

MODULE-3

Similarity-based Learning: Nearest-Neighbor Learning

Weighted K-Nearest-Neighbor Algorithm,

Nearest Centroid Classifier, Locally Weighted Regression (LWR).

Regression Analysis: Introduction to Regression, Introduction to Linear Regression

Multiple Linear Regression

Polynomial Regression, Logistic Regression

Decision Tree Learning: Introduction to Decision Tree Learning Model

Decision Tree Induction Algorithms.

Similarity-based Learning: Nearest-Neighbor Learning

Similarity based classifiers use similarity measures to locate the nearest neighbors and classify a test instance which works in contrast with other learning mechanisms such as decision trees / neural networks.

Similarity based learning is also called as instance based learning. It does not build an abstract model of the training instances and performs lazy learning when classifying a new instance. This learning mechanism simply stores all data and uses it only when it needs to classify an unseen instance.

The advantage of using this learning is that processing occurs only when a request to classify a new instance is given. This methodology is particularly useful when the whole data set is not available in the beginning but is collected in an incremental manner.

Drawback- Large memory to store the data since a global abstract model is not constructed initially with the training data.

Classification of instances is done based on the measure of similarity in the form of distance functions over data instances.

Several distance metrics are used to estimate the similarity /dissimilarity between instances required for clustering, nearest neighbor classification, anomaly detection and so on. Popular distance metrics used are Hamming distance, Euclidean distance, Manhattan distance, Cosine similarity, Pearson's correlation/ correlation similarity, Mean squared difference.

Generally , Similarity based classification problems formulate the features of test instance and training instance in Euclidean space to learn the similarity / dissimilarity between instances.

4.1.1 Differences Between Instance- and Model-based Learning

- An instance is an entity / an example in the training data set.
- It is described by a set of features / attributes.
- One attribute describes the class label/ category of an instance.
- Instance based methods learn / predict the class label/ a test only when a new instance is given for classification and until then it delays the processing of the training data set.

Table 4.1: Differences between Instance-based Learning and Model-based Learning

Instance-based Learning	Model-based Learning
Lazy Learners	Eager Learners
Processing of training instances is done only during testing phase	Processing of training instances is done during training phase
Instance-based Learning	Model-based Learning
No model is built with the training instances before it receives a test instance	Generalizes a model with the training instances before it receives a test instance
Predicts the class of the test instance directly from the training data	Predicts the class of the test instance from the model built
Slow in testing phase	Fast in testing phase
Learns by making many local approximations	Learns by creating global approximation

Instance based learning comes under the category of memory – based models which normally compare the given test instance with the trained instances that are stored in memory.

Memory based models classify a test instance by checking the similarity with the training instances.

Some examples of Instance-based learning algorithms are:

1. k -Nearest Neighbor (k -NN)
2. Variants of Nearest Neighbor learning
3. Locally Weighted Regression
4. Learning Vector Quantization (LVQ)
5. Self-Organizing Map (SOM)
6. Radial Basis Function (RBF) networks

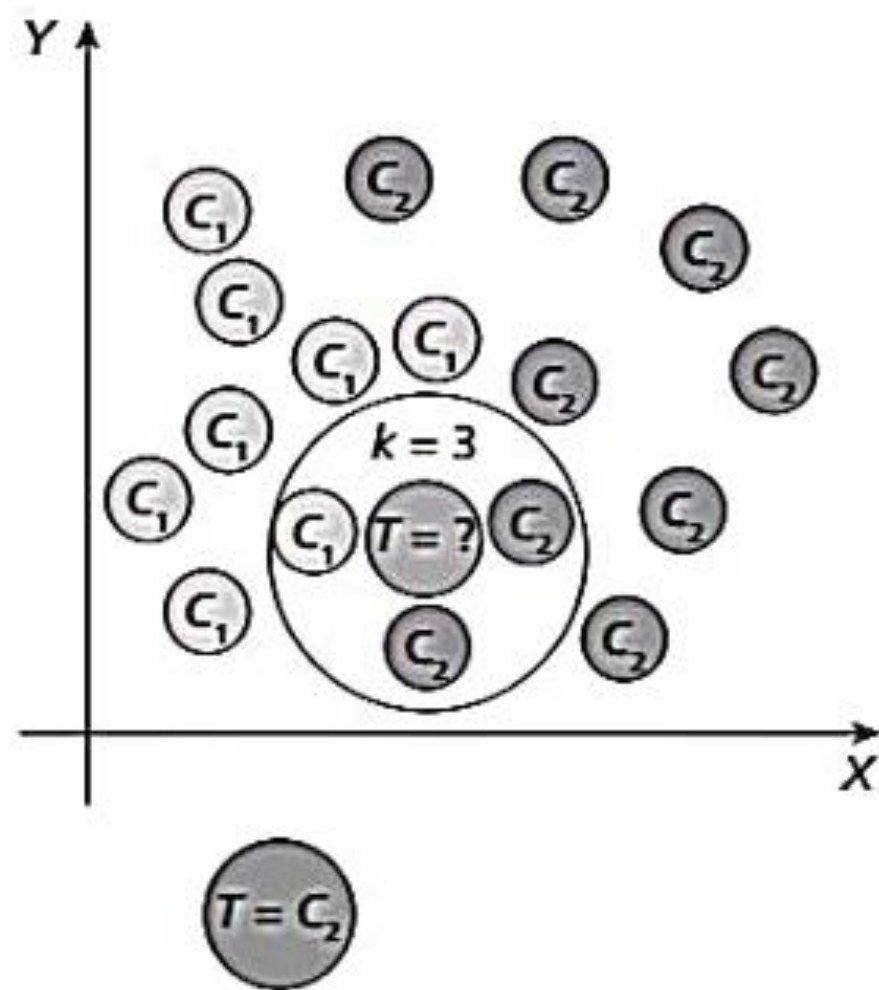


Figure 4.1: Visual Representation of k -Nearest Neighbor Learning

Algorithm 4.1: k-NN

Inputs: Training dataset T , distance metric d , Test instance t , the number of nearest neighbors k

Output: Predicted class or category

Prediction: For test instance t ,

1. For each instance i in T , compute the distance between the test instance t and every other instance i in the training dataset using a distance metric (Euclidean distance).

[Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as $\text{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]

[Categorical attributes (Binary) - Hamming Distance: If the value of the two instances is same, the distance d will be equal to 0 otherwise $d = 1$.]

2. Sort the distances in an ascending order and select the first k nearest training data instances to the test instance.
3. Predict the class of the test instance by majority voting (if target attribute is discrete valued) or mean (if target attribute is continuous valued) of the k selected nearest instances.

Example 4.1: Consider the student performance training dataset of 8 data instances shown in Table 4.2 which describes the performance of individual students in a course and their CGPA obtained in the previous semesters. The independent attributes are CGPA, Assessment and Project. The target variable is 'Result' which is a discrete valued variable that takes two values 'Pass' or 'Fail'. Based on the performance of a student, classify whether a student will pass or fail in that course.

Table 4.2: Training Dataset T

S.No.	CGPA	Assessment	Project Submitted	Result
1.	9.2	85	8	Pass
2.	8	80	7	Pass
3.	8.5	81	8	Pass

S.No.	CGPA	Assessment	Project Submitted	Result
4.	6	45	5	Fail
5.	6.5	50	4	Fail
6.	8.2	72	7	Pass
7.	5.8	38	5	Fail
8.	8.9	91	9	Pass

Solution: Given a test instance (6.1, 40, 5) and a set of categories {Pass, Fail} also called as classes, we need to use the training set to classify the test instance using Euclidean distance.

The task of classification is to assign a category or class to an arbitrary instance.

Assign $k = 3$.

Step 1: Calculate the Euclidean distance between the test instance (6.1, 40, and 5) and each of the training instances as shown in Table 4.3.

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2 - 6.1)^2 + (85 - 40)^2 + (8 - 5)^2}$ = 45.2063
2.	8	80	7	Pass	$\sqrt{(8 - 6.1)^2 + (80 - 40)^2 + (7 - 5)^2}$ = 40.09501
3.	8.5	81	8	Pass	$\sqrt{(8.5 - 6.1)^2 + (81 - 40)^2 + (8 - 5)^2}$ = 41.17961
4.	6	45	5	Fail	$\sqrt{(6 - 6.1)^2 + (45 - 40)^2 + (5 - 5)^2}$ = 5.001
5.	6.5	50	4	Fail	$\sqrt{(6.5 - 6.1)^2 + (50 - 40)^2 + (4 - 5)^2}$ = 10.05783
6.	8.2	72	7	Pass	$\sqrt{(8.2 - 6.1)^2 + (72 - 40)^2 + (7 - 5)^2}$ = 32.13114
7.	5.8	38	5	Fail	$\sqrt{(5.8 - 6.1)^2 + (38 - 40)^2 + (5 - 5)^2}$ = 2.022375
8.	8.9	91	9	Pass	$\sqrt{(8.9 - 6.1)^2 + (91 - 40)^2 + (9 - 5)^2}$ = 51.23319

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.4.

Table 4.4: Nearest Neighbors

Instance	Euclidean Distance	Class
4	5.001	Fail
5	10.05783	Fail
7	2.022375	Fail

Here, we take the 3 nearest neighbors as instances 4, 5 and 7 with smallest distances.

Step 3: Predict the class of the test instance by majority voting.

The class for the test instance is predicted as 'Fail'.

Data normalization/standardization is required when data (features) have different ranges or a wider range of possible values when computing distances and to transform all features to a specific range. This is probably done to eliminate the influence of one feature over another (i.e., to give all features equal chances). For example if one feature has values in the range of $[0-1]$ and another feature has values in the range of $[0-100]$, then the second feature will influence more even if there is a small variation than the first feature.

k -NN classifier performance is strictly affected by three factors such as the number of nearest neighbors (i.e., selection of k), distance metric and decision rule.

If the k value selected is small then it may result in overfitting or less stable and if it is big then it may include many irrelevant points from other classes. The choice of the distance metric selected also plays a major role and it depends on the type of the independent attributes in the training dataset.

The k -NN classification algorithm best suits lower dimensional data as in a high-dimensional space the nearest neighbors may not be very close at all.

4.3 WEIGHTED K-NEAREST-NEIGHBOR ALGORITHM

The Weighted k -NN is an extension of k -NN. It chooses the neighbors by using the weighted distance. The k -Nearest Neighbor (k -NN) algorithm has some serious limitations as its performance is solely dependent on choosing the k nearest neighbors, the distance metric used and the decision rule. However, the principle idea of Weighted k -NN is that k closest neighbors to the test instance are assigned a higher weight in the decision as compared to neighbors that are farther away from the test instance. The idea is that weights are inversely proportional to distances.

The selected k nearest neighbors can be assigned uniform weights, which means all the instances in each neighborhood are weighted equally or weights can be assigned by the inverse of their distance. In the second case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

Algorithm 4.2: Weighted k-NN

Inputs: Training dataset ' T ', Distance metric ' d ', Weighting function $w(i)$, Test instance ' t ', the number of nearest neighbors ' k '

Output: Predicted class or category

Prediction: For test instance t ,

1. For each instance ' i ' in Training dataset T , compute the distance between the test instance t and every other instance ' i ' using a distance metric (Euclidean distance).
[Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as $\text{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]
[Categorical attributes (Binary) - Hamming Distance: If the values of two instances are the same, the distance d will be equal to 0. Otherwise $d = 1$.]
2. Sort the distances in the ascending order and select the first ' k ' nearest training data instances to the test instance.
3. Predict the class of the test instance by weighted voting technique (Weighting function $w(i)$) for the k selected nearest instances:
 - Compute the inverse of each distance of the ' k ' selected nearest instances.
 - Find the sum of the inverses.
 - Compute the weight by dividing each inverse distance by the sum. (Each weight is a vote for its associated class).
 - Add the weights of the same class.
 - Predict the class by choosing the class with the maximum vote.

Example 4.2: Consider the same training dataset given in Table 4.1. Use Weighted k -NN and determine the class.

Solution:

Step 1: Given a test instance (7.6, 60, 8) and a set of classes {Pass, Fail}, use the training dataset to classify the test instance using Euclidean distance and weighting function.

Assign $k = 3$. The distance calculation is shown in Table 4.5.

Table 4.5: Euclidean Distance

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-7.6)^2 + (85-60)^2 + (8-8)^2}$ = 25.05115
2.	8	80	7	Pass	$\sqrt{(8-7.6)^2 + (80-60)^2 + (7-8)^2}$ = 20.02898
3.	8.5	81	8	Pass	$\sqrt{(8.5-7.6)^2 + (81-60)^2 + (8-8)^2}$ = 21.01928
4.	6	45	5	Fail	$\sqrt{(6-7.6)^2 + (45-60)^2 + (5-8)^2}$ = 15.38051
5.	6.5	50	4	Fail	$\sqrt{(6.5-7.6)^2 + (50-60)^2 + (4-8)^2}$ = 10.82636
6.	8.2	72	7	Pass	$\sqrt{(8.2-7.6)^2 + (72-60)^2 + (7-8)^2}$ = 12.05653
7.	5.8	38	5	Fail	$\sqrt{(5.8-7.6)^2 + (38-60)^2 + (5-8)^2}$ = 22.27644
8.	8.9	91	9	Pass	$\sqrt{(8.9-7.6)^2 + (91-60)^2 + (9-8)^2}$ = 31.04336

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.6.

Table 4.6: Nearest Neighbors

Instance	Euclidean Distance	Class
4	15.38051	Fail
5	10.82636	Fail
6	12.05653	Pass

Step 3: Predict the class of the test instance by weighted voting technique from the 3 selected nearest instances.

- Compute the inverse of each distance of the 3 selected nearest instances as shown in Table 4.7.

Table 4.7: Inverse Distance

Instance	Euclidean Distance	Inverse Distance	Class
4	15.38051	0.06502	Fail
5	10.82636	0.092370	Fail
6	12.05653	0.08294	Pass

- Find the sum of the inverses.

$$\text{Sum} = 0.06502 + 0.092370 + 0.08294 = 0.24033$$

- Compute the weight by dividing each inverse distance by the sum as shown in Table 4.8.

Table 4.8: Weight Calculation

Instance	Euclidean Distance	Inverse Distance	Weight = Inverse distance/Sum	Class
4	15.38051	0.06502	0.270545	Fail
5	10.82636	0.092370	0.384347	Fail
6	12.05653	0.08294	0.345109	Pass

- Add the weights of the same class.

$$\text{Fail} = 0.270545 + 0.384347 = 0.654892$$

$$\text{Pass} = 0.345109$$

- Predict the class by choosing the class with the maximum vote.

The class is predicted as 'Fail'.

4.4 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.

Algorithm 4.3: Nearest Centroid Classifier

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'.

4.5 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).

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

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_i

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 = Salary (in lakhs)	y = 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,

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

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

Weight of Instance 2 is:

Weight of Instance 3 is:

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

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

Weight of Instance 4 is:

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

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:

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

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

5.1 INTRODUCTION TO REGRESSION

Regression analysis is the premier method of supervised learning. This is one of the most popular and oldest supervised learning technique. Given a training dataset D containing N training points (x_i, y_i) , where $i = 1 \dots N$, regression analysis is used to model the relationship between one or more independent variables x_i and a dependent variable y_i . The relationship between the dependent and independent variables can be represented as a function as follows:

$$y = f(x) \tag{5.1}$$

Here, the feature variable x is also known as an explanatory variable, exploratory variable, a predictor variable, an independent variable, a covariate, or a domain point. y is a dependent variable. Dependent variables are also called as labels, target variables, or response variables.

Regression analysis determines the change in response variables when one exploration variable is varied while keeping all other parameters constant. This is used to determine the relationship each of the exploratory variables exhibits. Thus, regression analysis is used for prediction and forecasting.

Regression is used to predict continuous variables or quantitative variables such as price and revenue. Thus, the primary concern of regression analysis is to find answer to questions such as:

1. What is the relationship between the variables?
2. What is the strength of the relationships?
3. What is the nature of the relationship such as linear or non-linear?
4. What is the relevance of the attributes?
5. What is the contribution of each attribute?

There are many applications of regression analysis. Some of the applications of regressions include predicting:

1. Sales of a goods or services
2. Value of bonds in portfolio management
3. Premium on insurance companies
4. Yield of crops in agriculture
5. Prices of real estate

INTRODUCTION TO LINEARITY, CORRELATION, AND CAUSATION

The quality of the regression analysis is determined by the factors such as correlation and causation.

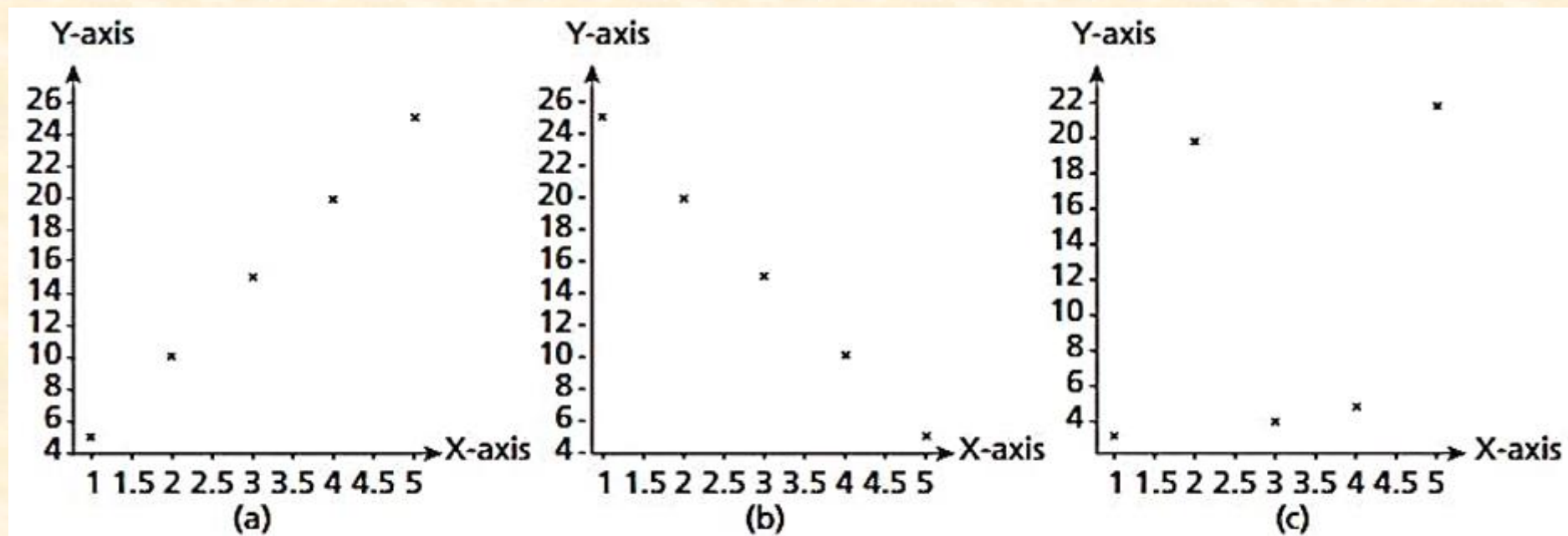


Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation
(c) Random Points with No Correlation

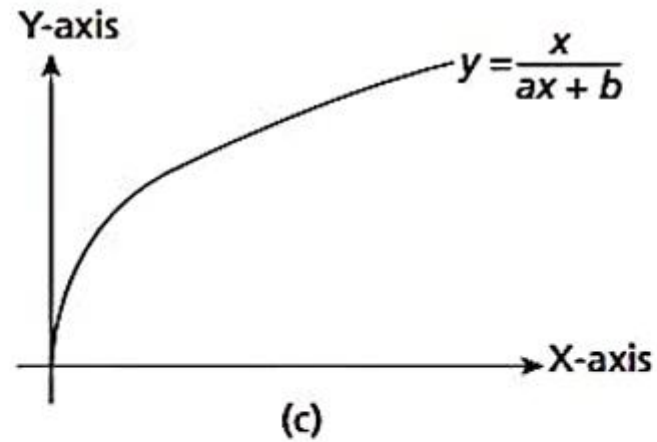
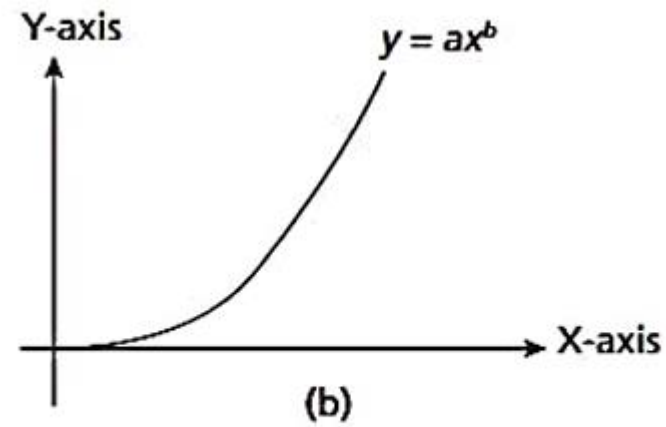
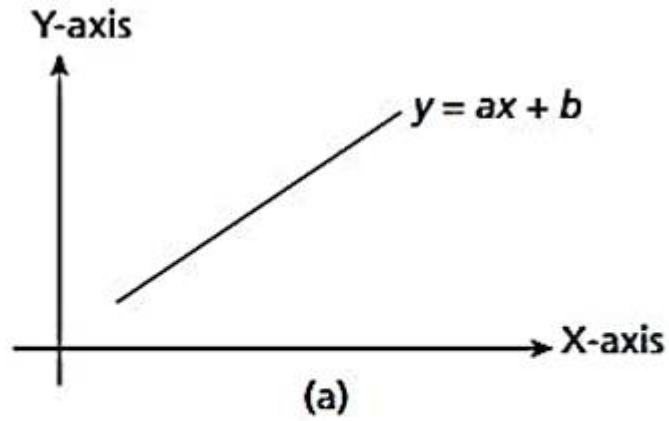


Figure 5.2: (a) Example of Linear Relationship of the Form $y = ax + b$ (b) Example of a Non-linear Relationship of the Form $y = ax^b$ (c) Examples of a Non-linear Relationship $y = \frac{x}{ax + b}$

Types of Regression Methods

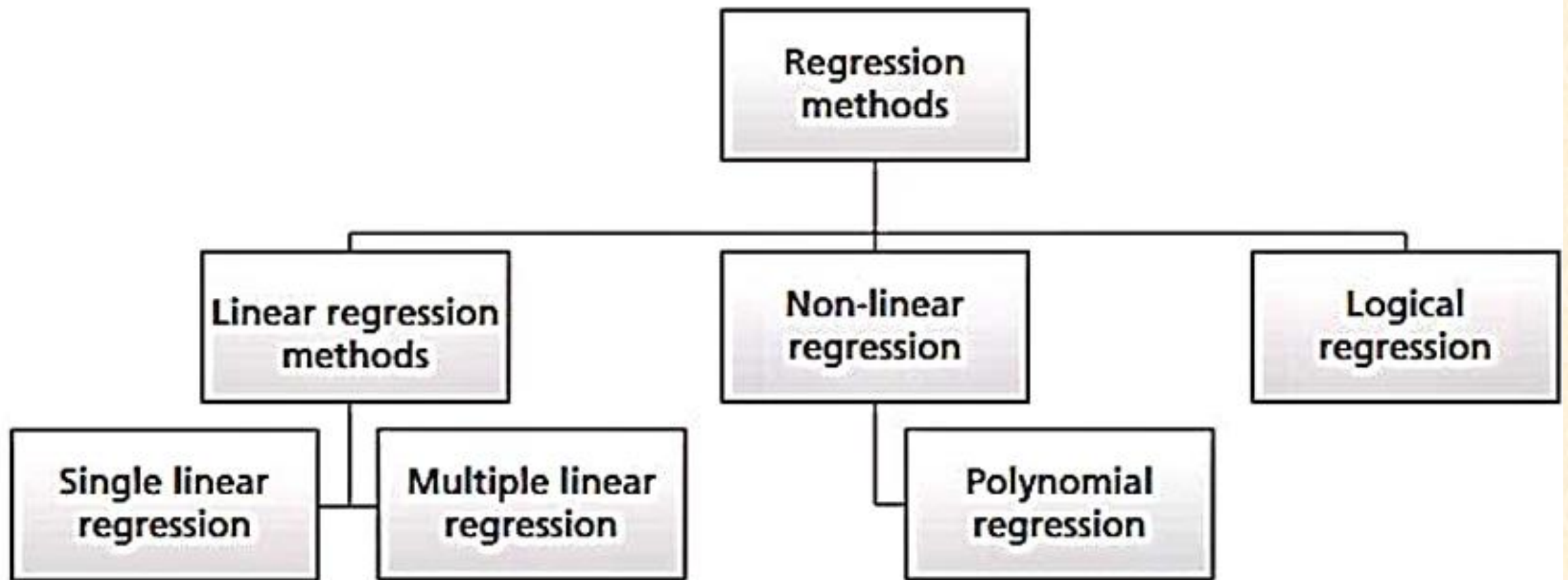


Figure 5.3: Types of Regression Methods

Limitations of Regression Method

1. Outliers – Outliers are abnormal data. It can bias the outcome of the regression model, as outliers push the regression line towards it.
2. Number of cases – The ratio of independent and dependent variables should be at least 20 : 1. For every explanatory variable, there should be at least 20 samples. Atleast five samples are required in extreme cases.
3. Missing data – Missing data in training data can make the model unfit for the sampled data.
4. Multicollinearity – If exploratory variables are highly correlated (0.9 and above), the regression is vulnerable to bias. Singularity leads to perfect correlation of 1. The remedy is to remove exploratory variables that exhibit correlation more than 1. If there is a tie, then the tolerance ($1 - R$ squared) is used to eliminate variables that have the greatest value.

INTRODUCTION TO LINEAR REGRESSION

In the simplest form, the linear regression model can be created by fitting a line among the scattered data points. The line is of the form given in Eq. (5.2).

$$y = a_0 + a_1 \times x + e \quad (5.2)$$

Here, a_0 is the intercept which represents the bias and a_1 represents the slope of the line. These are called regression coefficients. e is the error in prediction.

The assumptions of linear regression are listed as follows:

1. The observations (y) are random and are mutually independent.
2. The difference between the predicted and true values is called an error. The error is also mutually independent with the same distributions such as normal distribution with zero mean and constant variables.
3. The distribution of the error term is independent of the joint distribution of explanatory variables.
4. The unknown parameters of the regression models are constants.

The idea of linear regression is based on Ordinary Least Square (OLS) approach. This method is also known as ordinary least squares method. In this method, the data points are modelled using a straight line. Any arbitrarily drawn line is not an optimal line. In Figure 5.4, three data points and their errors (e_1, e_2, e_3) are shown. The vertical distance between each point and the line (predicted by the approximate line equation $y = a_0 + a_1x$) is called an error. These individual errors are added to compute the total error of the predicted line. This is called sum of residuals. The squares of the individual errors can also be computed and added to give a sum of squared error. The line with the lowest sum of squared error is called line of best fit.

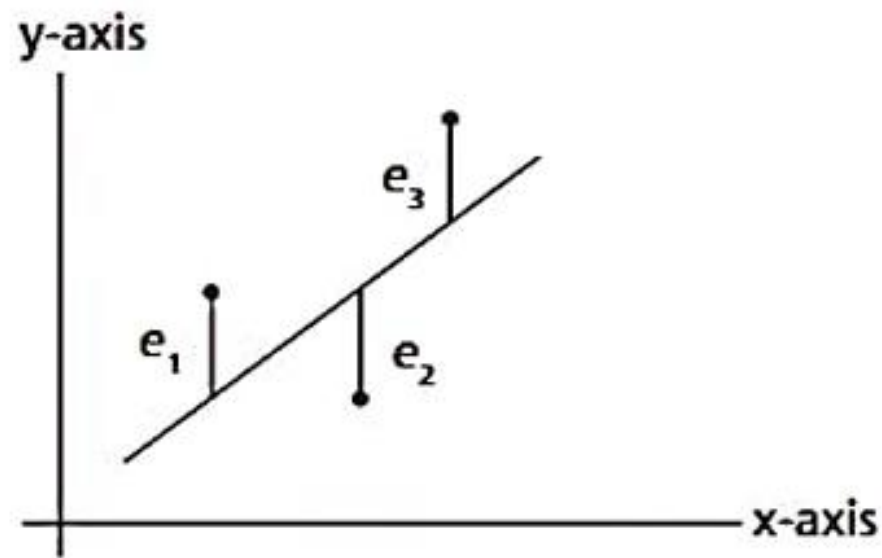


Figure 5.4: Data Points and their Errors

In another words, OLS is an optimization technique where the difference between the data points and the line is optimized.

Mathematically, based on Eq. (5.2), the line equations for points (x_1, x_2, \dots, x_n) are:

$$\begin{aligned}y_1 &= (a_0 + a_1x_1) + e_1 \\y_2 &= (a_0 + a_1x_2) + e_2 \\&\cdot \\&\cdot \\&\cdot \\y_n &= (a_0 + a_1x_n) + e_n\end{aligned}\tag{5.3}$$

In general, the error is given as: $e_i = y_i - (a_0 + a_1x_i)$ (5.4)

This can be extended into the set of equations as shown in Eq. (5.3).

Here, the terms (e_1, e_2, \dots, e_n) are error associated with the data points and denote the difference between the true value of the observation and the point on the line. This is also called as residuals. The residuals can be positive, negative or zero.

A regression line is the line of best fit for which the sum of the squares of residuals is minimum. The minimization can be done as minimization of individual errors by finding the parameters a_0 and a_1 such that:

$$E = \sum_{i=1}^n e_i = \sum_{i=1}^n (y_i - (a_0 + a_1x_i))\tag{5.5}$$

Or as the minimization of sum of absolute values of the individual errors:

$$E = \sum_{i=1}^n |e_i| = \sum_{i=1}^n |(y_i - (a_0 + a_1 x_i))| \quad (5.6)$$

Or as the minimization of the sum of the squares of the individual errors:

$$E = \sum_{i=1}^n (e_i)^2 = \sum_{i=1}^n (y_i - (a_0 + a_1 x_i))^2 \quad (5.7)$$

Sum of the squares of the individual errors, often preferred as individual errors (positive and negative errors), do not get cancelled out and are always positive, and sum of squares results in a large increase even for a small change in the error. Therefore, this is preferred for linear regression.

Therefore, linear regression is modelled as a minimization function as follows:

$$\begin{aligned} J(a_1, a_0) &= \sum_{i=1}^n [y_i - f(x_i)]^2 \\ &= \sum_{i=1}^n [y_i - (a_0 + a_1 x_i)]^2 \end{aligned} \quad (5.8)$$

Here, $J(a_0, a_1)$ is the criterion function of parameters a_0 and a_1 . This needs to be minimized. This is done by differentiating and substituting to zero. This yields the coefficient values of a_0 and a_1 . The values of estimates of a_0 and a_1 are given as follows:

$$a_1 = \frac{(\overline{xy}) - (\bar{x})(\bar{y})}{(\overline{x^2}) - (\bar{x})^2} \quad (5.9)$$

And the value of a_0 is given as follows:

$$a_0 = (\bar{y}) - a_1 \times \bar{x} \quad (5.10)$$

Example 5.1: Let us consider an example where the five weeks' sales data (in Thousands) is given as shown below in Table 5.1. Apply linear regression technique to predict the 7th and 9th month sales.

Table 5.1: Sample Data

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

Solution: Here, there are 5 items, i.e., $i = 1, 2, 3, 4, 5$. The computation table is shown below (Table 5.2). Here, there are five samples, so i ranges from 1 to 5.

Table 5.2: Computation Table

x_i	y_i	$(x_i)^2$	$x_i \times y_i$
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	Sum = 12.6	Sum = 55	Sum = 44.4
Average of (x_i)	Average of (y_i)	Average of (x_i^2)	Average of $(x_i \times y_i)$
$= \bar{x} = \frac{15}{5}$	$= \bar{y} = \frac{12.6}{5}$	$= \overline{x^2} = \frac{55}{5}$	$= \overline{xy} = \frac{44.4}{5}$
$= 3$	$= 2.52$	$= 11$	$= 8.88$

Let us compute the slope and intercept now using Eq. (5.9) as:

$$a_1 = \frac{8.88 - 3(2.52)}{11 - 3^2} = 0.66$$

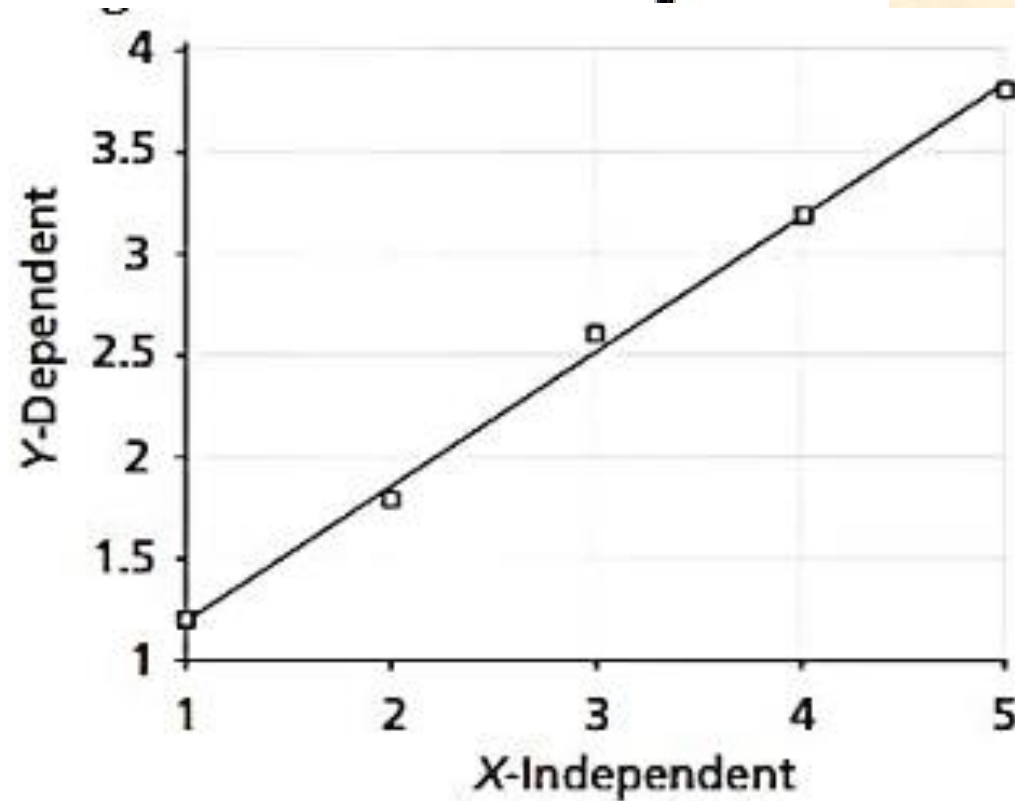
$$a_0 = 2.52 - 0.66 \times 3 = 0.54$$

Let us compute the slope and intercept now using Eq. (5.9) as:

$$a_1 = \frac{8.88 - 3(2.52)}{11 - 3^2} = 0.66$$

$$a_0 = 2.52 - 0.66 \times 3 = 0.54$$

The fitted line is shown in Figure 5.5.



— Regression line ($\hat{y} = 0.66x + 0.54$)

Figure 5.5: Linear Regression Model Constructed

Let us model the relationship as $y = a_0 + a_1 \times x$. Therefore, the fitted line for the above data is:
 $y = 0.54 + 0.66 \times x$.

The predicted 7th week sale would be (when $x = 7$), $y = 0.54 + 0.66 \times 7 = 5.16$ and the 12th month, $y = 0.54 + 0.66 \times 12 = 8.46$. All sales are in thousands.

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:

$$1. \text{ Computation of } (X^T X) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$$

$$2. \text{ Computation of matrix inverse of } (X^T X)^{-1} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$$

$$3. \text{ Computation of } ((X^T X)^{-1} X^T) = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix}$$

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

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

5.5 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:

$$\begin{aligned}y &= f(x_1, x_2) \\ &= a_0 + a_1x_1 + a_2x_2\end{aligned}\tag{5.21}$$

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

$$\begin{aligned}y &= f(x_1, x_2, x_3, \dots, x_n) \\ &= a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n + \varepsilon\end{aligned}\tag{5.22}$$

5.6 POLYNOMIAL REGRESSION

If the relationship between the independent and dependent variables is not linear, then linear regression cannot be used as it will result in large errors. The problem of non-linear regression can be solved by two methods:

1. Transformation of non-linear data to linear data, so that the linear regression can handle the data
2. Using polynomial regression

Transformations

The first method is called transformation. The trick is to convert non-linear data to linear data that can be handled using the linear regression method. Let us consider an exponential function $y = ae^{bx}$. The transformation can be done by applying log function to both sides to get:

$$\ln y = bx + \ln a \quad (5.24)$$

Similarly, power function of the form ($y = ax^b$) can be transformed by applying log function on both sides as follows:

$$\log_{10} y = b \log_{10} x + \log_{10} a \quad (5.25)$$

Once the transformation is carried out, linear regression can be performed and after the results are obtained, the inverse functions can be applied to get the desired result.

Polynomial Regression

It can handle non-linear relationships among variables by using n^{th} degree of a polynomial. Instead of applying transforms, polynomial regression can be directly used to deal with different levels of curvilinearity.

Polynomial regression provides a non-linear curve such as quadratic and cubic. For example, the second-degree transformation (called quadratic transformation) is given as: $y = a_0 + a_1x + a_2x^2$ and the third-degree polynomial is called cubic transformation given as: $y = a_0 + a_1x + a_2x^2 + a_3x^3$. Generally, polynomials of maximum degree 4 are used, as higher order polynomials take some strange shapes and make the curve more flexible. It leads to a situation of overfitting and hence is avoided.

Let us consider a polynomial of 2nd degree. Given points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, the objective is to fit a polynomial of degree 2. The polynomial of degree 2 is given as:

$$y = a_0 + a_1x + a_2x^2 \quad (5.26)$$

Such that the error $E = \sum_{i=1}^n [y_i - (a_0 + a_1x_i + a_2x_i^2)]^2$ is minimized. The coefficients a_0, a_1, a_2 of Eq. (5.26) can be obtained by taking partial derivatives with respect to each of the coefficients as $\frac{\partial E}{\partial a_0}, \frac{\partial E}{\partial a_1}, \frac{\partial E}{\partial a_2}$ and substituting it with zero. This results in 2 + 1 equations given as follows:

$$\begin{aligned}
na_0 + \left(\sum_{i=1}^n x_i\right)a_1 + \left(\sum_{i=1}^n x_i^2\right)a_2 &= \sum_{i=1}^n y_i \\
\left(\sum_{i=1}^n x_i\right)a_0 + \left(\sum_{i=1}^n x_i^2\right)a_1 + \left(\sum_{i=1}^n x_i^3\right)a_2 &= \sum_{i=1}^n x_i y_i \\
\left(\sum_{i=1}^n x_i^2\right)a_0 + \left(\sum_{i=1}^n x_i^3\right)a_1 + \left(\sum_{i=1}^n x_i^4\right)a_2 &= \sum_{i=1}^n x_i^2 y_i
\end{aligned} \tag{5.27}$$

The best line is the line that minimizes the error between line and data points. Arranging the coefficients of the above equation in the matrix form results in:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum(x_i, y_i) \\ \sum(x_i^2, y_i) \end{bmatrix} \tag{5.28}$$

This is of the form $Xa = B$. One can solve this equation for a as:

$$a = X^{-1}B \tag{5.29}$$