

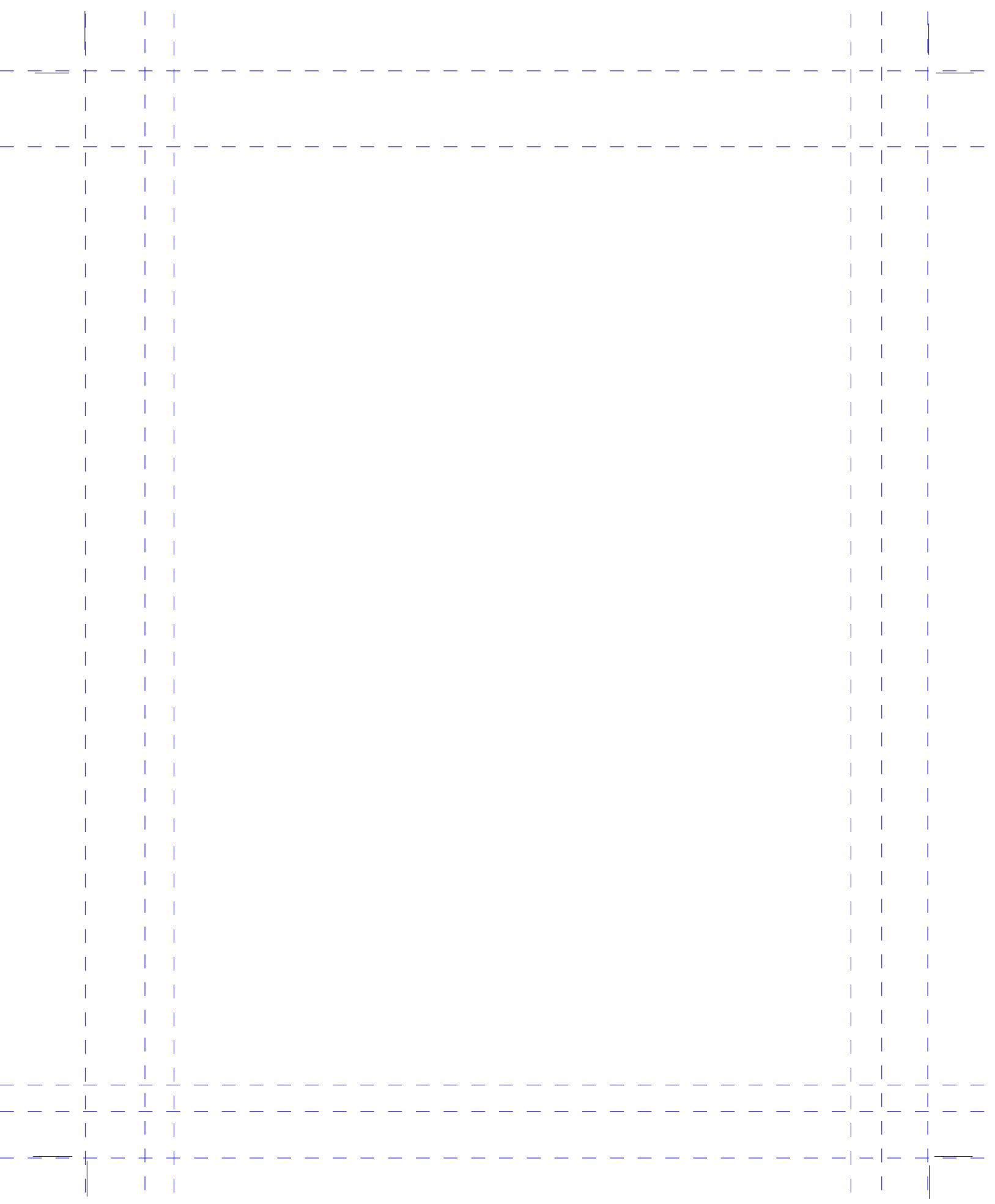
# Machine Learning



**Anuradha Srinivasaraghavan  
Vincy Joseph**

**WILEY**

# Machine Learning



# Machine Learning

**Anuradha Srinivasaraghavan**

*Associate Professor,  
Department of Computer Engineering,  
St. Francis Institute of Technology,  
Mumbai, Maharashtra*

**Vincy Joseph**

*Assistant Professor,  
Department of Computer Engineering,  
St. Francis Institute of Technology,  
Mumbai, Maharashtra*

**WILEY**

# Machine Learning

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002.

Cover Image: RooM RF/Getty Images

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or scanning without the written permission of the publisher.

**Limits of Liability:** While the publisher and the author have used their best efforts in preparing this book, Wiley and the author make no representation or warranties with respect to the accuracy or completeness of the contents of this book, and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose. There are no warranties which extend beyond the descriptions contained in this paragraph. No warranty may be created or extended by sales representatives or written sales materials. The accuracy and completeness of the information provided herein and the opinions stated herein are not guaranteed or warranted to produce any particular results, and the advice and strategies contained herein may not be suitable for every individual. Neither Wiley India nor the author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

**Disclaimer:** The contents of this book have been checked for accuracy. Since deviations cannot be precluded entirely, Wiley or its author cannot guarantee full agreement. As the book is intended for educational purpose, Wiley or its author shall not be responsible for any errors, omissions or damages arising out of the use of the information contained in the book. This publication is designed to provide accurate and authoritative information with regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services.

**Trademarks:** All brand names and product names used in this book are trademarks, registered trademarks, or trade names of their respective holders. Wiley is not associated with any product or vendor mentioned in this book.

Other Wiley Editorial Offices:

John Wiley & Sons, Inc. 111 River Street, Hoboken, NJ 07030, USA

Wiley-VCH Verlag GmbH, Pappelallee 3, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 1 Fusionopolis Walk #07-01 Solaris, South Tower, Singapore 138628

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada, M9W 1L1

First Edition: 2019

ISBN: 978-81-265-7851-1

ISBN: 978-81-265-8822-0 (ebk)

[www.wileyindia.com](http://www.wileyindia.com)

## **DEDICATION**

*We dedicate this book to our parents and family.*



# Preface

Machine learning belongs to a category of algorithms that allows for more accurate prediction without being explicitly programmed. It is similar to data mining and predictive modeling, especially where all three techniques require searching for patterns and adjusting program actions. Machine learning algorithms are classified into supervised and unsupervised learning algorithms. In the case of supervised learning algorithms, data scientists have a set of training data through which they build the model. The model is tested with testing data. In unsupervised learning algorithms, the model is built by not using the training dataset but by discovering new patterns. This book is for neophyte users who are getting acquainted to the concept of machine learning.

It is a myth that math is the primary prerequisite for machine learning. However, data analysis is the first skill requirement to work with machine learning algorithms. This is particularly true for beginners. Most of the work on using machine learning algorithms is centered on data preparation. After a compendious introduction on machine learning, the mathematics required for working with machine learning algorithms is dealt with in this book. This is followed in depth by the concept of data preprocessing. This helps the user understand the mathematics behind data preparation.

The year 2018 and beyond is set to witness an upward trend for professionals with skills in machine learning. These professionals would be the most sought after by recruiters and business enterprises in years to come.

This book is written keeping in mind the academic needs and requirements of undergraduate students. The language used in the book is simple and there are more than adequate examples. This book is an effort to deliver to the best of our abilities the concept of machine learning to beginners. It was a real challenge that was worth it at the end.

Our first-time writing experience has been a wonderful journey, from the time we decided to write the book till its present form. It has really been a challenging experience to complete the latest updates about the subject, keeping in mind that we are catering to first-time readers.

## About the Book

This book offers the readers the basics of machine learning in a very simple, user-friendly language. While browsing the Table of Contents, you will realize that you are given an introduction to every concept that comes under the umbrella of machine learning.

This book is aimed at students who are new to the topic of machine learning. It is meant for students studying machine learning in their undergraduate and postgraduate courses in information technology. It is also aimed at computer engineering students. It will help familiarize students with the terms and terminologies used in machine learning. We hope that this book serves as an entry point for students to pursue their future studies and careers in machine learning. The worked-out examples in the book are completely focused from academic point of view.

The book is organized in four parts. Part I in general introduces the subjects and revision of all the basics required for working with machine learning algorithms. There are five chapters in Part I. Chapter 1 gives a brief introduction to machine learning. It gives answers to the questions such as: What is machine learning? Where is machine learning used? What are the types of machine learning? Two real-time case studies

are also explained in this chapter. Once the reader gets acquainted to the basics of machine learning, Chapter 2 introduces the representation of the model. Chapters 3 and 4 elaborate on the basics of matrices and the basics of Python, which serve as the prerequisite for understanding machine learning. Chapter 5 gives an introduction to the different techniques required for data preprocessing.

Part II covers the topic of supervised learning algorithms. Chapter 6 introduces artificial neural networks (ANN). ANN of late has become very popular, with lots of applications using deep learning. So we felt it important that the concept of ANN be introduced to the readers. This is followed in Chapters 7 and 8, which detail regression techniques. Chapter 9 on decision tree, chapter 10 on support vector machine, and chapter 11 on Bayesian classification give extensive examples and solved numericals on supervised learning techniques which are popular and mostly used. Chapter 12 introduces the Hidden Markov Model in brief.

Part III has a chapter on unsupervised learning algorithms, which covers the basics of clustering. The chapter has extensive examples on different types of clustering and expectation maximization. This Part is dedicated only to unsupervised learning techniques.

Part IV has a chapter on optimization techniques. This chapter gives an introduction to formulate optimization problems, and the commonly used derivative-free and derivative-based optimization algorithms.

## Instructor Resources

---

The following resources are available for instructors on request. Please send in your requests to [acadmktg@wiley.com](mailto:acadmktg@wiley.com)

1. Chapter-wise PowerPoint Presentations (PPTs)
2. Chapter-wise Solution Manual

## Acknowledgements

---

This book is based on the research in the field of machine learning. It is framed as per the curriculum of the undergraduate engineering students. We are grateful to a number of friends, colleagues, and well-wishers who encouraged us to start the work, persevere with it, and finally to publish it.

The provocation behind writing this book is to provide the basics of machine learning in an easy, simple, and understandable form to the undergraduate students.

We would like to thank all the universities that have included this subject in their curriculum. Since this is a topic of great interest to us, writing this book was a great pleasure. We all know that knowledge shared is knowledge doubled, so this writing experience has really doubled our knowledge.

We would like to thank our parents and spouses for encouraging and motivating us to write this book. Our daughters were a source of inspiration and their constant support helped us in completing this book.

Above all, we thank God, the almighty, who is the author of knowledge and wisdom for showering on us unconditional love.

**Anuradha Srinivasaraghavan  
Vincy Joseph**

## About the Authors



**Anuradha Srinivasaraghavan** is an academician in the University of Mumbai. Her prime interests are in the areas of machine learning, soft computing, data mining, and databases. She actively participates in content development of the subjects. She also participates in research avenues in the areas of machine learning and soft computing. She completed her postgraduate in 2008 in Computer Engineering. Since then, she has been working in the areas of machine learning, data mining, soft computing, and artificial intelligence. She has around 15 plus publications in reputed national and international journals and conferences. She has guided more than 25 students in undergraduate and around 6 students in postgraduate projects and research work. She is currently Associate Professor in the Department of Computer Engineering, St. Francis Institute of Technology, University of Mumbai, for the past 15 years.



**Vincy Elizabeth Joseph** is working as Assistant Professor in the Department of Computer Engineering, St. Francis Institute of Technology, University of Mumbai, for the past 15 years. Her prime interests are in the areas of machine learning, system security, soft computing, digital signal processing, microprocessors, and computer organization and architectures. She actively participates in the design of experiments and content development of the subjects, especially in the areas of machine learning and soft computing. She completed her graduation from Cochin University of Science and Technology in 2004 and postgraduation in Computer Engineering in 2010. She has various publications in reputed national and international journals and conferences. She has guided many undergraduate and postgraduate students in postgraduate projects and research work.



# Contents

Preface	vii
About the Authors	ix

<b>Part 1 Basics of Machine Learning</b>	
<b>Chapter 1 Introduction to Machine Learning</b>	
Learning Objectives	3
Learning Outcomes	3
1.1 What is Machine Learning?	3
1.1.1 Common Definitions of Machine Learning	4
1.2 Where is Machine Learning Used?	6
1.3 Applications of Machine Learning	7
1.3.1 Marketing and Sales	7
1.3.2 Search Engines	7
1.3.3 Transportation	7
1.4 Types of Machine Learning	8
1.4.1 Supervised Learning	8
1.4.2 Unsupervised Learning	10
1.4.3 Reinforcement Learning	12
Case Study 1: Diagnosing Crop Disease with Machine Learning	14
Case Study 2: Learning to Decode the Immune System to Diagnose Disease	14
Summary	15
Multiple-Choice Questions	15
Very Short Answer Questions	16
Short Answer Questions	16
Review Questions	16
Answers	16
<b>Chapter 2 Model and Cost Function</b>	
Learning Objectives	17
Learning Outcomes	17
2.1 Introduction	17
2.2 Representation of a Model	17
2.3 Cost Function Notation for Measuring the Accuracy of a Hypothesis Function	18
2.4 Measuring Accuracy of a Hypothesis Function	21

2.5	Minimizing the Cost Function for a Single-Variable Function	21
2.6	Minimizing the Cost Function for a Two-Variable Function	23
2.7	Role of Gradient Function in Minimizing a Cost Function	24
	Summary	25
	Multiple-Choice Questions	25
	Very Short Answer Questions	26
	Short Answer Questions	26
	Review Questions	26
	Answers	26

---

<b>Chapter 3</b>	<b>Basics of Vectors and Matrices</b>	<b>27</b>
------------------	---------------------------------------	-----------

	Learning Objectives	27
	Learning Outcomes	27
3.1	Introduction	27
3.2	Notations	27
3.3	Types of Matrices	28
3.4	Matrix Operations	29
	<i>3.4.1 Matrix Addition</i>	29
	<i>3.4.2 Matrix Scalar Multiplication</i>	30
	<i>3.4.3 Negative of a Matrix</i>	30
	<i>3.4.4 Matrix Subtraction</i>	30
	<i>3.4.5 Matrix Multiplication</i>	31
	<i>3.4.6 Matrix Transpose</i>	32
3.5	Determinant of a Matrix	32
3.6	Inverse of a Matrix	33
	Summary	35
	Multiple-Choice Questions	35
	Very Short Answer Questions	36
	Review Questions	36
	Answers	36

---

<b>Chapter 4</b>	<b>Basics of Python</b>	<b>37</b>
------------------	-------------------------	-----------

	Learning Objectives	37
	Learning Outcomes	37
4.1	Introduction	37
	<i>4.1.1 Versions of Python</i>	38
	<i>4.1.2 Features of Python</i>	38
4.2	Installing Python	39
	<i>4.2.1 Running Python</i>	39
4.3	Anaconda	40
	<i>4.3.1 Installing Anaconda</i>	40

4.4	Running Jupyter Notebook	41
	4.4.1 Spyder	43
4.5	Python 3: Basic Syntax	43
4.6	Python Identifiers	44
	4.6.1 Reserved Words	44
	4.6.2 Lines and Indentation	44
	4.6.3 Variables in Python 3	45
	4.6.4 Standard Data Types	45
4.7	Basic Operators in Python	48
	4.7.1 Arithmetic Operators	48
	4.7.2 Comparison (Relational) Operators	49
	4.7.3 Assignment Operators	50
	4.7.4 Logical and Membership Operators	50
4.8	Python Decision-Making	50
4.9	Python Loops	53
	4.9.1 While loop	53
	4.9.2 For Loop	53
	4.9.3 Nested Loops	54
	4.9.4 Loop Control Statements	55
4.10	Numerical Python (NumPy)	57
	4.10.1 Pip Install NumPy	57
	4.10.2 Array Attributes of NumPy	59
4.11	NumPy Matplotlib	61
	4.11.1 Subplot	62
4.12	Introduction to Pandas	63
	4.12.1 Creating DataFrame in Pandas	63
	4.12.2 Manipulation of Columns	65
4.13	Introduction to Scikit-Learn	67
	Summary	68
	Multiple-Choice Questions	68
	Very Short Answer Questions	69
	Review Questions	70
	Answers	70

---

## Chapter 5 Data Preprocessing

71

---

	Learning Objectives	71
	Learning Outcomes	71
5.1	Overview of Data Preprocessing	71
5.2	Data Cleaning	72
	5.2.1 Filling in Missing Values	72
	5.2.2 Cleaning and Filling Missing Data	74
	5.2.3 Drop Missing Values	76
	5.2.4 Smoothing Noisy Data	77
	5.2.5 Removing Inconsistencies	80

5.3	Data Integration	80
5.4	Data Transformation	81
5.5	Data Reduction or Dimensionality Reduction	81
	<i>5.5.1 Principal Component Analysis</i>	81
	<i>5.5.2 Linear Discriminant Analysis</i>	97
	<i>5.5.3 Independent Component Analysis</i>	97
	Summary	100
	Multiple-Choice Questions	101
	Very Short Answer Questions	102
	Short Answer Questions	102
	Review Questions	102
	Answers	102

## Part 2 Supervised Learning Algorithms 103

<b>Chapter 6</b>	<b>Artificial Neural Networks</b>	<b>105</b>
Learning Objectives		105
Learning Outcomes		105
6.1	Introduction	105
	<i>6.1.1 Introduction to Artificial Neural Networks</i>	105
	<i>6.1.2 Use of Artificial Neural Networks</i>	106
6.2	Evolution of Neural Networks	106
6.3	Biological Neuron	107
	<i>6.3.1 From Human Neurons to Artificial Neurons</i>	107
6.4	Basics of Artificial Neural Networks	108
	<i>6.4.1 Network Architecture</i>	108
	<i>6.4.2 Learning</i>	110
6.5	Activation Functions	112
	<i>6.5.1 Bipolar Activation Functions</i>	112
	<i>6.5.2 Unipolar Activation Functions</i>	113
	<i>6.5.3 Identity Function</i>	113
	<i>6.5.4 Ramp Function</i>	113
6.6	McCulloch–Pitts Neuron Model	114
	<i>6.6.1 Solved Problems</i>	115
	Summary	120
	Multiple-Choice Questions	121
	Very Short Answer Questions	121
	Short Answer Questions	122
	Review Questions	122
	Answers	122

---

<b>Chapter 7 Linear Regression</b>	<b>123</b>
Learning Objectives	123
Learning Outcomes	123
7.1 Introduction to Supervised Learning and Regression	123
7.2 Statistical Relation between Two Variables and Scatter Plots	125
7.2.1 <i>Purpose of Regression</i>	126
7.2.2 <i>Linear Regression</i>	126
7.2.3 <i>Requirement of Linear Regression</i>	126
7.3 Steps to Establish a Linear Regression	126
7.4 Evaluation of Model Estimators	130
7.4.1 <i>Karl Pearson's Coefficient of Correlation</i>	130
7.4.2 <i>R-Square</i>	131
7.4.3 <i>Standard Error of Estimate</i>	132
7.5 Solved Problems on Linear Regression	133
Summary	135
Multiple-Choice Questions	135
Very Short Answer Questions	136
Short Answer Questions	136
Review Questions	137
Answers	137
<b>Chapter 8 Logistic Regression</b>	<b>139</b>
Learning Objectives	139
Learning Outcomes	139
8.1 <i>Introduction to Logistic Regression</i>	139
8.2 Scenarios Which Require Logistic Regression	139
8.2.1 <i>Examples of Binary Classification Model</i>	140
8.3 Odds	141
8.3.1 <i>Odd and Odds ratio</i>	141
8.4 Building Logistic Regression Model (Logit Function)	141
8.5 Maximum Likelihood Estimation	142
8.5.1 <i>Estimating the Parameters of Maximum Estimation</i>	143
8.5.2 <i>Using Gradient Descent Algorithm</i>	144
8.6 Example of Logistic Regression	144
Case Study	148
Summary	153
Multiple-Choice Questions	154
Very Short Answer Questions	155
Short Answer Questions	155
Review Questions	156
Answers	156

<b>Chapter 9 Decision Tree</b>	<b>157</b>
Learning Objectives	157
Learning Outcomes	157
9.1 Introduction to Classification and Decision Tree	157
9.2 Problem Solving Using Decision Trees	158
9.3 Basic Decision Tree Learning Algorithm	158
<i>9.3.1 Attribute Selection Measures</i>	158
<i>9.3.2 Iterative Dichotomiser 3 (ID3)</i>	162
9.4 Popularity of Decision Tree Classifiers	162
9.5 Steps to Construct a Decision Tree	162
9.6 Classification Using Decision Trees	169
9.7 Issues in Decision Trees	169
<i>9.7.1 Underfitting and Overfitting</i>	170
<i>9.7.2 Approaches to Identify Final Tree Size</i>	170
<i>9.7.3 Incorporating Continuous-Valued Attributes</i>	172
<i>9.7.4 Alternative Measures for Selecting Attributes</i>	173
<i>9.7.5 Handling Training Examples with Missing Attribute Values</i>	173
<i>9.7.6 Handling Attributes with Differing Costs</i>	173
9.8 Rule-Based Classification	173
<i>9.8.1 Using IF–THEN Rules for Classification</i>	173
<i>9.8.2 Working of Rule-Based Classifier</i>	174
9.9 Pruning the Rule Set	175
<i>9.9.1 Rule Induction Using a Sequential Covering Algorithm</i>	175
Summary	177
Multiple-Choice Questions	177
Very Short Answer Questions	178
Short Answer Questions	178
Review Questions	179
Answers	179
<b>Chapter 10 Support Vector Machines</b>	<b>181</b>
Learning Objectives	181
Learning Outcomes	181
10.1 Introduction to Support Vector Machines	181
10.2 Linear Support Vector Machines	182
<i>10.2.1 Separating Hyperplane</i>	182
10.3 Optimal Hyperplane	183
<i>10.3.1 Relationship between Margin and Optimal Hyperplane</i>	185
10.4 Basics of Vectors	186
<i>10.4.1 What is a Vector?</i>	186
<i>10.4.2 Norm of Vector</i>	187
<i>10.4.3 Sum of Two Vectors</i>	189

---

10.4.4	<i>Difference between Two Vectors</i>	189
10.4.5	<i>Dot Product</i>	190
10.4.6	<i>Orthogonal Projection of a Vector</i>	190
10.4.7	<i>Equation of Hyperplane</i>	192
10.4.8	<i>Computation of Distance from a Point to the Hyperplane</i>	192
10.4.9	<i>Computation of the Margin of the Hyperplane</i>	194
10.5	Radial Basis Functions	202
10.5.1	<i>Radial Basis Function Networks</i>	203
10.5.2	<i>Single Layer RBF Network</i>	203
10.5.3	<i>Basic Architecture of the RBF Network</i>	204
10.5.4	<i>Cover's Theorem</i>	205
	Summary	207
	Multiple-Choice Questions	208
	Very Short Answer Questions	208
	Short Answer Questions	208
	Review Questions	209
	Answers	209

## Chapter 11 Bayesian Classification 211

---

	Learning Objectives	211
	Learning Outcomes	211
11.1	Introduction to Bayesian Classifiers	211
11.1.1	<i>Bayes' Theorem</i>	211
11.2	Naive Bayes Classifier	212
11.2.1.	<i>Example of Naive Bayes Classifier</i>	212
11.3	Bayesian Belief Networks	214
11.3.1	<i>Learning Bayesian Network Parameters</i>	215
11.4	k-Nearest Neighbor (KNN)	216
11.4.1	<i>Choosing Parameters for Nearest Neighbor</i>	217
11.4.2	<i>KNN Algorithm</i>	218
11.4.3	<i>Numerical Example of KNN</i>	218
11.4.4	<i>Features of KNN</i>	219
11.4.5	<i>Advantages and Disadvantages of KNN</i>	220
11.5	Measuring Classifier Accuracy	220
11.5.1	<i>Confusion Matrix</i>	221
11.5.2	<i>Accuracy</i>	221
11.5.3	<i>Precision</i>	221
11.5.4	<i>Recall or Sensitivity</i>	222
11.5.5	<i>Specificity</i>	222
11.5.6	<i>F-Score</i>	222
	Summary	222
	Multiple-Choice Questions	223
	Very Short Answer Questions	223

Short Answer Questions	224
Review Questions	225
Answers	225

---

**Chapter 12 Hidden Markov Model** **227**

Learning Objectives	227
Learning Outcomes	227
12.1 Introduction to Hidden Markov Model	227
12.2 Issues in Hidden Markov Model	230
<i>12.2.1 Evaluation Problem</i>	231
<i>12.2.2 Decoding Problem</i>	232
<i>12.2.3 Learning Problem</i>	233
<i>12.2.4 Learning Problem (Classifier)</i>	234
Summary	234
Multiple-Choice Questions	235
Short Answer Questions	235
Review Questions	235
Answers	235

---

**Part 3 Unsupervised Algorithms** **237**

---

**Chapter 13 Introduction to Unsupervised Learning Algorithms** **239**

Learning Objectives	239
Learning Outcomes	239
13.1 Introduction to Clustering	239
<i>13.1.1 Applications of Clustering</i>	240
<i>13.1.2 Requirements of Clustering</i>	240
13.2 Types of Clustering	243
<i>13.2.1 Partitioning Method</i>	243
<i>13.2.2 Hierarchical Method</i>	243
<i>13.2.3 Density-Based Methods</i>	244
<i>13.2.4 Grid-Based Methods</i>	244
13.3 Partitioning Methods of Clustering	244
<i>13.3.1 k-Means Algorithm</i>	244
<i>13.3.2 k-Medoids</i>	249
13.4 Hierarchical Methods	252
<i>13.4.1 Agglomerative Algorithms</i>	252
Case Study	263
Case Study 1: Grouping of Similar Companies Based on Wikipedia Articles	263
Case Study 2: Telecom Cluster Analysis	263
Summary	264

---

Multiple-Choice Questions	265
Very Short Answer Questions	266
Short Answer Questions	266
Review Questions	267
Answers	268

**Part 4 Optimization Techniques****269**

---

<b>Chapter 14 Optimization</b>	<b>271</b>
Learning Objectives	271
Learning Outcomes	271
14.1 Introduction to Optimization	271
<i>14.1.1 Statement of an Optimization Problem</i>	272
14.2 Classification of Optimization Problems	272
14.3 Linear vs Nonlinear Programming Problems	272
14.4 Unconstrained Minimization Problems	273
14.5 Gradient-Based Methods (Descent Methods)	273
<i>14.5.1 Steepest Descent (Cauchy) Method</i>	273
<i>14.5.2 Newton's Method</i>	275
14.6 Introduction to Derivative-Free Optimization	277
<i>14.6.1 Random Search Method</i>	277
<i>14.6.2 Downhill Simplex Method</i>	277
<i>14.6.3 Genetic Algorithm</i>	279
<i>14.6.4 Simulated Annealing</i>	279
14.7 Derivative-Based vs Derivative-Free Optimization	281
Summary	282
Multiple-Choice Questions	283
Very Short Answer Questions	283
Short Answer Questions	283
Review Questions	284
Answers	284
Appendices	285
Bibliography	301
Index	305



■ ■

# PART 1

# Basics of Machine Learning

---

## **Chapter 1**

Introduction to Machine Learning

## **Chapter 2**

Model and Cost Function

## **Chapter 3**

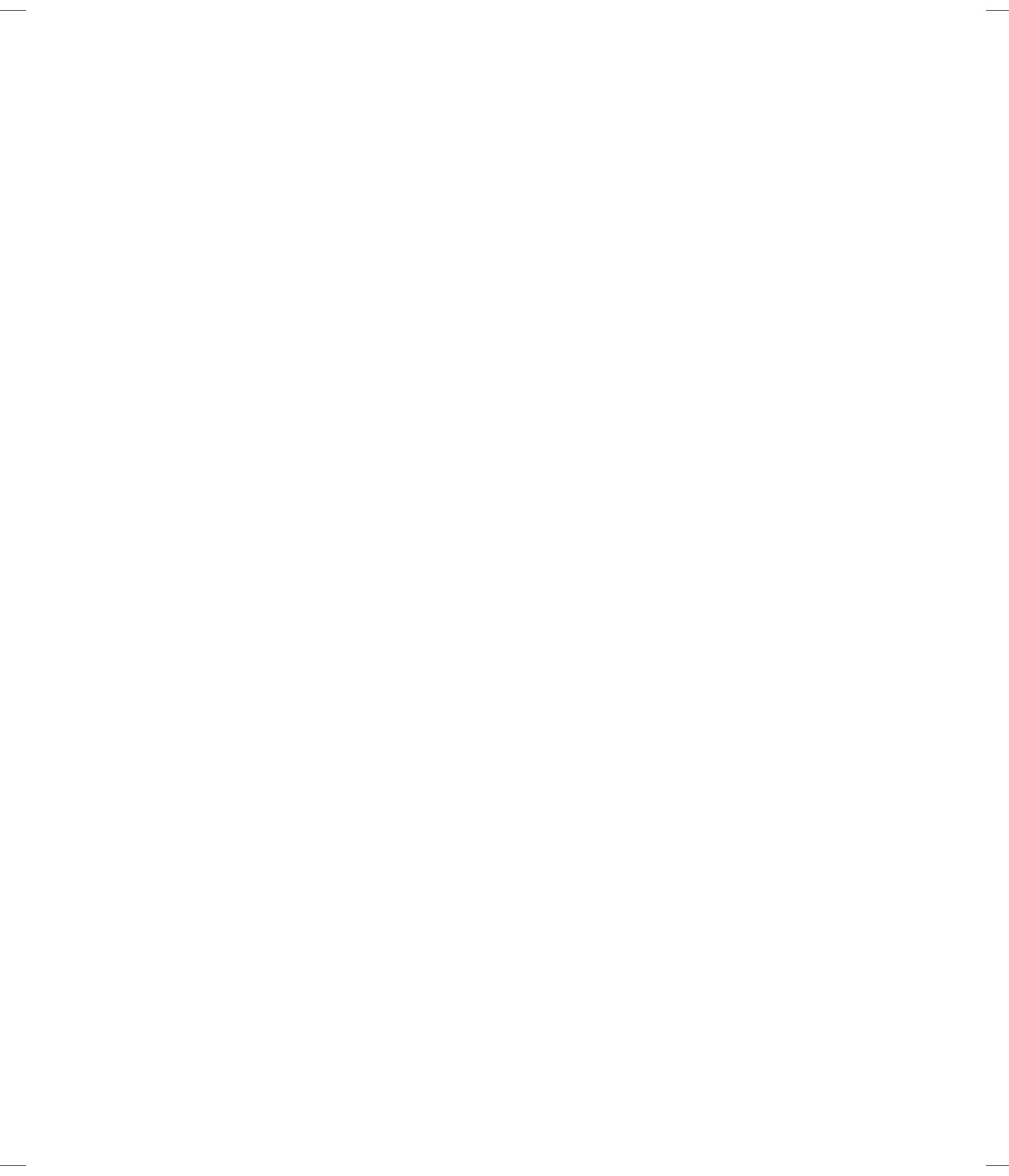
Basics of Vectors and Matrices

## **Chapter 4**

Basics of Python

## **Chapter 5**

Data Preprocessing



# 1

# Introduction to Machine Learning

## LEARNING OBJECTIVES

- To make students understand the importance and significance of machine learning.
- To differentiate between different types of machine learning techniques.
- To understand how machine learning techniques can be used for practical applications.

## LEARNING OUTCOMES

- Students will be able to define the term machine learning and comprehend different types of learning.
- Students will be able to differentiate between supervised, unsupervised and reinforcement learning techniques.
- Students will be able to sketch appropriate machine learning techniques for different applications.

### 1.1

### What is Machine Learning?

Machine learning (ML) is an area in computer science which involves teaching computers to do things naturally by learning through experience. This means that the computer system is turning data into information. Machine learning algorithms derive information from data without the help of any model. The algorithms adaptively improve their performance as we increase for example, the number of tuples in the dataset. It is equivalent to the fact that learning happens with experience.

Tom M Mitchell, one of the patrons of machine learning, defined it as follows:

*A computer program is said to learn from experience ( $E$ ) with respect to some class of tasks ( $T$ ) and performance measure ( $P$ ), if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .*

For example, a computer program that learns to play tic-tac-toe might improve its performance, as measured by its ability to win the game at the class of tasks involved in playing the game (all X's and O's should follow a horizontal, vertical or diagonal line), through experience obtained by playing games against itself. In general, to have a well-defined learning problem, we must identify the following three features,

1. Class of tasks.
2. Performance measurement that needs to be improved.
3. Source of experience.

Consider as an example, robot navigation in a maze. Here

1. Class of task: Reaching the end of the maze.
2. Performance measurement: Time taken to reach the end of the maze.
3. Source of experience: Navigating the maze from start to finish by the robot.

### 1.1.1 Common Definitions of Machine Learning

Nvidia defines machine learning as “Machine learning at its most basic is the practise of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.”

Stanford University defines it as “Machine learning is the science of getting computers to act without being explicitly programmed.”

McKinsey & Company states that “Machine learning is based on algorithms that can learn from data without relying on rules-based programming.”

According to Pedro Domingos from Department of Computer Engineering University of Washington, in his paper *A Few Useful Things to Know about Machine Learning* states that “Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.”

Carnegie Mellon University’s machine learning department has many research projects on the subject. According to the researches in that prominent university, “The field of machine learning seeks to answer the question ‘How can we build computer systems that automatically improve with experience, and the fundamental laws that govern all learning processes?’

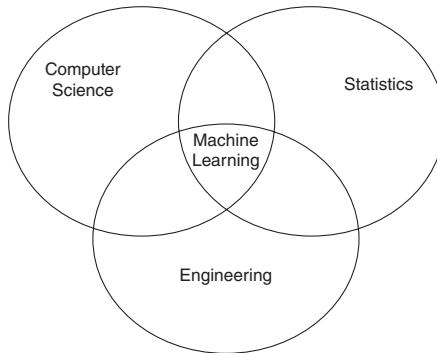
Machine learning lies at the intersection of computer science, engineering and statistics and often appears in other disciplines as shown in Fig. 1.1. It can be applied to many fields—from politics to geosciences. Any field that needs to interpret and act on data can benefit from machine learning techniques as shown in the following examples.

#### Example 1.1

On online traffic networks, mobile applications estimate price of ride and time of arrival at destination using machine learning. For instance, Uber ATG uses machine learning to define price of the ride based on the demand of riders and traffic in that region.

#### Example 1.2

Facebook, which continuously notices the friends in the list, profiles often visited, your interests, workplace, groups you are in and so on. Based on the information retrieved, Facebook gives you friend suggestions.



**Figure 1.1** Machine learning is an intersection.

---

**Example 1.3**

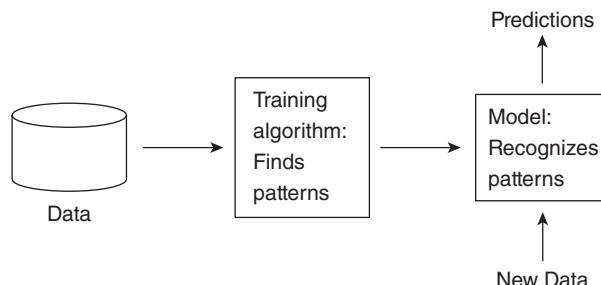
Consider that you purchased an item from Amazon. After your purchase, you may find pop-up advertisements for shopping. The product recommendations may be related to what you purchased. For instance, if you purchased a mobile phone online, then the site from where you purchased it immediately recommends a cover for the phone purchased as shown in Fig. 1.2.

The screenshot shows two separate Amazon order pages. The top section displays an order placed on December 27, 2017, for a Xiaomi Redmi MI 4 Defender Stylish Hard Back Armor Shock Proof Case Cover with a back stand feature. The total price was ₹ 258.00. The bottom section shows an order placed on December 21, 2017, for a Redmi 4 (Black, 64GB) at ₹ 10,398.00. Both sections include links for 'Return or replace items' and 'Write a product review'.

**Figure 1.2** Recommendation of cover with purchase of phone.

Machine learning uses algorithms to find patterns in data and then uses a model that recognizes those patterns to make predictions on new data. This is nothing but pattern recognition (Fig. 1.3).

Machine learning can be implemented in the healthcare sector, providing opportunities to facilitate and enhance the work of medical experts and to improve the efficiency and quality of medical care. In medical diagnosis, the main interest is in establishing the existence of a disease followed by its accurate identification. Machine learning can improve the accuracy of medical diagnosis by analyzing historical data of patients for various categories of diseases. Measurements in this application are typically results of certain medical tests (such as blood pressure, temperature and various blood tests), medical diagnostics (such as medical images), presence/absence/intensity of various symptoms and basic physical information



**Figure 1.3** Pattern recognition using machine learning.

of the patient (age, sex, weight, and so on). On the basis of the results of these measurements, a model can be created that analyses the test results of a new patient and predicts the chance and severity of the disease. Based on the results of the model, further tests can be suggested. Proper preventive and prognostic measures can then be advised by doctors.

## 1.2 Where is Machine Learning Used?

---

Are you fascinated by how Netflix recommends movies you might like? Have you wondered how Google shows you such accurate search results? Machine learning is behind these technological advances. It represents a key evolution in the fields of computer science, data analysis, software engineering and artificial intelligence.

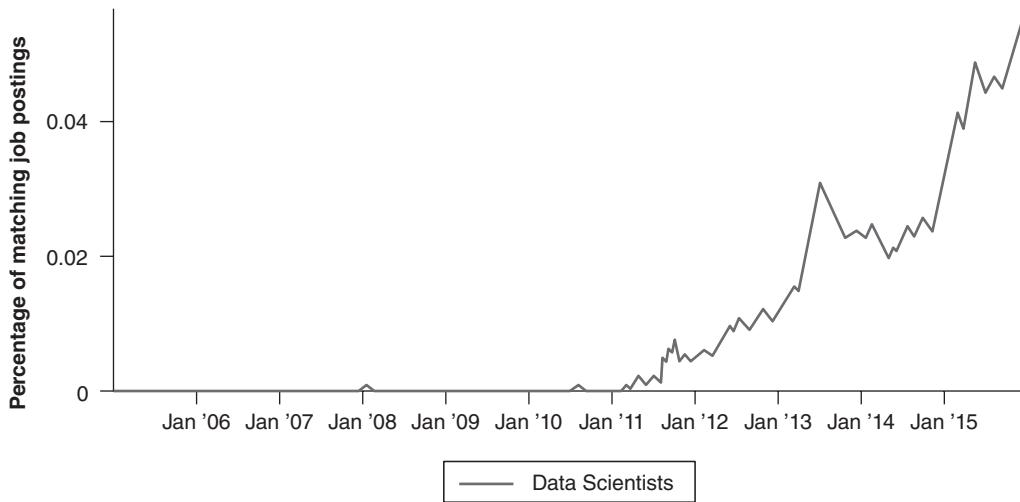
Figure 1.4 shows how machine learning is used in Google.

The screenshot shows a job search interface for 'Jobs Near Mumbai, Maharashtra'. At the top, there are filters for 'Computer and IT', 'Past 3 days', 'Full-time', 'Science and Engineering', 'Management', and 'Human Resources'. Below these filters, three job listings are displayed:

- C Data Science/ Machine Learning Engineer- Mumbai (4-5 Years Of...)**  
Confidential  
Mumbai, Maharashtra  
via Analytics Vidhya  
Over 1 month ago Full-time
- E Artificial Intelligence / Machine learning**  
ELITE RECRUITMENTS  
Mumbai, Maharashtra  
via TimesJobs  
6 days ago Full-time
- fractaleos Data Sciences - Fractal Analytics**  
Mumbai, Maharashtra  
via LinkedIn  
3 hours ago Full-time

**Figure 1.4** Google using machine learning.

Last decade has seen immense growth in the field of artificial intelligence and data science. “Indeed” is an American worldwide employment-related search engine for job listings. It was launched in November 2004. As a single-topic search engine, it is also an example of vertical search. Indeed is currently available in over 60 countries and 28 languages. In October 2010, Indeed.com passed Monster.com to become the highest-traffic job website in the United States. According to their statistics, there has been a 200% increase in “data scientist” job searches and a 50% increase in job listings. The statistical plot is given in Fig. 1.5.



**Figure 1.5** Job trends from Indeed.com.

Many technologies today benefit from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Recommendation engines, powered by machine learning, suggest what products you should buy based on user preferences. Self-driving cars using machine learning to navigate may soon be available to consumers.

## 1.3 Applications of Machine Learning

### 1.3.1 Marketing and Sales

The purchase pattern is complicated because customers engage in businesses through a variety of methods. Digital marketing creates a plethora of customer data that needs to be analyzed and acted upon to improve sales. Insights from the large dataset of sales are required to be collected to develop new strategies to improve the sale. This is where machine learning comes into picture. It uses algorithms to quickly interpret diverse datasets and build correlations. As a result, marketing and sales departments are able to analyze the path to purchase and understand how they can optimize the buyer's experience.

### 1.3.2 Search Engines

Search engines use machine learning algorithm to find the best personalized match for your query. Google, the world's biggest search engine now offers recommendations and suggestions based on previous user searches using machine learning. In 2015, Google introduced RankBrain—a machine learning algorithm used to decipher the semantic content of a search query. Through the use of an intuitive neural network, RankBrain identifies the intent behind a user's search and offers them tailored information on that particular topic.

### 1.3.3 Transportation

Based on travel history and pattern of traveling across various routes, machine learning can help transportation companies predict potential problems that could arise on certain routes, and accordingly advise their customers to opt for a different route. Transportation firms like Uber and delivery organizations like Swiggy

are increasingly using machine learning technology to carry out data analysis and data modeling to make informed decisions and ease their customer's experience.

## 1.4

## Types of Machine Learning

---

Two of the most widely adopted machine learning methods are supervised learning and unsupervised learning. Supervised learning trains algorithms based on example input and output data that is labeled by humans. Unsupervised learning provides the algorithm with no labeled data in order to allow it to find structure or pattern within its input data. Let's explore these two methods in more detail.

### 1.4.1 Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data. It is similar to how a teacher teaches his/her students. The teacher, or tutor, examines the performance of the students and corrects with the solution known to him/her or desired by him/her.

For example, suppose you have a fruit basket and your task is to arrange the fruit by type (Fig. 1.6).



**Figure 1.6** Fruit basket.

You can group the fruits based on any physical character. For example, if you arrange them according to color, then the groups will be arranged as such: red color group—apples and cherries, and green color group—watermelons and grapes (Fig. 1.7).

The other feature is the size of the fruit. We all know that apples and cherries are both round in shape. But the size of an apple is bigger than that of a cherry. In the second group, watermelon is much bigger than grapes in size.

Red color group: Apples and Cherries



Green color group: Watermelons and Grapes



**Figure 1.7** Fruit basket categorization based on color.

So if we want to separate the fruits from a fruit basket based on color, which is the first feature, then apples and cherries would be categorized in one group and watermelon and grapes would be in another group. But if we want to automate this process of fruit categorization then the feature “color” alone would not be sufficient to group them. So we make use of the additional feature “size”. If the color is red and size is small the fruit is classified as cherry else apple, as shown in Fig. 1.8. In the case of green fruits, if the color is green and size is big then the fruit is classified as watermelon else grape.



**Figure 1.8** Fruit basket categorization based on color and size.

This brings us to some rules for automatic categorization of some fruits in the fruit basket.

**Rule 1:** If the color of the fruit is “RED” and the size of the fruit is “SMALL” then the fruit is cherry.

**Rule 2:** If the color of the fruit is “RED” and the size of the fruit is “BIG” then the fruit is apple.

**Rule 3:** If the color of the fruit is “GREEN” and the size of the fruit is “SMALL” then the fruit is grape.

**Rule 4:** If the color of the fruit is “GREEN” and the size of the fruit is “BIG” then the fruit is watermelon.

A common use case of supervised learning is to utilize historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails.

All supervised learning techniques are a form of *classification* or *regression*.

1. Classification techniques predict discrete responses—for example, whether an email is genuine or spam, or whether a tumor is small, medium, or large. Classification models are trained to classify data into categories. Applications include medical imaging, speech recognition, and credit scoring.
2. Regression techniques predict continuous responses—for example, changes in temperature or fluctuations in electricity demand. Applications include forecasting stock prices, handwriting recognition, and acoustic signal processing.

When we are working on a classification problem, we begin by determining whether the problem is *binary* or *multiclass*. In a binary classification problem, a single training or test item (instance) can only be divided into two classes—for example, if we want to determine whether an email is genuine or spam. In a multiclass classification problem, it can be divided into more than two classes—for example, if we want to train a model to classify an image as a dog, cat, or other animal. Bear in mind that a multiclass classification problem is generally more challenging because it requires a more complex model.

### Example: Creating Algorithms that Can Analyze Works of Art

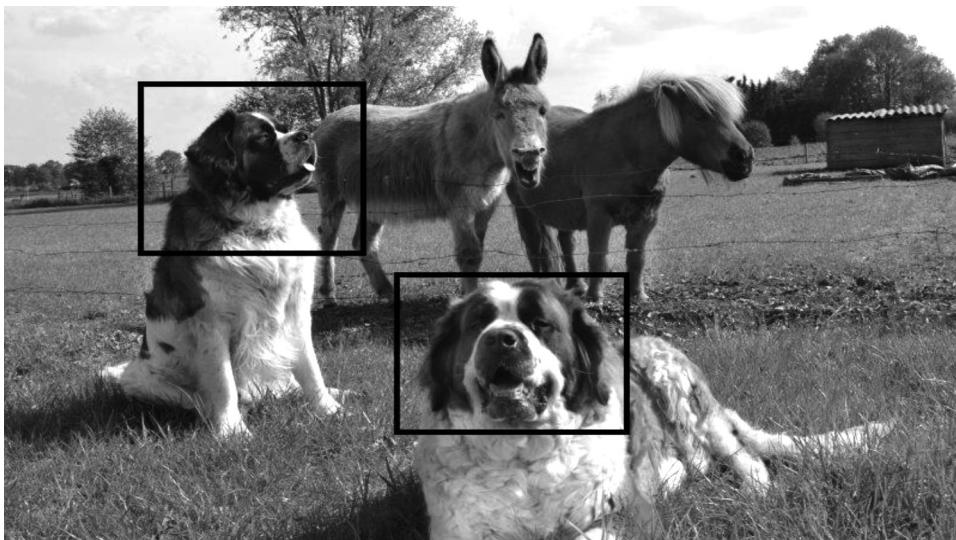
Researchers at the Art and Artificial Intelligence Laboratory at Rutgers University wanted to see whether a computer algorithm could classify paintings by style, genre and artist as easily as a human. They began by identifying visual features for classifying a painting's style. The algorithms they developed classified the styles of paintings in the database with 60% accuracy, outperforming typical non-expert humans. The researchers hypothesized that visual features are useful for style classification (a supervised learning problem). They used classification algorithms trained on Google images to identify specific objects. They tested the algorithms on more than 1,700 paintings from 66 different artists working over a span of 550 years. The algorithm readily identified connected works, including the influence of Diego Velazquez's "Portrait of Pope Innocent X" on Francis Bacon's "Study After Velazquez's Portrait of Pope Innocent X."

#### 1.4.2 Unsupervised Learning

In unsupervised learning, the data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data is more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable. The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a "correct" answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including fraudulent credit card purchases and recommender systems that suggest what products to buy next. As an example, in unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together (Fig. 1.9).

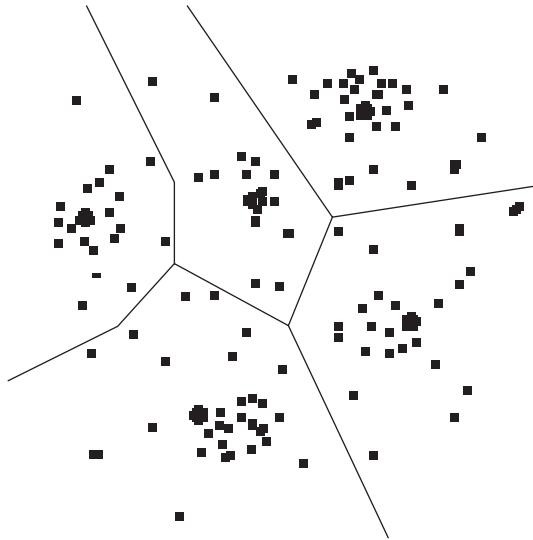


**Figure 1.9** Dogs getting tagged among different animals.

Most unsupervised learning techniques are a form of cluster analysis. In such analysis, data is partitioned into groups based on some measure of similarity or shared characteristic. Clusters are formed so that objects in the same cluster are very similar and objects in different clusters are very distinct. Clustering algorithms fall into two broad groups: (a) *hard clustering*, where each data point belongs to only one cluster and (b) *soft clustering*, where each data point can belong to more than one cluster. Hard or soft clustering techniques can be used if you already know the possible data groupings.

#### Example 1.4: Hard Clustering

Using  $k$ -means clustering to site cell phone towers, a cell phone company wants to know the number and placement of cell phone towers that will provide the most reliable service. For optimal signal reception, the towers must be located within clusters of people. The workflow begins with an initial guess of the number of clusters that will be needed. To evaluate this guess, the engineers compare service with three towers and four towers to see how well they are able to cluster for each scenario (in other words, how well the towers provide service). A phone can only talk to one tower at a time, so this is a hard clustering problem. The team uses  $k$ -means clustering because it treats each observation in the data as an object having a location in space. It finds a partition in which objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible. This is illustrated in Fig. 1.10; the small dots denote the customer's geometric location information and bigger black dots represent the suggested tower location.

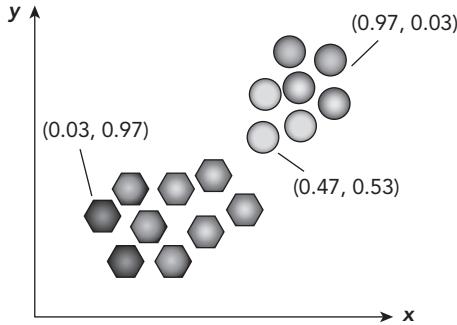


**Figure 1.10** Cell tower sites identified based on customer location.

After running the algorithm, the team can accurately determine the results of partitioning the data into three and four clusters.

### Example 1.5: Soft Clustering

A team of biologists is analyzing gene expression data from microarrays to better understand the genes involved in normal and abnormal cell division. (A gene is said to be “expressed” if it is actively involved in a cellular function such as protein production.) The microarray contains expression data from two tissue samples. The researchers want to compare the samples to determine whether certain patterns of gene expression are implicated in cancer proliferation. After preprocessing the data to remove noise, they cluster the data. Because the same genes can be involved in several biological processes, no single gene is likely to belong to one cluster only. The researchers apply a fuzzy  $c$ -means algorithm to the data. They then visualize the clusters to identify groups of genes that behave in a similar way. Fig. 1.11 shows two clusters represented by hexagons and circles. Each element is associated with a membership which gives the degree of belongingness to each cluster.



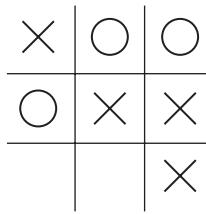
**Figure 1.11** Two clusters represented by hexagons and circles.

### 1.4.3 Reinforcement Learning

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take action in an environment so as to maximize some notion of cumulative reward. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, the field where reinforcement learning methods are studied is called approximate dynamic programming.

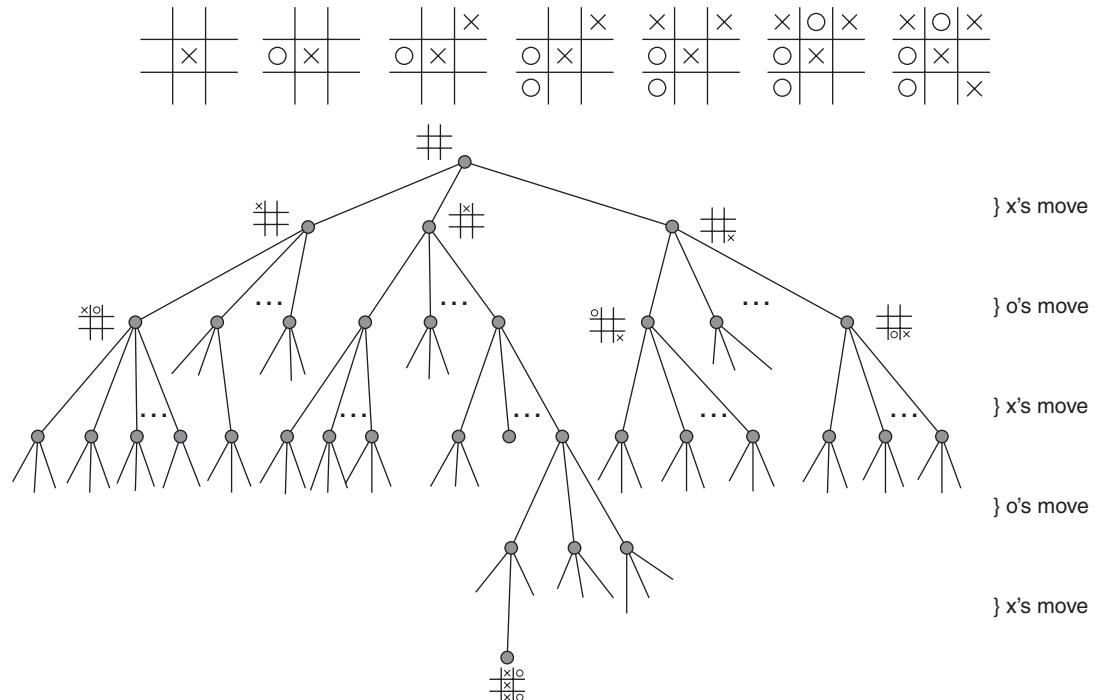
### Example 1.6

Consider the familiar child’s game of tic-tac-toe (Fig. 1.12). Two players take turns playing on a three-by-three board. One player plays X’s and the other O’s. A player wins by placing three marks horizontally, vertically, or diagonally. If the board fills up with neither player getting three in a row, the game is a draw. Because a skilled player can play so as never to lose, let us assume that we are playing against an imperfect player, one whose play is sometimes incorrect and allows us to win. In fact, let us consider draws and losses to be equally bad for us. How might we construct a player that will find the imperfections in its opponent’s play and learn to maximize his chances of winning?



**Figure 1.12** The tic-tac-toe game.

Although this is a simple problem, it cannot readily be solved in a satisfactory way through classical techniques. For example, the classical *minimax* solution from game theory is not applicable here because it assumes a particular way of playing by the opponent. A minimax player would never reach a game state from which it could lose, even if in fact it always wins from that state because of incorrect play by the opponent. Classical optimization methods for sequential decision problems, such as dynamic programming, can compute an optimal solution for any opponent, but require as input a complete specification of that opponent, including the probabilities with which the opponent makes each move in each board state. Let us assume that this information is not available for this problem, as is the case for the vast majority of problems of practical interest. Such information can be estimated from experience, in this case by playing many games against the opponent. About the best one can do with this problem is to learn a model of the opponent's behavior up to some level of confidence, and then apply dynamic programming to compute an optimal solution given the approximate opponent model (Fig. 1.13).



**Figure 1.13** Tree-like structure derived from playing tic-tac-toe.

## Case Study 1: Diagnosing Crop Disease with Machine Learning

An agricultural biotech client needed a way to diagnose crop diseases to help farmer community. Losses due to crop disease, whether caused by fungi, bacteria, or viruses, can be disastrous to the economy and dramatically threaten the global population's access to nutrition.

Crop disease can devastate natural ecosystems, leading to environmental problems such as habitat loss for certain species. Additionally, in extreme cases, diseased crops can produce toxins that result in serious health problems for consumers.

The following are reasons why this crop disease problem was chosen as a case study. Accurate diagnosis was the best way to target the exact solutions necessary to maintain crop health. This diagnosis cannot be done without deep subject matter expertise. Moreover, accurately describing a crop's symptoms to a plant pathologist over the phone is nearly impossible. The solution to this problem was the development of a mobile application that allows farmers to photograph diseased crops and receive accurate diagnostics in real-time. The diagnostic process occurs through the use of advanced image analytics powered by machine learning.

**Advanced image analytics:** The mobile application uses Google's Cloud Vision API to analyze each crop's color, size, texture, and decay patterns, then references these data points against a library of 50,000 images. These images are classified into categories and labeled.

**Machine learning:** Through pattern-recognition machine-learning techniques, the mobile application is trained to diagnose crop disease in near real-time.

**Data engineering:** A reusable platform enables data and model management, along with DevOps support for multi-cloud infrastructure to assist data scientists in performing data cleansing, learning, and service deployment of solutions, utilizing artificial intelligence analytics techniques at scale.

## Case Study 2: Learning to Decode the Immune System to Diagnose Disease

The human immune system is an astonishing diagnostic system, continuously adapting itself to detect any signal of disease in the body. Machine learning can enhance the scientific understanding of human health and provide a foundation for a new generation of precise medical diagnostic and treatment options. Adaptive Biotechnologies, a Seattle based bio-technological firm, along with Microsoft works towards coupling the latest advances in artificial intelligence and machine learning with recent breakthroughs in biotechnology to build a practical technology for mapping and decoding the human immune system. The goal of this research is to "create a universal blood test that reads a person's immune system to detect a wide variety of diseases including infections, cancers, and autoimmune disorders in their earliest stage, when they can be most effectively diagnosed and treated". The impact on human health of such a universal blood test that reads a person's exposure and response to disease would be, in a word, transformational. Put simply, sequencing the immune system can reveal what diseases the body currently is fighting or has ever fought. A blood sample, therefore, contains the key information needed to read what the immune system is currently detecting.

## Summary

- Machine learning is an area in computer science which involves teaching computers to do things naturally by learning through experience.
- According to Tom M Mitchell, *A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E.*
- Machine learning lies at the intersection of computer science, engineering, and statistics and often appears in other disciplines.
- Recommendation systems, recognition systems, classifiers and prediction systems are some of the applications that use machine learning.
- Supervised, unsupervised and reinforcement learning are types of machine learning algorithms.
- Supervised learning is also called learning with a tutor, builds a model with known input and corresponding outputs.
- Unsupervised learning algorithms try to identify patterns from the input of data samples.
- Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

## Multiple-Choice Questions

- Suppose your email program watches your way of marking the emails as spam or not. and based on these observations learns how to better filter spam. The task  $T$  in this case is
  - Classifying emails as spam or not spam
  - Watching you label emails as spam or not spam
  - The number of emails correctly classified as spam/or not spam
  - None of the above
- A toy shop wants to predict the number of Barbie dolls which would be sold the next three months. What type of problem is this?
  - Unsupervised
  - Supervised
  - Reinforcement
  - None of the above
- What kind of learning is required to group a set of unlabeled dolls as Barbie, Frozen, and Others?
  - Unsupervised learning
  - Supervised learning
  - Reinforcement learning
  - None of the above
- A software designed to evaluate whether a customer account is hacked or not is an example of
  - Unsupervised learning
  - Supervised classification
  - Supervised regression
  - None of the above
- Which of the following examples would be classified as an example of unsupervised learning algorithm?
  - Given email labeled as spam/not spam, learn a spam filter.
  - Given a set of news articles found on the web, group them in articles under finance, movies, and political domain.
  - Given a database of customer data, automatically discover market segments and group customers into different market segments.
  - Given a database of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.
    - Both i and iv
    - Both i and iii
    - Both ii and iii
    - Both iii and iv

## Very Short Answer Questions

---

1. Define machine learning as given by Tom Mitchell.
2. What is supervised learning?
3. What is unsupervised learning?
4. How is hard clustering different from soft clustering?
5. Define reinforcement learning with an example.

## Short Answer Questions

---

1. Explain machine learning with respect to designing a chess game by identifying the tasks (T), experience (E), and performance measure (P).
2. Illustrate with a simple example how supervised learning can be used in handling loan defaulters.
3. Organize the following activities under supervised or unsupervised learning. State appropriate reasons for the same.
  - (a) Establishing an appropriate algorithm that would identify research groups working in various domains.
  - (b) Computing the returns on investment when the initial values of investments are given.
  - (c) Identifying whether a breast lump is malignant or benign based on standard data sample taken from University of California Irvine repository.
4. Match the technique to the application.

Credit card fraud detection	Supervised classification
Word frequency of a featured article	Supervised regression
Identifying whether a mail is spam or ham	Unsupervised learning
Predicting the price of a stock	Outlier analysis

## Review Questions

---

1. Define machine learning.
2. Name some applications that require machine learning techniques.
3. What are the different types of machine learning algorithms?
4. How is supervised learning different from unsupervised learning techniques?
5. After a model is built using supervised learning, what do you do with the learned model?
6. What is cluster analysis?
7. How is learning accomplished in reinforcement learning?

## Answers

---

### Multiple-Choice Questions

1. (a)    2. (b)    3. (a)    4. (b)    5. (c)

# 2

# Model and Cost Function

## LEARNING OBJECTIVES

- To represent a hypothesis for a supervised learning model.
- To define and minimize the cost function for a single variable.
- To define and minimize the cost function for a multivariable.

## LEARNING OUTCOMES

- Students will be able to identify the hypothesis for any supervised learning model, along with inputs and outputs.
- Students will be able to compute and minimize a single variable cost function.
- Students will be able to compute and minimize a multivariable cost function.

### 2.1 Introduction

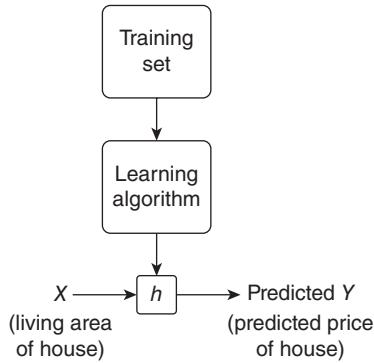
We have already seen in Chapter 1 that machine learning is a field of study that gives the computers the ability to learn without being programmed explicitly. The types of machine learning algorithm can be classified as supervised and unsupervised in a broader perspective. In *supervised learning*, we are given a dataset and already know what our correct output should look like, having an idea that there is a relationship between input and output. In *unsupervised learning*, we can derive the structure for data without knowing the effect of variables. This can be achieved using the concept of clustering the data. This chapter gives a method of mathematical representation of supervised learning problems.

### 2.2 Representation of a Model

Supervised learning algorithms try to map the relationships existing between the input (independent) variable and the output (dependent) variable. This relationship is represented in the form of hypothesis. This can be mathematically represented as a  $(x^{[i]}, y^{[i]})$  training pair, where  $x^{[i]}$  is a set of input variables with  $i$  ranging from 1 to  $m$  and  $y^{[i]}$  is a set of output variables with  $i$  ranging from 1 to  $m$ .

Every set of input variables has a corresponding set of output variables. A collection of such training examples which helps in designing the model is called the *training set*. The superscript  $i$  indicates a single training pair.  $X$  and  $Y$  can be used to denote the space of input and output variables. Another formal way of describing supervised learning problem is to find the relationship existing between the input space  $X$  and the output space  $Y$  and represent it in the form of a hypothesis.

The learning process in supervised learning can be represented Fig. 2.1.



**Figure 2.1** Supervised learning.

Source: Coursera. <https://www.coursera.org/learn/machine-learning/lecture/db3jS/model-representation>

In Fig. 2.1, the independent variable or the input variable is the living area of the house, based on which the price of the house is predicted which is the output variable or the dependent variable. Depending on the type of output variable, the supervised learning problem is classified as either *classification* or *regression problem*. If the output is a continuous value like the price of house then it becomes a regression problem. On the other hand, if the output takes only discrete values like (studio, one room, or two room apartment) then it becomes a classification problem.

## 2.3

### Cost Function Notation for Measuring the Accuracy of a Hypothesis Function

---

The hypothesis helps in providing the mapping between input variables and output variables and is measured using a *cost function*. This function calculates the average difference of all the results of hypothesis with inputs from  $X$  and the actual outputs from  $Y$ . To understand this better, consider housing prices in a metro city in India, with the corresponding carpet area of the housing property as shown in Table 2.1.

**Table 2.1** Property prices in Mumbai (Bandra)

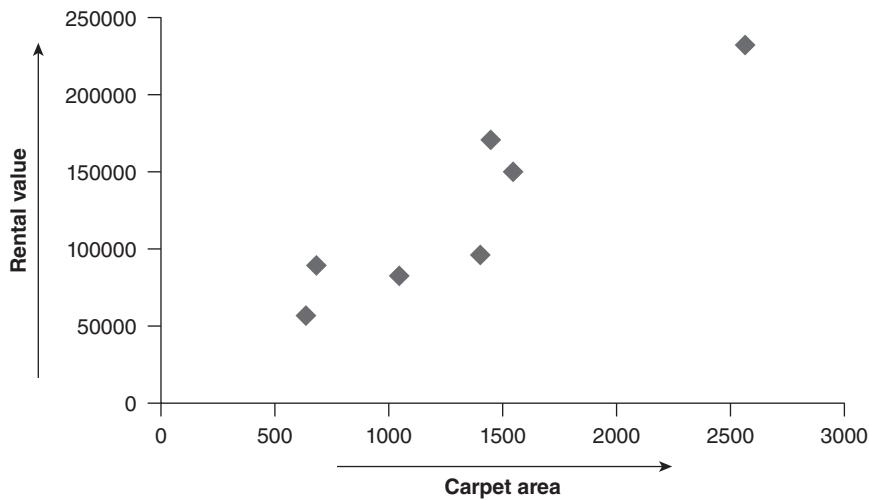
<i>Carpet Area (ft<sup>2</sup>)</i>	<i>No. of Bedrooms</i>	<i>Rental Value</i>
1500	3	1,70,000
2508	4	2,30,000
1600	3	1,50,000
1000	2	85,000
700	1	90,000
1335	2	95,000
650	1	55,000

*Source:* Magicbricks. <https://www.magicbricks.com/Property-Rates-Trends/ALL-RESIDENTIAL-rates-in-Mumbai>

In Table 2.1, the rental value of an apartment is dependent on carpet area and the number of bedrooms. Although for brevity only these two input parameters are chosen, various other parameters such as furnished type, availability of car park, and property developer can be other valid input parameters which determine the rental value of the apartment. Now let us consider one input parameter, say carpet area, on which the rental value of the apartment is dependent. This is a linear regression problem with a single variable. Table 2.2 shows the dataset.

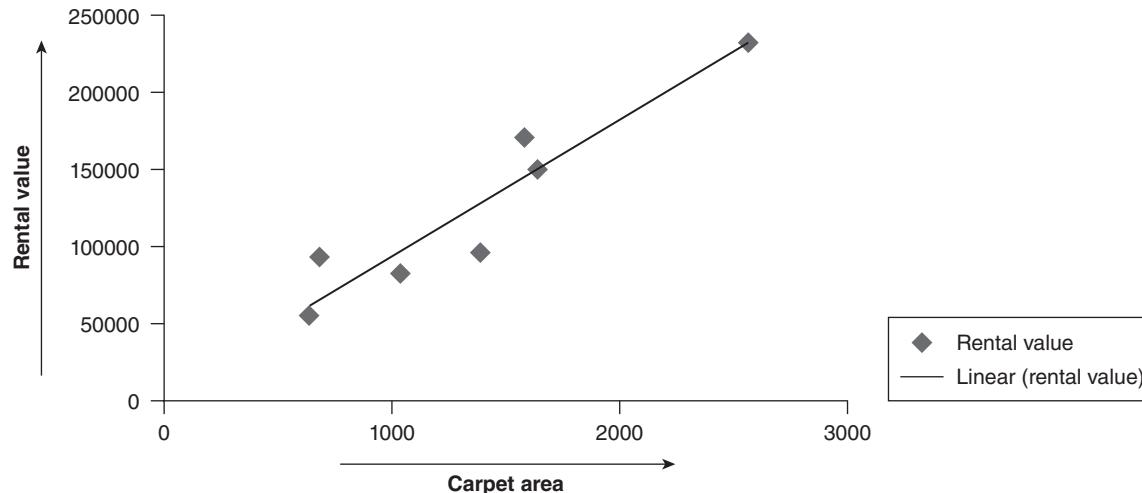
**Table 2.2** Sample dataset of linear regression with a single variable

<i>Carpet Area (ft<sup>2</sup>)</i>	<i>Rental Value</i>
1500	1,70,000
2508	2,30,000
1600	1,50,000
1000	85,000
700	90,000
1335	95,000
650	55,000



**Figure 2.2** Plot of carpet area (ft<sup>2</sup>) versus rental value.

In accordance to Table 2.1 showing property prices in Mumbai, Fig. 2.2 shows the plot of carpet area (ft<sup>2</sup>) versus rental value. When a regression line is plotted, given any carpet area in square feet, the approximate rent has to be predicted. For that, a regression line is plotted which passes through a maximum number of points. This is shown by a regression line plotted in Fig. 2.2. Figure 2.3 shows the regression line plotted that passes through most of the points.



**Figure 2.3** Linear regression line plotted for carpet area versus rental value.

The cost function is then computed as the average difference of all the results of the hypothesis with inputs ( $x$ ) and the actual output ( $y$ ).

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}_i - y_i)^2 \quad (2.1)$$

where  $\hat{y}_i = h_\theta(x)$ ,  $\hat{y}$  is the estimated value of the output for the input  $x$ .

The hypothesis should be so selected that the average difference between the estimated value and the actual value is minimized. This minimization function is given by the cost function, given by Eq. (2.1). This cost function is also known as *squared error function* or *mean squared error function*. The idea behind the cost function is that we have to choose the value of  $\theta_0$  and  $\theta_1$  so that  $h_\theta(x)$  is close to  $y$  for our training examples  $(x, y)$ .

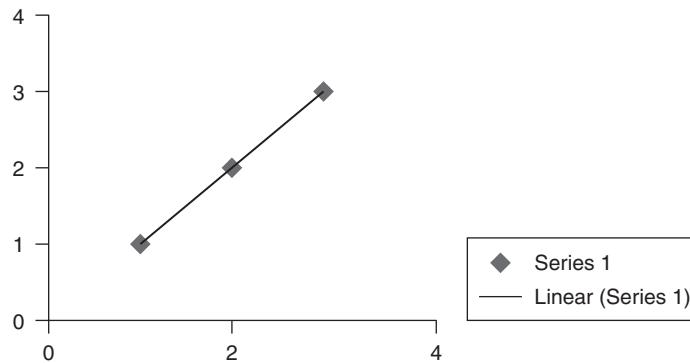
## 2.4 Measuring Accuracy of a Hypothesis Function

Visualizing the training dataset in a scatter plot on the  $x-y$  plane, we try to make a straight-line pass through the scattered data points. Since the straight line would not be able to pass through all the data points, our objective is to get the best possible line. The best possible line will be such that the average squared vertical distances of the scattered points from the line will be the least. The ideal situation being the straight line passing through all the points in the scatter plot so that the cost function is zero. But this is just an ideal situation. Nearing the ideal situation can be arrived at by using minimization of cost function. The parameters  $\theta_0$  and  $\theta_1$  can be adjusted so that we arrive at the minimum cost function.

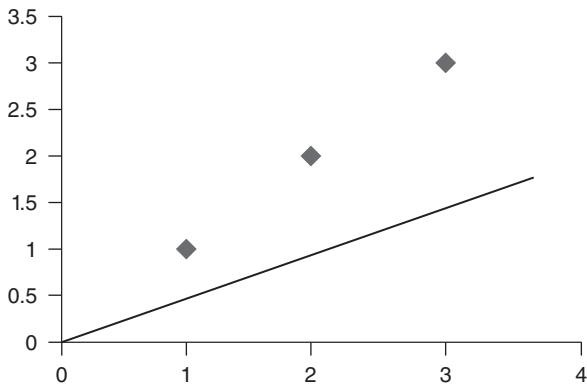
## 2.5 Minimizing the Cost Function for a Single-Variable Function

In Fig. 2.4 the value of  $\theta_0$  is 0 and  $\theta_1$  is 1. In this case, the squared error between the estimated and actual value of  $y$  is 0, that is

$$J_{(0,0)} = 0$$



**Figure 2.4** Ideal situation of the cost function.



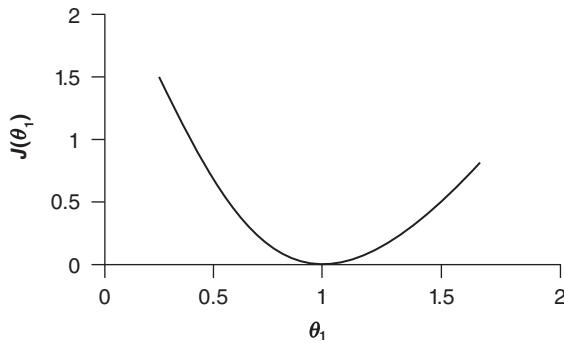
**Figure 2.5** Cost function with slope value of 0.5.

In Fig. 2.5, when the slope is 0.5, the average difference between the actual and the estimated value is given by

$$J_{(0,0.5)} = \frac{1}{6} \times ((1 - 0.5)^2 + (2 - 1)^2 + (3 - 1.5)^2) = \frac{1}{6} \times (0.25 + 1 + 2.25) = 0.58$$

Likewise, plotting the cost function for several values of  $\theta_1$  we get the graph given in Fig. 2.6. The goal in this case is to minimize the cost function. The minimum value of cost function in this case is 0 when  $\theta_1$  is 1.

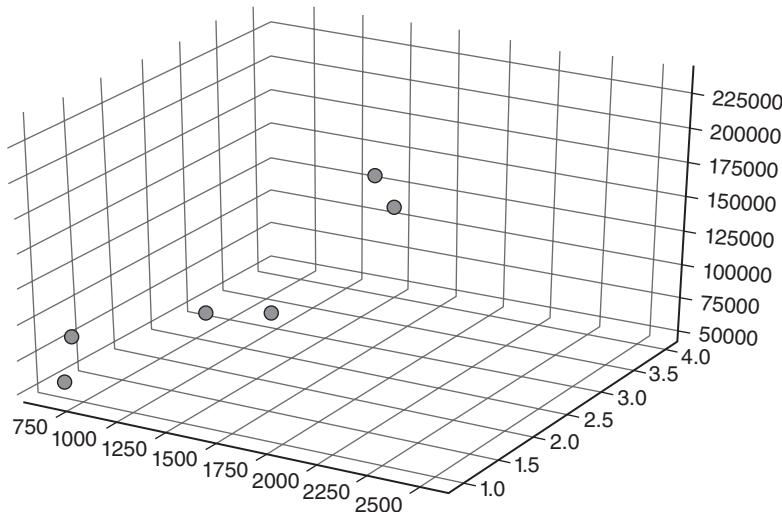
In the preceding example of predicting house prices given the carpet area, a valid assumption is made that the regression line passes through the origin and there exists one dependent variable for every independent variable. Based on this assumption the plot of cost function is an elliptical curve shown in Fig. 2.6. But in reality, there may exist more than one independent variable, and the regression line need not necessarily pass through the origin. So we end up with a contour plot which helps in understanding the minimization function. This would be dealt with in following section.



**Figure 2.6** Plot of cost function for various values of  $\theta_1$ .

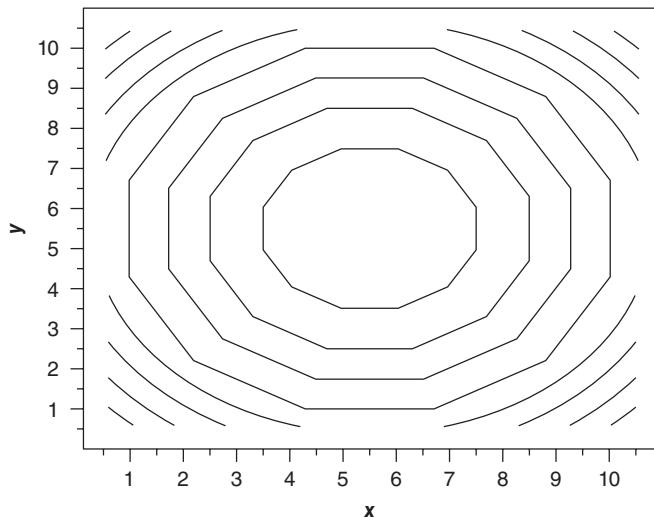
**2.6****Minimizing the Cost Function for a Two-Variable Function**

In Table 2.1, the independent variables are carpet area and number of bedrooms based on which the rental value can be predicted. This is an example of two-variable functions. Figure 2.7 shows the plot of Table 2.1.



**Figure 2.7** Plot of property prices depending on carpet area and number of bedrooms.

For finding the cost function in the case of two variables we should first understand the concept of a contour plot. A contour plot is a graphical technique for representing a three-dimensional surface by plotting constant  $z$  slices, called contours, on a two-dimensional format (Fig. 2.8). That is, given a value for  $z$ , lines are drawn for connecting the  $(x, y)$  coordinates where the  $z$  value occurs.



**Figure 2.8** A sample contour plot.

The contour plot is formed by,

Vertical axis: independent variable 2

Horizontal axis: independent variable 1

Points are the dependent values corresponding to the independent variables.

The independent variables are usually restricted to a regular grid. Using contour plot we can find out the change in  $Z$  as a function of  $X$  and  $Y$ . Contour plots are used for minimization of cost function when we have  $J(\theta_0, \theta_1)$  with different values of  $\theta_0$  and  $\theta_1$ . Along each circle the cost function remains the same for different values of  $\theta_0$  and  $\theta_1$ .

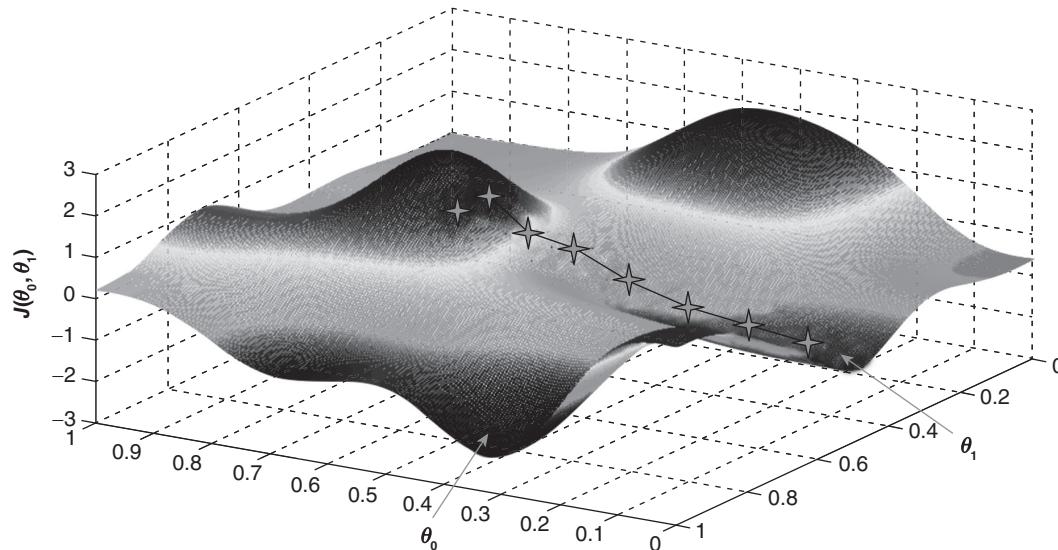
## 2.7

## Role of Gradient Function in Minimizing a Cost Function

So far, we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That is where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields  $\theta_0$  and  $\theta_1$  (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing  $x$  and  $y$ , but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put  $\theta_0$  on the  $x$ -axis and  $\theta_1$  on the  $y$ -axis, with the cost function on the vertical  $z$ -axis. The points on our graph are the result of the cost function using our hypothesis with those specific theta parameters. The graph shown in Fig. 2.9 depicts such a set-up.



**Figure 2.9** Graph of gradient descent for two variables.

We achieve the desired result when the cost function is at the very bottom of the pits in the graph, that is, when its value is at the minimum. The arrows in Fig. 2.9 show the minimum points.

To get this result we take the derivative (the tangential line to a function) of the cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move onwards. We make steps down the cost function in the direction of the steepest descent. The size of each step is determined

by the parameter  $\alpha$ , which is called the *learning rate*. For example, the distance between each “star” in Fig. 2.9 represents a step determined by our parameter  $\alpha$ . A smaller  $\alpha$  would result in a smaller step and a larger  $\alpha$  would result in a larger step. The direction in which the step is taken is determined by the partial derivative of  $J(\theta_1, \theta_2)$ . Depending on where one starts on the graph, one could end up at different points. Figure 2.9 shows us two different starting points that end up in two different places.

## Summary

- Supervised learning algorithms try to map the relationships existing between the input (independent) variable and the output (dependent) variable. This relationship is represented in the form of hypothesis.
- The hypothesis helps in providing the mapping between the input variables and output variables and is measured using a cost function.
- Cost function calculates the average difference of all the results of hypothesis with inputs from  $X$  and actual outputs from  $Y$ .
- When the inputs and the corresponding outputs are plotted in a graph, a regression line is drawn which passes through most of the points.
- The sum of the differences between the actual output and the estimated output is the error or the cost function and this is to be minimized.
- The idea behind the cost function is that we have to choose the value of  $\theta_0$  and  $\theta_1$  so that  $h_\theta(x)$  is close to  $y$  for our training examples  $(x, y)$ .
- When there is more than one input variable affecting the output, we get a contour plot.
- Since it is not technically possible to minimize cost function using trial and error methods, a gradient descent algorithm is used.

## Multiple-Choice Questions

- Supervised learning is also called as
    - Learning with a teacher
    - Learning by experience
    - Learning by intuition
    - None of the above
  - The cost function is also known as
    - Squared error function
    - Mean squared error
    - Least square estimator
    - Both (a) and (b)
  - When we have two or more input variables and an output variable then the cost function is represented by a
    - Convex curve
    - Concave curve
- (c) Contour plot  
(d) None of the above
- The parameters in the cost function are computed using
    - Gradient descent method
    - Gradient ascent method
    - None of the above
    - Both (a) and (b)
  - If the output of a supervised model is a categorical output then it is a
    - Regression
    - Classifier
    - Prediction
    - None of the above

## Very Short Answer Questions

1. What is a training set?
2. Why is a hypothesis required?
3. Write the notation of cost function. Explain the variables used in it.
4. What is the role of gradient function in minimizing the cost function?
5. What is the significance of learning rate in steepest descent algorithm?

## Short Answer Questions

1. Explain the formulation of cost function.
2. Illustrate with a suitable example the cost function when the age, height, and gender of a person would affect his/her weight.
3. Among the three given best-fit values in the table given below, which do you think would be the best. Justify your answer.

x	y	best_fit_1	best_fit_2	best_fit_3
1.00	1.00	0.50	1.00	1.50
2.00	2.50	1.00	2.00	3.00
3.00	3.50	1.50	3.00	4.00

## Review Questions

1. What is a supervised learning model?
2. How do you represent a model in supervised learning model?
3. How do you arrive at the cost function for a single variable input?
4. Why is the cost function also called mean squared error function?
5. Why is a contour plot used?

## Answers

### Multiple-Choice Questions

1. (a)    2. (d)    3. (c)    4. (a)    5. (b)

# 3

# Basics of Vectors and Matrices

## LEARNING OBJECTIVES

- To introduce the concept of matrices.
- To perform matrix operations.
- To calculate the determinant of a matrix.
- To compute the inverse of a matrix.

## LEARNING OUTCOMES

- Students will be able to represent data in the form of matrices.
- Students will be able to perform matrix operations like addition, subtraction and multiplication.
- Students will be able to compute the determinant of a matrix.
- Students will be able to determine the inverse of a matrix.

### 3.1 Introduction

A matrix is any rectangular array of numbers. If the array has  $n$  rows and  $m$  columns, then it is an  $n \times m$  matrix. The numbers  $n$  and  $m$  are called the dimensions of the matrix. We usually denote matrices with capital letters, like  $A$ ,  $B$  and  $C$ , although we sometimes use lower case letters for one-dimensional matrices (e.g.,  $1 \times m$  or  $n \times 1$  matrices). One-dimensional matrices are called vectors:  $n \times 1$  matrix is called a row vector and  $1 \times m$  matrix is called a column vector.

### 3.2 Notations

The notation  $A_{ij}$  is used to refer to the number in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

For example, in the matrix below,  $A_{12}$  refers to the first row and second column. This means  $A_{12} = 3$ .

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

The dimensions or order of a matrix gives the number of rows followed by the number of columns. The order of a matrix with 3 rows and 2 columns is  $3 \times 2$  or 3 by 2.

### 3.3 Types of Matrices

There are several types of matrices – row matrix, column matrix, zero matrix, square matrix, diagonal matrix, scalar matrix, upper triangular matrix, lower triangular matrix, unit matrix. It is possible for a matrix to belong to more than one type. Below are the different types of matrices along with their examples.

- Row matrix:** A row matrix is a matrix with only one row.

Example:

$$A = [a \ b \ c]$$

- Column matrix:** A column matrix is a matrix with only one column. It is also called vector matrix.

Example:

$$B = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- Zero matrix:** A zero matrix or a null matrix is a matrix whose elements are all zero.

Example:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- Square matrix:** A square matrix is a matrix with an equal number of rows and columns. For example,  $T$  is a square matrix of order  $2 \times 2$  and  $V$  is a square matrix of order  $3 \times 3$ .

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, V = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

- Diagonal matrix:** A diagonal matrix is a square matrix that has all its elements zero except for those in the diagonal from top left to bottom right, which is known as the leading diagonal of the matrix.

Example:

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$

- Scalar matrix:** A scalar matrix is a diagonal matrix where all the diagonal elements are equal. In such a matrix,  $A_{11} = A_{22} = A_{33} = A_{44} = A_{nn}$ .

Example:

$$E = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}$$

- Upper triangular matrix:** An upper triangular matrix is a square matrix where all the elements located below the diagonal are zero.

Example:

$$F = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

- 8. Lower triangular matrix:** A lower triangular matrix is a square matrix where all the elements located above the diagonal are zero.

Example:

$$G = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix}$$

- 9. Unit matrix:** A unit matrix is a diagonal matrix whose elements in the diagonal are all one.

Example:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## 3.4 Matrix Operations

There are five main matrix operations. These are discussed below.

### 3.4.1 Matrix Addition

Matrices can be added. It has to be ensured that the matrices to be added are compatible with their indices.

Let  $A$  and  $B$  be  $m \times n$  matrices.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$$

The sum of  $A$  and  $B$ , denoted as  $A + B$ , is

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}$$

$$\text{For example, if } A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}, B = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 2 & 2 \\ -1 & 0 & 4 \end{bmatrix}$$

then

$$A + B = \begin{bmatrix} 3 & 6 & 9 \\ 0 & 4 & 3 \\ 4 & 5 & 9 \end{bmatrix}$$

### 3.4.2 Matrix Scalar Multiplication

Let  $A$  be an  $m \times n$  matrix and let  $k$  be a scalar. The scalar multiplication of  $k$  and  $A$ , denoted  $kA$ , is

$$kA = \begin{bmatrix} ka_{11} & ka_{12} & \dots & ka_{1n} \\ ka_{21} & ka_{22} & \dots & ka_{2n} \\ \dots & \dots & \dots & \dots \\ ka_{m1} & ka_{m2} & \dots & ka_{mn} \end{bmatrix}$$

For example, if  $A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}$ , then  $5 \times A$  is

$$5A = \begin{bmatrix} 5 & 10 & 15 \\ -5 & 10 & 5 \\ 25 & 25 & 25 \end{bmatrix}$$

### 3.4.3 Negative of a Matrix

The negative of a matrix is simple. We have to take the negative of individual elements, as illustrated below:

$$\text{If } A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}, \text{ then}$$

$$-A = -\begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -3 \\ 1 & -2 & -1 \\ -5 & -5 & -5 \end{bmatrix}$$

### 3.4.4 Matrix Subtraction

To subtract two matrices, we need to subtract the numbers in the matching positions. Just as in matrix addition, it has to be ensured that the matrices are compatible with their indices.

Let  $A$  and  $B$  be  $m \times n$  matrices. The difference between  $A$  and  $B$ , denoted  $A - B$ , is

$$A - B = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} & \dots & a_{1n} - b_{1n} \\ a_{21} - b_{21} & a_{22} - b_{22} & \dots & a_{2n} - b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & \dots & a_{mn} - b_{mn} \end{bmatrix}$$

For example, if  $A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}$ ,  $B = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 2 & 2 \\ -1 & 0 & 4 \end{bmatrix}$

$$\text{then } A - B = \begin{bmatrix} -1 & -2 & -3 \\ -2 & 0 & -1 \\ 6 & 5 & 1 \end{bmatrix}$$

### 3.4.5 Matrix Multiplication

You can only multiply two matrices if their dimensions are compatible, which means the number of columns in the first matrix is the same as the number of rows in the second matrix.

If  $A = [a_{ij}]$  is an  $m \times n$  matrix and  $B = [b_{ij}]$  is an  $n \times p$  matrix, the product  $AB$  is an  $m \times p$  matrix.

$$A \times B = [c_{ij}]$$

where  $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$ .

Let us take the following problem, multiplying a  $2 \times 3$  matrix with a  $3 \times 2$  matrix, to get a  $2 \times 2$  matrix as the product. The entries of the product matrix are called  $c_{ij}$  where they belong to  $i^{\text{th}}$  row and  $j^{\text{th}}$  column.

Example: If  $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$ ,  $B = \begin{bmatrix} 3 & 5 \\ -1 & 0 \\ 2 & -1 \end{bmatrix}$  then

$$A \times B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} 3 & 5 \\ -1 & 0 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

To get  $c_{11}$ , multiply row 1 of the first matrix  $A$  by column 1 of matrix  $B$ .

$$c_{11} = [1 \ 0 \ 1] \times \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} = 1(3) + 0(-1) + 1(2) = 5$$

To get  $c_{12}$ , multiply row 1 of the first matrix  $A$  by column 2 of matrix  $B$ .

$$c_{12} = [1 \ 0 \ 1] \times \begin{bmatrix} 5 \\ 0 \\ -1 \end{bmatrix} = 1(5) + 0(0) + 1(-1) = 4$$

To get  $c_{21}$ , multiply row 2 of the first matrix  $A$  by column 1 of matrix  $B$ .

$$c_{21} = [0 \ 1 \ 2] \times \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} = 0(3) + 1(-1) + 2(2) = 3$$

To get  $c_{22}$ , multiply row 2 of the first matrix  $A$  by column 2 of matrix  $B$ .

$$c_{22} = [0 \ 1 \ 2] \times \begin{bmatrix} 5 \\ 0 \\ -1 \end{bmatrix} = 0(5) + 1(0) + 2(-1) = -2$$

Writing the product matrix, we get

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 3 & -2 \end{bmatrix}$$

**Note:**

1. You can only multiply two matrices  $A \times B$  if  $\dim(A) = m \times n$  and  $\dim(B) = n \times p$ .  $A \times B$  is not the same as  $B \times A$ .
2. The resultant matrix has dimensions  $\dim(C) = m \times p$ , so it is not the same size as the starting matrices (unless you are multiplying square matrices).
3. If  $A \times B$  is possible,  $B \times A$  is only possible if  $m = p$
4. However,  $A \times (B \times C) = (A \times B) \times C$ , and  $A \times (B + C) = A \times B + A \times C$

**3.4.6 Matrix Transpose**

To transpose a matrix, we have to swap its rows and columns. We put a “ $T$ ” in the top right-hand corner to indicate transpose.

$$\text{If } A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}, \text{ then}$$

$$A^T = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 2 & 1 \\ 5 & 5 & 5 \end{bmatrix}^T = \begin{bmatrix} 1 & -1 & 5 \\ 2 & 2 & 5 \\ 3 & 1 & 5 \end{bmatrix}$$

**3.5 Determinant of a Matrix**

---

The determinant of a matrix is a special number that can be calculated from a square matrix. The determinant tells us things about the matrix that is useful in systems of linear equations, helps us find the inverse of a matrix, and is useful in calculus and more. The symbol for determinant is two vertical lines on either side of the matrix name. For example, the determinant of matrix  $A$  is  $|A|$ . For calculating the determinant, the matrix must be square (i.e., have the same number of rows as columns).

Calculation of the determinant for different square matrices is shown below.

1.  **$2 \times 2$  matrix:** Let  $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ . The determinant is

$$|A| = ad - bc$$

Example: If  $B = \begin{bmatrix} 4 & 6 \\ 3 & 8 \end{bmatrix}$ , then

$$|B| = 4 \times 8 - 6 \times 3 = 32 - 18 = 14$$

2.  **$3 \times 3$  matrix:** Let  $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ . The determinant is

$$|A| = a(ei - fh) - b(di - fg) + c(dh - eg)$$

It may look complicated, but there is a pattern.

$$\begin{bmatrix} a & \times \\ e & f \\ h & i \end{bmatrix} - \begin{bmatrix} b & \times \\ d & f \\ g & i \end{bmatrix} + \begin{bmatrix} c & \times \\ d & e \\ g & h \end{bmatrix}$$

To work out the determinant of a  $3 \times 3$  matrix:

1. Multiply  $a$  by the determinant of the  $2 \times 2$  matrix that is not in  $a$ 's row or column.
2. Likewise for  $b$  and  $c$ .
3. Sum them up, but remember the minus in front of  $b$ .

Thus,

$$A = a \cdot \begin{vmatrix} e & f \\ b & i \end{vmatrix} - b \cdot \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \cdot \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

Example: If  $C = \begin{bmatrix} 6 & 1 & 1 \\ 4 & -2 & 5 \\ 2 & 8 & 7 \end{bmatrix}$

$$\begin{aligned} |C| &= 6 \times (-2 \times 7 - 5 \times 8) - 1 \times (4 \times 7 - 5 \times 2) + 1 \times (4 \times 8 - (-2 \times 2)) \\ &= 6 \times (-54) - 1 \times (18) + 1 \times (36) = -306 \end{aligned}$$

## 3.6 Inverse of a Matrix

The inverse of a matrix is calculated by the following steps:

1. Calculate the matrix of minors.
2. Turn it into the matrix of cofactors.
3. Adjugate the matrix of cofactors to get the adjoint matrix.
4. Multiply that by  $1/\text{determinant}$ .

It is best explained with an example. Each and every step of calculating the matrix of minors, matrix of cofactors, and adjugate of matrix of cofactors are computed in the example.

Example:

$$A = \begin{bmatrix} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$$

### Step 1: Calculate the matrix of minors

The first step is to create a matrix of minors. This step has the most calculations. For each element of the matrix:

1. Ignore the values on the current row and column
2. Calculate the determinant of the remaining values
3. Put those determinants into a matrix (the matrix of minors)

$$a_{11} = \begin{bmatrix} \bullet & \times & \times \\ \times & 0 & -2 \\ \times & 1 & 1 \end{bmatrix} = 2; a_{12} = \begin{bmatrix} \times & \bullet & \times \\ 2 & \times & -2 \\ 0 & \times & 1 \end{bmatrix} = 2; a_{13} = \begin{bmatrix} \times & \times & \bullet \\ 2 & 0 & \times \\ 0 & 1 & \times \end{bmatrix} = 2$$

$$a_{21} = \begin{bmatrix} \times & 0 & 2 \\ \bullet & \times & \times \\ \times & 1 & 1 \end{bmatrix} = -2; a_{22} = \begin{bmatrix} 3 & \times & 2 \\ \times & \bullet & \times \\ 0 & \times & 1 \end{bmatrix} = 3; a_{23} = \begin{bmatrix} 3 & 0 & \times \\ \times & \times & \bullet \\ 0 & 1 & \times \end{bmatrix} = 3$$

$$a_{31} = \begin{bmatrix} \times & 0 & 2 \\ \times & 0 & -2 \\ \bullet & \times & \times \end{bmatrix} = 0; a_{32} = \begin{bmatrix} 3 & \times & 2 \\ 2 & \times & -2 \\ \times & \bullet & \times \end{bmatrix} = -10; a_{33} = \begin{bmatrix} 3 & 0 & \times \\ 2 & 0 & \times \\ \times & \times & \bullet \end{bmatrix} = 0$$

$$\text{Matrix of minors} = \begin{bmatrix} 2 & 2 & 2 \\ -2 & 3 & 3 \\ 0 & -10 & 0 \end{bmatrix}$$

**Step 2:** Matrix of cofactors

To determine the matrix of cofactors, we have to apply a “checkerboard” of pluses and minuses to the matrix of minors. In other words, we need to change the signs of alternate cells, like below:

$$\begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}$$

Applying this checkerboard to the matrix of minors, we get

$$\text{Matrix of cofactors} = \begin{bmatrix} 2 & 2 & 2 \\ -2 & 3 & 3 \\ 0 & -10 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix} = \begin{bmatrix} 2 & -2 & 2 \\ 2 & 3 & -3 \\ 0 & 10 & 0 \end{bmatrix}$$

**Step 3:** Adjugate (also called adjoint matrix)

Now transpose all elements of the matrix of cofactors.

$$\begin{bmatrix} 2 & -2 & 2 \\ 2 & 3 & -3 \\ 0 & 10 & 0 \end{bmatrix}^T = \begin{bmatrix} 2 & 2 & 0 \\ -2 & 3 & 10 \\ 2 & -3 & 0 \end{bmatrix}$$

**Step 4:** Multiply by 1/determinant

$$\text{Determinant} = 3 \times 2 - 0 \times 2 + 2 \times 2 = 10$$

$$A^{-1} = \frac{1}{10} \begin{bmatrix} 2 & 2 & 0 \\ -2 & 3 & 10 \\ 2 & -3 & 0 \end{bmatrix} = \begin{bmatrix} 0.2 & 0.2 & 0 \\ -0.2 & 0.3 & 1 \\ 0.2 & -0.3 & 0 \end{bmatrix}$$

## Summary

- A matrix is any rectangular array of numbers. If the array has  $n$  rows and  $m$  columns, then it is an  $n \times m$  matrix.
- There are different types of matrices: row matrix, column matrix, zero matrix, square matrix, diagonal matrix, scalar matrix, upper triangular matrix, lower triangular matrix and unit matrix. It is possible for a matrix to belong to more than one type.
- Matrices can be added and subtracted. While performing these operations, it has to be ensured that the matrices are compatible with their indices. The resultant matrix is the sum of elements of the individual matrices in case of matrix addition or the difference between the different elements in case of matrix subtraction.
- Matrices can be multiplied if their dimensions are compatible, which means the number of columns in the first matrix should be the same as the number of rows in the second matrix.
- To transpose a matrix, we have to swap the rows and columns.
- The determinant of a matrix is a special number that can be calculated only for square matrices. The determinant tells us things about the matrix that is useful in systems of linear equations. It helps us find the inverse of a matrix and is useful in calculus.

## Multiple-Choice Questions

1. Transpose of a rectangular matrix is a
  - Rectangular matrix
  - Diagonal matrix
  - Square matrix
  - Scalar matrix
2. Transpose of a column matrix is
  - Zero matrix
  - Diagonal matrix
  - Column matrix
  - Row matrix
3. Two matrices  $A$  and  $B$  are multiplied to get  $AB$  if
  - Both are rectangular
  - Both have same order
  - Number of columns of  $A$  is equal to columns of  $B$
  - Number of columns of  $A$  is equal to number of rows of  $B$
4. Two matrices  $A$  and  $B$  are added if
  - Both are rectangular
  - Both have same order
  - Number of columns of  $A$  is equal to number of columns of  $B$
  - Number of rows of  $A$  is equal to number of columns of  $B$
5. Transpose of a row matrix is
  - Zero matrix
  - Diagonal matrix
  - Column matrix
  - Row matrix

**Very Short Answer Questions**

1. How is an upper triangular matrix different from lower triangular matrix?
2.  $A(3 \times 2)$  and  $B(3 \times 4)$  are two matrices. What is the sum of  $A + B$ ?
3.  $A = \begin{bmatrix} 3 & -2 \\ 4 & 1 \end{bmatrix}$ . Find  $2A^2$ .
4. If matrix  $A$  is  $(4 \times 3)$  and matrix  $B$  is  $(3 \times 7)$ , what is the dimension of  $(A \times B)^T$ ?
5. How do you compute the inverse of a matrix?

**Review Questions**

1. Identify the type of matrix.
    - (a)  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$ ; (b)  $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 7 & -5 \\ 0 & 0 & -4 \end{bmatrix}$ ;
    - (c)  $\begin{bmatrix} 1 & 0 & 0 \\ -4 & 7 & 0 \\ 12 & 5 & 3 \end{bmatrix}$ ; (d)  $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ ;
    - (e)  $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  2. Perform addition and subtraction of two matrices given below.
- $A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 2 \end{bmatrix}$  and  $B = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 3 \end{bmatrix}$
3. Multiply matrices  $A$  and  $B$ .
  4. Compute the determinant of the following matrices.
  5. Find the inverse of the matrix:
- $$A = \begin{bmatrix} 2 & -3 \\ 4 & -7 \end{bmatrix}$$
- $$A^{-1} = \frac{1}{|A|} \begin{bmatrix} -7 & 3 \\ -4 & 2 \end{bmatrix} = \frac{1}{-2} \begin{bmatrix} -7 & 3 \\ -4 & 2 \end{bmatrix} = \begin{bmatrix} 3.5 & -1.5 \\ 2 & -1 \end{bmatrix}$$

**Answers****Multiple-Choice Questions**

1. (a)    2. (d)    3. (d)    4. (b)    5. (c)

# 4

# Basics of Python

## LEARNING OBJECTIVES

- To make students understand the basics of Python. To understand how to plot graphs using Matplotlib.
- To introduce the concept of NumPy and how to use it to deal with arrays and matrices.
- To learn how to use Pandas to deal with DataFrames for implementing machine learning algorithms.
- To introduce the concept of scikit-learn.

## LEARNING OUTCOMES

- Students will be able to install Python, Anaconda, and Jupyter Notebook.
- Students will be able to work with Python lists, dictionary, and tuples.
- Students will be able to perform basic operations in Python.
- Students will be able to use Python loops and conditional statements.
- Students will be able to plot graphs using Matplotlib.
- Students will be able to work and manipulate Pandas DataFrames.

### 4.1 Introduction

Python is an interactive high-level programming language. It was created by Guido van Rossum during 1985–1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python was inspired by another programming language ABC, which was not successful. ABC was intended for a casual user and was originally implemented by Guido van Rossum. The research on ABC started before 1980. It was initially developed to be a user-friendly language with programming productivity. The first version of ABC was realized in 1987, and by the end of 1989 Guido started developing Python. This was because his research group needed an easy-to-use scripting language to distribute the OS. He chose the name Python because he was a fan of the English TV Series “Monty Python’s Flying Circus”. The language was developed at the Centrum Voor Wiskunde en informatica (CWI) in Netherlands.

The history of Python dates back to 1991. Its popularity was mainly because it was simple, understandable, easy-to-learn, portable, and interpreted. It is interpreted and interactive because Python can be processed at runtime by the interpreter and interacted directly. Being interpreted, Python does not require separate compilation and execution steps. All this conversion is done internally which makes it easier to run the programs. It supports object-oriented style of programming by encapsulating code within the objects.

#### 4.1.1 Versions of Python

The Python programming language is continually being updated and improved upon. There are different versions of Python.

1. **Python 1.0:** This was the initial version of Python. The major features in this version were functional programming tools like Lambda, Map, Filter, and Reduce. It had inspired keywords and had support for complex numbers. Till version Python 1.4 was released, an initiative Computer Programming for Everyone (CP4E) was started making Python accessible to casual users. Python 1.6 is the latest in the series of Python 1.X, which was released in September 2000.
2. **Python 2.0:** This version introduced list comprehensions, with statements, and warnings. The latest Python 2.X is Python version 2.7 which was released in July 2010.
3. **Python 3.0:** Python 3 emphasized on removing duplicative constructs and modules and is a multi-paradigm language. The first version of Python 3.0 was released in December 2008. The latest version, Python 3.7, was released in June 2018.

#### 4.1.2 Features of Python

Python is a very user-friendly language and was designed with the idea of making it accessible to the common man. Following are some of the salient features of Python, which makes it very popular now-a-days.

1. **Simple and easy-to-learn:** It is as easy as reading English but for the fact the English is “strict”. It concentrates much more on problem-solving rather than syntax.
2. **Free and open source:** Python’s source code is fairly easy-to-maintain and it is maintained by the community. Hence, sharing knowledge and problem resolution are the advantages.
3. **Broad standard library:** Python’s bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh. All Python programs can run on any of these platforms without making any changes at all.
4. **Interactive mode:** Python has support for an interactive mode, which allows interactive testing and debugging of snippets of code.
5. **Interpreted:** The code in Python is interpreted in every line unlike in C and C++, which are compiled programming languages.
6. **GUI programming and database compatibility:** Python supports Graphical User Interface (GUI) applications that can be created and ported to many system calls, libraries, and windows systems such as Windows MFC, Macintosh, and the X Window system of Unix. In addition, Python provides interfaces to all major commercial databases.
7. **Scalable:** Python provides better structure and support for large programs than shell scripting.
8. **Multi-paradigm support:** Python supports functional, procedural, and object-oriented programming methods.
9. **Dynamic data type:** Python is an example of dynamic type programming language and supports a wide variety of data types. The type checking is also done dynamically.
10. **Automatic garbage collection:** Unlike in C and C++, garbage collection in Python is taken care of as follows: Reference counting automatically deallocates objects when its reference becomes zero. But since this becomes a computational extensive job, the garbage collection feature of Python, unlike in C and C++, sets threshold and schedules garbage collection based on object allocations and deallocations. In Python gc can be imported and `get_threshold()` method is used to find the threshold that is to be set.
11. **Integration language:** Python is used as an integration language with C, C++, and CORBA. For this, some extension modules are required. SWIG is a software developmental tool used for this purpose.

## 4.2 Installing Python

The latest version of Python is available for download for Windows and Mac OS from the site <https://www.python.org/downloads/>. The latest releases for each of the OS are available from this site.

Python would be already installed in most of the platforms. If it is not the case, we need to install it. We also need to find out the current version that is installed. To do so, open a terminal window and type “python” to find out if it is already installed and which version is installed. If the version number of Python is not displayed, these are the steps taken in downloading and installing Python. For Windows: open <https://www.python.org/downloads/> in the web browser. Click on the version of Python which has to be installed. Save the installer file and run it on the local machine. This opens up the Python installation wizard which is easy to use. Accept the default settings and Python version required by you is installed for Windows.

Almost the same setup has to be followed when Python has to be installed for UNIX, LINUX or Machintosh.

The next step is to set up the PATH. PATH is the system variable that the operating system (OS) uses to locate the needed executables either from the command line or terminal window. This is set differently in different OS.

### 1. Setting path at Unix/Linux:

- (a) In the csh shell: type `setenv PATH "$PATH:/usr/local/bin/python3"` and press Enter.
- (b) In the bash shell (Linux): type `export PATH="$PATH:/usr/local/bin/python3"` and press Enter.
- (c) In the sh or ksh shell: type `PATH="$PATH:/usr/local/bin/python3"` and press Enter.

### 2. Setting path at Windows:

To add the Python directory to the path for a particular session in Windows, the path is set accordingly.

### 3. Setting path at the command prompt:

Type path %path%;C:\Python and press Enter.

### 4.2.1 Running Python

After installing the required version of Python and setting the Path, we need to examine the different ways to start working with Python. Basically Python can be started with the interactive interpreter mode, command line using a script, or Integrated Development Environment (IDE).

### 1. Interactive interpreter mode:

You can start Python from Windows or any other system that provides you a command-line interpreter or shell window by typing in “python”. The Python prompt immediately opens up for interactive interpreter.

Figure 4.1 shows how Python can be started from an interactive interpreter. Coding can be done right away in the interpreter.

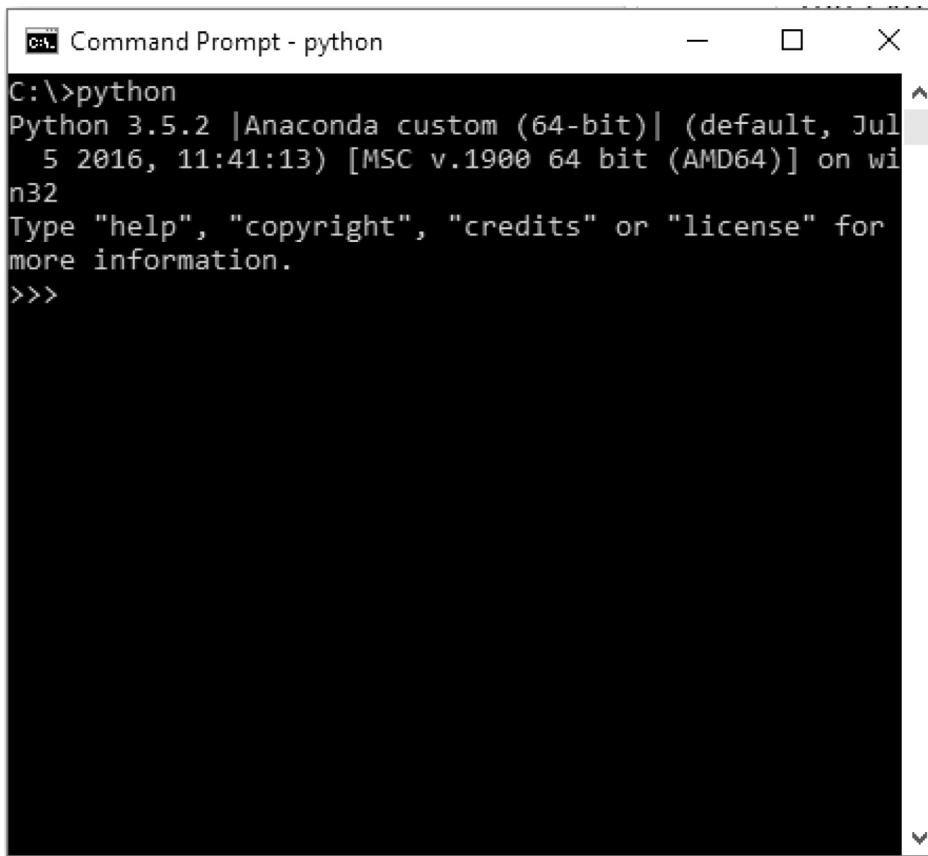
### 2. Script from the command line:

A Python script can be executed at the command line. Python files are stored with the extension .py. In the command prompt a stored Python script can be directly invoked.

### 3. Integrated development environment:

You can run Python from a GUI environment:

- (a) Windows: PythonWin provides a simple graphical interface for editing and running Python programs.
- (b) Unix: IDLE is Unix IDE for Python.
- (c) Macintosh: IDE for Mac is MacBinary or BinHex'd files.



```
C:\>python
Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul
 5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)] on wi
n32
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

**Figure 4.1** Python started in Windows using interactive interpreter mode.

## 4.3 Anaconda

---

Anaconda is an open-source for Python distribution and has a collection of hundreds of packages related to data science, scientific programming, and development. It also comes with a platform-agnostic package manager called *conda*. Jupyter Notebook helps in writing programs in Python with accessibility to lots of scientific and numerical libraries. For analytical programming, RStudio is a ready environment available for users who wish to code in R. Anaconda is a package manager. It tries to solve the *dependency hell* in Python, where different projects have different dependency versions, as otherwise they may interfere with each other.

### 4.3.1 Installing Anaconda

Anaconda can be installed by clicking on the link

<https://www.anaconda.com/download/>

The main screen of Anaconda is similar to Chrome, where you get stuffed with a lot of webpages and different tools like Jupyter, RStudio, Orange, Spyder, and so on. Figure 4.2 shows the main screen of Anaconda navigator.

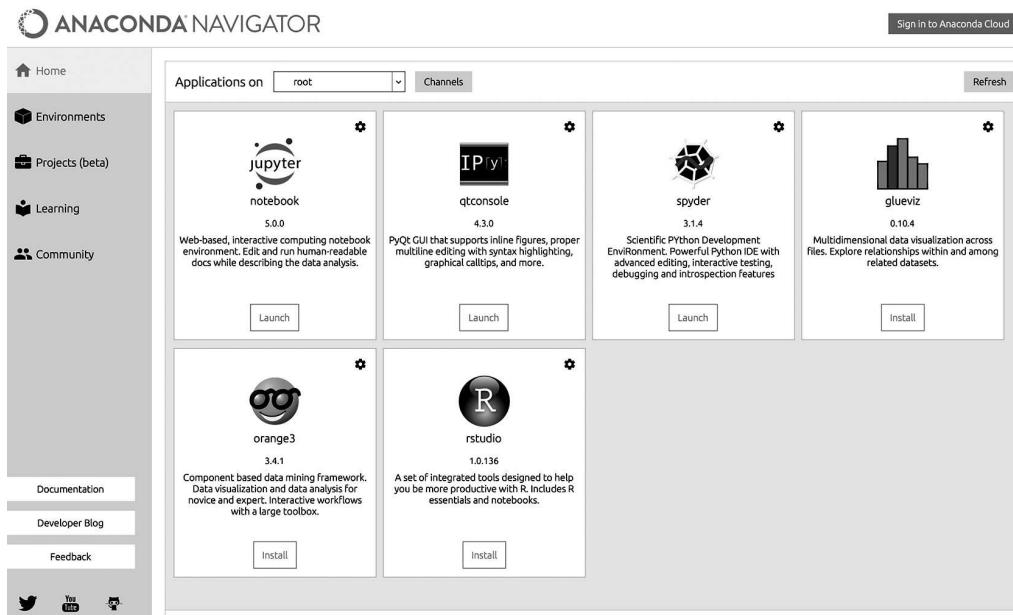


Figure 4.2 Main screen of Anaconda navigator.

## 4.4 Running Jupyter Notebook

Jupyter Notebook runs on the user's browser, so each time the notebook is opened the user will be directed to his/her default browser. To launch Jupyter Notebook, click on the launch button available in Anaconda Prompt. Figure 4.3 shows the Jupyter Notebook launch page.

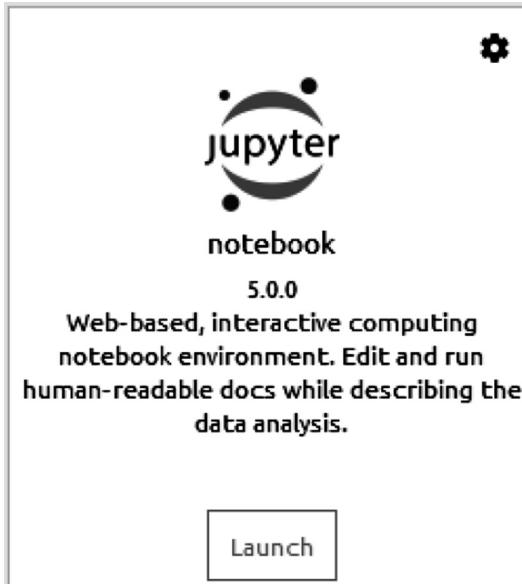
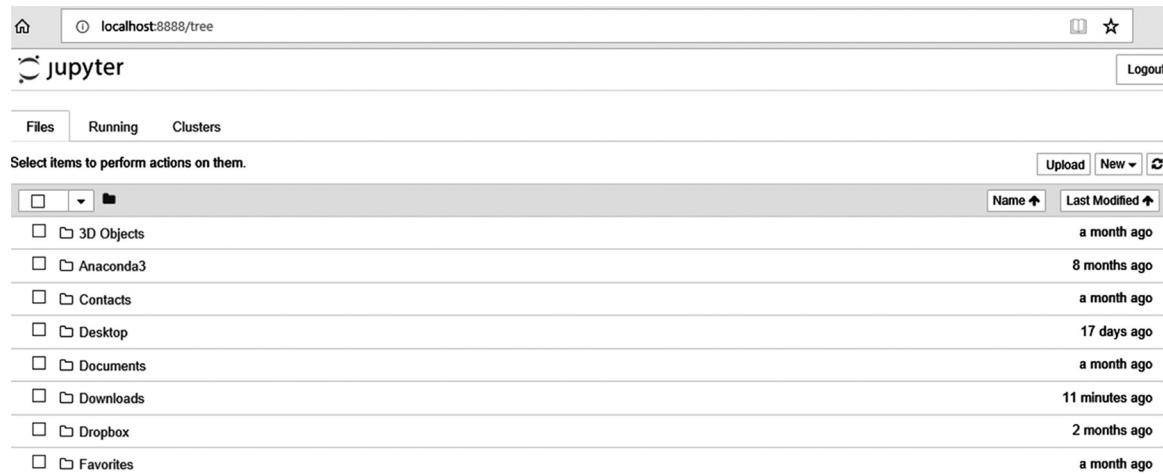


Figure 4.3 Launch page of Jupyter Notebook.



**Figure 4.4** Default browser page in Jupyter Notebook.

After clicking launch, the user will be directed to a default browser, as is displayed in Fig. 4.4. The default browser shows a list of folders available in the directory.

To enter the code in Python, click New → Python 3 as shown in Fig. 4.5.



**Figure 4.5** New notebook to be opened in Jupyter Notebook.

Once Python 3 in the menu is clicked, which is the first page of the Python environment, an interactive code window is opened that can be saved on run time. The best thing about Jupyter Notebook is that it saves automatically each time the code is edited. Jupyter is a presentation layer. It tries to solve the issue of reproducibility in analysis by enabling an iterative and hands-on approach to explain and visualize code. This is also done by using rich text documentations combined with visual representations in a single solution. Figure 4.6 shows how Jupyter Notebook can be used for interactive programming.

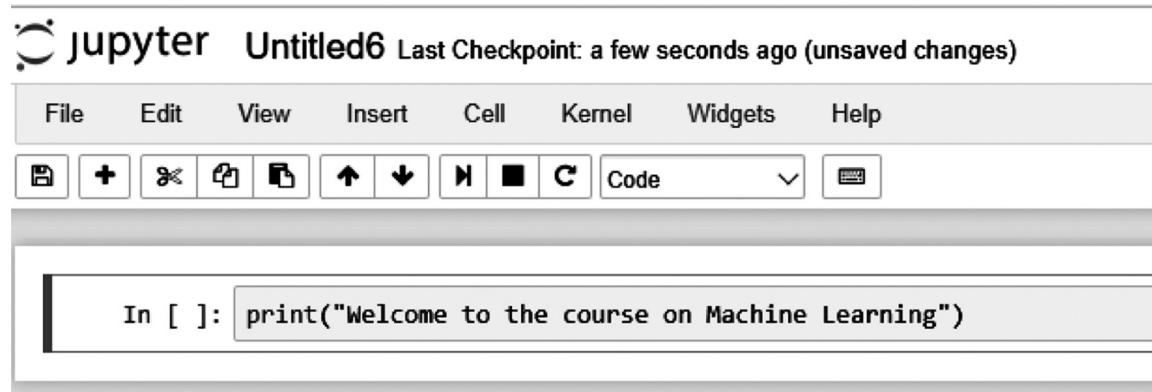


Figure 4.6 Interactive programming in Jupyter Notebook.

#### 4.4.1 Spyder

Spyder is an interactive Python development environment providing MATLAB-like features in a simple and light-weighted software. It also provides ready-to-use pure Python widgets for the application.

## 4.5 Python 3: Basic Syntax

Python version 3.0 was released in 2008. Python 3 is not backward compatible with Python 2.X versions. It was released keeping in mind that the duplicate programming constructs are removed. There are two main modes in programming for executing programs in Python 3.

1. **Interactive mode programming:** Programming in interactive mode without passing the script file in Jupyter Notebook is as follows:

```
print("hello world")
hello world
```

2. **Script mode programming:** In case of script mode, the file is stored separately with a .py extension. In this mode, the interpreter writes the name of file. The path also has to be correctly mentioned.

## 4.6 Python Identifiers

---

Identifiers in Python are names given to identify a variable, function, class, module, or any other object. The following protocols must be adhered to while naming identifiers in Python.

1. An identifier starts with a letter (A to Z or a to z) or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).
2. In identifiers, punctuation characters such as @, \$, and % are not allowed.
3. Identifiers are case sensitive. As an example “\_FirstProgram\_” and “\_firstprogram\_” are different identifiers in Python 3.
4. Names given for classes should always start with an uppercase letter. For other identifiers it is lowercase.
5. An identifier starting with an underscore indicates that it is private.
6. An identifier starting with two leading underscores indicates that it is a strong private identifier.
7. If the identifier starts and ends with two trailing underscores, then it is a language defined special name.

### 4.6.1 Reserved Words

Python has a set of reserved words or keywords and they cannot be used as constants, variables, or identifier names. It should be noted that all Python keywords contain only lowercase letters. Figure 4.7 shows a set of keywords used in Python.

and	print	del
exec	class	in
not	global	while
as	raise	elif
finally	continue	is
or	if	with
assert	return	else
for	def	lambda
pass	import	yield
break	try	except
from		

**Figure 4.7** Set of reserved keywords used in Python 3.

### 4.6.2 Lines and Indentation

Blocks of code in Python are denoted by line indentation. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. An example of code indentation is given in Fig. 4.8.

**Output:** yes

**Figure 4.8** Example of code indentation.

Python allows the use of the line continuation character (\) to denote that a line should continue. Consider the example shown in Fig. 4.9; item1 and item2 are declared. Item is the sum of item1 and item 2. In line 3 of the figure, item 2 is continued in the next line and the sum of item 1 and item 2 is computed and printed.

```

item1=3
item2=4
item =item1+\n
item2
print (item)
7

```

**Figure 4.9** Example of code for line continuation.

Python accepts single ('), double ("), and triple (‘‘ or “““) quotes to denote string literals, as long as the same type of quote starts and ends the string. Triple quotes are used to span the string across multiple lines. For example, all the quotes shown in Fig. 4.10 are legal. The quotes ‘word’, “sentence”, and “““sent”” use one, two, and three quotes, respectively. In “““sent””, the sentence is continued in the next line using triple quotes.

```

word='wording'
sentence="This is an example of quotes in python"
sent="""this is another
example of sentence in python"""
print(word)
print(sentence)
print(sent)

wording
This is an example of quotes in python
This is another
example of sentence in python

```

**Figure 4.10** Different quotes used in Python.

A comment in Python is marked by a hash symbol (#). Multiple statements on a single line can be separated by a semicolon (;).

### 4.6.3 Variables in Python 3

#### 4.6.3.1 Assigning Values to Variables

The equal sign (=) is used to assign values to variables. An example of the assignment operator is given in Fig. 4.11. The sum of *a*, *b*, and *c* is assigned to the value **tot** and printed.

```

a=1;b=2;c=3
tot=a+b+c
print(tot)
6

```

**Figure 4.11** Code for equal sign (=).

#### 4.6.3.2 Multiple Assignment

Multiple assignments in Python can be used as shown in Fig. 4.12. Variables *a*, *b*, and *c* are all assigned a value 1. Similarly, variables *x*, *y*, and *z* are assigned values 1, 2, and jack, respectively.

### 4.6.4 Standard Data Types

Python supports five basic data types. They are Python numbers, strings, lists, tuples, and dictionary. These data types are expanded in the following sections.

```
#Single assignment
a=b=c=1
print(a)
print(b)
print(c)
#multiple assignment
x,y,z=1,2,"jack"
print(x)
print(y)
print(z)
1
1
1
1
2
jack
```

**Figure 4.12** Example of multiple assignment.

#### 4.6.4.1 Python Numbers

Number data types store numeric values. Number objects are created when the user assigns a value to them. Python supports three kinds of number data types: (a) int, which are signed integers; (b) float, which are floating point real values; and (c) complex, which are complex numbers. In the example shown in Fig. 4.13, variables *m*, *n*, *o*, and *p* are assigned values 10, 20, 10.567, and 56+7j, respectively. When it is printed, the values are assigned automatically to the different number types.

```
m=10#integer number
n=20#integer number
o=10.567#float number
p=56+7j#complex number
print(m)
print(n)
print(o)
print(p)
10
20
10.567
(56+7j)
```

**Figure 4.13** Code showing different types of Python numbers.

#### 4.6.4.2 Python Strings

Strings in Python are identified as a contiguous set of characters represented in quotation marks. Python allows either a pair of single or double quotes. Subsets of strings can be taken using the *slice operator* ([ ]) and [:] with indexes starting at 0 in the beginning of the string and working their way from -1 to the end.

Plus sign (+) is the *string concatenation operator* and asterisk (\*) is the *repetition operator*. The example shown in Fig. 4.14 has a string *str* with the content “Hello World!”. *str*[3] displays the third character in *str* which is “l”. *str*[1:5] displays the output from the first character “ello”. *str*[4:] prints from the fourth character (i.e., “o world”). When *str*\*3 is printed it repeats “Hello World!” thrice.

#### 4.6.4.3 Python Lists

A Python list contains items separated by commas and enclosed within square brackets ([ ]). Lists in Python are similar to arrays in C, with a difference that items in the Python list can be of different data types.

```

str="Hello World"
print(str)#Complete string is printed
print(str[3])#The third character is printed
print(str[1:5])#The characters inbetween 1 and 5 are printed
print(str[4:])#From 4th character the string is printed
print(str*3)#prints the character 3 times
Hello World!
1
ello
o World!
Hello World!Hello World!Hello World!

```

**Figure 4.14** Example of strings and substrings in Python.

Just as in strings, values stored in a list can be accessed using the slice operator ([ ]) and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. Similarly, plus sign (+) is the list concatenation operator and asterisk (\*) is the repetition operator. The example shown in Fig. 4.15 has two lists: mylist and mysmalllist. mylist has a combination of strings, integers, and float datatype. The specific item in the list is displayed by using the index number. mylist [3] displays the element of the third index, which is 'ramesh'. mylist [-1] displays the first element from the right of the list. In this case the value of mylist [-1] is 70.2, which happens to be the last element of the list. So if the list has to be accessed in the reverse order from right to left then negative index values are to be used. Just like string elements, which can be repeated using the asterisk (\*) symbol, mysmalllist is replicated twice by using mysmalllist \*2.

```

mylist=['abcd', 123, 2.23, 'ramesh', 70.2]
mysmalllist=[123, 'john']
print(mylist)#print the entire list
print(mylist[3])#prints the 4th element in the list
print(mylist[-1])#print the last element in the list
'print(mysmalllist*2)#repeats the elements of mysmalllist twice
['abcd', 123, 2.23, 'ramesh', 70.2]
ramesh
70.2
[123, 'john', 123, 'john']

```

**Figure 4.15** Examples of different Python lists.

#### 4.6.4.4 Python Tuples

A Python tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parenthesis.

Lists are enclosed in square brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists. Figure 4.16 shows an example of a tuple that is assigned some values. Just like in Python lists, elements of the tuple can be individually accessed. Values can also be added to a tuple by concatenating with another tuple.

#### 4.6.4.5 Python Dictionaries

Python dictionaries are of hash-table type and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([ ]). In the example shown in Fig. 4.17, the dictionary is initiated in curly brackets. dict is an empty

```

tuple=('abcd', 123, 2.23, 'ramesh', 70.2)
tinytuple=(12345, 67890)
print(tuple)
print(tuple[1])
print(tuple[2:])
print(tuple+tinytuple)
('abcd', 123, 2.23, 'ramesh', 70.2)
123
(2.23,'ramesh'. 70.2)
('abcd', 123, 2.23, 'ramesh', 70.2, 12345, 67890)

```

**Figure 4.16** Example of tuples in Python.

dictionary. The first and second element of `dict` is given by `dict[1]` and `dict[2]`. `dict` variable is initiated by key way pairs. If the keys have to be displayed then `variablename.keys()` is used and if values have to be displayed, `variablename.values()` is used. So, `dict.keys()` displays the keys name, age, and department. `dict.values()` displays the values james, 25, and accounts.

```

dict={}#initiating an empty dictionary
dict[1]="#This is an example of dictionary"#The first key 1 has a value given in
      #the right hand side
dict[2]="#this is an example of key-value pair"#The second key 2 has a value given in
      #the right hand side
diction={"name":"james", " age":25, "department":"accounts"}
print(dict[2])#prints the value of the key
print(diction)#prints the entire dictionary
print(diction.keys())#prints the entire set of keys of diction
print(diction.values())#prints the values of the keys
This is an example of key-value pair
{'name':'james','age':25, 'department':'accounts'}
dict_keys(['name', 'age', 'department'])
dict_values(['james', 25, 'accounts'])

```

**Figure 4.17** Example of dictionary in Python.

In this section, we have learned about the five basic data types of Python:

1. Python numbers: Used to represent numeric values.
2. Python strings: Used to represent characters or continuous set of characters.
3. Python lists: Used to represent items of different kinds separated by commas.
4. Python tuples: Used to represent sequence of items like lists.
5. Python dictionary: Used to represent items in the key:value pair.

## 4.7 Basic Operators in Python

---

Operators are constructs that can manipulate the value of *operands*. For example, in the expression  $4 + 5 = 9$ , the numbers 4 and 5 are called operands and plus sign (+) is called *operator*. Python supports different types of operators. These are discussed in the following sections.

### 4.7.1 Arithmetic Operators

The basic arithmetic operations are addition, subtraction, division, multiplication, modulus, exponent, and floor division. Various arithmetic operators as executed in Jupyter Notebook are shown in Fig. 4.18 (e.g., if  $a = 10$ ,  $b = 20$ , and  $c = 0$ ). The sum, difference, product, division, modulus, and power operators are similar to other programming languages. In floor division, which is represented by `//`, the output is rounded off to the floor.

```

#Various Arithmetic Operators
a=10;b=20;c=0# variables are initialized
c=a+b;print("Addition is ",c)#Addition operation
c=a-b;print("Difference is ",c)#Substration operation
c=a*b;print("Product is ",c)#Multiplication operation
c=a/b;print("Quotient is ",c)#Division operation
c=a%b;print("Modulus is ",c)#Modulus operation
c=a**b;print("A raise to the power of b is ",c)
c=b//a;print("b floor divided by a is given by",c)

Addition is 30
Difference is -10
Product is 200
Quotient is 0.5
Modulus is 10
A raise to the power of b is 10000000000000000000000000000000
b floor divided by a is given by 2

```

**Figure 4.18** Various arithmetic operators of Python.

### 4.7.2 Comparison (Relational) Operators

Comparison operators compare the values on either side and decide the relation between them. They are also called relational operators. The different relational operators are shown in Fig. 4.19 (e.g., if  $a = 10$  and  $b = 20$ ). Variables  $a$ ,  $b$ , and  $c$  are initialized to values 10, 20, and 0, respectively. The outputs are then displayed based on comparisons made.

```

#Various Comparison or relational Operators
a=10;b=20;c=0# variables are initialized
if (a==b):
    print("a is equal to b")
else:
    print("a is not equal to b")
if (a<b):
    print("a is lesser than b")
else:
    print("a is not lesser than b")
if (a>b):
    print("a is greater than b")
else:
    print("a is not greater than b")

if (a<=b):
    print("a is lesser than or equal to b")
else:
    print("a is not lesser than or equal to b")

if (a!=b):
    print("a is not equal to b")
else:
    print("a is equal to b")

a is not equal to b
a is lesser than b
a is not greater than b
a is lesser than or equal to b
a is not equal to b

```

**Figure 4.19** Comparison operators in Python.

### 4.7.3 Assignment Operators

The assignment operator assigns the value of the right-hand side to the left-hand side. The assignment operator is used for assigning values in the right hand side to the variables in the left hand side. The different assignment operators are equal to, add and, subtract and, multiple and, divide and, modulus and, and exponential and. In Fig. 4.20, these operations are performed and the outputs are self-explanatory.

```
#Examples of Assignment operators
a=100
b=10
b+=a; print("example of add and is ",b)
b-=a; print("example of subtract and is",b)
b*=a; print("example of multiple and is ",b)
b/=a; print("example of divide and is ",b)
b%=a; print("The example of modulus and is ",b)
b**=a; print("The example of exponential and is ",b)

example of add and is 110
example of subtract and is 10
example of multiple and is 1000
example of divide and is 10.0
The example of modulus and is 10.0
The example of exponential and is 1e+100
```

Figure 4.20 Different assignment operators used in Python.

### 4.7.4 Logical and Membership Operators

The logical operators supported by Python language are logical AND, logical OR, and logical NOT. The membership operators are **in** and **not in** operators, demonstrated in the example shown in Fig. 4.21. In the list, the presence or absence of specific elements is tested using conditional statements **if (a in list):** or **if (b not in list):**. The appropriate output messages are displayed in each case.

```
#Examples of Logical operators
a=100
b=10
list = [2,4,6,8,10]
if(a in list):# Example of in operators
    print("a is present in the list")
else:
    print("a is not present in the list")
if(b not in list):#Example of not in operators
    print("b is not in list")
else:
    print("b is present in list")

a is not present in the list
b is present in list
```

Figure 4.21 Example of **in** and **not in** used in list.

## 4.8

## Python Decision-Making

---

In any programming language, the execution of commands is mainly controlled by decision-making constructs. Decision-making is the anticipation of conditions that occur during the execution of a program and the specified actions that have to be taken in accordance with these conditions.

Decision structures evaluate multiple expressions, which produce **TRUE** or **FALSE** as an outcome. The action taken is dependent on the **TRUE** or **FALSE** conditions.

The basic decision-making statements in Python, like any other language, are as follows:

1. **If statement:** The basic syntax for the **if** statement is:

**If expression:**  
    **statement(s)**

```
#Example of simple if block
a=1000
if(a%2==0):
    print("Even number")
```

Even number

**Figure 4.22** A simple **if** block to check whether a number is even or odd.

It should be noted that in case the **if** statement evaluates to be **TRUE**, then the statements after that should follow an indentation. In the example shown in Fig. 4.22, a check is made to see if a number is even. Since an even number is divisible by 2, the statements following the **if** statement are completed (i.e., the even number is printed).

2. **If...else statement:** An **else** statement can be added to the **if** statement when required. The syntax for **if...else** statement is:

**If expression:**  
    **statement(s)**  
**else:**  
    **statement(s)**

Figure 4.23 gives an example of the **if...else** statement to check whether a given number is odd or even.

```
#Example of if..else block
a=1000
if(a%2==0):
    print("Even number")
else:
    print("odd number")
```

Even number

**Figure 4.23** Code to demonstrate the **if...else** statement.

3. **If...elif...else statement:** The **elif** statement allows the user to check multiple expressions for **TRUE** and execute a block of code as soon as one of the conditions evaluates to **TRUE**. Similar to the **else** statement, the **elif** statement is optional. However, unlike the **else** statement, for which there can be at the most one statement, there can be an arbitrary number of **elif** statements following an **if**, as shown:

```
if expression1:
    statement(s)
elif expression2:
    statement(s)
elif expression3:
    statement(s)
else:
    statement(s)
```

In Fig. 4.24, the interest rate is computed as per the principal amount. When the principal is inputted, based on the **if...elif...else** block, the interest rate and the mount is computed. If more than one condition has to be checked then the **if...elif...else** block is used.

```
#Example of if..elif..else statements
#example to compute simple interest using if..elif..else statements
principle=int(input("Enter the principle amount"))
if (principle<1000):
    interest=10
elif (principle>1000 and principle<2500):
    interest=12
elif (principle>=2500 and principle<5000):
    interest=15
else:
    interest=20
amount = principle + principle*interest/100
print("The amount after one year is ",amount)

Enter the principle amount 2000
The amount after one year is 2240.0
```

**Figure 4.24** Code to demonstrate **if...elif...else** block.

- 4. Nested if statement:** There may be a situation when the user wants to check for another condition after it resolves to **TRUE**. In such a situation, he/she can use the nested **if** construct. In a nested **if** construct, the user can have an **if...elif...else** construct inside another **if...elif...else**, as shown below.

```
if expression1:
    statement(s)
if expression2:
    statement(s)
elif expression3:
    statement(s)
else
    statement(s)
elif expression4:
    statement(s)
else:
    statement(s)
```

In Fig. 4.25, the test of divisibility by 2 and 3 is best explained through code, using multiple **if** statements.

```
#Example of nested if statements
#Program to find test of divisibility of 2 and 3
num=int(input("Enter any number"))
if num%2==0:
    if num%3==0:
        print ("Divisible by 3 and 2")
    else:
        print ("divisible by 2 not divisible by 3")
else:
    if num%3==0:
        print ("divisible by 3 not divisible by 2")
    else:
        print ("not Divisible by 2 not divisible by 3")

Enter any number 12
Divisible by 3 and 2
```

**Figure 4.25** Code to demonstrate nested **if...else** block.

## 4.9 Python Loops

A loop statement allows the user to execute a statement or group of statements multiple times. Python programming provides the following types of loops to handle these requirements:

1. While loop
2. For loop
3. Nested loops

### 4.9.1 While loop

A **while** loop statement in Python programming repeatedly executes a target statement as long as the given condition is **TRUE**.

**while expression:**  
    **statement(s)**

An example of **while** loop is given in Fig. 4.26. Here, numbers 11 to 15 are printed by using this statement.

```
#Example of while loop
count=10
while (count<15) :
    count=count+1
    print(count)
11
12
13
14
15
```

**Figure 4.26** Execution of a simple **while** loop.

A loop becomes an infinite loop if a condition never becomes **FALSE**. The user must be cautious when using the **while** loop because of the possibility that the condition may never resolve to a **FALSE** value. This results in a loop that never ends.

#### 4.9.1.1 While Loop with an Else Block

An **else** statement can be used with the **while** loop and gets executed when the condition becomes **FALSE**.

Consider the example shown in Fig. 4.27. The variable *count* is first initialized to value 5. While this value is less than 10, a **print** statement is displayed and the counter is incremented by 1 every time. When the value of the counter becomes greater than 10 then the execution goes to the **else** block, where it displays that the counter value is greater than 10.

## 4.9.2 For Loop

The **for** statement in Python has the ability to iterate over a range of items of any sequence, such as a list or a string. If a sequence contains an expression list, it is evaluated first. Then the first item in the sequence is assigned the iterating variable, **iterating\_var**. Next, the statements block is executed. This is repeated for each item until the entire sequence is exhausted. For iterating over a sequence of numbers, a **range** function is used:

**for iterating\_var in sequence:**  
    **statements(s)**

```
#Example of while loop with else construct
count=5
while(count<=10):
    print(count,"is less than 10")
    count=count+1
else:
    print(count,"is greater than 10")

5 is less than 10
6 is less than 10
7 is less than 10
8 is less than 10
9 is less than 10
10 is less than 10
11 is greater than 10
```

**Figure 4.27** Code using **while** loop with **else** construct.

In the example shown in Fig. 4.28, the list of fruits are first declared and then a **for** loop is used. An index then iterates over the list to display the fruits present in the list.

```
fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
    print ('Current fruit :', fruits[index])

Current fruit : banana
Current fruit : apple
Current fruit : mango
```

**Figure 4.28** Example of a **for** loop.

Lists can be alternatively traversed by the code shown in Fig. 4.29, which is just another way of iterating over a **for** loop.

```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:           #traversal of list sequence
    print('Current fruit :', fruit)

Current fruit : banana
Current fruit : apple
Current fruit : mango
```

**Figure 4.29** Alternative use of **for** loop.

#### 4.9.2.1 Else Statement with For Loop

Conditions can be checked inside the **for** loop, and if the conditions are satisfied, the loop is followed by a **break** statement. This is shown in Fig. 4.30, where a test is performed to check whether a number is divisible by 5 or not.

#### 4.9.3 Nested Loops

Python programming allows the use of one loop inside another. The following syntax given is that of a nested **for** statement:

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statements(s)
    statements(s)
```

```

#Example of an else block in a for block
#Example of a break statement
numbers=[11,33,55,39,55,75,37,21,23,41,13]
for num in numbers:
    if num%5==0:
        print ('the list contains a multiple of 5')
        break
    else:
        print ('the list doesnot contain a multiple of 5')

the list contains a multiple of 5

```

**Figure 4.30** Example of **for** loop with an **else** block in it.

The syntax for a nested **while** loop is given as follows:

```

while expression:
    while expression:
        statement(s)
        statement(s)

```

In the example shown in Fig. 4.31, a two-dimensional array is printed using the concept of nested loops.

```

#Example of nested for loops
for i in range(1,5):
    for j in range (1,3) :
        k=i*j
        print (k, end=' ')
    print()

1 2
2 4
3 6
4 8

```

**Figure 4.31** Example of nested loop.

#### 4.9.4 Loop Control Statements

Loop control statements change the execution of a program from its normal sequence. When the execution leaves a scope, all automatic objects that were created in that scope are destroyed. The following control statements are supported in Python:

1. Break
2. Continue
3. Pass

##### 4.9.4.1 Break Statement

The **break** statement is used for premature termination of a loop. The most common use of **break** is when some external condition is triggered requiring a hasty exit from a loop. The **break** statement can be used in both **while** and **for** loops. If nested loops are used, the **break** statement stops execution of the innermost loop and starts executing the next line of the code after the block.

A frequently used code for separating words – based on the spacing between the words – works with the concept of a conditional **if** statement in the **for** loop followed by a **break** block (Fig. 4.32).

```
for letter in 'Machine Learning':    #Example of break statement for splitting words
    if letter=='':
        break
    print('Current Letter:',letter)

Current Letter:M
Current Letter:a
Current Letter:c
Current Letter:h
Current Letter:i
Current Letter:n
Current Letter:e
```

**Figure 4.32** Code showing the working of **break** statement.

#### 4.9.4.2 Continue Statement

The **continue** statement in Python returns the control to the beginning of the current loop. When encountered, the loop starts next iteration without executing the remaining statements in the current one. The **continue** statement can be used in both **while** and **for** loops.

In the example shown in Fig. 4.33, when the value of *var* is 5 then that particular value is skipped. The remaining digits from 9 to 1 are printed.

```
#Example of continue. With continue that particular variable 5 is skipped. But loop
continues
var=10
while var>0:
    var=var-1
    if var ==5:
        continue
    print ('Current variable value:', var)

Current variable value:9
Current variable value:8
Current variable value:7
Current variable value:6
Current variable value:4
Current variable value:3
Current variable value:2
Current variable value:1
Current variable value:0
```

**Figure 4.33** Example of a **while** statement used with **continue** statement.

#### 4.9.4.3 Pass Block

The **pass** block is used when a statement is required syntactically but the user does not want any command or code to execute. The **pass** statement is a *null operation*; nothing happens when it is executed. Consider the example shown in Fig. 4.34. When *letter* = ' ', then that particular space is passed and the sentence "This is pass block" is displayed.

```

#Example of pass statement
for letter in 'Machine Learning':
    if letter == '':
        pass
        print ('This is pass block')
    print ('Current Letter :',letter)

Current Letter : M
Current Letter : a
Current Letter : c
Current Letter : h
Current Letter : i
Current Letter : n
Current Letter : e
This is pass block
Current Letter :
Current Letter : L
Current Letter : e
Current Letter : a
Current Letter : r
Current Letter : n
Current Letter : i
Current Letter : n
Current Letter : g

```

**Figure 4.34** Example of a `pass` block.

## 4.10 Numerical Python (NumPy)

Numerical Python, or NumPy, contains multidimensional array objects and routines for processing those arrays. It is used for mathematical and logical operations on arrays. Fourier transform and linear algebra operations can also be performed using NumPy. There is a myriad of in-built functions in support of linear algebra in NumPy. NumPy is usually used with SciPy and MatlabPlots for generating mathematical functions and plotting. It has to be explicitly installed by using pip install NumPy.

### 4.10.1 Pip Install NumPy

The most important object defined in NumPy is an  $N$ -dimensional array type called `ndarray`.

```

#Examples of nd array
import numpy as np
a=np.array([1,2,3])#1D array
print ("Examples of 1D array",a)
b=np.array([[1,2],[3,4]])
print("Examples of 2D array",b)

Examples of 1D array [1 2 3]
Examples of 2D array [[1 2]
                     [3 4]]
c=np.array([1,2,3,4,5],dtype='complex')
print("Array of complex numbers",c)

Array of complex numbers [1.+0.j 2.+0.j 3.+0.j 4.+0.j 5.+0.j]

```

**Figure 4.35** Example of ndarray object in NumPy.

In order to implement NumPy, the user must first import it. NumPy has an array method, which is used to display any type of numbers. In Fig. 4.35, NumPy is used to display one-dimensional array, two-dimensional array, and complex number array. Using `dtype` parameters, a complex array of numbers can be assigned to a variable.

NumPy supports a much greater variety of numerical types than Python. Table 4.1 shows different scalar data types defined in NumPy.

**Table 4.1** Scalar data types defined in NumPy

S. No.	<i>Data Types and Description</i>
1	<b>bool_</b> Boolean (True or False) stored as a byte
2	<b>int_</b> Default integer type (same as C long; normally either int64 or int32)
3	<b>intc</b> Identical to C int (normally int32 or int64)
4	<b>intp</b> Integer used for indexing (same as C ssize_t; normally either int32 or int64)
5	<b>int8</b> Byte (-128 to 127)
6	<b>int16</b> Integer (-32768 to 32767)
7	<b>int32</b> Integer (-2147483648 to 2147483647)
8	<b>int64</b> Integer (-9223372036854775808 to 9223372036854775807)
9	<b>uint8</b> Unsigned integer (0 to 255)
10	<b>uint16</b> Unsigned integer (0 to 65535)
11	<b>uint32</b> Unsigned integer (0 to 4294967295)
12	<b>uint64</b> Unsigned integer (0 to 18446744073709551615)
13	<b>float_</b> Shorthand for float64
14	<b>float16</b> Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
15	<b>float32</b> Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
16	<b>float64</b> Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
17	<b>complex_</b> Shorthand for complex128

<i>S. No.</i>	<i>Data Types and Description</i>
18	<b>complex64</b> Complex number, represented by two 32-bit floats (real and imaginary components)
19	<b>complex128</b> Complex number, represented by two 64-bit floats (real and imaginary components)

Source: TutorialsPoint [https://www.tutorialspoint.com/numpy/numpy\\_data\\_types.htm](https://www.tutorialspoint.com/numpy/numpy_data_types.htm)

#### 4.10.2 Array Attributes of NumPy

Ndarray has different array attributes. The most important are listed below.

1. **ndarray.shape:** Specifies the dimensions of the array.
2. **ndarray.reshape:** Helps in changing the dimensions of the array.
3. **ndarray.arange:** Helps in assigning a range of integers to a variable.
4. **numPy.itemsize:** Returns the length of each element of array in bytes.
5. **np.empty(shape, dtype = float, order = "C"):** Creates an empty array with the specifications provided.
6. **np.zeros:** Returns a new array of specified size, filled with zeros.
7. **np.ones:** Returns a new array of specified size, filled with ones.
8. **numpy.linspace:** This function is similar to **arange()** function, but instead of step size, the number of evenly spaced values between the interval is specified.

Dimensions of the array can be found and reshaped using shape and reshape options (Fig. 4.36).

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
print ("The dimensions of the array is",a.shape)
a.shape=(3,2)
print ("The transposed array elements are",a)
b=a.reshape(3,2)
print("The reshaped array elements are",b)

The dimensions of the array is (2, 3)
The transposed array elements are [[1 2]
 [3 4]
 [5 6]]
The reshaped array elements are [[1 2]
 [3 4]
 [5 6]]
```

**Figure 4.36** The use of shape and reshape methods.

NumPy is a very powerful matrix manipulation package. Shape and reshape options are used in displaying the dimensions of array and reshaping the array. The reshape option in NumPy not only helps in determining the shape of a matrix, it is also used in converting a one-dimensional list to a multidimensional array. Figure 4.37 demonstrates this powerful feature of NumPy. This example shows how quickly a one-dimensional array can be converted into a three-dimensional array.

An empty array of any dimension can be initiated and populated with random numbers. In the example shown in Fig. 4.38, the **dtype** is int. If a float is mentioned, the user can initiate an empty array with random floating point integers.

Another important feature of NumPy is that it can create one-dimensional and multidimensional arrays of zeros and ones. Figure 4.39 shows how matrices can be created with values of ones or zeros. This feature has been invariably used in matrix computations.

```

#this is one dimensional array
import numpy as np
a=np.arange(24)
a.ndim
print(a)

#now reshape it
b=a.reshape(2,4,3)
print(b)
#b is having three dimensions

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[[[0 1 2]
  [3 4 5]
  [6 7 8]
  [9 10 11]]]

[[[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
```

**Figure 4.37** Example of the use of arange method.

```

x=np.empty([3,2,4],dtype=int)
print("The 3D array is given by",x)

The 3D array is given by [[[0 1 2 3]
  [4 5 6 7]]

[[8 9 10 11]
  [12 13 14 15]]

[[16 17 18 19]
  [20 21 22 23]]]
```

**Figure 4.38** Example of a dtype.

```

#Example of ones and Zeros
c=np.ones(3);m=np.zeros(3)
d=np.ones([4,4],dtype=float);n=np.zeros([4,4],dtype=float)
print("The single dimensional array of ones and zeros is given by",c,m)
print("The multidimensional array of ones and zeros is given by",d,n)

The single dimensional array of ones and zeros is given by [ 1. 1. 1.] [ 0. 0. 0.]
The multidimensional array of ones and zeros is given by [[ 1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]] [[ 0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]
```

**Figure 4.39** Example showing the use of np.ones and np.zeros.

NumPy can create a sequence with different start and stop values. The numpy.linspace option gives the number of elements between start and end.

```
numpy.linspace(start, stop, num, endpoint, retstep, dtype)
```

As shown in Fig. 4.40, between start (10) and end (20) there are five elements and these are displayed by *x*. The *retstep* option is used to include or exclude the stop value (Fig. 4.40).

```
import numpy as np
x=np.linspace (10,20,5)
y=np.linspace 10,20,10,retstep=False
print ("The list of integers from 10 to 20 with 5 numbers inbetween then are",x)
print(y)
```

The list of integers from 10 to 20 with 5 numbers inbetween then are [10. 12.5 15. 17.5 20.]  
[10. 11.11111111 12.22222222 13.33333333 14.44444444  
15.55555556 16.66666667 17.77777778 18.88888889 20.]

**Figure 4.40** Example of code using linspace.

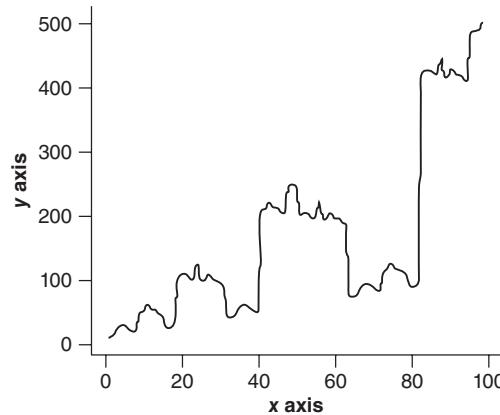
## 4.11 NumPy Matplotlib

Matplotlib is a plotting library for Python. It is used along with NumPy to provide an environment that is an effective open-source alternative for MATLAB. The package is imported into Python script by adding the following statement:

```
from matplotlib import pyplot as plt
```

Figure 4.41 shows how a two-dimensional plot can be easily made using NumPy and Matplotlib. First an independent variable *x* is created using ndarray object. The **np.arange()** function is used for populating an

```
#Plotting in numpy using matplotlib package
import numpy as np
from matplotlib import pyplot as plt
x=np. arange(1,100)
y=2*x^2+3*x + 5
plt.title("Matplotlib demo")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.plot(x,y)
plt.show()
```



**Figure 4.41** Example showing Matplotlib used in plotting graphs.

array for a continuous valued variable  $x$ . The relationship between the independent and dependent variable is given by the equation  $y = 2x^2 + 3x + 5$ . The plot function in the Matplotlib package is used for plotting the graph. The show method is finally used for displaying the graph. The **plot** function has options for different line and marker styles. There are options for different line colors.

#### 4.11.1 Subplot

The subplot () function allows the user to plot different things in the same figure. As a typical example, the sine and cosine functions can be plotted with the same  $x$ -axis values using the subplot () function [Figs. 4.42 (A) and (B)].

```
Export numpy as np
Export matplotlib.pyplot as plt

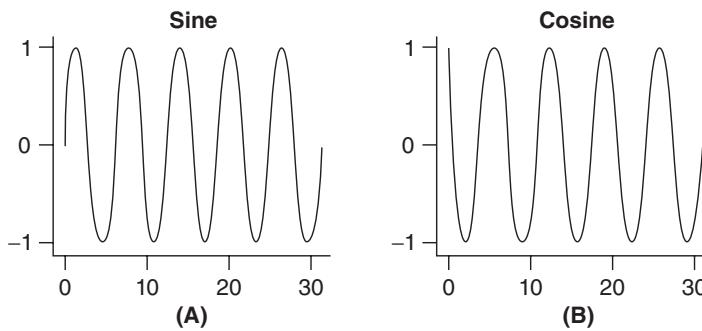
#Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 10*np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

#Set up a subplot grid that has height 2 and width 1,
#and set the first such subplot as active.
plt.subplot(2, 2, 1)

#Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

#Set the second subplot as active, and make the second plot.
plt.subplot(2, 2, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

#Show the figure.
plt.show()
```



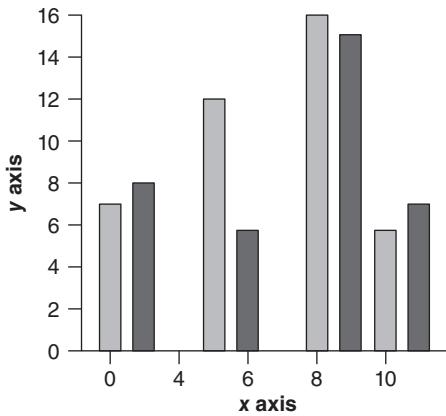
**Figure 4.42** Subplot options for (A) sine and (B) cosine functions.

Bar graphs can also be drawn in Python using the Matplotlib package. The bar method in the plot function is used for this purpose. An example of the code for bar graph is shown in Fig. 4.43, where two bar graphs are plotted on the same  $X$ -axis.

```
from matplotlib import pyplot plt
x1 = [2, 5, 8, 10]
y1 = [7, 12, 16, 6]

x2 = [3, 6, 9, 11]
y2 = [8, 6, 15, 7]
plt.bar(x1, y1)
plt.bar(x2, y2, color = 'r')
plt.title('Bar graph')
plt.ylabel('Y axis')
plt.xlabel('X axis')

plt.show label()
```



**Figure 4.43** Example of bar graph using Matplotlib.

## 4.12 Introduction to Pandas

---

Pandas is a popular Python package for data science. It offers powerful and expressive data structures that are flexible enough to make data manipulation and analysis easy, among other things. The main data structure used for this flexibility in Pandas is DataFrame. This is a two-dimensional structure and represents data in the form of a table with rows and columns. Arithmetic operations can be performed on DataFrame. It looks like a spreadsheet or a table in DataBase.

### 4.12.1 Creating DataFrame in Pandas

DataFrame can be created in different ways and from various inputs like lists, dictionary, series, and NumPy arrays.

#### 4.12.1.1 Creating DataFrame from Scratch

DataFrames are very helpful in *data munging*. It can be created from scratch as an Empty or Basic DataFrame as shown in Fig. 4.44. Pandas uses DataFrame method to create an empty DataFrame.

```
import pandas as pd
df = pd.DataFrame()
print (df)

Empty DataFrame
Columns: []
Index: []
```

**Figure 4.44** Creation of Empty DataFrame.

#### 4.12.1.2 Creating DataFrames from Lists

DataFrames can be created from lists having a single value or from those having more than one value. Figure 4.45 shows examples of both types. Columns should be named when more than one column is used.

```
import pandas as pd
content=[0, 1, 2, 3, 4, 5, 6, 7]
df = pd.DataFrame(content)
print (df)

0
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7

import pandas as pd
content=[[0,'Sun'],[1,'Mon'],[2,'Tues'],[3,'Wed'],[4,'Thu'],[5,'Fri'],[6,'Sat']]
df = pd.DataFrame(content,columns=['Number','DateofWeek'])
print (df)

   Number DateofWeek
0        0       Sun
1        1      Mon
2        2     Tues
3        3      Wed
4        4      Thu
5        5      Fri
6        6      Sat

import pandas as pd
content=[0, 1, 2, 3, 4, 5, 6, 7]
df = pd.DataFrame(content)
print (df)

0
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
```

**Figure 4.45** Creating single lists and multilists using DataFrame.

#### 4.12.1.3 Creating DataFrames from Dictionary

DataFrames can be created when a list of dictionaries is passed as input data. By default, the keys in the dictionary are used as column headings. This is demonstrated in Fig. 4.46.

```
import pandas as pd
data = [{"a":1,'b':2}, {"a":5,'b':10,'c':20}, {"a":10,'b':20,'c':30}]
df = pd.DataFrame(data)
print(df)

   a   b   c
0  1  2  NaN
1  5 10 20.0
2 10 20 30.0
```

**Figure 4.46** Dictionary as input to a DataFrame.

#### 4.12.1.4 Creating DataFrames from Series

DataFrames can be created from a series as input. Let us consider the following example. A series having countries and their capitals is given as input to the DataFrame (Fig. 4.47).

```
import pandas as pd
my_series = pd.Series({"United Kingdom": "London", "India": "New Delhi",
                      "United States": "Washington", "Belgium": "Brussels"})
print(pd.DataFrame(my_series))

          0
Belgium Brussels
India    New Delhi
United Kingdom    London
United States    Washington
```

**Figure 4.47** Creating a DataFrame using series.

Dimensions of the DataFrame can be found by the shape option:

```
df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
print(df.shape)
```

**Output:** (2, 3)

On the other hand, the `len()` function, in combination with index property, will only give information on the height of the DataFrame.

```
df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
print(len(df.index))
```

**Output:** 2

#### 4.12.2 Manipulation of Columns

In a DataFrame, specific columns or rows can be added, deleted, or selected.

```
import pandas as pd
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
      'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print df ['one']
```

The output is as follows:

```
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

Although there are different columns, only the first column can be displayed by mentioning its name and printing the content. A new column can be added using the following syntax:

```
import pandas as pd
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
      'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print ("Adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print df
```

Initially there are two columns (say, one and two), then a new column (say, three) can be added to the existing ones. When this new column is added the output is as follows:

```
Adding a new column by passing as Series:
     one    two    three
a    1.0     1   10.0
b    2.0     2   20.0
c    3.0     3   30.0
d    NaN     4    NaN
```

Columns can be deleted or popped. Pop and append are methods used to drop or add a column in Pandas DataFrame.

```
print ("Deleting the first column using DEL function:")
del df['one']
print df
df.pop('two')
print df
```

New rows can also be added. The **append** function is used to add rows to a DataFrame.

```
import pandas as pd
df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
df = df.append(df2)
print df
```

The output is as follows:

	a	b
0	1	2
1	3	4
0	5	6
1	7	8

Rows can also be deleted by using the **drop** function.

```
import pandas as pd
df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
df = df.append(df2)
# Drop rows with label 0
df = df.drop(0)
print df
```

The output is as follows:

	a	b
1	3	4
1	7	8

In the above example, the first two rows are appended. So there are four rows after appending **df2** to **df**. Then **df.drop(0)** drops the rows with label 0. Thus, two rows are removed.

## 4.13 Introduction to Scikit-Learn

Machine learning algorithms can be implemented in Python using scikit-learn. They are in general classified as supervised and unsupervised learning algorithms.

In supervised learning algorithms, a corresponding output is provided for every input. Based on this dataset, a hypothesis can be modeled. This model is further used in prediction. Regression algorithms, decision tree, and support vector machines are some of the commonly used supervised learning algorithms. Using scikit-learn, the following steps can be performed for supervised learning algorithms:

1. Load dataset.
2. Split data as training and testing dataset.
3. Build the model.
4. Evaluate the model.
5. Use the model for prediction.

In unsupervised learning, the dataset is given to learning algorithms. The dataset has only inputs. The data groups are based on similarity among different data elements.

## Summary

- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- The latest edition of Python 2 is Python 2.7.11. Python 3 is not backward compatible with Python 2. Python 3.6.5 is the latest version of Python 3.
- Python can be started in the interactive interpreter mode, command line using a script, or IDE.
- Python standard data types include python numbers, strings, lists, tuples, and dictionary.
- The basic operators in Python include arithmetic, comparison, assignment, logical, and membership.
- Looping and decision-making statements are similar to other programming languages.
- NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.
- ndarray is the most important object defined in NumPy.
- Matplotlib is a plotting library for Python. It is used along with NumPy to provide an environment that is an effective open-source alternative for MATLAB.
- Pandas is a popular Python package for data science. It offers powerful and expressive data structures that are flexible enough to make data manipulation and analysis easy, among other things.
- Rows and columns can be added, deleted, and indexed using Pandas DataFrame.
- Scikit-learn provides strong features for building models in supervised and unsupervised learning.

## Multiple-Choice Questions

1. What is the output of the following code?

```
Print 9//2
```

(a) 4.5  
(b) 4.0  
(c) 4  
(d) Error
2. What is the output of the following program?

```
i = 0
while i < 3:
    print i
    i++
    print i+1
```

(a) 0 2 1 3 2 4  
(b) 0 1 2 3 4 5  
(c) Error  
(d) 1 0 2 4 3 5
3. What is the output of the following code?

```
L = ['a','b','c','d']
print "".join(L)
```

(a) Error  
(b) None  
(c) abcd  
(d) ['a','b','c','d']
4. Suppose list1 is [3, 4, 5, 20, 5, 25, 1, 3], what is list1 after list1.pop(1)?  
(a) [3, 4, 5, 20, 5, 25, 1, 3]  
(b) [1, 3, 3, 4, 5, 5, 20, 25]  
(c) [3, 5, 20, 5, 25, 1, 3]  
(d) [1, 3, 4, 5, 20, 5, 25]
5. What is the output of the following program?

```
print "Hello World"[::-1]
```

(a) dlroW olleH

- (b) Hello Worl  
 (c) d  
 (d) Error
6. Given a function that does not return any value, what value is shown when executed at the shell?  
 (a) int  
 (b) bool  
 (c) void  
 (d) None
7. What is the output of the following program?  
`print 0.1 + 0.2 == 0.3`  
 (a) True  
 (b) False  
 (c) Machine dependent  
 (d) Error
8. If you have a NumPy array in a variable bats, how would you access the data item at the 5th column, 2nd row?  
 (a) bats[2,5]  
 (b) bats(1,4)
- (c) bats[1,4]  
 (d) bats[4,1]
9. Assume you are given two lists: `a = [1,2,3,4,5]` and `b = [6,7,8,9]`. Create a list that has all the elements of a and b in one dimension. Output: `a = [1,2,3,4,5,6,7,8,9]`. Which of the following options would you choose?  
 (a) `a.append`  
 (b) `a.extend(b)`  
 (c) Both (a) and (b)  
 (d) None of the above
10. How would you create this identity matrix in Python? Keep in mind that Library NumPy has been imported as np.  
 (a) `np.eye(3)`  
 (b) `identity(3)`  
 (c) `np.array([1, 0, 0], [0, 1, 0], [0, 0, 1])`  
 (d) All of the above

## Very Short Answer Questions

1. What is the significance of Anaconda?
2. How is Python interpreted?
3. What is the difference between list and tuple?
4. What does slicing mean in Python?
5. What is a negative index in Python?
6. How can you generate random numbers in Python?
7. Mention the use of // operator in Python.
8. What is the output of `L[2]` if `L = [1,2,3]`?
9. What is the output of `print tuple[2:]` if `tuple = ('abcd', 786 , 2.23, 'john', 70.2)`?
10. What is the output of `print list1 + list2`, if `list1 = ['abcd', 786, 2.23, 'john', 70.2]` and `list2 = [123, 'john']`?
11. What is the output of `print tinylist * 2`, if `tinylist = [123, 'john']`?
12. What is the output of `print str[2:5]`, if `str = 'Hello World!'`?
13. What is the purpose of \*\* operator?
14. How will you capitalize the first letter of a string?
15. How will you check in a string whose characters are all in lowercase?

## Review Questions

---

1. What are the features of Python 3.0?
2. Explain with examples the various looping options used in Python.
3. Why is NumPy so popular?
4. What is DataFrames in Pandas?
5. How is scikit-learn used in machine learning algorithms?

## Answers

---

### **Multiple-Choice Questions**

1. (c)    2. (c)    3. (c)    4. (c)    5. (a)    6. (d)    7. (b)    8. (c)    9. (b)    10. (a)

# 5

# Data Preprocessing

## LEARNING OBJECTIVES

- To perform the data preprocessing step for any given dataset.
- To use scikit-learn function and to remove null values in data cleansing.
- To use any one dimensionality technique to clean and preprocess data.

## LEARNING OUTCOMES

- Students will be able to represent clean data.
- Students will be able to eliminate null values and remove inconsistencies present in the data.
- Students will be able to perform dimensionality reduction techniques like LDA, PCA, and ICA.
- Students will be able to find out the extent to which PCA helps reduce dimensionality without sacrificing essential content of the data.

### 5.1 Overview of Data Preprocessing

In general, raw data stored in a database may not be in an understandable format. Often, raw data can be inconsistent, incomplete, and may contain many errors. Thus, we need to improve the quality of this data and simplify it by a method called *data preprocessing*. Only then can the data be studied and further processed through different data mining techniques. Thus, data preprocessing is one of the most critical steps in data mining. Data preprocessing methods are divided into the following categories.

1. Data cleaning.
2. Data integration.
3. Data transformation.
4. Data reduction.

Data cleaning is the process of removing noise and correcting inconsistencies in the data. Data integration involves merging of data from different sources. Data transformation, which is also known as *normalization*, is the process of organizing database to remove redundancy. Data reduction is the process of reducing data size by eliminating redundant features.

## 5.2 Data Cleaning

Dirty data can cause confusion and result in unreliable and poor output. Hence, the first step in data pre-processing is data cleaning. The cleaning of data is done by filling in missing values, smoothing noisy data by identifying and/or removing outliers, and resolving inconsistencies. The following are some data cleaning routines.

### 5.2.1 Filling in Missing Values

Missing data elements in a database is a common problem in the real-world scenario. The accuracy of predictions gets affected as missing data elements results in poor-quality database. Thus, we need to fill in the missing data elements before the database is used for any prediction or other data mining processes.

Missing data generally happens while people fill in details in online surveys of products. Some fields in the form cannot be held mandatory as customers find it a nuisance and stop filling the feedback form, which in turn affects data availability. Some customers share their experience, but not how long they are using the product; some customers share how long they are using the product and their experience but not their contact information. Thus, some part of the data is always missing, and it is very common in real world scenario.

Let us now see how to handle missing values (say, NA or NaN) using Pandas, with the following program:

```
# import the pandas library
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f', 'h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df
```

The output is as follows:

	one	two	three
a	0.057988	0.376149	0.865836
b	NaN	NaN	NaN
c	-0.590208	-0.351605	-2.501950
d	NaN	NaN	NaN
e	-1.000303	-0.888201	2.510072
f	-0.830230	-0.770473	1.246615
g	NaN	NaN	NaN
h	0.085100	0.632791	0.987415

With reindexing, a DataFrame is made with the missing values in it. The output NaN implies it is “not a number”. To detect missing values in a DataFrame, Pandas provides two functions on DataFrame objects: **isnull()** and **notnull()**.

**Programming Example 5.1**

```
import pandas as pdimport numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f', 'h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df['one'].isnull()
```

The output is as follows:

```
a False
b True
c False
d True
e False
f False
g True
h False
Name: one, dtype: bool
```

**Programming Example 5.2**

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f', 'h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df['one'].notnull()
```

The output is as follows:

```
a True
b False
c True
d False
e True
f True
g False
h True
Name: one, dtype: bool
```

### 5.2.1.1 Calculations with Missing Data

When the sum of the data is calculated, the not a number (NaN) will be treated as zero. If the entries of data are all NaN, then the sum will be NaN.

#### *Programming Example 5.3*

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f', 'h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df['one'].sum()
```

The output is as follows:

2.02357685917
---------------

#### *Programming Example 5.4*

```
import pandas as pd
import numpy as np
df = pd.DataFrame(index = [0,1,2,3,4,5], columns = ['one','two'])
print df['one'].sum()
```

The output is as follows:

nan
-----

### 5.2.2 Cleaning and Filling Missing Data

Pandas provides different methods for filling the missing values. The **fillna** function can “fill in” NaN values with non-null data in different methods. This is described as follows:

#### 1. Replace NaN with a scalar value:

Replacing NaN with a different value is very easy to implement. In general, we tend to replace NaN with value 0. The following program illustrates how to replace NaN with 0:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index = ['a', 'c', 'e'],
columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print df
print ("NaN replaced with '0':")
print df.fillna(0)
```

The initial output is as follows:

	one	two	three
a	-0.576991	-0.741695	0.553172
b	NaN	NaN	NaN
c	0.744328	-1.735166	1.749580

The output of NaN replaced with 0 is as follows:

	one	two	three
a	-0.576991	-0.741695	0.553172
b	0.000000	0.000000	0.000000
c	0.744328	-1.735166	1.749580

Here, the NaN values are filled with zeros. It can be filled with any other value as well.

## 2. Fill NA forward and backward:

Forward and backward filling involves filling the missing data with the data objects above or below the NaN values. If the data object prior to the missing value is filled, then it is called forward filling; and if the data object next to the missing value is filled, then it is called backward filling.

Method	Action
pad/fill	Fill methods forward
bfill/backfill	Fill methods backward

### Programming Example 5.5

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f', 'h'],
columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.fillna(method = 'pad')
```

The output is as follows:

	one	two	three
a	0.057988	0.376149	0.865836
b	0.057988	0.376149	0.865836
c	-0.590208	-0.351605	-2.501950
d	-0.590208	-0.351605	-2.501950
e	-1.000303	-0.888201	2.510072
f	-0.830230	-0.770473	1.246615
g	-0.830230	-0.770473	1.246615
h	0.085100	0.632791	0.987415

**Programming Example 5.6**

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f','h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.fillna(method = 'backfill')
```

The output is as follows:

	one	two	three
a	0.057988	0.376149	0.865836
b	-0.590208	-0.351605	-2.501950
c	-0.590208	-0.351605	-2.501950
d	-1.000303	-0.888201	2.510072
e	-1.000303	-0.888201	2.510072
f	-0.830230	-0.770473	1.246615
g	0.085100	0.632791	0.987415
h	0.085100	0.632791	0.987415

**5.2.3 Drop Missing Values**

If the missing values have to be removed from the analysis, then we use the **dropna** function along with axis argument. By default, axis = 0 (i.e., along a row), which means that if any value within a row is NaN then the whole row is excluded.

**Programming Example 5.7**

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f','h'],
columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna()
```

The output is as follows:

	one	two	three
a	0.057988	0.376149	0.865836
c	-0.590208	-0.351605	-2.501950
e	-1.000303	-0.888201	2.510072
f	-0.830230	-0.770473	1.246615
h	0.085100	0.632791	0.987415

**Programming Example 5.8**

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index = ['a', 'c', 'e',
'f','h'], columns = ['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna(axis = 1)
```

The output is as follows:

Empty DataFrame
Columns: [ ]
Index: [a, b, c, d, e, f, g, h]

**5.2.3.1 Replace Missing (or) Generic Values**

A generic value can be replaced with a specific value. This is achieved by applying the *replace method*. Replacing NaN with a scalar value is the equivalent behavior of the **fillna( )** function.

**Programming Example 5.9**

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print df.replace({1000:10,2000:60})
```

The output is as follows:

	one	two
0	10	10
1	20	0
2	30	30
3	40	40
4	50	50
5	60	60

**5.2.4 Smoothing Noisy Data**

Data smoothing refers to techniques that eliminate unwanted noise or behaviors in data, while outlier detection identifies data points that are significantly different from the rest. There are, in general, three different methods for smoothing data. They are binning method, regression, and outlier analysis.

- Binning method:** Binning method smooths a sorted data value by consulting its “neighborhood” (i.e., the values around it). The sorted values are distributed into a number of bins. Because binning methods consult a neighborhood of values, they perform *local smoothing*.

$a = 41 \times 10 = 410$	$b = 41 \times 15 = 615$	$c = 41 \times 13 = 533$
$d = 41 \times 12 = 492$	$e = 41 \times 19 = 779$	$f = 41 \times 16 = 656$
$g = 41 \times 17 = 697$	$h = 41 \times 18 = 738$	$i = 41 \times 14 = 574$
$j = 41 \times 23 = 943$	$k = 41 \times 29 = 1189$	$l = 41 \times 26 = 1066$
$m = 41 \times 35 = 1435$	$n = 41 \times 24 = 984$	$o = 41 \times 25 = 1025$
$p = 41 \times 31 = 1271$	$q = 41 \times 32 = 1312$	$r = 41 \times 28 = 1148$
$s = 41 \times 13 = 533$	$t = 41 \times 20 = 820$	$u = 41 \times 34 = 1394$
$v = 41 \times 27 = 1107$	$w = 41 \times 33 = 1353$	$x = 41 \times 50 = 2050$
$y = 41 \times 43 = 1763$	$z = 41 \times 11 = 451$	$aa = 41 \times 37 = 1517$

**Figure 5.1** Income of 27 students, from  $a$  to  $aa$ .

Consider the following example: a set of 27 students' names (from  $a$  to  $aa$ ) and the number of hours each student worked a part time job is given in Fig. 5.1. The income for each student is the product of the hourly rate (i.e., \$41) and the number of hours worked.

We can smooth the data by using the *median binning technique*, as shown below.

- (a) **Sorting the data:** The raw data needs to be sorted before performing any binning methodology. The data given in Fig. 5.1 is sorted in ascending order, as shown below:

410, 451, 492, 533, 533, 574, 615, 656, 697, 738, 779, 820, 943, 984, 1025, 1066, 1107, 1148, 1189, 1271, 1312, 1353, 1394, 1435, 1517, 1763, 2050

- (b) **Partition into (equal-frequency) bins:** Once sorted, the data has to be transferred into prefixed number of bins. Here, we consider a bin of size 3. Partitioning is done as shown below:

Bin 1: 410, 451, 492  
 Bin 2: 533, 533, 574  
 Bin 3: 615, 656, 697  
 Bin 4: 738, 779, 820  
 Bin 5: 943, 984, 1025  
 Bin 6: 1066, 1107, 1148  
 Bin 7: 1189, 1271, 1312  
 Bin 8: 1353, 1394, 1435  
 Bin 9: 1517, 1763, 2050

- (c) **Smoothing by bin medians:** Now, each bin value is replaced by the bin median. For example, the median of values 410, 451, and 492 in Bin 1 is 451. Therefore, each original value in this bin is replaced by the value 451.

Bin 1: 451, 451, 451  
 Bin 2: 533, 533, 533  
 Bin 3: 656, 656, 656  
 Bin 4: 779, 779, 779  
 Bin 5: 984, 984, 984  
 Bin 6: 1107, 1107, 1107  
 Bin 7: 1271, 1271, 1271  
 Bin 8: 1394, 1394, 1394  
 Bin 9: 1763, 1763, 1763

- (d) **Smoothing by means:** Each value in a bin is replaced by the mean value of the bin. For example, the mean of values 410, 451, and 492 in Bin 1 is 451. Therefore, each original value in this bin is replaced by the value 451.

Bin 1: 451, 451, 451  
 Bin 2: 547, 547, 547  
 Bin 3: 656, 656, 656  
 Bin 4: 779, 779, 779  
 Bin 5: 984, 984, 984  
 Bin 6: 1107, 1107, 1107  
 Bin 7: 1257, 1257, 1257  
 Bin 8: 1394, 1394, 1394  
 Bin 9: 1777, 1777, 1777

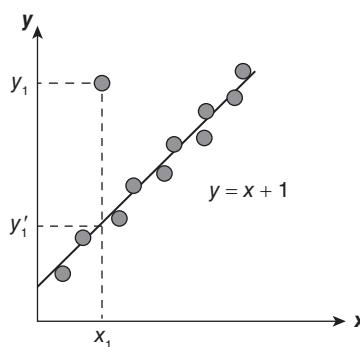
- (e) **Smoothing by boundaries (bin of size 4):** The minimum and maximum values in a given bin are identified as bin boundaries. Each bin value is then replaced by the closest boundary value. In general, the larger the width of the bin, the greater is the effect of smoothing.

Bin 1: 410, 451, 492, 533  
 Bin 2: 533, 574, 615, 656  
 Bin 3: 697, 738, 779, 820  
 Bin 4: 943, 984, 1025, 1066  
 Bin 5: 1107, 1148, 1189, 1271  
 Bin 6: 1312, 1353, 1394, 1435  
 Bin 7: 1517, 1763, 2050

After smoothing, we get:

Bin 1: 410, 410, 533, 533  
 Bin 2: 533, 533, 656, 656  
 Bin 3: 697, 697, 820, 820  
 Bin 4: 943, 943, 1066, 1066  
 Bin 5: 1107, 1107, 1271, 1271  
 Bin 6: 1312, 1312, 1435, 1435  
 Bin 7: 1517, 1517, 2050

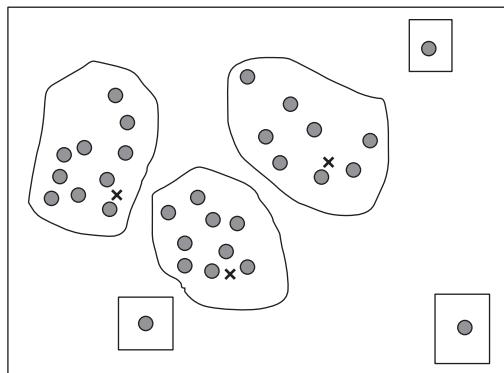
2. **Smoothing by regression:** Data smoothing can also be done by regression, a technique that conforms data values to a function. Linear regression involves finding the “best” line to fit two attributes (or variables) so that one attribute can be used to predict the other. The process of fitting data to a line is shown in Fig. 5.2.



**Figure 5.2** Smoothing using linear regression.

Multiple linear regression is an extension of linear regression, where more than two attributes are involved and the data is fit to a multidimensional surface. Regression is further described in detail.

- 3. Outlier analysis:** Outliers may be detected by clustering where similar values are organized into groups or “clusters.” Intuitively, values that fall outside the set of clusters may be considered outliers. In Fig. 5.3, the data points inserted in squares are considered outliers. The topic of clustering will be explained in detail.



**Figure 5.3** Outliers after clustering.

### 5.2.5 Removing Inconsistencies

In any database, there is usually data that is not consistent. Table 5.1 shows a database of books, with inconsistent data. In this case, the data is inconsistent because denominations of the price are in both rupees as well as dollars.

**Table 5.1** Database having inconsistent data

S. No.	Name of Book	Price	Publisher	Publication Year
1	Calculus I	₹90,000,00	Math Press	2007
2	Calculus II	₹91,000	Math Press	2008
3	Java Programming	\$7.50	Java Press	2010
4	Artificial Intelligence	\$8.80	Intel Press	2010
5	Speaking Art	₹10,000		
6	Bahasa Indonesia I	₹15,000	Indo Press	
7	Database Management System	₹10	Intel Press	2011

## 5.3

### Data Integration

Data integration involves combining the data from different sources using different technologies and different databases. The outcome of integration is a single unified view of the data. The biggest challenge in data integration

is the technical implementation of integrating data from different and incompatible data sources. This phase of preprocessing is done in the design phase and in the implementation phase.

In the design phase of data integration, the requirement of integration, source system analysis, non-functional requirements such as data processing window, system response time, and an estimation of concurrent users are first justified. All this is documented in the SRS with mutual consensus from all parties concerned. With the SRS in place, a feasibility study has to be performed to select the tools to implement the data integration system.

## 5.4 Data Transformation

Data transformation is the process of converting data from one format into another. It involves transforming the actual values from one representation to the target representation. Examples of data transformation tasks include mapping stock symbols to company names, changing date format from MM-DD to DD-MM, and replacing cities by their countries.

There are some transformations, such as litres to gallons, which can be performed by applying a formula or a program on the input values. Such types of transformations are referred to as *syntactic transformations*. Other types of transformations, such as company name to stock symbol and event to date, require finding the mappings between the input and output values in a repository of reference data. These types of transformation are referred to as *semantic transformations*.

## 5.5 Data Reduction or Dimensionality Reduction

Dimensionality reduction involves deriving new features from old ones. Generally, this is done by applying transforms to the dataset, which change the axes (coordinate system) of the graph by moving and rotating them. This can be written as a matrix that we apply to the data. It enables us to combine features and identify useful data.

### 5.5.1 Principal Component Analysis

Principal component analysis (PCA) is a useful statistical technique and is often used to find patterns in high dimensional datasets.

#### 5.5.1.1 Background Mathematics

The entire subject of statistics is based around the idea of analyzing a big set of data in terms of the relationships between the individual points. Some of the most common operations in statistics are given below:

- 1. Standard deviation:** There are a number of things that we can calculate in a dataset. For example, we can calculate the mean of the sample which is given by

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} \quad (5.1)$$

Standard deviation (SD) of a dataset is the measure of the spread of data. It is the average distance from the mean of the dataset to a point. The formula for SD is given in Eq. (5.2):

$$s = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n-1)}} \quad (5.2)$$

**Solved Problem 5.1**

Compute the SD for the following sets:

$$\text{Set 1} = [0 \ 8 \ 12 \ 20]$$

$$\text{Set 2} = [8 \ 9 \ 11 \ 12]$$

**Solution:**

Set 1:

$X$	$(X - \bar{X})$	$(X - \bar{X})^2$
0	-10	100
8	-2	4
12	2	4
20	10	100
Total		208
Divided by $(n-1)$		69.333
Square root		8.3266

Set 2:

$X_i$	$(X_i - \bar{X})$	$(X_i - \bar{X})^2$
8	-2	4
9	-1	1
11	1	1
12	2	4
Total		10
Divided by $(n-1)$		3.333
Square root		1.8257

The first set of values [0 8 12 20] has a larger standard deviation (i.e., 8.3266) when compared to the second set [8 9 11 12] (i.e., 1.8257). This is because the first set has a larger spread than the second.

- 2. Variance:** Variance is another measure of the spread of data in a dataset. It is almost identical to standard deviation. The formula for variance is given in Eq. (5.3).

$$s^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{(n-1)} \quad (5.3)$$

**3. Covariance:** Mean and SD are one-dimensional calculations. However, datasets have more than one dimension, and the aim of a statistical analysis of these datasets is usually to see if there is any relationship between these dimensions. Covariance is such a measure; it is always measured between two dimensions. If we calculate the covariance between one dimension and itself, we get the variance. The formula for covariance is given in Eq. (5.4).

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \quad (5.4)$$

Table 5.2 shows the data of the number of hours studied and the marks obtained by a group of students.

**Table 5.2** Number of hours studied and marks obtained by 12 students

<i>Student No.</i>	<i>Hours (H)</i>	<i>Marks (M)</i>
1	9	39
2	15	56
3	25	93
4	14	61
5	10	50
6	18	75
7	0	32
8	16	85
9	5	42
10	19	70
11	16	66
12	20	80
Total	167	749
Average	13.92	62.42

The covariance between the hours of study done and marks obtained is given in Table 5.3.

**Table 5.3** The covariance computation between hours and marks

<i>H</i>	<i>M</i>	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33

*(Continued)*

**Table 5.3** (Continued)

$H$	$M$	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

**Solved Problem 5.2**

Work out the covariance between the  $x$  and  $y$  dimensions in the following two-dimensional dataset. Describe what the result indicates about the data.

<i>Item No.</i>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>
$x$	10	39	19	23	28
$y$	43	13	32	21	20

**Solution:**

$x$	$y$	$(x - \bar{x}')$	$(y - \bar{y}')$	$(x - \bar{x}')(y - \bar{y}')$
10	43	-13.8	17.2	-237.36
39	13	15.2	-12.8	-194.56
19	32	-4.8	6.2	-29.76
23	21	-0.8	-4.8	3.84
28	20	4.2	-5.8	-24.36
23.8	25.8			-120.55

The exact value is not as important as its sign (i.e., positive or negative). If the value is positive, as shown in Table 5.3, then it indicates that both dimensions increase together. This means that, in general, the final mark increased with increase in the number of hours of study. If the value is negative, then as one dimension increases, the other decreases. If the covariance is zero, it indicates that the two dimensions are independent of each other.

**Covariance matrix:** Covariance is always measured between two dimensions. If we have a dataset with more than two dimensions, then more than one covariance measurement can be calculated. For example, from a three-dimensional dataset (dimensions  $x, y, z$ ), we can calculate  $\text{cov}(x, y)$ ,  $\text{cov}(x, z)$ , and  $\text{cov}(y, z)$ . In

fact, for an  $n$ -dimensional dataset, we can calculate  $\frac{n!}{((n-2)!\times 2)}$  different covariance values.

The definition for the covariance matrix of a set of data with  $n$ -dimensions is:

$$C^{n \times n} = \left( c_{i,j}, c_{i,j} = \text{cov}(\text{Dim}_i, \text{Dim}_j) \right)$$

If we have an  $n$ -dimensional dataset, then the matrix has  $n$  rows and  $n$  columns (i.e., it is a square matrix). Each entry in the matrix is the result of calculating the covariance between two separate dimensions. The covariance matrix for an imaginary three-dimensional dataset is given in Eq. (5.5).

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{pmatrix} \quad (5.5)$$

#### Points to remember

- Down the main diagonal, the covariance values are the variances for that dimension.
- Since  $\text{cov}(x, y) = \text{cov}(y, x)$ , the matrix is symmetrical about the main diagonal.

**4. Eigenvalues and Eigenvectors:** Let  $A$  be an  $n \times n$  matrix. A scalar  $\lambda$  is called an *eigenvalue* of  $A$  if there is a non-zero vector  $x$  such that  $Ax = \lambda x$ . Such a vector  $x$  is called an *eigenvector* of  $A$  corresponding to  $\lambda$ .

(a) **Finding eigenvalues:** To find the eigenvalues of matrix  $A$ , we have to find the values of  $\lambda$  which satisfy the characteristic equation of the matrix, namely those values of  $\lambda$  that satisfy Eq. (5.6).

$$\det(A - \lambda I) = 0 \quad (5.6)$$

where  $I$  is the identity matrix.

The steps in finding the eigenvalue of matrix  $A$  are as follows:

- Form the matrix  $A - \lambda I$
- Calculate  $\det|A - \lambda I|$
- Find solutions to  $\det|A - \lambda I| = 0$ , which gives the eigenvalue.

(b) **Finding eigenvectors:** To find the eigenvectors, we have to compute the following steps:

- Find  $x$  by Gaussian elimination.
- Convert the augmented matrix  $(A - \lambda I : 0)$  to row echelon form.
- Solve the resulting linear system by back substitution.

#### Solved Problem 5.3

Find the eigenvalues and eigenvectors for the matrix  $A$  given below.

$$A = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$$

**Solution:**

$$A = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$$

$$\lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$A - \lambda I = \begin{bmatrix} 7 - \lambda & 3 \\ 3 & -1 - \lambda \end{bmatrix}$$

$$|A - \lambda I| = \lambda^2 - 6\lambda - 16$$

$$\lambda^2 - 6\lambda - 16 = 0 \geq (\lambda - 8)(\lambda + 2) = 0$$

The eigenvalues are:  $\lambda_1 = 8$  and  $\lambda_2 = -2$

Eigenvector for  $\lambda_1 = 8$

We have,

$$A - \lambda I = \begin{bmatrix} 7 - \lambda & 3 \\ 3 & -1 - \lambda \end{bmatrix}$$

Substitute  $\lambda_1 = 8$ ,

$$\text{Say } B = A - 8I = \begin{bmatrix} -1 & 3 \\ 3 & -9 \end{bmatrix}$$

Solve  $Bx = 0$

$$\begin{bmatrix} -1 & 3 \\ 3 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

Compute row cancellation of R2:  $R2 \leftarrow 3R1 + R2$

$$\begin{bmatrix} -1 & 3 \\ 0 & 0 \end{bmatrix}$$

$$-x_1 + 3x_2 = 0$$

$$x_1 = 3x_2$$

Let  $x_2 = 1$ , then  $x_1 = 3$

$$\begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Check  $Ax = \lambda x$

$$Ax = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 24 \\ 8 \end{bmatrix}$$

$$\lambda x = 8 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 24 \\ 8 \end{bmatrix}$$

Eigenvector for  $\lambda_2 = -2$

We have,

$$A - \lambda I = \begin{bmatrix} 7 - \lambda & 3 \\ 3 & -1 - \lambda \end{bmatrix}$$

Substitute  $\lambda_2 = -2$

$$\text{Say } C = A - (-2)I = \begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix}$$

Solve  $Cx = 0$

$$\begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

Compute row cancellation of R1 : R1  $\leftarrow$  R2 + R1

$$\begin{bmatrix} 0 & 0 \\ 3 & 1 \end{bmatrix}$$

$$3x_1 + x_2 = 0$$

$$3x_1 = -x_2$$

Let  $x_1 = 1$ , then  $x_2 = -3$

$$\begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

Check  $Ax = \lambda x$

$$Ax = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -3 \end{bmatrix} = \begin{bmatrix} -2 \\ 6 \end{bmatrix}$$

$$\lambda x = -2 \times \begin{bmatrix} 1 \\ -3 \end{bmatrix} = \begin{bmatrix} -2 \\ 6 \end{bmatrix}$$

### 5.5.1.2 Computation of PCA

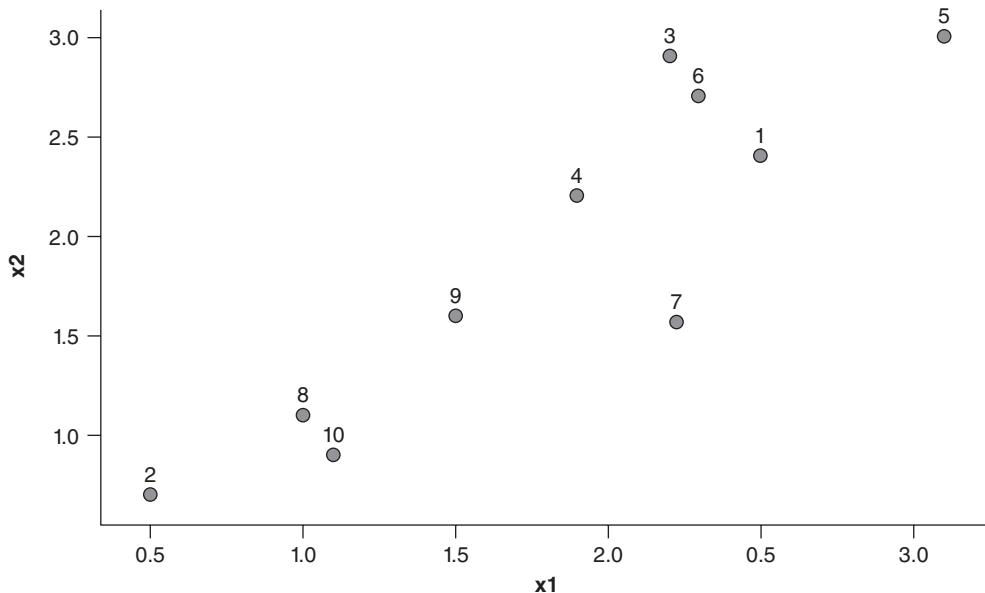
The five steps to compute PCA are discussed below:

- 1. Standardization of data:** To standardize data, we must compute a number of steps. Let us take the example of a two-dimensional dataset (Table 5.4).

**Table 5.4** Two-dimensional dataset

$x1$	$x2$
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

- (a) Make a plot of the data as shown in Fig. 5.4. Calculate the mean of  $x1$  and  $x2$ .

**Figure 5.4** Scatterplot of the points with respect to  $x1$  and  $x2$ .

Correlation coefficient,  $k = 0.926$

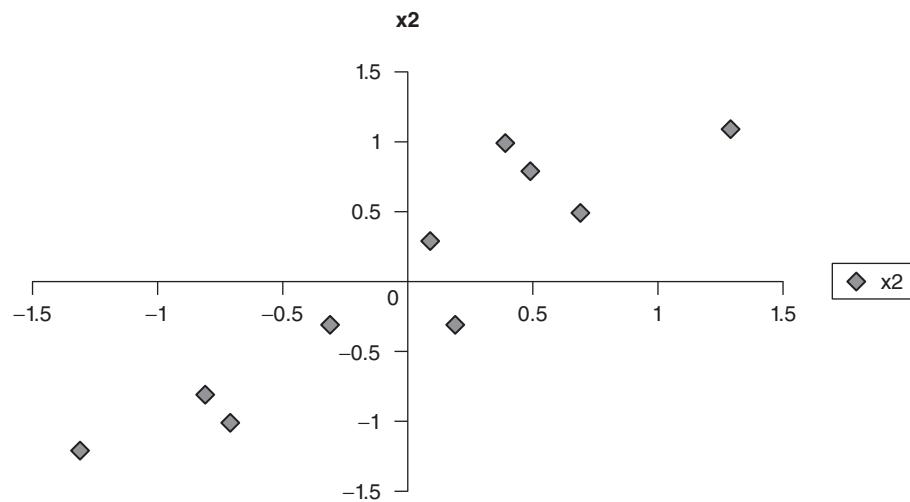
Mean of  $x1 = 1.81$

Mean of  $x2 = 1.91$

- (b) Subtract the means from the corresponding data component to recenter the dataset (Table 5.5) and reconstruct the scatterplot (Fig. 5.5).

**Table 5.5** The dataset after recentering

$x1$	$x2$
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01



**Figure 5.5** Scatterplot of the points of  $x1$  and  $x2$  with deviation from mean.

- (c) Write the “adjusted” data as a matrix. Note that the “adjusted” dataset will have a means of zero (Table 5.6). The spread of deviation is shown in Fig. 5.6.

**Table 5.6** The adjusted dataset

<b>x1</b>	<b>x2</b>
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

**2. Obtain the eigenvalues  $\lambda$  and eigenvectors  $x$ :** To obtain the eigenvalues  $\lambda$  and eigenvectors  $x$ , the singular vector decomposition will have to be computed:

(a) Compute the sample variance-covariance matrix  $C$ .

$$C = \frac{1}{N-1} (X - \bar{X})' (X - \bar{X}) = \frac{1}{N-1} X'X$$

(b) If the data is standardized, then  $C$  is the correlation matrix.

$$C = \frac{1}{N-1} Z'Z$$

$$C = \frac{1}{9} \begin{bmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{bmatrix}$$

$$C = \frac{1}{9} \begin{bmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{bmatrix}^T$$

$$C = \begin{bmatrix} 0.616556 & 0.615444 \\ 0.615444 & 0.716556 \end{bmatrix}$$

Since the non-diagonal elements in this covariance matrix are positive, both  $x1$  and  $x2$  variables increase together. Since it is symmetric, the eigenvectors are orthogonal.

- (c) Compute the eigenvalues  $\lambda$  and eigenvectors  $x$  (unit or normalized) of the correlation matrix  $C$ :

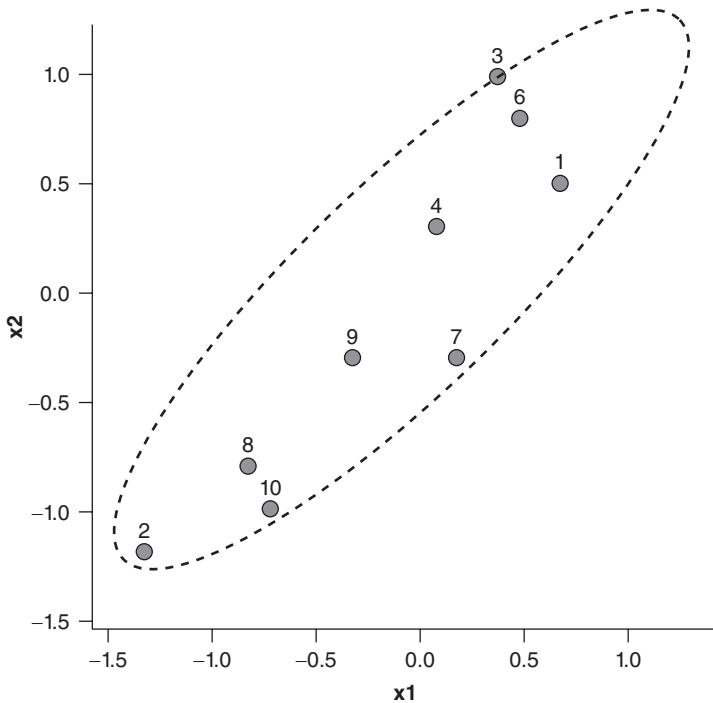
The eigenvalues are:

$$\lambda_1 = 1.28403 \text{ and } \lambda_2 = 0.0490834$$

The eigenvectors are:

$$\begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix} \text{ for } \lambda_1 = 1.28403$$

$$\begin{bmatrix} 0.735179 \\ -0.677873 \end{bmatrix} \text{ for } \lambda_2 = 0.0490834$$



**Figure 5.6** Spread of deviation.

The total sample variance (trace) = sum of eigenvalues

$$= 1.2840 + 0.0490 = 1.333$$

It can be seen that approximately 96% of the total variance is concentrated in eigenvector 1 and 4% in eigenvector 2, as shown in Table 5.7.

**Table 5.7** Energy compaction with PCA

<i>Variable</i>	<i>Eigenvector 1</i>	<i>Eigenvector 2</i>
$x_1$	0.678	0.735
$x_2$	0.735	-0.678
Eigenvalues	1.2840	0.0490
% of total variance	1.2840/1.333 = (96.3%)	0.0490/1.333 = (3.7%)

3. **Sort eigenvalues in descending order:** Once eigenvectors are computed from the covariance matrix, they will have to be arranged by eigenvalues — from highest to lowest. This gives the components in order of significance. The eigenvector with the highest eigenvalue is the principal component of the dataset. Components of lesser significance can be ignored. Some information may be lost; but if the eigenvalues are small, then not much is lost. If some components are left out, the final dataset will have less dimensions than the original. To be precise, if originally there are  $n$  dimensions in the data and  $p$  eigenvectors and eigenvalues are calculated, and only the first  $r$  eigenvectors are chosen, then the final dataset will have only  $r$  dimensions.
4. **Form a feature vector and construct the projection matrix  $W$ :** Projection matrix  $W$  is constructed by taking the eigenvectors that are chosen and forming a matrix with them in columns.

Given the example set of data and the fact that there are two eigenvectors, there are two choices of matrices. A feature vector can be formed with both of the eigenvectors:

$$V_1 = \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677873 \end{bmatrix}$$

If the smaller, less significant component is left out, then there will be a single column:

$$V_2 = \begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix}$$

5. **Transform the original dataset:** The new dataset is derived by taking  $Z = XV$ . Basically, the data is transformed so that it is expressed in terms of the patterns between them, where the patterns are the lines that most closely describe the relationships between the data.

#### Case 1:

$$V_1 = \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677873 \end{bmatrix}$$

Final data =  $XV_1$

$$\begin{aligned} &= \begin{bmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{bmatrix}^T \\ &\times \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677873 \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} 0.82797 & 0.175116 \\ -1.77758 & -0.142858 \\ 0.99220 & -0.384374 \\ 0.27421 & -0.130417 \\ 1.67580 & 0.209499 \\ 0.91295 & -0.175282 \\ -0.09911 & 0.349825 \\ -1.14457 & -0.046418 \\ -0.43805 & -0.017765 \\ -1.22382 & 0.162675 \end{bmatrix}$$

**Case 2:**

$$V_2 = \begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix}$$

Final data =  $XV_2$ 

$$= \begin{bmatrix} 0.69 & 0.49 \\ -1.31 & -1.21 \\ 0.39 & 0.99 \\ 0.09 & 0.29 \\ 1.29 & 1.09 \\ 0.49 & 0.79 \end{bmatrix} \times \begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix}$$

$$= \begin{bmatrix} 0.82797 \\ -1.77758 \\ 0.99220 \\ 0.27421 \\ 1.67580 \\ 0.91295 \\ -0.09911 \\ -1.14457 \\ -0.43805 \\ -1.22382 \end{bmatrix}$$

### 6. Reconstruction of data:

**Case 1:**

$$V_1 = \begin{bmatrix} 0.677873 & -0.735179 \\ 0.735179 & 0.677873 \end{bmatrix}$$

$$X = ZV_1^T$$

Row zero mean data = Final data  $XV_1^T$

$$\begin{aligned}
 &= \begin{bmatrix} 0.82797 & 0.175116 \\ -1.77758 & -0.142858 \\ 0.99220 & -0.384374 \\ 0.27421 & -0.130417 \\ 1.67580 & 0.209499 \\ 0.91295 & -0.175282 \\ -0.09911 & 0.349825 \\ -1.14457 & -0.046418 \\ -0.43805 & -0.017765 \\ -1.22382 & 0.162675 \end{bmatrix} \times \begin{bmatrix} 0.677873 & 0.735179 \\ 0.735179 & -0.677873 \end{bmatrix} \\
 &= \begin{bmatrix} 0.69 & 0.49 \\ -1.31 & -1.21 \\ 0.39 & 0.99 \\ 0.09 & 0.29 \\ 1.29 & 1.09 \\ 0.49 & 0.79 \\ 0.19 & -0.31 \\ -0.81 & -0.81 \\ -0.31 & -0.31 \\ -0.71 & -1.01 \end{bmatrix}
 \end{aligned}$$

Row original data = Row zero mean data + Original mean

$$\text{Row original data} = \begin{bmatrix} 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \\ 1.9 & 2.2 \\ 3.1 & 3.0 \\ 2.3 & 2.7 \\ 2.0 & 1.6 \\ 1.0 & 1.1 \\ 1.5 & 1.6 \\ 1.1 & 0.9 \end{bmatrix}$$

**Case 2:**

$$V_2 = \begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix}$$

$$X = ZV_2^T$$

Row zero mean data = Final data  $XV_2^T$ 

$$= \begin{bmatrix} 0.82797 \\ -1.77758 \\ 0.99220 \\ 0.27421 \\ 1.67580 \\ 0.91295 \\ -0.09911 \\ -1.14457 \\ -0.43805 \\ -1.22382 \end{bmatrix} \times \begin{bmatrix} 0.677873 \\ 0.735179 \end{bmatrix}^T$$

$$= \begin{bmatrix} 0.82797 \\ -1.77758 \\ 0.99220 \\ 0.27421 \\ 1.67580 \\ 0.91295 \\ -0.09911 \\ -1.14457 \\ -0.43805 \\ -1.22382 \end{bmatrix} \times [0.677873 \quad 0.735179]$$

$$= \begin{bmatrix} 0.56126 & 0.60870 \\ -1.201497 & -1.30684 \\ 0.67258 & 0.72994 \\ 0.18558 & 0.20159 \\ 1.13598 & 1.23201 \\ 0.61886 & 0.67118 \\ -0.06718 & -0.07286 \\ -0.77587 & -0.84146 \\ -0.29694 & -0.32204 \\ -0.82959 & -0.899731 \end{bmatrix}$$

Row original data = Row zero mean data + Original mean

$$\text{Row original data} = \begin{bmatrix} 0.56126 & 0.60870 \\ -1.201497 & -1.30684 \\ 0.67258 & 0.72994 \\ 0.18558 & 0.20159 \\ 1.13598 & 1.23201 \\ 0.61886 & 0.67118 \\ -0.06718 & -0.07286 \\ -0.77587 & -0.84146 \\ -0.29694 & -0.32204 \\ -0.82959 & -0.899731 \end{bmatrix} + \begin{bmatrix} 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \\ 1.81 & 1.91 \end{bmatrix}$$

$$= \begin{bmatrix} 2.37126 & 2.51870 \\ 0.60503 & 0.60316 \\ 2.48258 & 2.63994 \\ 1.99558 & 2.11159 \\ 2.94598 & 3.14201 \\ 2.42886 & 2.58118 \\ 1.74282 & 1.83714 \\ 1.03414 & 1.06854 \\ 1.51306 & 1.58796 \\ 0.98041 & 1.01027 \end{bmatrix}$$

It is seen that if only the first eigenvector is considered, then the data can be reconstructed similar to the original dataset.

### 5.5.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is most commonly used as dimensionality reduction technique. It is very similar to the PCA approach. If PCA involves finding the component axes that maximize the variance of the entire data, LDA involves finding the axes that maximize the separation between multiple classes. Thus, LDA projects a feature space of size  $n$  onto a smaller subspace  $k$  [where  $k \leq (n-1)$ ] while maintaining the class-discriminatory information.

### 5.5.3 Independent Component Analysis

Independent component analysis (ICA) is a method that extracts the factors or components from multi-dimensional statistical data. It extracts the components that are non-Gaussian and statistically independent. Consider a dataset in which the observed data contains a number of variables. These variables can be denoted by  $m$  and the number of observations by  $T$ . The data can be represented as  $x_i(t)$  where the indices take the values  $i = 1, 2, \dots, m$  and  $T = 1, 2, \dots, T$ . ICA involves forming a function from an  $m$ -dimensional space to an  $n$ -dimensional space. The transformed variables can give information that is hidden in the large dataset. Linear functions are mostly considered since the interpretations and computations are relatively easier than other functions. Thus, every component ( $y_i$ ) is expressed as a linear combination of the observed variable, as shown in Eq. (5.7).

$$y_i(t) = \sum_j w_{ij} x_j(t), \text{ for } i = 1, 2, \dots, n, j = 1, 2, \dots, m \quad (5.7)$$

where  $w_{ij}$  is the coefficient that defines the representation.

Thus, the problem revolves around the determination of coefficient  $w_{ij}$ . The linear equations derived can be expressed in terms of matrices, as shown in Eq. (5.8). Collecting the coefficients  $w_{ij}$  in a matrix  $W$ , the equation becomes a form of matrix multiplication.

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{pmatrix} = W \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_m(t) \end{pmatrix} \quad (5.8)$$

Consider the input  $x_i(t)$  as a set of  $T$  realizations of  $m$  random variables. Then each set  $x_i(t)$  (where  $t = 1, \dots, T$  is a sample of one random variable) is denoted by the random variable  $x_i$ . The matrix  $W$  can then be determined by the statistical properties of transformed components  $y_i$ .

#### 5.5.3.1 Other Dimension Reduction Methods

The decision to choose matrix  $W$  is to limit the number of components  $y_i$  to be quite a small number (say, 1 or 2), and to determine  $W$  so that the  $y_i$  contains as much information on the data as possible. This method is termed as *principal component analysis* (as described in Section 5.4.1). Another way of finding  $W$  is *independence*: in this case the components  $y_i$  should be statistically independent. This implies that the value of any one component gives no information on the values of the other components. In PCA, it is often claimed that the factors are independent, but this is only partly true, because PCA analysis assumes that the data has a Gaussian distribution. If the data is Gaussian, it is simple to find components that are independent, because uncorrelated components are always independent in Gaussian data. In reality, the data does not follow a Gaussian distribution; hence we need ICA to find statistically independent components when data is non-Gaussian.

### 5.5.3.2 Blind Source Separation

Blind source separation is a technique used to extract original signal from a mixture of signals. The process of extraction of signals is explained in detail in the following sections.

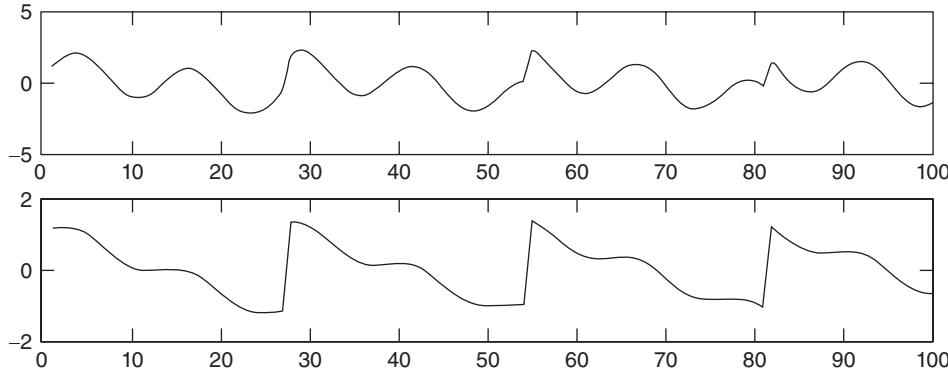
#### 5.5.3.2.1 Observing Mixtures of Unknown Signals

Consider a situation where a number of signals are emitted by some independent sources. Assume further that there are several sensors or receivers kept in different positions, so that each records a mixture of the original source signals with slightly different weights.

Let us say there are two underlying source signals denoted by  $s_1(t)$  and  $s_2(t)$ , and also two observed signals denoted by  $x_1(t)$  and  $x_2(t)$ . Then  $x_i(t)$  is the weighted sums of  $s_i(t)$ , where the coefficients depend on the distances between the sources and the sensors as shown in Eq. (5.9).

$$\begin{aligned}x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) \\x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t)\end{aligned}\quad (5.9)$$

The constant coefficients,  $a_{ij}$ , correspond to the weights associated with each source. They are unknown since the values of  $a_{ij}$  cannot be determined without knowing the properties of the mixing elements, which is extremely difficult in general. The source signals  $s_i$  are unknown as well, since they cannot be recorded directly.



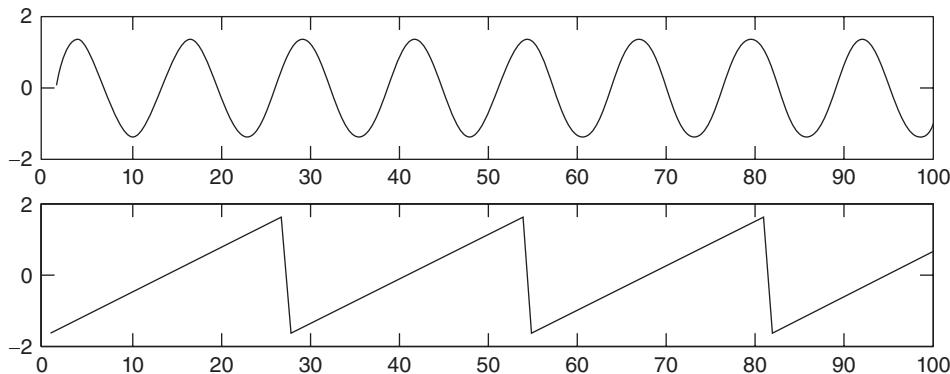
**Figure 5.7** Linear mixtures.

Consider the waveforms shown in Fig. 5.7. These are two linear mixtures  $x_i$  of some original source signals. Even though they look as if they were completely noise, they are somewhat structured and have some underlying source signals hidden in them. The problem to find the original signals from the mixtures  $x_1(t)$  and  $x_2(t)$  is called *blind source separation* (BSS) problem. It is called blind as very little is known about the original sources. Assume that the coefficients  $a_{ij}$  are different enough to make the matrix they form an invertible matrix. Thus, there is a matrix  $W$  with coefficients  $w_{ij}$  where  $s_i$  can be separated as shown in Eq. (5.10).

$$\begin{aligned}s_1(t) &= w_{11}x_1(t) + w_{12}x_2(t) \\s_2(t) &= w_{21}x_1(t) + w_{22}x_2(t)\end{aligned}\quad (5.10)$$

Matrix  $W$  is the inverse of the matrix that consists of the mixing coefficients  $a_{ij}$ . Thus, the problem is similar to finding a good representation for the random data in  $x_i(t)$ .

Figure 5.8 displays the graphs when the sources are segregated.



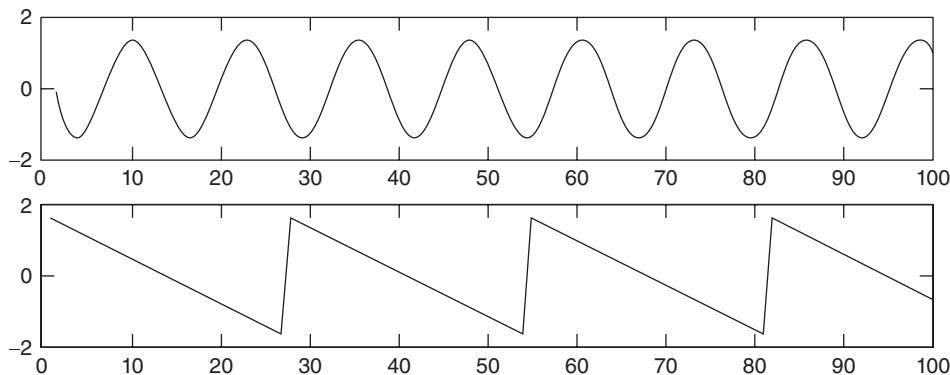
**Figure 5.8** Separated sources.

#### 5.5.3.2.2 Source Separation Based on Independence

Now we have to find a way to estimate the coefficients  $w_{ij}$ . We need a general method that provides a single answer to finding a good representation of multivariate data. One of the solutions to this problem is to consider the statistical independence of the signals. In fact, if the signals are not Gaussian, it is easy to determine the coefficients  $w_{ij}$ , so that the signals shown in Eq. (5.11) are statistically independent.

$$\begin{aligned} y_1(t) &= w_{11}x_1(t) + w_{12}x_2(t) \\ y_2(t) &= w_{21}x_1(t) + w_{22}x_2(t) \end{aligned} \quad (5.11)$$

If the signals  $y_1$  and  $y_2$  are independent, then they are equal to the original signals  $s_1$ , and  $s_2$ . Using this information on the statistical independence, we can estimate the coefficient matrix  $W$  for the signals as shown in Fig. 5.9.



**Figure 5.9** The independent signals obtained from the mixture.

### 5.5.3.3 Solution to Blind Source Separation Problem

The problem of BSS is nothing but finding a linear representation in which the components are statistically independent. In practical situations, we cannot in general find a representation where the components are really independent. Thus, we try to find at least components that are as independent as possible. For a set of given observations of random variables  $x_1(t), x_2(t), \dots, x_n(t)$ , where  $t$  is the time or sample index, let us assume that they are generated as a linear mixture of independent components.

$$\begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} = A \begin{pmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{pmatrix} \quad (5.12)$$

It is clear that  $A$  is an unknown matrix; the matrix equivalent of linear equations is given in Eq. (5.12). Thus, ICA involves estimation of both the matrix  $A$  and the  $s_i(t)$ , when we only observe the  $x_i(t)$ .

### 5.5.3.4 Methods for Estimating ICA

There are several methods of estimating ICA. The most popular among these methods are:

- 1. Nonlinear decorrelation:** Nonlinear decorrelation method involves finding the matrix  $W$  so that for any  $i \neq j$ , the components  $y_i$  and  $y_j$  are completely uncorrelated. The transformed components  $g(y_i)$  and  $h(y_j)$  are also uncorrelated, where  $g$  and  $h$  are some suitable nonlinear functions. In this method of estimating ICA, if the nonlinearities are properly chosen, the method does find the independent components. The main problem in this method is to address how the nonlinearities  $g$  and  $h$  are chosen. One of the approaches to select the nonlinear functions is to use *maximum likelihood method* in information theory.
- 2. Maximum non-Gaussianity:** The maximum non-Gaussianity approach for estimating independent component involves finding the local maxima of non-Gaussianity of a linear combination,  $y = \sum b_i x_i$ , under the constraint that the variance of  $y$  is constant. Each local maximum corresponds to one independent component. In practice, *kurtosis* is used to measure non-Gaussianity. Kurtosis is a higher order cumulate method, which involves some ways of generalizations of variance using higher order polynomials. Cumulants are used for ICA as they have important algebraic and statistical properties.

## Summary

- Data preprocessing is a data mining technique that involves transforming raw data into an understandable format.
- Data preprocessing methods are divided into data cleaning, data integration, data transformation, and data reduction.
- Data cleaning can be applied to remove noise and correct inconsistencies.
- Data integration merges data from multiple sources into a coherent data store.
- Data reduction can reduce the data size by aggregating and eliminating redundant features.
- Data cleaning routines can be used to fill in missing values, smooth noisy data, identify outliers and correct data inconsistencies.
- Python Pandas provides features like `fillna`, replacing the null values with a scalar value, forward padding or backward padding, `dropna` function to drop null values and replacing missing values by generic values; which helps in eliminating the concept of missing values.
- Data smoothing refers to techniques for eliminating unwanted noise or behaviors in data. Binning methods can be employed to smoothen data.

- The three major binning methods are partition using equal-frequency bins, smoothing by bin means, smoothing by means and by boundaries. Smoothing can also be performed by using the concept of regression.
- Outlier analysis is used as part of data preprocessing to identify the data points that do not fall in a cluster and eliminate them as an outlier.
- Other inconsistencies present in the data like datatype mismatch has to be standardized before the processed data goes for data reduction or dimensionality reduction techniques.
- The three most important dimensionality reduction techniques are principal component analysis, linear discriminant analysis and independent component analysis.
- Principal component analysis (PCA) is a useful statistical technique and is used for finding patterns in high dimensional data. It requires a fair knowledge of the concepts of mean, standard deviation, variance, eigenvalues and eigenvectors.
- PCA consists of the following steps: standardizing the data, computing the eigenvectors and eigenvalues from the covariance matrix or correlation matrix, sorting the eigenvalues in descending order and choosing the  $r$  eigenvectors that correspond to the  $r$  largest eigenvalues, constructing the projection matrix  $W$  from the selected  $r$  eigenvectors, and transforming the original dataset  $X$  via  $W$  to obtain a  $r$ -dimensional feature subspace  $Y$ .
- Linear discriminant analysis (LDA) is most commonly used as dimensionality reduction technique in the preprocessing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid over fitting (curse of dimensionality) and also reduce computational costs. In the LDA approach, the axes that maximize the separation between multiple classes are computed in addition to finding the component axes that maximize the variance of the data.
- Independent component analysis (ICA) is a method for finding underlying factors or components from multivariate statistical data. It looks for components that are both statistically independent and non-Gaussian.
- Statistically independent and non-Gaussian components can be separated using a blind source separation method. Nonlinear decorrelation and maximum non-Gaussianity are the methods used in ICA.

## Multiple-Choice Questions

1. The first factor in a solution has an eigenvalue of 3.87. The total of all the eigenvalues is 9. What proportion of variance is accounted for by this first factor?
  - (a) 43%
  - (b) 34.83%
  - (c) 38.70%
  - (d) 11.10%
2. A principal component analysis was run and the following eigenvalue results were obtained: 2.731, 2.218, 0.442, 0.341, 0.183, 0.085. How many factors would you retain using the eigenvalues to retain 45% of variance?
  - (a) 1
  - (b) 2
  - (c) 4
  - (d) 6
3. Which of the following is an alternative technique to principal component analysis?
  - (a) Factor analysis
  - (b) Independent component analysis
  - (c) Latent semantic analysis
  - (d) All of the above
4. Which of the following is a measure of the spread of data?
  - (a) Mean
  - (b) Variance
  - (c) Covariance
  - (d) All of the above

5. Which of the following is not a data preprocessing method?  
 (a) Data cleaning  
 (b) Data integration  
 (c) Data transformation  
 (d) All of the above

### Very Short Answer Questions

---

1. Why is data preprocessing indispensable?
2. What are the different options available for cleaning and filling missing data?
3. Why is data smoothed?
4. Why is normalization of variables necessary?
5. How is data integration accomplished? What are the problems associated with data integration?

### Short Answer Questions

---

1. What is the difference between covariance and correlation?
2. How many covariance values have to be computed for a five-dimensional data?
3. What are the methods of estimating ICA?
4. How is mean deviation, standard deviation, and variance related to each other?

### Review Questions

---

1. Enumerate the different techniques used in data cleaning.
2. Explain the different types of binning methods using in data cleaning.
3. Indicate the different types of transformations the data has to be subjected to before dimensionality reduction techniques can be applied.
4. Compute the standard deviation for the series:  
 (a) 3, 5, 2, 7, 6, 4, 9. ( $\sigma = \sqrt{4.978} = 2.231$ )  
 (b) 3, 5, 2, 7, 6, 4, 9, 1. ( $\sigma = \sqrt{6.234} = 2.497$ )
5. Calculate the covariance matrix for this three-dimensional set of data.

<i>Item No.</i>	<i>1</i>	<i>2</i>	<i>3</i>
<i>x</i>	1	-1	4
<i>y</i>	2	1	3
<i>z</i>	1	3	-1

6. Find the eigenvalues and eigenvectors for the matrix given below:  

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 3 & 3 & 20 \end{bmatrix}$$
7. Illustrate the concept of PCA with an appropriate example.
8. Describe the steps followed in LDA.
9. Describe the ICA and the blind source separation problem.

### Answers

---

#### Multiple-Choice Questions

1. (c)    2. (b)    3. (d)    4. (b)    5. (d)



## PART 2

# Supervised Learning Algorithms

---

### **Chapter 6**

Artificial Neural Networks

### **Chapter 7**

Linear Regression

### **Chapter 8**

Logistic Regression

### **Chapter 9**

Decision Tree

### **Chapter 10**

Support Vector Machines

### **Chapter 11**

Bayesian Classification

### **Chapter 12**

Hidden Markov Model



# 6

# Artificial Neural Networks

## LEARNING OBJECTIVES

- To introduce the concept of artificial neural networks.
- To compare the workings of biological neurons versus artificial neurons.
- To introduce different types of learning.
- To introduce the different architectures of artificial neural networks.
- To use McCulloch–Pitts model for solving basic classification problems.

## LEARNING OUTCOMES

- Students will be able compare biological neurons with artificial neurons.
- Students will be able compare the use of activation function of biological neurons and artificial neurons.
- Students will be able to discriminate between supervised and unsupervised models.
- Students will be able use McCulloch–Pitts model for solving basic logical operations like AND, OR, and NOT. Students will be able to solve nonlinear problems like EXOR problems using this model.

### 6.1 Introduction

Modern digital computers are truly astounding in the matter of power and speed. Humans are not so intelligent to compute millions of mathematical operations in a second. Humans cannot search a particular document from among the millions in a computer, whereas a computer can do it in milliseconds. However, there are some tasks where even the most powerful computers cannot compete with the human brain. Human beings are good in narration, while computers are good in logic and mathematics. The art of storytelling which humans have is not possessed by a computer. Imagine the power of a machine if it can accommodate capabilities of both computers and humans. It would turn out to be the most remarkable invention of all time. This is the aim of artificial intelligence, in general.

#### 6.1.1 Introduction to Artificial Neural Networks

A neural network's ability to perform computations is based on the hope that we can reproduce some of the flexibility and power of the human brain by artificial means. It tries to mimic the structure and function of our nervous system. An artificial neural network (ANN) is used as a methodology for information processing and the method got its inspiration from biological nervous systems. These systems consist of

innumerable highly interconnected neurons working together to solve different kinds of problems. Learning in biological systems takes place by adjusting the synaptic connections that exist between the neurons. ANN learns mostly by example and thus tries to emulate this structure.

### 6.1.2 Use of Artificial Neural Networks

Neural networks are highly capable of deriving information from complicated or imprecise data. They can extract patterns and detect trends which are too complex for humans or computers to perform. A trained network can work as an expert to analyze the given data to extract information. The information can be extracted from newly obtained data by comparing its characteristics to the trained data. Not only can ANN be thought of as an expert, it has other advantages as well.

- 1. Adaptive Learning:** Neural networks have the ability to do tasks after learning from experience gained from previous data.
- 2. Self-Organization:** ANN is capable of organizing and representing the information it receives from training data.
- 3. Real-Time Operation:** ANN computations can be carried out in parallel. Special hardware devices can be designed and manufactured so that we can take advantage of this capability.
- 4. Fault Tolerance:** If a neuron fails to work, the performance of the network will not stop, but it will give less accurate results.

## 6.2 Evolution of Neural Networks

The evolution of neural networks dates way back to the 1870s. Different theories were invented from 1870 onwards, which is shown in Table 6.1. Starting from a single neuron, the evolution of ANN started centuries before; right now we are in the era of deep learning.

**Table 6.1** Evolution of neural networks

Year	Theory Name	Inventor	Features
1871–73	Reticular theory	Joseph von Gerlach	Nervous system is a single continuous network.
1888–91	Neuron doctrine	Santiago Ramon y Cajal	Golgi's technique to study the nervous system. Proposed that it is actually made up of discrete individual cells forming a network.
1891	Neuron term coined	Heinrich Wilhelm Gottfried von Waldeyer-Hartz	Consolidation of neuron doctrine
1950	Neuron doctrine was accepted	Visualized using Electron Microscope	Nerve cells were individual cells interconnected through synapses. This was found by electron microscope.
1943	McCulloch–Pitts Neuron	McCulloch and Pitts	Simplified model of a neuron.
1957–58	Perceptron	Frank Rosenblatt	The perceptron may be able to learn, make decisions, and translate languages.

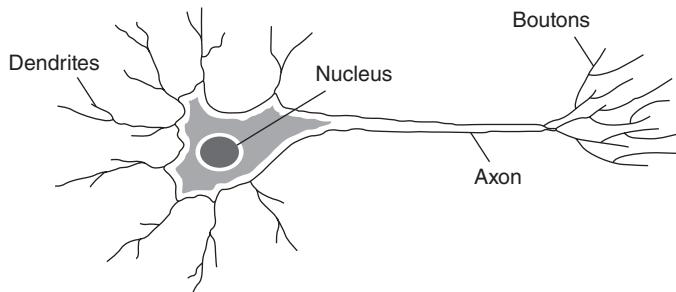
(Continued)

**Table 6.1** (Continued)

Year	Theory Name	Inventor	Features
1965–68	Multilayer perceptron	Ivakhnenko <i>et al.</i>	Though perceptron were advanced of McCulloch–Pitts model it had its own limitations.
1960–70	Back Propagation	Became popular by Rumelhart <i>et al.</i> in 1986	A multilayered network of neurons with hidden layer(s) can be used to approximate any continuous function to any desired precision
2006	Unsupervised learning	Hinton and Salakhutdinov	Unsupervised pre-training. Used in training a very deep learner

## 6.3 Biological Neuron

Dendrites are structures used for collecting input for a neuron. It collects and sums up the inputs and if the result is greater than its firing threshold, the neuron fires else it inhibits. Figure 6.1 shows the structure of a biological neuron. When a neuron fires, it sends an electrical impulse from its nucleus to its boutons. The boutons can then network to more neurons via connections called *synapses*. Learning takes place by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The human brain consists of about one hundred billion (100,000,000,000) neurons, each with about 1000 synaptic connections. Our intelligence depends on the effectiveness of these synaptic connections.

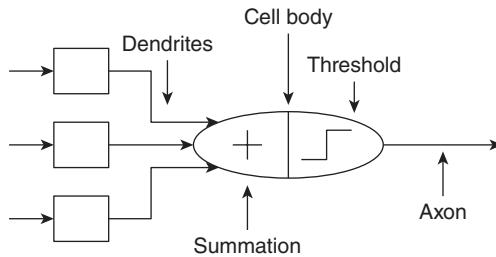


**Figure 6.1** Structure of a biological neuron.

### 6.3.1 From Human Neurons to Artificial Neurons

First we try to deduce the essential features of neurons and their interconnections. Figure 6.2 shows a direct mapping between a human neuron and a biological neuron. We then typically program a computer to simulate these features. However, because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons.

Network computation is performed by a dense mesh of computing nodes and connections. They operate collectively and simultaneously on most or all data and inputs. The basic processing elements of neural networks are called *artificial neurons*, or simply neurons. Often we simply call them nodes. Neurons perform as summing and nonlinear mapping junctions. In some cases, they can be considered as threshold units that fire when their total input exceeds certain levels. Neurons usually operate in parallel and are configured in regular architectures. They are often organized in layers, and feedback connections both within



**Figure 6.2** An artificial neuron mimicking a biological neuron.

the layer and toward adjacent layers are allowed. Each connection strength is expressed by a numerical value called *weight*, which can be modified.

Artificial neural systems function as parallel distributed computing networks. Their most basic characteristic is their architecture. Only some of the networks provide instantaneous responses. Others need time to respond and are characterized by their time-domain behavior, which we often refer to as *dynamics*.

Neural networks also differ from each other in their learning modes. There are a variety of learning rules that establish when and how the connecting weights change. Finally, networks exhibit different speeds and efficiency of learning. As a result, they also differ in their ability to accurately respond to the cues presented at the input.

Vast discrepancies exist between both the architectures and capabilities of artificial and natural neural networks. Knowledge about actual brain functions is so limited, however, and there is little to guide those who would try to emulate them. No models have been successful in duplicating the performance of the human brain. Therefore, the brain has been and still is only a metaphor for a wide variety of neural network configurations that have been developed.

## 6.4 Basics of Artificial Neural Networks

The models of ANN are specified by the three basic entities:

1. Model's synaptic connections.
2. Training/learning rules adopted for adjusting weights.
3. Activation functions.

### 6.4.1 Network Architecture

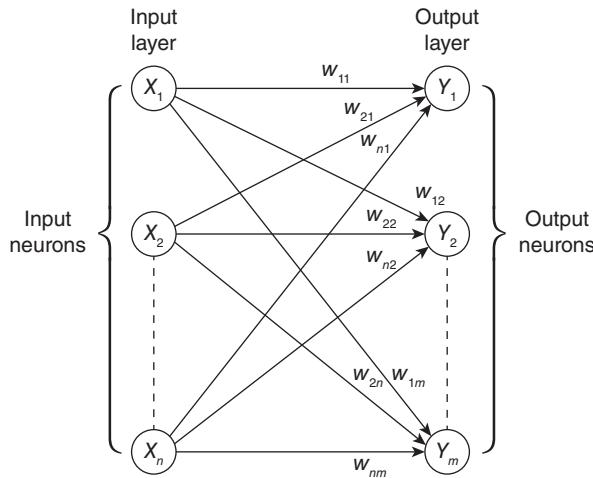
The arrangement of neurons form layers and the connection pattern formed within and between layers is called network architecture. Neural networks can be classified as single-layer or multilayer neural networks. Each layer is formed by a set of processing elements. Layer is a stage that can link between input layer and output layer. How the linking takes place leads to different types of network architectures.

#### 6.4.1.1 Single Layer Feed-forward Network

There can be a type of network in which the input layer is directly connected to output layer without any intermediate layers. Such a network is called single-layer feed forward network (Fig. 6.3).

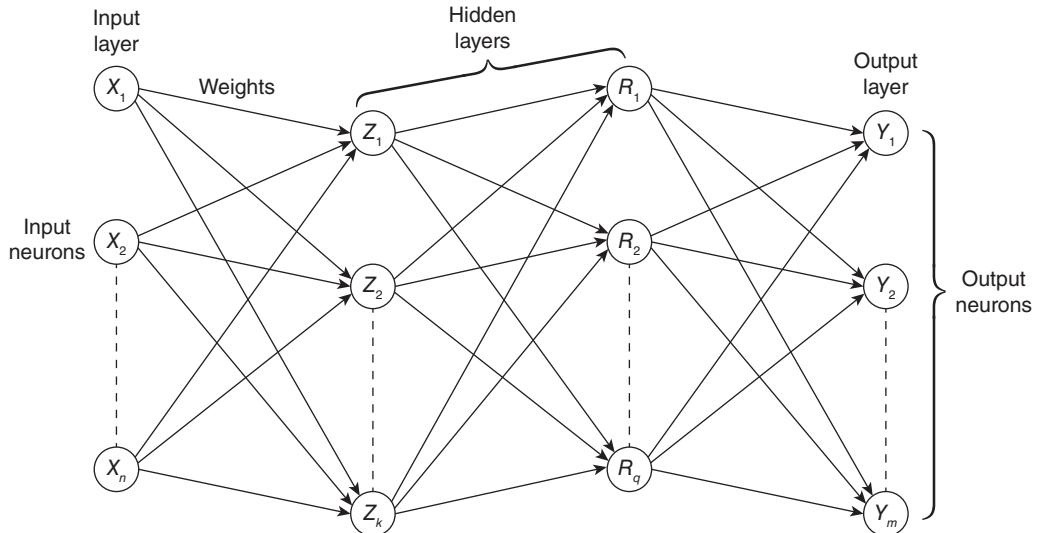
#### 6.4.1.2 Multilayer Feed-forward Network

A multilayer feed-forward network is formed by the interconnection of one or more intermediate layers. The input layer receives the input of the neural network and buffers the input signal. The output layer generates



**Figure 6.3** Single layer feed-forward network.

the output of the network. A layer that is formed between the input and output layers is called *hidden layer*. The hidden layer does not contact with the external environment directly. There can be zero to several hidden layers. The more the number of hidden layers, the more complex is the network. This may improve the efficiency of the network but requires more time to train it. In a fully connected network, every output from one layer is connected to every node in the next layer. This is illustrated in Fig. 6.4. In a feed-forward network, no neuron from the output layer is connected as input to a node in the same layer or any of the preceding layers.



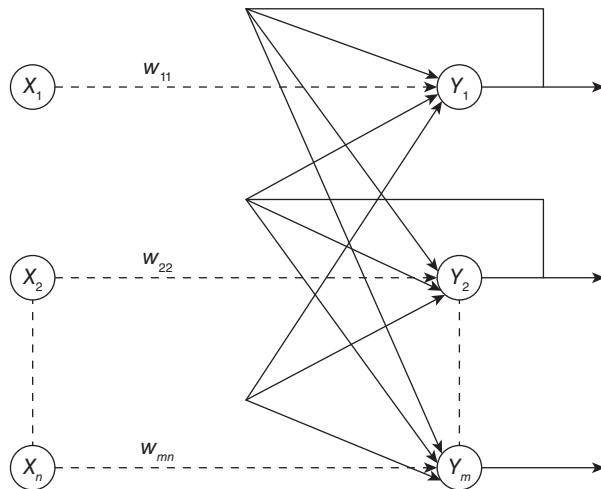
**Figure 6.4** Multilayer feed-forward network.

#### 6.4.1.3 Feedback Network

When outputs are directed back as inputs to same or preceding layer nodes, it results in the network architecture of feedback networks. When the feedback of the output layer is connected back to the same layer, then it is called *lateral feedback*.

#### 6.4.1.4 Recurrent Network

Recurrent network is a feedback network with a closed loop. This type of network can be a single layer network or multilayer network. In a single layer network with a feedback connection, a processing element's output can be directed back to itself or to another processing element or to both as shown in Fig. 6.5.



**Figure 6.5** Recurrent network.

When the feedback is directed back to the hidden layers it forms a *multilayer recurrent network*. In addition, a processing element output can be directed back to itself and to other processing elements in the same layer.

#### 6.4.2 Learning

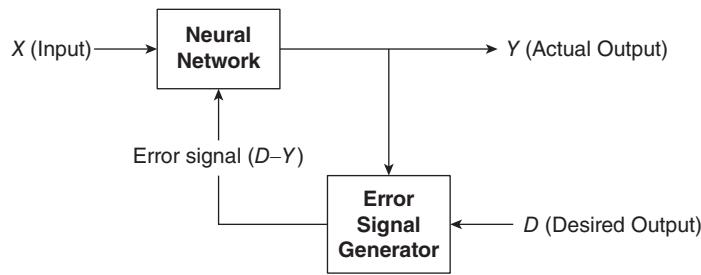
Learning (or training) is a process in which the neural network responds to a stimulus correctly by adopting proper adjustments of values of network parameters producing the desired response for each stimulus. Learning can be classified into three categories: supervised learning, unsupervised learning, and reinforcement learning.

##### 6.4.2.1 Supervised Learning

In supervised learning, the learning is done with the help of a teacher. Consider the example of a child learning to sing. At first, he/she does not know how to sing. He/she tries to sing a song the same way as the singer. The training involves listening to the song again and again till he/she can reproduce it in the same tone and manner. Here, singing takes place by trying to sing in the same manner as the singer.

In supervised learning, each input vector is associated with the output which is desired. The input vector and the corresponding output vector results in a training pair. Here, the network knows what should be the output.

During training, the input vector is given to the network to produce an output. This output is the actual output. Then this actual output is checked whether it is same as the desired output. The block diagram of supervised learning algorithm is shown in Fig. 6.6. The difference between the actual and desired output is considered as the error signal and is generated by the network. This error signal can be used to adjust the weights of the network layers so that for all training pair the actual output becomes the desired output.

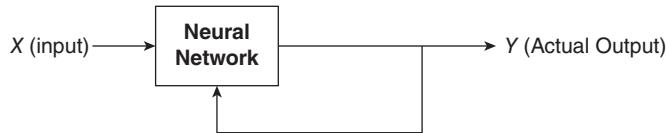


**Figure 6.6** Blocks of supervised learning algorithm.

#### 6.4.2.2 Unsupervised Learning

Just as the name suggests, unsupervised learning is done without the help of a teacher. Consider how a fish learns to swim. It is not taught how to do so but it develops the skills on its own. Thus, it is clear that this type of learning is independent and not taught by a teacher.

In unsupervised learning, the inputs of a similar category are grouped together without the help of any training. The network clubs together the similar input patterns to form clusters in the training process. When a new input is applied, the network gives an output response indicating the class to which it belongs. If an input does not belong to any cluster, a new cluster is formed. This is shown in Fig. 6.7.

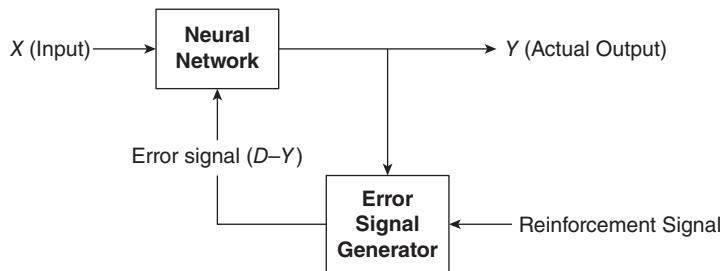


**Figure 6.7** Block diagram of unsupervised learning algorithm.

Here, there is no feedback from the environment to decide if the output is correct. Here the network discovers its own patterns by changing its parameters. This is termed as *self-organization*.

#### 6.4.2.3 Reinforcement Learning

Reinforcement learning is similar to supervised learning in that information is available. However, in case of reinforcement learning, only critical information is available. The exact information needs to be obtained from this critical information. The process of extracting real information from critical information is termed as reinforcement learning (Fig. 6.8).



**Figure 6.8** Reinforcement model.

## 6.5 Activation Functions

The neuron as a processing node performs the operation of summation of its weighted inputs, or the scalar product computation to obtain the net. Subsequently, it performs the nonlinear operation  $f(\text{net})$  through its activation function. Typical activation functions used are *bipolar activation functions* and *unipolar activation functions*.

### 6.5.1 Bipolar Activation Functions

There are two types of bipolar activation functions: bipolar binary and bipolar continuous.

The bipolar binary function is defined as

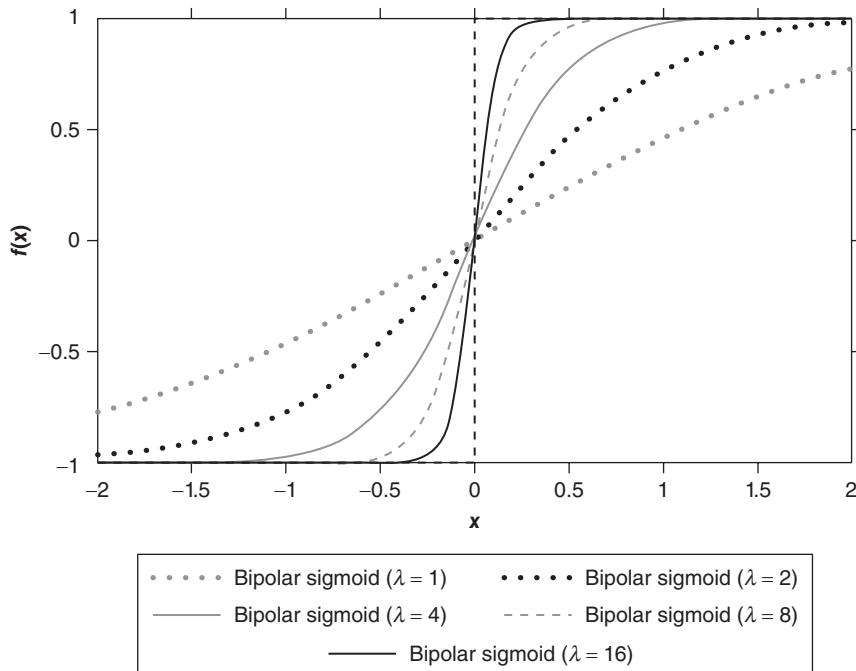
$$f(\text{net}) \stackrel{\Delta}{=} \text{sgn}(\text{net}) = \begin{cases} +1, & \text{net} > 0 \\ -1, & \text{net} < 0 \end{cases} \quad (6.2)$$

The bipolar continuous function is defined as

$$f(\text{net}) \stackrel{\Delta}{=} \frac{2}{1 + \exp(-\lambda \text{net})} - 1 \quad (6.3)$$

where  $\lambda > 0$  is proportional to the neuron gain determining the steepness of the continuous function  $f(\text{net})$  near  $\text{net} = 0$ .

The continuous activation function for various  $\lambda$  is shown in Fig. 6.9.



**Figure 6.9** Activation functions for different values of  $\lambda$ .

Notice that as  $\lambda \rightarrow \infty$ , the continuous function becomes the  $\text{sgn}(\text{net})$  function. The word “bipolar” is used to point out that both positive and negative responses of neurons are produced for this definition of the activation function.

### 6.5.2 Unipolar Activation Functions

By shifting and scaling the bipolar activation functions, unipolar continuous and unipolar binary activation functions can be obtained.

The unipolar continuous activation function is defined as

$$f(\text{net}) \stackrel{\Delta}{=} \frac{1}{1 + \exp(-\lambda \text{net})} \quad (6.4)$$

The unipolar binary activation function is defined as

$$f(\text{net}) \stackrel{\Delta}{=} \begin{cases} 1, & \text{net} > 0 \\ 0, & \text{net} \leq 0 \end{cases} \quad (6.5)$$

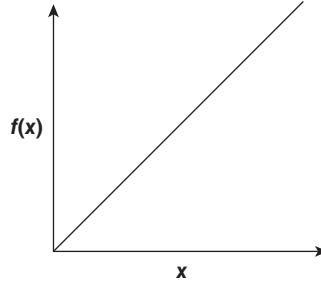
Again, the unipolar binary function is the limit of  $f(\text{net})$  when  $\lambda \rightarrow \infty$ .

### 6.5.3 Identity Function

The identity function is a linear function and can be defined as

$$f(x) = x \quad \text{for all } x$$

Figure 6.10 shows the identity function.



**Figure 6.10** Identify function.

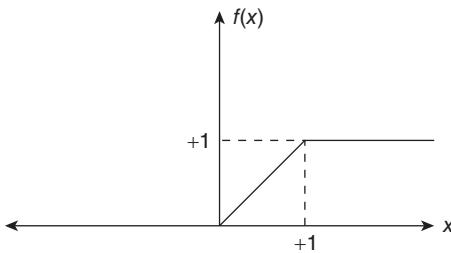
The output here remains the same as input. The input layer uses the identity activation function.

### 6.5.4 Ramp Function

The ramp function is defined as

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases} \quad (6.6)$$

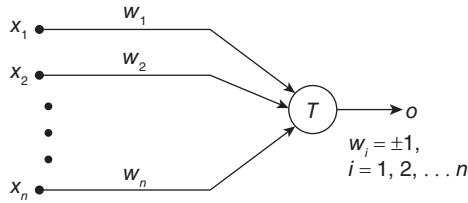
Figure 6.11 shows the ramp function. It is clear from the figure that  $f(x)$  takes the value 0 for values of  $x < 0$ ,  $x$  for values of  $0 < x < 1$ , and 1 for values of  $x > 1$ .



**Figure 6.11** Ramp function.

## 6.6 McCulloch–Pitts Neuron Model

The first definition of a synthetic neuron was based on the simplified biological model formulated by McCulloch and Pitts (1943). The McCulloch–Pitts model of the neuron is shown in Fig. 6.12.



**Figure 6.12** McCulloch–Pitts model.

The inputs  $x_i$ , for  $i = 1, 2, \dots, n$ , are 0 or 1, depending on the absence or presence of the input impulse at instant  $k$ . The neuron's output signal is denoted as "o". The firing rule for this model is given in Eq. (6.7).

$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^n w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^n w_i x_i^k < T \end{cases} \quad (6.7)$$

where

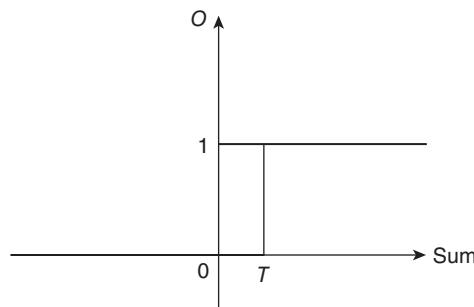
$k = 0, 1, 2, \dots$  denotes the discrete-time instant.

$w_i$  is the multiplicative weight connecting the  $i$ th input with the neuron's membrane.

We will assume that a unity delay elapses between the instants  $k$  and  $k + 1$ . Note that for this model  $w_i = +1$  for excitatory synapses,  $w_i = -1$  for inhibitory synapses, and  $T$  is the neuron's threshold value, which needs to exceed by the weighted sum of signals for the neuron to fire.

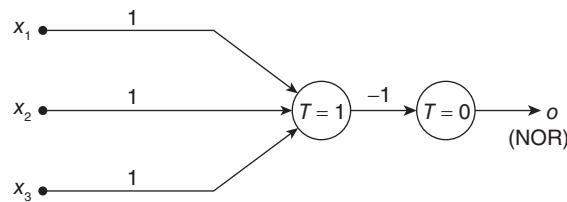
The function  $f$  is a linear step function at threshold  $T$  as shown in Fig. 6.13.

Although this neuron model is simplistic, it has substantial computing potential. It can perform the basic logic operations NOT, OR, and, provided its weight and threshold are appropriately selected. The basic McCulloch–Pitts model for NOR gate is shown in Fig. 6.14.



**Figure 6.13** Linear threshold function.

In McCulloch–Pitts neuron, only analysis can be performed. Hence, we need to assume both weights  $w_1$  and  $w_2$  are excitatory and analyze. If the weights are not suitable, we have to try one weight as excitatory and the other weight as inhibitory and analyze.



**Figure 6.14** Basic McCulloch–Pitts model for NOR gate.

### 6.6.1 Solved Problems

#### Solved Problem 6.1

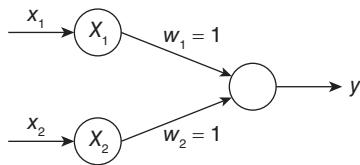
Implement AND function using McCulloch–Pitts neuron.

#### **Solution:**

Consider the truth table for AND function.

$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	0
1	1	1

As already mentioned, only analysis can be performed in the McCulloch–Pitts model. Hence, let us assume the weights  $w_1 = 1$  and  $w_2 = 1$ . The network architecture is shown in Fig. 6.15.



**Figure 6.15** AND implementation using McCulloch–Pitts model.

With these assumed weights, the net input is calculated for four inputs as

$$\begin{aligned} (1,1), \quad y_{in} &= x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times 1 = 2 \\ (1,0), \quad y_{in} &= x_1 w_1 + x_2 w_2 = 1 \times 1 + 0 \times 1 = 1 \\ (0,1), \quad y_{in} &= x_1 w_1 + x_2 w_2 = 0 \times 1 + 1 \times 1 = 1 \\ (0,0), \quad y_{in} &= x_1 w_1 + x_2 w_2 = 0 \times 1 + 0 \times 1 = 0 \end{aligned}$$

Thus, the output of neuron  $Y$  can be written as

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

where 2 represents the threshold value.

### Solved Problem 6.2

Implement ANDNOT function using McCulloch–Pitts neuron (use binary data representation).

#### **Solution:**

In case of ANDNOT function, the response is true if the first input is true and the second input is false. For all other input variations, the response is false. The truth table for ANDNOT function is given as

$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	1
1	1	0

#### **Case 1:**

Assume that both weights  $w_1$  and  $w_2$  are excitatory, i.e.,  $w_1 = 1$  and  $w_2 = 1$ . Then for the four inputs, we calculate the net input using the following equation.

$$y_{in} = x_1 w_1 + x_2 w_2$$

Now, we calculate the net input as follows

$$(1,1), \quad y_{in} = 1 \times 1 + 1 \times 1 = 2$$

$$(1,0), \quad y_{in} = 1 \times 1 + 0 \times 1 = 1$$

$$(0,1), \quad y_{in} = 0 \times 1 + 1 \times 1 = 1$$

$$(0,0), \quad y_{in} = 0 \times 1 + 0 \times 1 = 0$$

From the calculated net inputs, it is not possible to fire the neuron for input (1, 0) only. Hence, these weights are not suitable.

### Case 2:

Assume one weight as excitatory and the other as inhibitory, i.e.,  $w_1 = 1$  and  $w_2 = -1$ . Now, we calculate the net input as follows

$$(1,1), \quad y_{in} = 1 \times 1 + 1 \times -1 = 0$$

$$(1,0), \quad y_{in} = 1 \times 1 + 0 \times -1 = 1$$

$$(0,1), \quad y_{in} = 0 \times 1 + 1 \times -1 = -1$$

$$(0,0), \quad y_{in} = 0 \times 1 + 0 \times -1 = 0$$

From the calculated net inputs, it is now possible to fire the neuron for input (1, 0) by fixing a threshold of 1, i.e.,  $T \geq 1$ . Thus,  $w_1 = 1$  and  $w_2 = -1$ . The output of the neuron can be written as

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases}$$

### Solved Problem 6.3

Implement XOR function using McCulloch–Pitts neuron.

#### **Solution:**

The truth table for XOR function is given as

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

The XOR function cannot be represented by a simple and single logic function. It is represented by the following two equations.

$$y = x_1 \bar{x}_2 + \bar{x}_1 x_2$$

$$y = z_1 + z_2$$

where  $z_1 = x_1 \bar{x}_2$ ,  $z_2 = \bar{x}_1 x_2$ ,  $y = z_1 + z_2$ .

A single layer net is not sufficient to represent the function. An intermediate layer is necessary.

**First function ( $z_1 = x_1 \bar{x}_2$ ):**

The truth table for function  $z_1$  is shown as

$x_1$	$x_2$	$z_1$
0	0	0
0	1	0
1	0	1
1	1	0

**Case 1:**

Assume both weights as excitatory, i.e.,  $w_{12} = w_{22} = 1$ .

We calculate the net inputs

$$(0,0), z_{1in} = 0 \times 1 + 0 \times 1 = 0$$

$$(0,1), z_{1in} = 0 \times 1 + 1 \times 1 = 1$$

$$(1,0), z_{1in} = 1 \times 1 + 0 \times 1 = 1$$

$$(1,1), z_{1in} = 1 \times 1 + 1 \times 1 = 2$$

Hence it is not possible to obtain  $z_1$  using these weights.

**Case 2:**

Assume one weight as excitatory and the other as inhibitory, i.e.,  $w_{12} = 1$  and  $w_{22} = -1$ .

We calculate the net inputs

$$(0,0), z_{1in} = 0 \times 1 + 0 \times -1 = 0$$

$$(0,1), z_{1in} = 0 \times 1 + 1 \times -1 = -1$$

$$(1,0), z_{1in} = 1 \times 1 + 0 \times -1 = 1$$

$$(1,1), z_{1in} = 1 \times 1 + 1 \times -1 = 0$$

**Case 3:**

Assume one weight as inhibitory and the other as excitatory, i.e.,  $w_{12} = -1$  and  $w_{22} = 1$

We calculate the net inputs

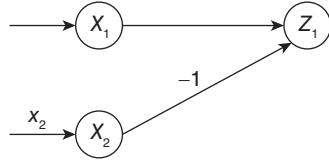
$$(0,0), z_{1in} = 0 \times -1 + 0 \times 1 = 0$$

$$(0,1), z_{1in} = 0 \times -1 + 1 \times 1 = 1$$

$$(1,0), z_{1in} = 1 \times -1 + 0 \times 1 = -1$$

$$(1,1), z_{1in} = 1 \times -1 + 1 \times 1 = 0$$

It is possible to get the desired output based on this calculated net input. Thus  $w_{12} = -1$  and  $w_{22} = -1$  and threshold,  $T \geq 1$  for  $z_2$  neuron. The neuron can be represented as follows:



### Second function ( $z_2 = \bar{x}_1 x_2$ ):

The truth table for function  $z_2$  is given as

$x_1$	$x_2$	$z_1$
0	0	0
0	1	1
1	0	0
1	1	0

### Case 1:

Assume both weights as excitatory, i.e.,  $w_{11} = w_{21} = 1$

We calculate the net inputs

$$(0,0), z_{1in} = 0 \times 1 + 0 \times 1 = 0$$

$$(0,1), z_{1in} = 0 \times 1 + 1 \times 1 = 1$$

$$(1,0), z_{1in} = 1 \times 1 + 0 \times 1 = 1$$

$$(1,1), z_{1in} = 1 \times 1 + 1 \times 1 = 2$$

### Case 2:

Assume one weight as excitatory and the other as inhibitory, i.e.,  $w_{12} = 1$  and  $w_{22} = -1$ .

We calculate the net inputs

$$(0,0), z_{1in} = 0 \times 1 + 0 \times -1 = 0$$

$$(0,1), z_{1in} = 0 \times 1 + 1 \times -1 = -1$$

$$(1,0), z_{1in} = 1 \times 1 + 0 \times -1 = 1$$

$$(1,1), z_{1in} = 1 \times 1 + 1 \times -1 = 0$$

### Case 3:

Assume one weight as excitatory and the other as inhibitory, i.e.,  $w_{12} = -1$  and  $w_{22} = 1$ .

We calculate the net inputs

$$(0,0), z_{1in} = 0 \times -1 + 0 \times 1 = 0$$

$$(0,1), z_{1in} = 0 \times -1 + 1 \times 1 = 1$$

$$(1,0), z_{1in} = 1 \times -1 + 0 \times 1 = -1$$

$$(1,1), z_{1in} = 1 \times -1 + 1 \times 1 = 0$$

It is possible to get the desired output based on this calculated net input. Thus  $w_{12} = -1$ ,  $w_{22} = -1$ , and threshold  $T \geq 1$  for  $z_2$  neuron. The neuron can be represented as follows:

### Third function ( $y = z_1 \text{ OR } z_2$ ):

The truth table is given as

$x_1$	$x_2$	$y$	$z_1$	$z_2$
0	0	0	0	0
0	1	1	0	1
1	0	1	1	0
1	1	0	0	0

Here the net input is calculated as

$$y_{in} = v_1 z_1 + v_2 z_2$$

Assume both weights as excitatory, i.e.,  $v_1 = v_2 = 1$ . We calculate the net inputs

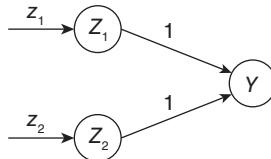
$$(0,0), y_{in} = 0 \times 1 + 0 \times 1 = 0$$

$$(0,1), y_{in} = 0 \times 1 + 1 \times 1 = 1$$

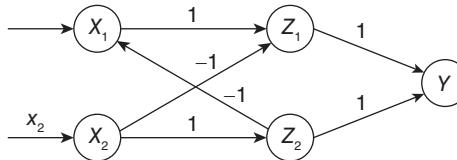
$$(1,0), y_{in} = 1 \times 1 + 0 \times 1 = 1$$

$$(1,1), y_{in} = 0 \times 1 + 0 \times 1 = 0$$

By setting threshold  $\Theta \geq 1$ , the network can be implemented.



The McCulloch–Pitts model for XOR function is given as follows:



## Summary

- An artificial neural network (ANN) is an information processing system inspired by biological nervous systems.
- A trained neural network can be thought of as an “expert” in the category of information it has been given to analyze.

- A trained neural network has other advantages like adaptive learning, self-organization, real-time operation, and fault tolerance.
- A neuron collects inputs using a structure called dendrites. It effectively sums all of the inputs from the dendrites. If the resulting value is greater than its firing threshold, then the neuron fires.
- ANN has various network architectures. These are single layer feed-forward network, multilayer feed-forward network, feedback network, and recurrent network.
- ANN is used both in supervised learning and unsupervised learning.
- ANN uses both discrete and continuous activation functions.
- The McCulloch-Pitt model is a basic model of ANN which replicates the biological neuron.
- By manipulating the weights and thresholds, basic logic operations like AND, OR, and NOT can be implemented using the McCulloch–Pitts model.
- Since EXOR is linearly inseparable two layers of neurons are required to solve this issue.

## Multiple-Choice Questions

1. What is unsupervised learning?
  - (a) The features of a group are not explicitly stated
  - (b) The number of a group may not be known
  - (c) Both (a) and (b)
  - (d) None of the above
2. Signal transmission at synapse is a
  - (a) Physical process
  - (b) Chemical process
  - (c) Both (a) and (b)
  - (d) None of the above
3. The function of a dendrite is to act as a
  - (a) Receptor
  - (b) Transmitter
  - (c) Both (a) and (b)
  - (d) None of the above
4. Learning is a
  - (a) Slow process
  - (b) Fast process
5. The change in weight vector depends on what parameter?
  - (a) Learning
  - (b) Input vector
  - (c) Learning signal
  - (d) All of the above
6. Which of the following are advantages of a neural network over a conventional computer?
  - i. A neural network has the ability to learn by example
  - ii. A neural network is more fault tolerant
  - iii. A neural network is more suited for real-time operation due to its high “computational” rate
  - (a) (i) and (ii)
  - (b) (i) and (iii)
  - (c) All three statements are true
  - (d) None of the statements are true

## Very Short Answer Questions

1. What is a simple artificial neuron?
2. List some commercial practical applications of artificial neural networks.
3. What is a perceptron in machine learning?
4. What are the advantages of neural networks?
5. List different activation neurons or functions.

## Short Answer Questions

---

1. Mention what you can and cannot do with an ANN.
2. What are deterministic models?
3. What are the requirements of learning rules in ANN?
4. What are linearly separable problems of interest for neural network researchers?

## Review Questions

---

1. Explain the working of a biological neuron with the help of a neat diagram.
2. What are the similarities between biological neuron and artificial neuron?
3. What are the different categories of learning algorithms?
4. What activation functions are used in artificial neural network?
5. Why do linearly inseparable problems require two layers? Demonstrate with the example of EXOR problem using the McCulloch–Pitts model.

## Answers

---

### Multiple-Choice Questions

1. (d)
2. (b)
3. (a)
4. (a)
5. (d)
6. (c)

# 7

# Linear Regression

## LEARNING OBJECTIVES

- To understand the basic concept of linear regression and where to apply regression
- To be able to apply linear regression in predicting the value of a consequent, given the antecedent from the equation of the regression line.
- To get acquainted with the different metrics used in the judgment of the accuracy of prediction.

## LEARNING OUTCOMES

- Students will be able to differentiate between discriminative approach and statistical approach.
- Students will be able to compute the linear regression equation.
- Students will be able to predict the outcome based on linear regression equation.
- Students will be able to decide whether the regression line is the best fit based on different metrics.

### 7.1

### Introduction to Supervised Learning and Regression

Supervised learning is the learning in which a supervisor teaches or trains a machine using labeled data with correct answers. After that, the machine is provided with a new set of data so that the supervised learning algorithm can produce the correct outcome from the labeled data with the help of training examples. Regression analysis falls under supervised machine learning. Here, the system tries to predict a value for an input based on previous information. The most important characteristics of regression are:

1. The responses obtained from the model are always quantitative in nature.
2. The model can be constructed only if past data is available.

Simple linear regression is a statistical method that allows us to summarize and study the relationship between two continuous (quantitative) variables.

1. The first variable, denoted by  $x$ , is regarded as the predictor, explanatory, or independent variable.
2. The second variable, denoted by  $y$ , is regarded as the response, outcome, or dependent variable.

Simple linear regression gets its adjective “simple” because it concerns the study of only one predictor variable. On the other hand, multiple linear regression gets its adjective “multiple” because it concerns the study of two or more predictors.

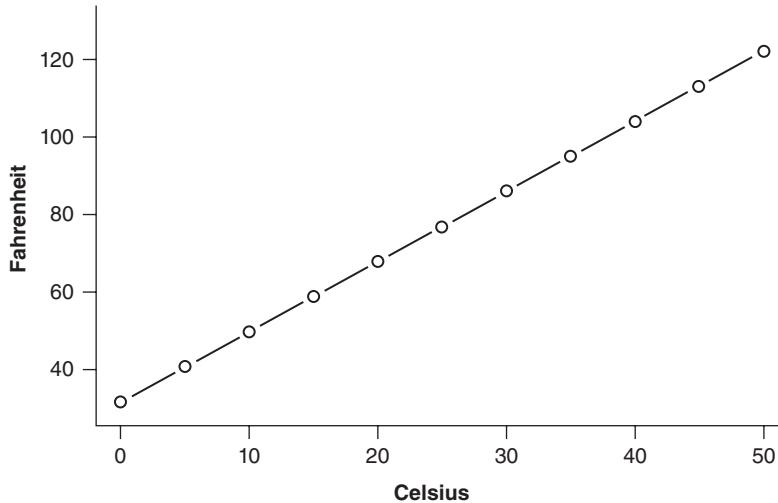
In deterministic relationships, the equation describes the exact relationship between the two variables. Below are some examples of deterministic relationships:

1. Circumference =  $\pi \times$  Diameter
2. Ohm's law:

$$I = \frac{V}{r}$$

where  $V$  is the voltage applied,  $r$  is the resistance, and  $I$  is the current.

Another example of deterministic relationships is the conversion between temperature scales. Figure 7.1 shows the graph between Fahrenheit and Celsius scales as a linear scale, which is deterministic in nature.

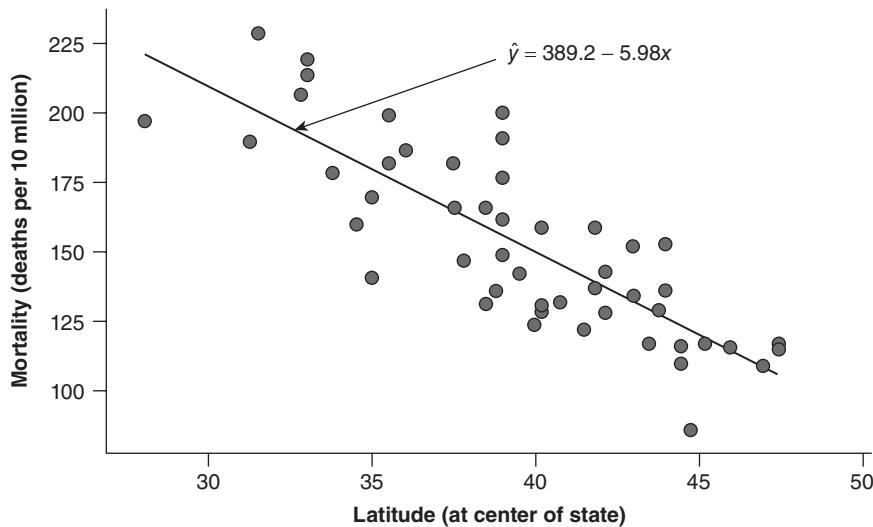


**Figure 7.1** Graph showing the deterministic relationship between Fahrenheit and Celsius scales.

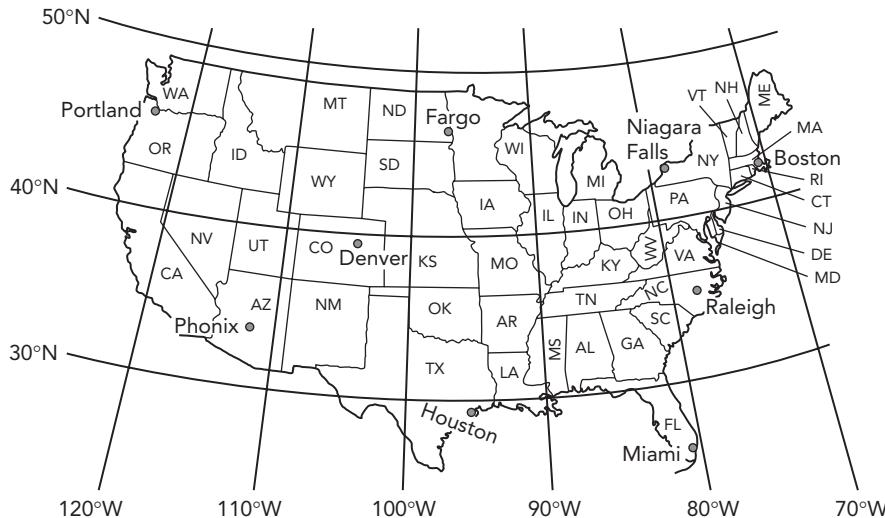
In contrast to deterministic relationships, the relationship between variables is not perfect in statistical relationships. An example of statistical relationship is as follows:

The response variable  $y$  is the mortality due to skin cancer (number of deaths per 1,00,00,000 people), and the predictor variable  $x$  is the latitude (degrees North) at the center of each of the 49 states in the United States (Fig. 7.2). Figure 7.3 shows the latitude-longitude map of the United States from which  $x$  is determined.

It may be anticipated that the higher the latitude of the place, the less exposed one is to the harmful rays of the sun, hence lesser is the risk of death due to skin cancer. The scatter plot shown in Fig. 7.2 supports such a hypothesis. There appears to be a negative linear relationship between latitude and mortality due to skin cancer; however, the relationship is not perfect. Indeed, the plot exhibits some “trend”, but it also exhibits some “scatter”. Therefore, it is a statistical relationship and not a deterministic one.



**Figure 7.2** The plot of mortality due to skin cancer versus latitude.



**Figure 7.3** Different states in the United States.

## 7.2

## Statistical Relation between Two Variables and Scatter Plots

In statistical relationships, the plot is called a *scatter diagram* or *scatter plot* because of the scattering of points in the relation. In statistical terminology, each point in the scatter diagram represents a trial or a case. Statistical relations can be highly useful, even though they do not have the exactitude of a functional relation.

### 7.2.1 Purpose of Regression

The purpose of regression models is to identify a functional relationship between the target variable and a subset of the remaining attributes contained in the dataset. This enables us to

1. Highlight and interpret the dependency of the target variable on other variables.
2. Predict the future value of the target variable based on the functional relationship identified and future value of explanatory variables.

### 7.2.2 Linear Regression

A linear regression line has an equation of the form

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

where  $\underline{x}$  is the explanatory variable,  $\underline{y}$  is the dependent variable,  $b$  is the slope of the line, and  $a$  is the intercept (the value of  $\underline{y}$  when  $\underline{x} = 0$ ).

### 7.2.3 Requirement of Linear Regression

Most types of nonlinear relationships may be reduced to linear cases by means of appropriate preliminary transformations to the original observations.

Let us consider the following examples.

1. Consider the following quadratic relationship:

$$Y = b + wX + dX^2 \quad (7.1)$$

This equation can be linearized through the transformation  $z = x^2$ , which converts it into a linear relationship with two predictors:

$$Y = b + wX + dZ \quad (7.2)$$

2. Consider the following exponential relationship:

$$Y = e^{b+wX} \quad (7.3)$$

This equation can be linearized through a logarithmic transformation  $Z = \log Y$ , which converts it into the linear relationship:

$$Z = b + wX \quad (7.4)$$

These examples indicate that linear models are actually much more general than may appear at first sight.

## 7.3 Steps to Establish a Linear Regression

A simple example of linear regression is to predict the weight of a person when only her/his height is known. To do this we need to know the relationship between height and weight. The steps to create the relationship are as follows:

1. Carry out an experiment of gathering a sample of observed values of height and corresponding weight.
2. Create a relationship model.

3. Find the coefficients from the model created and establish the mathematical equation using these.
4. Get a summary of the relationship model to know the average error in prediction. This average error is also called *residual*.
5. Predict the weight of other people.

On a small dataset given, let us create the linear regression model for predicting weight of the person given the height.

1. **Input data:** Sort the data gathered into tabular form. Table 7.1 shows the observations of heights and corresponding weights of different people.

**Table 7.1** Data sample showing the height and weight of different people

<i>Height</i>	<i>Weight</i>
151	63
174	81
138	56
186	91
128	47
136	57
179	76
163	72
152	62
131	48

2. **Create the relationship model:** The basic regression model is with only one predictor variable and the regression function is linear. The model can be stated as

$$Y_i = \beta_0 + \beta_1 X_i \quad (7.5)$$

where  $Y_i$  is a known constant; it is the value of the response variable in the  $i^{\text{th}}$  trial.  $X_i$  is a known constant; it is the value of the predictor variable in the  $i^{\text{th}}$  trial.  $\beta_0$  and  $\beta_1$  are parameters.

To find “good” estimators of the regression parameters  $\beta_0$  and  $\beta_1$ , we employ the *method of least squares*. In the observations  $(X_i, Y_i)$  for each case, the method of least squares considers the deviation of  $Y_i$  from its expected value as given in Eq. (7.6).

$$Y_i - (\beta_0 + \beta_1 X_i) \quad (7.6)$$

The method of least squares requires that we consider the sum of the  $n$  squared deviations. This criterion is denoted by  $Q$ :

$$Q = \sum_{i=1}^n (Y_i - \beta_0 - \beta_1 X_i)^2 \quad (7.7)$$

According to the method of least squares, the estimators of  $\beta_0$  and  $\beta_1$  are those values  $b_0$  and  $b_1$ , respectively, that minimize the criterion  $Q$  for the given sample observations  $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ .

The objective of this method is to find estimates  $b_0$  and  $b_1$  for  $\beta_0$  and  $\beta_1$ , respectively, for which  $Q$  is a minimum. These estimates will provide a “good” fit of the linear regression function.

- 3. Least squares estimators:** The estimates  $b_0$  and  $b_1$  that satisfy the least squares criterion can be found in two basic ways:

- (a) *Numerical search procedures* can be used to evaluate, in a systematic fashion, the least squares criterion  $Q$  for different estimates  $b_0$  and  $b_1$  until the ones that minimize  $Q$  are found.
- (b) *Analytical procedures* are often used to find the values of  $b_0$  and  $b_1$  that minimize  $Q$ . The analytical approach is feasible when the regression model is not mathematically complex.

Using analytical approach for a regression model, the values  $b_0$  and  $b_1$  that minimize  $Q$  for any particular set of sample data are given by the following simultaneous equations:

$$\begin{aligned}\sum Y_i &= nb_0 + b_1 \sum X_i \\ \sum X_i Y_i &= b_0 \sum X_i + b_1 \sum X_i^2\end{aligned}\tag{7.8}$$

These simultaneous equations are called *normal equations*;  $b_0$  and  $b_1$  are called *point estimators* of  $\beta_0$  and  $\beta_1$ , respectively.

- 4. Derivation:** The normal equations can be derived by calculus. For a given sample of observations  $(X_i, Y_i)$ , the quantity  $Q$  is a function of  $\beta_0$  and  $\beta_1$ . This cost function  $Q$  is given by Eq. (7.7). The values of  $\beta_0$  and  $\beta_1$  that minimize  $Q$  can be derived by differentiating Eq. (7.7) with respect to  $\beta_0$  and  $\beta_1$ . We then set these partial derivatives equal to zero, using  $b_0$  and  $b_1$  to denote the particular values of  $\beta_0$  and  $\beta_1$  that minimize  $Q$ :

$$\begin{aligned}\frac{\partial Q}{\partial \beta_0} &= -2 \sum (Y_i - \beta_0 - \beta_1 X_i) \\ \frac{\partial Q}{\partial \beta_1} &= -2 \sum X_i (Y_i - \beta_0 - \beta_1 X_i)\end{aligned}\tag{7.9}$$

Equating the partial derivatives of Eq. (7.9) to 0, we get

$$\begin{aligned}-2 \sum (Y_i - b_0 - b_1 X_i) &= 0 \\ -2 \sum X_i (Y_i - b_0 - b_1 X_i) &= 0\end{aligned}\tag{7.10}$$

Simplifying these equations, we obtain

$$\begin{aligned}\sum_{i=1}^n (Y_i - b_0 - b_1 X_i) &= 0 \\ \sum_{i=1}^n X_i (Y_i - b_0 - b_1 X_i) &= 0\end{aligned}\tag{7.11}$$

Simplifying and expanding Eq. (7.11), we get Eq. (7.12).

$$\begin{aligned}\sum Y_i - nb_0 - b_1 \sum X_i &= 0 \\ \sum X_i Y_i - b_0 \sum X_i - b_1 \sum X_i^2 &= 0\end{aligned}\tag{7.12}$$

Rearranging the terms of Eq. (7.12) and solving simultaneously for  $b_0$  and  $b_1$ , we get

$$\begin{aligned} b_1 &= \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2} = 0 \\ b_0 &= \frac{1}{n} \left( \sum Y_i - b_1 \sum X_i \right) = \bar{Y} - b_1 \bar{X} \end{aligned} \quad (7.13)$$

where  $\bar{X}$  and  $\bar{Y}$  are the means of  $X_i$  and  $Y_i$  observations, respectively.

- 5. Prediction:** Having found the  $b_0$  and  $b_1$ , we can find the value of  $y$  for any given  $x$ . Here given the height of the person, the weight of the person will be predicted.

### Solved Problem 7.1

Create the relationship model for the data given in Table 7.2 to find the relation between height and weight of students.

**Table 7.2** Dataset of height and weight observations

S. No.	Height (X) cm	Weight (Y) kg	$(X_i - \bar{X})$	$(Y_i - \bar{Y})$	$(X_i - \bar{X})(Y_i - \bar{Y})$	$(X_i - \bar{X})^2$
1	151	63	-2.8	-2.3	6.44	7.84
2	174	81	20.2	15.7	317.14	408.04
3	138	56	-15.8	-9.3	146.94	249.64
4	186	91	32.2	25.7	827.54	1036.8
5	128	47	-25.8	-18.3	472.14	665.64
6	136	57	-17.8	-8.3	147.74	316.84
7	179	76	25.2	10.7	269.64	635.04
8	163	72	9.2	6.7	61.64	84.64
9	152	62	-1.8	-3.3	5.94	3.24
10	131	48	-22.8	-17.3	394.44	519.84
	$\bar{X} =$	$\bar{Y} =$			$\Sigma = 2649.6$	$\Sigma = 3927.6$
	153.8	65.3				

### Solution:

Now

$$b_1 = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2} = \frac{2649.6}{3927.6} = 0.6746$$

$$b_0 = \frac{1}{n} \left( \sum Y_i - b_1 \sum X_i \right) = \bar{Y} - b_1 \bar{X} = 65.3 - (0.6746 \times 153.8) = -38.4551$$

The values calculated in solved problem 1 are then used to predict the weight of student whose height is 170 cm.

When  $x = 170$

$$y = b_1 \times + b_0 = 0.6746 \times (-38.4551) = 76.23$$

## 7.4 Evaluation of Model Estimators

Once the model is established, you need to confirm whether the model is good enough to make predictions in future or that the relationship you built between dependent and independent variables is good enough. For this purpose, various metrics can be used.

### 7.4.1 Karl Pearson's Coefficient of Correlation

Karl Pearson's coefficient of correlation is a helpful statistical formula that quantifies the strength between two variables. This coefficient value helps in determining how strong that relationship is between the two variables. The Pearson coefficient is given by

$$r = \frac{N \sum xy - \sum x \sum y}{\sqrt{[N \sum x^2 - (\sum x)^2][N \sum y^2 - (\sum y)^2]}} \quad (7.14)$$

where  $x$  and  $y$  are variables and  $N$  is the number of instances we have to compute the coefficient.

It has a value between +1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. A value of 1 implies that a linear equation describes the relationship between  $X$  and  $Y$  perfectly, with all data points lying on a line for which  $Y$  increases as  $X$  increases. A value of -1 implies that all data points lie on a line for which  $Y$  decreases as  $X$  increases. A value of 0 implies that there is no linear correlation between the variables.

### Solved Problem 7.2

Compute the correlation coefficient for the data given in Table 7.3 to find the relation between height and weight of 10 students.

**Table 7.3** Dataset of height and weight observations of 10 data samples

S. No.	Height ( $x$ )	Weight ( $y$ )	$xy$	$x^2$	$y^2$
1	151	63	9513	22801	3969
2	174	81	14094	30276	6561
3	138	56	7728	19044	3136
4	186	91	16926	34596	8281
5	128	47	6016	16384	2209
6	136	57	7752	18496	3249
7	179	76	13604	32041	5776

(Continued)

**Table 7.3** (Continued)

S. No.	Height (x)	Weight (y)	$xy$	$x^2$	$y^2$
8	163	72	11736	26569	5184
9	152	62	9424	23104	3844
10	131	48	6288	17161	2304
			$\Sigma$	$\Sigma$	$\Sigma$
	$\bar{x} = 153.8$	$\bar{y} = 65.3$	103081	240472	44513

**Solution:**

We will evaluate the Karl Pearson's coefficient to find out if there is a strong relationship between the height and weight of a person. Thus, we can consider the linear regression equation to evaluate weight of the person from the height.

$$r = \frac{N \sum xy - \sum x \sum y}{\sqrt{\left[ N \sum x^2 - (\sum x)^2 \right] \left[ N \sum y^2 - (\sum y)^2 \right]}}$$

$$r = \frac{(10 \times 103081) - (1538 \times 653)}{\sqrt{[(10 \times 240472) - 1538^2][(10 \times 44513) - 653^2]}}$$

$$r = 0.9771$$

The main advantage of this coefficient is that it summarizes in one value the degree and direction of correlation. The limitations of the Pearson's coefficient are listed below:

1. It always assumes linear relationship.
2. Interpreting the value of  $r$  is difficult.
3. Value of the correlation coefficient is affected by extreme values.
4. It is time consuming.

**7.4.2 R-Square**

R-square gives information about the goodness-of-fit measure for linear regression models. It indicates percentage of variance in the dependent-independent variable pair. It measures the strength of the relationship in a 0 to 100% scale. For each observation in our data, we can compute

$$\bar{y}_i = b_0 + b_1 x_i$$

These are called *fitted values*. Thus, the  $i^{\text{th}}$  fitted value,  $\bar{y}_i$ , is the point on the least squares regression line corresponding to  $x_i$ . For the  $i^{\text{th}}$  observation, we can compute ordinary least squares residuals as shown in Eq. (7.15).

$$e_i = y_i - \bar{y}_i \quad (7.15)$$

One of the properties of the residuals is that their sum is zero.

The following quantities are also computed.

$$\begin{aligned} SST &= \sum (y_i - \bar{y})^2 \\ SSR &= \sum (\hat{y}_i - \bar{y})^2 \\ SSE &= \sum (y_i - \hat{y}_i)^2 \end{aligned} \quad (7.16)$$

where SST is the total sum of squared deviations in  $y$  from its mean. SSR is the sum of squares due to regression. SSE is the sum of squared residuals (errors).

The ratio  $R^2 = \text{SSR}/\text{SST}$  can be interpreted as the proportion of the total variation in  $y$  that is accounted for by the predictor variable  $x$ . The high value of  $R^2$  indicates a strong linear relationship.

Let us consider the following example of dataset of observations (Table 6.4) to compute  $R^2$ .

**Table 7.4** The observations of height and weight to compute  $R^2$

Height (cm)	Weight (kg)	$\bar{y}$	SSR	SST	SSE
151	63	63.411	3.56828322	5.29	0.16892922
174	81	78.927	185.696219	246.49	4.29716316
138	56	54.641	113.612576	86.49	1.84666357
186	91	87.022	471.860924	660.49	15.82162
128	47	47.895	302.934721	334.89	0.8009892
136	57	53.292	144.195426	68.89	13.7503023
179	76	82.3	289.00306	114.49	39.691134
163	72	71.506	38.5185321	44.89	0.24371007
152	62	64.086	1.47471878	10.89	4.34981078
131	48	49.919	236.581006	299.29	3.68183182
			1787.44547	1872.1	84.6521541

$$R^2 = \text{SSR}/\text{SST} = 1784.45/1872.1 = 0.9548$$

#### 7.4.2.1 Multiple R

Multiple R is a correlation coefficient. It gives us an idea of the strength of a linear relationship. For example, a value of 1 means a perfect positive relationship and 0 means no relationship at all. It is the square root of R squared.

$$\text{Multiple R} = \sqrt{R^2}$$

In the above example,

$$\text{Multiple R} = \sqrt{R^2} = 0.9771$$

#### 7.4.3 Standard Error of Estimate

The standard error of the estimate is a measure of the accuracy of predictions. It is given by

$$\sigma_{\text{est}} = \sqrt{\frac{\sum (Y - Y')^2}{N}} \quad (7.17)$$

The denominator is the sample size reduced by the number of model parameters estimated from the same data,  $(n - p)$  for  $p$  regressors or  $(n - p - 1)$  if an intercept is used. In this case,  $p = 1$  so the denominator  $n - 2$ .

In the above example,

$$\sigma_{\text{est}} = \sqrt{\frac{84.65}{8}} = 3.2529$$

## 7.5

## Solved Problems on Linear Regression

### Solved Problem 7.3

The rent of a property is related to its area. Given the area in square feet and rent in dollars (Table 7.5), find the relationship between area and rent using the concept of linear regression. Also predict the rent for a property of  $790 \text{ ft}^2$ .

**Table 7.5** Area of a property and its corresponding rental value

S. No.	Area ( $\text{ft}^2$ )	Rent (₹)
1	340	500
2	1080	1700
3	640	1100
4	880	800
5	990	1400
6	510	500

**Solution:**

X	Y	$(X - X')$	$(Y - Y')$	$(X - X')(Y - Y')$	$(X - X')^2$
340	500	-400	-500	200000	160000
1080	1700	340	700	238000	115600
640	1100	-100	100	-10000	10000
880	800	140	-200	-28000	19600
990	1400	250	400	100000	62500
510	500	-230	-500	115000	52900
$X' = 740$	$Y' = 1000$			615000	420600

Hence

$$b_1 = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2} = 1.462197$$

$$b_0 = \frac{1}{n} \left( \sum Y_i - b_1 \sum X_i \right) = \bar{Y} - b_1 \bar{X} = -82.0257$$

Therefore

$$Y(790) = Y_i = \beta_0 + \beta_1 X_i = 1.46(790) - 82 = 1071$$

### Solved Problem 7.4

The marks obtained by a student are dependent on her/his study time. Given the study time in minutes and marks out of 2000, find the relationship between study time and marks using the concept of linear regression. Also predict the marks for a student if she/he studied for 790 minutes.

<i>S. No.</i>	<i>Study Time (min)</i>	<i>Marks Obtained</i>
1	350	520
2	1070	1600
3	630	1000
4	890	850
5	940	1350
6	500	490

**Solution:**

<i>X</i>	<i>Y</i>	<i>X – X'</i>	<i>Y – Y'</i>	$(X - X')(Y - Y')$	$(X - X')(X - X')$
350	520	-380	-448	170354	144400
1070	1600	340	631.7	214778	115600
630	1000	-100	31.7	-3170	10000
890	850	160	-118	-18928	25600
940	1350	210	381.7	80157	44100
$X' = 730$ $Y' = 968.33$				553200	392600

Hence

$$\beta_1 = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2} = 1.409068$$

$$\beta_0 = \frac{1}{n} \left( \sum Y_i - \beta_1 \sum X_i \right) = \bar{Y} - \beta_1 \bar{X} = -60.2861$$

Therefore

$$Y(790) = Y_i = \beta_0 + \beta_1 X_i = 1.41(790) - 60 = 1054$$

This means that if a student studies for 790 minutes, he can score 1054/2000

## Summary

---

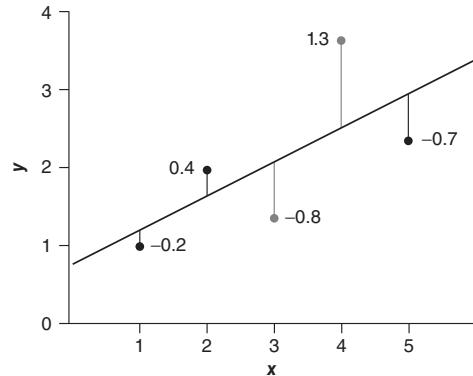
- Supervised learning is similar to learning from a teacher. The input and corresponding outputs are given to the model; the data-centric model learns by generating a hypothesis. This is tested across samples with known outputs. This helps in computing the accuracy of the model. The model is judged based on the accuracy of the model.
- Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables.
- Statistical relations can be highly useful, even though they do not have the exactitude of a functional relation.
- The purpose of regression models is to identify a functional relationship between the target variable and a subset of the remaining attributes contained in the dataset.
- The steps to establish a linear regression are
  - (a) carrying out the experiment of gathering a sample of observed values of height and corresponding weight,
  - (b) creating a relationship model,
  - (c) finding the coefficients from the model created and establishing the mathematical equation using these,
  - (d) getting a summary of the relationship model to know the average error in prediction,
  - (e) predicting the value of an unknown sample.
- Least square estimator helps in computing the minimum of the squares of errors between actual and predicted values.
- Accuracy of the model is predicted by Karl Pearson's coefficient, whose values ranges between -1 and +1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation.
- R-square variable ( $R^2$ ) also helps determine the correlation between input and output variables. High value of  $R^2$  indicates a strong linear relationship.
- Standard error of the estimate is a measure of the accuracy of predictions.

## Multiple-Choice Questions

---

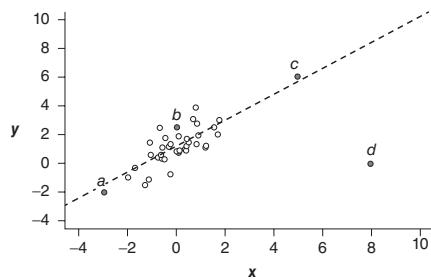
- Which of the following plot is best suited to test linear relationship of  $y$ (dependent) and  $x$ (independent) continuous variables?
  - (a) Scatter plot
  - (b) Barchart
  - (c) Histograms
  - (d) None of the above
- A correlation between age and health of a person found to be -1.09. On the basis of this you would tell the doctors that:
  - (a) Age is good predictor of health and is positively correlated.
  - (b) Age is poor predictor of health and is negatively correlated.
  - (c) Age is good predictor of health and is negatively correlated.
  - (d) Age is poor predictor of health and is positively correlated.
- How many coefficients do you need to estimate a simple linear regression model (one independent variable)?
  - (a) 1
  - (b) 2
  - (c) 3
  - (d) 4

4. The graph below represents a regression line predicting  $Y$  from  $X$ . The values on the graph shows the residuals for each prediction value. Use this information to compute the SSE.



- (a) 3.02                    (b) 0.75  
 (c) 1.01                    (d) None of the above

5. Consider the following dataplot



Which bold point if removed will have the largest effect on the fitted regression line (dashed line) shown in the figure?

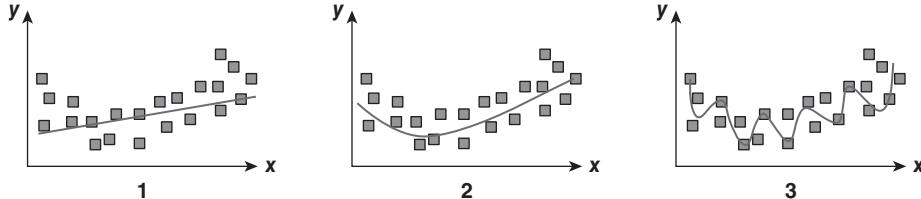
- (a) a  
 (b) b  
 (c) c  
 (d) d

### Very Short Answer Questions

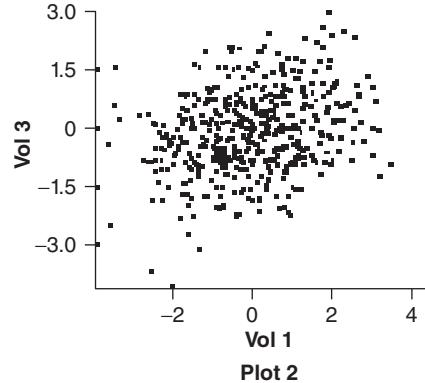
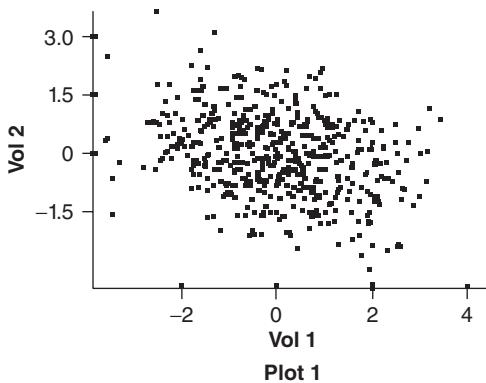
- What is the trade-off between an underfitting and an overfitting model in regression modeling?
- If the correlation coefficient is 0.9, what can you comment on the relationship between  $X$  and  $Y$ ?
- Why is a scatter plot best suited to show the relationship between  $X$  and  $Y$ ?
- If the correlation coefficient between age and health of the person is  $-1.09$ , what is your inference? Is the inference justified?

### Short Answer Questions

- In the figure below, 1, 2, and 3 are three plots for the same data points. Which one of these is best fitted and which is overfitted? In which case is the training error higher.



2. The figure below shows two plots of height and weight. What do you feel should be the correct scatter plot if  $y$ -axis represents height and  $x$ -axis represents weight?



## Review Questions

- What is supervised learning?
- How is regression different from classification?
- What are the metrics used in judging the accuracy of prediction?
- The sales of a company (in million dollars) for each year are shown in the table below.

$x$ (year)	2005	2006	2007	2008	2009
$y$ (sales)	12	19	29	37	45

- (a) Find the least square regression line,  $y = ax + b$ .  
 (b) Use the least squares regression line as a model to estimate the sales of the company in 2012.

- The city's transportation department is interested in studying the relationship between the temperature and the number of passengers that ride the main bus line in order to better serve their customers. The manager recorded the temperature at the beginning of the hour, and then had a bus driver record the number of passengers that boarded the bus throughout the

hour. Their findings are listed in the following table.

Temperature ( $^{\circ}\text{C}$ )	Number of Passengers
42	173
37	149
46	185
30	123
50	201
43	174
43	175
46	188
46	186
49	198

Use the above information to obtain a least squares regression equation. Predict the number of passengers when the temperature is at  $45^{\circ}\text{C}$ . For the pair of data (temperature =  $42^{\circ}\text{C}$ , passengers = 173), i.e., observation one, what is the residual?

## Answers

### Multiple-Choice Questions

1. (a)    2. (c)    3. (b)    4. (a)    5. (d)



# 8

# Logistic Regression

## LEARNING OBJECTIVES

- To understand the requirement of logistic regression.
- To understand between logistic regression and linear regression.
- To explain the concepts of odds and odds ratio.
- To understand the computation of parameters using maximum likelihood estimation.

## LEARNING OUTCOMES

- Students will be able distinguish cases where logistic regression needs to be used.
- Students will be able to use maximum likelihood estimation for computation of parameters for logistic regression.
- Students will be able to prediction the probability of the output class using the concept of logistic regression.

### 8.1 Introduction to Logistic Regression

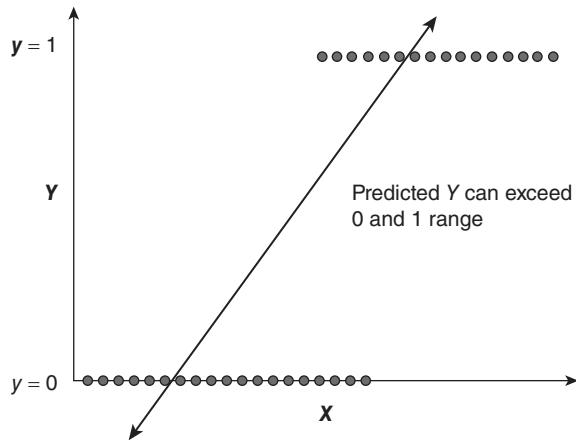
In Chapter 7 of linear regression, the problems dealt with were of the form that computes the exact output value given the inputs. In our discussion of linear regression, the response variable  $Y$  has been regarded as a continuous quantitative variable. However, sometimes we may need to predict a discrete variable; for example, a model that can predict whether a person is male or female based on their height. This type of prediction where we are bothered about the output label and not the exact value is called *classification problem*. Logistic regression is a binary classification algorithm used when the response variable is dichotomous (1 or 0). The output  $y_i$  is thus a realization of a random variable  $Y_i$  that can take the values 1 and 0 with probabilities  $p_i$  and  $1 - p_i$ , respectively.

### 8.2 Scenarios Which Require Logistic Regression

Let  $p$  denote the probability that  $Y = 1$  when  $X = x$ . If we use the standard linear model to describe  $p$ , as given in Eq. (8.1), then the model for probability will be:

$$p = P_r(Y = 1 \mid X = x) = \beta_0 + \beta_1 x \quad (8.1)$$

Since  $p$  is a probability, it must lie between 0 and 1. The linear function given in Eq. (8.1) is unbounded and hence cannot be used to model probability. In Fig. 8.1, the output is either 1 or 0. Thus a regression line will not be able to build such a classifier model which gives categorical output. This shows that linear regression



**Figure 8.1** Graph showing why linear regression is not suitable for a binary output.

is not a suitable for those class of classification problems where we need to know if the probability output belongs to a particular category.

### 8.2.1 Examples of Binary Classification Model

In the examples given below, we need to decide on the output based on the input dataset. The exact value of the output is not required. Only the category of the output is required. The dataset of this class of classification problems will be of the type  $\langle x_1, x_2, \dots, x_n \rangle, y_0$  or  $y_1$ , where  $x_1 \dots x_n$  are the inputs of  $n$  attributes and  $y_0$  or  $y_1$  are the corresponding outputs.

1. **Spam detection:** Predicting if an email is spam or not, given inputs parameters such as matching character strings and length of uninterrupted strings. The corresponding input belongs to nominal class attribute {0, 1}.
2. **Credit card fraud detection:** Predicting if a given credit card transaction is fraud or not. A study was carried out using 23 input variables and a default payment of credit card. Some of the significant inputs were amount of given credit, gender, education, marital status, age, history of past payments, amount of bill payments, and amount of previous payments.
3. **Cancer detection:** Predicting if a given mass of tissue is benign or malignant. For this, the dataset pertaining to cell nucleus were taken. Some prominent features considered for inputs were radius, texture, perimeter, area, smoothness, compactness, and concavity. The output diagnosis was either M (malignant) or B (benign).
4. **Marketing:** Predicting if a given user will deposit in a term deposit, based on some important inputs including age, job, marital status, education, having previous loans, and number of employees.
5. **Banking:** Predicting if a customer will default on a loan based on some banking particulars such as nature of job, age, and previous loan details.

In logistic regression, we get a probability score that reflects the probability of occurrence of the event, in contrast to linear regression which gives the actual predicted output.

### 8.3 Odds

Let us first try and understand the concept of odds and odds ratio. Most people regard probability as a “natural” way to quantify the chances that an event will occur. We automatically think in terms of numbers ranging from 0 to 1, with 0 implying that the event will certainly not occur and 1 implying that it will occur. But there are other ways to represent the chances of the event occurring, one of which – the odds – has nearly an equal claim of being “natural”.

The odds of an event occurring is the ratio of the expected number of times that the event will occur to the expected number of times it will not occur. There is a simple relationship between probabilities and odds. If  $p$  is the probability of an event and  $O$  is the odds of the event, then:

$$O = \frac{p}{1-p} = \frac{\text{probability of event}}{\text{probability of no event}}$$

#### 8.3.1 Odd and Odds Ratio

Unlike probability, there is no upper bound on the odds. How  $X$  affects  $Y$  would be difficult to express using probability. The effect of the input on the output would be difficult to ascertain using the concept of probability. A single number on the probability scale to convey the relationship between the predictor and the probability of a response is not possible. However, using odds and odds ratio we can convey the relationship between the input and output.

### 8.4 Building Logistic Regression Model (Logit Function)

Transforming probability to odds removes the upper bound. If we then take the logarithm of the odds, we also remove the lower bound. Thus, we get the logistic model frame using Eq. (8.2).

$$\log\left[\frac{p_i}{1-p_i}\right] = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} \quad (8.2)$$

The expression  $\log\left[\frac{p_i}{1-p_i}\right]$  is called *logit function*.

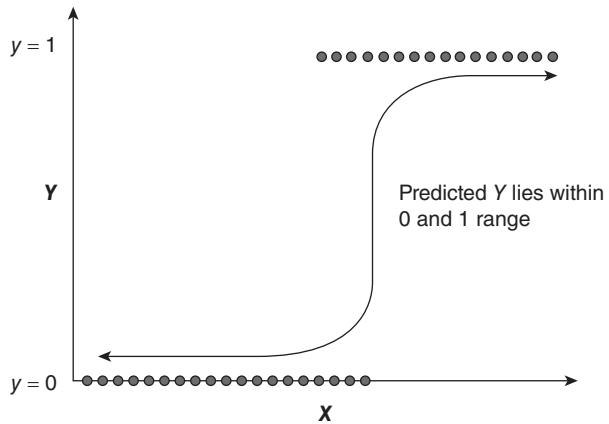
We can solve the logit equation for  $p_i$  to obtain expression given in Eq. (8.3).

$$p_i = \frac{\exp(\alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik})}{1 + \exp(\alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik})} \quad (8.3)$$

Equation (8.3) can be simplified by dividing both numerator and denominator by the numerator itself, to obtain the expression for  $p_i$  given in Eq. (8.4).

$$p_i = \frac{1}{1 + \exp(-\alpha - \beta_1 x_{i1} - \beta_2 x_{i2} - \cdots - \beta_k x_{ik})} \quad (8.4)$$

This function is the *logistic regression function*. It is a nonlinear function as shown in Fig. 8.2.



**Figure 8.2** Logistic regression function used in classification.

As shown in Figs. 8.1 and 8.2, the output of logistic regression either falls in class 0 or class 1. The regression curve in this case is that of the logit function, which gives the probability that the output will fall in class 1 or class 0.

## 8.5 Maximum Likelihood Estimation

In linear regression, we used the method of least squares to estimate regression coefficients. In logistic regression, we will use another approach called the *maximum likelihood estimation*. The maximum likelihood estimate of a parameter is that value that maximizes the probability of the observed data.

The likelihood function is used to estimate the probability by observing the data. A “likelihood” is the probability that the observed values of the dependent variable may be predicted from the observed values of the independent variables. The likelihood varies from 0 to 1, like any other probability. Practically, it is easier to work with the logarithm of the likelihood function. This function is known as the *log-likelihood*.

In logistic regression, we invariably observe binary outcome. Let us consider the following example. Suppose in a population, each individual has the same probability,  $p$ , that an event will occur. In other words, for each individual in the sample of size  $n$ , the output  $Y_i = 1$ . This indicates that the event occurs for the  $i^{\text{th}}$  subject, otherwise  $Y_i = 0$ . The observed data are  $Y_1, \dots, Y_n$  corresponding to the inputs  $X_1, \dots, X_n$ .

The joint probability of the data (the likelihood) is given in Eq. (8.5).

$$L = \prod_{i=1}^n p(y/x)^{Y_i} (1 - p(y/x))^{1-Y_i} = p(y/x)^{\sum_{i=1}^n Y_i} (1 - p(y/x))^{n - \sum_{i=1}^n Y_i} \quad (8.5)$$

where  $p(y/x)$  is the probability of the output  $y$  given the input  $x$ . The log-likelihood function is computed by taking the natural logarithm of the likelihood, as given in Eq. (8.6).

$$l = \log(L) = \sum_{i=1}^n Y_i \log[p(y/x)] + \left( n - \sum_{i=1}^n Y_i \right) \log[1 - p(y/x)] \quad (8.6)$$

The probability of  $y$  occurring given  $x$  is given in Eq. (8.7).

$$p(y/x) = \frac{e^{a+\beta x}}{1 + e^{a+\beta x}} \quad (8.7)$$

Estimation of parameters  $\alpha$  and  $\beta$  is done using the first derivatives of log-likelihood, and solving them for  $\alpha$  and  $\beta$ . For this, iterative computing is used. An arbitrary value for the coefficients (usually 0) is first chosen. Then log-likelihood is computed and variations of coefficient values are observed. Reiteration is then performed until maximization of  $l$  (equivalent to maximizing  $L$ ). The results are the maximum likelihood estimates of  $\alpha$  and  $\beta$ .

### 8.5.1 Estimating the Parameters of Maximum Estimation

In logistic regression, we try to predict the probability of a given example belonging to the “1” class and not the “0” class. Specifically, we will try to learn a function of the expression given in Eq. (8.8), which gives the hypothesis for classification.

$$h(x_i) = g(\beta^T x_i) = \frac{1}{1 + e^{-\beta^T x_i}} \quad (8.8)$$

This can be defined by conditional probabilities for two labels (0 and 1) for  $i^{\text{th}}$  observation as given in Eq. (8.9).

$$\begin{aligned} P(y_i = 1|x_i; \beta) &= h(x_i) \\ P(y_i = 0|x_i; \beta) &= 1 - h(x_i) \end{aligned} \quad (8.9)$$

We can write Eq. (8.9) more compactly as given in Eq. (8.10):

$$P(y_i|x_i; \beta) = (h(x_i))^{y_i} (1 - h(x_i))^{1-y_i} \quad (8.10)$$

Maximum likelihood tries to find the value of coefficients ( $\beta_0, \beta_1$ ) such that the predicted probabilities [ $h(x)$  and  $(1 - h(x))$ ] are as close to the observed probabilities [ $y_i$  and  $(1 - y_i)$ ] as possible. In other words, for a binary classification (1/0), maximum likelihood will try to find values of  $\beta_0$  and  $\beta_1$  such that the resultant probabilities are closest to either 1 or 0. The likelihood function is then written as:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_i'=0} (1 - p(x_i')) \quad (8.11)$$

Applying Eq. (8.10) on Eq. (8.11), the likelihood function can be given as shown in Eq. (8.12). We consider  $\beta$  as a vector. In this case, we have  $\beta_0$  and  $\beta_1$ .

$$L(\beta) = \prod_{i=1}^n (h(x_i))^{y_i} (1 - h(x_i))^{1-y_i} \quad (8.12)$$

The multiplication can be converted into addition if we take the logarithm of the likelihood, which is given in Eq. (8.13).

$$\ell(\beta) = \log(L(\beta))$$

$$\text{or, } \ell(\beta) = \sum_{i=1}^n y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i)) \quad (8.13)$$

The cost function for logistic regression is proportional to inverse of likelihood of parameters. Hence, we can obtain an expression for cost function,  $J$ , using log-likelihood equation as:

$$J(\beta) = \sum_{i=1}^n -y_i \log(h(x_i)) - (1 - y_i) \log(1 - h(x_i)) \quad (8.14)$$

Our aim is to estimate  $\beta$  so that cost function is minimized.

### 8.5.2 Using Gradient Descent Algorithm

We take partial derivatives of  $J(\beta)$ , with respect to each  $\beta$ , to derive the stochastic gradient descent rule (only the final derived value is presented here) as given in Eq. (8.15).

$$\frac{\partial J(\beta)}{\partial \beta_j} = (h(x) - y)x_j \quad (8.15)$$

Here,  $y$  and  $h(x)$  represent the response vector and predicted response vector, respectively. Also,  $x_j$  is the vector representing the observation values for  $j^{\text{th}}$  feature.

Now, in order to get  $\min J(\beta)$ ,

Repeat {

$$\beta_j := \beta_j - \alpha \sum_{i=1}^n (h(x_i) - y_i)x_{ij}$$

(Simultaneously update all  $\beta_j$ )

}

where  $\alpha$  is called *learning rate* and needs to be set explicitly.

Gradient descent is one of the many ways to estimate  $\beta$ . Basically, these are more advanced algorithms that can be easily run in Python once cost function and gradients have been defined.

## 8.6 Example of Logistic Regression

Let us now see an example of logistic regression. In this example, we will predict whether a customer is a loan non-defaulter or not based on the amount of yearly savings (in lakhs). Let us consider the following situation:

A group of 20 customers, having yearly savings between 0 and 6 lakhs, are considered for analysis. How does the savings of a customer reflect in the repayment of loan. In other words, if there is any relationship between the savings of a customer and his/her being a loan defaulter.

The reason for using logistic regression in this problem is that the values of the dependent variable (whether non-defaulter and defaulter), while represented by “1” and “0”, are not cardinal numbers.

Table 8.1 shows the amount of saving of individual customers (in lakhs) and whether they are loan non-defaulters (0 indicates loan defaulters and 1 indicates loan non-defaulters)

**Table 8.1** Dataset showing the amount of saving (lakhs) and loan non-defaulter (yes/no)

Amount in Savings (Lakhs)	Loan Non-Defaulter
0.5	0
0.75	0
1.00	0
1.25	0
1.5	0
1.75	0
1.75	1

(Continued)

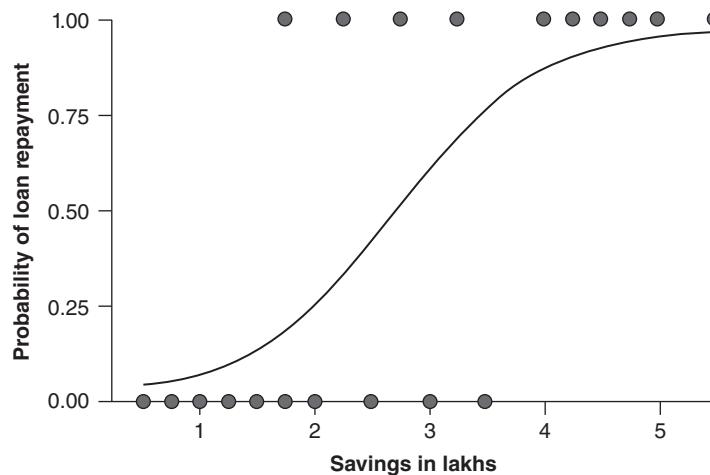
**Table 8.1** (Continued)

<i>Amount in Savings (Lakhs)</i>	<i>Loan Non-Defaulter</i>
2.00	0
2.25	1
2.5	0
2.75	1
3.00	0
3.25	1
3.50	0
4.00	1
4.25	1
4.5	1
4.75	1
5.00	1
5.5	1

Figure 8.3 shows savings (lakhs) versus the probability of loan repayment, with the logistic regression curve fitted to the data.

The logistic regression analysis using the maximum likelihood estimate gives the following output:

The coefficients are  $\beta_0 = -4.07778$  and  $\beta_1 = 1.5046$

**Figure 8.3** Probability of loan repayment with yearly savings.

These coefficients are entered into the logistic regression equation to estimate the probability of being a loan non-defaulter:

$$\text{Probability of being non-defaulter} = \frac{1}{1 + \exp(-(1.5046 \cdot \text{savings} - 4.0777))}$$

For example, for a customer with 2 lakhs saving per year, the estimated probability of being loan non-defaulter is as follows:

$$\text{Probability of being non-defaulter} = \frac{1}{1 + \exp(-(1.5046 \cdot 2 - 4.0777))}$$

Similarly, for a customer with 4 lakhs savings per year, the estimated probability of loan repayment is 0.87. The predicted and actual outputs for the non-defaulter/defaulter are shown in Table 8.2. Here, fitted value > 0.5 is taken as output 1, else output 0.

**Table 8.2** Predicted and actual values of defaulter/non-defaulter

Savings (Lakhs)	Non-Defaulter/Defaulter	Fitted Value	Prediction
0.5	0	0.034710025	0
0.85	0	0.049771971	0
1	0	0.070889852	0
1.25	0	0.100024715	0
1.5	0	0.139337907	0
1.75	0	0.190826302	0
1.75	1	0.190826302	0
2	0	0.255688447	0
2.25	1	0.333510508	0
2.5	0	0.421602115	0
2.75	1	0.514983013	1
3	0	0.607329347	1
3.25	1	0.692588758	1
3.5	0	0.766454783	1
4	1	0.874429026	1
4.25	1	0.910262967	1
4.5	1	0.936612324	1
4.75	1	0.955602124	1
5	1	0.969090667	1
5.5	1	0.985190994	1

**Table 8.3** Confusion matrix for the defaulter/non-defaulter prediction

	Predicted 0	Predicted 1
Actual 0	True negatives (TN)	False negatives (FN)
Actual 1	False positives (FP)	True positives (TP)

The matrix, also called *confusion matrix*, shown in Table 8.3 helps classify the values that were correctly predicted using the model built.

These classifications are used to calculate accuracy, precision (or positive predictive value), recall (or sensitivity), specificity, and negative predictive value. These are a few metrics used to identify the accuracy or fit of the logistic model based on the confusion matrix shown in Table 8.4.

**Table 8.4** Metrics to evaluate logistic regression for defaulter/non-defaulter prediction

Accuracy	$(TP + TN) / \text{Total Number of Observations}$
Precision or positive predictive value	$TP / (TP + FP)$
Negative predictive rate	$TN / (TN + FN)$
Sensitivity	$TP / (TP + FN)$
Specificity	$TN / (TN + FP)$

Let us consider the following example of confusion matrix:

	Predicted 0	Predicted 1
Actual 0	8 (TN)	2(FP)
Actual 1	2(FN)	8(TP)

$$\text{Accuracy} = \frac{(TP + TN)}{N} = \frac{16}{20} = 0.8$$

$$\text{Precision} = \frac{TP}{(TP + FP)} = \frac{8}{10} = 0.8$$

$$\text{Negative predictive rate} = \frac{TN}{(TN + FN)} = \frac{8}{10} = 0.8$$

$$\text{Sensitivity} = \frac{TP}{(TP + FN)} = \frac{8}{10} = 0.8$$

$$\text{Specificity} = \frac{TN}{(TN + FP)} = \frac{8}{10} = 0.8$$

In this sample dataset the accuracy obtained is 80%. This is how accuracy and precision are computed in a real time dataset.

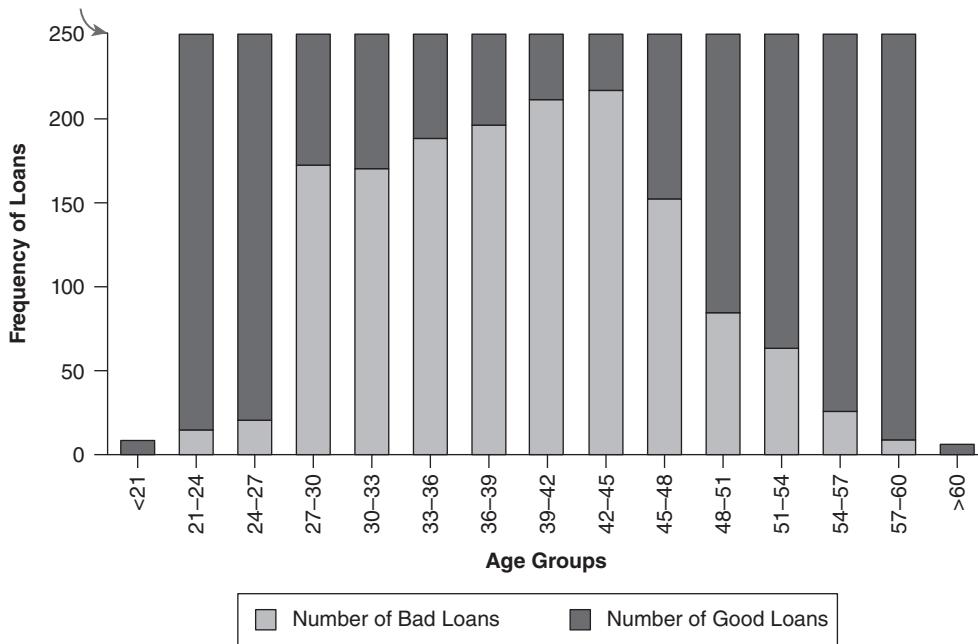
## Case Study

Logistic regression is an appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. Cynlate Bank disburses loans for vehicles. Mr. Rajesh took over as a CRO of the bank. He noticed that in the quarter of April–June 2015, more than 50,000 auto loans were disbursed. Of this figure, 3% was tagged as bad loans. Logistic regression is used in such real-time applications. Since age-group is one factor that decides loan repayment, therefore the loans are divided as good and bad loans as per the age group of persons. This is shown in Table 8.5.

**Table 8.5** Dataset on auto loan arranged as per age group

<i>Age Group</i>	<i>Total Number of Loans</i>	<i>Number of Bad Loans</i>	<i>Number of Good Loans</i>
<21	9	2	7
21–24	310	14	296
24–27	511	20	491
27–30	4000	172	3828
30–33	4568	169	4399
33–36	5698	188	5510
36–39	8209	197	8012
39–42	8117	211	7906
42–45	9000	216	8784
45–48	7600	152	7448
48–51	6000	84	5916
51–54	4000	64	3936
54–57	2000	26	1974
57–60	788	9	779
>60	6	0	6

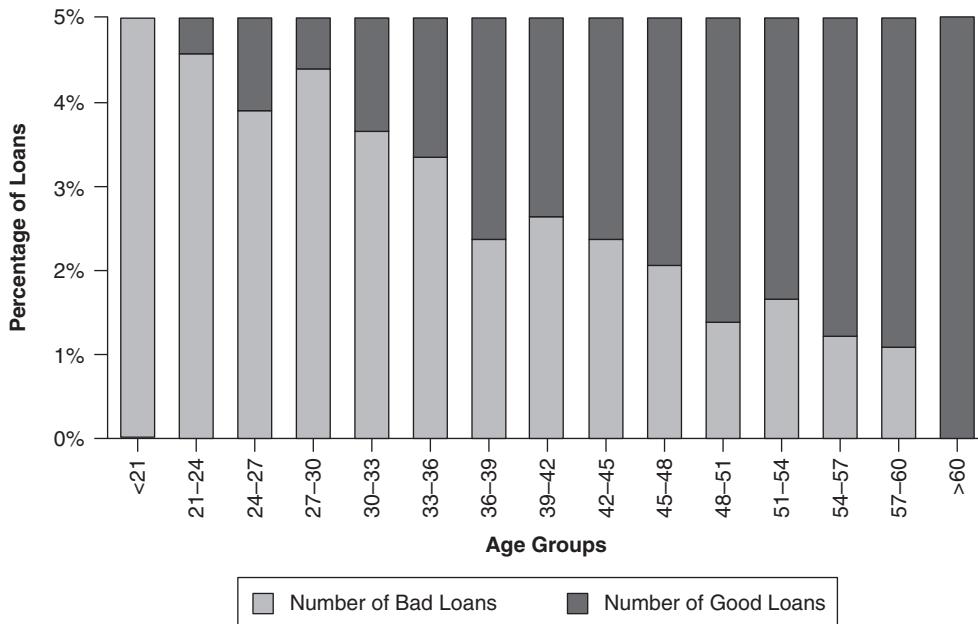
The histogram made based on Table 8.5 is shown in Fig. 8.4.

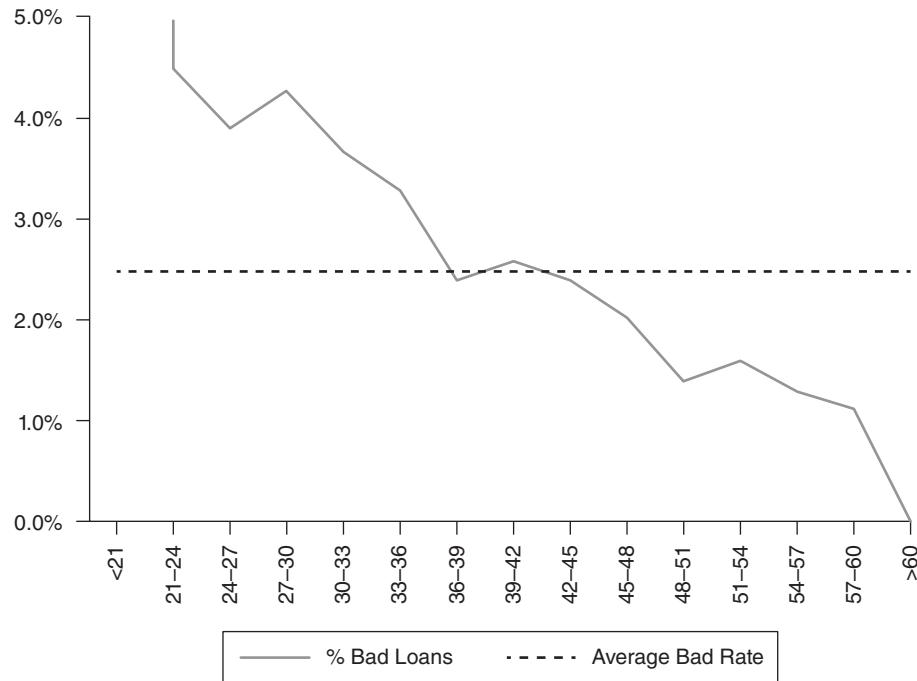


**Figure 8.4** Plot of frequency of loans as per age group.

The graph in Figure 8.4 shows that the distribution of loans across age groups is a reasonably smoothly distributed curve, without too many outliers. The maximum bad loans are in the age group of 42–45 years. The data is really thin on the fringe groups (that is, <21 and >60 years groups with only 9 and 6 data points, respectively).

On the other hand, the histogram made when age is scaled by 100% is shown in Fig. 8.5.





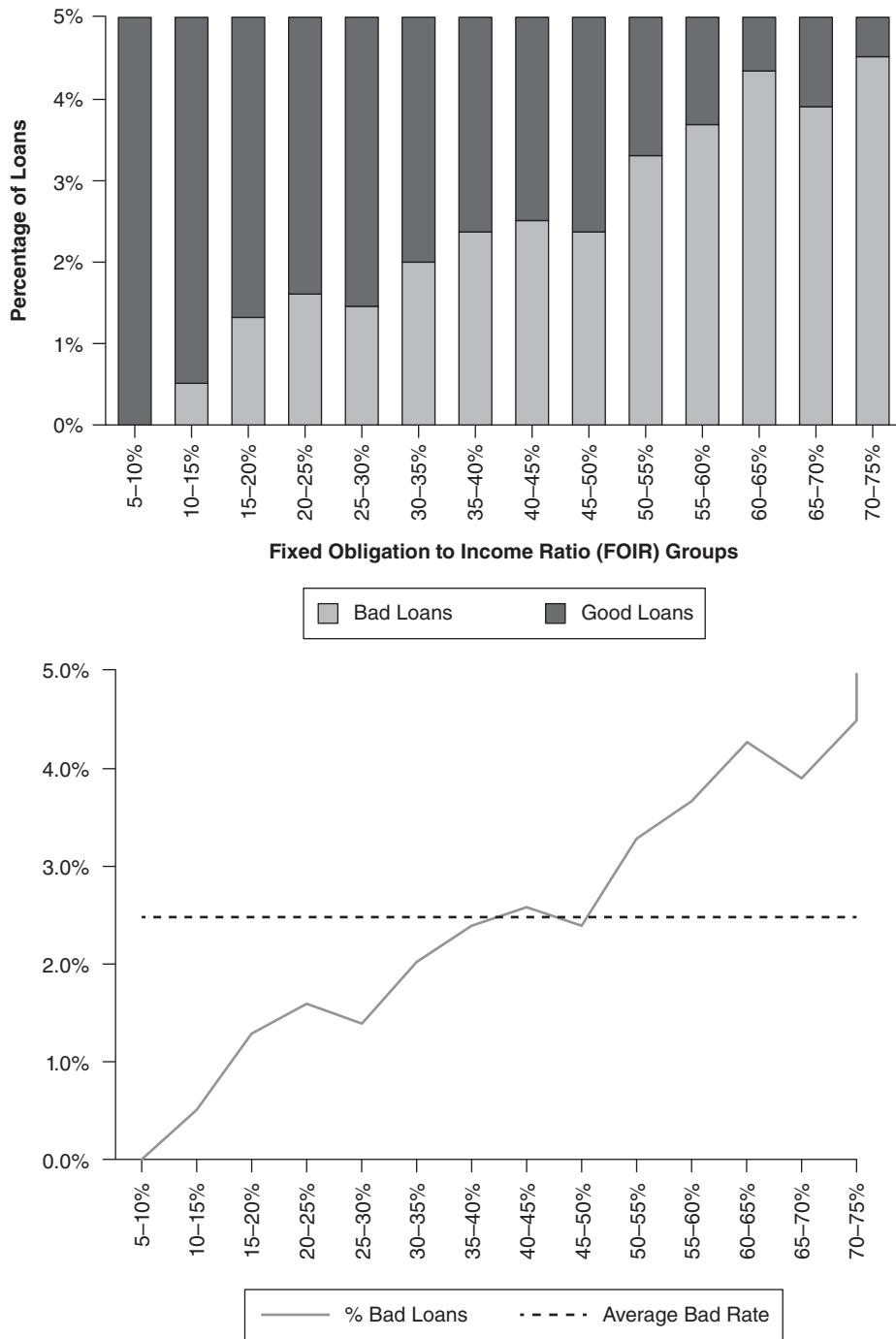
**Figure 8.5** Plots of frequency of loans as per age group, with age scaled by 100%.

The above plots are completely different from the original frequency count plot. They present the information in a completely different light.

There is a definite trend in terms of bad rates and age groups. As the borrowers get older, they are less likely to default on their loans. That is a good insight. However, the fringe groups (that is, <21 and >60 years groups) have thin data, and this information cannot be obtained from the normalized plot. Hence, you need to have the frequency plot handy to treat such thin data differently. A handy rule of thumb is to have at least 10 records of both cases (good and bad) before taking the information seriously, otherwise it is not statistically significant. The same check is made between the income groups and frequency of loans.

When corporate analysts try to analyze financials of a company, they often work with several financial ratios. Combined variables often convey much higher information. Seasoned analysts understand this really well. Moreover, variables creation is a creative exercise that requires sound domain knowledge.

For credit analysis, the ratio of the sum of obligations to income is highly informative since this provides an insight about percentage disposable income for the borrower. Let us try to understand this with an example. Neha has an annual income of ₹1,00,000. She has a home loan with an annual obligation (EMI) of ₹40,000 and a car loan with ₹10,000. Hence, she is spending ₹10,000 + ₹40,000 on paying the EMIs out of her income of ₹1,00,000. Her fixed obligation to income ratio (FOIR) in this case is equal to  $50/100 = 50\%$ . She is left with just 50% of her income to run her other expenses. The normalized histogram plot for FOIR is given in Fig. 8.6.



**Figure 8.6** Normalized histogram plot for FOIR.

Clearly, there is a directly proportional relationship between FOIR and bad rate. Additionally, FOIR has little correlation with age. The probability of a bad loan is given by

$$P(\text{BadLoan}) = \frac{e^Z}{1 + e^Z} = \frac{e^{\beta \times \text{Age} + \text{Constant}}}{1 + e^{\beta \times \text{Age} + \text{Constant}}}$$

The probability of a good loan is given by  $1 - P(\text{BadLoan})$ . The ratio of bad loan probability to good loan probability is given by

$$\frac{P(\text{BadLoan})}{P(\text{GoodLoan})} = e^{\beta \times \text{Age} + \text{Constant}}$$

This is categorized into course classes as shown in Table 8.6

**Table 8.6** Course groups for loan categorization

Age Group	Original Data				Course Classes					
	Total Number of Loans	Number of Bad Loans	Number of Good Loans	% Bad Loans	Age Group	Total Number of Loans	Number of Bad Loans	Number of Good Loans	% Bad Loans	Number of Course Groups
21–24	310	14	296	4.5						
24–27	511	20	491	3.9	21–30	4821	206	4615	4.3	G <sub>1</sub>
27–30	4000	172	3828	4.3	30–36	10266	357	9909	3.5	G <sub>2</sub>
30–33	4568	169	4399	3.7						
33–36	5698	188	5510	3.3	36–48	32926	776	32150	2.4	G <sub>3</sub>
36–39	8209	197	8012	2.4						
39–42	8117	211	7906	2.6						
42–45	9000	216	8784	2.4						
45–48	7600	152	7448	2.0	48–60	12788	183	12605	1.4	G <sub>4</sub>
48–51	6000	84	5916	1.4						
51–54	4000	64	3936	1.6						
54–57	2000	26	1974	1.3						
57–60	788	9	779	1.1						

The final logistic regression is given by

$$\frac{P(\text{BadLoan})}{P(\text{GoodLoan})} = e^{\beta_1 \times G_1 + \beta_2 \times G_2 + \beta_3 \times G_3 + \text{Constant}} \quad (8.16)$$

We can arrive at the values for those shown in Table 8.6 by using commercial software programs like R and Python, as given in Table 8.7.

**Table 8.7** Results of logistic regression (age groups and bad rates)

Predictor	Coefficient	Std. Er	Z	P	Odds Ratio
Constant	-4.232	0.074456	-56.84	0	
$G_1$	1.123	0.103026	10.9	0	3.07
$G_2$	0.909	0.0919	9.89	0	2.48
$G_3$	0.508	0.082846	6.14	0	1.66

Substituting the coefficients in the Eq. (8.16) we get

$$\frac{P(\text{BadLoan})}{P(\text{GoodLoan})} = e^{1.123 \times G_1 + 0.909 \times G_2 + 0.508 \times G_3 - 4.232}$$

$G_1$ ,  $G_2$ , and  $G_3$  are the categories based on age groups. They can take values 0 or 1. At any point of time only one can be 1. So if  $G_1$  is equal to 1, then this ratio becomes

$$\frac{P(\text{BadLoan})}{P(\text{GoodLoan})} = e^{1.123 - 4.232} = 0.0446$$

Similarly, we could find the estimated value of bad rate for  $G_1$ .

$$P(\text{BadLoan}) = \frac{0.0446}{1 + 0.0446} = 4.3\%$$

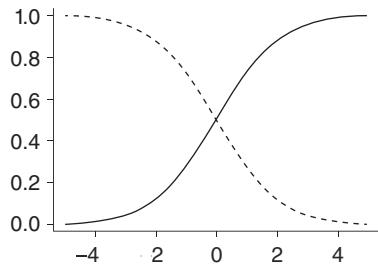
Hence, we can see that logistic regression is doing a good job for estimation of bad rate. So logistic regression helps identify the variation of which variable deters the output class in general.

## Summary

- Logistic regression comes under the category of classification algorithms, where the focus is on getting the label for the output class.
- The concept of regression techniques cannot be applied in case of logistic regression because the output labels are binary in nature.
- Logistic regression is used in those class of problems like identifying whether an email is spam or not, whether the tumor is benign or malignant, whether a user would invest in term deposit or not.
- Logistic regression uses the concept of odds ratio, which is different from probability. Odds ratio is the ratio of probability of an event to the probability of no event.
- Odds ratio is used because it does not have upper bounds like probability. Therefore, it is easier to denote the effect of inputs on outputs using this ratio.
- The ratio of odds is called logit function.
- Maximum likelihood estimation is used for estimating the values of the logistic regression parameters.
- The coefficients are iteratively computed using the concept of gradient descent algorithm.
- After the coefficients are computed, the logit function helps estimate whether the output belongs to the yes or the no class.
- The confusion matrix helps in arriving at the accuracy of the predictions.

## Multiple-Choice Questions

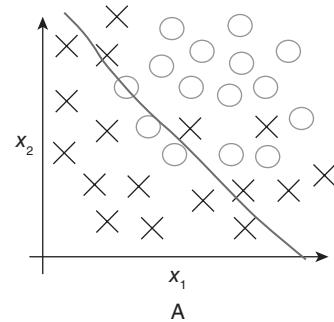
1. Which of the following methods do we use to best fit the data in logistic regression?
  - (a) Least square error
  - (b) Maximum likelihood
  - (c) Jaccard distance
  - (d) Both (a) and (b)
2. Which of the following is true regarding the logistic function for any value  $x$ , given the following statements:
  - i. Logistic( $x$ ) is a logistic function of any number  $x$ .
  - ii. Logit( $x$ ) is a logit function of any number  $x$ .
  - iii. Logit\_inv( $x$ ) is an inverse logit function of any number  $x$ .
  - (a) Logistic( $x$ ) = Logit( $x$ )
  - (b) Logistic( $x$ ) = Logit\_inv( $x$ )
  - (c) Logit\_inv( $x$ ) = Logit( $x$ )
  - (d) None of the above
3. Suppose you applied a logistic regression model on a given data and got a training accuracy  $X$  and testing accuracy  $Y$ . Now, you want to add a few new features in the same data. Select the option(s) which is/are correct in such a case.
  - (a) Training accuracy increases
  - (b) Training accuracy increases or remains the same
  - (c) Testing accuracy decreases
  - (d) Testing accuracy increases or remains the same
4. Below are two different logistic models with different values for  $\beta_0$  and  $\beta_1$ .



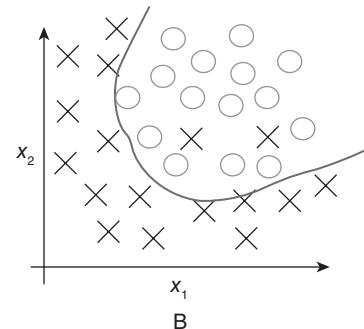
Which of the following statement(s) is/are true about  $\beta_0$  and  $\beta_1$  values of two logistics models

(solid line, dashed line)? Note: consider  $Y = \beta_0 + \beta_1 \times X$ . Here,  $\beta_0$  is intercept and  $\beta_1$  is coefficient.

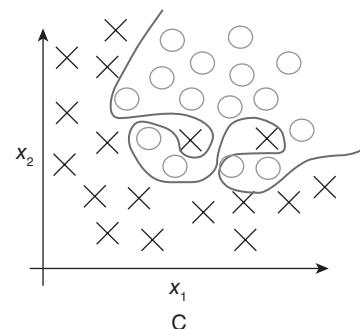
- (a)  $\beta_1$  for solid line is greater than dashed line
  - (b)  $\beta_1$  for solid line is lower than dashed line
  - (c)  $\beta_1$  for both models is same
  - (d) None of the above
5. Below are the three scatter plots (A, B, and C) and hand-drawn decision boundaries for logistic regression.



A



B



C

- Which of the figure shows that the decision boundary is overfitting the training data?
- A
  - B
  - C
  - None of the above
6. Let us suppose the following graph is a cost function for logistic regression.
- 
- Now, how many local minimas are present in the graph?
- 1
  - 2
  - 3
  - 4
7. Logistic regression comes under the category of what algorithm?
- Supervised regression
  - Supervised classification
- (c) Unsupervised learning  
 (d) None of the above
8. Can a logistic regression classifier perform a perfect classification on the data given below?
- 
- Note: You can use only  $x_1$  and  $x_2$  variables where  $x_1$  and  $x_2$  can take only two binary values (0, 1).
- No. The patterns are not linearly separable
  - Yes. There is no necessity of the concept of linear separability here
  - Logistic regression cannot be used in this scenario
  - None of the above

### Very Short Answer Questions

- When is logistic regression considered to be ordinal?
- What is logit transformation?
- What is the equivalent of exponential function of the linear regression function?
- Why is maximum likelihood estimation method used?
- When can you conclude that the model fit is significantly improved?

### Short Answer Questions

- Why is it called “logistic regression” if it is used for classification?
- What is the interpretation of odds ratio in logistic regression?
- How do you interpret logit coefficients?

## Review Questions

1. How is logistic regression different from linear regression?
2. Why does logistic regression come under the category of classifications problems?
3. Explain odds ratio.
4. How does maximum likelihood estimation help in computing the parameters of logit function?
5. Explain, with an example, how to analyze the data given coefficients and actual output using logistic regression.

## Answers

---

### Multiple-Choice Questions

1. (b)    2. (a)    3. (a) and (b)    4. (b)    5. (c)    6. (c)    7. (b)    8. (c)

# 9

# Decision Tree

## LEARNING OBJECTIVES

- To comprehend the basics of classification and decision trees.
- To understand decision tree algorithm by learning the concept of entropy, gain ratio, and Gini index.
- To make decision tree algorithm issues quite clear.
- To give an insight into rule-based reasoning along with sequential covering algorithm.

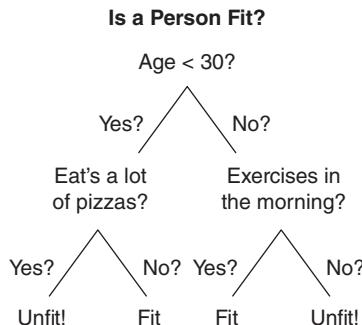
## LEARNING OUTCOMES

- Students will be able to differentiate between classification and regression.
- Students will be able to build a decision tree model.
- Students will be able to judge whether the model is overfitting or underfitting from the datasets.
- Students will be able to find the rules using sequential covering algorithm.

### 9.1 Introduction to Classification and Decision Tree

Decision tree algorithms come under the category of supervised learning algorithms. We have already learned in Chapter 8 that logistic regression is a type of classification algorithm where the output is categorical. In logistic regression, we try to find the probability of prediction. In the case of decision trees, we build a tree in which the leaf nodes contain the output category. We can predict the class of the output based on the rules generated from the tree structure. Learned trees can be represented as a set of IF–THEN rules as well.

The topmost node in a decision tree is the root node. A typical decision tree is shown in Fig. 9.1.



**Figure 9.1** Tree structure to represent whether a person is fit or not.

Figure 9.1 predicts whether a person is fit or not based on whether he/she eats a lot of pizzas and whether he/she exercises in the morning. This prediction involves people who are below 30 years of age.

## 9.2 Problem Solving Using Decision Trees

---

Decision trees can be used to solve problems that have the following features:

1. Instances or tuples are represented as attribute value pairs, where the attributes take a small number of disjoint possible values. For example, temperature can be defined as hot, cold, and medium instead of continuous values.
2. The target function has discrete output values such as yes or no. For example, a person would apply for a house loan based on factors including income level, educational qualification (graduate or not graduate), gender, and age group. Here applying for house loan can take discrete values yes or no. Decision trees require disjunctive descriptions which implies that the output of the decision tree can be represented using a rule-based classifier.
3. A decision tree can be used when training data contains errors and when it contains missing attribute values.

A tree is made by splitting the dataset into subsets based on attribute values. This process is repeated on each derived subset of data by means of recursive partitioning. The recursion is completed when the entire dataset is partitioned. Decision trees can handle multidimensional data.

The construction of a decision tree classifier does not require any domain knowledge or parameter setting and is therefore appropriate for exploratory knowledge discovery. In general, a decision tree classifier has good accuracy.

Decision tree induction is a typical inductive approach to learn knowledge on classification. Thus, decision tree is used in basic classification problems.

## 9.3 Basic Decision Tree Learning Algorithm

---

A decision tree learning algorithm is a variation of the top-down Greedy search algorithm. Two basic algorithms are the Iterative Dichotomiser 3 (ID3) algorithm and the C4.5 algorithm, which are explained in the following subsections. The best attribute among the set of input attributes is first selected. This attribute is chosen to be the root node. Descendent of root node is created for each possible value of the root node. The algorithm is repeated using training examples associated with each descendant node to select the best attribute to test at that point in the tree. Therefore, the algorithm gets the name “iterative” because it iteratively checks for other attributes till all the leaf nodes are complete.

### 9.3.1 Attribute Selection Measures

Attribute selection is a heuristic measure for selecting the splitting criterion that “best” separates a given data of class-labeled training tuples into individual classes. Attribute selection measures are also known as *splitting rules* because they determine how the tuples at a given node are to be split. This measure provides a ranking for each attribute describing the given training tuples. The attribute having the best score for the measure is chosen as the splitting attribute for the given tuples. The three popular attribute selection measures are information gain, gain ratio, and Gini index.

#### 9.3.1.1 Information Gain

In decision tree algorithm, the main selection measure that is used is information gain. This algorithm will always try to maximize information gain, which measures how well a given attribute separates the training

examples according to their target classification. When the decision tree is constructed, the attribute with highest information gain will be tested first. This becomes the root node and the split is made based on the values of the root node. Entropy is an entity that controls the split in data. It computes the homogeneity of examples. The formula for entropy is

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \quad (9.1)$$

where  $p$  stands for the probability of various instances under consideration.

The entropy is 0 if all members of  $S$  belong to the same class and 1 if there are equal numbers of positive and negative examples. Entropy ranges between 0 and 1 if there is unequal number of positive and negative examples. It specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of  $S$ . It is computed to the base 2 because encoding length is measured in bits.

As already mentioned, information gain is the measure of effectiveness of an attribute in classifying the training data. In other words, it is the expected reduction in entropy. The formula for information gain is

$$\begin{aligned} \text{Gain}(S, A) &\equiv \text{Entropy}(S) \\ &- \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \end{aligned} \quad (9.2)$$

### 9.3.1.2 Gain Ratio

The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values. For example, consider an attribute that acts as a unique identifier such as product ID. A split on product ID would result in a large number of partitions (as many as there are values), each one containing just one tuple. The information gain obtained by partitioning this attribute is maximum. Clearly, such a partitioning is useless for classification. C4.5 algorithm, a successor of ID3, uses an extension to information gain known as *gain ratio* which attempts to overcome this bias. It applies a kind of normalization to information gain using a “split information” value as given by

$$\text{Split Info}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right) \quad (9.3)$$

Gain ratio is defined as

$$\text{Gain Ratio}(A) = \frac{\text{Gain}(A)}{\text{Split Info}_A(D)}$$

The attribute with the maximum gain ratio is selected as the splitting attribute.

#### Solved Problem 9.1

Compute the gain ratio for the attribute income in the dataset given in Table 9.1.

**Table 9.1** Sample dataset

<b>S. No.</b>	<b>Age</b>	<b>Income</b>	<b>Student</b>	<b>Credit</b>	<b>Buy</b>
1	<30	High	No	Fair	No
2	<30	High	No	Excellent	No
3	31–40	High	No	Fair	Yes

(Continued)

**Table 9.1** (Continued)

<i>S. No.</i>	<i>Age</i>	<i>Income</i>	<i>Student</i>	<i>Credit</i>	<i>Buy</i>
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31–40	Low	Yes	Excellent	Yes
8	<30	Medium	No	Fair	No
9	<30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<30	Medium	Yes	Excellent	Yes
12	31–40	Medium	No	Excellent	Yes
13	31–40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

**Solution:**

The split information of income is computed as shown below. A test on income splits the data into three partitions – low, medium, and high – containing four, six, and four tuples, respectively. To compute the gain ratio of income, we have

$$\text{Split Info}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)$$

$$\text{Split Info}_{\text{income}}(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 1.557$$

We have Gain (income) = 0.029. Hence

$$\text{Gain Ratio (income)} = \frac{0.029}{1.557} = 0.019$$

**9.3.1.3 Gini Index**

The Gini index is used in Classification and Regression Trees (CART). This index measures the impurity of set of training tuples. Mathematically,

$$\text{Gini}(D) = 1 - \sum_{i=1}^m p_i^2 \quad (9.4)$$

where  $p_i$  is the probability that a tuple belongs to class  $C_i$ .

The Gini index considers a binary split for each attribute. Consider the case where  $A$  is a discrete-valued attribute having  $v$  distinct values,  $\{a_1, a_2, \dots, a_v\}$  occurring in training set. To determine the best binary split

on  $A$ , we examine all the possible subsets that can be formed using known values of  $A$ . We exclude the power set and the empty set from consideration since, conceptually, they do not represent a split. Therefore, there are  $2^v - 2$  possible ways to form two partitions of the data based on a binary split on  $A$ . For example, if income has three possible values, namely {low, medium, high} then the possible subsets are {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and {}. We exclude the power set {low, medium, high} and the empty set.

When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on  $A$  partitions  $D$  into  $D_1$  and  $D_2$ , the Gini index of  $D$  given that partitioning is

$$\text{Gini}_A(D) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{|D|} \text{Gini}(D_2) \quad (9.5)$$

For each attribute, each of the possible binary splits is considered. For a discrete-valued attribute, the subset that gives the minimum Gini index for that attribute is selected as its splitting subset.

The reduction in impurity that would be incurred by a split on attribute  $A$  is

$$\Delta \text{Gini}(A) = \text{Gini}(D) - \text{Gini}_A(D)$$

The attribute that maximizes the reduction in impurity (or equivalently has the minimum Gini index) is selected as the splitting attribute.

With respect to dataset given in Table 9.1, there are nine tuples belonging to the class buys\_computer = yes and the remaining five tuples belong to the class buys\_computer = no. The Gini index for the training data  $D$  is

$$\text{Gini}(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459$$

To find the splitting criterion for the tuples in  $D$ , we need to compute the Gini index for each attribute. Let us start with the attribute income and consider each of the possible splitting subsets. Consider the subset {low, medium}. This would result in 10 tuples in partition  $D_1$  satisfying the condition “income  $\in$  {low, medium}”. The remaining four tuples of  $D$  would be assigned to partition  $D_2$ . The Gini index value computed based on this partitioning is

$$\begin{aligned} & \text{Gini}_{\text{income } \in \{\text{low, medium}\}}(D) \\ &= \frac{10}{14} \text{Gini}(D_1) + \frac{4}{14} \text{Gini}(D_2) \\ &= \frac{10}{14} \left( 1 - \left( \frac{7}{10} \right)^2 - \left( \frac{3}{10} \right)^2 \right) + \frac{4}{14} \left( 1 - \left( \frac{2}{4} \right)^2 - \left( \frac{2}{4} \right)^2 \right) \\ &= 0.443 \\ &= \text{Gini}_{\text{income } \in \{\text{high}\}}(D) \end{aligned}$$

Similarly, the Gini index values for splits on the remaining subsets are 0.458 (for the subsets {low, high} and {medium}) and 0.450 (for the subsets {medium, high} and {low}). Therefore, the best binary split for attribute income is on {low, medium} or {high} because it minimizes the Gini index.

### 9.3.2 Iterative Dichotomiser 3 (ID3)

In decision tree learning, the most popular algorithm is the Iterative Dichotomiser 3 (ID3) algorithm. It is used to generate a decision tree from a given training dataset.

#### 9.3.2.1 Stopping Condition of ID3

Recursive computation of the selection attribute can be stopped in any one of the following cases:

1. Every element in the subset belongs to the same class and then the node is turned into a leaf node and labelled with the name of that class.
2. There can be no more attributes to be selected, but the examples still do not belong to the same class. Then the final node is made a leaf node and labelled with the most common class in the subset.
3. There can be no more examples in the subset, which happens when no example in the parent set was found to match a specific value of the selected attribute. For example, let us consider the absence of an adult among the children's dataset of age over 13 years. In such a case, the category changes from child to teenager. Then a leaf node is created and labelled with the most common class of the example in the parent node's set.

#### 9.3.2.2 Limitations of ID3

The following are the main disadvantage of ID3:

1. It maintains only a single current hypothesis as it searches through the space of decision trees.
2. It does not have the ability to determine alternative decision trees.
3. It does not perform backtracking in search. Hence, there are chances of getting stuck in local optima.
4. It is less sensitive to error because information gain, which is a statistical property is used.

## 9.4 Popularity of Decision Tree Classifiers

---

The construction of decision tree classifiers does not require any domain knowledge or parameter setting and is thus appropriate for exploratory knowledge discovery. For example, in traditional data mining algorithms, we need to know about the system under consideration to classify the dataset. This is not the case in decision tree algorithms. Another issue with traditional algorithm is some parameters like similarity index need to be fed into the algorithm to classify data. This issue is not there in decision tree since the classification is based on probability alone.

Decision trees can handle multidimensional data. The representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. Moreover, decision tree classifiers have good accuracy.

## 9.5 Steps to Construct a Decision Tree

---

To construct a decision tree, first the dataset with corresponding input and output pairs is taken. The data is split into training data and test data. Let us assume that the training data has  $n$  attributes. One row of the training data sample is of the form  $\langle x_{i1}, x_{i2}, \dots, x_{in}, y_i \rangle$  for  $i$  tuples with  $n$  attributes. From these  $n$  attributes, the best attribute to become the root node is chosen. Based on this attribute the tuples are split. This procedure is iteratively carried out until the leaf nodes contain tuples of the same output class. The following are steps to construct a decision tree:

1. Compute the entropy for the given dataset.
2. For every attribute/feature:
  - Calculate entropy for all categorical values.

- Take the average information entropy of the current attribute.
  - Calculate gain for the current attribute.
3. Pick the highest gain attribute.
  4. Repeat until the desired tree is complete.

We will follow these steps while solving the numerical problem below.

### Solved Problem 9.2

Consider the dataset given in Table 9.1. Construct the decision tree to predict whether a customer will buy a computer or not.

#### **Solution:**

#### **Step 1: Compute Entropy**

For computing the entropy, first we have to count the number of distinct outputs. In this case, decision has to be taken whether a costumer will buy a computer or not. Out of the total 14 samples, based on the attributes like age, income, student, and credit rating, 9 samples belong to the “yes” category and 5 samples belong to the “no” category.

Number of “Yes” = 9

Number of “No” = 5

We have

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

$$= -\left(\frac{9}{14}\right) \log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right) \log_2\left(\frac{5}{14}\right) = 0.940$$

The initial entropy is computed using Eq. (9.1). For each of the attributes – age, income, student, and credit – the count of output class “buys\_computer” belonging to the yes or the no class is first computed.

#### **Step 2: Calculations for Every Attribute**

In this case, the attributes are age, income, student, and credit. Based on whether a person buys a computer or not for the age group category, we can arrange the data as shown in Table 9.2.

**Table 9.2** Data rearranged as per the attribute “age”

<i>Attribute: AGE</i>	<i>Buy Computer</i>	
	<i>Yes</i>	<i>No</i>
<30	2	3
31–40	4	0
>40	3	2
Gain	0.246	

We then carry out the calculations for each attribute. The entropy for each category of age (<30, 31–40, and >40) is computed. The gain of age is the difference between the overall entropy and the product of probability and entropy of individual instances. We have

$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

$$\text{Gain}(S, \text{Age}) = \text{Entropy}(S) - \left(\frac{5}{14}\right)^* \text{Entropy}(S_{<30}) - \left(\frac{4}{14}\right)^* \text{Entropy}(S_{31-40})$$

$$- \left(\frac{5}{14}\right)^* \text{Entropy}(S_{>40})$$

$$= 0.940 - \left(\frac{5}{14}\right)^* 0.971 - \left(\frac{4}{14}\right)^* 0 - \left(\frac{5}{14}\right)^* 0.971 = 0.246$$

Similarly, we calculate the gain of the other attributes.

**For Income Attribute:**

Based on the income of the person, we can arrange the data as shown in Table 9.3.

**Table 9.3** Data rearranged as per the attribute "income"

<i>Attribute: INCOME</i>	<i>Buy Computer</i>	
	<i>Yes</i>	<i>No</i>
High	2	2
Medium	4	2
Low	3	1
Gain	0.029	

$$\text{Gain}(S, \text{Income}) = \text{Entropy}(S) - \left(\frac{4}{14}\right)^* \text{Entropy}(S_{\text{High}}) - \left(\frac{5}{14}\right)^* \text{Entropy}(S_{\text{Med}})$$

$$- \left(\frac{5}{14}\right)^* \text{Entropy}(S_{\text{Low}})$$

$$= 0.029$$

**For Occupation Attribute:**

Based on the occupation of the person, we can arrange the data as shown in Table 9.4.

**Table 9.4** Data rearranged as per the attribute “occupation”

Attribute: STUDENT	Buy Computer	
	Yes	No
Yes	6	1
No	3	4
Gain	0.151	

$$\begin{aligned} \text{Gain}(S, \text{Student}) &= \text{Entropy}(S) - \left(\frac{7}{14}\right)^* \text{Entropy}(S_{\text{Yes}}) - \left(\frac{7}{14}\right)^* \text{Entropy}(S_{\text{No}}) \\ &= 0.151 \end{aligned}$$

**For Credit Attribute:**

Based on the person's credit review, we can arrange the data as shown in Table 9.5.

**Table 9.5** Data rearranged as per the attribute “credit”

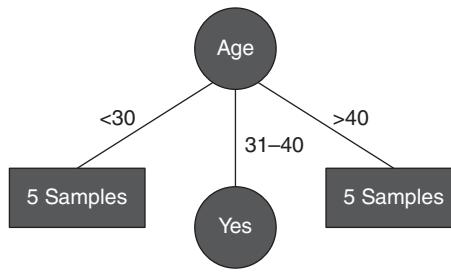
Attribute: CREDIT	Buy Computer	
	Yes	No
Fair	6	2
Excellent	3	3
Gain	0.046	

$$\begin{aligned} \text{Gain}(S, \text{Credit}) &= \text{Entropy}(S) - \left(\frac{8}{14}\right)^* \text{Entropy}(S_{\text{Fair}}) - \left(\frac{6}{14}\right)^* \text{Entropy}(S_{\text{Excellent}}) \\ &= 0.046 \end{aligned}$$

**Step 3: Pick the Highest Gain Attribute**

The attribute having the maximum gain is taken as the root attribute. Gain is a parameter that measures how much would be gained by branching on any attribute, say,  $A$ . It estimates the expected reduction in the information requirement caused by knowing the value of  $A$ . Thus, attribute  $A$  with the highest information gain is chosen as the splitting attribute at node  $N$ . This will result in the “best classification”, so that the amount of information still required to terminate classifying the tuples is minimal.

In this case, gain ( $S$ , Age) is maximum; hence it is chosen as the splitting attribute. Using age as the root node, the initial split of attributes belonging to age  $< 30$ , age between 31 and 40, age  $> 40$  is arrived (Fig. 9.2). In the category age between 31 and 40, all the leaf nodes belong to the same category of buys\_computer = yes and hence it is a leaf node and not to be considered further.



**Figure 9.2** Age is chosen as the root node and the splitting of attributes are based on age.

#### Step 4: Repeat until Decision Tree is Complete

Recursively, we then find the splitting attributes at the next level. We now consider only part of the sample dataset for unclassified samples. Here, we have 5 samples for Age <30 and 5 samples for Age >40.

##### For Age <30:

The dataset for this category has 5 samples as given in Table 9.6.

**Table 9.6** Dataset of persons whose age is <30

S. No.	Age	Income	Student	Credit	Buy
1	<30	High	No	Fair	No
2	<30	High	No	Excellent	No
8	<30	Medium	No	Fair	No
9	<30	Low	Yes	Fair	Yes
11	<30	Medium	Yes	Excellent	Yes

Based on whether a person buys computer or not and based on whether the buyer is student or not in the dataset, we construct Table 9.7.

**Table 9.7** Dataset as per attribute “student” of people whose age is <30

Attribute: STUDENT	Buy Computer	
	Yes	No
Yes	2	–
No	–	3
Gain	0.971	

$$\begin{aligned}
 \text{Gain}(\text{Age } <30, \text{Student}) &= \text{Entropy}(\text{Age } <30) - \left(\frac{2}{5}\right) * \text{Entropy}(S_{\text{Yes}}) - \left(\frac{3}{5}\right) * \text{Entropy}(S_{\text{No}}) \\
 &= 0.971 - \left(\frac{2}{5}\right) * 0 - \left(\frac{3}{5}\right) * 0 \\
 &= 0.971
 \end{aligned}$$

Similarly, we can construct Table 9.8 based on the person's credit review.

**Table 9.8** Dataset as per the attribute "credit" of people whose age is <30

<i>Attribute: CREDIT</i>	<i>Buy Computer</i>	
	<i>Yes</i>	<i>No</i>
Fair	1	2
Excellent	1	1
Gain	0.0208	

$$\begin{aligned} \text{Gain}(\text{Age} < 30, \text{Credit}) &= \text{Entropy}(\text{Age} < 30) - \left( \frac{3}{5} \right) * \text{Entropy}(S_{\text{Fair}}) - \left( \frac{2}{5} \right) * \text{Entropy}(S_{\text{Excellent}}) \\ &= 0.971 - \left( \frac{3}{5} \right) * 0.917 - \left( \frac{2}{5} \right) * 1 = 0.0208 \end{aligned}$$

Similarly, we can construct Table 9.9 based on the income.

**Table 9.9** Dataset as per the attribute "income" of people whose age is <30

<i>Attribute: INCOME</i>	<i>Buy Computer</i>	
	<i>Yes</i>	<i>No</i>
High	0	2
Medium	1	1
Low	1	0
Gain	0.57	

$$\begin{aligned} \text{Gain}(\text{Age} < 30, \text{Income}) &= \text{Entropy}(\text{Age} < 30) - \left( \frac{2}{5} \right) * \text{Entropy}(S_{\text{High}}) \\ &\quad - \left( \frac{2}{5} \right) * \text{Entropy}(S_{\text{Medium}}) - \left( \frac{1}{5} \right) * \text{Entropy}(S_{\text{High}}) \\ &= 0.971 - \left( \frac{2}{5} \right) * 0 - \left( \frac{2}{5} \right) * 1 - \left( \frac{1}{5} \right) * 0 = 0.57 \end{aligned}$$

Since gain (Age <30, Student) is maximum, it is chosen as the splitting attribute.

**For Age >40:**

The dataset for this category has 5 samples as given in Table 9.10.

**Table 9.10** Dataset of persons whose age is >40

S. No.	Age	Income	Student	Credit	Buy
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
10	>40	Medium	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

Based on whether a person buys computer or not and based on the status of his/her credit in the dataset, we construct Table 9.11.

**Table 9.11** Dataset as per attribute “credit” of people whose age is >40

Attribute: CREDIT	Buy Computer	
	Yes	No
Fair	3	0
Excellent	0	2
Gain	0.971	

$$\begin{aligned}
 \text{Gain}(\text{Age} > 40, \text{Credit}) &= \text{Entropy}(\text{Age} > 40) - \left(\frac{3}{5}\right) * \text{Entropy}(S_{\text{Fair}}) - \left(\frac{2}{5}\right) * \text{Entropy}(S_{\text{Excellent}}) \\
 &= 0.971 - \left(\frac{3}{5}\right) * 0 - \left(\frac{2}{5}\right) * 0 \\
 &= 0.971
 \end{aligned}$$

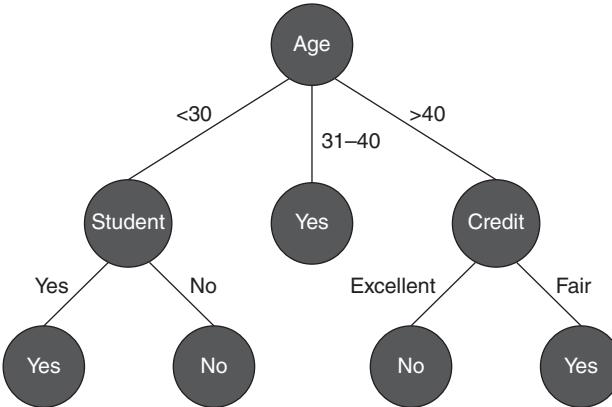
Similarly, we can construct Table 9.12 for income.

**Table 9.12** Dataset as per attribute “income” of people whose age is >40

Attribute: INCOME	Buy Computer	
	Yes	No
High	0	0
Medium	2	1
Low	1	1
Gain	0.0208	

$$\begin{aligned}\text{Gain}(\text{Age} > 40, \text{Income}) &= \text{Entropy}(\text{Age} > 40) - \left(\frac{3}{5}\right) * \text{Entropy}(S_{\text{Medium}}) - \left(\frac{2}{5}\right) * \text{Entropy}(S_{\text{Low}}) \\ &= 0.971 - \left(\frac{3}{5}\right) * 0.917 - \left(\frac{2}{5}\right) * 1 = 0.0208\end{aligned}$$

Since gain (Age > 40, Credit) is maximum, it is chosen as the splitting attribute. Thus, we can draw the final decision tree as shown in Fig. 9.3.



**Figure 9.3** Final tree for buys\_computer.

## 9.6 Classification Using Decision Trees

Given a tuple  $X$  for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node of the decision tree shown in Fig. 9.3, which holds the class prediction for that tuple. Decision trees can easily be converted into classification rules. For example, from the tree we can infer that a customer will purchase a computer if his/her age is less than 30 and if he/she is a student.

## 9.7 Issues in Decision Trees

A decision tree is not the best classifier because it considers almost all attributes for classification, resulting in a large number of classes. It suffers mainly from underfitting and overfitting of data. The following are practical issues in construction of decision trees:

1. Handling continuous attributes.
2. Choosing an appropriate attribute selection measure.
3. Handling training data with missing attribute values.
4. Handling attributes with differing costs.
5. Improving computational efficiency.

The main issues related to tree size and how to overcome them are explained in detail in subsequent subsections.

### 9.7.1 Underfitting and Overfitting

Underfitting occurs when a machine learning algorithm cannot capture the underlying trend of the data. It implies that the classification is not considering the required information from the dataset. This results in different categories of dataset being merged into the same category after classification, which is not ideal for any classification algorithm.

Overfitting occurs when a machine learning algorithm captures the noise of the data. This happens when the dataset is classified into too many categories by taking the slightest variation in the dataset into consideration. Similar data gets classified into different classes, making the tree grow infinitesimally. This leads to too many categories of data which is again not suitable for a classification algorithm. Intuitively, overfitting occurs when the model or the algorithm fits the data too well.

#### 9.7.1.1 Avoiding Overfitting of Data

A decision tree algorithm can produce trees that overfit the training examples when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. We say that a hypothesis overfits the training examples if a different hypothesis that fits the training examples with less accuracy performs better over the entire distribution of instances.

Given a hypothesis space  $H$ , a hypothesis  $h \in H$  is said to overfit the training data if there exists some alternative hypothesis  $h' \in H$ , such that  $h$  has smaller error than  $h'$  over the training examples, but  $h'$  has a smaller error than  $h$  over the entire distribution of instances.

There are several approaches to avoid overfitting in decision tree learning. These can be grouped into two classes:

1. Approaches that stop the growth of tree before it reaches the point where it perfectly classifies the training data.
2. Approaches that allow the tree to overfit the data, and then post-prune it.

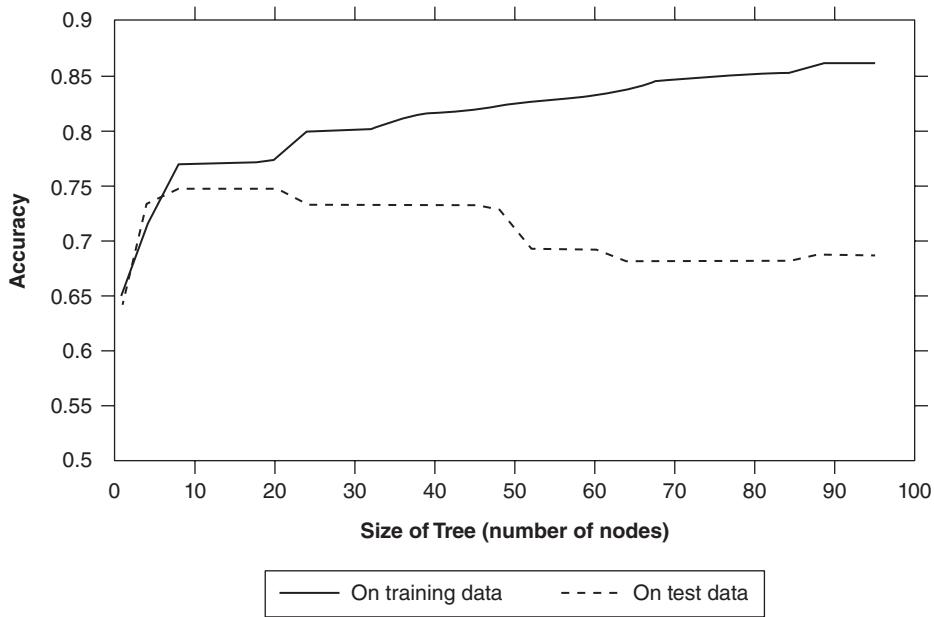
Although the first approach may seem more direct, the second approach of post-pruning overfit trees has been found to be more successful in practice. This is due to the difficulty in the first approach of estimating precisely when to stop growing the tree.

### 9.7.2 Approaches to Identify Final Tree Size

There are different approaches to decide on the final tree size. Some of them are as follows:

1. Use separate dataset for training and for evaluation. This is done by the *training and validation set approach*.
2. Use the entire dataset for training but use a statistical test (*chi square test*) for estimation.
3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This is done using the *minimum description length principle*.

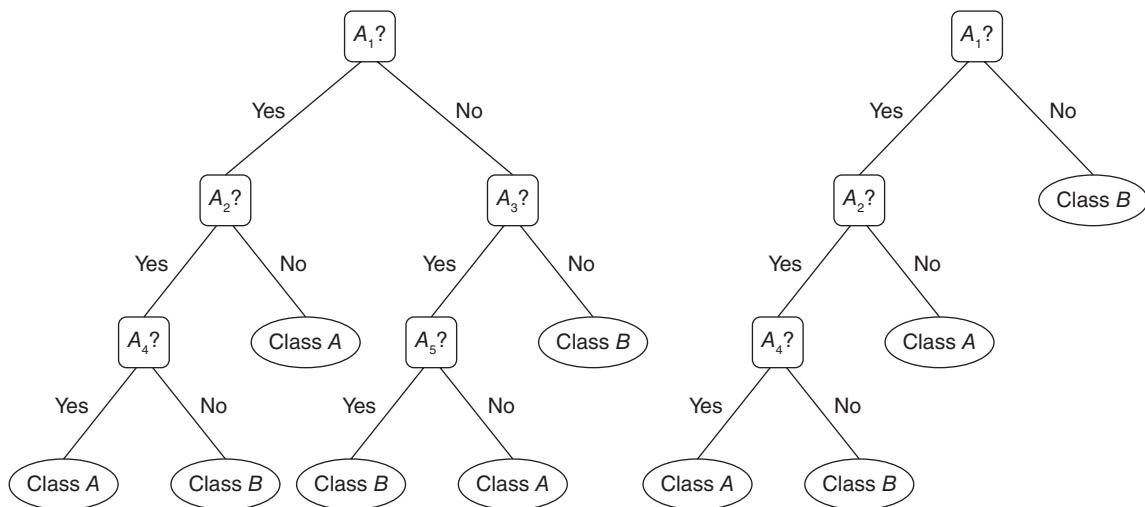
One of the most commonly used methods among the above approaches is the training and validation set approach. In this approach, the available data is split into two sets: training set and validation set. The training set is used for forming the hypothesis, while the validation set is used to evaluate the accuracy of hypothesis and the effects of pruning. The accuracy of the training set and the validation set are compared with respect to the tree size as shown in Fig. 9.4. The appropriate tree size is chosen if the accuracy of the results is acceptable (in this case, 10–20 are acceptable). Normally, two-thirds of the data is used for training and one-third for validation. This is feasible only if the validation set is large enough to provide a statistically significant set. To prevent overfitting through validation, the approaches used are *reduced error pruning* and *rule post-pruning*.



**Figure 9.4** Accuracy variation with respect to size of the decision tree.

### 9.7.2.1 Reduced Error Pruning Approach

In the reduced error pruning method, each decision node in the tree is considered a candidate for pruning. By pruning, the subtree rooted at the node is replaced by a leaf node assigning it the most common classification. Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set. An example of pruning is illustrated in Fig. 9.5.



**Figure 9.5** Decision trees before and after pruning.

Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree's accuracy over the validation set. Pruning of nodes continues until further pruning is harmful (that is, decreases the accuracy of the tree over the validation set).

#### 9.7.2.2 Rule Post-Pruning Approach

In the rule post-pruning method, the decision tree is inferred from the training set. The tree is grown until the training data is fit as well as possible, and overfitting is allowed to occur. The learned tree is converted into an equivalent set of rules by creating one rule for each path from the root node to a leaf node. Each rule is pruned (generalized) by removing any preconditions that result in improving its estimated accuracy. The pruned rules are sorted by their estimated accuracy, and considered in sequence when classifying subsequent instances.

Let us consider the following example:

IF(Outlook = Sunny)  $\wedge$  (Humidity = High) THEN Play Tennis = No

Remove each antecedent (Outlook = Sunny) and (Humidity = High) and check the improvement on the estimated rule accuracy.

In this method, decision trees are converted into rules before pruning because of the following reasons:

1. Converting to rules allows distinguishing among the different contexts in which a decision node is used. Since each distinct path produces a distinct rule, the pruning decision regarding that attribute test can be made differently for each path.
2. It removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves.
3. It improves readability since it is very easy to get output of rules.

#### 9.7.3 Incorporating Continuous-Valued Attributes

The attributes under consideration can be continuous integer values, rather than discrete instances. It should be noted that considering continuous values will result in multiple classes. For example, if temperature takes integer values between 30 and 60, there will be 31 different classes which is not appropriate.

Let us consider the example of the dataset of relation between temperature and the possibility of playing tennis, as given in Table 9.13.

**Table 9.13** Dataset of relation between temperature and playing tennis

Temperature	40	48	60	72	80	90
Play Tennis	No	No	Yes	Yes	Yes	No

The following threshold-based Boolean attribute should be defined based on temperature:

1. By sorting the examples according to the continuous attribute  $A$ , then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of  $A$ .
2. Pick up a threshold  $c$  which produces the highest information gain. In this case it appears between 48 and 60. So we take the average which is 54 and compute information gain at  $\leq 54$ ,  $< 54$ , and  $> 54$ . We similarly compute for values between 80 and 90 (that is, 85). The value that gives the highest information gain will be the splitting node.

### 9.7.4 Alternative Measures for Selecting Attributes

Higher information gain of some attributes, although good, would lead to overfitting. So the additional term split information is used. In this case, the gain ratio is computed based on split information. Gini index is also used for selecting attributes. These measures have been already discussed in Section 8.6.1.

### 9.7.5 Handling Training Examples with Missing Attribute Values

Let us consider that  $\text{Gain}(S, A)$  is to be calculated at node  $n$  in the decision tree to evaluate whether the attribute  $A$  is the best attribute to test at this decision node. Suppose that  $(x, c(x))$  is one of the training examples in  $S$  and that the value  $A(x)$  is unknown. To handle this, we assign the value that is most common among training examples at node  $n$ . We also assign a probability to each of the possible values of  $A$  rather than assigning the most common value to  $A(x)$ . For example, for a Boolean attribute  $A$  if a node has 6 cases of  $A = 1$  and 4 cases of  $A = 0$ , then  $A(x) = 1$  is 0.6 and  $A(x) = 0$  is 0.4. A fractional 0.6 of  $x$  is distributed down  $A = 1$  and 0.4 of  $x$  is distributed down  $A = 0$ .

### 9.7.6 Handling Attributes with Differing Costs

In some learning tasks the instance attributes have associated costs. In medical classification, different costs are assigned to different attributes. In such cases low-cost attributes are preferred over high-cost attributes. ID3 is modified to take into account the attribute costs by introducing a cost term into the attribute selection measure. Gain is divided by the cost of the attribute. This has higher bias towards low cost attribute. Given below are examples of two different approaches for handling attributes with differing costs.

1. Tan and Schlimmer's approach for Robot Object Recognition

$$\frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}$$

2. Approach for Nunez's approach (1988) for medical diagnosis

$$\frac{(2^{\text{Gain}(S, A)} - 1)}{(\text{Cost}(A) + 1)^W}$$

## 9.8

## Rule-Based Classification

Rule-based classifiers are ones in which the learned model is represented as a set of IF–THEN rules.

### 9.8.1 Using IF–THEN Rules for Classification

Creation of rules from the decision tree is the best way of representing information. A rule-based classifier can be constructed by using a set of IF–THEN rules. IF–THEN rule is an expression written in the following format:

IF ⟨condition⟩ THEN ⟨conclusion⟩

Let us see this statement with an example given as:

*R1: IF age >40 AND credit = fair THEN buys\_computer = yes.*

The “IF” part (or left side) of the rule is called *rule antecedent* and the “THEN” part (or right side) is known as the *rule consequent*. In rule antecedent, the condition can be of the form of one or more

attribute tests. The rule consequent is composed of prediction of the output class. The rule  $R1$  can also be written as

$$R1: (\text{age} > 40) \wedge (\text{credit} = \text{fair}) \rightarrow (\text{buys\_computer} = \text{yes})$$

If the attribute tests in a rule antecedent holds true for a given tuple, then the rule is satisfied and the rule covers the tuple.

A rule  $R1$  can be assessed with the help of two terms – coverage and accuracy. Given a tuple  $A$  from a class labeled dataset  $D$ , we can define the coverage and accuracy of  $R1$  as

$$\text{Coverage}(R1) = \frac{m_{\text{covers}}}{|D|}$$

$$\text{Accuracy}(R1) = \frac{m_{\text{correct}}}{m_{\text{covers}}}$$

where  $m_{\text{covers}}$  is the number of tuples covered by  $R1$ ,  $m_{\text{correct}}$  is the number of tuples correctly classified by  $R1$ , and  $|D|$  is the number of tuples in  $D$ .

For the tuple  $X$ , the rule  $R1$  is said to be triggered if it is satisfied by  $X$  (that is, if  $X$  satisfies  $R1$  it triggers the rule). If  $R1$  is the only rule satisfied, then the rule fires by returning the class prediction for  $X$ .

### 9.8.2 Working of Rule-Based Classifier

A rule-based classifier categorizes a tuple based on the rule triggered by it. The two important properties of the rule generated by a rule-based classifier are:

- Mutually exclusive rules:** The rules in the rule set  $R$  are said to be mutually exclusive if no two rules in  $R$  are triggered by the same record. This property ensures that each record is covered by at least one rule in  $R$ .
- Exhaustive rules:** A rule set  $R$  is said to be exhaustive coverage if there is a rule for each attribute. This property ensures that every record is covered by at least one rule in  $R$ .

Together, these properties assure that each record is covered by exactly one rule. It is not necessary for rule-based classifiers to satisfy both properties. If the rule set is not exhaustive, then it is necessary to add a default rule that covers the remaining cases. A default rule has an empty antecedent and is triggered when all other rules fail. The consequent of the default rule is called *default class*.

If the rule set is not mutually exclusive, then a tuple can be covered by more than one rule and can result in conflicting classes. There are two ways to overcome this problem.

- Ordered rules:** In this approach, the rules are ordered in decreasing order of their priority. This ordering can be based on accuracy, coverage, total description length, or the order in which the rules are generated. An ordered rule is also called *decision list*. When a tuple is presented, it is classified by the highest-ranked rule that covers the record. This avoids the problem of having conflicting class prediction.

Size ordering assigns the highest priority to the triggering rule that has the toughest requirements (toughness = rule antecedent size). The triggering rule with the most attribute tests is fired.

- Unordered rules:** This approach allows a tuple to trigger multiple rules and considers the consequent of each rule for a particular class. The consequents are then tallied to determine the class label of the tuple. The tuple is usually assigned to the class that gives the highest number of consequents. In some cases, the consequents may be weighted by the rule's accuracy. Using unordered rules has its own advantages and disadvantages. Unordered rules are less susceptible to errors caused by the wrong rule getting selected to classify a tuple. Model building is also less expensive since the order of rules are not important. The disadvantage of this method is that the classification of a tuple can become computationally expensive because the attributes of the test record must be compared against the precondition of every rule in the rule set.

### 9.8.2.1 Rule-Ordering Schemes

The rule ordering scheme does the prioritization of the rules. Ordering can be done on the basis of class or rule. In ordering based on class, the classes are sorted in the order of decreasing “importance”. That is, all the rules for the most important class come first, then those for the next important class follows, and so on. Within each class, the rules may not be in order. With rule ordering, the triggering rule that appears first in the list has the highest priority and it fires the class prediction. All other rules are ignored. Most rule-based systems use a class-based rule-ordering.

### 9.8.2.2 Rule Extraction from a Decision Tree

To extract rules from a decision tree, one rule is created for each path from the root node to the leaf node. Each splitting criterion in each path is logically ANDed to form the rule antecedent. The leaf node holds the class prediction.

Consider the decision tree shown in Fig. 9.3. The classification rules extracted from this decision tree are listed in Table 9.14.

**Table 9.14** Classification rules extracted from decision tree

R1: IF age < 30	AND student = no	THEN buys_computer = no
R2: IF age < 30	AND student = yes	THEN buys_computer = yes
R3: IF age 31–40		THEN buys_computer = yes
R4: IF age > 40	AND credit = excellent	THEN buys_computer = yes
R5: IF age > 40	AND credit = fair	THEN buys_computer = no

Since the rules are extracted from the tree, they are mutually exclusive and exhaustive.

## 9.9 Pruning the Rule Set

For a given rule antecedent, any condition that does not improve the estimated accuracy of the rule can be pruned (that is, removed), thereby generalizing the rule. After rule pruning, the rules will no longer be mutually exclusive and exhaustive. Alternatively, the rules can be extracted directly from the training data using sequential covering algorithm.

### 9.9.1 Rule Induction Using a Sequential Covering Algorithm

Rules are learned sequentially, where each rule for a given class will ideally cover many of the class tuples. A covering algorithm is an algorithm that develops a cover taking into account only the set of positive examples but none of the negative examples. This is called *sequential covering* because it learns one rule at a time and repeats this process to gradually cover the full set of positive examples. The most effective approach to learn-one-rule is *beam search*. The sequential covering algorithm is described as follows:

#### Algorithm

1. Start with an empty cover
2. Find the best hypothesis using learn-one-rule
3. If the just-learnt-rule satisfies the threshold then
  - (a) Put just-learnt-rule to the cover.
  - (b) Remove examples covered by just-learnt-rule.
  - (c) Go to step 2.

4. Sort the cover according to its performance.
5. Return: cover.

### 9.9.1.1 Learn-One-Rule Procedure

The learn-one-rule procedure is implemented using general-to-specific beam search. Beam search is a search algorithm that explores a graph by considering the most promising node first. It generates the rules from the most general category and then specializes those complexes until the desired performance is reached.

### 9.9.1.2 Evaluation Performance Function

The performance evaluation of sequential covering algorithm can be done in many ways. One of the methods for performance evaluation is by calculating the entropy. The set with highest entropy has the most information content. The entropy is computed by using Eq. (9.1).

Let us consider the example of the dataset given in Table 9.15. We first consider the “Yes” category of the dataset given in the table.

**Table 9.15** Dataset showing relation between size, shape, and whether the item is expensive

ID	Size	Color	Shape	Weight	Expensive
1	Big	Black	Square	Heavy	Yes
2	Small	Blue	Triangle	Light	Yes
3	Big	Green	Square	Heavy	No
4	Small	Black	Triangle	Light	Yes

We compute the entropy for each attribute and select the one with the highest value.

*Size*

2 Big and 2 Small

$$E(\text{Size}) = -\frac{2}{4} \log_2 \left( \frac{2}{4} \right) - \frac{2}{4} \log_2 \left( \frac{2}{4} \right) = 1$$

*Color*

2 Black, 1 Green, and 1 Blue

$$E(\text{Color}) = -\frac{2}{4} \log_2 \left( \frac{2}{4} \right) + \frac{1}{4} \log_2 \left( \frac{1}{4} \right) + \frac{1}{4} \log_2 \left( \frac{1}{4} \right) = 2$$

*Shape*

2 Square and 2 Triangle

$$E(\text{Shape}) = -\frac{2}{4} \log_2 \left( \frac{2}{4} \right) + \frac{2}{4} \log_2 \left( \frac{2}{4} \right) = 1$$

*Weight*

2 Light and 2 Heavy

$$E(\text{Weight}) = -\frac{2}{4} \log_2 \left( \frac{2}{4} \right) + \frac{2}{4} \log_2 \left( \frac{2}{4} \right) = 1$$

By selecting color first, we can define the rules. The final set of rules is as follows:

Expensive = Yes if:

1. Color = Black. (covers ID 1, 4)
2. Color = Blue and Shape = Triangle. (covers ID 2)

Decision trees build trees first and use exhaustive approaches to generate rules, while the rule-based approach works with the concept of beam search and is comparatively faster.

## Summary

- A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. It comes under the category of supervised learning algorithms.
- Supervised learning algorithm is used when the tuples take a small amount of disjoint possible values and the target output has discrete values.
- ID3 algorithm is a basic decision tree algorithm and is a top-down greedy search algorithm.
- The three popular attribute selection measures are information gain, gain ratio, and Gini index.
- The homogeneity of examples is evaluated using entropy function.
- Information gain is the measure of effectiveness of an attribute in classifying the training data. It is expected reduction in entropy.
- Decision tree classifiers are popular because they do not require any domain knowledge, can handle multidimensional data, they are fast and have good accuracy. But they do have their own set of limitations.
- C 4.5 uses gain ratio to overcome bias.
- Gini index is used in Classification and Regression Trees (CART).
- Underfitting and overfitting are the major issues in decision tree. Either the trees are not able to fit the model in underfitting or the trees are over trained with the dataset.
- Overfitting is defined as “Given a hypothesis space  $H$ , a hypothesis  $h \in H$  is said to overfit the training data if there exists some alternative hypothesis  $h' \in H$ , such that  $h$  has smaller error than  $h'$  over the training examples, but  $h'$  has a smaller error than  $h$  over the entire distribution of instances.”
- In rule-based classification, the learned model is represented in the form of rules.
- Rule is accessed by its coverage and accuracy.
- Rules are learned sequentially where each rule for a given class will ideally cover many of the class tuples by using sequential covering algorithm.

## Multiple-Choice Questions

1. A decision tree is a
  - (a) Flowchart
  - (b) Structure in which the internal node represents test on an attribute, each branch represents outcome of test, and each leaf node represents class label
  - (c) Flowchart and structure in which the internal node represents test on an attribute, each branch represents outcome of test, and each leaf node represents class label
  - (d) None of the above
2. Which of the following are advantage(s) of decision trees?
  - (a) Possible scenarios can be added
  - (b) The use of a white box model, if the given result is provided by a model
  - (c) Worst, best, and expected values can be determined for different scenarios
  - (d) All of the above

3. What does decision node illustrate in a decision tree?
- Data process description
  - Test specification
  - Class of instance
  - Data value description
4. In the figure given below,  $x_1$  and  $x_2$  are two features and data point is represented by dots ( $-1$  is negative class and  $+1$  is a positive class). The data is first split based on feature  $x_1$  (say, splitting point is  $x_{11}$ ), which is shown in the figure by the vertical line. Every value less than  $x_{11}$  will be predicted as positive class and values greater than  $x_{11}$  will be predicted as negative class.
- 
5. How many data points have been misclassified in the image above?
- 1
  - 2
  - 3
  - 4
6. Consider the example in question 4. Which of the following splitting points on feature  $x_1$  will classify the data correctly?
- Greater than  $x_{11}$
  - Less than  $x_{11}$
  - Equal to  $x_{11}$
  - None of the above
7. In which of the following scenarios would a gain ratio be preferred over information gain?
- When a categorical variable has very large number of category
  - When a categorical variable has very small number of category
  - Both (a) and (b)
  - None of the above

### Very Short Answer Questions

- What is pruning in decision trees?
- When is classification preferred over regression?
- In Gini Index, what does purity imply?
- What is the significance of entropy in ID3 algorithm?
- Name the major issues in decision tree.
- What is the goal of building a decision tree?

### Short Answer Questions

- What is the significance of information gain?
- Compare pre-pruning with post-pruning.
- How do you decide feature suitability when working on a decision tree?
- Map the relationship between entropy, information gain, and feature selection.

## Review Questions

1. What is a decision tree?
2. Explain with the help of an example the working of a decision tree.
3. What are the different attribute selection measures?
4. What are the issues with decision tree?
5. What is rule-based classifier?
6. Explain sequential covering algorithm with the help of an example.
7. Details of different chocolates in a chocolate box is given in the following table. Derive the rules for the system using sequential covering algorithm.

ID	Size	Color	Shape	Weight	Expensive
1	Big	Black	Square	Heavy	Yes
2	Small	Blue	Triangle	Light	Yes
3	Small	Blue	Square	Light	No
4	Big	Green	Triangle	Heavy	No
5	Big	Blue	Square	Light	No
6	Big	Green	Square	Heavy	Yes
7	Small	Black	Triangle	Light	Yes

## Answers

### Multiple-Choice Questions

1. (c)
2. (d)
3. (b)
4. (a)
5. (d)
6. (a)



# 10

# Support Vector Machines

## LEARNING OBJECTIVES

- To understand the basics of support vector machines.
- To understand the concept of margin and optimal hyperplane.
- To introduce the concept of vector and how to use it in computing the equation of the hyperplane.
- To introduce the concept of nonlinear separability and how radial basis function is used in solving such problems.

## LEARNING OUTCOMES

- Students will be able to compute the margin and equation of hyperplane.
- Students will be able to compute the norm.
- Students will be able to solve numericals on SVM.
- Students will be able to identify linearly and nonlinearly separable problems.
- Students will be able to compute basis functions using cover's theorem.

### 10.1 Introduction to Support Vector Machines

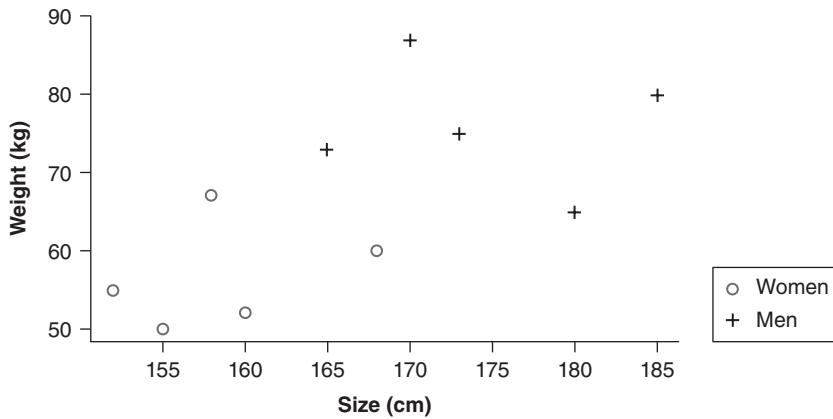
Support vector machines (SVMs) are supervised machine learning algorithms that were introduced in 1992. They became popular because of their success in handwritten digit recognition. Experimentally, it was proved that SVMs have low error rates. (1.1% test error rate for SVMs. This is the same as the error rates of a carefully constructed neural network.) SVMs are now regarded as an important example of “kernel methods”, one of the key areas in machine learning. SVMs can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems. SVM tries to map an input space into an output space using a nonlinear mapping function  $\Phi$  such that, the problem or the data points become linearly separable in the output space. When the points become linearly separable then SVM discovers the optimal separating hyperplane.

The idea behind SVMs is to make use of a (nonlinear) mapping function  $\Phi$  that transforms data in input space to data in feature space in such a way so as to render a problem linearly separable. The SVMs then automatically discover the optimal separating hyperplane (which is mapped back into input space via  $\Phi^{-1}$ ), which is nothing but a complex decision surface. To illustrate the basic ideas, we will initially begin with a linear SVM (that is, a model that assumes the data is linearly separable).

## 10.2 Linear Support Vector Machines

The goal of an SVM is to find the optimal separating hyperplane which maximizes the margin of the training data.

SVM is a supervised learning algorithm and needs training data for finding the optimal separating hyperplane. For instance, consider the training data shown in Fig. 10.1.



**Figure 10.1** Plot of height and weight of individuals.

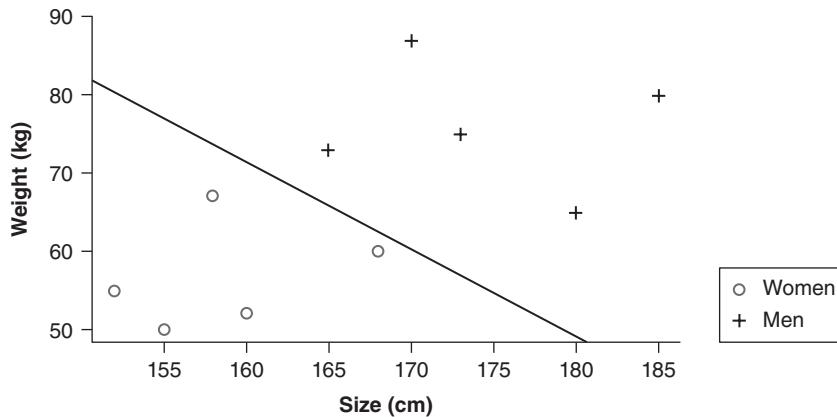
The plot in Fig. 10.1 is between the height and weight of a group of individuals. Based on the plot the person with gender as male has to be distinguished from the person whose gender is female. So the plot raises the question: Given a particular data point (weight and height), is the person male or female? When we can model the dividing line, which distinguishes between male and female we can use the model for prediction. For instance, if someone measures 175 cm and weights 80 kg, is this person a man or a woman? So the line or the hyperplane which acts as a demarcation between both the classes has to be first determined. This brings us to the next task of understanding the concept of a separating hyperplane.

### 10.2.1 Separating Hyperplane

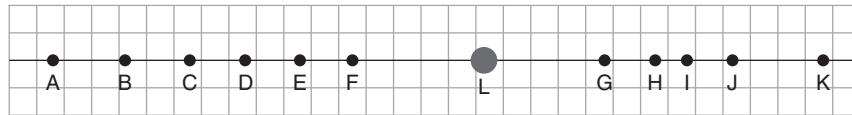
Just by looking at the plot, we can see that it is possible to separate the data. For instance, we could trace a line and then all the data points representing men will be above the line, and all the data points representing women will be below the line. Such a line is called a *separating hyperplane* and is depicted in Fig. 10.2.

For two input parameters shown in Fig. 10.2, a line acts a separator between both the classes. Even though we use a very simple example with data points lying in a plane, SVM can work with any number of dimensions.

- In one dimension, a hyperplane is called a *point*, as shown in Fig. 10.3.
- In two dimensions, it is called a *line*, as shown in Fig. 10.2.
- In three dimensions, it is called a *plane*.
- In more dimensions, we call it a *hyperplane*.



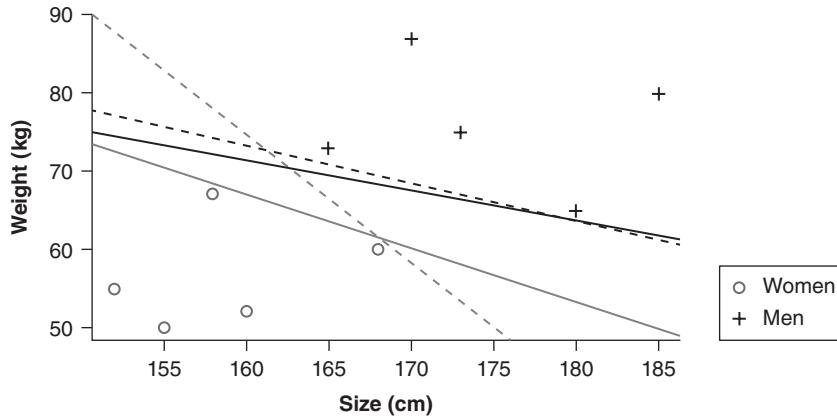
**Figure 10.2** The separating line between the two output classes.



**Figure 10.3** Point hyperplane in one dimension.

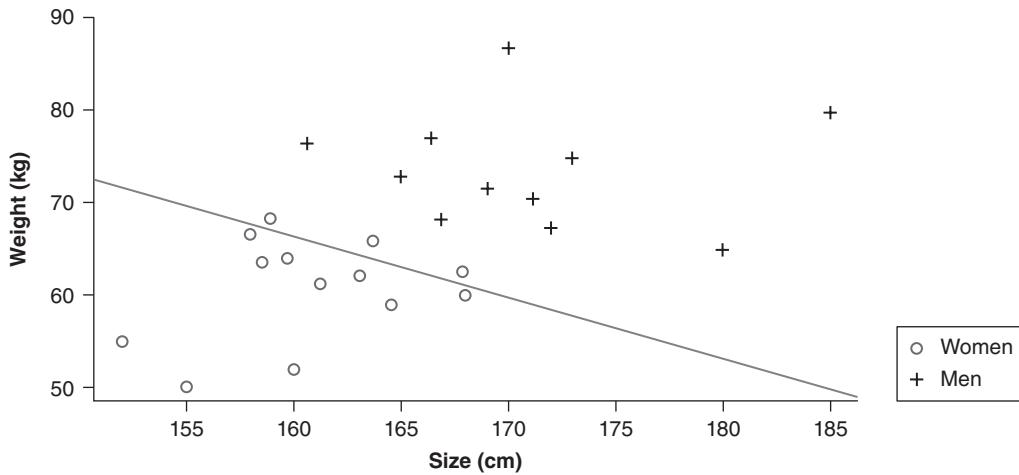
### 10.3 Optimal Hyperplane

The fact that you can find a separating hyperplane does not mean it is the best one. Figure 10.4 shows that there can be several separating hyperplanes. Each of them is valid as it successfully separates the dataset with men on one side and women on the other side.



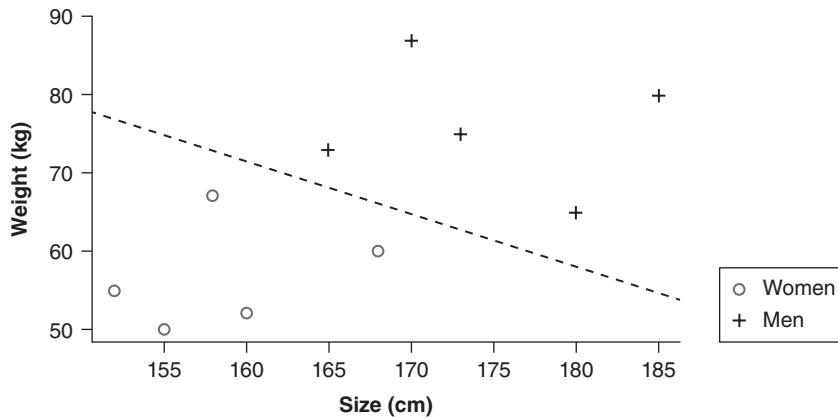
**Figure 10.4** Various hyperplanes between the two categories of outputs.

Suppose we select the first hyperplane (given as line 1 in Fig. 10.4) shown in Fig. 10.4 and use it to classify on real-life data. This is shown in Fig. 10.5.



**Figure 10.5** Line 1 hyperplane segregating male and female classes.

As shown in Fig. 10.5, this hyperplane does not generalize well. It fails to classify three women correctly. Intuitively, this can be because if a hyperplane is selected which is close to the data points of one class, then it might not generalize well. Thus, we need to select a hyperplane as far as possible from the data points of each category. Fig. 10.6 shows one such hyperplane (given as line 4 in Fig. 10.4) where the classes are distinctly classified.

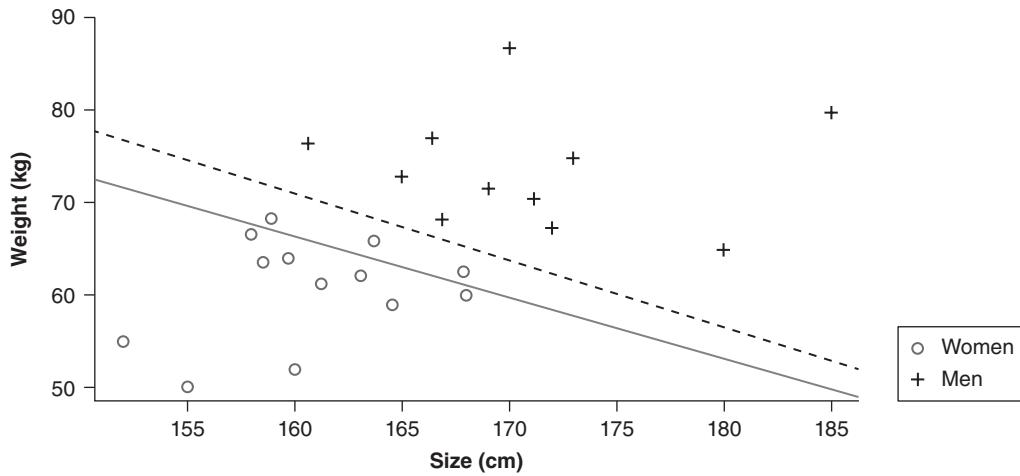


**Figure 10.6** A hyperplane for perfect classification using line 4.

The hyperplane in Fig. 10.6 using line 4 classifies the data much better. When we use it with real-life data, we can see it still makes the perfect classification.

As shown in Fig. 10.7, line 4 hyperplane classifies more accurately than the line 1 hyperplane. The objective of an SVM is to find the optimal separating hyperplane because

1. It correctly classifies the training data.
2. It generalizes better with unseen data.



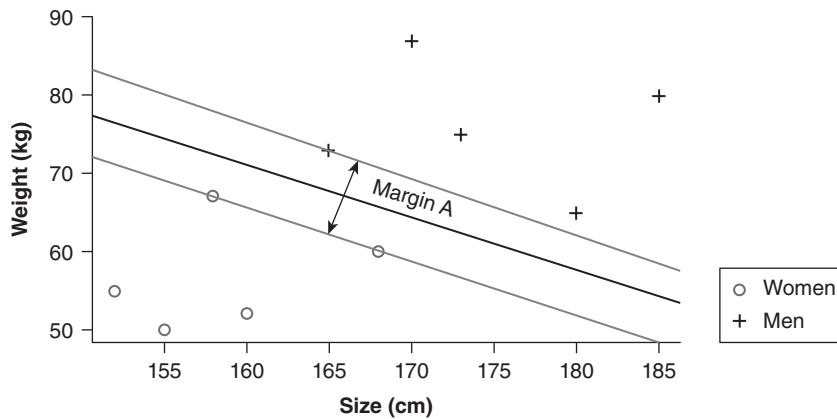
**Figure 10.7** Comparison between two hyperplanes.

### 10.3.1 Relationship between Margin and Optimal Hyperplane

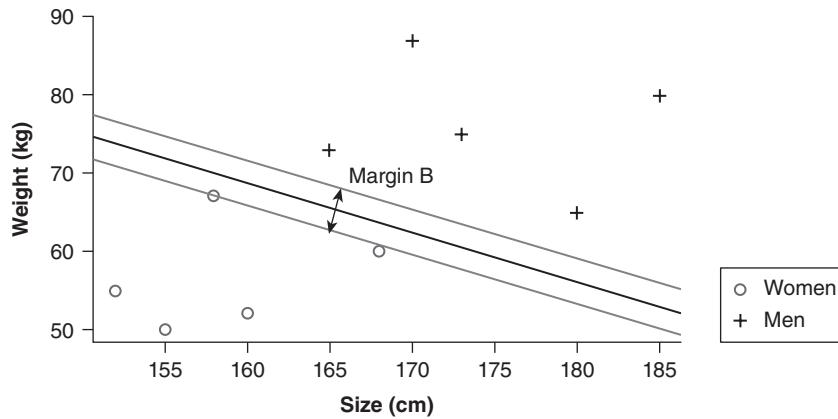
Given a particular hyperplane, we can compute the distance between the hyperplane and the closest data point. Once we have this value, if we double it we will get what is called the *margin*. The margin for the dataset for classifying male and female is shown in Fig. 10.8. Basically, the margin is a no man's land. There will never be any data point inside the margin. A margin can have varying widths. The width of the hyperplane is dependent on the location of data points. For another hyperplane, the margin will look the one shown in Fig. 10.9.

It can be visualized from Fig. 10.8 and Fig. 10.9 that Margin B is smaller than Margin A. This observation leads to the following inferences.

1. If a hyperplane is very close to a data point, its margin will be small.
2. The further a hyperplane is from a data point, the larger will be its margin.



**Figure 10.8** Margin of an optimal hyperplane.



**Figure 10.9** Hyperplane with a narrow margin.

These inferences lead to a conclusion. The optimal hyperplane will be the one with the biggest margin. That is why the objective of SVM is to find the optimal separating hyperplane which maximizes the margin of the training data.

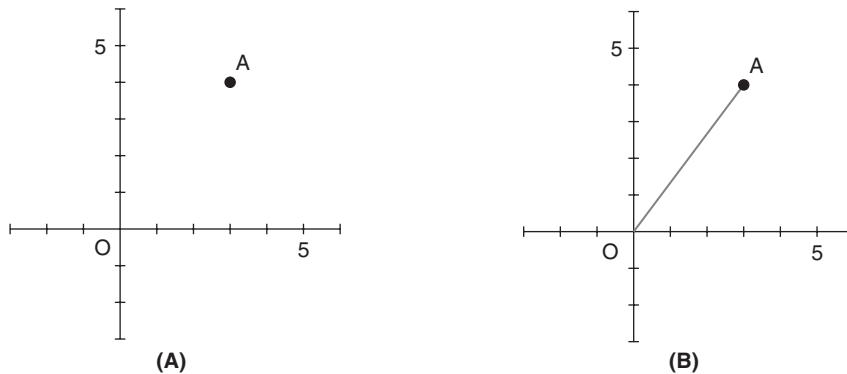
## 10.4 Basics of Vectors

### 10.4.1 What is a Vector?

A vector is an object that has both magnitude and direction. If we define a point  $A(3, 4)$  in the plane  $R^2$ , we can plot it as shown in Fig. 10.10(A).

Any point  $x = (x_1, x_2), x \neq 0$ , in  $R^2$  specifies a vector in the plane, namely the vector starting at the origin and ending at  $x$ . This means that there exists a vector  $OA$  between the origin  $O$  and point  $A$  as shown in Fig. 10.10(B).

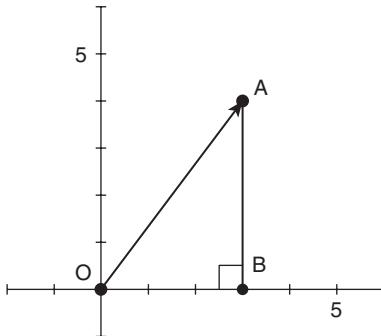
If we say that the point at the origin is the point  $O(0, 0)$  then the vector above is the vector  $\overrightarrow{OA}$ . We could also give it an arbitrary name such as  $u$ .



**Figure 10.10** (A) Point  $A$  plotted in  $R^2$  plane and (B) Vector  $OA$ .

### 10.4.2 Norm of Vector

The magnitude or length of vector  $x$  is written as  $\|x\|$  and is called its *norm*. For vector  $\overrightarrow{OA}$ ,  $\|OA\|$  is the length of the segment OA as shown in Fig. 10.11.



**Figure 10.11** Computing the length of a vector.

We can easily calculate the distance OA using Pythagoras' theorem:

$$OA^2 = OB^2 + AB^2$$

$$OA^2 = 3^2 + 4^2$$

$$OA^2 = 25$$

$$OA = \sqrt{25}$$

$$\|OA\| = OA = 5$$

The direction is the second component of a vector. The direction of a vector  $u$  ( $u_1, u_2$ ) is the vector  $w \left( \frac{u_1}{\|u\|}, \frac{u_2}{\|u\|} \right)$ . To find the direction of a vector, we need to use its angles.

The direction of vector  $u$  is defined by the angle  $\theta$  with respect to the horizontal axis and the angle  $\alpha$  with respect to the vertical axis.

The cosine of the angles is used to compute the direction. In a right triangle, the cosine of an angle  $\beta$  is defined by

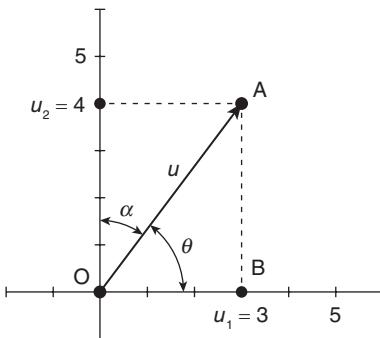
$$\cos(\beta) = \frac{\text{Adjacent}}{\text{Hypotenuse}}$$

In Fig. 10.12, there are two right triangles and in both cases the adjacent side will be on one of the axes. This means that the definition of the cosine implicitly contains the axis related to an angle.

We can also define the direction of the vector  $u$  by the cosine of angle  $\theta$  and the cosine of angle  $\alpha$  as given by Eqs. (10.1) and (10.2).

$$\cos(\theta) = \frac{u_1}{\|u\|} \quad (10.1)$$

$$\cos(\alpha) = \frac{u_2}{\|u\|} \quad (10.2)$$



**Figure 10.12** Computing the direction of vector OA.

Hence the original definition of vector  $w$ . That is why its coordinates are also called *direction cosines*.

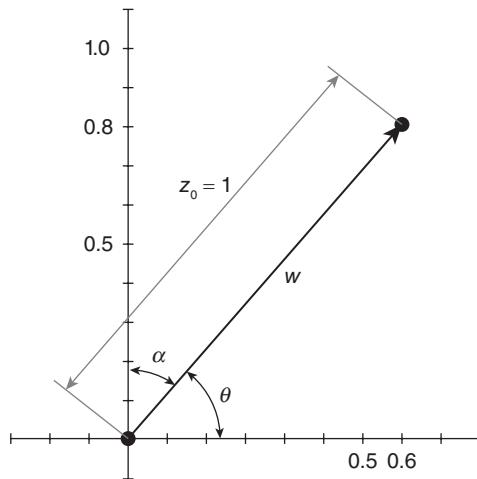
For the vector shown in Fig. 10.12 the direction vectors are computed using Eqs. (10.3) and (10.4) and their values are 0.6 and 0.8, respectively.

$$\cos(\theta) = \frac{u_1}{\|u\|} = \frac{3}{5} = 0.6 \quad (10.3)$$

$$\cos(\alpha) = \frac{u_2}{\|u\|} = \frac{4}{5} = 0.8 \quad (10.4)$$

The direction of  $u(3, 4)$  is the vector  $w(0.6, 0.8)$ . If we draw this vector we get what is shown in Fig. 10.13.

We can see that  $w$  has the same look as  $u$  except that it is smaller. Something interesting about direction vectors like  $w$  is that their norm is equal to 1. That is why they are also called *unit vectors*.



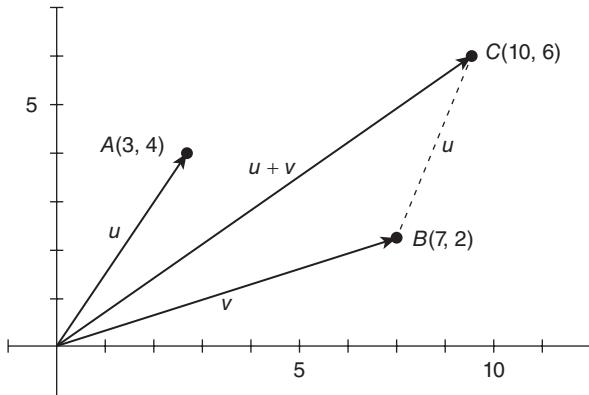
**Figure 10.13** Vector having magnitude and direction.

### 10.4.3 Sum of Two Vectors

The sum of two vectors is a third vector whose coordinates are the sums of the respective coordinates of the original vectors. Thus, the sum of two vectors  $u(u_1, u_2)$  and  $v(v_1, v_2)$  is given by

$$u + v = (u_1 + v_1, u_2 + v_2) \quad (10.5)$$

For example, the sum of two vectors  $A(3, 4)$  and  $B(7, 2)$  is  $C(10, 6)$ , as shown in Fig. 10.14.



**Figure 10.14** Sum of two vectors.

### 10.4.4 Difference between Two Vectors

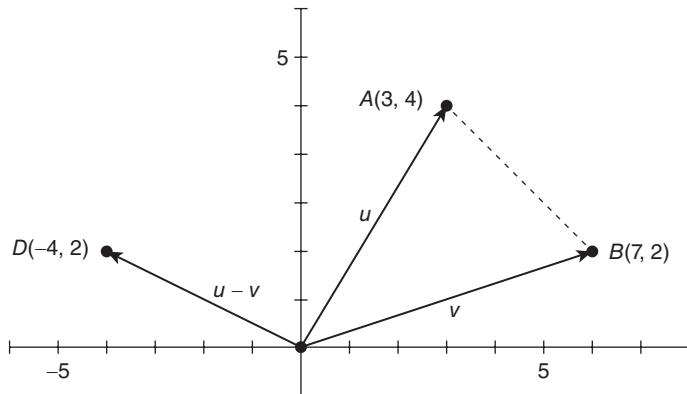
The difference between two vectors is a third vector whose coordinates are the difference between the respective coordinates of the original vectors. Thus, the difference of two vectors  $u(u_1, u_2)$  and  $v(v_1, v_2)$  is given by

$$u - v = (u_1 - v_1, u_2 - v_2)$$

Since the subtraction is not commutative, we can also consider the case

$$v - u = (v_1 - u_1, v_2 - u_2)$$

For example, the difference between two vectors  $A(3, 4)$  and  $B(7, 2)$  is  $D(-4, 2)$ , as shown in Fig. 10.15.



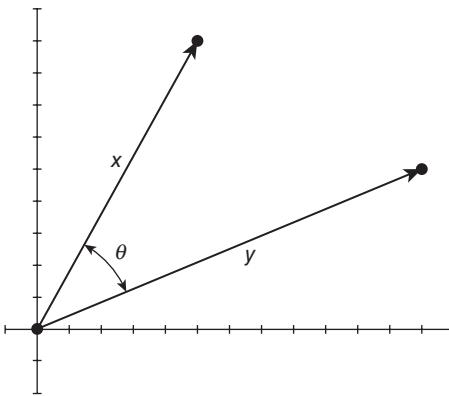
**Figure 10.15** Difference between two vectors.

### 10.4.5 Dot Product

The dot product of two vectors is the product of the Euclidian magnitudes of the two vectors and the cosine of the angle between them.

If we have two vectors  $x$  and  $y$  and there is an angle  $\theta$  between them, their dot product is given by Eq. (10.6) as shown in Fig. 10.16.

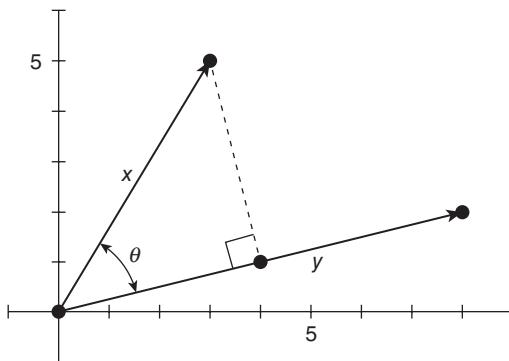
$$x \cdot y = \|x\| \|y\| \cos(\theta) \quad (10.6)$$



**Figure 10.16** Dot product of two vectors.

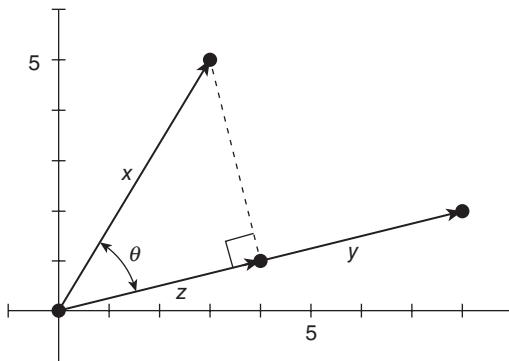
### 10.4.6 Orthogonal Projection of a Vector

The concept of orthogonal projection of a vector is better explained with an example. Given two vectors  $x$  and  $y$ , we would like to find the orthogonal projection of  $x$  onto  $y$ . To do this we project the vector  $x$  onto  $y$  as shown in Fig. 10.17.



**Figure 10.17** Projection of vector  $x$  onto vector  $y$ .

This gives us the vector  $z$  as shown in Fig. 10.18.



**Figure 10.18** Vector  $z$  as a result of projection of  $x$  onto  $y$ .

To find the orthogonal projection of a vector, we first need to calculate  $\cos(\theta)$

$$\begin{aligned}\cos(\theta) &= \frac{\|z\|}{\|x\|} \\ \|z\| &= \|x\| \cos(\theta)\end{aligned}\tag{10.7}$$

By definition of dot product

$$\cos(\theta) = \frac{x \cdot y}{\|x\| \|y\|}$$

Replacing dot product  $\|z\| = \frac{x \cdot y}{\|y\|}$  in Eq. (10.7)

$$\|z\| = \|x\| \frac{x \cdot y}{\|x\| \|y\|}$$

If we define the vector  $u$  as the direction of  $y$  then

$$u = \frac{y}{\|y\|} \text{ and } \|z\| = u \cdot x \text{ (writing } z \text{ in terms of unit vector } u \text{ with dot product of } x)$$

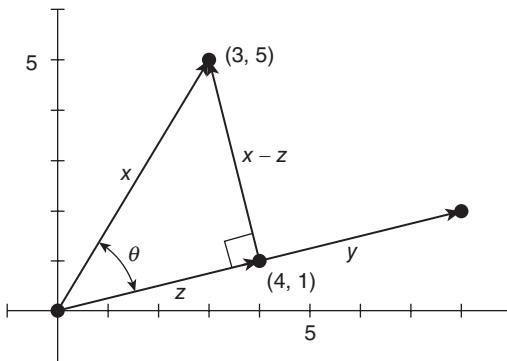
We now have a simple way to compute the norm of vector  $z$ . Since this vector is in the same direction as  $y$  it has the direction  $u$ .

$$\begin{aligned}u &= \frac{z}{\|z\|} \\ z &= \|z\| u\end{aligned}$$

The vector  $z = (u \cdot x)u$  is the orthogonal projection of  $x$  onto  $y$ . This orthogonal projection allows us to compute the distance between  $x$  and the line which goes through  $y$  as shown in Fig. 10.19.

For the points marked in the graph given in Fig. 10.19, the length of the vector is computed using the distance formula

$$\|x - z\| = \sqrt{(3 - 4)^2 + (5 - 1)^2} = \sqrt{17}$$



**Figure 10.19** Orthogonal projection of a point on vector  $x$  onto vector  $y$ .

#### 10.4.7 Equation of Hyperplane

We know that the equation of a line is:  $y = ax + b$ . However, although hyperplane is a line, its equation is

$$w^T x = 0 \quad (10.8)$$

where  $w$  and  $x$  are vectors and  $w^T x$  is the computation of the dot product of the two vectors.

Note that  $y = ax + b$  or  $y - ax - b = 0$ .

Given two vectors  $w \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$  and  $x \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$w^T x = y - ax - b \quad (10.9)$$

The equation of the hyperplane can also be written as given by Eq. (10.6). The hyperplane equation of  $w^T x$  is used in the place of  $y = ax + b$  because it is easier to work in more than two dimensions with this notation and vector  $w$  will always be normal to the hyperplane.

#### 10.4.8 Computation of Distance from a Point to the Hyperplane

Let us take the example of the hyperplane shown in Fig. 10.20. This hyperplane separates two groups of data.

To simplify this example, we have set  $w_0 = 0$ . The equation for the hyperplane shown in Fig. 10.20 is given by

$$x_2 = -2x_1 \quad (10.10)$$

which is equivalent to  $w^T x = 0$  with  $w \begin{pmatrix} 2 \\ 1 \end{pmatrix}$  and  $x \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ .

Note that the vector  $w$  is not a data point. We would like to compute the distance between the point  $A(3, 4)$  and the hyperplane. This is the distance between  $A$  and its projection onto the hyperplane as shown in Fig. 10.21 and its vector equivalent as shown in Fig. 10.22.

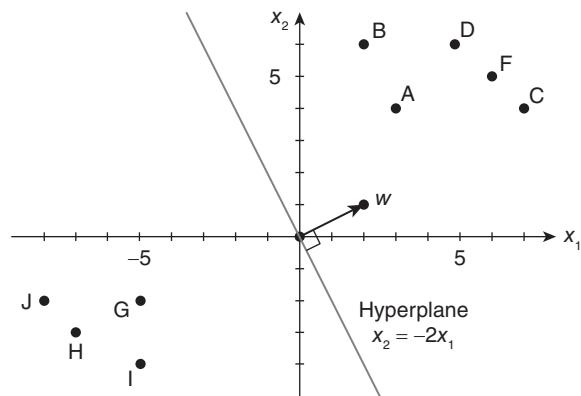


Figure 10.20 Hyperplane separating two groups of data.

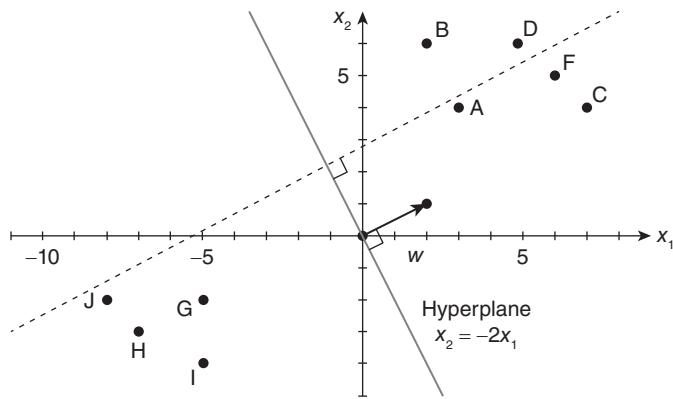


Figure 10.21 Projection of  $A$  onto the hyperplane.

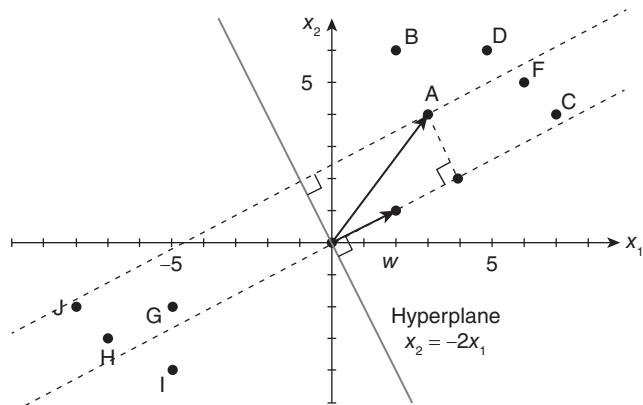
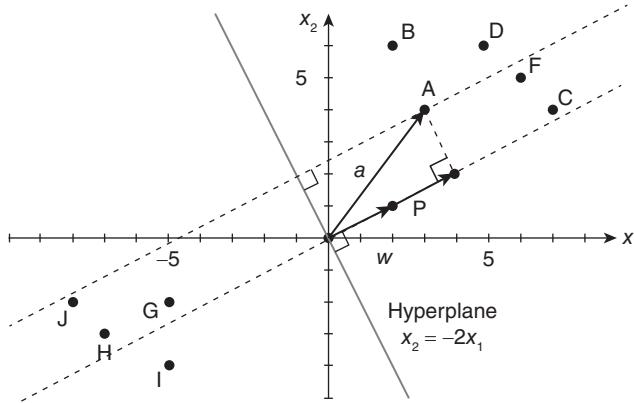


Figure 10.22 Distance between  $A$  and the hyperplane.

If we project it onto the normal vector  $w$ , we get the vector  $p$  as shown in 10.23.



**Figure 10.23** Projection of hyperplane and vector  $p$ .

Our goal is to find the distance between the point  $A(3, 4)$  and the hyperplane. We can see that this distance is the same as  $\|p\|$ . Let us compute this value. We start with two vectors,  $w = (2, 1)$  which is normal to the hyperplane, and  $A(3, 4)$  which is the vector between the origin and  $A$ .

$$\|w\| = \sqrt{2^2 + 1^2} = \sqrt{5}$$

Let the vector  $u$  be the direction of  $w$ .

$$u = \left( \frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right)$$

$p$  is the orthogonal projection of  $A$  onto  $w$ , so

$$\begin{aligned} p &= (u \cdot a)u = \left( 3 \times \frac{2}{\sqrt{5}} + 4 \times \frac{1}{\sqrt{5}} \right)u = \left( \frac{6}{\sqrt{5}} + \frac{4}{\sqrt{5}} \right)u = \left( \frac{10}{\sqrt{5}} \right)u \\ &= \left( \frac{10}{\sqrt{5}} \right) \times \left( \frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right) = \left( \frac{20}{5}, \frac{10}{5} \right) = (4, 2) \end{aligned}$$

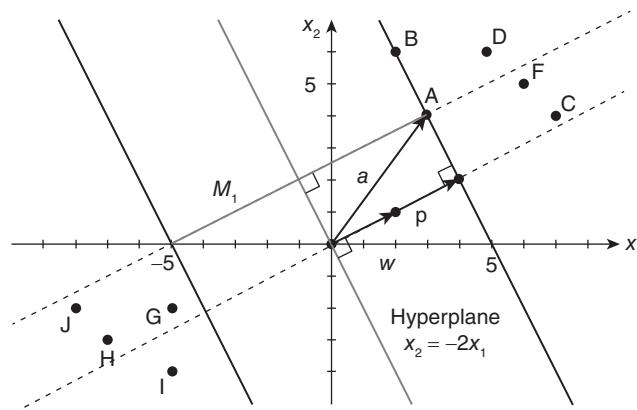
$$\|p\| = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$

#### 10.4.9 Computation of the Margin of the Hyperplane

Now that we have the distance  $\|p\|$  between  $A$  and the hyperplane, the margin is given by

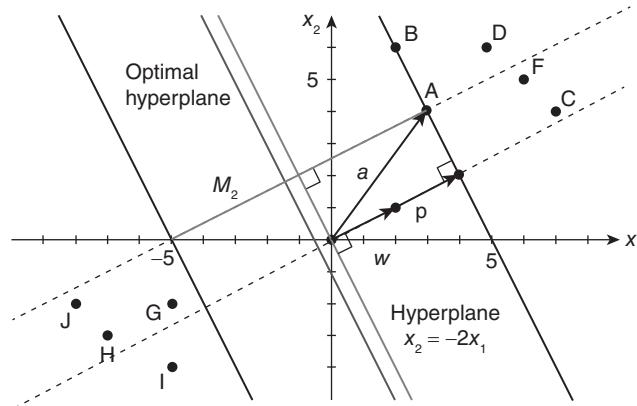
$$\text{Margin} = 2\|p\| = 4\sqrt{5}$$

Thus, we first computed the distance  $\|p\|$  between a point  $A$  and a hyperplane. We then computed the margin which was equal to  $2\|p\|$ . However, this was not the optimal hyperplane even if it did quite a good job at separating the data. The optimal hyperplane is the one which maximizes the margin of the training data.



**Figure 10.24** Margin  $M_1$  between hyperplane and data point  $A$ .

We can have different margins as given in Figs. 10.24 and 10.25. The margin  $M_1$  in Fig. 10.24 is not the biggest margin perfectly separating the data. The biggest margin is the margin  $M_2$  shown in Fig. 10.25.



**Figure 10.25** Margin  $M_2$  between hyperplane and data point  $A$ .

Finding the biggest margin is the same thing as finding the optimal hyperplane.

#### 10.4.9.1 Finding the Biggest Margin

As already mentioned, in order to find the biggest margin, we need to

1. Consider a dataset
2. Select two hyperplanes which separate the data with no points between them
3. Maximize their distance (the margin)

##### 10.4.9.1.1 Dataset for Computing Margin

Let us assume the data is composed of  $n$  vectors of  $x_i$ . Each  $x_i$  will also be associated with a value  $y_i$  indicating if the element belongs to the class (+1) or not (-1). Note that  $y_i$  can only have two possible values -1 or +1. Let us also consider that  $x_i$  is a  $p$ -dimensional vector. The dataset  $D$  is the set of  $n$  couples of element

$(x_p, y_p)$ . Finding two hyperplanes separating some data is easy when we have a pencil and paper. But with some  $p$ -dimensional data it becomes more difficult because we cannot draw it. Moreover, even if the data is only two-dimensional it may not be possible to find a separating hyperplane if the data is not linearly separable. Let us assume that the dataset  $D$  is linearly separable. The next step is to find two hyperplanes with no points between them.

#### 10.4.9.2 Hyperplanes with No Points between Them

The equation of a hyperplane can be written as given in Eq. (10.8). Any hyperplane can be written as the set of points  $x$  satisfying  $w \cdot x + b = 0$ .

Given a hyperplane  $H_0$  separating the dataset and satisfying  $w \cdot x + b = 0$ , select two other hyperplanes  $H_1$  and  $H_2$ , which also separate the data and have the following equations:

$$w \cdot x + b = \delta \quad (10.11)$$

and

$$w \cdot x + b = -\delta \quad (10.12)$$

so that  $H_0$  is equidistant from  $H_1$  and  $H_2$ .

We can set  $\delta = 1$  to simplify the problem. Hence

$$w \cdot x + b = 1$$

and

$$w \cdot x + b = -1$$

Now we want to be sure that they have no points between them. We will not select any hyperplane; we will only select those which meet the two following constraints:

For each vector  $x_i$  either:

$$w \cdot x_i + b \geq 1 \text{ for } x_i \text{ having the class 1}$$

or

$$w \cdot x_i + b \leq -1 \text{ for } x_i \text{ having the class -1}$$

By defining these constraints, we found a way to reach our initial goal of selecting two hyperplanes without points between them. Combining both these constraints, we get

$$w \cdot x_i + b \geq 1 \text{ for } x_i \text{ having the class 1}$$

or

$$w \cdot x_i + b \leq -1 \text{ for } x_i \text{ having the class -1}$$

Multiply both sides by  $y_i$  since  $y_i = -1$

$$y_i(w \cdot x_i + b) \geq y_i(-1)$$

$$y_i(w \cdot x_i + b) \geq 1 \text{ for all } 1 \leq i \leq n$$

We now have a unique constraint instead of two. They are mathematically equivalent, so their effect is the same.

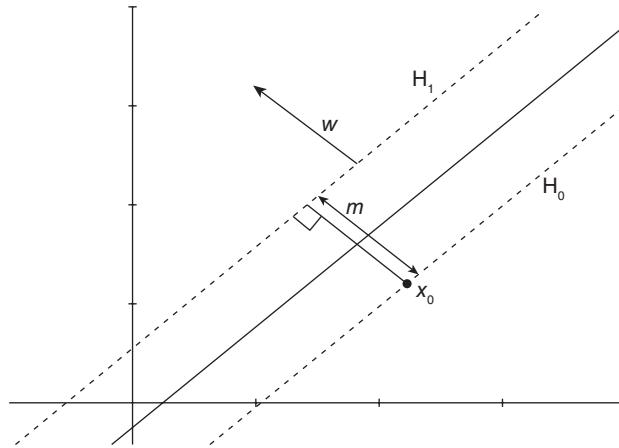
### 10.4.9.3 Maximize the Distance between the Hyperplanes

Let  $H_0$  be the hyperplane having the equation  $w \cdot x_i + b = -1$ ,  $H_1$  be the hyperplane having the equation  $w \cdot x_i + b = 1$ , and  $x_0$  be a point in the hyperplane  $H_0$ .  $m$  the margin, is the perpendicular distance from  $x_0$  to the hyperplane. As  $x_0$  is in  $H_0$ ,  $m$  is the distance between hyperplanes  $H_0$  and  $H_1$ .

Thus, we need a vector

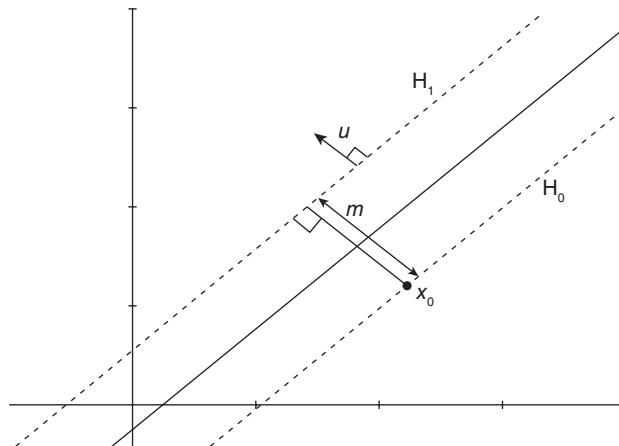
1. to have a magnitude of  $m$
2. to be perpendicular to the hyperplane  $H_1$

Fortunately, we already know a vector perpendicular to  $H_1$ , i.e.,  $w$ . (Because  $H_1 = w \cdot x_i + b = 1$ ). This is shown in Fig. 10.26.



**Figure 10.26** Vector  $w$  having magnitude  $m$  and perpendicular to  $H_1$ .

Let us define  $u = \frac{w}{\|w\|}$  as the unit vector of  $w$ . Since it is a unit vector  $\|u\| = 1$  and has the same direction as  $w$ , it is perpendicular to the hyperplane as shown in Fig. 10.27.

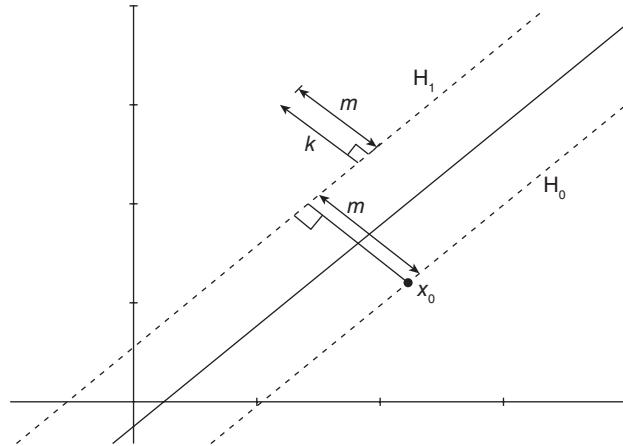


**Figure 10.27** Unit vector in the direction of  $w$ .

If we multiply  $u$  by  $m$  we get the vector  $k = mu$ , where

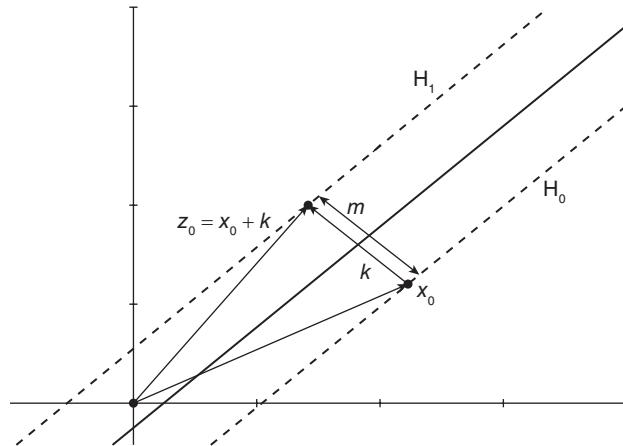
- $\|k\| = m$
- $k$  is perpendicular to  $H_1$  (because it has the same direction as  $u$ )

From these properties we can see that  $k$  is the correct vector and  $k = mu = \frac{mw}{\|w\|}$ .



**Figure 10.28** Vector of length  $k$  in the direction of unit vector.

We transformed our scalar  $m$  into vector  $k$  which we can use to perform an addition with vector  $x_0$ . This is shown in Fig. 10.28. If we start from the point  $x_0$  and add  $k$  we find that the point  $z_0 = x_0 + k$  is in the hyperplane  $H_1$ .



**Figure 10.29** Computing  $z_0$ .

The fact that  $z_0$  is in  $H_1$  means that

$$w \cdot z_0 + b = 1$$

We can replace  $z_0$  by  $(x_0 + k)$

$$w \cdot (x_0 + k) + b = 1$$

Replacing  $k$

$$w \cdot \left( x_0 + \frac{m \cdot w}{\|w\|} \right) + b = 1$$

Expanding

$$\left( wx_0 + m \frac{w \cdot w}{\|w\|} \right) + b = 1$$

$$\left( wx_0 + m \frac{\|w\|^2}{\|w\|} \right) + b = 1$$

$$(wx_0 + m\|w\|) + b = 1$$

$$w \cdot x_0 + b = 1 - m\|w\|$$

As  $x_0$  is in  $H_0$  then  $w \cdot x_0 + b = -1$

$$-1 = 1 - m\|w\|$$

$$m\|w\| = 2$$

$$m = \frac{2}{\|w\|}$$

This is well demonstrated in Fig. 10.29. To compute the margin

$$m = \frac{2}{\|w\|}$$

The only variable we can change in this formula is the norm of  $w$ . Let us try to give it different values:

When

$$\|w\| = 1 \text{ then } m = 2$$

$$\|w\| = 2 \text{ then } m = 1$$

$$\|w\| = 4 \text{ then } m = 0.5$$

It should be noted that the bigger the norm, the smaller the margin. This implies that maximizing the margin is the same thing as minimizing the norm of  $w$ .

Our goal is to maximize the margin. Among all possible hyperplanes meeting the constraints, we will choose the hyperplane with the smallest  $\|w\|$  because it will have the biggest margin.

This gives us the following optimization problem:

$$\boxed{\begin{aligned} & \text{Minimize}_{(w,b)} \\ & \|w\| \\ & \text{subject to } y_i(w \cdot x_i + b) \geq 1 \\ & (\text{for any } i = 1, \dots, n) \end{aligned}}$$

Solving this problem is like solving an equation. Once we solve it, we will have the couple  $(w, b)$  for which  $\|w\|$  is the smallest possible margin and the constraints we fixed are met which means that we will have the equation of the optimal hyperplane.

### Solved Problem 10.1

Find optimal hyperplane for the set of data points:  $\{(1, 1), (2, 1), (1, -1), (2, -1), (4.0), (5, 1), (5, -1), (6, 0)\}$  and classes:

Class 1:  $\{(1, 1), (2, 1), (1, -1), (2, -1)\}$

Class 2:  $\{(4.0), (5, 1), (5, -1), (6, 0)\}$

The plot of the points belonging to the two classes is shown in Fig. 10.30.

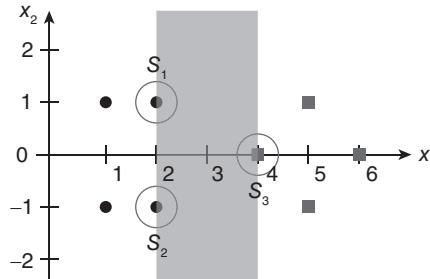


Figure 10.30 Plot of points.

### Solution:

We select three support vectors  $S_1$ ,  $S_2$ , and  $S_3$ , where

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

We use the vectors augmented with 1 as a bias input.

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

Now we need to find three parameters  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  based on the following three linear equations:

$$\alpha_1 \tilde{S}_1 \cdot \tilde{S}_1 + \alpha_2 \tilde{S}_2 \cdot \tilde{S}_1 + \alpha_3 \tilde{S}_3 \cdot \tilde{S}_1 = -1 \text{ (-ve class)}$$

$$\alpha_1 \tilde{S}_1 \cdot \tilde{S}_2 + \alpha_2 \tilde{S}_2 \cdot \tilde{S}_2 + \alpha_3 \tilde{S}_3 \cdot \tilde{S}_2 = -1 \text{ (-ve class)}$$

$$\alpha_1 \tilde{S}_1 \cdot \tilde{S}_3 + \alpha_2 \tilde{S}_2 \cdot \tilde{S}_3 + \alpha_3 \tilde{S}_3 \cdot \tilde{S}_3 = +1 \text{ (+ve class)}$$

Let us substitute the values for  $\tilde{S}_1$ ,  $\tilde{S}_2$ , and  $\tilde{S}_3$  in the above equations.

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}, \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

After simplification, we get

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

Simplifying the above three simultaneous equations, we get

$$\alpha_1 = \alpha_2 = -3.25$$

$$\alpha_3 = 3.5$$

The hyperplane that discriminates the positive class from the negative class is given by

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

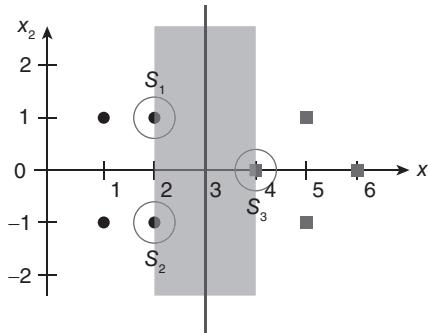
Substituting the values, we get

$$\tilde{w} = \alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\tilde{w} = (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$$

The vectors are augmented with a bias. Hence, we can equate the entry in  $\tilde{w}$  as the hyperplane with an offset  $b$ . Therefore, the separating hyperplane equation  $y = wx + b$  with  $w = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$  and offset  $b = -3$ .

This hyperplane segregating two classes as shown in Fig. 10.31.



**Figure 10.31** Hyperplane segregating the two classes.

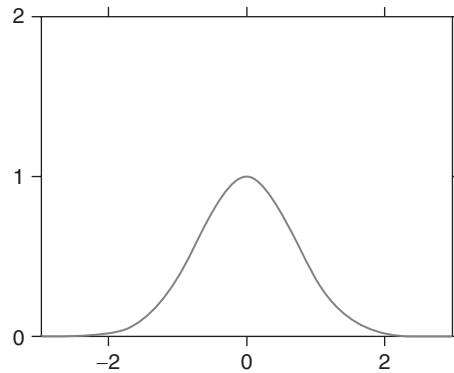
## 10.5 Radial Basis Functions

Radial basis functions (RBF) are a special class of functions for nonlinear models. The basis of RBF is based on Cover's Theorem. As per Cover's Theorem, a nonlinear problem can be linearly separable when the problem is elevated to higher dimensional space. This feature monotonically increases or decreases with distance from a central point. The center, the distance, and the shape of the radial function are parameters of the model. A typical radial function which is invariably used is the Gaussian function which in the case of a scalar input is

$$h(x) = \exp\left(-\frac{(x - c)^2}{r^2}\right) \quad (10.13)$$

where  $c$  is the center and  $r$  is the radius.

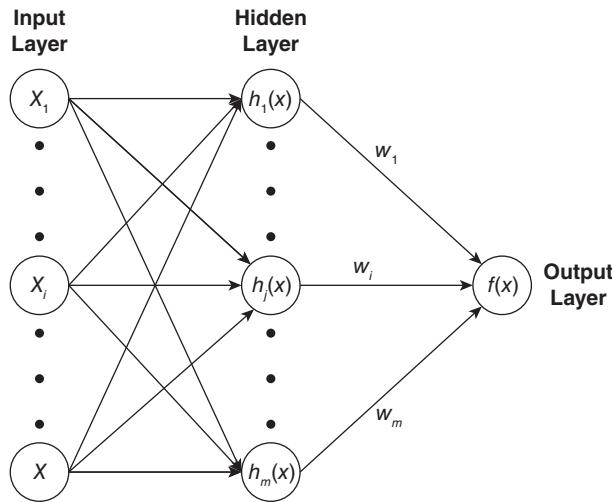
A Gaussian RBF monotonically decreases with distance from the center or in other words the function has a single peak at the center and the edges tapering monotonically on both ends (Fig. 10.32).



**Figure 10.32** Gaussian function.

### 10.5.1 Radial Basis Function Networks

This sort of function can be employed in any sort of model, which can be either linear or nonlinear. It can also be employed with a single layer or multilayered network. The architecture of RBF has three layers – an input layer, a hidden layer, and an output layer. The basic architecture of RBF is shown in Fig. 10.33.



**Figure 10.33** RBF architecture.

The focus here is on a single layer network with a single hidden layer where the nonlinear mappings of higher dimensionality space takes place.

### 10.5.2 Single Layer RBF Network

In a single perceptron, we can establish linear separability, as in the case of AND and OR functions. For AND and OR functions, the outputs are either 1 or 0 and the outputs can be linearly separable. But this is not the case in EXOR function. This can be understood diagrammatically if we look at Fig. 10.34.

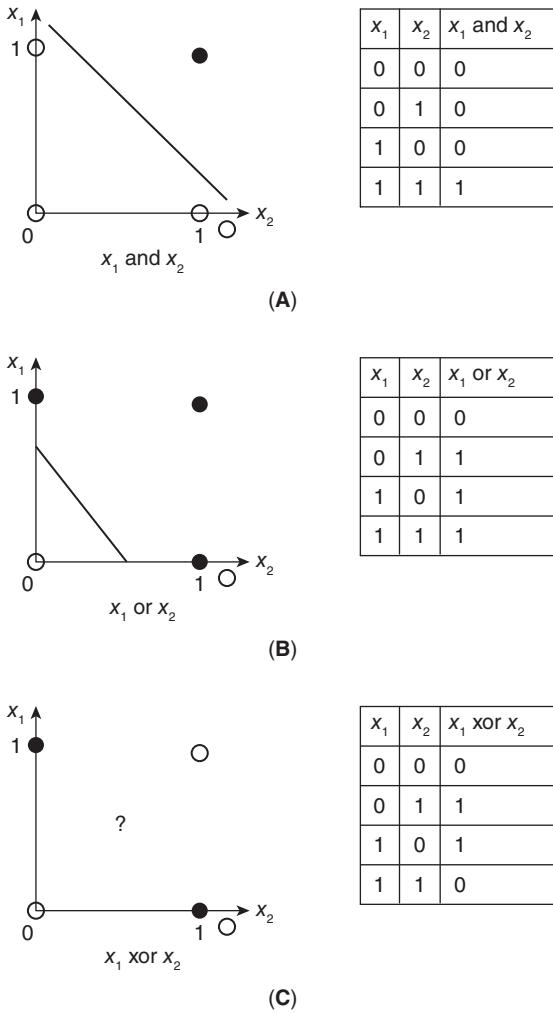


Figure 10.34 (A) AND gate (B) OR gate (C) EXOR gate.

Figures 10.28(A) and 10.28(B) show AND gate and OR gate as linearly separable models. Figure 10.28(C) shows EXOR gate as a linearly inseparable model.

The EXOR gate as seen in Fig. 10.28(C) is linearly inseparable in contrast to the AND gate and OR gate. For separating the input patterns, we need at least one hidden layer. The RBF Network (RBFN) transforms the input signal into another form and this is fed into the network to get linear separability. RBFN is structurally same as a perceptron.

### 10.5.3 Basic Architecture of the RBF Network

The basic architecture of RBFN is shown in Fig. 10.35. The RBFN is composed of input, hidden, and output layers. If an RBFN is having exactly one hidden layer, then this hidden layer is known as *feature vector*. We apply the nonlinear transfer function to the feature vector before we solve the classification problem. When we increase the dimension of the feature vector, the linear separability of the feature vector increases.

#### 10.5.4 Cover's Theorem

Cover's theorem states that "A complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space."

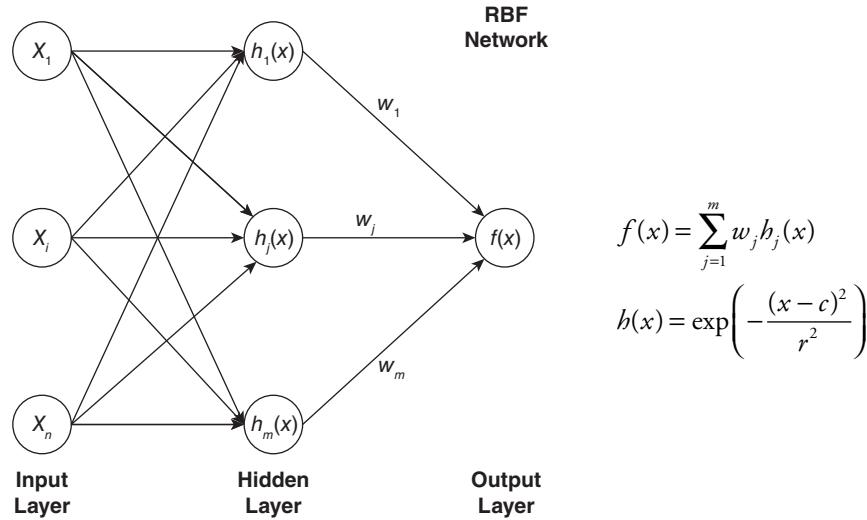


Figure 10.35 Architecture of RBF network.

Consider the RBFN architecture where  $n$  is the number of input features/values and  $m$  is the number of transformed vector dimensions (hidden layer width).

For nonlinearity separation:  $m \geq n$ . Each node in the hidden layer performs a set of nonlinear RBF. The output  $C$  remains the same as for the classification problems (certain number of class labels as predefined).

#### Solved Problem 10.2

Construct an RBF pattern classifier such that

1.  $(0, 0)$  and  $(1, 1)$  are mapped to 0, class C1
2.  $(1, 0)$  and  $(0, 1)$  are mapped to 1, class C2

See Fig. 10.36 for reference.

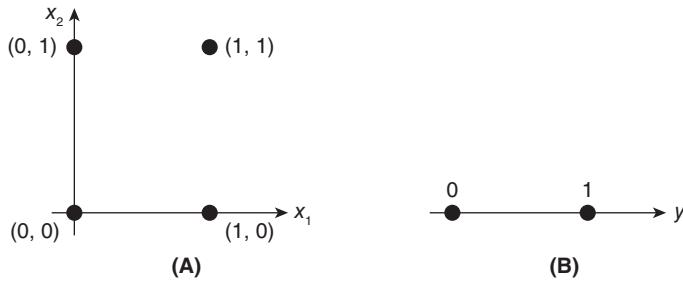


Figure 10.36 (A) Inputs of EXOR (B) Outputs of EXOR.

**Solution:**

1. Centers are selected at random  $\mu_1 = (0, 0)$  and  $\mu_2 = (1, 1)$
2. The activation function of hidden neuron is computed as follows

$$\varphi_i(\|x\|) = \exp\left(-\frac{m_1}{d_{\max}^2} \|x - \mu_i\|^2\right)$$

where  $m_1$  is the number of centers and  $d_{\max}$  is the maximum distance between two centers.

Now

$$m_1 = 2$$

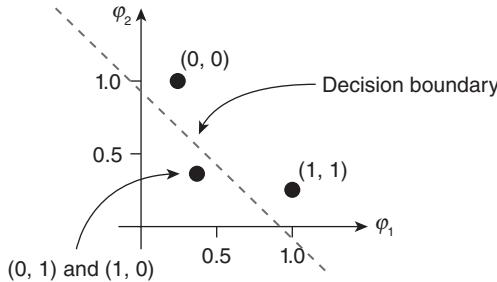
$$d_{\max} = \sqrt{(0-1)^2 + (0-1)^2} = 1.414$$

$$\varphi_1(x) = \exp(-\|x - \mu_1\|^2)$$

$$\varphi_2(x) = \exp(-\|x - \mu_2\|^2)$$

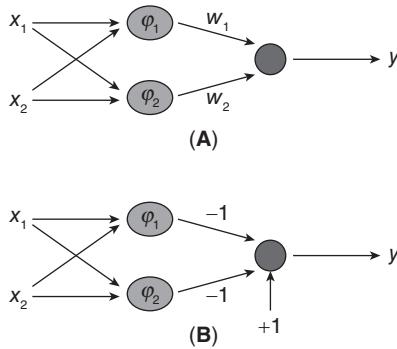
$X_1$	$X_2$	$\varphi_1(x)$	$\varphi_2(x)$
0	0	1.000	0.1353
0	1	0.3679	0.3679
1	0	0.3679	0.3679
1	1	0.1353	1.000

Plotting them, we get Fig. 10.30. The plot is reduced to a higher dimensional space and is made between  $\varphi_1$  and  $\varphi_2$  and not between  $x_1$  and  $x_2$ . As shown in Fig. 10.37, the patterns are linearly separable when plotted using higher dimensional points.



**Figure 10.37** Decision line classifying class 0 and class 1.

3. The output layer is trained using the least mean square algorithm, which is a gradient descent technique (Fig. 10.38).



**Figure 10.38** The output  $Y$  trained using gradient descent technique  
**(A)** Before training; **(B)** After training.

## Summary

- A support vector machine (SVM) is a supervised machine learning algorithm that is more commonly used in classification problems.
- The goal of an SVM is to find the optimal separating hyperplane which maximizes the margin of the training data.
- A separating plane can be a point, a line, or a plane depending on the number of dimensions. In general, the separating plane is called hyperplane.
- The objective of an SVM is to find the optimal separating hyperplane, because it correctly classifies the training data and it generalizes better with unseen data.
- Given a particular hyperplane, we can compute the distance between the hyperplane and the closest data point. Once we have this value, we can double it to get the margin.
- A vector is an object that has both magnitude and direction. The direction of a vector  $u(u_1, u_2)$  is the vector  $w\left(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}\right)$ .
- The vector  $z = (u \cdot x)u$  is the orthogonal projection of  $x$  onto  $y$ .
- The equation of the hyperplane is given by  $y_i(w \cdot x_i + b) \geq 1$  for all  $1 \leq i \leq n$ .
- Maximizing the margin is equivalent to minimizing the norm is given as an optimization problem

$$\begin{aligned} &\text{Minimize}_{(w,b)} \|w\| \\ &\text{subject to } y_i(w \cdot x_i + b) \geq 1 \\ &\quad (\text{for any } i = 1, \dots, n) \end{aligned}$$

- Radial functions are a special class of functions for non-linear models. Their characteristic feature is that their response decreases (or increases) monotonically with distance from a central point.
- RBFN is composed of input, hidden, and output layer. If a RBFN has exactly one hidden layer, then this hidden layer is known as feature vector. We apply nonlinear transfer function to the feature vector before we go for classification problem.
- A complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space.
- The hidden layers are computed using the basis functions and the output layer is trained using the least mean square algorithm, which is a gradient descent technique.

## Multiple-Choice Questions

---

1. What do you mean by generalization error in terms of SVM ?
  - (a) It is a measure of how far the hyperplane is from support vectors
  - (b) It is a measure of how accurately the SVM can predict outcomes of unseen data
  - (c) It is the amount of error
  - (d) None of the above
2. What do you mean by a hard margin?
  - (a) The SVM allows low errors in classification
  - (b) The SVM allows high amount of error in classification
  - (c) Both of the above
  - (d) None of the above
3. The minimum time complexity for training an SVM is  $O(n^2)$ . According to this fact, what sizes of datasets are not best suited for SVMs?
  - (a) Large datasets
  - (b) Small datasets
  - (c) Medium sized datasets
  - (d) Size does not matter
4. SVMs are less effective when
  - (a) The data is linearly separable
  - (b) The data is clean and ready to use
  - (c) The data is noisy and contains overlapping points
  - (d) None of the above
5. Which of the following are real world applications of SVM?
  - (a) Text and hypertext categorization
  - (b) Image classification
  - (c) Clustering of news articles
  - (d) All of the above

## Very Short Answer Questions

---

1. What do you mean by generalization error in terms of SVM?
2. What do you mean by hard margin?
3. What are support vectors?
4. What does cost parameter of SVM mean?
5. What type of classifier is SVM?

## Short Answer Questions

---

1. Explain in brief the working of SVM.
2. What are the real-world applications of SVM?
3. What is the goal of SVM?
4. Why is SVM more accurate than logistic regression?

## Review Questions

---

1. What is the goal of SVM?
2. How is the margin computed?
3. What is the role of radial basis functions in separating nonlinear patterns?
4. Using RBF how do you incorporate the concept of linear separability in EXOR function?
5. Using neat diagrams show that basic AND, OR, NAND, and NOR are linearly separable, while EXOR is not.

## Answers

---

### Multiple-Choice Questions

1. (b)    2. (a)    3. (a)    4. (c)    5. (d)



# 11

# Bayesian Classification

## LEARNING OBJECTIVES

- To understand the importance of Bayesian classifiers.
- To classify a given data using naive Bayes classifier.
- To understand the differences between naive Bayes classifier and Bayesian belief networks.
- To understand how  $k$ -nearest neighbor is different from other classification algorithms.

## LEARNING OUTCOMES

- Students will be able to solve numericals based on Bayesian classifiers.
- Students will be able to differentiate between class independency and class dependency.
- Students will be able to differentiate between Bayesian classifiers and Bayesian belief networks.
- Students will be able to use  $k$ -nearest neighbor for classification.

### 11.1 Introduction to Bayesian Classifiers

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes' theorem. A simple Bayesian classifier called *naive Bayes classifier* is comparable in performance with decision tree and selected neural network classifiers. Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases.

Naive Bayes classifiers assume that the effect of an attribute value on a given class is independent of the values of other attributes. This assumption is called *class conditional independence*. It is made to simplify the computations involved, and in this sense, is considered “naive.”

#### 11.1.1 Bayes' Theorem

Bayes' theorem defines the probability of an event based on prior knowledge of conditions related to that event. Let us consider the following example. If arthritis is related to age, then using Bayes' theorem, a person's age can be used to predict the probability of that person suffering from arthritis. We will learn about Bayes' theorem in detail with the help of an example given below.

Let  $X$  be a data tuple. In Bayesian terms,  $X$  is considered “evidence”. It is described by measurements made on a set of  $n$  attributes. Let  $H$  be some hypothesis such as that the data tuple  $X$  belongs to a specified class  $C$ . For classification problems, we want to determine  $P(H|X)$ , the probability that the hypothesis  $H$  holds given the “evidence” or observed data tuple  $X$ . In other words, we are looking for the probability that tuple  $X$  belongs to class  $C$ , given that we know the attribute description of  $X$ .

$P(H|X)$  is the posterior probability, or a posteriori probability, of  $H$  conditioned on  $X$ . For example, suppose our world of data tuples is confined to customers described by the attributes age and income,

respectively. Suppose that  $H$  is the hypothesis that our customer will buy a computer. Then  $P(H/X)$  reflects the probability that customer  $X$  will buy a computer given that we know the customer's age and income.

$P(H)$  is the prior probability, or a priori probability, of  $H$ . This is the probability that any given customer will buy a computer, regardless of age, income, or any other information.

$P(X/H)$  is the posterior probability of  $X$  conditioned on  $H$ . That is, it is probable that a customer  $X$  is 35 years old and earns ₹40,000, given that we know the customer will buy a computer.

Bayes' theorem is useful in that it provides a way of calculating the posterior probability,  $P(H/X)$  from  $P(H)$ ,  $P(X/H)$  and  $P(X)$ .

Bayes' theorem is given by

$$P\left(\frac{H}{X}\right) = \frac{P\left(\frac{X}{H}\right)P(H)}{P(X)} \quad (11.1)$$

## 11.2 Naive Bayes Classifier

The naive Bayes classifier, or simple Bayesian classifier, works as follows: Let  $D$  be a set of tuples where each tuple is an  $n$ -dimensional attribute vector given by  $x: (x_1, x_2, x_3, \dots, x_n)$ . Let there be  $m$  classes:  $C_1, C_2, C_3, \dots, C_m$ . Naive Bayes classifier predicts  $X$  belongs to class  $C_i$  if

$$P\left(\frac{C_i}{X}\right) > P\left(\frac{C_j}{X}\right) \text{ for } 1 \leq j \leq m, j \neq i$$

The Maximum Posteriori Hypothesis is given by Eq. (11.2).

$$P\left(\frac{C_i}{X}\right) = P\left(\frac{X}{C_i}\right) \frac{P(C_i)}{P(X)} \quad (11.2)$$

This reduces to Eq. (11.3)

$$\text{Maximize } P\left(\frac{X}{C_i}\right) P(C_i) \text{ as } P(X) \text{ is constant} \quad (11.3)$$

With many attributes, it is computationally expensive to evaluate  $P(X/C_i)$ . For  $n$  attributes, we follow the naive assumption of "class conditional independence", which is given by Eq. (11.4).

$$\left. \begin{aligned} P\left(\frac{X}{C_i}\right) &= \prod_{k=1}^n P\left(\frac{x_k}{C_i}\right) \\ P\left(\frac{X}{C_i}\right) &= P\left(\frac{x_1}{C_i}\right) \times P\left(\frac{x_2}{C_i}\right) \times \dots \times P\left(\frac{x_n}{C_i}\right) \end{aligned} \right\} \quad (11.4)$$

### 11.2.1 Example of Naive Bayes Classifier

We will now learn how to carry out classification from a dataset using naive Bayes classifier. Table 11.1 is a dataset which gives the class of whether a person buys a computer or not, along with attributes age, income, whether he/she is student, and his/her credit rating. We will now predict a class label using naive Bayes classification, given this sample training data.

**Table 11.1** Sample data for computing Bayesian classification

S. No.	Age	Income	Student	Credit	Buy
1	<30	High	No	Fair	No
2	<30	High	No	Excellent	No
3	31–40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31–40	Low	Yes	Excellent	Yes
8	<30	Medium	No	Fair	No
9	<30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<30	Medium	Yes	Excellent	Yes
12	31–40	Medium	No	Excellent	Yes
13	31–40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

The data tuples are described by the attributes age, income, student, and credit rating. The class label attribute “buys\_computer” has two distinct values (namely, yes and no). Let C1 correspond to the class buys\_computer = yes and C2 correspond to buys\_computer = no. The tuple we wish to classify is

$$X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$$

### Step 1:

We need to maximize  $P(X|C_i) P(C_i)$  for  $i = 1, 2$ . The prior probability of each class,  $P(C_i)$ , can be computed based on the training tuples:

$$P(\text{buys\_computer} = \text{yes}) = \frac{9}{14} = 0.643$$

$$P(\text{buys\_computer} = \text{no}) = \frac{5}{14} = 0.357$$

### Step 2:

To compute  $P(X|C_i)$  for  $i = 1, 2$ , we need to compute the following conditional probabilities:

$$P(\text{age} = \text{youth} | \text{buys\_computer} = \text{yes}) = \frac{2}{9} = 0.222$$

$$P(\text{age} = \text{youth} | \text{buys\_computer} = \text{no}) = \frac{3}{5} = 0.600$$

$$P(\text{income} = \text{medium} | \text{buys\_computer} = \text{yes}) = \frac{4}{9} = 0.444$$

$$P(\text{income} = \text{medium} | \text{buys\_computer} = \text{no}) = \frac{2}{5} = 0.400$$

$$P(\text{student} = \text{yes} | \text{buys\_computer} = \text{yes}) = \frac{6}{9} = 0.667$$

$$P(\text{student} = \text{yes} | \text{buys\_computer} = \text{no}) = \frac{1}{5} = 0.200$$

$$P(\text{credit\_rating} = \text{fair} | \text{buys\_computer} = \text{yes}) = \frac{6}{9} = 0.667$$

$$P(\text{credit\_rating} = \text{fair} | \text{buys\_computer} = \text{no}) = \frac{2}{5} = 0.400$$

**Step 3:**

Using these probabilities, we obtain

$$\begin{aligned} P(X | \text{buys\_computer} = \text{yes}) &= P(\text{age} = \text{youth} | \text{buys\_computer} = \text{yes}) \\ &\quad \times P(\text{income} = \text{medium} | \text{buys\_computer} = \text{yes}) \\ &\quad \times P(\text{student} = \text{yes} | \text{buys\_computer} = \text{yes}) \\ &\quad \times P(\text{credit\_rating} = \text{fair} | \text{buys\_computer} = \text{yes}) \\ &= 0.222 \times 0.444 \times 0.667 \times 0.667 \times 0.444 \end{aligned}$$

Similarly,

$$P(X | \text{buys\_computer} = \text{no}) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019$$

To find the class  $C_i$  that maximizes  $P(X | C_i)$ , we compute

$$P(X | \text{buys\_computer} = \text{yes}) P(\text{buys\_computer} = \text{yes}) = 0.044 \times 0.643 = 0.028$$

$$P(X | \text{buys\_computer} = \text{no}) P(\text{buys\_computer} = \text{no}) = 0.019 \times 0.357 = 0.007$$

Therefore, the naive Bayes classifier predicts  $\text{buys\_computer} = \text{yes}$  for tuple  $X$ .

**11.3****Bayesian Belief Networks**

---

Naive Bayes classifier considers class conditional independence for classification which simplifies computation. When there is class conditional independence, then the naive Bayes classifier is the best classifier to use for computation. In practical situations, dependencies can exist between the attributes. The Bayesian belief networks are the best solution for this problem. These networks consider class dependency in terms of joint conditional probability distributions. A graphical model of causal relationships is presented using learning that can be performed. After training, Bayesian belief networks can be used for classification. They are also called belief networks, Bayesian networks, or probabilistic networks.

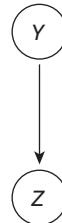
Bayesian networks can be built from human knowledge (from theory) or they can be machine-learned (from data). Also, due to their graphical structure, machine-learned Bayesian networks are visually

interpretable, thereby promoting human learning and theory building. Bayesian networks allow human learning and machine learning to work in tandem. In other words, Bayesian networks can be developed from a combination of human and artificial intelligence.

Bayesian networks facilitate causal inference. As a result, these networks are a versatile modeling framework, making them suitable for many problem domains. A belief network can be defined by two components:

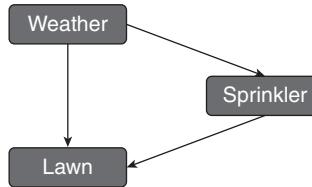
1. Directed acyclic graph.
2. Set of conditional probability tables.

Attributes are represented as nodes in the directed acyclic graph. The attributes can be discrete-valued or continuous-valued. They may correspond to actual attributes given in the data or hidden variables. The arc in the directed acyclic graph represents probabilistic dependence. If an arc is drawn from node  $Y$  to node  $Z$ , then  $Y$  is the parent of  $Z$  and  $Z$  is a descendant of  $Y$ . Each attribute is of its non-descendants in the graph, as shown in Fig. 11.1.



**Figure 11.1** Nodes in a directed acyclic graph.

Figure 11.2 represents a simple belief network for three variables. The arcs include representation of relationship between the attributes. For example, the lawn being wet or dry depends on whether the sprinkler is on or off, respectively. The directed acyclic graph shown in Fig. 11.1 has attributes as nodes. The weather attribute can take values sunny, cloudy, and rainy. The sprinkler attribute can take values on and off. The lawn attribute can take values wet and dry. The arcs represent the relationship between the attributes. Here, lawn is a child of its two parents: weather and sprinkler. The relation is that either rainy weather or turning on the sprinkler may cause the lawn to get wet.



**Figure 11.2** Belief network for lawn status.

The belief network has one conditional probability table (CPT) for each variable. The CPT for different variables are shown in Tables 11.2, 11.3, and 11.4.

### 11.3.1 Learning Bayesian Network Parameters

Given a qualitative Bayesian network structure, the CPTs are typically estimated with the maximum likelihood approach from the observed frequencies in the dataset associated with the network. From the various

**Table 11.2** The conditional probability table for attribute “weather”

Weather	
Sunny	10%
Cloudy	30%
Rainy	60%

instances in the dataset, we can train the Bayesian belief networks. Several algorithms exist for learning the network topology from the observed values in the training data. Experts generally decide on conditional dependencies that hold in the domain under analysis, which helps in network design. These experts must specify conditional probabilities for the nodes that participate in direct dependencies. These probabilities can then be used to compute the remaining probability values. From the trained data, we can infer the reason for wetness in the lawn, if it is due to rainy weather or the work of the sprinkler.

**Table 11.3** The conditional probability table for attribute “sprinkler”

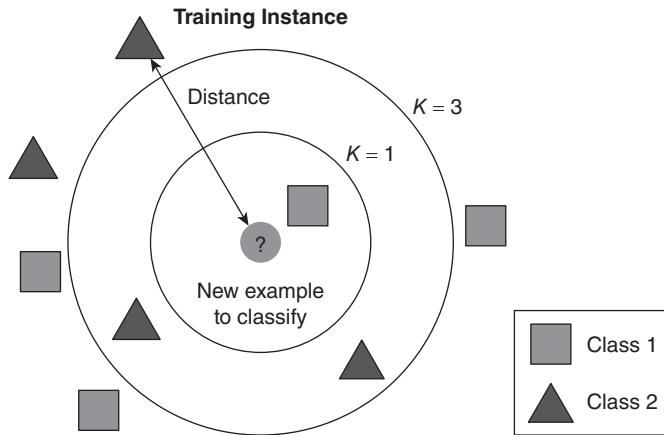
Sprinkler		
Weather	On (%)	Off (%)
Sunny	20	80
Cloudy	10	90
Rainy	0	100

**Table 11.4** The conditional probability table for attribute “lawn”

Lawn			
Weather	Sprinkler	Wet (%)	Dry (%)
Sunny	On	20	80
Cloudy	On	40	60
Rainy	On	100	0
Sunny	Off	0	100
Cloudy	Off	10	90
Rainy	Off	100	0

## 11.4 k-Nearest Neighbor (KNN)

The *k*-nearest neighbor, or KNN, algorithm is one of the simplest classification algorithms and it is one of the most used learning algorithms. This algorithm is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.



**Figure 11.3** Example of a KNN classifier.

It is considered as a non-parametric algorithm because it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. It is quite useful because in the “real world”, most of the data does not obey the typical theoretical assumptions made (for example, in linear regression, data points may not fit to the line equation derived). Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data.

It is considered a lazy algorithm because it does not use the training data points to do any generalization. There is very minimal or no explicit training phase and it is quite fast. Lack of generalization means that KNN keeps all the training data. To be more exact, all or most of the training data is needed during the testing phase.

KNN algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point. Figure 11.3 shows an example of KNN classifier. In Fig. 11.3, the test sample (central circle) should be classified either to the first class of squares or to the second class of triangles. If  $K = 3$  (solid line circle) it is assigned to the second class because there are two triangles and only one square inside the inner circle.

KNN can be used for classification in which the output is a predicted class that is discrete in nature. An object is classified based on the number of votes of its neighbors, with the object being assigned to the class most common among its nearest neighbors. It can also be used for regression output which has output in continuous domain. This value chosen is the average of the values of its nearest neighbors.

#### 11.4.1 Choosing Parameters for Nearest Neighbor

A distance measure is used to determine which of the  $K$  instances in the training dataset are most similar to a new input. For real-valued input variables, the most popular distance measure is Euclidean distance. Other popular distance measures include:

1. **Hamming distance:** It is the distance between binary vectors.
2. **Manhattan distance:** It is the distance between real vectors using the sum of their absolute difference. It is also called city block distance.
3. **Minkowski distance:** It is the generalization of Euclidean and Manhattan distance.

There are many other distance measures such as Tanimoto, Jaccard, Mahalanobis, and cosine distance. You can choose the best distance metric based on the properties of the data. If you are unsure, you can experiment with different distance metrics and different values of  $K$  together and see which mix results in

the most accurate models. Euclidean is a good distance measure to use if the input variables are similar in type (for example, all measured widths and heights). Manhattan distance is a good measure to use if the input variables are not similar in type (such as age, gender, height, etc.).

The value for  $K$  can be found by algorithm tuning. It is a good idea to try many different values for  $K$  (such as values from 1 to 21) and see what works best for the problem.

### 11.4.2 KNN Algorithm

The algorithm for KNN is given as follows:

Let  $m$  be the number of training data samples. Let  $p$  be an unknown point.

1. Store the training samples in an array of data points  $\text{arr}[]$ . This means that each element of this array represents a tuple  $(x, y)$ .
2. For  $i = 0$  to  $m$ , calculate Euclidean distance  $d(\text{arr}[i], p)$ .
3. Make set  $S$  of  $K$  smallest distances obtained. Each of these distances corresponds to an already classified data point.
4. Return the majority label among  $S$ .

### 11.4.3 Numerical Example of KNN

#### Solved Problem 11.1

Let us consider the data from a questionnaires survey and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Four training samples are listed in Table 11.5.

**Table 11.5** Sample table for classification using KNN

<i>X1 = Acid Durability (s)</i>	<i>X2 = Strength (kg/m<sup>2</sup>)</i>	<i>Y = Classification</i>
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now, the factory produces a new paper tissue that pass laboratory test with  $X1 = 3$  and  $X2 = 7$ . Without another expensive survey, can we classify this new tissue?

#### Solution:

We can classify the new tissue by computing the following five steps:

**Step 1:** Determine parameter  $K$  = number of nearest neighbors. Let us consider  $K = 3$ .

**Step 2:** Calculate the distance between the query-instance  $(3, 7)$  and all the training samples as shown in Table 11.6.

**Step 3:** Sort the distance and determine nearest neighbors based on the  $K^{\text{th}}$  minimum distance as shown in Table 11.7.

**Step 4:** Gather the category  $Y$  of the nearest neighbors. The categories are available in Table 11.5. The result is shown in Table 11.8.

**Table 11.6** Computation of square distance to (3, 7)

$X_1 = \text{Acid Durability (s)}$	$X_2 = \text{Strength (kg/m}^2\text{)}$	$\text{Square Distance to (3, 7)}$
7	7	$(7 - 3)^2 + (7 - 7)^2 = 16$
7	4	$(7 - 3)^2 + (4 - 7)^2 = 25$
3	4	$(3 - 3)^2 + (4 - 7)^2 = 9$
1	4	$(1 - 3)^2 + (4 - 7)^2 = 13$

**Table 11.7** Determination of nearest neighbors based on Kth minimum distance

$X_1 = \text{Acid Durability (s)}$	$X_2 = \text{Strength (kg/m}^2\text{)}$	$\text{Square Distance to (3, 7)}$	$\text{Rank Based on Distance}$	$\text{Included in Neighborhood?}$
7	7	16	3	Yes
7	4	25	4	No
3	4	9	1	Yes
1	4	13	2	Yes

**Table 11.8** Determination of category Y of the nearest neighbors

$X_1 = \text{Acid Durability (s)}$	$X_2 = \text{Strength (kg/m}^2\text{)}$	$\text{Square Distance to (3, 7)}$	$\text{Rank Based on Distance}$	$\text{Included in Neighborhood?}$	$\text{Category}$
7	7	16	3	Yes	Bad
7	4	25	4	No	---
3	4	9	1	Yes	Good
1	4	13	2	Yes	Good

**Step 5:** Use simple majority of the category of nearest neighbors as the prediction value of the query instance. We have 2 “good” and 1 “bad” categories. Since  $2 > 1$ , by voting the tissue can be classified as “good” category.

#### 11.4.4 Features of KNN

The  $k$ -nearest neighbor is an algorithm which stores all the dataset and classifies new cases based on a similarity measures such as Euclidean distance. It has application in statistical estimation and pattern recognition. It requires no parameters to be estimated and has been used since the 1970s. Listed below are the main features of KNN.

- It stores the entire training dataset which it uses as its representation. All training data is used to classify every single data instance. This can affect the performance of the system, especially in terms of computational time required.

2. It does not learn any model. Since there is no model, the prediction of class is instantaneous. However, the classification can take more time for larger datasets.
3. It makes just-in-time predictions by calculating the similarity between an input sample and each training instance.

#### 11.4.5 Advantages and Disadvantages of KNN

Understanding the strengths and drawbacks of KNN will help us decide whether to use this algorithm for a particular application. The following are advantages of the KNN algorithm:

1. **Insensitive to outliers:** Outliers are data elements that lie outside the boundaries of the different classes. The presence of outliers adversely affect the accuracy of classification. Since the distance from the outliers will be more, there are very less chance for an outlier to get selected in the neighborhood.
2. **No assumptions about data:** The data patterns can be linear or nonlinear in nature. Classification of linear data is easier compared to nonlinear data. The pattern of the data distribution has no effect on classification since we consider a neighborhood for classification.
3. **Simple algorithm:** The algorithm is very simple to understand and apply and so not much coding is required. Since there are no parameters associated, there is no requirement for finding the best value for the parameter. Thus, this algorithm is easy to implement.
4. **High accuracy (relatively):** The accuracy of this algorithm is comparable with other standard algorithms for classification. However, there are some better supervised learning models.
5. **Versatile:** The traditional algorithms can be used either for classification or regression. However, KNN allows us to use the same algorithm for the purpose of classification and regression.

The above listed advantages throws light on the performance of KNN algorithm. However, all algorithms do have a few minor setbacks. The following are disadvantages of the KNN algorithm:

1. **Computationally expensive:** Since the algorithm requires the evaluation of similarity measure of all training instances, the computational time associated with it is very high.
2. **High memory requirement:** Since all the training data is required for computation of the output of a single instance, this algorithm requires a large amount of memory to store this data.
3. **Prediction stage might be slow:** Since the entire dataset needs to be considered for computation, this algorithm requires a lot of time to compute similarity measures for all instances and select an output with the majority of votes.
4. **Sensitive to irrelevant features:** Irrelevant features cannot decide the outcome. Let us take the above mentioned example. The size of tissue cannot decide whether the quality of tissue is good or bad. However, using KNN, that feature is considered for classification by default.
5. **Scale of data:** Another problem with this method is the data values. For example, if thickness of the tissue is represented in cm instead of mm, the similarity distance measure will be very small in number and hence it may not have significant effect on classification. But in reality, it is one of the important features that decides the quality of the tissue paper.

### 11.5 Measuring Classifier Accuracy

---

There are different performance metrics for supervised learning algorithms, especially classification. Some of the important metrics are confusion matrix, accuracy, precision, recall or sensitivity, specificity, and F-score. Let us study each of these metrics with appropriate examples.

### 11.5.1 Confusion Matrix

The confusion matrix is one of the simplest metrics used for finding the accuracy of any classification model. Suppose we want to make a prediction whether a person will buy a computer or not based on some input parameters, we have assign two target variables: does buys\_computer is 1 and does not buys\_computer as 0. When we built the model, we set aside some tuples as testing tuples. It is to be noted that the test data has an actual output. We can predict the output based on the model.

The confusion matrix is the matrix between this actual and predicted values. It is represented in tabular form as

	POSITIVE	NEGATIVE
POSITIVE	TP	FP
NEGATIVE	FN	TN

The columns represent the actual values and the rows the predicted values. So, the matrix can be filled by four values given by

1. **True Positive (TP):** The actual and predicted values are true.
2. **False Positive (FP):** The actual values are false but the predicted values are true. For example, actual values of buys\_computer is “NO” but predicted value is “YES”.
3. **True Negative (TN):** The actual and predicted values are false.
4. **False Negative (FN):** False negatives are the cases when the actual class of the data point was true and the predicted is false. For example, actual values of buys\_computer is “YES” but predicted value is “NO”.

### 11.5.2 Accuracy

Accuracy of prediction is how correctly the classifier functions. That is, it should correctly classify True as True and False as False. So, the accuracy is given by the ratio of

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FN + FP + TN)} \quad (11.5)$$

It is evident from Eq. (11.5) that accuracy nears 100% when FN and FP are lesser. Accuracy is a good measure when the target variable classes in the data are nearly balanced. It would also be a good measure if the classifier must identify the output class as orange or apple based on shape and color. Moreover, accuracy would be an appropriate measure if the classifier is used to predict breast cancer as malignant or benign, where majority of samples belong to benign class. On the other hand, precision should be used for unbalanced class of outputs.

### 11.5.3 Precision

Precision is a measure that tells us what proportion are TP against all positive samples. As an example, precision tells us the number of patients that we diagnosed as having cancer, actually had cancer. From confusion matrix the precision is computed as

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

Let us consider for example, if only 5 patients actually had cancer out of 55 who were suspected cancer patients. Here, the precision is given by

$$\text{Precision} = \frac{5}{(5+55)} \times 100 = 8.33\%$$

#### **11.5.4 Recall or Sensitivity**

Recall, or sensitivity, is a measure that tells us what proportion of patients that actually had cancer were diagnosed by the algorithm as having cancer. Recall is given by

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

#### **11.5.5 Specificity**

Specificity is a measure that tells us what proportion of output classes which is false were predicted to be false. In the case of classifying the cancer in breast cancer patients as malignant and benign, specificity is the measure of patients who did not have cancer (benign) were predicted by the model as non-cancerous (benign). Specificity is given by

$$\text{Specificity} = \frac{\text{TN}}{(\text{TN} + \text{FP})}$$

#### **11.5.6 F-Score**

Instead of having different metrics like precision and recall, F-score combines them into a single metric. The F-score is given by

$$\text{F-score} = \frac{(2 \times \text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$$

It is to be noted that F-score uses the harmonic mean instead of arithmetic mean.

## Summary

---

- Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class.
- Naive Bayes classifiers assume that the effect of an attribute value on a given class is independent of the values of other attributes. This assumption is called *class conditional independence*.
- Bayes' theorem defines probability of an event, based on prior knowledge of conditions related to that event.
- Bayesian belief networks consider class dependency in terms of joint conditional probability distributions. A graphical model of causal relationships is presented using learning that can be performed.
- Given a qualitative Bayesian network structure, the conditional probability tables are typically estimated with the maximum likelihood approach from the observed frequencies in the dataset associated with the network.
- KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.
- A distance measure is used to determine which of the K instances in the training dataset are

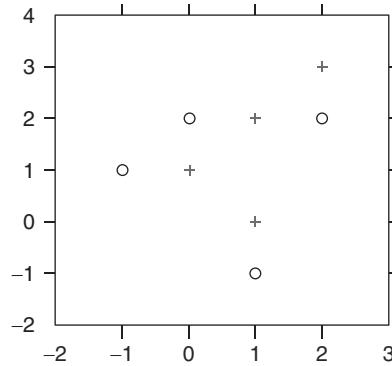
most similar to a new input. For real-valued input variables, the most popular distance measure is Euclidean distance.

- $k$ -nearest neighbor is insensitive to outliers. It is simple, versatile, and has high accuracy. But it

is computationally expensive and requires high memory.

- Classifier accuracy can be determined using confusion matrix, accuracy, recall or sensitivity, specificity, and F-score.

## Multiple-Choice Questions



## **Very Short Answer Questions**

1. Why is naive Bayes classifier called naive?
  2. List some advantages and disadvantages of naive Bayes classifier.
  3. Mention when might  $k$ -nearest neighbors fail.
  4. How does one measure the effectiveness of KNN?
  5. With four tosses of a fair coin, what is the probability to get exactly heads-tails-tails-heads, in this order?

## Short Answer Questions

1. With four tosses of a fair coin, what is the probability to get each heads and tails twice, regardless of the order?
2. How is KNN different from  $k$ -means clustering?
3. Given loaded coin tosses are independent events and that a different loaded coin has the following probability for coming up heads twice in a row:  $P(\text{Loaded Coin} = \text{heads}, \text{Loaded Coin} = \text{heads}) = 0.16$ . What is the probability that the loaded coin will come up tails twice in a row? In other words, what is  $P(\text{Loaded Coin} = \text{tails}, \text{Loaded Coin} = \text{tails})$ ?
4. Use KNN in the given training set to classify an unknown case (Age = 48 years and Loan = ₹1,42,000)
5. For a unknown tuple  $t = \langle \text{No}, \text{Business}, \text{High} \rangle$ , use naive Bayes classifier to find whether the class for defaulted is yes or no. The dataset is given below.

<i>ID</i>	<i>Homeowner Status</i>	<i>Income</i>	<i>Defaulted</i>
1	Yes	Employed	High
2	No	Business	Average
3	No	Employed	Low
4	Yes	Business	High
5	No	Unemployed	Average
6	No	Business	Low
7	Yes	Unemployed	High
8	No	Employed	Average
9	No	Business	Low
10	No	Employed	Average

<i>Age</i>	<i>Loan (₹)</i>	<i>Default</i>
25	40,000	N
35	60,000	N
45	80,000	N
20	20,000	N
35	1,20,000	N
52	18,000	N
23	95,000	Y
40	62,000	Y
60	1,00,000	Y
48	2,20,000	Y
33	1,50,000	Y

## Review Questions

- What is Bayes' theorem? How does a naive Bayes classifier help in classifying the output class?
- Based on the table given below classify Drew, a blue-eyed person with long hair and over 170 cm, as “male” or “female”.

Name	Over 170 cm	Eye	Hair Length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

- How is the Bayesian belief network different from naive Bayes classifier?
- How is classification accomplished using the  $k$ -nearest neighbor algorithm?
- Use KNN to predict the t-shirt size of a new customer given that he/she weighs 61 kg and is 161 cm tall. All details necessary for computation are listed in the table below.

Height (cm)	Weight (kg)	T-Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

## Answers

### Multiple-Choice Questions

1. (b)    2. (d)    3. (c)    4. (a)    5. (a)



# 12

# Hidden Markov Model

## LEARNING OBJECTIVES

- To introduce the category of recognition problems using temporal patterns.
- To understand the issues in HMM.
- To introduce the basics of Hidden Markov Model (HMM).

## LEARNING OUTCOMES

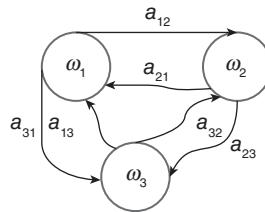
- Students will understand and appreciate the use of HMM as a classifier.
- Students will get an insight on speech recognition problems.
- Students will understand the issues in HMM.

### 12.1 Introduction to Hidden Markov Model

Speech recognition problems come under the category of recognition problems with temporal patterns. We can usually see some persons with some mannerisms. As an example, some may use lots of hand gestures while talking. Some frequently repeat some words like “Got it”, “Catch my point”. So in a temporal sequence, that in a speech signal if some patterns are frequently repeated its called as temporal pattern. Speech signals are time varying signals (i.e., at different instances of time we have different signal strength). For speech recognition purposes, the speech is divided into a number of phonemes. The word spoken can be identified based on the sequence in which the phonemes occur.

Activity recognition is another application of temporal pattern recognition. These activities can be recognized by sign language recognition using hand movements. Based on the sequence of hand gestures, we can interpret the message conveyed. An application area of activity recognition is security surveillance where abnormal movement or activity can be monitored. Movement in security surveillance is nothing but body postures at different instances of time. So if the movement is found to be abnormal, it implies that it would be from a suspicious person.

To recognize or identify such temporal sequence we need a machine which is similar to a sequential machine or finite state machine. In other words, a machine takes us through a finite set of output symbols from a finite set of input symbols. Thus, we have a finite set of states through which the machine makes a transition, i.e., at any time of time the machine moves from one state to another, it takes an input and emits an output.



**Figure 12.1** State transitions of HMM.

The machine makes a transition from one input state to an output state. If the set of outputs are limited to only two symbols, 0 and 1, the sequence machine becomes a finite state automaton and has huge application in sequence generation and sequence detection. Moreover, it is a useful concept in communication. So if a finite state automaton detects the beat sequence, the next portion of the sequence can also be detected.

Hidden Markov Model (HMM) is a statistical model that takes statistical property of signal into account. There are two types of states in HMM:

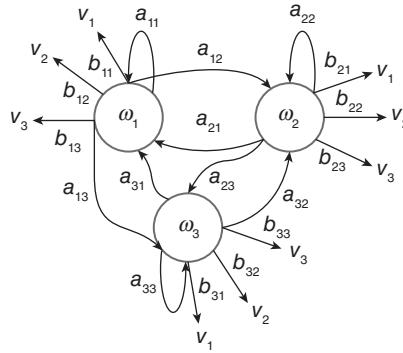
1. Visible state ( $V$ ).
2. Hidden state ( $\omega$ ).

HMM can have a number of hidden states and visible states. As an example, let us consider an HMM model ( $\theta$ ) having three hidden states ( $\omega$ ) and three visible states ( $V$ ). This is given by  $\omega = \{\omega_1, \omega_2, \omega_3\}$  and  $V = \{V_1, V_2, V_3\}$ .

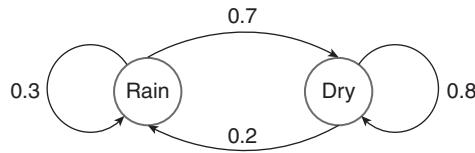
Figure 12.1 shows the state transitions of the HMM from state  $\omega_{t-1}$  at time  $t - 1$  to state  $t$  with a probability  $a_{ij}$ .

In each state, the machine can emit one of the visible states. The probability of emission from a visible state is given by  $P(v_k|\omega_j) = b_{jk}$ , that is the probability of  $v_k$  given  $\omega_j$ . So from  $\omega_1$  the machine emits  $v_2$  with a probability  $v_{12}$  and  $v_3$  with a probability  $v_{13}$ . These are the probabilities of emission of visible states from different hidden states of HMM. The above process is called *observable Markov model*, since the output of the process is the set of states at each instant of time, where each state corresponds to an observable event. Thus, we can see that given any state there will always be transition to some of the states including the original state itself.

Redrawing Fig. 12.1 with the visible states, hidden states, emission probabilities, and transition probabilities, we get Fig. 12.2.



**Figure 12.2** State transition diagrams showing emission and transition probabilities.



**Figure 12.3** Markov chain for Rain and Dry.

It can also be seen that at any point of time

$$\sum a_{ij} = 1, \quad \forall i \quad \text{and} \quad \sum b_{jk} = 1, \quad \forall j$$

Let us consider the example of two hidden states Rain and Dry with the initial probabilities as shown in Fig. 12.3.

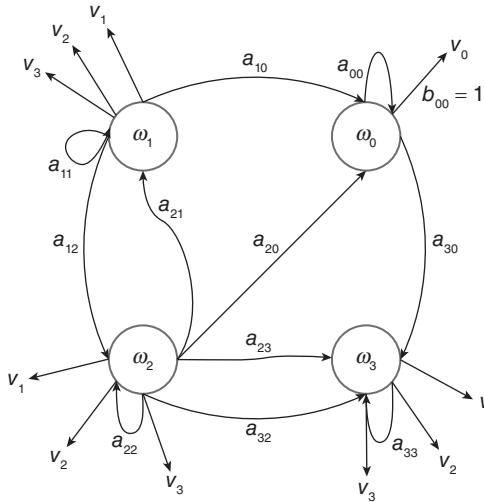
Figure 12.3 shows a simple example of Markov chain with two hidden states Rain and Dry. The different state transition probabilities when the state changes from state Rain to Rain, Rain to Dry, Dry to Dry, Dry to Rain are given below, along with the initial probability of Rain and Dry.

$$\begin{aligned} P(\text{Rain}|\text{Rain}) &= 0.3 \\ P(\text{Dry}|\text{Dry}) &= 0.8 \\ P(\text{Dry}|\text{Rain}) &= 0.7 \\ P(\text{Rain}|\text{Dry}) &= 0.2 \\ P(\text{Rain}) &= 0.4 \\ P(\text{Dry}) &= 0.6 \end{aligned}$$

It is seen that the sum of all transition probabilities is equal to 1 and the sum of emission probabilities is also equal to 1, as given in the calculations below.

$$\begin{aligned} P(\text{Rain}|\text{Rain}) + P(\text{Dry}|\text{Rain}) &= 0.3 + 0.7 = 1 \\ P(\text{Dry}|\text{Dry}) + P(\text{Rain}|\text{Dry}) &= 0.2 + 0.8 = 1 \end{aligned}$$

Finally, HMM has a specific hidden state called *receiving state* or *accepting state*. Once the machine reaches that state it cannot come out. Only one visible state can be emitted from this hidden state. Incorporating this concept in Fig. 12.2 and redrawing it we get Fig. 12.4. It is seen that  $v_0$  is the final visible state that is emitted from the hidden state. There are no transitions to other hidden states from this state.



**Figure 12.4** Markov model with visible and hidden states.

This HMM model is used for temporal pattern recognition. As discussed in Section 12.1, a temporal pattern is a frequently recurring pattern in a sequence of signals. Thus, HMM is used in the identification of such patterns in a sequence.

## 12.2 Issues in Hidden Markov Model

There are three central issues that need to be addressed in HMM:

1. Evaluation Problem.
2. Decoding Problem.
3. Learning Problem.

If an HMM  $\theta$  has number of hidden states  $w$ , number of visible states  $v$ , transmission probability from one hidden state to another  $a_{ij}$ , emission probability from visible state  $b_{jk}$ , then the probability that a particular visible state is emitted is given by  $P(V^T/\theta)$  where technically

- $\theta = \theta_1, \theta_2, \dots, \theta_c$
- $\theta \Rightarrow w, v, a_{ij}, b_{jk}$
- $V^T \Rightarrow$  sequence of visible symbols

where  $\theta$  is the Markov model having hidden states, visible states, transition probabilities, and emission probabilities.

In other words, the evaluation problem is given the observation sequence  $V^T$ . Now, the question is: What is the probability that this sequence is generated by any of the models  $\theta = \theta_1, \theta_2, \dots, \theta_c$ ? We can think of this as a scoring problem. If you have to choose between many competing models, then one with maximum probability will give better results. In other words, the model with maximum probability will be the best model for the sequence of visible states. Thus, finding this probability  $P(V^T/\theta)$  is the evaluation problem. So when there are ' $c$ ' number of model sequences for every sequence, there exists an HMM  $\theta$ , where  $\theta_1$  is the first HMM,  $\theta_2$  is the second HMM,  $\theta_n$  is the  $n^{\text{th}}$  sequence HMM. The probability has to be computed given the unknown sequence  $P(V^T/\theta^n)$ , where it can be produced by  $\theta_1, \theta_2, \dots, \theta_n$ . Once this probability is known, the visible sequence  $V^T$  can be classified provided we know the HMM. So if there are ' $c$ ' number of sequences, the unknown sequence has to be classified into one of these sequences. Every sequence has an HMM  $\theta$ . So if there are ' $c$ ' sequences, then HMM ranges from  $\theta_1$  to  $\theta_c$ .

Every  $\theta$  in the sequence of  $\theta_1, \theta_2, \dots, \theta_c$  will have its own  $a_{ij}, b_{jk}, w, v$ . The evaluation problem is: Given a  $V^T$ , what is the probability that this state is produced by any of  $\theta_1, \theta_2, \dots, \theta_c$ ?

The decoding problem is what is the most likely  $\omega(t)$  of the hidden state which has led to the generation of  $V^T$ , where  $\omega^T$  is the generated  $V^T$ . Problem 2 is one which attempts to uncover the hidden part of the problem. There is no correct solution to this problem, in practice we usually use optimal criterion to find best possible solution.

The learning problem is estimating  $a_{ij}$  and  $b_{jk}$  knowing  $w$  and  $V$ . Here, we try to optimize model parameters to describe how a given observation sequence comes out. The observation sequence used here is called *training sequence* since it is used for training HMM. Training is one of the crucial elements of HMM. It allows us to adjust model parameters as to create the best model for a given training sequence. This learning problem is important not only in HMM, but in any classifier problem. Now let us see each of these issues in depth.

Let us take a simple example of a coin toss problem to explain HMM. Assume the following scenario. You are isolated in a room, and you cannot see what is going on outside. Your friend outside tosses a coin and tells you the result of each coin flip. Thus, a sequence of heads and tails would be performed by your

friend and you know the sequence. A typical sequence would have (H, T, T, H, T, H, ..., H), where H stands for heads and T stands for tails. To model this coin tossing event with HMM, two questions are raised.

1. What state is the status of the model?
2. How many states should there be in the model?

This depends on the number of coins being tossed. If it is only a single coin then we have either heads or tails. But if the number of coins is more than one, then we get a sequence from the different coins that are tossed. In other words, we get a matrix with rows corresponding to the states and columns corresponding to the coins that are tossed. The person inside the isolated room is not aware about which coin is tossed. He can only hear the sequence of heads and tails. This problem corresponding to these coins would be similar to the one depicted in Fig. 12.2.

### 12.2.1 Evaluation Problem

The evaluation problem is to find the probability of  $V^T$  given  $\theta$  when  $\theta$  and  $V^T$  are known. For this all the possible sequences of hidden state of length  $T$  have to be found out. This is given by

$$P(V^T | \theta) = \sum_{r=1}^{r_{\max}} P\left(\frac{V^T}{\omega_r^T}\right) P(\omega_r^T) \quad (12.1)$$

where

- $r$  is one of the sequences
- $\omega^T \rightarrow$  sequence of  $T$  number of those possible sequences of visible states
- $r_{\max}$  is the number of such sequences.
- $W_r^T = \{\omega(1), \omega(2) \dots \omega(T)\}$

If this is on  $N$  number of hidden states then

$$r_{\max} = N^T$$

This means that there are  $N^T$  number of possible sequences of hidden states of length  $T$ .

$N \rightarrow$  # of hidden states

$P(\omega_r^T)$  is the probability of the sequence from state at time  $t$  to a state at  $t - 1$ . It is a sequence of  $T$  number of such states. It is given by

$$P(\omega_r^T) = P(\omega(t) | \omega(t-1)) \quad (12.2)$$

The product of all such transition probabilities for consecutive time sequences is given by

$$P(\omega_r^T) = \prod_{t=1}^T P(\omega(t) | \omega(t-1)) \quad (12.3)$$

Let us consider the example of the problem given in Fig. 12.3. Suppose we want to calculate the probability of a sequence of states {Dry, Dry, Rain, Rain}.

$$\begin{aligned} P\{\text{Dry, Dry, Rain, Rain}\} &= P(\text{Rain} | \text{Rain}) \cdot P(\text{Rain} | \text{Dry}) \cdot P(\text{Dry} | \text{Dry}) \cdot P(\text{Dry}) \\ &= 0.3 \times 0.2 \times 0.8 \times 0.6 = 0.0288 \end{aligned}$$

The probability of finding a visible state given a hidden state is

$$P(V^T | \omega_r^T) = \prod_{t=1}^T P(v(t) | \omega(t)) \quad (12.4)$$

Substituting Eqs. (12.3) and (12.4) in Eq. (12.5), we get

$$P(V^T | \theta) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^T P(v(t) | \omega(t)) \cdot P\left(\frac{\omega(t)}{\omega(t-1)}\right) \quad (12.5)$$

Complexity of this expression is  $O(N^T T)$ . This is because there are  $N^T$  state paths, each costing  $O(T)$  calculations leading to  $O(N^T T)$  time complexity. Thus, instead of using a complex expression for computing a visible state, a recursive algorithm can be used. The recursive algorithm is: Given a set of  $V^T$ , what would be the probability that HMM would be at a particular state at a particular time?

At  $t = 0$  the HMM is at  $\omega_1$ . Then at  $t = 1$  the probability of whether the machine would be in  $\omega = 0, 1, 2, \dots$  can be determined. The reason is that from  $\omega_1$  it is possible to make a transition to  $\omega_1, \omega_0, \omega_2, \dots, \omega_r$ .

If  $V(1) = V(0)$  then the hidden state has been emitted by the previous state  $\omega_1$ . However, if  $V(1) \notin V(0)$  then it has not been emitted from the previous state. This can be written in the form of probability of emission of visible symbol at time step  $t$ .

At the time instant  $t$ , if the machine is in a particular state, then the probability of transition from  $(t-1)$  state to  $t$  state can be computed. The corresponding probabilities are given by  $\alpha_i(t-1), \alpha_2(t-2) - \alpha_i(t-1)$ ; the transition probability  $a_{ij}$  is also given.

The probability that the machine is in state  $\alpha$  at  $(t-1)$  instant is given by

$$\alpha_j(t) = \begin{cases} 0, & t = 0 \text{ and } j \neq \text{initial state} \\ 1, & t = 0 \text{ and } j = \text{initial state} \\ \left[ \sum_{i=1}^N \alpha_i(t-1) a_{ij} \right] \cdot b_{jk} V(t) & \dots \text{otherwise} \end{cases}$$

The recursive algorithm can be either forward algorithm or backward algorithm. The forward algorithm is given by

Initialize:  $t \leftarrow 0, a_{ij}, b_{jk}, V^T, \alpha_j(0)$  for  $t \leftarrow t + 1$

$$\alpha_j(t) = b_{jk}(i) \cdot \sum_{i=1}^N \alpha_i(t-1) a_{ij}$$

Until  $t = T$

Return  $P\left(\frac{V^T}{\theta}\right) \leftarrow \alpha_0(T)$  for final state end.

### 12.2.2 Decoding Problem

The decoding problem finds out the most likely sequence or most probable sequence of hidden states through which the machine has transmitted while generating  $V^T$ . In other words, the decoding problem finds the most probable state. A trellis diagram is used for this purpose. A trellis diagram is made up of a matrix of nodes. The column of the nodes represents all the possible status at a certain time. The first

column represents all the possible status at the time instant 0. Similarly, the second and third columns represent the status at different time intervals. Thus, it would be easy to interpret and view the state changes at different time intervals using a trellis diagram. This diagram is used in convolution networks along with state diagram. This seems an easy way of understanding the flow to signals during every state change. Since HMM has a different number of hidden states, the trellis diagram is used in this case especially to find out the value of each most probable sequence of hidden states.

The algorithm for decoding problem is

Given  $V^T$  find the most probably sequence of hidden states  $\Rightarrow$  Decoding problem.

Also

Initialize: Path  $\leftarrow \{ \}$   $t \leftarrow 0, j \leftarrow 0$

for  $t \leftarrow t + 1$

$j \leftarrow j + 1$

for  $j \leftarrow j + 1$

$$\alpha_j(t) \leftarrow b_{jk} v(t) \cdot \sum_{i=1}^n \alpha_i(t-1) a_{ij}$$

until  $j = N$  (where  $N$  = number of in-between states)

$$j' \leftarrow \arg \max \alpha_j(t)$$

Appending  $\omega_{j'}$  to path

until  $t = T$  (length of sequence)

Return Path

### 12.2.3 Learning Problem

In the learning process of HMM, we have to estimate the transmission probability ( $a_{ij}$ ) and emission probability ( $b_{jk}$ ). For learning, we use a number of known sequences. For example, if there exists a string of visible states  $\langle V_1, V_3, V_1, V_5, V_7, V_2, V_0 \rangle$ , then

$\alpha_3(4)$  means the probability that the machine is in state  $\omega_3$  and generates sequence  $V_1, V_3, V_1, V_5$ .

$\beta_3(4)$  means the probability that the machine is in state  $\omega_3$  and generates the next three visible states.

In general,  $\beta_i(t)$  is the probability that the model will be in  $\omega_i(t)$  and will generate the remainder of the given target sequence  $V^T$ . All symbols from  $V(t+1)$  to  $V(T)$  will be generated. The backward algorithm helps in computing  $\beta_i(t)$ .

$$\beta_i(t) = \begin{cases} 0, & \omega_i(t) \neq \omega_0 \text{ and } t = T \\ 1, & \omega_i(t) = \omega_0 \text{ and } t = T \\ \sum_j \beta_j(t+1) a_{ij} \cdot b_{jkv}(t+1) & \dots \text{otherwise} \end{cases}$$

The HMM backward is

Initialize  $\beta_i(T); t \leftarrow T$   $a_{ij}, b_{jk}, V^T$   
for  $t \leftarrow t - 1$

$$\beta_i(t) = \sum_j \beta_j(t+1) a_{ij} b_{jkv}(t)$$

until  $t = 1$

Return  $P(V^T) \leftarrow \beta_i(0)$  for the known initial state  
end.

For estimation of model parameters  $a_{ij}$  and  $b_{jk}$ , both forward and backward algorithms have to be used simultaneously.

#### 12.2.4 Learning Problem (Classifier)

The final goal is to classify the sequence of symbols. Bayes rule is used for this, which has already been discussed in detail in Chapter 9.

$P(x|\omega_i)$  is the class or conditional probability of  $x$  given the hidden state. The task is to classify  $x$  in one of the classes. For this classification we need to compute  $P(\omega_i|x)$  and  $i$  for which  $P(\omega_i|x)$  is maximum. The unknown sample would be classified in that particular class.

$P(\omega_i|x) \rightarrow$  posterior probability

$P(x|\omega_i) \rightarrow$  prior probability

This is given by

$$P(\omega_i|x) = \frac{P(x|\omega_i) \cdot P(\omega_i)}{P(x)} \quad (12.6)$$

If there are two classes, we compute

$$P(\omega_j|x) > P(\omega_i|x)$$

Then the unknown sample is classified under that class.

In HMM, we find  $P(V^T|\theta)$  in the evaluation step. In the classification step, we need to compute  $P(\theta|V^T)$ . Applying Bayes rule as shown in Eq. (12.7), we find the probability of whether the given sequence belongs to a particular class given the visible state.

$$P(\theta|V^T) = \frac{P(V^T|\theta) \cdot P(\theta)}{P(V^T)} \quad (12.7)$$

If there are two models then we have  $\theta_1$  and  $\theta_2$ . We then compute  $P(\theta_1|V^T)$  and  $P(\theta_2|V^T)$ . If  $P(\theta_i|V^T) > P(\theta_j|V^T)$  then  $V^T$  can be classified in  $\theta_i$ .

The learning process of HMM is supervised learning because we try to learn with sequences of visible states.

## Summary

- HMM is used in problems which have temporal patterns.
- Speech signals are time-baring signals and are made of phonemes. The word spoken can be identified based on the sequence in which the phonemes occur.
- To recognize or identify such temporal sequence we need a machine, which is similar to a sequential machine or finite state machine.
- HMM has two types of states: a visible state ( $V$ ) and a hidden state ( $\omega$ ).
- $a_{ij}$  is the probability with which HMM moves from one state to another.
- The probability of emission from a visible state is given by  $P(v_k|\omega_j)$ .
- Issues in HMM are evaluation problem, decoding problem, and learning problem.
- The evaluation problem is to find the probability  $P(V^T|\theta)$ .
- The decoding problem is what is the most likely  $\omega(t)$  of the hidden state which has led to the generation of  $V^T$ , where  $\omega^T$  is the generated  $V^T$ .
- The learning problem is estimating  $a_{ij}$  and  $b_{jk}$  having known  $w$  and  $V$ .
- The final goal is to classify the sequence of symbols. Bayes rule is used for this purpose.

## Multiple-Choice Questions

---

1. Which algorithm is used for solving temporal probabilistic reasoning?
  - (a) Hill-climbing
  - (b) Hidden Markov model
  - (c) Depth-first search
  - (d) Breadth-first search
2. The state of the process in HMM is described by which of the following.
  - (a) Random variable
  - (b) Literal
  - (c) Single discrete random variable
  - (d) None of the above
3. Where is HMM used?
  - (a) Speech recognition
  - (b) Computational Biology
  - (c) Both (a) and (b)
  - (d) None of the above
4. What does training of HMM mean?
  - (a) Probability of the states
  - (b) Probability of states given the model
  - (c) Estimation of the emission and transition probabilities
  - (d) None of the above

## Short Answer Questions

---

1. What are the components of Markov decision process (MDP)?
2. What is the difference between Markov Model and Hidden Markov Model?
3. What are some of the common applications of Hidden Markov Model?

## Review Questions

---

1. What is HMM?
2. What types of learning can be accomplished using HMM. How can this be used?
3. Briefly explain the issues in HMM.
4. How is HMM used as the classifier after learning.
5. What is the significance of trellis diagram in solving the decoding problem?

## Answers

---

### Multiple-Choice Questions

1. (b)    2. (c)    3. (c)    4. (c)





## PART 3

# Unsupervised Algorithms

---

### Chapter 13

Introduction to Unsupervised Learning Algorithms



# 13

# Introduction to Unsupervised Learning Algorithms

## LEARNING OBJECTIVES

- To understand the basics of clustering.
- To understand and appreciate the requirements of clustering.
- To introduce the concept of partitioning-based clustering ( $k$ -means and  $k$ -medoids).
- To learn how to create dendograms using agglomerative-based clustering

## LEARNING OUTCOMES

- Students will be able to understand and appreciate clustering as an unsupervised learning method.
- Students will be able to solve numericals on partitioning-based clustering techniques.
- Students will be able to solve numericals on agglomerative-based clustering using single, complete, and average linkages.

### 13.1 Introduction to Clustering

Clustering is the process of grouping together data objects into multiple sets or clusters, so that objects within a cluster have high similarity as compared to objects outside of it. The similarity is assessed based on the attributes that describe the objects. Similarity is measured by distance metrics. The partitioning of clusters is not done by humans. It is done with the help of algorithms. These algorithms allow us to derive some useful information from the data which was previously unknown. Clustering is also called *data segmentation* because it partitions large datasets into groups according to their similarity.

Clustering can also be used for outlier detection. Outliers are objects which do not fall into any cluster because of too much dissimilarity with other objects. We can utilize them for special applications like credit card fraud detection. In credit card transactions, very expensive and infrequent purchases may be signs of fraudulent cases and we can apply one more level of security to avoid such transactions.

Clustering is known as unsupervised learning because the class label information is not present. You have already seen in supervised learning algorithm that every input has a corresponding output, which helps in designing a model. That is why supervised learning is called learning by example, while unsupervised learning is called learning by observation.

### 13.1.1 Applications of Clustering

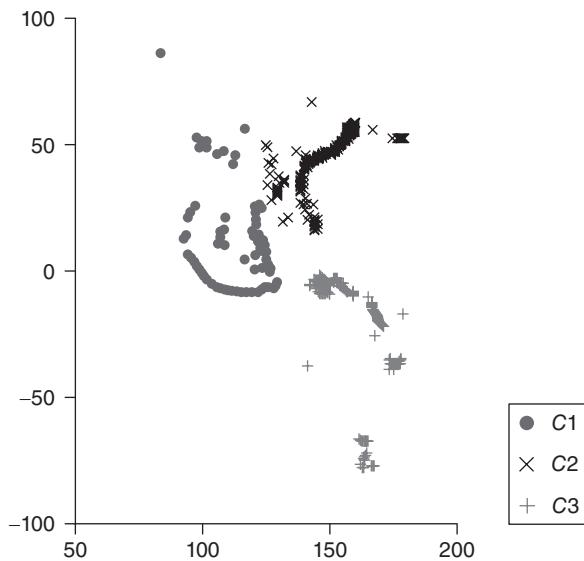
Cluster analysis has been widely used in many applications such as business intelligence, pattern recognition, image processing, bioinformatics, web technology, search engines, and text mining.

1. **Business intelligence:** Cluster analysis helps in target marketing, where marketers discover groups and categorize them based on the purchasing patterns. The information retrieved can be used in market segmentation, product positioning (i.e., allocating products to specific areas), new product development, grouping of shopping items, and selecting test markets.
2. **Pattern recognition:** Here, the clustering methods group similar patterns into clusters whose members are more similar to each other. In other words, the similarity of members within a cluster is much higher when compared to the similarity of members outside of it. There is no prior knowledge of patterns of clusters or even how many clusters are appropriate.
3. **Image processing:** Extracting and understanding information from images is very important in image processing. The images are initially segmented and the different objects of interest in them are then identified. This involves division of an image into areas of similar attributes. It is one of the important and most challenging tasks in image processing where clustering can be applied. Image processing has applications in many areas such as analysis of remotely sensed images, traffic system monitoring, and fingerprint recognition.
4. **Bioinformatics:** This is a growing field in terms of research activities and is a part of biotechnology and genetic engineering. In this case, clustering techniques are required to derive plant and animal taxonomies, categorize genes with similar functionalities, and gain insight into structures inherent to populations. Biological systematics is another field which involves study of the diversification of living forms and the relationships among living things through time. The scientific classification of species can be done on the basis of similar characteristics using clustering. This field can give more information about both extinct and extant organisms.
5. **Web technology:** Clustering helps classifying documents on the web for information delivery.
6. **Search engines:** The success of Google as a search engine is because of its intensive searching capabilities. Whenever a query is fired by a user, the search engine provides the result for the searched data according to the nearest similar object which are clustered around the data to be searched. The speed and accuracy of the retrieved resultant is dependent on the use of the clustering algorithm. Better the clustering algorithm used, better are the chances of getting the required result first. Hence the definition of similar object plays a crucial role in getting better search results.
7. **Text mining:** Text mining involves the process of extracting high quality information from text. High quality in text mining means clustering in terms of relevance, novelty, and interestingness. It can be used for sentiment analysis and document summarization.

### 13.1.2 Requirements of Clustering

The requirements of clustering can be enumerated and explained as follows:

1. **Scalability:** A clustering algorithm is considered to be highly scalable if it gives similar results independent of the size of the database. Generally, clustering on a sample dataset may give different results compared to a larger dataset. Poor scalability of clustering algorithms leads to distributed clustering for partitioning large datasets. Some algorithms cluster large-scale datasets without considering the entire dataset at a time. Data can be randomly divided into equal-sized disjoint subsets and clustered using a standard algorithm. The centroids of subsets form an ensemble which can be solved by a centroid correspondence algorithm. The centroids are combined to form a global set of centroids.
2. **Dealing with different types of attributes:** Algorithms are designed to cluster numeric data. However, applications may require clustering other data types like nominal, binary, and ordinal. Nominal data is in alphabetical form and not in integer form. Binary attribute is of two types: symmetric binary and



**Figure 13.1** Clusters of arbitrary shapes.

asymmetric binary. In symmetric data, both values are equally important. For example, in gender, it is male and female. In asymmetric data, both values are not equally important. For example, in result, it is pass and fail. The clustering algorithm should also work for complex data types such as graphs, sequences, images, and documents.

3. **Discovery of clusters with arbitrary shape:** Generally, clustering algorithms are to determine spherical clusters. Due to the characteristics and diverse nature of the data used, clusters may be of arbitrary shapes and can be nested within one another. For example, the cluster pattern for active and inactive volcanoes has chain-like patterns as shown in Fig. 13.1.

Traditional clustering algorithms, such as  $k$ -means and  $k$ -medoids, fail to detect non-spherical shapes. Thus, it is important to have clustering algorithms that can detect clusters of any arbitrary shape.

4. **Avoiding domain knowledge to determine input parameters:** Many algorithms require domain knowledge like the desired number of clusters in the form of input. Thus, the clustering results may become sensitive to the input parameters. Such parameters are often hard to determine for high dimensionality data. Domain knowledge requirement affects the quality of clustering and burdens the user.

For example, in  $k$ -means algorithm, the metric used to compare results for different values of  $k$  is the mean distance between data points and their cluster centroid. Increasing the number of clusters will always reduce the distance of data points to the extreme of reaching zero when  $k$  is the same as the number of data points. Thus, this cannot be used. Instead, to roughly determine  $k$ , the mean distance to the centroid is plotted as a function of  $k$  and the “elbow point”, where the rate of decrease sharply shifts. This is shown in Fig. 13.2.

5. **Handling noisy data:** Real-world data, which is the input of clustering algorithms, are mostly affected by noise. This results in poor-quality clusters. Noise is an unavoidable problem, which affects the data collection and data preparation processes. Therefore, the algorithms we use should be able to deal with noise. There are two types of noise:

- Attribute noise includes implicit errors introduced by measurement tools. They are induced by different types of sensors.

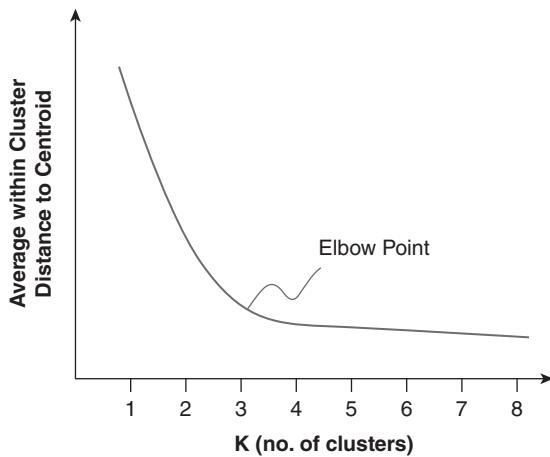


Figure 13.2 Elbow method.

- Random errors introduced by batch processes or experts when the data is gathered. This can be induced in document digitalization process.
6. **Incremental clustering:** The database used for clustering needs to be updated by adding new data (incremental updates). Some clustering algorithms cannot incorporate incremental updates but have to recompute a new clustering from scratch. The algorithms which can accommodate new data without reconstructing the clusters are called *incremental clustering algorithms*. It is more effective to use incremental clustering algorithms.
  7. **Insensitivity to input order:** Some clustering algorithms are sensitive to the order in which data objects are entered. Such algorithms are not ideal as we have little idea about the data objects presented. Clustering algorithms should be insensitive to the input order of data objects.
  8. **Handling high-dimensional data:** A dataset can contain numerous dimensions or attributes. Generally, clustering algorithms are good at handling low-dimensional data such as datasets involving only two or three dimensions. Clustering algorithms which can handle high-dimensional space are more effective.
  9. **Handling constraints:** Constrained clustering can be considered to contain a set of must-link constraints, cannot-link constraints, or both. In a must-link constraint, two instances in the must-link relation should be included in the same cluster. On the other hand, a cannot-link constraint specifies that the two instances cannot be in the same cluster. These sets of constraints act as guidelines to cluster the entire dataset. Some constrained clustering algorithms cancel the clustering process if they cannot form clusters which satisfy the specified constraints. Others try to minimize the amount of constraint violation if it is impossible to find a clustering which satisfies the constraints. Constraints can be used to select a clustering model to follow among different clustering methods. A challenging task is to find data groups with good clustering behavior that satisfy specified constraints.
  10. **Interpretability and usability:** Users require the clustering results to be interpretable, usable, and include all the elements. Clustering is always tied with specific semantic interpretations and applications. The applications should be able to use the information retrieved after clustering in a useful manner.

## 13.2 Types of Clustering

Clustering algorithms can be classified into two main subgroups:

1. **Hard clustering:** Each data point either belongs to a cluster completely or not.
2. **Soft clustering:** Instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.

Clustering algorithms can also be classified as follows:

1. Partitioning method.
2. Hierarchical method.
3. Density-based method.
4. Grid-based method.

However, the focus in this chapter is on partitioning method and hierarchical-based methods.

### 13.2.1 Partitioning Method

Partitioning means division. Suppose we are given a database of  $n$  objects and we need to partition this data into  $k$  partitions of data. Within a partition there exists some similarity among the items. So each partition will represent a cluster and  $k \leq n$ . It means that it will classify the data into  $k$  groups, each group contains at least one object and each object must belong to exactly one group. Although this is the general requirement, in soft clustering an object can belong to two clusters also. Most partitioning methods are distance-based.

For a given number of partitions (say  $k$ ), the partitioning method will create an initial partitioning. Then it uses the iterative relocation technique to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are close to each other, whereas objects in different clusters are far from each other. Some other criteria can be used for judging the quality of partitions.

Partition-based clustering is often computationally expensive and hence most of the methods apply heuristic methods, including the greedy approach which improves the quality of cluster arriving at a local optimum.

These heuristic clustering methods result in spherical clusters. For complex-shaped clusters and for large datasets, some extensions are required for partition-based methods.

### 13.2.2 Hierarchical Method

Hierarchical clustering is an alternative approach to partitioning clustering for identifying groups in a dataset. It does not require prespecifying the number of clusters to be generated. The result of hierarchical clustering is a tree-based representation of the objects, which is also known as *dendrogram*. Observations can be subdivided into groups by cutting the dendrogram at a desired similarity level. We classify hierarchical methods on the basis of how the hierarchical decomposition is formed. There are two approaches:

1. **Agglomerative approach:** This approach is also known as the *bottom-up approach*. In this approach, we start with each object forming a separate group. It keeps on merging the objects or groups that are close to one another. It keeps on doing so until all of the groups are merged into one or until the termination condition holds.
2. **Divisive approach:** This approach is also known as the *top-down approach*. In this approach, we start with all of the objects in the same cluster. In the continuous iteration, a cluster is split up into smaller clusters. It is done until each object is in one cluster or the termination condition holds. This method is rigid, that is, once a merging or splitting is done, it can never be undone.

### 13.2.3 Density-Based Methods

Density-based clustering algorithm finds nonlinear shapes clusters based on the density. Density-based spatial clustering of applications with noise (DBSCAN) is the most widely used density-based algorithm. It uses the concept of density reachability and density connectivity.

1. **Density reachability:** A point “p” is said to be density reachable from a point “q” if it is within  $\varepsilon$  distance from point “q” and “q” has sufficient number of points in its neighbors that are within distance  $\varepsilon$ .
2. **Density connectivity:** Points “p” and “q” are said to be density-connected if there exists a point “r” which has sufficient number of points in its neighbors and both the points are within  $\varepsilon$  distance. This is called *chaining process*. So, if “q” is neighbor of “r”, “r” is neighbor of “s”, “s” is neighbor of “t”, and “t” is neighbor of “p”; this implies that “q” is neighbor of “p”.

### 13.2.4 Grid-Based Methods

The grid-based clustering approach differs from the conventional clustering algorithms in that it is concerned not with data points but with the value space that surrounds the data points. In general, a typical grid-based clustering algorithm consists of the following five basic steps:

1. Create the grid structure, i.e., partitioning the data space into a finite number of cells.
2. Calculating the cell density for each cell.
3. Sorting the cells according to their densities.
4. Identifying cluster centers.
5. Traversal of neighbor cells.

## 13.3 Partitioning Methods of Clustering

---

The most fundamental clustering method is the partitioning method. This method assumes that we already know the number of clusters to be formed, which organizes the objects of a set into several exclusive groups or clusters. If  $k$  is the number of clusters to be formed given a dataset  $D$  of  $n$  objects, the partitioning algorithm organizes the objects into  $k$  partitions ( $k \leq n$ ), where each partition represents a cluster. The objective function in this type of partitioning is that the similarity among the data items within a cluster is higher than the elements in a different cluster. In other words, inter-cluster similarity is higher than intra-cluster similarity.

### 13.3.1 $k$ -Means Algorithm

The most well-known clustering algorithm is probably  $k$ -means. It is taught in a lot of introductory data science and machine learning classes. It is easy to understand and implement.

The main concept is to define  $k$  cluster centers. The cluster centers should be kept in such a way that it covers the data points of the entire dataset. The best way to do so is to keep data points as far away from each other as possible. We can then associate each data point to the nearest cluster center. The initial grouping of data is completed when there is no data point remaining. Once the grouping is done, new centroids are computed. These again form clusters based on the new cluster centers. The process is repeated till no more changes are done, which implies the cluster centers do not change any more. This algorithm aims at minimizing an objective function known as squared error function, which is given by

$$J(V) = \sum_{i=1}^c \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2 \quad (13.1)$$

where

$\|x_i - v_j\|$  is the Euclidean distance between  $x_i$  and  $v_j$ .

$c_i$  is the number of data points in  $i$ th cluster.

$c$  is the number of cluster centers.

The objective function aims for high intra-cluster similarity and low inter-cluster similarity. This function tries to make the resulting  $k$  clusters as compact and as separate as possible. Optimizing the within-cluster variation is computationally challenging since the problem is NP-hard in general even for two clusters (that is,  $k = 2$ ). To overcome the prohibitive computational cost for the exact solution, greedy approaches are often used in practice.

### 13.3.1.1 Steps in $k$ -Means Clustering Algorithm

Let us study the steps in  $k$ -means clustering algorithm with the following example. Let  $X = \{x_1, x_2, x_3, \dots, x_n\}$  be the set of data points and  $V = \{v_1, v_2, \dots, v_c\}$  be the set of centers. The steps are:

1. Randomly select  $c$  cluster centers.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster having a minimum distance from it and the cluster center.
4. Recalculate the new cluster center using Eq. (13.2).

$$v_i = \left( \frac{1}{c_i} \right) \sum_{j=1}^{c_i} x_j \quad (13.2)$$

where  $c_i$  represents the number of data points in the  $i$ th cluster.

5. Recalculate the distance between each data point and the newly obtained cluster centers.
6. If no data point was reassigned then stop, otherwise repeat steps 3 to 5.

The advantages of  $k$ -means clustering algorithm are:

1. Fast, robust, and easier to understand.
2. Relatively efficient: The computational complexity of the algorithm is  $O(tknd)$ , where  $n$  is the number of data objects,  $k$  is the number of clusters,  $d$  is the number of attributes in each data object, and  $t$  is the number of iterations. Normally,  $k, t, d \ll n$ . That is, the number of clusters attributes and iterations will be very small compared to the number of data objects in the dataset.
3. Gives best result when dataset are distinct or well separated from each other.

The disadvantages of  $k$ -means clustering algorithm are:

1. It requires prior specification of number of clusters.
2. It is not able to cluster highly overlapping data.
3. It is variant to nonlinear transformations, that is, with different representation of data we get different results (data represented in form of Cartesian coordinates and polar coordinates will give different results).
4. It provides the local optima of the squared error function.
5. Random choosing of the cluster center cannot lead to fruitful result.
6. Applicable only when the mean is defined, that is, fails for categorical data.
7. Unable to handle noisy data and outliers.

### 13.3.1.1.1 *k*-Means Solved Examples in One-Dimensional Data

#### Solved Problem 13.1

Apply *k*-means algorithm in given data for  $k = 3$  (that is, 3 clusters). Use  $C_1(2)$ ,  $C_2(16)$ , and  $C_3(38)$  as initial cluster centers. Data: 2, 4, 6, 3, 31, 12, 15, 16, 38, 35, 14, 21, 23, 25, 30.

#### **Solution:**

The initial cluster centers are given as  $C_1(2)$ ,  $C_2(16)$ , and  $C_3(38)$ . Calculating the distance between each data point and cluster centers, we get the following table.

Data Points	Distance from $C_1(2)$	Distance from $C_2(16)$	Distance from $C_3(38)$
2	$(2 - 2)^2 = 0$	$(2 - 16)^2 = 196$	$(2 - 38)^2 = 1296$
4	$(4 - 2)^2 = 4$	$(4 - 16)^2 = 144$	$(4 - 38)^2 = 1156$
6	$(6 - 2)^2 = 16$	$(6 - 16)^2 = 100$	$(6 - 38)^2 = 1024$
3	$(3 - 2)^2 = 1$	$(3 - 16)^2 = 169$	$(3 - 38)^2 = 1225$
31	$(31 - 2)^2 = 841$	$(31 - 16)^2 = 225$	$(31 - 38)^2 = 49$
12	$(12 - 2)^2 = 100$	$(12 - 16)^2 = 16$	$(12 - 38)^2 = 676$
15	$(15 - 2)^2 = 169$	$(15 - 16)^2 = 1$	$(15 - 38)^2 = 529$
16	$(16 - 2)^2 = 196$	$(16 - 16)^2 = 0$	$(16 - 38)^2 = 484$
38	$(38 - 2)^2 = 1296$	$(38 - 16)^2 = 484$	$(38 - 38)^2 = 0$
35	$(35 - 2)^2 = 1089$	$(35 - 16)^2 = 361$	$(35 - 38)^2 = 9$
14	$(14 - 2)^2 = 144$	$(14 - 16)^2 = 4$	$(14 - 38)^2 = 576$
21	$(21 - 2)^2 = 361$	$(21 - 16)^2 = 25$	$(21 - 38)^2 = 289$
23	$(23 - 2)^2 = 441$	$(23 - 16)^2 = 49$	$(23 - 38)^2 = 225$
25	$(25 - 2)^2 = 529$	$(25 - 16)^2 = 81$	$(25 - 38)^2 = 169$
30	$(30 - 2)^2 = 784$	$(30 - 16)^2 = 196$	$(30 - 38)^2 = 64$

By assigning the data points to the cluster center whose distance from it is minimum of all the cluster centers, we get the following table.

$C_1(2)$	$C_2(16)$	$C_3(38)$
$m1 = 2$	$m2 = 16$	$m3 = 38$
{2, 3, 4, 6}	{12, 14, 15, 16, 21, 23, 25}	{31, 35, 38}
<b>New cluster centers</b>		
$m1 = 3.75$	$m2 = 18$	$m3 = 34.67$

Similarly, using the new cluster centers we can calculate the distance from it and allocate clusters based on minimum distance. It is found that there is no difference in the cluster formed and hence we stop this procedure. The final clustering result is given in the following table.

$C_1(3.75)$	$C_2(18)$	$C_3(34.67)$
$m1 = 3.75$	$m2 = 18$	$m3 = 34.67$
{2, 3, 4, 6}	{12, 14, 15, 16, 21, 23, 25}	{31, 35, 38}

**Solved Problem 13.2**

Apply  $k$ -means algorithm in given data for  $k = 2$ . Use  $C_1(80)$  and  $C_2(250)$  as initial cluster centers. Data: 234, 123, 456, 23, 34, 56, 78, 90, 150, 116, 117, 118, 199.

**Solution:**

We solve the numerical by following the calculations carried out in solved problem 13.1. The result is presented in the following table.

$C_1(80)$	$C_2(250)$
$m1 = 80$	$m2 = 250$
{23, 34, 56, 78, 90, 116, 117, 118, 123}	{150, 199, 234, 456}
$m1 = 83.9$	$m2 = 259.75$
{23, 34, 56, 78, 90, 116, 117, 118, 123}	{150, 199, 234, 456}
$m1 = 90.5$	$m2 = 296.3$
{23, 34, 56, 78, 90, 116, 117, 118, 123, 150}	{199, 234, 456}
$m1 = 90.5$	$m2 = 296.3$
{23, 34, 56, 78, 90, 116, 117, 118, 123, 150}	{199, 234, 456}

**13.3.1.1.2  $k$ -Means Solved Examples in Two-Dimensional Data****Solved Problem 13.3**

Apply  $k$ -means clustering for the datasets given in Table 13.1 for two clusters. Tabulate all the assignments.

**Table 13.1** Sample dataset for  $k$ -means clustering

<i>Sample No.</i>	<i>X</i>	<i>Y</i>
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

**Solution:**

Sample No.	X	Y	Assignment
1	185	72	C1
2	170	56	C2

**Centroid:**  $C1 = (185, 72)$  and  $C2 = (170, 56)$

**First Iteration:**

Distance from  $C1$  is Euclidean distance between  $(185, 72)$  and  $(168, 60) = 20.808$

Distance from  $C2$  is Euclidean distance between  $(170, 56)$  and  $(168, 60) = 4.472$

Since  $C2$  is closer to  $(168, 60)$ , the sample belongs to  $C2$ .

Sample No.	X	Y	Assignment
1	185	72	C1
2	170	56	C2
3	168	60	C2
4	179	68	
5	182	72	
6	188	77	

Similarly,

1. Distance from  $C1$  for  $(179, 68) = 7.21$   
Distance from  $C2$  for  $(179, 68) = 15$   
Since  $C1$  is closer to  $(179, 68)$ , the sample belongs to  $C1$ .
2. Distance from  $C1$  for  $(182, 72) = 3$   
Distance from  $C2$  for  $(182, 72) = 20$   
Since  $C1$  is closer to  $(182, 72)$ , the sample belongs to  $C1$ .
3. Distance from  $C1$  for  $(188, 77) = 5.83$   
Distance from  $C2$  for  $(188, 77) = 27.66$   
Since  $C1$  is closer to  $(188, 77)$ , the sample belongs to  $C1$ .

Sample No.	X	Y	Assignment
1	185	72	C1
2	170	56	C2
3	168	60	C2
4	179	68	C1
5	182	72	C1
6	188	77	C1

The new centroid for  $C1$  is

$$\left( \frac{185+179+182+188}{4}, \frac{72+68+72+77}{4} \right) = (183.5, 73)$$

The new centroid for  $C2$  is

$$\left( \frac{170+168}{2}, \frac{56+60}{2} \right) = (169, 58)$$

#### **Second Iteration:**

Distance from  $C1$  is Euclidean distance between  $(183.5, 73)$  and  $(168, 60) = 20.2$

Distance from  $C2$  is Euclidean distance between  $(169, 58)$  and  $(168, 60) = 2.24$

Since  $C2$  is closer to  $(168, 60)$ , the sample belongs to  $C2$ .

Similarly,

1. Distance from  $C1$  for  $(179, 68) = 6.73$   
Distance from  $C2$  for  $(179, 68) = 14.14$   
Since  $C1$  is closer to  $(179, 68)$ , the sample belongs to  $C1$ .
2. Distance from  $C1$  for  $(182, 72) = 1.80$   
Distance from  $C2$  for  $(182, 72) = 19.10$   
Since  $C1$  is closer to  $(182, 72)$ , the sample belongs to  $C1$ .
3. Distance from  $C1$  for  $(188, 77) = 6.02$   
Distance from  $C2$  for  $(188, 77) = 26.87$   
Since  $C1$  is closer to  $(188, 77)$ , the sample belongs to  $C1$ .

<i>Sample No.</i>	<i>X</i>	<i>Y</i>	<i>Assignment</i>
1	185	72	<i>C1</i>
2	170	56	<i>C2</i>
3	168	60	<i>C2</i>
4	179	68	<i>C1</i>
5	182	72	<i>C1</i>
6	188	77	<i>C1</i>

After the second iteration, the assignment has not changed and hence the algorithm is stopped and the points are clustered.

#### **13.3.2 *k*-Medoids**

The  $k$ -medoids algorithm is a clustering algorithm very similar to the  $k$ -means algorithm. Both  $k$ -means and  $k$ -medoids algorithms are partitional and try to minimize the distance between points and cluster center. In contrast to the  $k$ -means algorithm,  $k$ -medoids chooses data points as centers and uses Manhattan distance to define the distance between cluster centers and data points. This technique clusters the dataset of  $n$  objects into  $k$  clusters, where the number of clusters  $k$  is known in prior. It is more robust to noise and outliers as compared to  $k$ -means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances. A medoid is defined as an object of a cluster whose average dissimilarity to all the objects in the cluster is minimal.

The Manhattan distance between two vectors in an  $n$ -dimensional real vector space is given by Eq. (13.2). It is used in computing the distance between a data point and its cluster center.

$$d_1(p, q) = \|p - q\|_1 = \sum_{i=1}^n |p_i - q_i| \quad (13.2)$$

The most common algorithm in  $k$ -medoid clustering is Partitioning Around Medoids (PAM) algorithm. PAM uses a greedy search which is faster than the exhaustive search and may not find the optimum solution. It works as follows:

1. Initialize: select  $k$  of the  $n$  data points as the medoids.
2. Associate each data point to the closest medoid.
3. While the cost of the configuration decreases: For each medoid  $m$  and for each non-medoid data point  $o$ :
  - Swap  $m$  and  $o$ , recompute the cost (sum of distances of points to their medoid).
  - If the total cost of the configuration increased in the previous step, undo the swap.

### Solved Problem 13.4

Cluster the following dataset of 6 objects into two clusters, that is,  $k = 2$ .

X1	2	6
X2	3	4
X3	3	8
X4	4	2
X5	6	2
X6	6	4

#### Solution:

**Step 1:** Two observations  $c1 = X2 = (3, 4)$  and  $c2 = X6 = (6, 4)$  are randomly selected as medoids (cluster centers).

**Step 2:** Manhattan distances are calculated to each center to associate each data object to its nearest medoid.

Data Object		Distance To	
Sample	Point	$c1 = (3, 4)$	$c2 = (6, 4)$
X1	(2, 6)	3	6
X2	(3, 4)	0	3
X3	(3, 8)	4	7
X4	(4, 2)	3	4
X5	(6, 2)	5	2
X6	(6, 4)	3	0
Cost		10	2

**Step 3:** We select one of the non-medoids  $O'$ . Let us assume  $O' = (6, 2)$ . So now the medoids are  $c1(3, 4)$  and  $O'(6, 2)$ . If  $c1$  and  $O'$  are the new medoids. We calculate the total cost involved.

Data Object		Distance To	
Sample	Point	$c1 = (3, 4)$	$c2 = (6, 2)$
X1	(2, 6)	3	8
X2	(3, 4)	0	5
X3	(3, 8)	4	9
X4	(4, 2)	3	2
X5	(6, 2)	5	0
X6	(6, 4)	3	2
Cost		7	4

So cost of swapping medoid from  $c2$  to  $O'$  is 11. Since the cost is less, this is considered as a better cluster assignment. Here swapping is done as the cost is less.

**Step 4:** We select another non-medoid  $O'$ . Let us assume  $O' = (4, 2)$ . So now the medoids are  $c1(3, 4)$  and  $O'(4, 2)$ . If  $c1$  and  $O'$  are new medoids, we calculate the total cost involved.

Data Object		Distance To	
Sample	Point	$c1 = (3, 4)$	$c2 = (4, 2)$
X1	(2, 6)	3	6
X2	(3, 4)	0	3
X3	(3, 8)	4	7
X4	(4, 2)	3	0
X5	(6, 2)	5	2
X6	(6, 4)	3	4
Cost		7	8

So cost of swapping medoid from  $c2$  to  $O'$  is 15. Since the cost is more, this cluster assignment is not considered and the swapping is not done.

Thus, we try other non-medoids points to get minimum cost. The assignment with minimum cost is considered the best. For some applications,  $k$ -medoids show better results than  $k$ -means. The most time-consuming part of the  $k$ -medoids algorithm is the calculation of the distances between objects. The distances matrix can be computed in advance to speed-up the process.

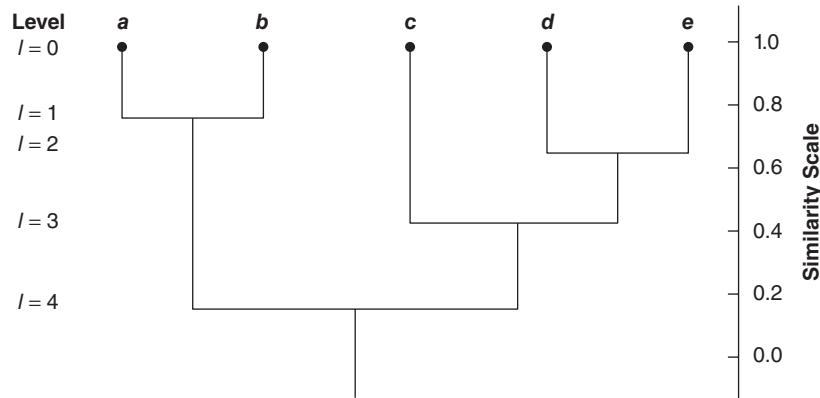
## 13.4 Hierarchical Methods

The hierarchical agglomerative clustering methods are most commonly used. The construction of a hierarchical agglomerative classification can be achieved by the following general algorithm.

1. Find the two closest objects and merge them into a cluster.
2. Find and merge the next two closest points, where a point is either an individual object or a cluster of objects.
3. If more than one cluster remains, return to step 2.

### 13.4.1 Agglomerative Algorithms

Agglomerative algorithm follows a bottom-up strategy, treating each object from its own cluster and iteratively merging clusters until a single cluster is formed or a terminal condition is satisfied. According to some similarity measure, the merging is done by choosing the closest clusters first. A dendrogram, which is a tree like structure, is used to represent hierarchical clustering. Individual objects are represented by leaf nodes and the clusters are represented by root nodes. A representation of a dendrogram is shown in Fig. 13.3.



**Figure 13.3** Dendrogram.

#### 13.4.1.1 Distance Measures

One of the major factors in clustering is the metric that is used to measure the distance between two clusters, where each cluster is generally a set of objects. The distance between two objects or points  $p$  and  $p'$  are computed using Eqs. (13.3) to (13.6). Let  $C_i$  be the cluster and  $n_i$  is the number of objects in  $C_i$ . They are also known as linkage measures.

Minimum distance:

$$\text{dist}_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\} \quad (13.3)$$

Maximum distance:

$$\text{dist}_{\max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\} \quad (13.4)$$

Mean distance:

$$\text{dist}_{\text{mean}}(C_i, C_j) = |m_i - m_j| \quad (13.5)$$

Average distance:

$$\text{dist}_{\text{avg}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'| \quad (13.6)$$

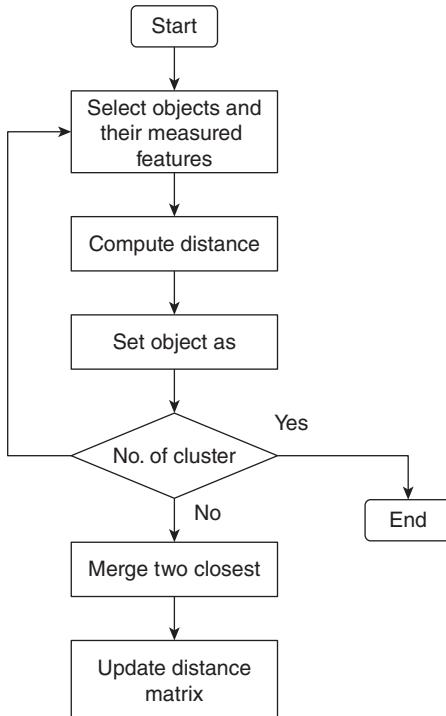
When an algorithm uses the minimum distance,  $d_{\min}(C_i, C_j)$ , to measure the distance between clusters, it is called *nearest-neighbor clustering algorithm*. If the clustering process is terminated when the distance between the nearest clusters exceeds a user-defined threshold, it is called *single-linkage algorithm*. Agglomerative hierarchical clustering algorithm (with minimum distance measure) is called *minimum spanning tree algorithm* since spanning tree of a graph is a tree that connects all vertices and a minimal spanning tree is one with the least sum of edge weights.

An algorithm that uses the maximum distance,  $d_{\max}(C_i, C_j)$ , to measure the distance between clusters is called *farthest-neighbor clustering algorithm*. If clustering is terminated when the maximum distance exceeds a user-defined threshold, it is called *complete-linkage algorithm*.

The minimum and maximum measures tend to be sensitive to outliers or noisy data. The third method thus suggests to take the average distance to rule out outlier problems. Another advantage is that it can handle categoric data as well.

**Algorithm:** The agglomerative algorithm is carried out in three steps and the flowchart is shown in Fig. 13.4.

1. Convert object attributes to distance matrix.
2. Set each object as a cluster (thus, if we have  $N$  objects, we will have  $N$  clusters at the beginning).
3. Repeat until number of clusters is one.
  - Merge two closest clusters.
  - Update distance matrix.



**Figure 13.4** Flowchart of agglomerative algorithm.

### 13.4.1.2 Agglomerative Algorithm: Single Link

Single-nearest distance or single linkage is the agglomerative method that uses the distance between the closest members of the two clusters.

#### Solved Problem 13.5

Find the clusters using single link technique. Use Euclidean distance and draw the dendrogram.

<i>Sample No.</i>	<i>X</i>	<i>Y</i>
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

**Solution:**

To compute distance matrix:

$$d((x, y)(a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

Euclidean distance:

$$\begin{aligned} d(P1, P2) &= \sqrt{(0.4 - 0.22)^2 + (0.53 - 0.38)^2} \\ &= \sqrt{(0.18)^2 + (0.15)^2} \\ &= \sqrt{0.0324 + 0.0225} = 0.23 \end{aligned}$$

The distance matrix is:

$$\begin{pmatrix} & P1 & P2 & P3 & P4 & P5 & P6 \\ P1 & 0 & & & & & \\ P2 & 0.23 & 0 & & & & \\ P3 & & & 0 & & & \\ P4 & & & & 0 & & \\ P5 & & & & & 0 & \\ P6 & & & & & & 0 \end{pmatrix}$$

Similarly,

$$\begin{aligned} d(P1, P3) &= \sqrt{(0.4 - 0.35)^2 + (0.53 - 0.32)^2} \\ &= \sqrt{(0.05)^2 + (0.21)^2} \\ &= \sqrt{0.0025 + 0.0441} = 0.216 = 0.22 \end{aligned}$$

$$\begin{aligned} d(P1, P4) &= \sqrt{(0.4 - 0.26)^2 + (0.53 - 0.19)^2} \\ &= \sqrt{(0.14)^2 + (0.34)^2} \\ &= \sqrt{0.0196 + 0.1156} = 0.3676 = 0.37 \end{aligned}$$

$$\begin{aligned} d(P1, P5) &= \sqrt{(0.4 - 0.08)^2 + (0.53 - 0.41)^2} \\ &= \sqrt{(0.32)^2 + (0.12)^2} \\ &= \sqrt{0.1024 + 0.0144} = 0.3417 = 0.34 \end{aligned}$$

$$\begin{aligned} d(P1, P6) &= \sqrt{(0.4 - 0.45)^2 + (0.53 - 0.30)^2} \\ &= \sqrt{(0.05)^2 + (0.23)^2} \\ &= \sqrt{0.0025 + 0.0529} = 0.2354 = 0.24 \end{aligned}$$

$$\begin{aligned} d(P2, P3) &= \sqrt{(0.22 - 0.35)^2 + (0.38 - 0.32)^2} \\ &= \sqrt{(-0.13)^2 + (0.06)^2} \\ &= \sqrt{0.0169 + 0.0036} = 0.1432 = 0.14 \end{aligned}$$

$$\begin{aligned} d(P2, P4) &= \sqrt{(0.22 - 0.26)^2 + (0.38 - 0.19)^2} \\ &= \sqrt{(-0.04)^2 + (0.19)^2} \\ &= \sqrt{0.0016 + 0.0361} = 0.1942 = 0.19 \end{aligned}$$

$$\begin{aligned} d(P2, P5) &= \sqrt{(0.22 - 0.08)^2 + (0.38 - 0.41)^2} \\ &= \sqrt{(0.14)^2 + (-0.03)^2} \\ &= \sqrt{0.0196 + 0.0009} = 0.1432 = 0.14 \end{aligned}$$

$$\begin{aligned} d(P2, P6) &= \sqrt{(0.22 - 0.45)^2 + (0.38 - 0.30)^2} \\ &= \sqrt{(-0.23)^2 + (0.08)^2} \\ &= \sqrt{0.0529 + 0.0064} = 0.2435 = 0.24 \end{aligned}$$

$$\begin{aligned} d(P3, P4) &= \sqrt{(0.35 - 0.26)^2 + (0.32 - 0.19)^2} \\ &= \sqrt{(0.03)^2 + (0.13)^2} \\ &= \sqrt{0.0009 + 0.0169} = 0.1334 = 0.13 \end{aligned}$$

$$\begin{aligned}
 d(P3, P5) &= \sqrt{(0.35 - 0.08)^2 + (0.32 - 0.41)^2} \\
 &= \sqrt{(0.27)^2 + (-0.09)^2} \\
 &= \sqrt{0.0729 + 0.0081} = 0.2846 = 0.28
 \end{aligned}$$

$$\begin{aligned}
 d(P3, P6) &= \sqrt{(0.35 - 0.45)^2 + (0.32 - 0.30)^2} \\
 &= \sqrt{(-0.1)^2 + (0.02)^2} \\
 &= \sqrt{0.01 + 0.0004} = 0.10198 = 0.10
 \end{aligned}$$

$$\begin{aligned}
 d(P4, P5) &= \sqrt{(0.26 - 0.08)^2 + (0.19 - 0.41)^2} \\
 &= \sqrt{(0.18)^2 + (-0.22)^2} \\
 &= \sqrt{0.0049 + 0.0484} = 0.2309 = 0.23
 \end{aligned}$$

$$\begin{aligned}
 d(P4, P6) &= \sqrt{(0.26 - 0.45)^2 + (0.19 - 0.30)^2} \\
 &= \sqrt{(-0.19)^2 + (-0.11)^2} \\
 &= \sqrt{0.0361 + 0.0121} = 0.2195 = 0.22
 \end{aligned}$$

$$\begin{aligned}
 d(P5, P6) &= \sqrt{(0.08 - 0.45)^2 + (0.41 - 0.30)^2} \\
 &= \sqrt{(-0.37)^2 + (0.11)^2} \\
 &= \sqrt{0.1369 + 0.0121} = 0.3860 = 0.39
 \end{aligned}$$

The distance matrix is:

$$\left( \begin{array}{cccccc} & P1 & P2 & P3 & P4 & P5 & P6 \\ P1 & 0 & & & & & \\ P2 & 0.23 & 0 & & & & \\ P3 & 0.22 & 0.14 & 0 & & & \\ P4 & 0.37 & 0.19 & 0.13 & 0 & & \\ P5 & 0.34 & 0.14 & 0.28 & 0.23 & 0 & \\ P6 & 0.24 & 0.24 & 0.10 & 0.22 & 0.39 & 0 \end{array} \right)$$

Merging the two closest members of the two clusters and finding the minimum element in distance matrix, we get

$$\begin{pmatrix} & P1 & P2 & P3 & P4 & P5 & P6 \\ P1 & 0 & & & & & \\ P2 & 0.23 & 0 & & & & \\ P3 & 0.22 & 0.14 & 0 & & & \\ P4 & 0.37 & 0.19 & 0.13 & 0 & & \\ P5 & 0.34 & 0.14 & 0.28 & 0.23 & 0 & \\ P6 & 0.24 & 0.24 & 0.10 & 0.22 & 0.39 & 0 \end{pmatrix}$$

Here the minimum value is 0.10 and hence we combine P3 and P6. Now, form cluster of elements corresponding to minimum value and update distance matrix. To update the distance matrix

$$\min((P3, P6), P1) = \min((P3, P1), (P6, P1)) = \min(0.22, 0.24) = 0.22$$

$$\min((P3, P6), P2) = \min((P3, P2), (P6, P2)) = \min(0.14, 0.24) = 0.14$$

$$\min((P3, P6), P4) = \min((P3, P4), (P6, P4)) = \min(0.13, 0.22) = 0.13$$

$$\min((P3, P6), P5) = \min((P3, P5), (P6, P5)) = \min(0.28, 0.39) = 0.28$$

$$\begin{pmatrix} & P1 & P2 & P3, P6 & P4 & P5 \\ P1 & 0 & & & & \\ P2 & 0.23 & 0 & & & \\ P3, P6 & 0.22 & 0.14 & 0 & & \\ P4 & 0.37 & 0.19 & 0.13 & 0 & \\ P5 & 0.34 & 0.14 & 0.28 & 0.23 & 0 \end{pmatrix}$$

Merging two closest members of the two clusters and finding the minimum element in distance matrix.

$$\begin{pmatrix} & P1 & P2 & P3, P6 & P4 & P5 \\ P1 & 0 & & & & \\ P2 & 0.23 & 0 & & & \\ P3, P6 & 0.22 & 0.14 & 0 & & \\ P4 & 0.37 & 0.19 & 0.13 & 0 & \\ P5 & 0.34 & 0.14 & 0.28 & 0.23 & 0 \end{pmatrix}$$

Here the minimum value is 0.13 and hence we combine P3, P6 and P4. Now, form cluster of elements corresponding to minimum values and update distance matrix. To update the distance matrix

$$\min((P3, P6), P4), P1) = \min((P3, P6), P1), (P4, P1)) = \min(0.22, 0.37) = 0.22$$

$$\min((P3, P6), P4), P2) = \min((P3, P6), P2), (P4, P2)) = \min(0.14, 0.19) = 0.14$$

$$\min((P3, P6), P4), P5) = \min((P3, P6), P5), (P4, P5)) = \min(0.28, 0.23) = 0.23$$

$$\begin{pmatrix} & P1 & P2 & P3, P6, P4 & P5 \\ P1 & 0 & & & \\ P2 & 0.23 & 0 & & \\ P3, P6, P4 & 0.22 & 0.14 & 0 & \\ P5 & 0.34 & 0.14 & 0.23 & 0 \end{pmatrix}$$

Merging two closest members of the two clusters and finding the minimum element in distance matrix.

$$\begin{pmatrix} & P1 & P2 & P3, P6, P4 & P5 \\ P1 & 0 & & & \\ P2 & 0.23 & 0 & & \\ P3, P6, P4 & 0.22 & 0.14 & 0 & \\ P5 & 0.34 & 0.14 & 0.23 & 0 \end{pmatrix}$$

Here the minimum value is 0.14 and hence we combine P2 and P5. Now, form cluster of elements corresponding to minimum values and update distance matrix. To update the distance matrix

$$\min((P2, P5), P1) = \min((P2, P1), (P5, P1)) = \min(0.23, 0.34) = 0.23$$

$$\min((P2, P5), (P3, P6, P4)) = \min((P2, (P3, P6, P4)), (P5, (P3, P6, P4))) = \min(0.14, 0.23) = 0.14$$

$$\begin{pmatrix} & P1 & P2, P5 & P3, P6, P4 \\ P1 & 0 & & \\ P2, P5 & 0.23 & 0 & \\ P3, P6, P4 & 0.22 & 0.14 & 0 \end{pmatrix}$$

Merging two closest members of the two clusters and finding the minimum element in distance matrix.

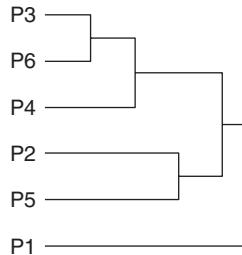
$$\begin{pmatrix} & P1 & P2, P5 & P3, P6, P4 \\ P1 & 0 & & \\ P2, P5 & 0.23 & 0 & \\ P3, P6, P4 & 0.22 & 0.14 & 0 \end{pmatrix}$$

Here the minimum value is 0.14 and hence we combine P2, P5 and P3, P6, P4. Now, form cluster of elements corresponding to minimum values and update distance matrix. To update the distance matrix

$$\min((P2, P5, P3, P6, P4), P1) = \min((P2, P5), P1), ((P3, P6, P4), P1)) = \min(0.23, 0.22) = 0.22$$

$$\begin{pmatrix} & P1 & P2, P5, P3, P6, P4 \\ P1 & 0 & \\ P2, P5, P3, P6, P4 & 0.22 & 0 \end{pmatrix}$$

The dendrogram can now be drawn as shown in Fig. 13.5.



**Figure 13.5** Dendrogram of the cluster formed.

#### 13.4.1.3 Agglomerative Algorithm: Complete Link

Complete farthest distance or complete linkage is the agglomerative method that uses the distance between the members that are farthest apart.

##### Solved Problem 13.6

For the given set of points, identify clusters using complete link agglomerative clustering.

##### **Solution:**

To compute distance matrix:

$$d[(x, y)(a, b)] = \sqrt{(x - a)^2 + (y - b)^2}$$

The Euclidean distance is:

$$\begin{aligned} d(P_1, P_2) &= \sqrt{(1.0 - 1.5)^2 + (1.0 - 1.5)^2} \\ &= \sqrt{0.25 + 0.25} = \sqrt{0.5} = 0.71 \end{aligned}$$

The distance matrix is:

$$\left( \begin{array}{cccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\ P_1 & 0 & & & & & \\ P_2 & 0.71 & 0 & & & & \\ P_3 & 5.66 & 4.95 & 0 & & & \\ P_4 & 3.6 & 2.92 & 2.24 & 0 & & \\ P_5 & 4.24 & 3.53 & 1.41 & 1.0 & 0 & \\ P_6 & 3.20 & 2.5 & 2.5 & 0.5 & 1.12 & 0 \end{array} \right)$$

Merging two closest members of the two clusters and finding the minimum element in distance matrix and forming the clusters, we get

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.71	0				
P3	5.66	4.95	0			
P4	3.6	2.92	2.24	0		
P5	4.24	3.53	1.41	1.0	0	
P6	3.20	2.5	2.5	0.5	1.12	0

Here the minimum value is 0.5 and hence we combine P4 and P6. To update the distance matrix  
 $\max(d(P4, P6), P1)) = \max(d(P4, P1), d(P6, P1)) = \max(3.6, 3.2) = 3.6$

	P1	P2	P3	P4, P6	P5
P1	0				
P2	0.71	0			
P3	5.66	4.95	0		
P4, P6	3.6	2.92	2.5	0	
P5	4.24	3.53	1.41	1.12	0

Merging two closest by finding the minimum element in distance matrix and forming the clusters, we get

	P1, P2	P3	P4, P6	P5
P1, P2	0			
P3	5.66	0		
P4, P6	3.6	2.5	0	
P5	4.24	1.41	1.12	0

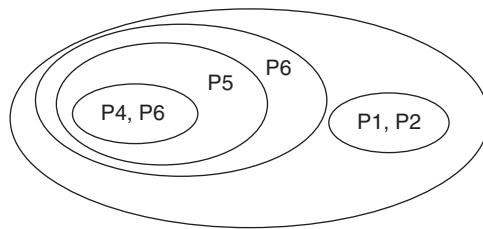
Merging two closest by finding the minimum element in distance matrix and forming the clusters, we get

	P1, P2	P3	P4, P6, P5
P1, P2	0		
P3	5.66	0	
P4, P6, P5	3.6	2.5	0

Merging two closest by finding the minimum element in distance matrix and forming the clusters, we get

	P1, P2	P4, P6, P5, P3
P1, P2	0	
P4, P6, P5, P3	5.66	0

The final cluster formed can now be drawn as shown in Fig. 13.6.



**Figure 13.6** Cluster formed after merging all data points.

#### 13.4.1.4 Agglomerative Algorithm: Average Link

Average-average distance or average linkage is the method that involves looking at the distances between all pairs and averages all of these distances. This is also called *Unweighted Pair Group Mean Averaging*.

##### Solved Problem 13.7

For the given set of points, identify clusters using average link agglomerative clustering.

	<i>A</i>	<i>B</i>
P1	1	1
P2	1.5	1.5
P3	5	5
P4	3	4
P5	4	4
P6	3	3.5

##### Solution:

The distance matrix is:

$$\begin{pmatrix} & \text{P1} & \text{P2} & \text{P3} & \text{P4} & \text{P5} & \text{P6} \\ \text{P1} & 0 & & & & & \\ \text{P2} & 0.71 & 0 & & & & \\ \text{P3} & 5.66 & 4.95 & 0 & & & \\ \text{P4} & 3.6 & 2.92 & 2.24 & 0 & & \\ \text{P5} & 4.24 & 3.53 & 1.41 & 1.0 & 0 & \\ \text{P6} & 3.20 & 2.5 & 2.5 & 0.5 & 1.12 & 0 \end{pmatrix}$$

Merging two closest members of the two clusters and finding the minimum element in distance matrix, we get

$$\begin{pmatrix} & P1 & P2 & P3 & P4 & P5 & P6 \\ P1 & 0 & & & & & \\ P2 & 0.71 & 0 & & & & \\ P3 & 5.66 & 4.95 & 0 & & & \\ P4 & 3.6 & 2.92 & 2.24 & 0 & & \\ P5 & 4.24 & 3.53 & 1.41 & 1.0 & 0 & \\ P6 & 3.20 & 2.5 & 2.5 & 0.5 & 1.12 & 0 \end{pmatrix}$$

Here the minimum value is 0.5 and hence we combine P4 and P6. To update the distance matrix average  $(d(P4, P6), P1) = \text{average}(d(P4, P1), d(P6, P1)) = \text{average}(3.6, 3.2) = 3.4$

$$\begin{pmatrix} & P1 & P2 & P3 & P4, P6 & P5 \\ P1 & 0 & & & & \\ P2 & 0.71 & 0 & & & \\ P3 & 5.66 & 4.95 & 0 & & \\ P4, P6 & 3.2 & 2.71 & 2.37 & 0 & \\ P5 & 4.24 & 3.53 & 1.41 & 1.06 & 0 \end{pmatrix}$$

Merging two closest by finding the minimum element in distance matrix and forming the clusters:

$$\begin{pmatrix} & P1, P2 & P3 & P4, P6 & P5 \\ P1, P2 & 0 & & & \\ P3 & 5.31 & 0 & & \\ P4, P6 & 2.96 & 2.5 & 0 & \\ P5 & 3.89 & 1.41 & 1.12 & 0 \end{pmatrix}$$

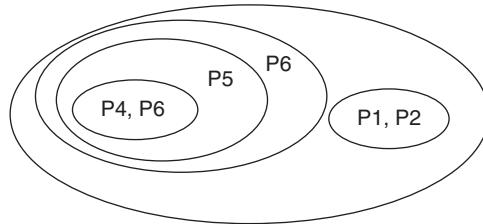
Merging two closest by finding the minimum element in distance matrix and forming the clusters:

$$\begin{pmatrix} & P1, P2 & P3 & P4, P6, P5 \\ P1, P2 & 0 & & \\ P3 & 5.66 & 0 & \\ P4, P6, P5 & 3.43 & 1.96 & 0 \end{pmatrix}$$

Merging two closest by finding the minimum element in distance matrix and forming the clusters:

$$\begin{pmatrix} & P1, P2 & P4, P6, P5, P3 \\ P1, P2 & 0 & \\ P4, P6, P5, P3 & 4.55 & 0 \end{pmatrix}$$

The final cluster formed can now be drawn as shown in Fig. 13.7.



**Figure 13.7** The final cluster formed merging all data points.

## Case Study

There are diverse applications using clustering. As discussed in previous sections, the concept of clustering is one wherein the output is not known. In other words, the training dataset has only input data samples. So based on some metrics of grouping or clustering, similar data items are grouped together in one cluster. Let us now see some of the major areas wherein this concept is extensively used.

### Case Study 1: Grouping of Similar Companies Based on Wikipedia Articles

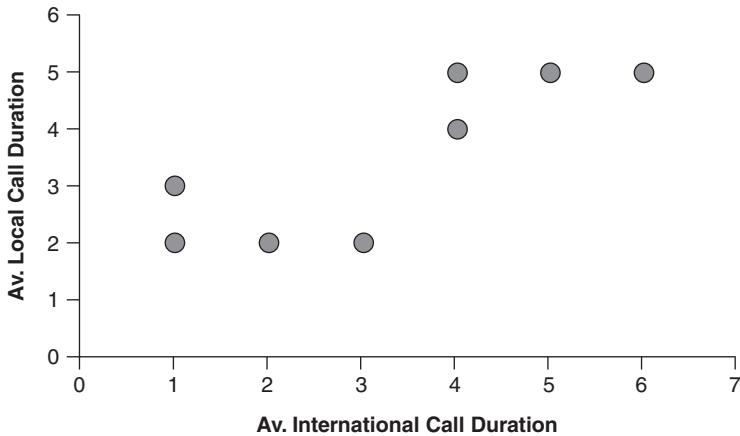
The  $k$ -means clustering algorithm is used to segment S&P (500 index) listed companies based on the text of Wikipedia articles about each one. Initially, the data is taken and preprocessed. This data is nothing but articles from Wikipedia about the companies. In the preprocessing phase, the Wikipedia formatting is removed, all texts are converted to lowercase, and non-alphanumeric characters are removed.

Around 500 companies were taken as input data. From this input data, feature hashing module tokenizes the text string and transforms the data into a series of numbers based on the hash value of each token. No linguistic analysis is performed in this step. Internally, the feature hashing module creates a dictionary of n-grams. After the dictionary has been built, the feature hashing module converts the dictionary terms into hash values. It then computes whether a feature was used in each case. For each row of text data, the module outputs a set of columns – one column for each hashed feature. Here, the dimensionality was reduced using PCA. So based on the highest variance, the first one or two columns of the transformed matrix is selected and clusters were built using the transformed dataset.

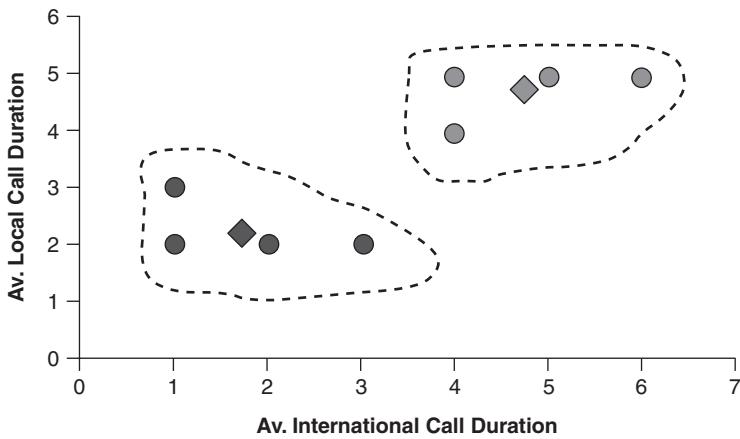
### Case Study 2: Telecom Cluster Analysis

We know that there are a number of telecom companies attract customers with a variety of packages. In the telecom sector, you have to realize that not every customer has similar needs and you need to strategize accordingly to attract all of them. Based on customer segmentation, the company as well as customers can have a the win-win scenario. Taking a sample of customers and based on their international and national call durations, clustering techniques are used to classify clusters into two main categories.

Let us take a small sample size of eight customers. Based on their duration of national and international calls, a scatter plot is drawn as shown below.



Using Euclidean distance metric to compute the centroids, the final clusters forms are shown in the figure below.



Based on the proximity between cluster centres and centroid of the formed clusters, the customer plans can be decided so that the company as well as the customers benefit from a chosen plan.

## Summary

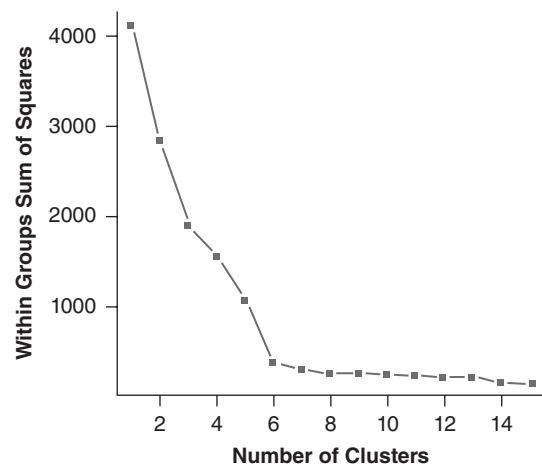
- Clustering is the process of grouping together data objects into multiple sets or clusters, so that objects within a cluster have high similarity when compared to objects outside of it.
- Similarity is measured by distance metrics and the most common among them is the Euclidean distance metrics.
- Clustering is also called data segmentation because clustering partitions large datasets into groups according to their similarity.
- Clustering is known as unsupervised learning because the class label information is not present.

- The applications of clustering are varied and include business intelligence, pattern recognition, image processing, biometrics, web technology, search engine, and text mining.
- The requirements of clustering is dependent on its scalability, number and types of attributes which have to be clustered, shape of the cluster to be identified, efficiency in handling noisy data and incremental data points to the existing clusters, handling high-dimensional data, and data with constraints.
- The basic types of clustering are hard clustering and soft clustering depending on whether the data points belong to only one cluster or whether they can be shared among clusters.
- Clustering algorithms are classified based on partitioning, hierarchical, density-based and grid-based clustering.
- Partitioning-based clustering algorithms are distance based.  $k$ -means and  $k$ -medoids are popular partition-based clustering algorithms. The number of clusters to be formed is initially specified.
- The result of hierarchical clustering is a tree-based representation of the objects, which is also known as dendrogram.
- Density-based clustering algorithm finds non-linear shaped clusters based on density. Density-based spatial clustering of applications with noise (DBSCAN) is the most widely used density-based algorithm. It uses the concept of density reachability and density connectivity.
- The grid-based clustering approach differs from conventional clustering algorithms in that it is concerned not with the data points but with the value space that surrounds the data points.

## Multiple-Choice Questions

1. Which of the following statements is true?
  - (a) Assignment of observations to clusters does not change between successive iterations in  $k$ -means.
  - (b) Assignment of observations to clusters changes between successive iterations in  $k$ -means.
  - (c) Assignment of observations to clusters always decrease between successive iterations in  $k$ -means.
  - (d) Assignment of observations to clusters always increase between successive iterations in  $k$ -means.
2. Which of the following can act as possible termination conditions in  $k$ -means?
  - i. For a fixed number of iterations.
  - ii. Assignment of observations to clusters does not change between iterations, except for cases with a bad local minimum.
  - iii. Centroids do not change between successive iterations.
- iv. Terminate when RSS falls below a threshold.
  - (a) i, iii, and iv
  - (b) i, ii, and iii
  - (c) i, ii, and iv
  - (d) All of the above
3. Which of the following algorithm is most sensitive to outliers?
  - (a)  $k$ -means clustering algorithm
  - (b)  $k$ -medians clustering algorithm
  - (c)  $k$ -modes clustering algorithm
  - (d)  $k$ -medoids clustering algorithm
4. Which of the following is finally produced by hierarchical clustering?
  - (a) Final estimate of cluster centroids
  - (b) Tree showing how close things are to each other
  - (c) Assignment of each point to clusters
  - (d) All of the above

5. What is the best choice for the number of clusters based on the following graph?



- (a) 5  
 (b) 6  
 (c) 14  
 (d) None of the above

### Very Short Answer Questions

- Give an example of an application for  $k$ -means clustering algorithm. Explain in brief.
- Explain the different distance measures used for clustering.
- Using  $k$ -means clustering, cluster the following data into two clusters. Show each step.
- {2, 4, 10, 12, 3, 20, 30, 11, 25}
- Compare between single link, complete link, and average link based on distance formula.
- Draw the flowchart of  $k$ -means algorithm.

### Short Answer Questions

- Compute the distance matrix for the  $x-y$  coordinates given in the following table.
- Use  $k$ -means algorithm to cluster the following dataset consisting of the scores of two variables on each of the seven individuals.

Point	x coordinate	y coordinate
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022

- How will you define the number of clusters in  $k$ -means clustering algorithm?

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

4. Use the  $k$ -means algorithm and Euclidean distance to cluster the following eight examples into three clusters:  $A_1 = (2, 10)$ ,  $A_2 = (2, 5)$ ,  $A_3 = (8, 4)$ ,  $A_4 = (5, 8)$ ,  $A_5 = (7, 5)$ ,  $A_6 = (6, 4)$ ,  $A_7 = (1, 2)$ ,  $A_8 = (4, 9)$ . The distance matrix based on the Euclidean distance is given in the following table.

	$A_1$	$A_2$	$A_3$	$A_4$	$A_5$	$A_6$	$A_7$	$A_8$
$A_1$	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
$A_2$		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
$A_3$			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
$A_4$				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
$A_5$					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
$A_6$						0	$\sqrt{29}$	$\sqrt{29}$
$A_7$							0	$\sqrt{58}$
$A_8$								0

Suppose the initial seeds (centers of each cluster) are  $A_1$ ,  $A_4$ , and  $A_7$ . Run the  $k$ -means algorithm for 1 epoch only. At the end of this epoch show:

- (a) The new clusters (that is, the examples belonging to each cluster).  
 (b) The centers of the new clusters.  
 5. Use single and complete link agglomerative clustering to group the data given in the following distance matrix. Show the dendograms.

	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

## Review Questions

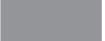
1. Use  $k$ -means algorithm to create three clusters for a given set of values:  $\{2, 3, 6, 8, 9, 12, 15, 18, 22\}$ .  
 2. Apply agglomerative clustering algorithm on the given data and draw the dendrogram. Show three clusters with its allocated points by using the single link method.

	a	b	c	d	e	f
a	0	$\sqrt{2}$	$\sqrt{10}$	$\sqrt{17}$	$\sqrt{5}$	$\sqrt{20}$
b	$\sqrt{2}$	0	$\sqrt{8}$	3	1	$\sqrt{18}$
c	$\sqrt{10}$	$\sqrt{8}$	0	$\sqrt{5}$	$\sqrt{5}$	2
d	$\sqrt{17}$	3	$\sqrt{5}$	0	2	3
e	$\sqrt{5}$	1	$\sqrt{5}$	2	0	$\sqrt{13}$
f	$\sqrt{20}$	$\sqrt{18}$	$\sqrt{2}$	3	$\sqrt{13}$	0

3. Apply complete link agglomerative clustering techniques on the given data to find the prominent clusters.

	P1	P2	P3	P4	P5	P6
P1	0	0.23	0.22	0.37	0.34	0.24
P2	0.23	0	0.14	0.19	0.14	0.24
P3	0.22	0.14	0	0.13	0.28	0.10
P4	0.37	0.19	0.13	0	0.23	0.22
P5	0.34	0.14	0.28	0.23	0	0.39
P6	0.24	0.24	0.10	0.22	0.39	0

4. Explain expectation–maximization algorithm.  
 5. What are the requirements for clustering?  
 6. What are the applications of clustering?



## Answers

---

**Multiple-Choice Answers**

1. (a)    2. (d)    3. (a)    4. (b)    5. (b)



## PART 4

# Optimization Techniques

---

### **Chapter 14**

Optimization



# 14

# Optimization

## LEARNING OBJECTIVES

- To understand the basics of optimization.
- To understand how to formulate constraint and unconstraint minimization problems.
- To introduce the concept of gradient descent and Newton's method.
- To introduce derivative free optimization techniques.
- To emphasize the differences between derivative-free and derivative-based optimization techniques.

## LEARNING OUTCOMES

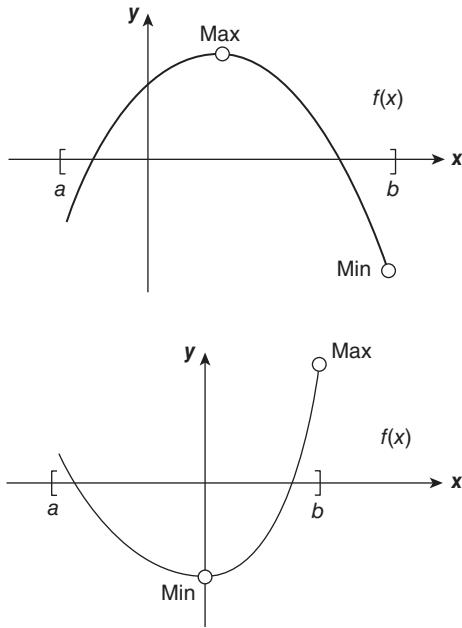
- Students will be able to appreciate the used of optimization in solving engineering problems.
- Students will be able to formulate the objective functions with or without constraints for constraint and unconstraint optimization problems, respectively.
- Students will be able to solve gradient descent and Newton's based optimization problems.
- Students will be able to appreciate the differences between derivative-free and derivative-based optimization techniques.

### 14.1 Introduction to Optimization

Optimization is the process of obtaining the best results under any given circumstances. We carry out optimization either to minimize the effort required or to maximize the desired benefit. The effort required or the benefit desired can be written as a function of decision variables. Thus, optimization is the process of finding the conditions that maximize or minimize a function. If we find a point  $x$  corresponding to the minimum value of function  $f(x)$ , the same point is the maximum value of the negative of the function  $-f(x)$ , as shown in Fig. 14.1.

Thus, we can consider optimization as a minimization problem since the maximum of a function can be found by seeking the minimum of its negative.

There is no single method to solve any type of optimization problem. Thus, we have different types of optimization methods. *Operations research* is a branch of mathematics concerned with these methodologies.



**Figure 14.1** Minimum of  $f(x)$  is same as maximum of  $-f(x)$ .

#### 14.1.1 Statement of an Optimization Problem

An optimization problem can be stated mathematically as follows.

Find  $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ , which minimizes  $f(X)$  subject to the constraints

$$h_j(X) \leq 0; j = 1, 2, \dots, m; \text{ and } k_j(X) = 0; j = 1, 2, \dots, p.$$

Where  $X$  is an  $n$ -dimensional vector called design vector,  $f(X)$  is the objective function, and  $h_j(X)$  and  $k_j(X)$  are the inequality and equality constraints, respectively. This is a constrained optimization problem. However, some optimization problems do not have any constraints.

## 14.2 Classification of Optimization Problems

Based on the nature of expressions for the objective function and the constraints, optimization problems can be classified as linear, nonlinear, geometric, and quadratic programming problems.

## 14.3 Linear vs Nonlinear Programming Problems

If the objective function and constraints are linear functions of the decision variables, then it is called linear programming (LP) problem. If any of the function is nonlinear, then it is a nonlinear programming (NLP) problem. Most of the general programming problems are nonlinear in nature.

## 14.4 Unconstrained Minimization Problems

A point  $X^*$  will be a relative minimum of  $f(X)$  if the following conditions are satisfied:

$$\frac{\partial f}{\partial x_i} = (X = X^*) = 0$$

where  $i = 1, 2, \dots, n$

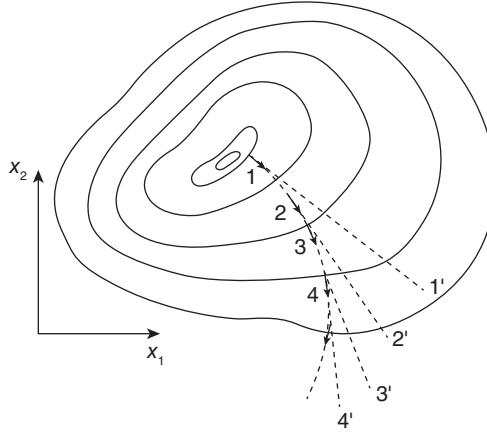
The point  $X^*$  is a relative minimum if the Hessian matrix is positive definite. A Hessian matrix is a square matrix of partial derivatives of scalar valued functions. This matrix is used to test if a critical point  $x$  is a local maximum, local minimum, or a saddle point.

1. If Hessian is positive definite at  $x$ , then the function  $f$  obtains a local minimum.
2. If Hessian is negative definite at  $x$ , then the function  $f$  obtains a local maximum.
3. If Hessian has both positive and negative eigenvalues at  $x$ , then  $x$  is a saddle point. This point is one where the slopes in the orthogonal directions are all zero but is not a local extremum of the function.

Thus, we can find the optimum point. If the function is not differentiable, then the minimum point cannot be found.

## 14.5 Gradient-Based Methods (Descent Methods)

The gradient of the objective function has a very important property that if we move along the gradient direction from any point, the function value increases at the fastest rate. Hence, the gradient direction is known as the direction of steepest ascent. This is illustrated in Fig. 14.2.



**Figure 14.2** Steepest ascent directions.

It is seen in Fig. 14.2 that the gradient vectors  $\nabla f$  evaluated at points 1, 2, 3, and 4 lie along the directions 11', 22', 33', and 44', respectively. It shows that the use of the gradient vector can give minimum point faster. All the descent methods make use of the gradient vector to find the optimum point.

### 14.5.1 Steepest Descent (Cauchy) Method

The steepest descent method, also called the Cauchy method, was first put forward by Cauchy in 1847. In this method, we choose an initial trial point  $X_1$  and iteratively move along the steepest descent direction to

find the optimum point. Minimization occurs in the negative direction of the gradient vector. The steps in this method are as follows:

**Step 1:** Start with an arbitrary initial point  $X_1$ . Consider the iteration number be represented by  $i$ . Then set  $i = 1$ .

**Step 2:** Find the search direction  $S_i$  as given in Eq. (14.1).

$$S_i = -\nabla f_i = -\nabla f(X_i) \quad (14.1)$$

**Step 3:** Find the optimal step length  $\lambda_i^*$  in the direction  $S_i$  as given in Eq. (14.2).

$$X_{i+1} = X_i + \lambda_i^* S_i = X_i - \lambda_i^* \nabla f_i \quad (14.2)$$

**Step 4:** Test the new point,  $X_{i+1}$ , for optimality condition whose derivative equals to zero. In other words, check if the new point is lesser than the previous one. If it is lesser, then the direction of movement is ideal and  $X_{i+1}$  becomes the optimum point. If  $X_{i+1}$  is the optimum point then stop the process, otherwise go to step 5.

**Step 5:** Set the new iteration number  $i = i + 1$  and go to step 2.

### Solved Problem 14.1

Minimize  $f(x_1, x_2) = 4x_1 - 2x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$  starting from point  $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

#### Solution:

**Step 1:** We have

$$f(x_1, x_2) = 4x_1 - 2x_2 + 2x_1^2 + 2x_1x_2 + x_2^2.$$

The gradient of  $f$  is given by

$$\nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix} = \begin{Bmatrix} 4 + 4x_1 + 2x_2 \\ -2 + 2x_1 + 2x_2 \end{Bmatrix}$$

$$\nabla f_1 = \nabla f(X_1) = \begin{Bmatrix} 4 \\ -2 \end{Bmatrix}$$

**Step 2:** We have

$$S_i = -\nabla f_i = -\nabla f(X_i)$$

Hence,

$$S_1 = -\nabla f_1 = \begin{Bmatrix} -4 \\ 2 \end{Bmatrix}$$

**Step 3:** Determine the optimal step length  $\lambda_1^*$  in the direction  $S_1$ . To find  $X_2$ , we need to find the optimal step length  $\lambda_1$ . For this, we minimize  $f(X_1 + \lambda_1 S_1)$  with respect to  $\lambda_1$ :

$$f(X_1 + \lambda_1 S_1) = f(-\lambda_1, \lambda_1) = \lambda_1^2 - 2\lambda_1$$

$$= X_1 + \lambda_1 S_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} -4 \\ 2 \end{bmatrix}$$

We have  $f(x_1, x_2) = 4x_1 - 2x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

Hence

$$f(\lambda_1, \lambda_1) = -4\lambda_1 - 2\lambda_1 + 2\lambda_1^2 - 2\lambda_1 \lambda_1 + \lambda_1^2 = \lambda_1^2 - 6\lambda_1 \quad (14.3)$$

Substituting  $\frac{\partial f}{\partial \lambda_1} = 0$  in Eq. (14.3), we get

$$\frac{\partial}{\partial \lambda_1} (\lambda_1^2 - 6\lambda_1) = 2\lambda_1 - 6 = 0$$

$$\lambda_1 = 3$$

Now

$$X_2 = X_1 + \lambda_1^* S_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} -4 \\ 2 \end{bmatrix} = \begin{bmatrix} -12 \\ 6 \end{bmatrix}$$

As  $\nabla f_2 = \nabla f(X_2) \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ,  $X_2$  is not optimum

If you perform many iterations, you will get the optimal solution.

### 14.5.2 Newton's Method

In Newton's method, the quadratic approximation of the function  $f(X)$  at  $X=X_i$  is considered in the Taylor's series expansion as

$$f(X) = f(X_i) + \nabla f_i^T (X - X_i) + \frac{1}{2} (X - X_i)^T [J_i] (X - X_i) \quad (14.4)$$

where  $[J_i] = [J] | X_i$  is the matrix of second partial derivatives (Hessian matrix) of  $f(X)$  evaluated at the point  $X_i$ . By setting the partial derivatives to 0, we obtain

$$\frac{\partial f(X)}{\partial x_j} = 0$$

where  $j = 1, 2, \dots, n$ .

Equation (14.4) reduces to Eq. (14.5).

$$\nabla f = \nabla f_i + [J_i](X - X_i) = 0 \quad (14.5)$$

If  $[J_i]$  is nonsingular, the above equation solution can be obtained as given in Eq. (14.6).

$$X_{i+1} = X_i - [J_i]^{-1} \nabla f_i \quad (14.6)$$

Since higher-order terms of the Taylor series are not considered, the Eq. (14.6) is used iteratively to find the optimum solution  $X^*$ .

**Solved Problem 14.2**

Minimize  $f(x_1, x_2) = 2x_1 - 2x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$  starting from the point  $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

**Solution:**

We have

$$X_{i+1} = X_i - [J_i]^{-1} \nabla f_i \text{ and } \nabla f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix} = \begin{Bmatrix} 2 + 4x_1 + 2x_2 \\ -2 + 2x_1 + 2x_2 \end{Bmatrix}$$

Hence

$$[J_1] = \begin{Bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{Bmatrix}_{X_1} = \begin{Bmatrix} 4 & 2 \\ 2 & 2 \end{Bmatrix}$$

$$[J_1]^{-1} = \frac{1}{4} \begin{Bmatrix} +2 & -2 \\ -2 & 4 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{Bmatrix}$$

Now

$$\nabla f_1 = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix}_{X_1} = \begin{Bmatrix} 2 + 4x_1 + 2x_2 \\ -2 + 2x_1 + 2x_2 \end{Bmatrix}_{(0,0)} = \begin{Bmatrix} 2 \\ -2 \end{Bmatrix}$$

Thus

$$X_2 = X_1 - [J_1]^{-1} g_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} - \begin{Bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{Bmatrix} \begin{Bmatrix} 2 \\ -2 \end{Bmatrix} = \begin{Bmatrix} 2 \\ 3 \end{Bmatrix}$$

To check if  $X_2$  is the optimum point, we evaluate

$$\nabla f_2 = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix}_{X_2} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

Thus,  $X_2$  is the optimum point.

## 14.6 Introduction to Derivative-Free Optimization

The four of the main derivative free optimization methods are genetic algorithms, simulated annealing, random search method, and downhill simplex search. They are extensively used for both continuous and discrete optimization problems. Among the different derivative free optimization methods we will consider the following methodologies.

1. Random search method.
2. Downhill simplex method.
3. Genetic algorithm.
4. Simulated annealing.

### 14.6.1 Random Search Method

The random search method is for continuous optimization problems. Let  $f$  be the fitness or cost function which must be minimized and  $x$  be the candidate solution in the search space. The basic steps of this algorithm are:

**Step 1:** Initialize  $x$  with a random position in the search space.

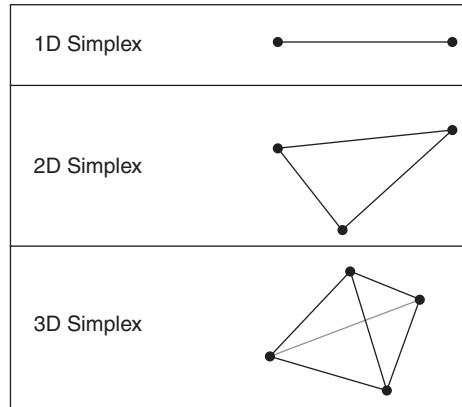
**Step 2:** Until a termination criterion is met (for example, number of iterations performed or adequate fitness reached), repeat the following:

1. Sample a new position  $y$  from the search space surrounding the current position  $x$ .
2. If  $f(y) < f(x)$  then move to the new position by setting  $x = y$ .

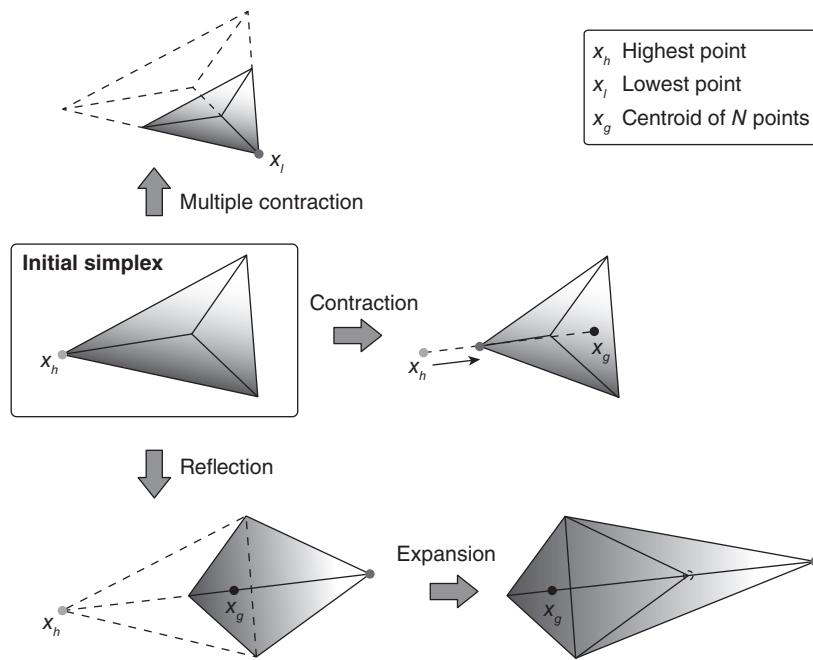
### 14.6.2 Downhill Simplex Method

The downhill simplex method is a multidimensional optimization method which uses geometric relationships to aid in finding function minimums. One distinct advantage of this method is that it does not require taking the derivative of the function being evaluated. Instead it creates its own pseudo-derivative by evaluating enough points to define a derivative for each independent variable of the function.

The method revolves around a simplex; a geometric object which contains  $n + 1$  vertex where  $n$  is the number of independent variables. For instance, in the case of a one-dimensional optimization the simplex would contain two points and represent a line segment. In a two-dimensional optimization, a three-pointed triangle would be used. This is shown in Fig. 14.3.



**Figure 14.3** Downhill simplex types.



**Figure 14.4** Downhill simplex method with contraction, reflection, expansion, and multiple contraction.

The downhill simplex method is an optimization algorithm that requires only function evaluations, and not calculation of derivatives. In the  $N$ -dimensional space, a simplex is a polyhedron with  $N + 1$  vertex. We chose the  $N + 1$  point and defined an initial simplex. The method iteratively updates the worst point by four operations: reflection, expansion, one-dimensional contraction, and multiple contractions. This is illustrated in Fig. 14.4.

Reflection involves moving the worst point (vertices) of the simplex (where the value of the objective function is the highest) to a point reflected through the remaining  $N$  points. If this point is better than the best point, then the method attempts to expand the simplex along this line. This operation is called *expansion*.

On the other hand, if the new point is not much better than the previous point, then the simplex is contracted along one dimension from the highest point. This procedure is called *contraction*. Moreover, if the new point is worse than the previous points, the simplex is contracted along all dimensions toward the best point and steps down the valley. By repeating this series of operations, the method finds the optimal solution.

#### 14.6.2.1 Algorithm

The downhill simplex method repeats the following steps:

1. Order and relabel the  $N + 1$  points so that  $f(x_{N+1}) > \dots > f(x_2) > f(x_1)$
2. Generate a trial point  $x_r$  by reflection, such that  $x_r = x_g + \alpha \times (x_g - x_{N+1})$ , where  $x_g$  is the centroid of the  $N$  best points in the vertices of the simplex
3. If  $f(x_1) < f(x_r) < f(x_N)$ , replace  $x_{N+1}$  by  $x_r$
4. If  $f(x_r) < f(x_1)$ , generate a new point  $x_e$  by expansion, such that

$$x_e = x_r + \beta \times (x_r - x_g)$$

5. If  $f(x_c) < f(x_r)$ , replace  $x_{N+1}$  by  $x_c$ , otherwise replace  $x_{N+1}$  by  $x_r$
6. If  $f(x_r) > f(x_N)$ , generate a new point  $x_c$  by contraction, such that

$$x_c = x_g + \gamma \times (x_{N+1} - x_g)$$

7. If  $f(x_c) < f(x_{N+1})$ , replace  $x_{N+1}$  by  $x_c$
8. If  $f(x_c) > f(x_{N+1})$ , contract along all dimensions toward  $x_i$ . Evaluate the objective function values at the  $N$  new vertices,

$$x_i = x_i + \eta \times (x_i - x_1)$$

### 14.6.3 Genetic Algorithm

The steps for the genetic algorithm are:

**Step 1:** Initialize a population with randomly generated individuals and evaluate the fitness value of each individual.

**Step 2:**

1. Select two members from the population with probabilities proportional to their fitness values.
2. Apply crossover with a probability equal to the crossover rate.
3. Apply mutation with a probability equal to the mutation rate.
4. Repeat steps 1 to 4 until enough members are generated to form the next generation.

**Step 3:** Repeat step 2 until a stopping criterion is met.

### 14.6.4 Simulated Annealing

Simulated annealing is another derivative-free optimization method suitable for continuous as well as discrete optimization problems. The principle behind this method is analogous to what happens when metals are cooled after being heated in a controlled way. The slowly falling temperature ends up in a regular crystalline structure. But if the temperature goes down too quickly, it results in an amorphous material. The most important part of this method is the annealing schedule, which specifies how rapidly the temperature is lowered from high to low values. The schedule of annealing is application specific and requires some trial-and-error procedures.

#### 14.6.4.1 Terminologies Used in Simulated Annealing

Following are some of the terminologies used in simulated annealing.

1. **Objective function:** An objective function  $f(x)$  maps an input vector  $x$  into a scalar  $E$ :  $E = f(x)$ . Each value of  $x$  is viewed as a point in an input space. The task of simulated annealing is to consider elements from the input space effectively to find an  $x$  that minimizes  $E$ .
2. **Generating function:** A generating function  $g$  finds out the probability density function of the difference between the current point and the next point to be considered.  $\Delta x (= x_{\text{new}} - x)$  is selected randomly with probability density function  $g(\Delta x, T)$ , where  $T$  is the temperature.
3. **Acceptance function:** After a new point  $x_{\text{new}}$  is found and objective function is evaluated, simulated annealing decides whether to accept or reject the new point based on the evaluation of the acceptance function  $h$ . The frequently used acceptance function is the Boltzmann probability distribution, which is given by

$$h(\Delta E, T) = \frac{1}{1 + \exp(\Delta E / cT)} \quad (14.7)$$

where  $c$  is a constant,  $T$  is the temperature, and  $\Delta E$  is the energy difference between  $x_{\text{new}}$  and  $x$ .

$$\Delta E = f(x_{\text{new}}) - f(x)$$

The general way is to accept  $x_{\text{new}}$  with some probability  $h(\Delta E, T)$ . When  $\Delta E$  is negative, the method tends to accept the new point because it reduces the energy. When  $\Delta E$  is positive, the method may accept the new point and end up in a higher energy state. In other words, simulated annealing can go either uphill or downhill. However, the lower the temperature, the less likely it is to accept any significant uphill actions.

- 4. Annealing schedule:** An annealing schedule regulates how rapidly the temperature varies as a function of time or iteration counts. One of the methodology to set annealing schedule is to decrease the temperature  $T$  by a certain percentage at each iteration.

Now that we understand the various terminologies, let us look at the basic steps involved in a general simulated annealing method.

**Step 1:** Choose a start point  $x$  and set a randomly high starting temperature  $T$ . Set the iteration count to 1.

**Step 2:** Evaluate the objective function  $E = f(x)$ .

**Step 3:** Select  $\Delta x$  with probability obtained from the generating function  $g(\Delta x, T)$ . Set the new point  $x_{\text{new}} = x + \Delta x$ .

**Step 4:** Evaluate the new value's objective function:

$$E_{\text{new}} = f(x_{\text{new}})$$

**Step 5:** Set  $x$  to  $x_{\text{new}}$  and  $E$  to  $E_{\text{new}}$  with probability obtained from the acceptance function  $h(\Delta E, T)$ , where  $\Delta E = E_{\text{new}} - E$ .

**Step 6:** Decrease the temperature  $T$  with respect to the annealing schedule by setting  $T$  equal to  $\eta T$ , where  $\eta$  is a constant between 0 and 1.

**Step 7:** Increase the iteration count  $k$  by 1. If  $k$  reaches the maximum iteration count then stop the iteration, otherwise go back to step 3.

In a general case, the generating function is a Gaussian probability density function, which is given by

$$g(\Delta x, T) = (2\pi T)^{-n/2} \exp\left[-\|\Delta x\|^2/(2T)\right] \quad (14.8)$$

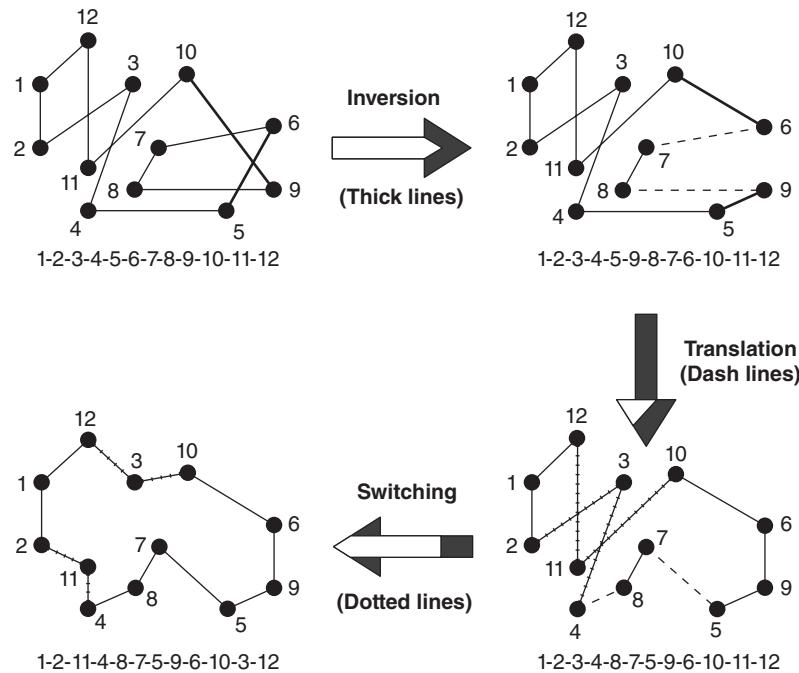
where  $\Delta x (= x_{\text{new}} - x)$  is the change in the  $x$  vector space,  $T$  is the temperature, and  $n$  is the dimension of the space under exploration. It has been proved that generating function  $g$  can find a global optimum of  $f(x)$  if the temperature  $T$  is reduced not faster than  $T_0/\ln k$ .

For discrete or combinatorial optimization problems, each  $x$  is not necessarily an  $n$ -element vector with unconstrained values. Instead, each  $x$  is confined to be one of the  $N$  points that constitute the solution space or input space. Usually  $N$  is finite, but it is very large such that the use of an exhaustive search method is impossible. Thus, adding randomly generated  $\Delta x$  to a current point  $x$  may not generate another legal point in the solution space, so generating functions are rarely used. Instead, to find the next legal point to explore, we usually define a move set, denoted by  $M(x)$ , as the set of legal points available for exploration after  $x$ . Usually, the move set  $M(x)$  represents a set of neighboring points of the current point  $x$  in the sense that the objective function at any point of the move set will not differ too much from the objective function

at  $x$ . The definition of a move set is problem-dependent and reflects our knowledge about the problem under consideration. Once the move set is defined,  $x_{\text{new}}$  is usually selected at random from this set, where every member stands an equal probability of being chosen.

A well-known instance of combinatorial optimization is the traveling salesperson problem, which is tackled using the simulated annealing technique as shown in Fig. 14.5. For a common traveling salesperson problem, we can define at least three move sets for simulated annealing.

1. **Inversion:** Remove two edges from the tour and replace them to make it another legal tour. This is equivalent to removing a section (6–7–8–9) of the tour and then replacing with the same cities running in the opposite order.
2. **Translation:** Remove a section (8–7) of the tour and then replace it in between two randomly selected consecutive cities (4 and 5).
3. **Switching:** Randomly select two cities (3 and 11) and switch them in a tour.



**Figure 14.5** Simulated annealing techniques for solving the traveling salesman problem.

## 14.7

## Derivative-Based vs Derivative-Free Optimization

While using optimization in problem solving, engineers can choose between derivative-based or derivative-free optimization. The choice is dependent on the inherent features of these two widely used optimization techniques as shown in Table 14.1.

**Table 14.1** Differences between derivative-based and derivative-free optimization

S. No.	<i>Derivative-Based Optimization</i>	<i>Derivative-Free Optimization</i>
1	Uses derivative information with objective function	Uses only objective function for evaluation
2	Extra information is required to find optimal solution	No extra information is required
3	Faster convergence	Slow convergence
4	Search direction is given by the derivative of the function	Search direction follows heuristic guidelines
5	Follows mathematical methodologies	Concepts are usually bio-inspired
6	Functions differentiable can only be considered here	Allows any objective function
7	Can converge to local optimum	Use random numbers to determine search directions leading to global optimum given enough computation time
8	Generic methodology is used	Methodologies used in dependent on problem domain
9	Convergence is dependent on derivative = 0 condition	Needs stopping criteria to determine when to terminate the optimization process
10	Heuristic search of solution is done	Randomized search of solution is done

## Summary

- Optimization is the act of obtaining the best results under given circumstances. The ultimate goal of optimization is either to minimize the effort required or to maximize the desired benefit.
- If the objective function and all the constraints are linear functions of the design variables, then the optimization problem is called linear programming problem.
- If any of the functions among the objective and constraint functions is nonlinear, the problem is called nonlinear programming problem.
- All the unconstrained minimization methods are iterative in nature. Thus, they start from an initial trial solution and proceed toward the minimum point.
- The use of the negative of the gradient vector as a direction for minimization was first made by Cauchy in 1847. In this method, we start from an initial trial point and iteratively move along the steepest descent directions until the optimum point is found.
- Derivative-free optimization is intuitive, slow, flexible, random, derivative free, and exhibits analytic opacity. They are iterative and arrive at approximate solutions that are acceptable.
- Evolutionary techniques like genetic algorithm, simulated annealing, downhill simplex, and random search methods are commonly used derivative-free optimization techniques.

## Multiple-Choice Questions

1. Which of the following is not required for using Newton's method for optimization?
  - (a) The lower bound for search region.
  - (b) Twice differentiable optimization function.
  - (c) The function to be optimized.
  - (d) A good initial estimate that is reasonably close to the optimal.
2. Which of the following statements are incorrect?
  - (a) If the second derivative at  $x_i$  is negative, then  $x_i$  is a maximum.
  - (b) If the first derivative at  $x_i$  is 0, then  $x_i$  is an optimum.
  - (c) If  $x_i$  is a minimum, then the second derivative at  $x_i$  is positive
  - (d) The value of the function can be positive or negative as any optima.
3. For what value of  $x$  is the function  $x^2 - 2x - 6$  minimized?
 

(a) 0	(b) 1
(c) 5	(d) 3
4. We need to enclose a field with a fence. We have 500 ft. of fencing material. There is a building on one side of the field where we will not need any fencing. Determine the maximum area of the field that can be enclosed by the fence. The field is rectangle in shape having length  $x$  and breadth  $y$ .
  - (a)  $x = 125$  ft.;  $y = 250$  ft.
  - (b)  $x = 150$  ft.;  $y = 200$  ft.
  - (c)  $x = 125$  ft.;  $y = 100$  ft.
  - (d)  $x = 200$  ft.;  $y = 150$  ft.
5. A rectangular box with a square base and no top has a surface area of  $108 \text{ ft}^2$ . Find the dimensions that will maximize the volume. Conduct two iterations using an initial guess of  $l = 3$  ft.
  - (a) Base edge length is 4.15 ft. and height is 4.85 ft.
  - (b) Base edge length is 6.15 ft. and height is 2.85 ft.
  - (c) Base edge length is 6.00 ft. and height is 3.00 ft.
  - (d) Base edge length is 3.85 ft. and height is 6.15 ft.

## Very Short Answer Questions

1. List some advantages of derivative-based optimization techniques.
2. List some disadvantages of derivative-based optimization techniques.
3. List some advantages of derivative-free optimization techniques.
4. List some disadvantages of derivative-free optimization techniques.
5. What is the principle behind simulated annealing?

## Short Answer Questions

1. List the steps in simulated annealing.
2. Explain the steps in genetic algorithm.
3. Explain the steps in downhill simplex method.
4. List and explain the steps in gradient descent algorithm.
5. Write a code in Python to find a local minimum of the function  $f(x) = x^4 - 3x^3 + 2$

## Review Questions

1. Define optimization. Explain the different types of optimization.
2. What is the steepest descent method? Explain with an example.
3. Explain Newton's method with the help of an example.
4. Write a short note on derivative-free optimization methods.
5. State the differences between derivative-based and derivative-free methods.

## Answers

---

### Multiple-Choice Questions

1. (a)    2. (c)    3. (b)    4. (a)    5. (c)

The steps for any machine learning algorithm using scikit-learn are as follows. This appendix gives a step-by-step approach for building a linear regression model.

### **Step 1: Import necessary functions**

```
Import matplotlib.pyplot as plt  
Import numpy as np  
From sklearn import datasets, linear_model  
From sklearn.metrics import mean_squared_error, r2_score
```

### **Step 2: Load the datasets**

```
diabetes = datasets.load_diabetes()
```

Diabetes dataset has 10 baseline variables – age, sex, body mass index, average blood pressure, and 6 blood serum measurements 442 diabetes patients. It also has the response of interest, which is a quantitative measure of disease progression one year after baseline. Each of these 10 feature variables have been mean centered and scaled by the standard deviation.

#### **Step 2.1: Select features for which you require the plotting**

```
diabetes_X = diabetes.data[:, np.newaxis, 2]  
The second feature is selected for plotting.
```

### **Step 3: Split the data into training data and testing data**

```
diabetes_X_train = diabetes_X[:-20]  
diabetes_X_test = diabetes_X[-20:]  
diabetes_y_train = diabetes.target[:-20]  
diabetes_y_test = diabetes.target[-20:]
```

In the above case 422 data samples are used for building the model or training the model and the remaining 20 data samples are used for testing the model. This can be any number. But the thumb rule which is followed is 2/3 data is used for building the model and 1/3 data is used for testing the model. The output target variables are also split according to the train and test data split.

### **Step 4: Create the linear regression model**

```
regr = linear_model.LinearRegression()  
Linear_model has a LinearRegression method.
```

**Step 5: Train the model using the training set**

```
regr.fit(diabetes_X_train, diabetes_y_train)
```

The fit method takes the <input,output> pair and tries to fit the linear regression line.

**Step 6: Make predictions on the testing set**

```
diabetes_y_pred = regr.predict(diabetes_X_test)
```

For predicting the diabetes\_x\_test, data is used and the predictions are made.

**Step 7: Display the regression coefficients**

```
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f"
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test,
                                         diabetes_y_pred))
```

The coefficients, Mean-Squared-Error and the Variance score are computed.

**Step 8: Plotting the outputs**

```
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```

While plotting the test data of input is plotted with the test data of output as a scatter plot. The linear regression line is plotted and displayed.

# Logistic Regression

Logistic regression is a supervised algorithm, and gives the probability with which the output belongs to a particular class. For executing logistic regression in Python, the sample dataset taken is iris dataset. Iris data is a dataset which classifies iris flowers into three categories (iris setosa, iris Virginia, and iris versicolor) based on the petal length, petal width, sepal length, and sepal width. So the steps in using logistic regression for its classification are as follows.

## **Step 1: Import the necessary functions**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
```

## **Step 2: Load iris dataset**

```
iris = datasets.load_iris()
```

## **Step 3: Use sepal length and sepal width attributes with the corresponding output class**

```
X = iris.data[:, :2]
Y = iris.target
```

## **Step 4: Fit the logistic regression function**

```
logreg = LogisticRegression(C=1e5, solver='lbfgs',
                            multi_class='multinomial')
logreg.fit(X, Y)
```

## **Step 5: Use logistic regression for fitting the data**

```
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = .02 # step size in the mesh
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
```

**Step 6: Display the output**

```
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = .02 # step size in the mesh
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))

z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])
# Put the result into a color plot
Z = z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k',
cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```

*Source:* sci-kit learn.org

*Note:* Iris dataset is available at <https://archive.ics.uci.edu/ml/datasets/iris>

# Support Vector Machines

Support vector machine (SVM) is a supervised classification learning algorithm. Using scikit-learn, we can import SVM and fit the model. The iris dataset is used for SVM model. This dataset was used in Appendix A. The description of the dataset is also explained in Appendix A.

## **Step 1: Import the necessary header functions**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
```

## **Step 2: Use iris dataset with training and test datasets**

```
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
```

## **Step 3: Data is fitted using SVM fit function**

```
C = 1.0
svc = svm.SVC(kernel='rbf', C=1, gamma='auto').fit(X, y)
```

## **Step 4: Plot the function**

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                      np.arange(y_min, y_max, h))
plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```

*Source:* sci-kit learn.org

*Note:* Iris dataset is available at <https://archive.ics.uci.edu/ml/datasets/iris>

# Naive Bayes Classifier

Bayesian classifier comes under the category of supervised learning algorithm, which uses the concept of probability. While implementing the naive Bayes classifier, an inbuilt Gaussian function is used in scikit-learn. The steps for building a Bayes classifier are as follows:

### **Step 1: Import the necessary libraries**

```
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd
```

### **Step 2: Import the dataset**

The dataset used is the iris dataset. This dataset has four input features – sepal length, sepal width, petal length, and petal width. It has one output that is the species of iris, which can be either iris setosa, iris Virginia, or iris versicolor.

```
filename="iris.csv"  
dataset=pd.read_csv(filename)  
x=dataset[['sepal_length','sepal_width','petal_length','petal_width']]  
y=dataset['species']
```

### **Step 3: Split the train and test data**

For any supervised learning algorithm, the dataset is split as training and testing datasets. The model is built using the training samples and tested with the testing samples.

```
from sklearn.model_selection import train_test_split  
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.35)
```

(**Note:** 0.35 implies that 35% data is set aside for testing.)

### **Step 4: Perform feature scaling**

```
from sklearn.preprocessing import StandardScaler  
sc=StandardScaler()  
xtrain=sc.fit_transform(xtrain)  
xtest=sc.transform(xtest)
```

(**Note:** Feature scaling is a method used to standardize the range of independent variables or features of data. It is also known as data normalization and is generally performed during the data preprocessing step.)

**Step 5: Fit the Gaussian Naive Bayes**

```
from sklearn.naive_bayes import GaussianNB  
gmodel=GaussianNB()  
gmodel.fit(xtrain,ytrain)
```

(**Note:** Gaussian naive Bayes is the most popular naive Bayesian classifier. Bernoulli's classifier and multinomial classifiers are the other types of classifiers provided by scikit-learn)

**Step 6: Predict the result and arrive at the confusion matrix**

```
gpred=gmodel.predict(xtest)  
from sklearn.metrics import confusion_matrix  
gcm=confusion_matrix(ytest,gpred)  
print(gcm)
```

**Step 7: Compute the accuracy score**

```
from sklearn.metrics import accuracy_score  
accuracy_score(ytest,gpred)
```

**Step 8: Arrive at the probability of an unknown sample**

```
print("Probabilities of falling into each class")  
print(gmodel.predict_proba([[5,3,1.2,2]]))  
print("Input is classified into")  
ans=gmodel.predict([[5,3,1.2,2]])  
print(ans)
```

*Note:* Iris dataset is available at <https://archive.ics.uci.edu/ml/datasets/iris>

# Clustering Using $k$ -Means

The  $k$ -means algorithm is a simple unsupervised learning algorithm that requires only three lines of code in scikit-learn for its implementation. First, we import the necessary libraries.

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from sklearn.cluster import KMeans
```

As an example, let us enter  $X$  which is a 2D matrix.

```
X = np.array([[5,3],
[10,15],
[15,12],
[24,10],
[30,45],
[85,70],
[71,80],
[60,78],
[55,52],
[80,91],])
```

Plot the scatter plots of the points.

```
plt.scatter(X[:,0],X[:,1], label='True Position')
```

Use  $k$ -means and fit it into two clusters.

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
```

The cluster centers are then printed.

```
print(kmeans.cluster_centers_)
```

Now, print the labels of the data points and label the clusters.

```
print(kmeans.labels_)
```

Use scatter plots and plot the points with different colors for different classes.

```
plt.scatter(X[:,0],X[:,1], c=kmeans.labels_, cmap='rainbow')
```

When the number of clusters formed is to be three, there is only a slight change in the code.

```
kmeans.fit(X)
```

- Printing the cluster centers

```
print(kmeans.cluster_centers_)
```

- Labeling the cluster centers.

```
print(kmeans.labels_)
```

- Plotting the data points into three different clusters.

```
plt.scatter(X[:,0],X[:,1], c=kmeans.labels_, cmap='rainbow')
```

Represent the different data points along with the different cluster centers.

```
plt.scatter(X[:,0],X[:,1], c = kmeans.labels_, cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,0], color='black')
```

# Principal Component Analysis

Principal component analysis is a dimensionality reduction technique used for capturing principal components from a larger dataset. It is useful for measuring data in principal components rather than in normal  $x$ - and  $y$ -axis.

## Step 1: Import the necessary libraries

```
import numpy as np  
from numpy import linalg as la
```

## Step 2: Give the input dataset. For convenience, $x$ and $y$ are inputted as an array of 10 values. The mean of the values are also computed.

```
x = np.array([2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2., 1., 1.5, 1.1])  
y = np.array([2.4, 0.7, 2.9, 2.2, 3, 2.7, 1.6, 1.1, 1.6, 0.9])  
data = np.array([x, y])  
print(x)  
print(y)  
print(data)
```

```
[ 2.5  0.5  2.2  1.9  3.1  2.3  2.   1.   1.5  1.1]  
[ 2.4  0.7  2.9  2.2  3.   2.7  1.6  1.1  1.6  0.9]  
[[2.5  0.5  2.2  1.9  3.1  2.3  2.   1.   1.5  1.1]  
 [2.4  0.7  2.9  2.2  3.   2.7  1.6  1.1  1.6  0.9]]
```

```
xMean = np.mean(x)  
yMean = np.mean(y)  
print(xMean)  
print(yMean)
```

```
1.81  
1.91
```

```
data.shape
```

```
(2L, 10L)
```

**Step 3: Compute the mean adjusted values by subtracting each point from its mean. This helps in normalizing the values.**

```
meanAdjusted = np.zeros((2, 10))
for i in range(len(data[0])):
    meanAdjusted[0][i] = data[0][i] - xMean
for i in range(len(data[1])):
    meanAdjusted[1][i] = data[1][i] - yMean
print(meanAdjusted)
```

```
[[ 0.69 -1.31  0.39  0.09  1.29  0.49  0.19 -0.81 -0.31 -0.71]
 [ 0.49 -1.21  0.99  0.29  1.09  0.79 -0.31 -0.81 -0.31 -1.01]]
```

**Step 4: Compute the covariance matrix of the mean adjusted data.**

```
cov_mat = np.cov(data)
```

**Step 5: Having computed the covariance matrix, compute the eigenvalues and eigenvectors. The advantage of Python is that this cumbersome process is computed in a single line of code.**

```
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors \n%s' %eig_vecs)
print('\nEigenvalues \n%s' %eig_vals)

Eigenvectors
[[-0.73517866 -0.6778734 ]
 [ 0.6778734  -0.73517866]]

Eigenvalues
[ 0.0490834   1.28402771]
```

**Step 6: Arrange the eigenvalues in descending order. Also, print the eigenvectors in descending order.**

```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in
range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs.sort()
eig_pairs.reverse()

# Visually confirm that the list is correctly sorted by decreasing
eigenvalues
print('Eigenvalues in descending order:')
for i in eig_pairs:
    print(i[0])
```

Eigenvalues in descending order:

```
1.28402771217  
0.0490833989383
```

```
print('Eigenvectors in descending order:')
```

```
for i in eig_pairs:  
    print(i[1])
```

Eigenvectors in descending order:

```
[-0.6778734 -0.73517866]  
[-0.73517866 0.6778734 ]
```

```
eig_pairs[0][1]
```

```
array([-0.6778734 , -0.73517866])
```

**Step 7: Retaining only those eigenvalues having maximum values, transform the data and display it.**

```
transformedData1 = np.matmul(meanAdjusted.T, eig_pairs[0][1])  
transformedData1  
  
array([-0.82797019, 1.77758033, -0.99219749, -0.27421042, -1.67580142,  
      -0.9129491, 0.09910944, 1.14457216, 0.43804614, 1.22382056])  
  
transformedData2 = np.matmul(meanAdjusted.T, eig_pairs[1][1])  
transformedData2  
  
array([-0.17511531, 0.14285723, 0.38437499, 0.13041721, -0.20949846,  
      0.17528244, -0.3498247, 0.04641726, 0.01776463, -0.16267529])  
  
transformedData =[transformedData1,transformedData2]  
transformedData =np.transpose(transformedData)  
transformedData  
  
array([[ -0.82797019, -0.17511531],  
       [ 1.77758033,  0.14285723],  
       [-0.99219749,  0.38437499],  
       [-0.27421042,  0.13041721],  
       [-1.67580142, -0.20949846],  
       [-0.9129491 ,  0.17528244],  
       [ 0.09910944, -0.3498247 ],  
       [ 1.14457216,  0.04641726],  
       [ 0.43804614,  0.01776463],  
       [ 1.22382056, -0.16267529]])
```

```
matrix_w = np.hstack((eig_pairs[0][1].reshape(2,1),
eig_pairs[1][1].reshape(2,1)))
matrix_w

array([[-0.6778734 , -0.73517866],
       [-0.73517866,  0.6778734 ]])
```

**Step 8: Reconstruct and transform the original data.**

```
# Reconstruction
originalData = np.matmul(transformedData, matrix_w)
originalData[:, :] = originalData[:, :] + [xMean, yMean]
originalData

array([[ 2.5,   2.4],
       [ 0.5,   0.7],
       [ 2.2,   2.9],
       [ 1.9,   2.2],
       [ 3.1,   3. ],
       [ 2.3,   2.7],
       [ 2. ,   1.6],
       [ 1. ,   1.1],
       [ 1.5,   1.6],
       [ 1.1,   0.9]])
```

# Decision Tree

A decision tree is a simple supervised learning algorithm. The steps for implementing the decision tree algorithm in scikit-learn is given below:

**Step 1: Import the necessary header files. In case of a decision tree algorithm, we need to import the decision tree classifier from sklearn. The iris dataset is used in building the decision tree.**

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

**Step 2: Iris dataset is used and its split into training and testing dataset.**

```
# Generate the iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
random_state=42)
```

**Step 3: Build the DecisionTreeClassifier model.**

```
# Build a model using the default setting of fully developing the
tree
tree = DecisionTreeClassifier(random_state=0)
```

**Step 4: Fit the classifier on the training dataset.**

```
# Fit the classifier on training set
tree.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=0, splitter='best')
```

**Step 5: Make the predictions on test set.**

```
predictions = tree.predict(X_test)
print("Test set predictions: {}".format(predictions))
```

```
Test set predictions: [0 1 1 1 0 1 2 2 2 2 1 2 1 1 0 0 0 0 1 0 1 2 1 1
1 2 1 0 2 0 1 2 2 0 0 0 0 2 1]
```

**Step 6: Evaluate the model.**

```
accuracy = tree.score(X_test, y_test)
print("Test set accuracy: {:.2f}".format(accuracy))
```

```
Test set accuracy: 0.95
```

For any supervised learning algorithm, these six steps are followed to train the data and test the model with the testing dataset.

*Note:* Iris dataset is available at <https://archive.ics.uci.edu/ml/datasets/iris>



# Bibliography

1. Han, J. and Kamber, M. (2000) *Data Mining: Concepts and Techniques*, McGraw-Hill Inc., New York.
2. Harrington, P. (2012) *Machine Learning in Action*, Manning Publications, New York.
3. Hartman, G. (2010) Fundamentals of Matrix Algebra. CreateSpace Independent Publishing Platform
4. Marsland, S. (2011) *Machine Learning: An Algorithmic Perspective*, CRC Press, Florida.
5. Mitchell, T. (1997) *Machine Learning*, McGraw-Hill Inc., New York.
6. Simonoff, J. S. (1996) *Smoothing Methods in Statistics*, Springer, New York.
7. Sivanandam, S. N. and Deepa, S. N. (2008) *Principles of Soft Computing*, Wiley, Delhi.
8. [https://in.mathworks.com/campaigns/products/display/machine-learning-with-matlab-conf.html?elq\\_sid=1489063116163&potential\\_use=Education](https://in.mathworks.com/campaigns/products/display/machine-learning-with-matlab-conf.html?elq_sid=1489063116163&potential_use=Education)
9. <http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning>
10. <https://quantdare.com/machine-learning-a-brief-breakdown/>
11. <https://ipullrank.com/clustering-vs-classification-speed-keyword-research/>
12. <https://d4datascience.wordpress.com/>
13. <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>
14. <https://machinelearningmastery.com/practical-machine-learning-problems/>
15. <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>
16. <https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-choice>
17. [https://saylordotorg.github.io/text\\_introduction-to-psychology/s11-01-learning-by-association-classi.html](https://saylordotorg.github.io/text_introduction-to-psychology/s11-01-learning-by-association-classi.html)
18. [https://docs.oracle.com/cd/B28359\\_01/datamine.111/b28129/classify.htm#DMCON004](https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#DMCON004)
19. [https://www.tutorialspoint.com/data\\_mining/dm\\_classification\\_prediction.htm](https://www.tutorialspoint.com/data_mining/dm_classification_prediction.htm)
20. <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
21. [https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set)
22. <https://in.mathworks.com/help/vision/examples/digit-classification-using-hog-features.html>
23. [https://scikit-learn.org/stable/auto\\_examples/index.html](https://scikit-learn.org/stable/auto_examples/index.html)
24. <http://www.maths.manchester.ac.uk/~kd/ma2m1/matrices.pdf>
25. [https://math.dartmouth.edu/archive/m16s07/public\\_html/matrices](https://math.dartmouth.edu/archive/m16s07/public_html/matrices)
26. <http://www.ijsr.com/articles/IJSRDV3I31577.pdf>
27. [https://pandas.pydata.org/pandas-docs/stable/missing\\_data.html](https://pandas.pydata.org/pandas-docs/stable/missing_data.html)
28. [https://www.tutorialspoint.com/python\\_pandas/python\\_pandas\\_missing\\_data.html](https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.html)
29. [https://in.mathworks.com/help/matlab/data\\_analysis/data-smoothing-and-outlier-detection.html](https://in.mathworks.com/help/matlab/data_analysis/data-smoothing-and-outlier-detection.html)
30. <https://t4tutorials.com/binning-methods-for-data-smoothing-in-data-mining/>
31. [http://mercury.webster.edu/aleshunas/Support%20Materials/Data\\_preprocessing.pdf](http://mercury.webster.edu/aleshunas/Support%20Materials/Data_preprocessing.pdf)
32. <https://people.revoledu.com/kardi/tutorial/LDA/Numerical%20Example.html>
33. <https://onlinecourses.science.psu.edu/stat505/node/49/>
34. <https://onlinecourses.science.psu.edu/stat505/print/book/export/html/49/>
35. <https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc552.htm>
36. [http://www.cs.otago.ac.nz/cosc453/student\\_tutorials/principal\\_components.pdf](http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf)

37. [https://www.cgg.com/technicaldocuments/cggy\\_0000014063.pdf](https://www.cgg.com/technicaldocuments/cggy_0000014063.pdf)
38. <https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial>
39. [https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition\\_jp.pdf](https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf)
40. [https://www.math.uwaterloo.ca/~aghodsib/courses/f06stat890/readings/tutorial\\_stat890.pdf](https://www.math.uwaterloo.ca/~aghodsib/courses/f06stat890/readings/tutorial_stat890.pdf)
41. <https://www.thecalculator.co/math/Covariance-Calculator-705.html>
42. <https://www.coursera.org/lecture/machine-learning/principal-component-analysis-algorithm-ZYIPa>
43. <https://onlinecourses.science.psu.edu/stat505/node/54/>
44. <https://coolstatsblog.com/2015/03/21/principal-component-analysis-explained/>
45. [https://www.apsl.net/blog/2017/06/21/using-principal-component-analysis-pac-data-explore-step-step/](https://www.apsl.net/blog/2017/06/21/using-principal-component-analysis-pca-data-explore-step-step/)
46. <https://people.revoledu.com/kardi/tutorial/LDA/Numerical%20Example.html>
47. <https://research.ics.aalto.fi/ica/icademo/>
48. <http://wwwf.imperial.ac.uk/~nsjones/TalkSlides/HyvarinenSlides.pdf>
49. [http://arnauddelorme.com/ica\\_for\\_dummies/](http://arnauddelorme.com/ica_for_dummies/)
50. <https://www.cs.helsinki.fi/u/ahyvarin/papers/NN00new.pdf>
51. <http://www.cs.jhu.edu/~ayuille/courses/Stat161-261-Spring14/HyvO00-icatut.pdf>
52. [http://cis.legacy.ics.tkk.fi/aapo/papers/IJCNN99\\_tutorialweb/](http://cis.legacy.ics.tkk.fi/aapo/papers/IJCNN99_tutorialweb/)
53. <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>
54. <https://developer.ibm.com/articles/l-neural/>
55. <https://www.willamette.edu/~gorr/classes/cs449/motivate.htm>
56. <https://www.explainthatstuff.com/introduction-to-neural-networks.html>
57. [http://neuron.eng.wayne.edu/tarek/MITbook/t\\_contents.html](http://neuron.eng.wayne.edu/tarek/MITbook/t_contents.html)
58. <http://onlinestatbook.com/2/regression/accuracy.html>
59. <https://www.hackerearth.com/practice/machine-learning/linear-regression/univariate-linear-regression/tutorial/>
60. <http://www.statgraphics.com/regression-analysis#Simple%20Regression>
61. [https://www.tutorialspoint.com/r/r\\_linear\\_regression.htm](https://www.tutorialspoint.com/r/r_linear_regression.htm)
62. <https://machinelearningmastery.com/simple-linear-regression-tutorial-for-machine-learning/>
63. <https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06>
64. <https://machinelearningmastery.com/linear-regression-for-machine-learning/>
65. <https://www.dezyre.com/data-science-in-r-programming-tutorial/linear-regression-tutorial>
66. <https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/regression-analysis/>
67. <http://people.duke.edu/~rnau/regintro.htm>
68. [http://www.analyzemath.com/statistics/linear\\_regression.html](http://www.analyzemath.com/statistics/linear_regression.html)
69. <http://www.massey.ac.nz/~mbjones/Book/Chap9.pdf>
70. <https://web.csulb.edu/~msaintg/ppa696/696regs.htm>
71. <http://www.stat.wmich.edu/wang/160/egs/slر.pdf>
72. [https://www.medcalc.org/manual/logistic\\_regression.php](https://www.medcalc.org/manual/logistic_regression.php)
73. <http://faculty.cas.usf.edu/mbrannick/regression/Logistic.html>
74. <http://r-statistics.co/Logistic-Regression-With-R.html>
75. <http://ufdl.stanford.edu/tutorial/supervised/LogisticRegression/>
76. <https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/logistic-regression-analysis-r/tutorial/>
77. [https://gerardnico.com/data\\_mining/simple\\_logistic\\_regression](https://gerardnico.com/data_mining/simple_logistic_regression)
78. <https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/>
79. [https://www.mc.vanderbilt.edu/gcrc/workshop\\_files/2004-11-12.pdf](https://www.mc.vanderbilt.edu/gcrc/workshop_files/2004-11-12.pdf)
80. <https://datascienceplus.com/building-a-logistic-regression-in-python-step-by-step/>

81. <https://www.solver.com/logistic-regression-example>
82. <http://data.princeton.edu/wws509/notes/c3.pdf>
83. <https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/>
84. <https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8>
85. <https://www.analyticsvidhya.com/blog/2015/10/basics-logistic-regression/>
86. <https://onlinecourses.science.psu.edu/stat501/node/379/>
87. <https://stats.idre.ucla.edu/stata/dae/logistic-regression/>
88. <https://www.geeksforgeeks.org/understanding-logistic-regression/>
89. <https://www.r-bloggers.com/machine-learning-ex4-logistic-regression-and-newtons-method/>
90. <https://www.dezyre.com/data-science-in-r-programming-tutorial/logistic-regression-tutorial>
91. <https://github.com/perborgen/LogisticRegression>
92. <https://machinelearningmastery.com/implement-logistic-regression-stochastic-gradient-descent-scratch-python/>
93. <http://aimotion.blogspot.com/2011/11/machine-learning-with-python-logistic.html>
94. [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)
95. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
96. <http://uregina.ca/~gingrich/ch10.pdf>
97. <https://onlinecourses.science.psu.edu/statprogram/reviews/statistical-concepts/chi-square-tests>
98. [https://medium.com/@rishabhjain\\_22692/decision-trees-it-begins-here-93ff54ef134](https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134)
99. <https://clearpredictions.com/Home/DecisionTree>
100. <https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart>
101. <https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/>
102. <https://data-flair.training/blogs/svm-support-vector-machine-tutorial/>
103. <https://www.csie.ntu.edu.tw/~cjlin/papers/nusvmtutorial.pdf>
104. <https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners>
105. <http://web.mit.edu/zoya/www/SVM.pdf>
106. <http://www.svms.org/tutorials/Berwick2003.pdf>
107. <https://machinelearningmastery.com/support-vector-machines-for-machine-learning/>
108. <https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf>
109. <https://core.ac.uk/download/pdf/6302770.pdf>
110. <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
111. <http://www.statsoft.com/Textbook/Support-Vector-Machines>
112. <http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf>
113. <http://axon.cs.byu.edu/Dan/678/miscellaneous/SVM.example.pdf>
114. <http://blog.aylien.com/support-vector-machines-for-dummies-a-simple/>
115. [https://www.csie.ntu.edu.tw/~cjlin/papers/bottou\\_lin.pdf](https://www.csie.ntu.edu.tw/~cjlin/papers/bottou_lin.pdf)
116. [https://cw.fel.cvut.cz/old/\\_media/courses/a4b33opt/lecture\\_svm.pdf](https://cw.fel.cvut.cz/old/_media/courses/a4b33opt/lecture_svm.pdf)
117. <http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/>
118. <https://towardsdatascience.com/radial-basis-functions-neural-networks-all-we-need-to-know-9a88cc053448>
119. <https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf>
120. <https://cs.fit.edu/~dmitra/SciComp/13Spr/KimJessie-rbf3.pdf>
121. <http://num.math.uni-goettingen.de/schaback/teaching/sc.pdf>
122. <http://www.cs.bham.ac.uk/~jxb/INC/15.pdf>
123. <https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d>
124. <https://www.coursera.org/lecture/probabilistic-graphical-models/independencies-in-bayesian-networks-JRkCU>

125. <https://www.slideshare.net/GiladBarkan/bayesian-belief-networks-for-dummies>
126. <http://robotics.stanford.edu/~koller/NIPStut01/tut6.pdf>
127. <https://people.cs.pitt.edu/~milos/courses/cs2001/cs2001-2.pdf>
128. <http://www.cs.cmu.edu/afs/cs/project/theo-20/www/mlbook/ch6.pdf>
129. <https://www.bayesserver.com/docs/walkthroughs/walkthrough-1-a-simple-network>
130. <https://www.bayesserver.com/docs/introduction/bayesian-networks>
131. <https://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf>
132. <https://link.springer.com/content/pdf/10.1023%2FA%3A1007465528199.pdf>
133. <https://dslpitt.org/uai/papers/11/p153-cussens.pdf>
134. <https://www.bayesserver.com/docs/introduction/dynamic-bayesian-networks>
135. [https://s3.amazonaws.com/bayesia.us/book/BayesiaLab\\_Book\\_V18.pdf?submissionGuid=b050868a-d04d-4d66-a25e-fb4484fa4fc2](https://s3.amazonaws.com/bayesia.us/book/BayesiaLab_Book_V18.pdf?submissionGuid=b050868a-d04d-4d66-a25e-fb4484fa4fc2)
136. <http://www.bayesia.com/book2>
137. <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
138. <https://www.geeksforgeeks.org/k-nearest-neighbours/>
139. <https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>
140. <https://scikit-learn.org/stable/modules/neighbors.html>
141. <https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7>
142. [http://research.cs.tamu.edu/prism/lectures/pr/pr\\_l23.pdf](http://research.cs.tamu.edu/prism/lectures/pr/pr_l23.pdf)
143. <http://cecas.clemson.edu/~ahoover/ece854/refs/Ramos-Intro-HMM.pdf>
144. [https://fenix.tecnico.ulisboa.pt/downloadFile/3779577326932/Modelos\\_prob\\_12.pdf](https://fenix.tecnico.ulisboa.pt/downloadFile/3779577326932/Modelos_prob_12.pdf)
145. <http://www.cs.cmu.edu/~guestrin/Class/10701/slides/hmms-structurelearn.pdf>
146. [http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/D2PAGES/TUTORIALS/hmm\\_isspr.pdf](http://homepages.inf.ed.ac.uk/rbf/IAPR/researchers/D2PAGES/TUTORIALS/hmm_isspr.pdf)
147. <https://codefying.com/2016/09/15/a-tutorial-on-hidden-markov-model-with-a-stock-price-example/>
148. [http://www.lwebzem.com/cgi-bin/courses/hidden\\_markov\\_model\\_online.cgi](http://www.lwebzem.com/cgi-bin/courses/hidden_markov_model_online.cgi)
149. <https://web.math.princeton.edu/~rvan/orf557/hmm080728.pdf>
150. [http://idiom.ucsd.edu/~rlevy/teaching/winter2009/ligncse256/lectures/hmm\\_viterbi\\_mini\\_example.pdf](http://idiom.ucsd.edu/~rlevy/teaching/winter2009/ligncse256/lectures/hmm_viterbi_mini_example.pdf)
151. <http://www.robots.ox.ac.uk:5000/~vgg/rg/papers/hmm.pdf>
152. <https://www2.spsc.tugraz.at/www-archive/AdvancedSignalProcessing/SS08-SpeechSynthesis/HMM-Basics-Report.pdf>
153. <http://firsttimeprogrammer.blogspot.com/2015/08/basic-hidden-markov-model.html>
154. [http://www.cs.columbia.edu/~smaskey/CS6998/slides/statnlp\\_week5.pdf](http://www.cs.columbia.edu/~smaskey/CS6998/slides/statnlp_week5.pdf)
155. <https://people.revoledu.com/kardi/tutorial/Clustering/Numerical%20Example.htm>
156. <http://benalexkeen.com/k-means-clustering-in-python/>
157. <http://mnemstudio.org/clustering-k-means-example-1.htm>
158. <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
159. <https://www.geeksforgeeks.org/k-nearest-neighbours/>
160. <https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>
161. <https://scikit-learn.org/stable/modules/neighbors.html>
162. <https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7>

# Index

## A

accuracy of hypothesis function, 21  
accuracy of prediction, 221  
activity recognition, 227  
agglomerative algorithm, 252, 253, 254, 259, 261  
Anaconda, 40–41  
    downloading and installing, 40  
    screen of, 41  
append function, 66  
approximate dynamic programming, 12  
artificial neural network (ANN), 105–106  
    network architecture, 108  
    uses of, 106  
artificial neurons, 107  
attribute selection, 158

## B

Bayesian belief networks, 214–216  
Bayes’ theorem, 211–212  
beam search, 175  
binary classification model, 140  
binary classification problem, 9  
biological neuron, 107–108  
bipolar activation functions, 112–113  
bipolar binary function, 112  
bipolar continuous function, 112  
blind source separation, 98–100

## C

cancer detection, 140  
Centrum Voor Wiskunde en informatica (CWI), Netherlands, 37  
class conditional independence, 211  
clustering, 239  
    applications of, 240  
    requirements of, 240–242  
    types of, 243–263  
column matrix, 28  
column vector, 27  
complete-linkage algorithm, 253  
computer algorithm, 10  
computer program, 3  
conda, 40  
confusion matrix, 221  
contours, 23  
control theory, 12  
cost function, 18, 20  
    minimization of, 21–24  
Cover’s theorem, 205  
credit card fraud detection, 140

## D

data cleaning  
    calculations with missing data, 72–80  
    filling in missing values, 72  
    removing inconsistencies, 80  
data integration, 80–81  
data preprocessing, 71  
data reduction, 81–100  
data transformation, 81  
decision list, 174  
decision tree algorithms, 157–158  
    classification using, 169  
    issues in, 169–173  
    problem solving using, 158  
    rule extraction from, 175  
    steps to construct, 162–163  
decision tree classifiers, 162  
default class, 174  
density-based clustering, 244  
derivative-based optimization, 281–282  
derivative free optimization, 277–281  
design vector, 272  
determinant of matrix, 32–33  
deterministic relationships, 124  
    between Fahrenheit and Celsius scales, 124  
diagonal matrix, 28  
digital marketing, 7  
dimensionality reduction. *see* data reduction  
dimensions of matrix, 27  
downhill simplex method, 277–278  
drop function, 67  
dynamics, 108  
dynamic type programming language, 38

## E

eigenvalues and eigenvectors, 90–92  
elbow method, 242  
entropy, 159

exhaustive rules, 174  
exponential relationship, 126

**F**

farthest-neighbor clustering algorithm, 253  
feature vector, 92  
feedback network, 109  
feed-forward network, 108–109  
fruit categorization, process of, 8–9  
F-score, 222

**G**

gain ratio, 159  
game theory, 12  
Gaussian distribution, 97  
genetic algorithm, 279  
Gini index, 160–161  
GNU General Public License (GPL), 37  
gradient-based methods (descent methods), 273–276  
gradient descent algorithm, 144  
gradient function, 24–25  
grid-based clustering, 244

**H**

hard clustering, 11  
hidden layer, 109  
Hidden Markov model (HMM), 227–230  
    decoding problem, 232–233  
    evaluation problem, 231–232  
    issues in, 230–231  
    learning problem, 233–234  
hierarchical clustering, 243, 252–263  
hyperplane, 182  
hyperplane, equation of, 192  
    biggest margin, 195–196  
    computation of distance from a point, 192–194  
    computation of margin, 194–195  
    maximization of distance between hyperplanes, 197–200  
    with no points between them, 196

**I**

identity function, 113  
IF-THEN rules, 157, 173–174  
independent component analysis (ICA), 97, 100  
information gain, 158–159  
information theory, 12  
Integrated Development Environment (IDE), 39  
Iterative Dichotomiser 3 (ID3) algorithm, 158, 162  
    limitations of, 162  
    stopping condition of, 162

**J**

Jupyter, 40  
Jupyter Notebook, 41–43

default browser page in, 42  
interactive programming in, 43  
launch page of, 41

**K**

Karl Pearson's coefficient of correlation, 130–131  
*k*-means clustering algorithm, 244–245  
*k*-medoids algorithm, 249–250  
*k*-nearest neighbor (KNN), 216–220  
kurtosis, 100

**L**

lateral feedback, 109  
learning by observation, 239  
learning process in supervised learning, 18  
learning rate, 25, 144  
learn-one-rule procedure, 176  
least squares estimators, 128  
likelihood function, 143  
linear discriminant analysis (LDA), 97  
linear programming (LP) problem, 272  
linear regression  
    requirement of, 126  
    steps, 126–129  
linear regression problem, 19–20  
logistic regression, 139–140  
    example, 144–153  
logistic regression function, 141–142  
logit function, 141  
log-likelihood equation, 143  
lower triangular matrix, 29

**M**

machine learning (ML)  
    definition of, 3–6  
    features of, 3–4  
    in marketing and sales, 7  
    in medical diagnosis, 5  
    search engines and, 7  
    in transportation, 7–8  
    types of, 8–13  
    uses, 5–7  
matrix  
    addition, 29  
    determinant of, 32–33  
    dimensions of, 27  
    inverse of, 33–34  
    multiplication, 31–32  
    negative of, 30  
    scalar multiplication, 30  
    subtraction, 30  
    transpose, 32  
    types of, 28–29

maximum likelihood estimation, 142–143  
maximum likelihood method, 100  
maximum non-Gaussianity approach, 100  
McCulloch–Pitts model of neuron, 114–115  
McKinsey & Company, 4  
mean squared error function, 21  
median binning technique, 78  
method of least squares, 127  
minimax solution, 13  
minimum description length principle, 170  
minimum spanning tree algorithm, 253  
minimum value of cost function  
    role of gradient function in, 24–25  
    for single variable, 21–22  
    for two-variable function, 23–24  
Mitchell, Tom M, 3  
multi-agent systems, 12  
multiclass classification problem, 9  
multilayer feed-forward network, 108–109  
multilayer recurrent network, 110  
multiple R, 132  
mutually exclusive rules, 174

## N

naive Bayes classifier, 211, 212–214  
ndarray, 57  
nearest-neighbor clustering algorithm, 253  
neural networks, evolution of, 106–107  
Newton’s method, 275–276  
nodes, 107  
nonlinear decorrelation method, 100  
nonlinear programming (NLP) problem, 272  
normal equations, 128  
notations, 27  
np.arange( ) function, 61  
Numerical Python (NumPy), 57–61  
    array attributes of, 59–61  
    data types in, 58–59  
    Matplotlib, 61–63  
    pip install, 57–58  
Nvidia, 4

## O

odds of an event, 141  
odds ratio, 141  
Ohm’s law, 124  
operations research, 12, 271  
optimal hyperplane, 183–186  
optimization, 271  
optimization problem, 272  
Orange, 40  
order of matrix, 27  
outliers, 80  
overfitting, 170

## P

Pandas  
    DataFrame in, 63–65  
    manipulation of columns, 65–67  
partitioning clustering, 243–251  
pattern recognition, 5  
performance evaluation, 176  
plane, 182  
point estimators, 128, 182  
precision, 221–222  
principal component analysis, 97  
principal component analysis (PCA), 81–96  
Python  
    arithmetic operations in, 48–49  
    assignment operator, 50  
    break statement in, 55–56  
    comparison (relational) operators, 49  
    continue statement in, 56  
    data types in, 45–46  
    decision-making, 50–52  
    dictionaries, 47–48  
    downloading and installing, 39  
    features of, 38  
    garbage collection in, 38  
    GUI programming and database compatibility, 38  
    history of, 37  
    identifiers in, 44–48  
    as integration language, 38  
    interactive mode in, 38  
    library of, 38  
    lines and indentation, 44–45  
    lists, 46–47  
    logical operators of, 50  
    loop control statements in, 55  
    loops, 53–57  
    for loops, 53–54  
    membership operators of, 50  
    multi-paradigm support for, 38  
    multiple assignments in, 45  
    nested loops, 54–55  
    numbers, 46  
    pass block, 56–57  
    pass statement in, 56–57  
    reserved words or keywords of, 44  
    running, 39–40  
    source code, 37–38  
    strings, 46  
    tuples, 47  
    variables in, 45  
    versions of, 38  
    while loops, 53  
Python 3.0  
    interactive mode, 43  
    script mode programming, 43

**Q**

quadratic relationship, 126

**R**

radial basis functions (RBF), 202–205  
 ramp function, 113–114  
 random search method, 277  
 RankBrain, 7  
 recall, 222  
 recurrent network, 110  
 reduced error pruning method, 171–172  
 regression analysis  
     characteristics of, 123  
     simple linear regression, 123–124  
 regression problem, 18  
 reinforcement learning, 12, 111  
 representation of a model, 17–18  
 residual, 127  
 robot navigation, 4  
 Rossum, Guido van, 37  
 row matrix, 28  
 row vector, 27  
 R-square, 131–132  
 RStudio, 40  
 rule antecedent, 173–174  
 rule-based classification, 173–174  
 rule consequent, 173–174  
 rule-ordering schemes, 175  
 rule post-pruning method, 172  
 rule pruning, 175–177

**S**

scalar matrix, 28  
 scikit-learn, 67  
 search engines, 7  
 self-organization, 111  
 semantic transformations, 81  
 sensitivity, 222  
 separating hyperplane, 182  
 sequential covering, 175  
 sgn(net) function, 113  
 simple linear regression, 123–124  
 simulated annealing, 279–281  
 simulation-based optimization, 12  
 single-layer feed forward network, 108  
 single-linkage algorithm, 253  
 smoothing  
     by bin medians, 78  
     by boundaries, 79  
     by means, 79  
     by regression, 79  
 soft clustering, 12  
 spam detection, 140

specificity, 222

splitting rules, 158  
 Spyder, 40, 43  
 squared error function, 21  
 square matrix, 28  
 standard error of the estimate, 132–133  
 Stanford University, 4  
 statistical relationship  
     mortality due to skin cancer, 124–125  
     between two variables, 125–126  
 statistics and genetic algorithms, 12  
 steepest descent (Cauchy) method, 273–275  
 subplot() function, 62  
 supervised learning, 8–9, 17, 110–111, 123  
     learning process in, 18  
 support vector machines (SVMs), 181  
     linear, 182–183  
 swarm intelligence, 12  
 synapses, 107

**T**

telecom cluster analysis, 263–264  
 tic-tac-toe game, 12–13  
 training set, 18  
 transpose of matrix, 32

**U**

unconstrained minimization problems, 273  
 underfitting, 170  
 unipolar activation functions, 113  
 unipolar binary activation function, 113  
 unipolar binary function, 113  
 unipolar continuous activation function, 113  
 unit matrix, 29  
 unsupervised learning, 10–11, 17, 111. *See also* clustering  
 unweighted pair group mean averaging, 261  
 upper triangular matrix, 28

**V**

vectors, 27  
     definition of, 186  
     difference between two, 189  
     dot product of two, 190  
     norm of, 187–188  
     orthogonal projection of, 190–192  
     sum of, 189

**W**

weight, 108

**Z**

zero matrix, 28

*Machine Learning* is an application of artificial intelligence and provides systems with the ability to automatically learn and improve from experience, without being explicitly programmed. This subject is an important technological breakthrough of recent time. It has its application in virtually every domain and active research is going on in the use of machine learning to various applications. In fact, with the amount of data that is available along with the exceptional computing power, machine learning can prove to be extremely useful in the future.

More than 80% of companies around the world are already planning to use machine learning and artificial intelligence to enhance their customer experience. The biggest challenge is the lack of expertise. This has opened up lots of job opportunities in this area. This is the reason why it has become a most sort after elective among students, especially in the graduate and undergraduate studies.

This book presents the content in a concise yet comprehensive manner, since it is focused on the students' point of view, which will help cater to their academic needs. Starting with why machine learning is required to covering various supervised and unsupervised learning algorithms and optimizations.

follow us on



[facebook.com/wileyindia](https://facebook.com/wileyindia)



[twitter.com/wileyindiapl](https://twitter.com/wileyindiapl)



[linkedin.com/in/wileyindia](https://linkedin.com/in/wileyindia)



[google.com/+wileyindia](https://google.com/+wileyindia)

SHELVING CATEGORY  
Engineering

### Wiley India Pvt. Ltd.

Customer Care +91 120 6291100  
[csupport@wiley.com](mailto:csupport@wiley.com)  
[www.wileyindia.com](http://www.wileyindia.com)  
[www.wiley.com](http://www.wiley.com)

### SALIENT FEATURES OF THE BOOK

- Detailed explanation of data preprocessing and Hidden Markov Model.
- Study of Python with a foundation on datasets and generic program execution.
- Elucidation of supervised, unsupervised, and reinforcement learning with examples.
- Study on artificial neural networks and optimization techniques.
- Description of linear regression and logistic regression.
- Explanation of decision tree, support vector machine, and Bayesian classification, with ample worked-out examples.
- Examination of matrices and vectors and detailed evaluation of a model from datasets.
- Coverage of different clustering techniques with numericals.
- Detailed step-by-step worked out examples of algorithms.
- Presentation of case studies where relevant.
- Sample codes for each of the learning algorithms.

ISBN: 978-81-265-7851-1



WILEY