled data points:

$$\{(3,1),\ (3,-1),\ (6,1),\ (6,-1)\}$$

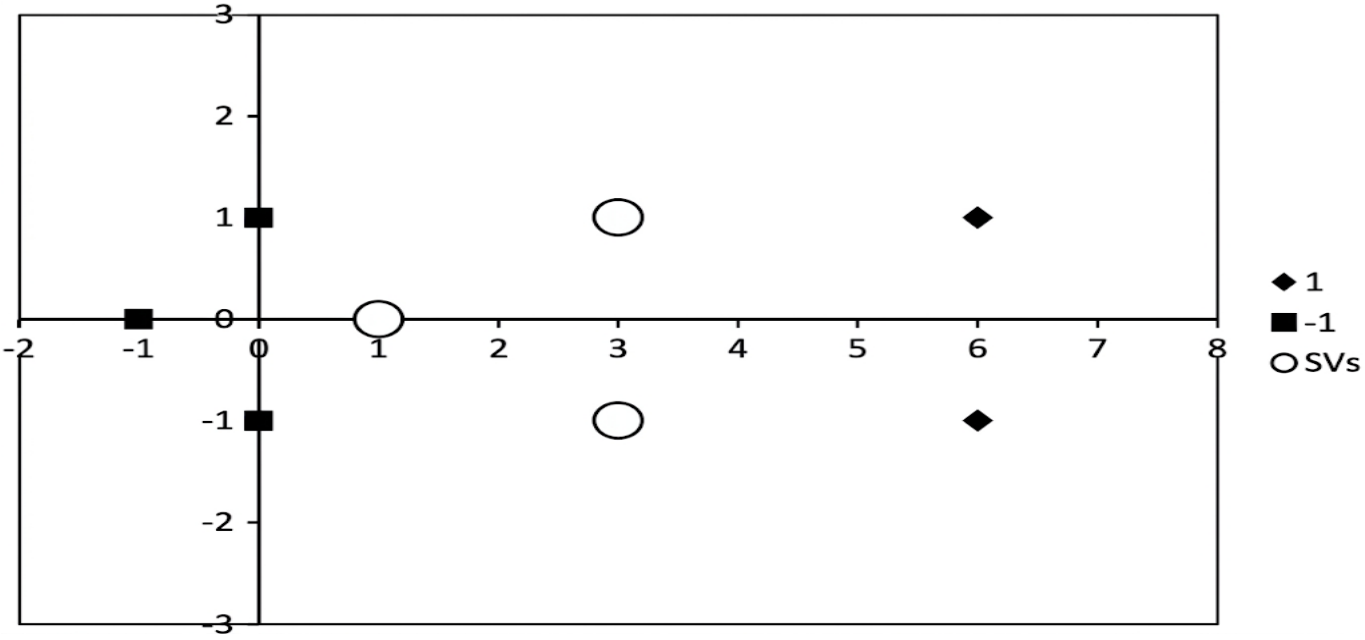
And the following negatively labeled data points:

$$\{(1,0),\ (0,1),\ (0,-1),\ (-1,0)\}$$



- By inspection, it should be obvious that there are **three support vectors**,

$$\left\{ s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\quad s_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix},\quad s_3 = \begin{pmatrix} 3 \\ -1 \end{pmatrix} \right\}$$



- Each vector is augmented with a **1 as a bias input**
  - So,

$$s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{then} \quad \tilde{s}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

  - Similarly,

$$s_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \quad \text{then} \quad \tilde{s}_2 = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

$$s_3 = \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \quad \text{then} \quad \tilde{s}_3 = \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}$$

$$\begin{aligned} \alpha_1 \tilde{s_1} \cdot \tilde{s_1} + \alpha_2 \tilde{s_2} \cdot \tilde{s_1} + \alpha_3 \tilde{s_3} \cdot \tilde{s_1} &= -1 \\ \alpha_1 \tilde{s_1} \cdot \tilde{s_2} + \alpha_2 \tilde{s_2} \cdot \tilde{s_2} + \alpha_3 \tilde{s_3} \cdot \tilde{s_2} &= +1 \\ \alpha_1 \tilde{s_1} \cdot \tilde{s_3} + \alpha_2 \tilde{s_2} \cdot \tilde{s_3} + \alpha_3 \tilde{s_3} \cdot \tilde{s_3} &= +1 \end{aligned}$$

$$\begin{aligned} \alpha_1(1+0+1)+\alpha_2(3+0+1)+\alpha_3(3+0+1) &= -1 \\ \alpha_1(3+0+1)+\alpha_2(9+1+1)+\alpha_3(9-1+1) &= 1 \\ \alpha_1(3+0+1)+\alpha_2(9-1+1)+\alpha_3(9+1+1) &= 1 \end{aligned}$$

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} = 1$$

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}\begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = 1$$

$$\begin{aligned} 2\alpha_1 + 4\alpha_2 + 4\alpha_3 &= -1 \\ 4\alpha_1 + 11\alpha_2 + 9\alpha_3 &= 1 \\ 4\alpha_1 + 9\alpha_2 + 11\alpha_3 &= 1 \end{aligned}$$

$$\begin{aligned} \alpha_1 &= -3.5 \\ \alpha_2 &= 0.75 \\ \alpha_3 &= 0.75 \end{aligned}$$

$$\tilde{w} = \sum_i \alpha_i \tilde{s}_i$$

$$= -3.5 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

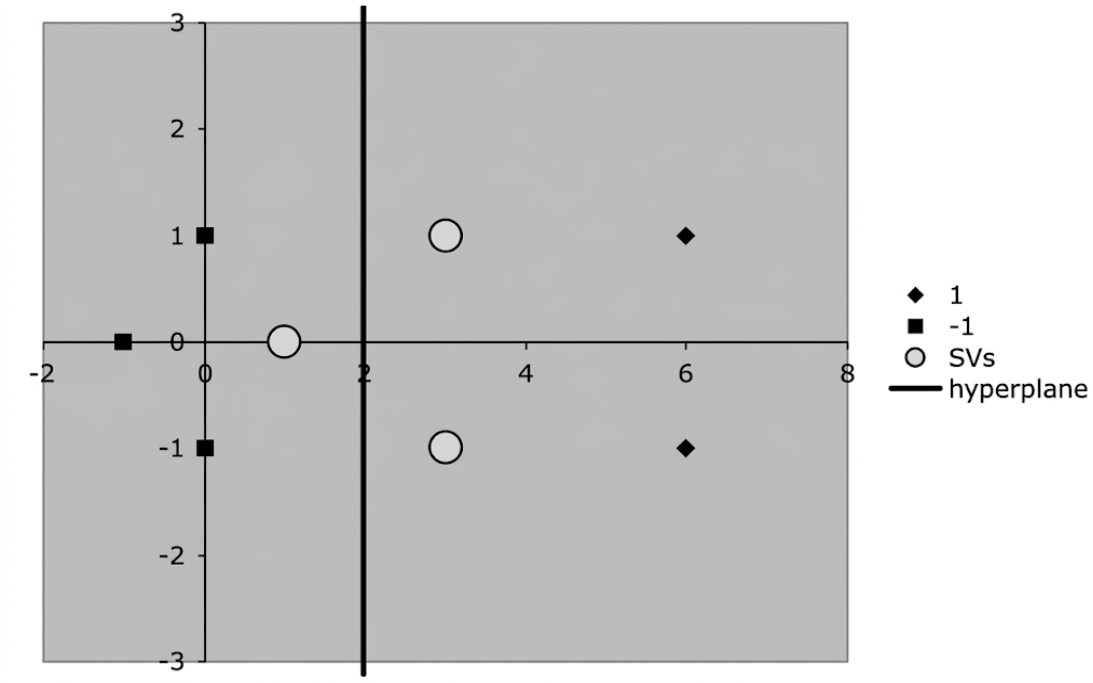
- Finally, remembering that our vectors are augmented with a bias.

- We can equate the last entry in $\tilde{w}$ as the hyperplane offset b and write the separating:
  - Hyperplane equation:

$$y = wx + b$$

  - with:

$$w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad b = -2$$

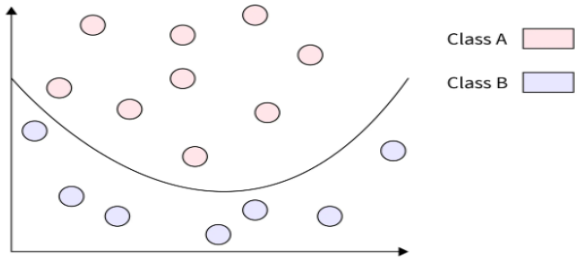


Linearly Non-Separable Case:

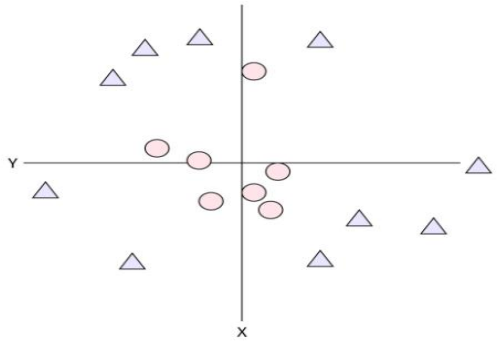
A linearly non-separable case occurs when data classes cannot be separated by a straight line or hyperplane, requiring advanced techniques to handle mixed or complex data distributions. Common approaches include using soft-margin SVMs (allowing misclassifications via slack variables) or **kernel functions** (mapping data to higher dimensions), such as RBF or polynomial kernels, to create flexible, non-linear boundaries.

Non-linear SVM:

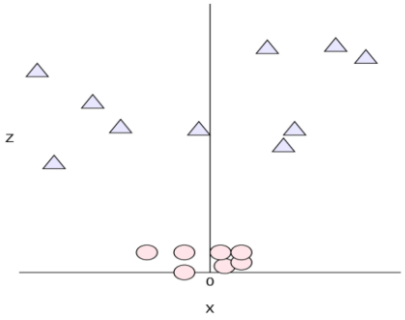
- When the data is not linearly separable, we use the non-linear SVM classifier to separate the data points.



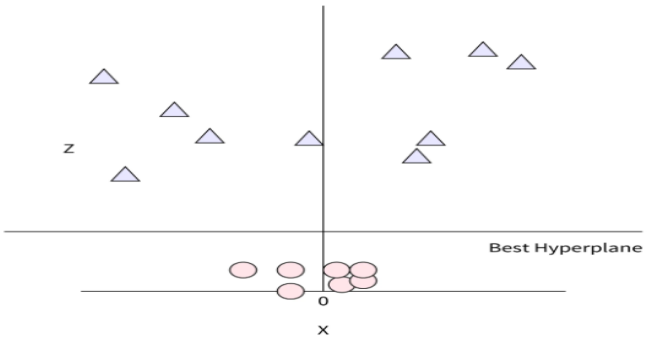
We cannot separate the above data points with a single line. Also, if we give more than two classes, it is impossible to separate them with a single straight line. Consider the following example: -



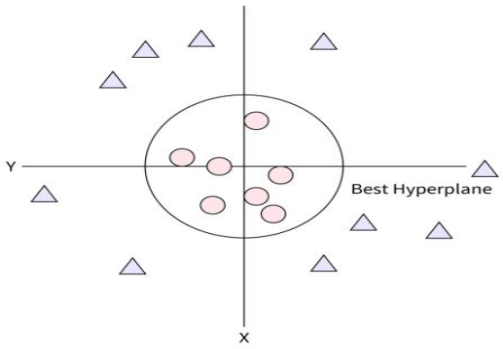
There are two classes of data points that a straight line cannot separate. But a circular hyperplane can separate them, hence we can introduce a coordinate Z, with the help of X and Y, where $Z=X^2+Y^2$. Now after introducing the third dimension, the graph changes to: -



The above depiction of data points is linearly separable and can be separated by a straight-line hyperplane.



This representation is in 3-d with z-axis. In 2-D, the graph looks like this: -



This is what a non-linear SVM does! It takes the data points to a higher dimension to be linearly separable in that dimension, and then the algorithm classifies them.

Radial Basis Function Kernel

The radial basis kernel is a kernel function used in machine learning to find a non-linear classifier or regression line.

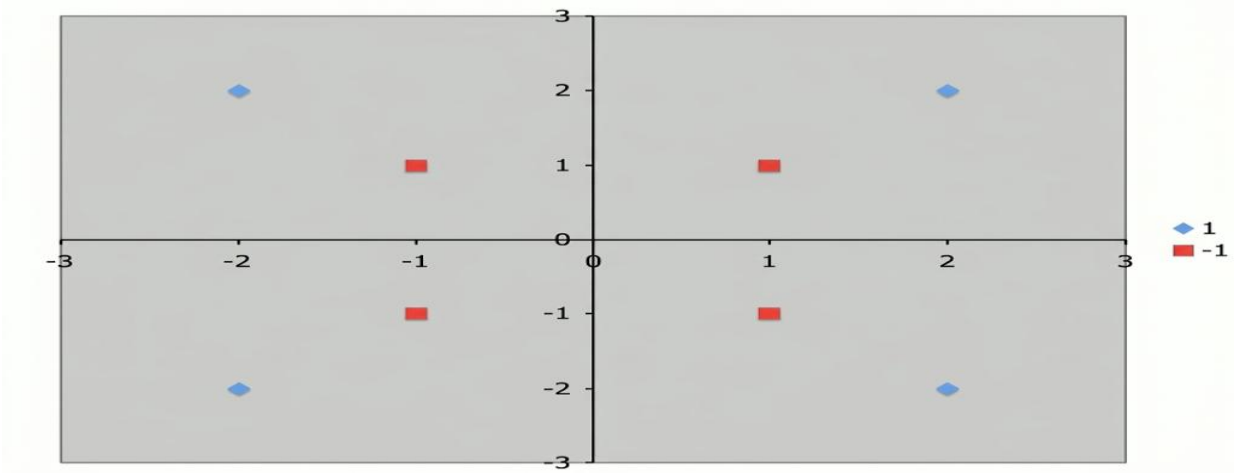
Example:

Suppose we are given the following **positively labeled data points**:

$$\left\{\begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\-2\end{pmatrix}, \begin{pmatrix}-2\\-2\end{pmatrix}, \begin{pmatrix}-2\\2\end{pmatrix}\right\}$$

and the following **negatively labeled data points**:

$$\left\{\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}\right\}$$



- Our goal, again, is to discover a separating hyperplane that accurately discriminates the two classes.
- Of course, it is obvious that no such hyperplane exists in the input space.

- Therefore, we must use a nonlinear SVM (that is, we need to convert data from one feature space to another).

$$\Phi_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 4 - x_2 + |x_1 - x_2| \\ 4 - x_1 + |x_1 - x_2| \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} > 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases}$$

Positive Examples

Given points

$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix} \right\}$$

Check the condition

$$\sqrt{x_1^2 + x_2^2} > 2$$

For all these points the condition is true, so we apply the nonlinear mapping.

Transformations

Thus

$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix} \right\} \rightarrow \left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 10 \\ 6 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 6 \\ 10 \end{pmatrix} \right\}$$

Negative Examples

Given

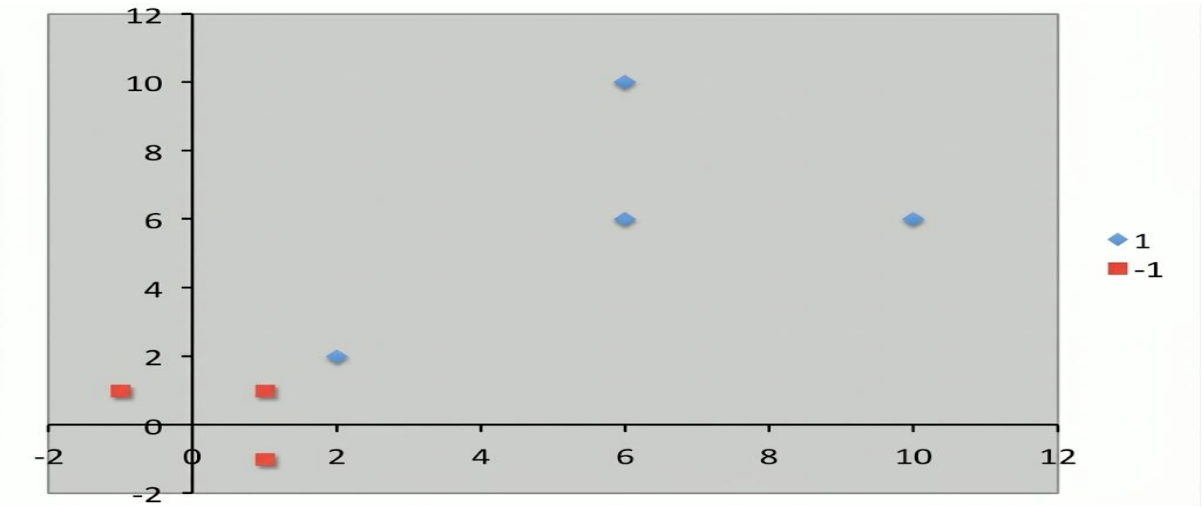
$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$

For these points

$$\sqrt{x_1^2 + x_2^2} < 2$$

Therefore the **otherwise condition** applies and the points remain unchanged.

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\} \rightarrow \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$$



- Now we can easily identify the support vectors,

$$\left\{ s_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\}$$

- Each vector is augmented with a 1 as a bias input

$$\tilde{s}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \tilde{s}_2 = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_1 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_1 = -1 \qquad \alpha_1(1+1+1) + \alpha_2(2+2+1) = -1$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_2 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_2 = +1 \qquad \alpha_1(2+2+1) + \alpha_2(4+4+1) = 1$$

$$\alpha_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1 \qquad 3\alpha_1 + 5\alpha_2 = -1$$

$$5\alpha_1 + 9\alpha_2 = 1$$

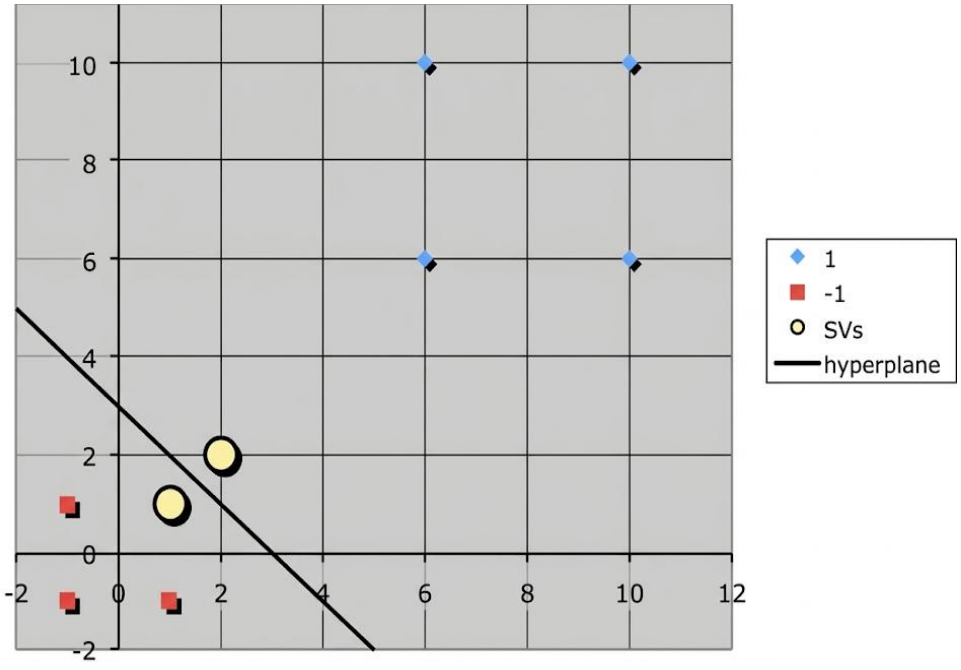
$$\alpha_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = 1 \qquad \alpha_1 = -7$$

$$\alpha_2 = 4$$

Here is the text extracted from the image:

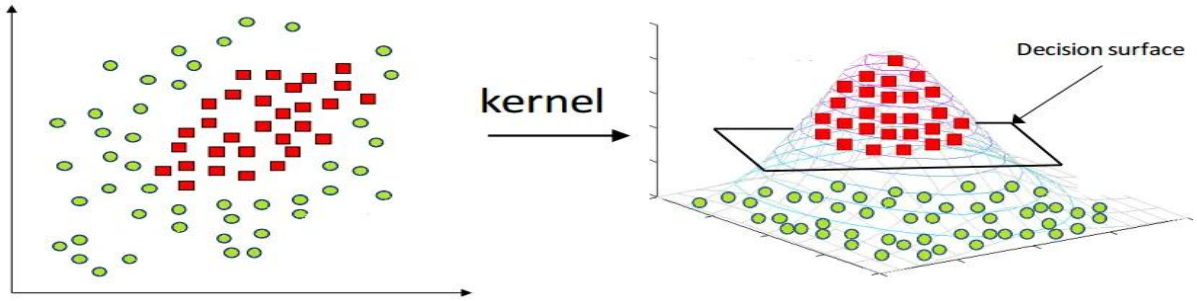
$$\tilde{w} = \sum_i \alpha_i \tilde{s}_i = -7 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix}$$

- Finally, remembering that our vectors are augmented with a bias.
- We can equate the last entry in $\tilde{w}$ as the hyperplane offset b and write the separating
- Hyperplane equation $y = wx + b$
- with $w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $b = -3$.



Kernel Trick:

The kernel trick is a method used in SVMs to enable them to classify non-linear data using a linear classifier. By applying a kernel function, SVMs can implicitly map input data into a higher-dimensional space where a linear separator (hyperplane) can be used to divide the classes. This mapping is computationally efficient because it avoids the direct calculation of the coordinates in this higher space.



Types of Kernel Functions

- **Linear Kernel**: No mapping is needed as the data is already assumed to be linearly separable.

$$k(x, y) = x^T . y$$

- **Polynomial Kernel**: Maps inputs into a polynomial feature space, enhancing the classifier's ability to capture interactions between features.

$$k(x, y) = (x^T y)^q$$

- **Radial Basis Function (RBF) Kernel**: Also known as the Gaussian kernel, it is useful for capturing complex regions by considering the distance between points in the input space.

$$k(x, y) = e^{\frac{-(x-y)^2}{2\sigma^2}}$$

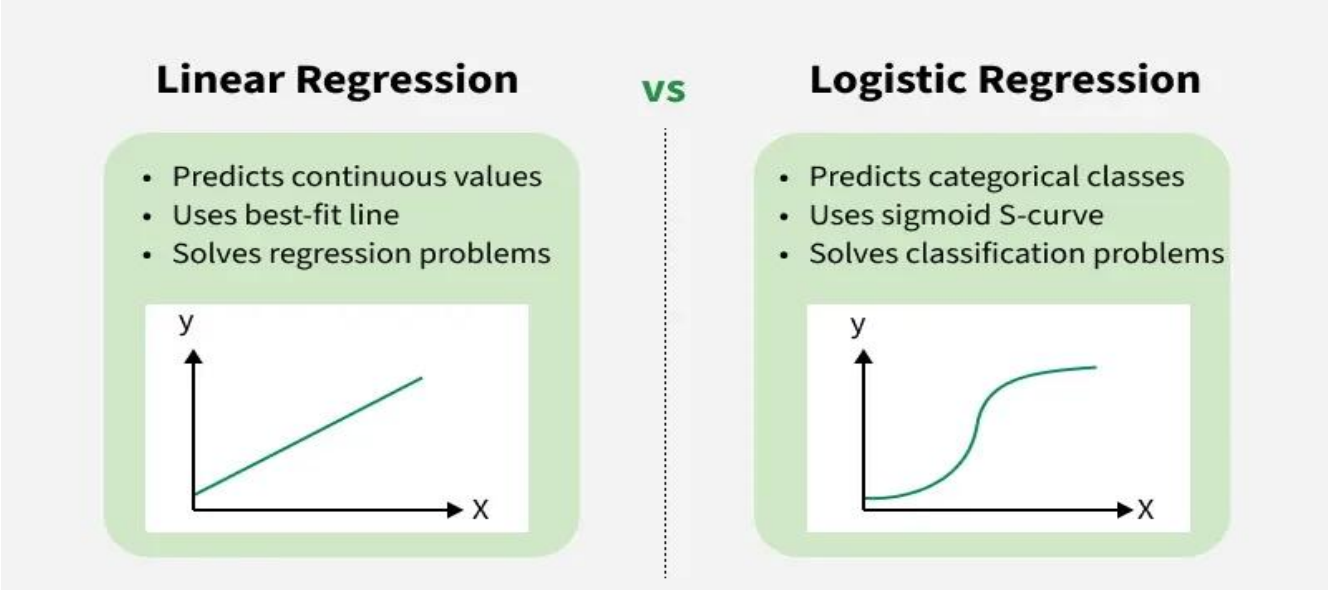
- **Sigmoid Kernel**: Mimics the behavior of neural networks by using a sigmoid function as the kernel.

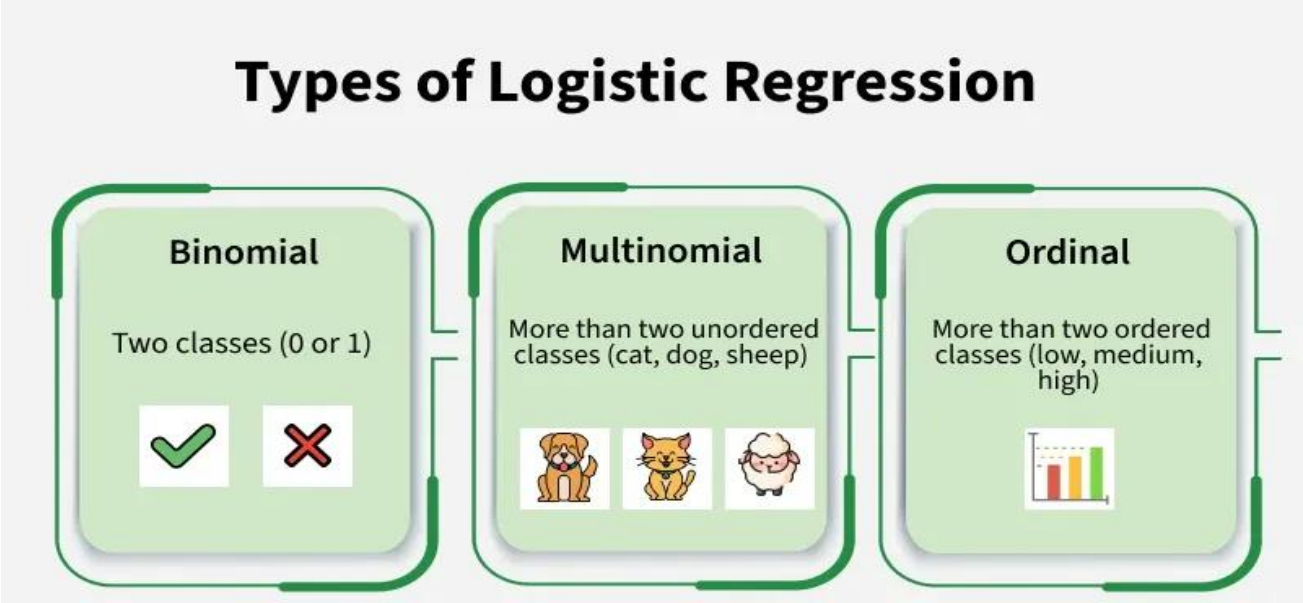
$$k(x_i, y_i) = \tanh(kx_i y_j - \sigma)$$

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.





Logistic regression model transforms the linear regression function continuous value output into categorical value output using a sigmoid function which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.

Suppose we have input features represented as a matrix:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e 0 or 1.

$$Y = \begin{cases} 0 & \text{if } Class\ 1 \\ 1 & \text{if } Class\ 2 \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = (\sum_{i=1}^{n} w_i x_i) + b$$

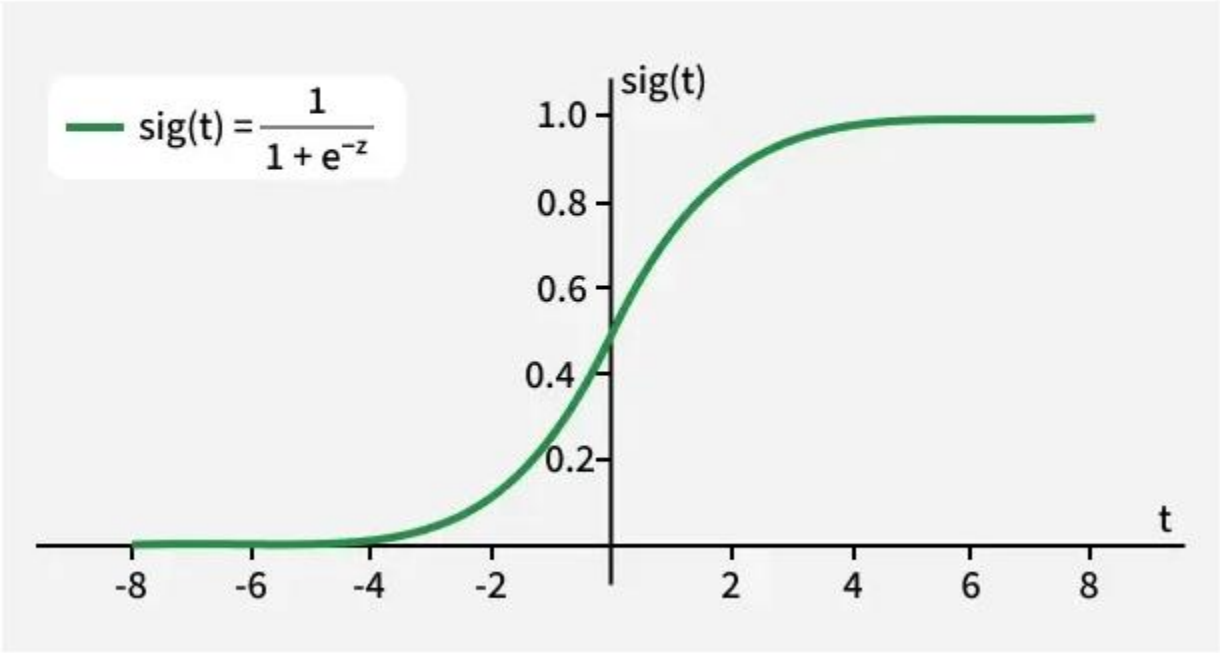
Here x_i is the ith observation of X, $w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient and $b$is the bias term also known as intercept. Simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

At this stage, *z* is a continuous value from the linear regression. Logistic regression then applies the sigmoid function to *z* to convert it into a probability between 0 and 1 which can be used to predict the class.

Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

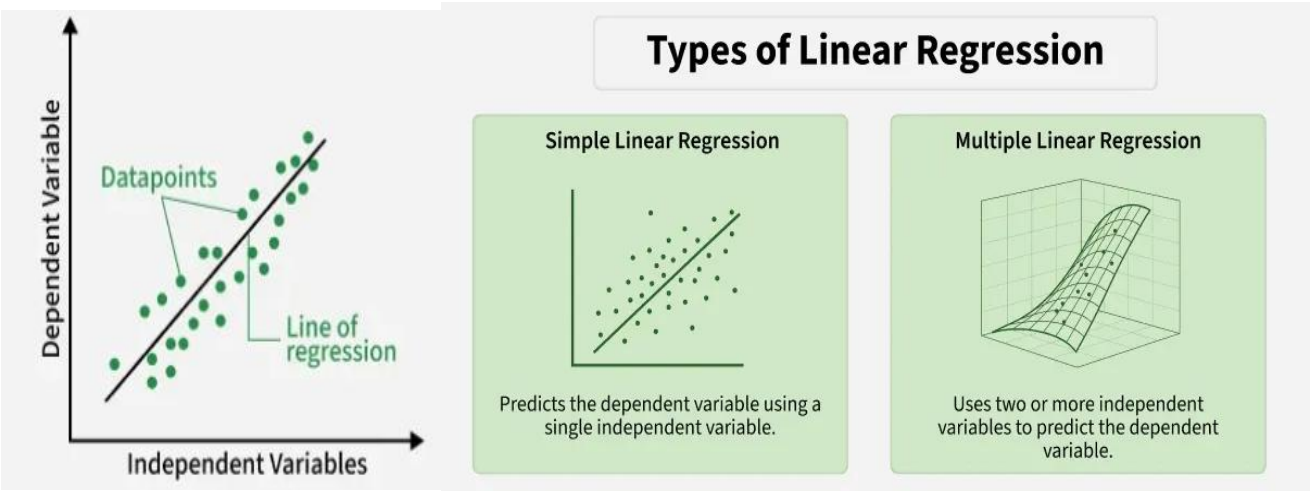


Sigmoid function

Linear Regression:

Linear Regression is a fundamental supervised learning algorithm used to model the relationship between a dependent variable and one or more independent variables. It predicts continuous values by fitting a straight line that best represents the data.

- It assumes that there is a linear relationship between the input and output
- Uses a best-fit line to make predictions
- Commonly used in forecasting, trend analysis, and predictive modelling

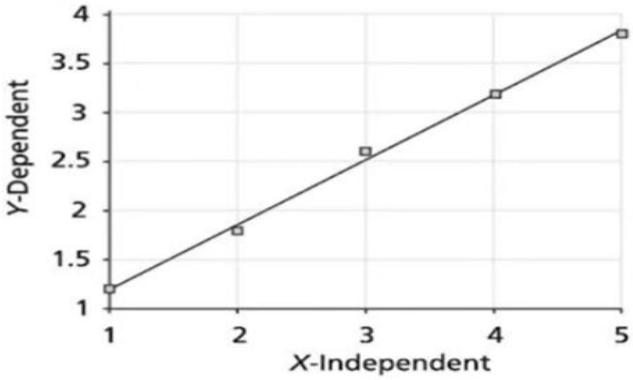


Example:

• Let us consider an example where the five weeks' sales data (in Thousands) is given as shown in Table.

• Apply linear regression technique to predict the 7th and 12th week sales.

x_i (Week)	y_j (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8



Here is the **text from the image**:

- Linear regression equation is given by

$$y = a_0 + a_1 * x + e$$

- where

$$a_1 = \frac{\overline{xy} - (\bar{x})(\bar{y})}{\overline{x^2} - (\bar{x})^2}$$

$$a_0 = \bar{y} - a_1 * \bar{x}$$

- Here, there are 5 items, i.e., i = 1, 2, 3, 4, 5.

	x_i (Week)	y_j (Sales In Thousands)	x_i^2	$x_i * y_j$
	1	1.2	1	1.2
	2	1.8	4	3.6
	3	2.6	9	7.8
	4	3.2	16	12.8
	5	3.8	25	19
Sum	15	12.6	55	44.4
Average	$\bar{x} = 3$	$\bar{y} = 2.52$	$\overline{x^2} = 11$	$\overline{xy} = 8.88$

- $\bar{x} = 3 \quad \bar{y} = 2.52 \quad \overline{x^2} = 11 \quad \overline{xy} = 8.88$
- $a_1 = \frac{(\overline{xy}) - (\bar{x})(\bar{y})}{\overline{x^2} - \bar{x}^2} = \frac{8.88 - 3 \times 2.52}{11 - 3^2} = 0.66$
- $a_0 = \bar{y} - a_1 \times \bar{x} = 2.52 - 0.66 \times 3 = 0.54$
- **Regression equation is**
- $y = a_0 + a_1 \times x$
- $y = 0.54 + 0.66 \times x$

- **Regression equation is**
- $y = a_0 + a_1 \times x$
- $y = 0.54 + 0.66 \times x$
- The predicted 7th week sale (when $x = 7$) is,
- $y = 0.54 + 0.66 \times 7 = 5.16$
- The predicted 12th week sale (when $x = 12$) is,
- $y = 0.54 + 0.66 \times 12 = 8.46$

Linear Regression using Least Squares Method

Linear Regression is a statistical technique used to model the relationship between a **dependent variable y** and an **independent variable x** by fitting the **best straight line** to the data.

The equation of the regression line is:

$$\mathbf{y = a + bx}$$

Where

- y = predicted value
- x = independent variable
- a = intercept
- b = slope of the line

- Here, **X is independent variable** and **y is dependent variable**.
- Equation of linear regression looks like:

$$y = a + bx$$

- **a** is the intercept and **b** is the slope or coefficient.

$$b = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

$$\bar{y} = a + b\bar{x}$$

Algorithm / Steps

1. Calculate the mean of x values $(\bar{x})$.
2. Calculate the mean of y values $(\bar{y})$.
3. Compute $x - \bar{x}$ and $y - \bar{y}$.
4. Calculate $(x - \bar{x})(y - \bar{y})$.
5. Calculate $(x - \bar{x})^2$.
6. Find the sums.
7. Compute slope b.
8. Compute intercept a.
9. Substitute into regression equation $y = a + bx$.

Example Given data:

x	y
1	2
2	3
3	5
4	4

Step 1: Compute Means

$$\bar{x} = \frac{1 + 2 + 3 + 4}{4} = 2.5$$

$$\bar{y} = \frac{2 + 3 + 5 + 4}{4} = 3.5$$

Step 2: Calculation Table

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
1	2	-1.5	-1.5	2.25	2.25
2	3	-0.5	-0.5	0.25	0.25
3	5	0.5	1.5	0.75	0.25
4	4	1.5	0.5	0.75	2.25

Now compute totals:

$$\sum (x - \bar{x})(y - \bar{y}) = 4$$

$$\sum (x - \bar{x})^2 = 5$$

Step 3: Calculate Slope

$$b = \frac{4}{5}$$

$$b = 0.8$$

Step 4: Calculate Intercept

$$a = \bar{y} - b\bar{x}$$

$$a = 3.5 - (0.8)(2.5)$$

$$a = 3.5 - 2$$

$$a = 1.5$$

Final Regression Equation

$$y = a + bx$$

$$y = 1.5 + 0.8x$$

This is the **best fit regression line** for the given dataset.

Applications of Linear Regression

1. Predicting future values
2. Trend analysis
3. Machine learning prediction models
4. Economics and business forecasting
5. Data analysis and statistics

Multi-Layer Perceptron's (MLPs):

A Multilayer Perceptron (MLP) is an artificial neural network that consists of at least three layers: an input layer, one or more hidden layers, and an output layer.

Components of an MLP

Input Layer: Receives the input data. Each node in this layer corresponds to a feature in the input dataset.

Hidden Layer(s): One or more layers where computations and learning occur. Each neuron in a hidden layer applies an activation function to the weighted sum of inputs.

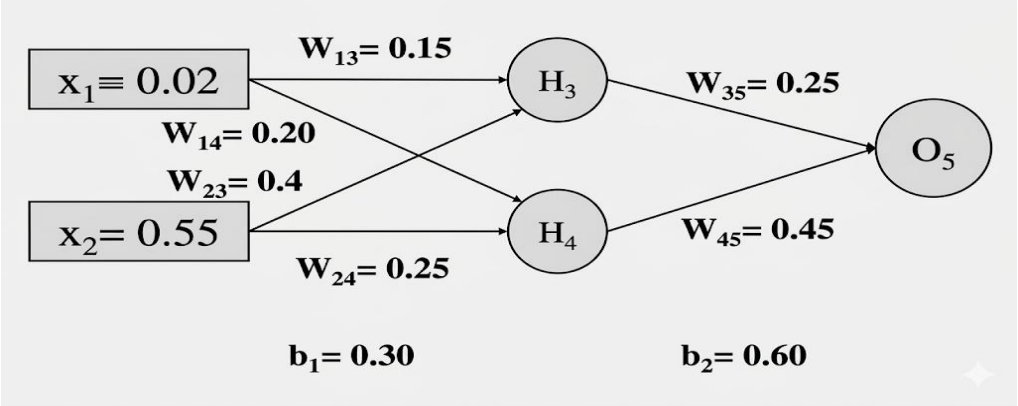
Output Layer: Produces the final result or prediction.

Structure:

Feedforward: The data flows in one direction from the input layer through the hidden layers to the output layer. There is no cycle or loop in the network.

Fully Connected: Each neuron in one layer is connected to every neuron in the next layer.

Given Data



Inputs:

$$x_1 = 0.02, \quad x_2 = 0.55$$

Weights:

$$W_{13} = 0.15,\ W_{23} = 0.40,\ W_{14} = 0.20,\ W_{24} = 0.25$$
$$W_{35} = 0.25,\ W_{45} = 0.45$$

Bias:

$$b_1 = 0.30, \quad b_2 = 0.60$$

Activation Function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

2. Hidden Layer Calculation

Neuron H3

Step 1: Weighted Sum

$$z_3 = (x_1 \cdot W_{13}) + (x_2 \cdot W_{23}) + b_1$$

$$z_3 = (0.02 \cdot 0.15) + (0.55 \cdot 0.40) + 0.30$$

$$z_3 = 0.003 + 0.22 + 0.30 = 0.523$$

Step 2: Activation

$$H_3 = \sigma(0.523) = \frac{1}{1 + e^{-0.523}} \approx 0.628$$

Neuron H4

Step 1: Weighted Sum

$$z_4 = (x_1 \cdot W_{14}) + (x_2 \cdot W_{24}) + b_1$$

$$z_4 = (0.02 \cdot 0.20) + (0.55 \cdot 0.25) + 0.30$$

$$z_4 = 0.004 + 0.1375 + 0.30 = 0.4415$$

Step 2: Activation

$$H_4 = \sigma(0.4415) = \frac{1}{1 + e^{-0.4415}} \approx 0.609$$

3. Output Layer Calculation

Step 1: Weighted Sum

$$z_5 = (H_3 \cdot W_{35}) + (H_4 \cdot W_{45}) + b_2$$

$$z_5 = (0.628 \cdot 0.25) + (0.609 \cdot 0.45) + 0.60$$

$$z_5 = 0.157 + 0.27405 + 0.60 = 1.03105$$

Step 2: Final Activation

$$y = \sigma(1.03105) = \frac{1}{1 + e^{-1.03105}} \approx 0.737$$

The MLP performs forward propagation by computing weighted sums, applying activation functions, and passing activated outputs to the next layer to obtain the final prediction.

Backpropagation for Training an MLP:

Backpropagation enables the model to reduce loss across epochs and effectively learn complex patterns from data.

Working of Back Propagation Algorithm

The Back Propagation algorithm involves two main steps: the Forward Pass and the Backward Pass.

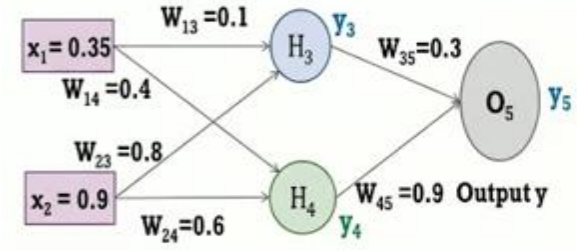
1. Forward Pass Work

In forward pass the input data is fed into the input layer. These inputs combined with their respective weights are passed to hidden layers. For example, in a network with two hidden layers (h1 and h2) the output from h1 serves as the input to h2. Before applying an activation function, a bias is added to the weighted inputs.

2. Backward Pass

In the backward pass the error (the difference between the predicted and actual output) is propagated back through the network to adjust the weights and biases. One common method for error calculation is the Mean Squared Error (MSE) given by:

$$\text{MSE} = (\text{Predicted Output} - \text{Actual Output})^2$$



- Forward Pass: Compute output for y3, y4 and y5.

$$a_j = \sum_j (w_{i,j} * x_i) \qquad y_j = F(a_j) = \frac{1}{1 + e^{-a_j}}$$

$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$
$= (0.1 * 0.35) + (0.8 * 0.9) = 0.755$
$y_3 = f(a_1) = 1/ (1 + e^{-0.755}) = 0.68$

$a_2 = (w_{14} * x_1) + (w_{24} * x_2)$
$= (0.4 * 0.35) + (0.6 * 0.9) = 0.68$
$y_4 = f(a_2) = 1/ (1 + e^{-0.68}) = 0.6637$

$a_3 = (w_{35} * y_3) + (w_{45} * y_4)$
$= (0.3 * 0.68) + (0.9 * 0.6637) = 0.801$
$y_5 = f(a_3) = 1/ (1 + e^{-0.801}) = 0.69$ (Network Output)

$$\text{Error} = y_{\text{target}} - y_5 = -0.19$$

$$0.5 - 0.69$$

Backward Pass: Compute δ_3, δ_4 and δ_5

In backpropagation, we calculate the error term (δ) for each neuron to determine how much each weight contributed to the final error.

1. For Output Unit (O_5):

The formula used is $\delta_j = o_j(1 - o_j)(t_j - o_j)$, where t_j is the target and o_j is the output.

- **Calculation:**

$$\delta_5 = y_5(1 - y_5)(y_{\text{target}} - y_5)$$

$$\delta_5 = 0.69 \cdot (1 - 0.69) \cdot (0.5 - 0.69) = \mathbf{-0.0406}$$

2. For Hidden Units (H_3 and H_4):

The formula used is $\delta_j = o_j(1 - o_j) \sum \delta_k w_{kj}$, which propagates the error from the output layer back to the hidden layer.

- **For Hidden Unit H_3:**

$$\delta_3 = y_3(1 - y_3) \cdot w_{35} \cdot \delta_5$$

$$\delta_3 = 0.68 \cdot (1 - 0.68) \cdot (0.3 \cdot -0.0406) = \mathbf{-0.00265}$$

- **For Hidden Unit H_4:**

$$\delta_4 = y_4(1 - y_4) \cdot w_{45} \cdot \delta_5$$

$$\delta_4 = 0.6637 \cdot (1 - 0.6637) \cdot (0.9 \cdot -0.0406) = \mathbf{-0.0082}$$

- **Each weight changed by:**

$$\Delta w_{ji} = \eta \, \delta_j \, o_i$$

$$\delta_j = o_j(1 - o_j)(t_j - o_j) \quad \text{if } j \text{ is an output unit}$$

$$\delta_j = o_j(1 - o_j) \sum_k \delta_k w_{kj} \quad \text{if } j \text{ is a hidden unit}$$

- Where:
  - η is a constant called the **learning rate**
  - t_j is the correct teacher output for unit j
  - δ_j is the **error measure** for unit j

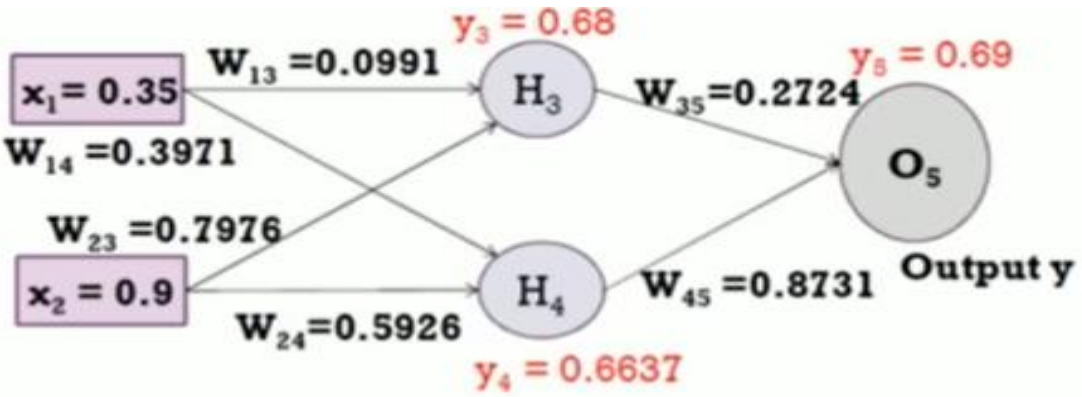
Compute New Weights

$$\Delta w_{ji} = \eta \delta_j o_i$$

$\Delta w_{45} = \eta\ \delta_5\ y_4 = 1 * -0.0406 * 0.6637 = -0.0269$
$\Delta w_{24} = \eta\ \delta_4\ x_2 = 1 * -0.0082 * 0.9 = -0.00738$
$w_{24}\ (new) = \Delta w_{24} + w_{24}(old) = -0.00738 + 0.6 = 0.5926$
$\Delta w_{35} = \eta\ \delta_5\ y_3 = 1 * -0.0406 * 0.68 = -0.02761$
$w_{35}\ (new) = \Delta w_{35} + w_{35}(old) = -0.02761 + 0.3 = 0.2724$

- Similarly, update all other weights

i	j	w_{ij}	δ_j	x_i	η	Updated w_{ij}
1	3	0.1	-0.00265	0.35	1	0.0991
2	3	0.8	-0.00265	0.9	1	0.7976
1	4	0.4	-0.0082	0.35	1	0.3971
2	4	0.6	-0.0082	0.9	1	0.5926
3	5	0.3	-0.0406	0.68	1	0.2724
4	5	0.9	-0.0406	0.6637	1	0.8731



- Forward Pass: Compute output for y3, y4 and y5.

$$a_j = \sum_j (w_{i,j} * x_i) \qquad yj = F(aj) = \frac{1}{1 + e^{-aj}}$$

$$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$$
$$= (0.0991 * 0.35) + (0.7976 * 0.9) = 0.7525$$
$$y_3 = f(a_1) = 1/ (1 + e^{-0.7525}) = 0.6797$$

$$a_2 = (w_{14} * x_1) + (w_{24} * x_2)$$
$$= (0.3971 * 0.35) + (0.5926 * 0.9) = 0.6723$$
$$y_4 = f(a_2) = 1/ (1 + e^{-0.6723}) = 0.6620$$

$$a_3 = (w_{35} * y_3) + (w_{45} * y_4)$$
$$= (0.2724 * 0.6797) + (0.8731 * 0.6620) = 0.7631$$
$$y_5 = f(a_3) = 1/ (1 + e^{-0.7631}) = 0.6820 \text{ (Network Output)}$$

$$\text{Error} = y_{\text{target}} - y_5 = -0.182$$

Continue the above process until we get error rate 0.