

Module-3

Chapter – 01 - Similarity-based Learning

Nearest-Neighbor Learning

k-Nearest-Neighbors (k-NN) Learning

Definition:

- **k-NN** is a non-parametric, similarity-based algorithm used for both **classification** and **regression**.
- It predicts the class or value of a test instance based on the '**K**' nearest neighbors in the training data.

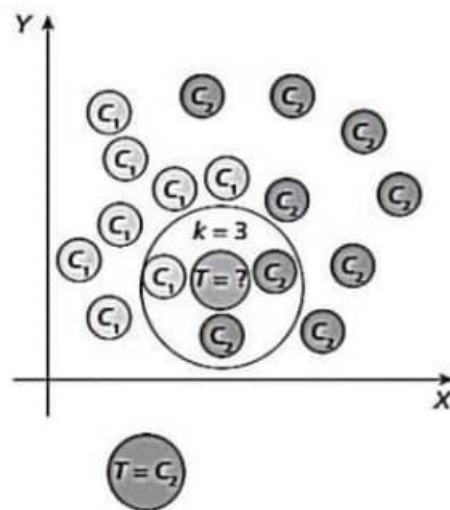


Figure 4.1: Visual Representation of k-Nearest Neighbor Learning

Working:

- **Classification:**
 - The algorithm determines the class of a test instance by considering the '**K**' nearest neighbors and selecting the class with the majority vote.

- **Regression:**
 - The output is the **mean** of the target variable values of the 'K' nearest neighbors.

Assumption:

- k-NN relies on the assumption that **similar objects are closer to each other** in the feature space.

Instance-Based Learning:

- **Memory-Based:** The algorithm does not build a prediction model ahead of time, but stores training data for predictions to be made at the time of the test instance.
- **Lazy Learning:** No model is constructed during training; the learning process happens only during testing when predictions are required.

Distance Metric:

- The most common distance metric used is **Euclidean distance** to measure the closeness of training data instances to the test instance.

Formula for Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

where x_i and y_i are the feature values of the two instances.

Choosing 'K':

- The value of 'K' determines how many neighbors should be considered for the prediction. It is typically selected by experimenting with different values of K to find the optimal one that produces the most accurate predictions.

Classification Process:

- **For a discrete target variable (classification):** The class of the test instance is determined by the **majority vote** of the 'K' nearest neighbors.
- **For a continuous target variable (regression):** The output is the **mean** of the output variable values of the 'K' nearest neighbors.

Advantages:

- **Simple and intuitive.**
- Effective for small to medium-sized datasets.
- Can handle multi-class classification.

Disadvantages:

- Computationally expensive during prediction because it requires calculating distances to all training data instances.
- Performance may degrade with high-dimensional data (curse of dimensionality).

Weighted K-Nearest-Neighbor Algorithm

Overview:

- Weighted k-NN is an **extension of the k-NN algorithm**.
- It improves upon k-NN by assigning **weights** to neighbors based on their **distance** from the test instance.

Motivation:

- Traditional k-NN assigns equal importance to all the 'k' nearest neighbors, which can lead to poor performance when:
 - Neighbors are at varying distances.
 - The nearest instances are more relevant than the farther ones.

- Weighted k-NN addresses this by making closer neighbors more influential.

Working Principle:

Weights are inversely proportional to distance:

- Closer neighbors get **higher weights**, while farther neighbors get **lower weights**.

Weighting Formula (Inverse Distance Weighting):

$$w_i = \frac{1}{d_i}$$

where w_i is the weight and d_i is the distance of the i^{th} neighbor.

- The final prediction is based on the **weighted majority vote (classification)** or the **weighted average (regression)** of the k nearest neighbors.

Weight Assignment:

- **Uniform Weighting:** All neighbors are given the same weight (as in standard k-NN).
- **Distance-Based Weighting:** Weights are computed based on the inverse distance, giving closer neighbors more influence.

Advantages:

- Addresses the limitations of standard k-NN by considering the **relative importance** of neighbors.
- Performs better in datasets where closer neighbors are more relevant to the prediction.

Applications:

- Classification: Predict the class of the test instance by **weighted voting** of the k nearest neighbors.

- Regression: Predict the output value by computing the **weighted mean** of the k nearest neighbors.

Limitations:

- Computational cost increases as distance calculations and weight assignments are performed for each query.
- Sensitive to the choice of the **distance metric** (e.g., Euclidean, Manhattan, etc.).

Nearest Centroid Classifier

A simple alternative to k-NN classifiers for similarity-based classification is the Nearest Centroid Classifier.

It is a simple classifier and also called as Mean Difference classifier.

The idea of this classifier is to classify a test instance to the class whose centroid/mean is closest to that instance.

The **Nearest Centroid Classifier** (also known as the **Mean Difference Classifier**) is a simple alternative to k-Nearest Neighbors (k-NN) for similarity-based classification.

Algorithm

Inputs: Training dataset T, Distance metric d, Test instance t

Output: Predicted class or category

1. Compute the mean/centroid of each class.
2. Compute the distance between the test instance and mean/centroid of each class (Euclidean Distance).
3. Predict the class by choosing the class with the smaller distance.

Example 4.3: Consider the sample data shown in Table 4.9 with two features x and y . The target classes are 'A' or 'B'. Predict the class using Nearest Centroid Classifier.

Table 4.9: Sample Data

X	Y	Class
3	1	A
5	2	A
4	3	A
7	6	B
6	7	B
8	5	B

Solution:

Step 1: Compute the mean/centroid of each class. In this example there are two classes called 'A' and 'B'.

Centroid of class 'A' = $(3 + 5 + 4, 1 + 2 + 3)/3 = (12, 6)/3 = (4, 2)$

Centroid of class 'B' = $(7 + 6 + 8, 6 + 7 + 5)/3 = (21, 18)/3 = (7, 6)$

Now given a test instance $(6, 5)$, we can predict the class.

Step 2: Calculate the Euclidean distance between test instance $(6, 5)$ and each of the centroid.

$$\text{Euc_Dist}[(6, 5); (4, 2)] = \sqrt{(6-4)^2 + (5-2)^2} = \sqrt{13} = 3.6$$

$$\text{Euc_Dist}[(6, 5); (7, 6)] = \sqrt{(6-7)^2 + (5-6)^2} = \sqrt{2} = 1.414$$

The test instance has smaller distance to class B. Hence, the class of this test instance is predicted as 'B'.

Locally Weighted Regression (LWR)

- Locally Weighted Regression (LWR) is a non-parametric supervised learning algorithm that performs local regression by combining regression model with nearest neighbor's model.
- LWR is also referred to as a memory-based method as it requires training data while prediction but uses only the training data instances locally around the point of interest.

- ✚ Using nearest neighbors algorithm, we find the instances that are closest to a test instance and fit linear function to each of those 'K' nearest instances in the local regression model.
- ✚ The key idea is that we need to approximate the linear functions of all 'K' neighbors that minimize the error such that the prediction line is no more linear but rather it is a curve.

Ordinary linear regression finds out a linear relationship between the input x and the output y .

Given training dataset T ,

Hypothesis function $h_{\beta}(x)$, the predicted target output is a linear function where β_0 is the intercept and β_1 is the coefficient of x .

It is given in Eq. (4.1) as,

$$h_{\beta}(x) = \beta_0 + \beta_1 x \quad (4.1)$$

The cost function is such that it minimizes the error difference between the predicted value $h_{\beta}(x)$ and true value ' y ' and it is given as in Eq. (4.2).

$$J(\beta) = \frac{1}{2} \sum_{i=1}^m (h_{\beta}(x_i) - y_i)^2 \quad (4.2)$$

where ' m ' is the number of instances in the training dataset.

Now the cost function is modified for locally weighted linear regression including the weights only for the nearest neighbor points. Hence, the cost function is given as in Eq. (4.3).

$$J(\beta) = \frac{1}{2} \sum_{i=1}^m w_i (h_{\beta}(x_i) - y_i)^2 \quad (4.3)$$

where w_i is the weight associated with each x_i .

The weight function used is a Gaussian kernel that gives a higher value for instances that are close to the test instance, and for instances far away, it tends to zero but never equals to zero.

w_i is computed in Eq. (4.4) as,

$$w_i = e^{-\frac{(x_i - x)^2}{2\tau^2}} \quad (4.4)$$

where, τ is called the bandwidth parameter and controls the rate at which w_i reduces to zero with distance from x_r

Example 4.4: Consider a simple example with four instances shown in Table 4.10 and apply locally weighted regression.

Table 4.10: Sample Table

S.No.	Salary (in lakhs)	Expenditure (in thousands)
1.	5	25
2.	1	5
3.	2	7
4.	1	8

Solution: Using linear regression model assuming we have computed the parameters:

$$\beta_0 = 4.72, \beta_1 = 0.62$$

Given a test instance with $x = 2$, the predicted y' is:

$$y' = \beta_0 + \beta_1 x = 4.72 + 0.62 \times 2 = 5.96$$

Applying the nearest neighbor model, we choose $k = 3$ closest instances.

Table 4.11 shows the Euclidean distance calculation for the training instances.

Table 4.11: Euclidean Distance Calculation

S.No.	$x = \text{Salary (in lakhs)}$	$y = \text{Expenditure (in thousands)}$	Euclidean Distance
1.	5	25	$\sqrt{(5-2)^2} = 3$
2.	1	5	$\sqrt{(1-2)^2} = 1$
3.	2	7	$\sqrt{(2-2)^2} = 0$
4.	1	8	$\sqrt{(1-2)^2} = 1$

Instances 2, 3 and 4 are closer with smaller distances.

The mean value = $(5 + 7 + 8)/3 = 20/3 = 6.67$.

Using Eq. (4.4) compute the weights for the closest instances, using the Gaussian kernel,

$$w_i = e^{\frac{-(x_i - x)^2}{2\tau^2}}$$

Hence the weights of the closest instances is computed as follows,

Weight of Instance 2 is:

$$w_2 = e^{\frac{-(x_2 - x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2 \times 0.4^2}} = e^{-3.125} = 0.043$$

Weight of Instance 3 is:

$$w_3 = e^{\frac{-(x_3 - x)^2}{2\tau^2}} = e^{\frac{-(2-2)^2}{2 \times 0.4^2}} = e^0 = 1 \quad [w_3 \text{ is closer hence gets a higher weight value}]$$

Weight of Instance 4 is:

$$w_4 = e^{\frac{-(x_4 - x)^2}{2\sigma^2}} = e^{\frac{-(1-2)^2}{2 \times 0.4^2}} = e^{-3.125} = 0.043$$

The predicted output for the three closer instances is given as follows:

The predicted output of Instance 2 is:

$$y_2' = h_{\beta}(x_2) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 1 = 5.34$$

The predicted output of Instance 3 is:

$$y_3' = h_{\beta}(x_3) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 2 = 5.96$$

The predicted output of Instance 4 is:

$$y_4' = h_{\beta}(x_4) = \beta_0 + \beta_1 x_2 = 4.72 + 0.62 \times 1 = 5.34$$

The error value is calculated as:

$$J(\beta) = \frac{1}{2} \sum_{i=1}^n w_i (h_{\beta}(x_i) - y_i)^2 = \frac{1}{2} (0.043(5.34 - 5)^2 + 1(5.96 - 7)^2 + 0.043(5.34 - 8)^2) = 0.6953$$

Now, we need to adjust this cost function to minimize the error difference and get optimal β parameters.

Chapter – 02

Regression Analysis

Introduction to Regression

Definition:

Regression analysis is a **supervised learning technique** used to model the relationship between one or more independent variables (x) and a dependent variable (y).

Objective:

The goal is to predict or forecast the dependent variable (y) based on the independent variables (x), which are also called **explanatory, predictor, or independent variables**.

Mathematical Representation:

The relationship is represented by a function:

$$y = f(x)$$

where y is the dependent variable (target variable) and x is the independent variable (predictor).

Purpose:

Regression analysis helps to determine how the dependent variable changes when an independent variable is varied while others remain constant.

It answers key questions such as:

- What is the relationship between variables?
- What is the strength and nature (linear or non-linear) of the relationship?
- What is the relevance and contribution of each variable?

Applications:

- Sales forecasting
- Bond values in portfolio management
- Insurance premiums
- Agricultural yield predictions
- Real estate pricing

Prediction Focus:

Regression is primarily used for predicting **continuous or quantitative variables**, such as price, revenue, and other measurable factors.

Introduction to Linear Regression

Definition:

Linear Regression is a fundamental **supervised learning algorithm** used to model the relationship between one or more independent variables (**predictors**) and a dependent variable (**target**).

It assumes a **linear relationship** between the variables.

Objective:

The primary goal of linear regression is to find a linear equation that best fits the data points. This equation is used to predict the dependent variable based on the values of the independent variables.

Mathematical Representation:

The relationship is represented as:

Assumptions:

- **Linearity:** The relationship between x and y is linear.
- **Independence:** Observations are independent of each other.

- **Homoscedasticity:** Constant variance of errors across all levels of xxx.
- **Normality:** The residuals (errors) are normally distributed.

Types of Linear Regression:

- **Simple Linear Regression:** Involves one independent variable.
- **Multiple Linear Regression:** Involves two or more independent variables.

Applications:

- Predicting **house prices** based on features like size and location.
- Estimating **sales** based on advertising expenditure.
- Forecasting **stock prices** or other financial metrics.
- Modeling **growth trends** in industries.

Advantages:

- Easy to implement and interpret.
- Efficient for linearly separable data.

Limitations:

- Struggles with non-linear relationships.
- Sensitive to outliers, which can distort predictions.

Linear Regression in Matrix Form

Matrix notations can be used for representing the values of independent and dependent variables. This is illustrated through Example 5.2.

The Eq. (5.3) can be written in the form of matrix as follows:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \quad (5.11)$$

This can be written as:

$Y = Xa + e$, where X is an $n \times 2$ matrix, Y is an $n \times 1$ vector, a is a 2×1 column vector and e is an $n \times 1$ column vector.

Example 5.2: Find linear regression of the data of week and product sales (in Thousands) given in Table 5.3. Use linear regression in matrix form.

Table 5.3: Sample Data for Regression

x_i (Week)	y_i (Product Sales in Thousands)
1	1
2	3
3	4
4	8

Solution: Here, the dependent variable X is be given as:

$$x^T = [1 \ 2 \ 3 \ 4]$$

And the independent variable is given as follows:

$$y^T = [1 \ 3 \ 4 \ 8]$$

The data can be given in matrix form as follows:

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix}. \text{ The first column can be used for setting bias.}$$

$$\text{and } Y = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix}$$

The regression is given as:

$$a = ((X^T X)^{-1} X^T) Y$$

The computation order of this equation is shown step by step as:

1. Computation of $(X^T X) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$
2. Computation of matrix inverse of $(X^T X)^{-1} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$
3. Computation of $((X^T X)^{-1} X^T) = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix}$
4. Finally, $((X^T X)^{-1} X^T) Y = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix} = \begin{pmatrix} -1.5 \\ 2.2 \end{pmatrix} \begin{pmatrix} \text{Intercept} \\ \text{slope} \end{pmatrix}$

Thus, the substitution of values in Eq. (5.11) using the previous steps yields the fitted line as $2.2x - 1.5$.

Multiple Linear Regression

Multiple regression model involves multiple predictors or independent variables and one dependent variable.

This is an extension of the linear regression problem. The basic assumptions of multiple linear regression are that the independent variables are not highly correlated and hence multicollinearity problem does not exist.

Also, it is assumed that the residuals are normally distributed.

For example, the multiple regression of two variables x_1 and x_2 is given as follows:

$$\begin{aligned} y &= f(x_1, x_2) \\ &= a_0 + a_1 x_1 + a_2 x_2 \end{aligned} \quad (5.21)$$

In general, this is given for 'n' independent variables as:

$$\begin{aligned} y &= f(x_1, x_2, x_3, \dots, x_n) \\ &= a_0 + a_1 x_1 + a_2 x_2 + \dots + a_n x_n + \varepsilon \end{aligned} \quad (5.22)$$

Here, $(x_1, x_2, \dots, x_n)$ are predictor variables, y is the dependent variable, $(a_0, a_1, \dots, a_n)$ are the coefficients of the regression equation and ε is the error term. This is illustrated through Example 5.5.

Example 5.5: Apply multiple regression for the values given in Table 5.7 where weekly sales along with sales for products x_1 and x_2 are provided. Use matrix approach for finding multiple regression.

Table 5.7: Sample Data

x_1 (Product One Sales)	x_2 (Product Two Sales)	y Output Weekly Sales (in Thousands)
1	4	1
2	5	6
3	8	8
4	2	12

Solution: Here, the matrices for Y and X are given as follows:

$$X = \begin{pmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{pmatrix} \text{ and } Y = \begin{pmatrix} 1 \\ 6 \\ 8 \\ 12 \end{pmatrix}.$$

The coefficient of the multiple regression equation is given as $a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$.

The regression coefficient for multiple regression is calculated the same way as linear regression:

$$\hat{a} = ((X^T X)^{-1} X^T) Y \quad (5.23)$$

Using Eq. (5.23), and substituting the values (Similar to Problem 5.2), one gets $\hat{a}$ as:

$$\begin{aligned} &= \left(\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{pmatrix} \right)^{-1} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 5 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 6 \\ 8 \\ 12 \end{pmatrix} \\ &= \begin{pmatrix} -1.69 \\ 3.48 \\ -0.05 \end{pmatrix} \end{aligned}$$

Here, the coefficients are $a_0 = -1.69$, $a_1 = 3.48$ and $a_2 = -0.05$. Hence, the constructed model is:
 $y = -1.69 + 3.48x_1 - 0.05x_2$

Definition:

Multiple Linear Regression (MLR) is an extension of simple linear regression, where **multiple independent variables** (predictors) are used to model the relationship with a **single dependent variable** (target).

Mathematical Representation:

The relationship is represented as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

where:

- y : Dependent variable (target).
- $x_1, x_2, \dots, x_n$: Independent variables (predictors).
- β_0 : Intercept.
- $\beta_1, \beta_2, \dots, \beta_n$: Coefficients of each predictor.
- ϵ : Error term (residuals).

Assumptions of Multiple Linear Regression:

No Multicollinearity: The independent variables should not be highly correlated with each other. Multicollinearity can cause issues in estimating the coefficients accurately.

Normality of Residuals: The residuals (errors) should be **normally distributed** for valid inference and hypothesis testing.

- **Linearity:** The relationship between each independent variable and the dependent variable should be linear.
- **Independence of Errors:** Observations should be independent of each other.
- **Homoscedasticity:** The variance of residuals should be constant across all levels of the independent variables.

Applications:

- **Predicting house prices** based on multiple features (size, location, number of rooms, etc.).
- **Estimating the sales** of a product based on various factors (price, advertising budget, competition, etc.).
- **Modeling health outcomes** based on multiple risk factors (age, BMI, physical activity, etc.).

Advantages:

- Can model the relationship between multiple predictors and a single outcome.
- Provides insights into how different predictors influence the dependent variable.

Limitations:

- If **multicollinearity** exists (high correlation between predictors), it can affect the stability and interpretability of the model.
- Can be computationally complex with a large number of predictors.
- Sensitive to outliers, which can distort the relationship between variables.

Polynomial Regression

Introduction to Polynomial Regression

Definition:

Polynomial Regression is a form of regression analysis that models the relationship between the independent variable(s) and the dependent variable as a **polynomial function**.

It is used when the relationship between variables is **non-linear** and cannot be effectively modeled using linear regression.

Purpose:

When the data exhibits a non-linear trend, linear regression may result in **large errors**.

Polynomial regression overcomes this limitation by fitting a **curved line** to the data.

Approaches to Handle Non-Linearity:**Transformation:**

Non-linear data can be transformed into linear data, allowing it to be handled using linear regression.

Example: For an exponential function $y = ae^{bx}$, applying the natural log transformation gives:

$$\ln(y) = \ln(a) + bx$$

This converts the data into a linear form.

Polynomial Regression:

Instead of transforming data, a polynomial regression fits a higher-degree polynomial function directly:

$$y = \beta_0 + \beta_1x + \beta_2x^2 + \dots + \beta_nx^n + \epsilon$$

where n is the degree of the polynomial.

Features of Polynomial Regression:

- Captures **curved relationships** between variables.
- Provides a more flexible model compared to linear regression.

Applications:

- Modeling **growth trends** in populations or markets.
- Predicting **real-world phenomena** such as temperature variations, physics experiments, or chemical reactions.
- **Engineering designs** involving complex relationships.

Advantages:

- Capable of modeling non-linear relationships without transforming the data.
- Provides a better fit for datasets with curved trends.

Limitations:

- Increasing the polynomial degree can lead to **overfitting** the training data.
- Sensitive to **outliers**, which can significantly distort the fitted curve.
- May require careful tuning of the degree to balance bias and variance.

Logistic Regression

Introduction to Logistic Regression

Definition:

Logistic Regression is a **supervised learning algorithm** used for classification problems, particularly **binary classification**, where the output is a categorical variable with two possible outcomes (e.g., yes/no, pass/fail, spam/not spam).

Purpose:

Logistic Regression predicts the **probability of a categorical outcome** and maps the prediction to a value between 0 and 1. It works well when the dependent variable is binary.

Applications:

- Email classification: Is the email spam or not?
- Student admission prediction: Should a student be admitted or not based on scores?
- Exam result classification: Will the student pass or fail based on marks?

Core Concept:

- Logistic Regression models the probability of a particular response variable.

- For instance, if the predicted probability of an email being spam is 0.7, there is a 70% chance the email is spam.

Challenges with Linear Regression for Classification:

- Linear regression can predict values outside the range of 0 to 1, which is unsuitable for probabilities.
- Logistic Regression overcomes this by using a **sigmoid function** to map values to the range [0, 1].

Sigmoid Function:

The **sigmoid function** (also called the logit function) is used to map any real number to the range [0, 1]. It is mathematically represented as:

$$p(x) = \frac{1}{1 + e^{-x}}$$

Difference between Odds and Probability:

Aspect	Odds	Probability
Definition	The ratio of the probability of an event occurring to the probability of it not occurring.	The likelihood of an event occurring, expressed as a fraction between 0 and 1.
Formula	$\text{Odds} = \frac{P(\text{event})}{1 - P(\text{event})}$	$\text{Probability (P)} = \frac{\text{Favorable Outcomes}}{\text{Total Outcomes}}$
Range	Ranges from 0 to $+\infty$.	Ranges from 0 to 1.
Interpretation	Describes the ratio of success to failure.	Describes the proportion of success relative to the total number of possibilities.
Example (Event = Success)	If the probability is 0.75, the odds are: $\frac{0.75}{1 - 0.75} = 3$ (3:1 in favor of success).	If there's a 75% chance of success, the probability is 0.75.
Use in Logistic Regression	Used to model the relationship in the log-odds transformation.	Used as a direct measure of the likelihood of an event.
Additive or Multiplicative	Odds are multiplicative (e.g., doubling the odds means a multiplicative factor).	Probability is additive (e.g., increasing the probability by 0.1 means adding directly).

For example:

If the probability of an event is 0.75, the odds are:

$$\text{Odds} = \frac{0.75}{1 - 0.75} = 3$$

Features of Logistic Regression:

- Logistic Regression predicts the probability of a class label.
- It applies a threshold (e.g., 0.5) to determine the class label.
- It is based on the **log-odds** transformation to linearize the relationship between variables.

Advantages:

- Simple and efficient for binary classification.
- Works well when the relationship between the dependent and independent variables is linear (in terms of log-odds).
- Outputs interpretable probabilities.

Limitations:

- Struggles with **non-linear decision boundaries** (can be addressed with extensions like polynomial logistic regression).
- Sensitive to **outliers** in the dataset.

For Understanding Purpose Only

Regression Analysis – A mathematical technique of modelling the relationship between input and output variables.

Independent Variable – The feature variables (x) in function $y = f(x)$ are called independent variables. These are also known as explanatory, exploratory, covariate or domain points.

Dependent Variable – In the function, $y = f(x)$, y is called a dependent variable as it is dependent on x . Also known as labels or response variables.

Correlation – Correlation indicates the strength of the relationship. Positive correlation among variables x and y is that they increase together in tandem. Negative correlation means as one variable increases, the other one decreases.

Scatter Plot – A plot of explanatory variables and response variables.

Linearity – The relationship between variables can be visualized as a straight line.

Non-linearity – The relationship between variables is complex and cannot be fitted within a line.

Multicollinearity – If the explanatory variables are highly correlated, then regression is vulnerable to bias. This condition is called Multicollinearity.

Singularity – If the correlation among explanatory variables is perfect, it is called singularity.

Ordinary Least Square (OLS) – Optimization technique where the distance between the data points and the line is optimized.

Best Fit – The fitted line with the least sum of squared error is called best-fit.

Residual or Error – The difference between the actual and predicted variables is called a residual.

Mean Absolute Error (MAE) – The mean of residuals is called MAE.

Means Squared Error (MSE) – The sum of square of residuals is called MSE.

r^2 – This is called coefficient of determination. This is square of the correlation between the predicted and response variables.

Multiple Regression – The technique that finds the relationship between two or more independent variables and one dependent variable.

Polynomial Regression – This is a technique of modelling using N^{th} order polynomial.

Odds – The ratio of probability of an occurring event and a non-occurring event is called odds.

Logit – Logit is log of odds. Otherwise, it is the log of outcome of logistic regression.

Logistic Function – A logistic function is the one that takes a value and outputs either 0 or 1 by minimizing the error in the probability predicted by the model.

LASSO – LASSO stands for 'Least Absolute Shrinkage and Selection Operator'. This uses L_1 regularization.

Ridge Regression – A regression technique that uses L_2 regularization.

Elastic Net – A hybrid regression method of Ridge and LASSO regression.

Chapter – 03

Decision Tree Learning

Introduction to Decision Tree Learning Model

Overview:

- Decision tree learning is a popular supervised predictive model for classification tasks.
- It performs inductive inference, generalizing from observed examples.
- It can classify both categorical and continuous target variables.
- The model is often used for solving complex classification problems with high accuracy.

Structure of a Decision Tree:

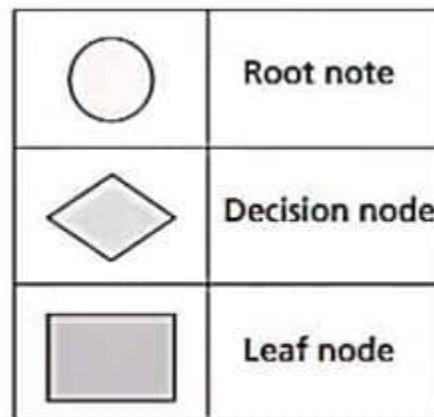


Figure 6.1: Nodes in a Decision Tree

- **Root Node:** The topmost node that represents the entire dataset.
- **Internal/Decision Nodes:** These are nodes that perform tests on input attributes and split the dataset based on test outcomes.
- **Branches:** Represent the outcomes of a test condition at a decision node.

- **Leaf Nodes/Terminal Nodes:** Represent the target labels or output of the decision process.
- **Path:** A path from root to leaf node represents a logical rule for classification.

Process of Building a Decision Tree:

Goal: Construct a decision tree from the given training dataset.

Tree Construction:

- Start from the root and recursively find the best attribute for splitting.
- This process continues until the tree reaches leaf nodes that cannot be further split.
- The tree represents all possible hypotheses about the data.

Output: A fully constructed decision tree that represents the learned model.

Inference or Classification:

Goal: For a given test instance, classify it into the correct target class.

Classification:

- Start at the root node and traverse the tree based on the test conditions for each attribute.
- Continue evaluating test conditions until reaching a leaf node, which provides the target class label for the instance.

Advantages of Decision Trees:

1. Easy to model and interpret.
2. Simple to understand.
3. Can handle both discrete and continuous predictor variables.
4. Can model non-linear relationships between variables.

5. Fast to train.

Disadvantages of Decision Trees:

1. It is difficult to determine how deep the tree should grow and when to stop.
2. Sensitive to errors and missing attribute values in training data.
3. Computational complexity in handling continuous attributes, requiring discretization.
4. Risk of overfitting with complex trees.
5. Not suitable for classifying multiple output classes.
6. Learning an optimal decision tree is an NP-complete problem.

Decision Tree Induction Algorithms

Several decision tree algorithms are widely used in classification tasks, including **ID3**, **C4.5**, and **CART**, among others.

These algorithms differ in their splitting criteria, handling of attributes, and robustness to data characteristics.

Popular Decision Tree Algorithms:

ID3 (Iterative Dichotomizer 3):

- Developed by **J.R. Quinlan** in 1986.
- Constructs univariate decision trees (splits based on a single attribute).
- Uses **Information Gain** as the splitting criterion.
- Assumes attributes are discrete or categorical.
- Works well with large datasets but is prone to **overfitting** on small datasets.
- Cannot handle missing values or continuous attributes directly (requires discretization).
- No pruning is performed, making it sensitive to **outliers**.

C4.5:

- An extension of ID3 developed by **J.R. Quinlan** in 1993.
- Uses **Gain Ratio** as the splitting criterion, which normalizes Information Gain.
- Can handle both **categorical** and **continuous attributes**.
- Handles missing values by estimating the best split based on available data.
- Prone to **outliers**, which can affect the tree construction.

CART (Classification and Regression Trees):

- Developed by **Breiman et al.** in 1984.
- Can handle **categorical** and **continuous-valued target variables**.
- Uses the **GINI Index** as the splitting criterion for classification tasks.
- Builds **binary decision trees** (only two splits per node).
- Handles missing values and is robust to **outliers**.
- Can be used for **regression tasks**, making it versatile.

Univariate vs. Multivariate Decision Trees:

Univariate Decision Trees:

- Split based on a single attribute at each decision node.
- Examples: **ID3** and **C4.5**.
- Simple and axis-aligned splits.

Multivariate Decision Trees:

- Consider multiple attributes for splitting at a single decision node.
- Example: **CART**.
- More complex and better suited for non-linear relationships.

Features of Decision Tree Algorithms

Algorithm	Splitting Criterion	Attributes Supported	Handles Missing Values?	Handles Outliers?
ID3	Information Gain	Discrete/Categorical only	No	No
C4.5	Gain Ratio	Both Categorical & Continuous	Yes	No (prone to outliers)
CART	GINI Index	Both Categorical & Continuous	Yes	Yes

Advantages and Limitations of ID3, C4.5, and CART:

Algorithm	Advantages	Limitations
ID3	Simple and works well with large datasets.	Overfitting on small datasets; cannot handle missing or continuous attributes.
C4.5	Handles continuous attributes and missing values; more robust than ID3.	Prone to outliers; computationally expensive for large datasets.
CART	Can handle outliers and missing values; suitable for both classification and regression.	Requires more computation for binary splits; can overfit without pruning.

Algorithm

Algorithm 6.2: Procedure to Construct a Decision Tree using ID3

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9) and Information_Gain Eq. (6.10) for each of the attribute in the training dataset.
3. Choose the attribute for which entropy is minimum and therefore the gain is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Definitions

Let T be the training dataset.

Let A be the set of attributes $A = \{A_1, A_2, A_3, \dots, A_n\}$.

Let m be the number of classes in the training dataset.

Let P_i be the probability that a data instance or a tuple ' d ' belongs to class C_i .

It is calculated as,

$$P_i = \frac{\text{Total no of data instances that belongs to class } C_i \text{ in } T}{\text{Total no of tuples in the training set } T} \quad (6.6)$$

Mathematically, it is represented as shown in Eq. (6.7).

$$P_i = \frac{|d_{C_i}|}{|T|} \quad (6.7)$$

Expected information or Entropy needed to classify a data instance d' in T is denoted as $\text{Entropy_Info}(T)$ given in Eq. (6.8).

$$\text{Entropy_Info}(T) = - \sum_{i=1}^m P_i \log_2 P_i \quad (6.8)$$

Entropy of every attribute denoted as $\text{Entropy_Info}(T, A)$ is shown in Eq. (6.9) as:

$$\text{Entropy_Info}(T, A) = \sum_{i=1}^v \frac{|A_i|}{|T|} \times \text{Entropy_Info}(A_i) \quad (6.9)$$

where, the attribute A has got ' v ' distinct values $\{a_1, a_2, \dots, a_v\}$, $|A_i|$ is the number of instances for distinct value ' i ' in attribute A , and $\text{Entropy_Info}(A_i)$ is the entropy for that set of instances.

Information_Gain is a metric that measures how much information is gained by branching on an attribute A . In other words, it measures the reduction in impurity in an arbitrary subset of data.

It is calculated as given in Eq. (6.10):

$$\text{Information_Gain}(A) = \text{Entropy_Info}(T) - \text{Entropy_Info}(T, A) \quad (6.10)$$

It can be noted that as entropy increases, information gain decreases. They are inversely proportional to each other.