SGOU
Higher Education for All

MACHINE LEARNING

COURSE CODE: M25CA06DC
Master of Computer Applications
Discipline Core Course
Self Learning Material



SGOU

SREENARAYANAGURU OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU OPEN UNIVERSITY

SGOU

Vision

To increase access of potential learners of all categories to higher education, research and training, and ensure equity through delivery of high quality processes and outcomes fostering inclusive educational empowerment for social advancement.

Mission

To be benchmarked as a model for conservation and dissemination of knowledge and skill on blended and virtual mode in education, training and research for normal, continuing, and adult learners.

Pathway

Access and Quality define Equity.

Machine Learning

Course Code: M25CA06DC

Semester - II

SGOU

Discipline Core Course
Postgraduate Programme
Master of Computer Applications
Self Learning Material

SREENARAYANAGURU
OPEN UNIVERSITY

SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SREENARAYANAGURU
OPEN UNIVERSITY

MACHINE LEARNING

Course Code: M25CA06DC
Semester- II
Discipline Core Course
Master of Computer Applications

Academic Committee

Prof. (Dr.) Sabu M.K.
Dr. G. Santhoshkumar T
Prof. (Dr.) Vinod Chandra S.S.
Dr. Vinod P.
Dr.Lajish V.L.
Sreekanth M.S.
Dr. Vivek P.
Dr. Arun K.S.
Dr. Abdul Jebbar P.

Development of the Content

Shamin S.
Greeshma P.P.
Sreerekha V.K.
Kanitha D.K.
Subi Priya Laxmi S.B.N.
Aswathy V.S.

Review and Edit

Dr. Leena Silvoster M.

Scrutiny

Shamin S.
Greeshma P.P.
Sreerekha V. K.
Dr. Kanitha D.K.
Subi Priya Laxmi S.B.N.
Aswathy V.S.

Cover Design

Jobin J.

Co-ordination

HoS, School of Communication and Information Science

Scan this QR Code for reading the SLM on a digital device.

Edition
September 2026

Copyright
© Sreenarayanaguru Open University

ISBN 978-81-181574-2-7

SGOU

All rights reserved. No part of this work may be reproduced in any form, by mimeograph or any other means, without permission in writing from Sreenarayanaguru Open University. Printed and published on behalf of Sreenarayanaguru Open University by Registrar, SGOU, Kollam.

www.sgou.ac.in

Visit and Subscribe our Social Media Platforms

MESSAGE FROM VICE CHANCELLOR

Dear Learner,

It gives me immense pleasure and a deep sense of pride to warmly welcome you to Sreenarayanaguru Open University—a vibrant and progressive institution committed to transforming lives through inclusive, flexible, and high-quality education.

Established in September 2020 as a forward-looking initiative of the Government of Kerala, the University stands as a beacon of opportunity for learners seeking to advance their academic and professional aspirations through the open and distance learning mode. Guided by our foundational principle that "access and quality define equity," we are steadfast in our mission to democratize education while upholding uncompromising academic standards.

Our university is inspired by the timeless vision and philosophy of Sree Narayana Guru, whose ideals of knowledge, equality, and social transformation continue to guide our academic journey. His enduring legacy instils in us the responsibility to create an educational environment that empowers individuals, nurtures critical thinking, and contributes meaningfully to society.

Understanding the dynamic needs of contemporary learners, we have adopted a robust and learner-centric blended learning model, seamlessly integrating Self-Learning Materials, Academic Counselling, and Advanced Digital Learning Platforms. This holistic approach ensures flexibility without sacrificing academic depth, enabling you to learn at your own pace while remaining meaningfully connected to a vibrant academic ecosystem.

The Master of Computer Applications (MCA) programme you are embarking upon is carefully designed to position you at the forefront of the digital revolution. It uniquely blends strong theoretical foundations with practical, industry-oriented competencies. The curriculum emphasizes algorithmic thinking, system design, programming, database management, networking, and emerging technologies such as artificial intelligence, data science, and cloud computing.
What sets this programme apart is its forward-thinking design, which offers:

- Opportunities for skill enhancement aligned with industry needs
- Multidisciplinary learning pathways for broader intellectual development
- Exposure to innovative and emerging technology domains
- Multiple specialization options tailored to evolving career landscapes
- A strong focus on employability, entrepreneurship, and global career readiness
-

Our Self-Learning Materials are meticulously developed by experts, enriched with contemporary case studies, real-world applications, and practical insights to ensure clarity, engagement, and relevance. We are committed to equipping you not just with knowledge, but with the confidence and competence to excel in a competitive global environment.

At Sreenarayanaguru Open University, you are never alone in your learning journey. Our dedicated learner support system is designed to provide continuous academic guidance, timely assistance, and effective grievance redressal. We encourage you to actively engage with us, share your concerns, and make the most of the resources and support available to you.

As you begin this important phase of your academic journey, I urge you to embrace learning with curiosity, discipline, and determination. The world of technology is ever-evolving, and your willingness to adapt, innovate, and grow will define your success.

Remember, this is not just a programme—it is a pathway to transforming your future.
I wish you a fulfilling learning experience and a successful, inspiring career ahead.

Warm regards,



Prof. (Dr.) Jagathy Raj V. P.
Vice Chancellor

SGOU

Contents

BLOCK 1 Fundamentals of Machine Learning **1**

UNIT 1 Foundations of Machine Learning 2
UNIT 2 Types of Machine Learning 11
UNIT 3 Data Preprocessing 21
UNIT 4 Evaluation Metrics 48

BLOCK 2 Supervised Learning **69**

UNIT 1 Classification Algorithms 70
UNIT 2 Regression Algorithms 86
UNIT 3 SVM and Ensemble Method 104
UNIT 4 Model Tuning and Cross-validation 127

BLOCK 3 Unsupervised Learning **143**

UNIT 1 Clustering Techniques 144
UNIT 2 Dimensionality Reduction 181
UNIT 3 Association Rule Learning 216
UNIT 4 Reinforcement Learning Basics 252

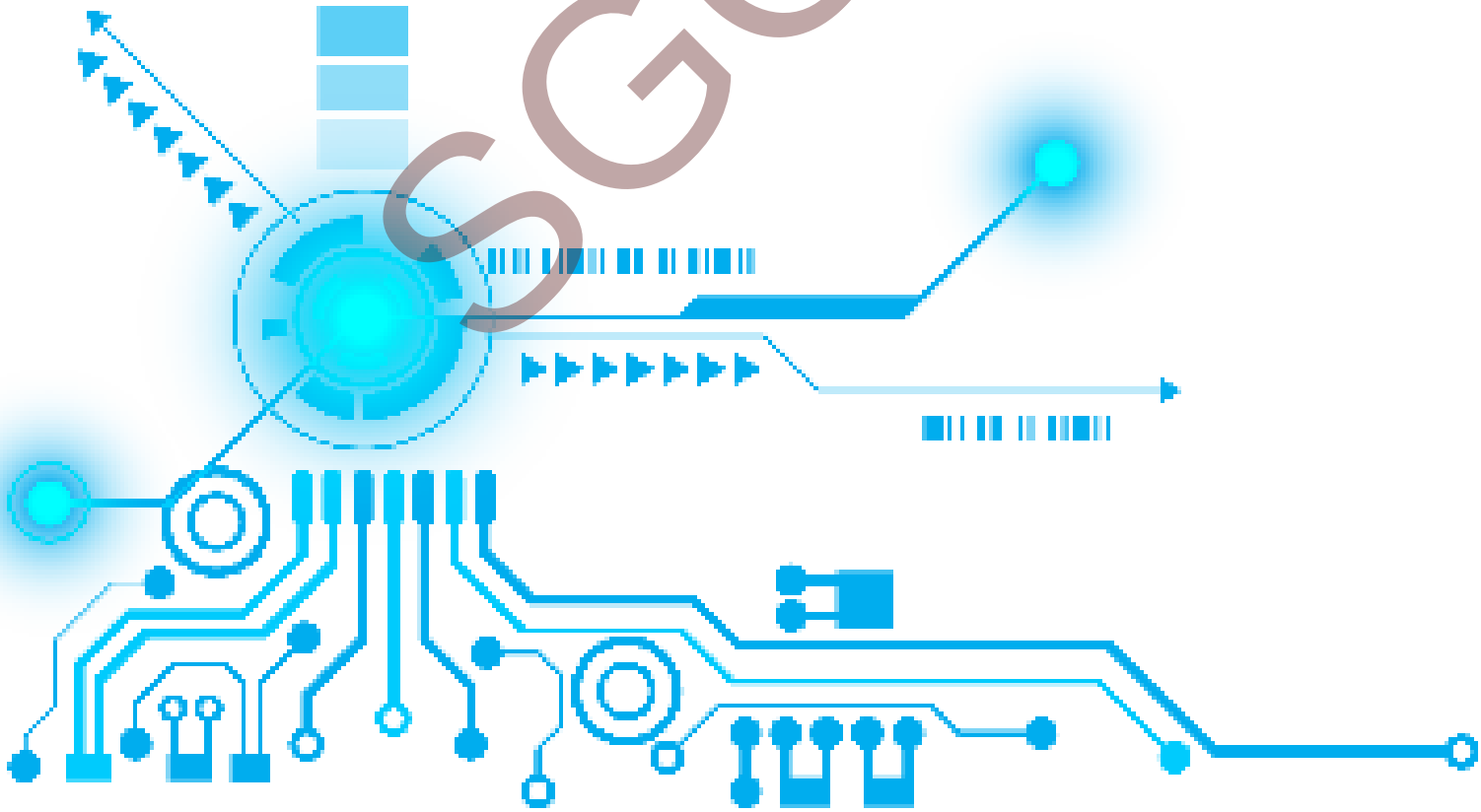
BLOCK 4 Advanced Machine Learning **269**

UNIT 1 Neural Network Basic 270
UNIT 2 Deep Learning Architectures 288
UNIT 3 Model Deployment and Case Studies 305
UNIT 4 Recent Trends and Research in Machine Learning 319

MODEL QUESTION PAPER SETS **334**

SGOU

BLOCK 1
Fundamentals of Machine Learning



SGOU

1 UNIT Foundations of Machine Learning

Learning Outcomes

After completion of this unit, the learner will be able to:

- define Machine Learning and explain its basic concepts
- differentiate between traditional programming and Machine Learning
- describe the scope and real-world applications of Machine Learning
- explain the evolution of Machine Learning from basic algorithms to deep learning
- distinguish between Artificial Intelligence, Machine Learning, and Deep Learning

SGOU

Background

In today's digital era, an enormous amount of data is generated every second through smartphones, social media, online transactions, and various digital platforms. Traditional methods of data processing are no longer sufficient to handle such large volumes of information efficiently. This has led to the development of advanced technologies like Machine Learning, which can analyze data, identify patterns, and make intelligent decisions automatically.

Machine Learning emerged as a part of Artificial Intelligence with the goal of enabling machines to mimic human learning. Instead of relying solely on predefined rules, ML systems learn from data and improve their performance over time. This capability has made Machine Learning a key technology in solving real-world problems across various domains such as healthcare, finance, education, and transportation.

The importance of studying Machine Learning lies in its wide range of applications and its growing demand in the industry. From recommendation systems and voice assistants to fraud detection and self-driving cars, Machine Learning is transforming the way technology interacts with humans. Understanding its concepts provides a strong foundation for developing intelligent systems and exploring future innovations in Artificial Intelligence.

Keywords

Machine Learning, Artificial Intelligence, Deep Learning, Data, Model, Algorithm, Prediction, Pattern Recognition, Big Data, Neural Networks, Automation

Discussion

1.1.1 Definition of Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence that focuses on developing systems that can learn from data and improve their performance without being explicitly programmed. In traditional programming, a developer writes fixed rules that the computer follows to produce output from given input. However, in Machine Learning, the system learns these rules automatically by analyzing data. This makes ML systems more flexible and capable of adapting to new situations over time. For example, instead of manually defining rules to identify spam emails, a Machine Learning model learns from a dataset of emails and identifies patterns such as repeated keywords or suspicious sender behavior to classify emails as spam or not spam.

Machine Learning works based on the concept of learning from experience. The more data the system is exposed to, the better it becomes at making predictions or decisions. For instance, an online shopping website tracks user behavior such as searches, clicks, and purchases. Based on this data, the system learns user preferences and recommends products that the user is likely to buy. Over time, these recommendations become more accurate, improving the user experience.

Another important feature of Machine Learning is its ability to identify hidden patterns in data. These patterns may not be easily visible to humans. For example, in banking, ML systems analyze transaction data to detect fraudulent activities by identifying unusual patterns, such as sudden high-value transactions or transactions from unfamiliar locations. Similarly, in weather forecasting, ML models analyze historical data to predict future weather conditions.

SGOU

Machine Learning also plays a crucial role in handling large volumes of data efficiently. In today's digital world, massive amounts of data are generated every second. ML algorithms are designed to process and analyze such large datasets quickly and accurately, making them essential for modern applications.

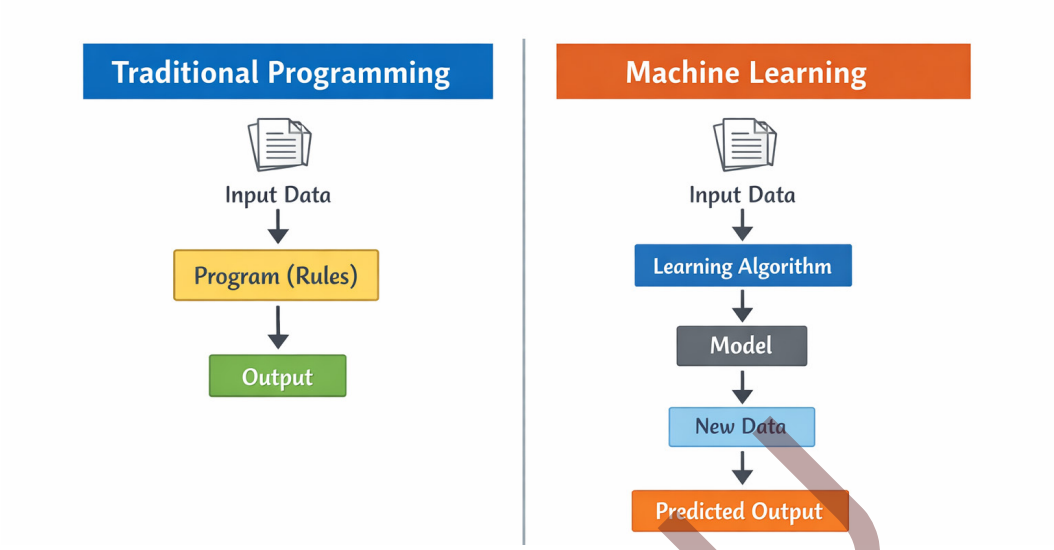


Fig. 1.1.1 Traditional Programming vs Machine Learning

1.1.2 Scope of Machine Learning

The scope of Machine Learning is vast and continuously expanding due to the rapid growth of data and advancements in computational power. It is widely used across various industries, transforming traditional processes into intelligent systems capable of making data-driven decisions.

In the healthcare sector, Machine Learning is used for disease prediction and diagnosis. By analyzing patient data such as medical history, test reports, and lifestyle habits, ML models can predict the likelihood of diseases like diabetes or heart conditions. For example, a system can analyze blood sugar levels and other health indicators to warn doctors about potential risks, enabling early treatment.

In the finance sector, Machine Learning plays a critical role in fraud detection and risk management. Financial institutions use ML algorithms to monitor transactions and identify suspicious activities. For instance, if a credit card is used for an unusually large transaction in a foreign country, the system may flag it as fraudulent. ML is also used in credit scoring, where it evaluates a person's financial behavior to determine their ability to repay loans.

In education, Machine Learning enables personalized learning by adapting content to individual student needs. For example, online learning platforms track a student's performance and recommend lessons or exercises based on their strengths and weaknesses. This ensures that students receive customized learning experiences.

In business, Machine Learning is used for customer segmentation, sales forecasting, and recommendation systems. E-commerce platforms analyze user behavior to suggest products that match their interests, increasing customer satisfaction and sales. In

SGOU

transportation, Machine Learning powers autonomous vehicles that can detect obstacles, recognize traffic signals, and make driving decisions without human intervention.

Machine Learning is important because it automates complex processes, improves accuracy, and enables organizations to make faster and better decisions. It reduces human effort and allows systems to operate efficiently even with large datasets.

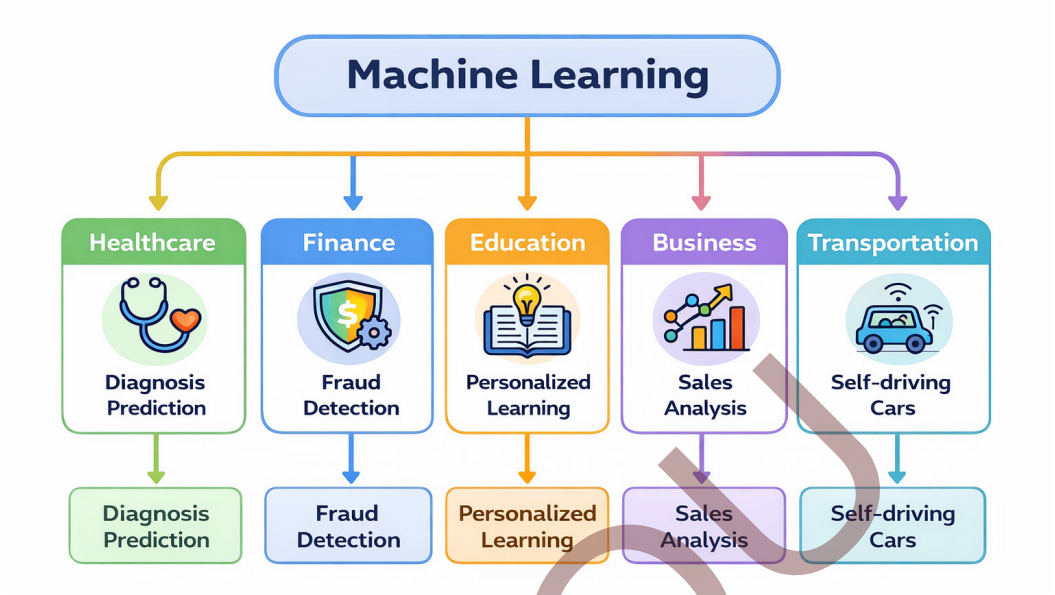


Fig. 1.1.2 Scope of Machine Learning Across Domains

1.1.3 Evolution of Machine Learning

Machine Learning has evolved significantly over time, progressing from simple algorithms to highly advanced deep learning systems. This evolution has been driven by improvements in computing power, availability of large datasets, and advancements in algorithms.

In the early stages (1950s–1970s), Machine Learning focused on basic algorithms such as the perceptron. These models were capable of performing simple pattern recognition tasks, such as distinguishing between basic shapes. Although limited in capability, they laid the foundation for future developments.

During the 1980s and 1990s, neural networks and statistical learning methods became more popular. Techniques such as backpropagation allowed neural networks to learn more effectively by adjusting weights based on errors. This enabled systems to perform more complex tasks, such as handwriting recognition and speech processing. However, limitations in computational power and data availability slowed progress.

In the 2000s, the rise of big data and data mining transformed Machine Learning. The availability of large datasets allowed ML models to identify patterns and trends more effectively. Applications such as search engines, recommendation systems, and customer analytics became more advanced and accurate.

From the 2010s to the present, Deep Learning has revolutionized Machine Learning. Deep Learning models use multi-layered neural networks to process complex data

SGOU

such as images, audio, and text. For example, facial recognition systems can identify individuals in images, and voice assistants can understand spoken language. These advancements have significantly improved the capabilities of Machine Learning systems.

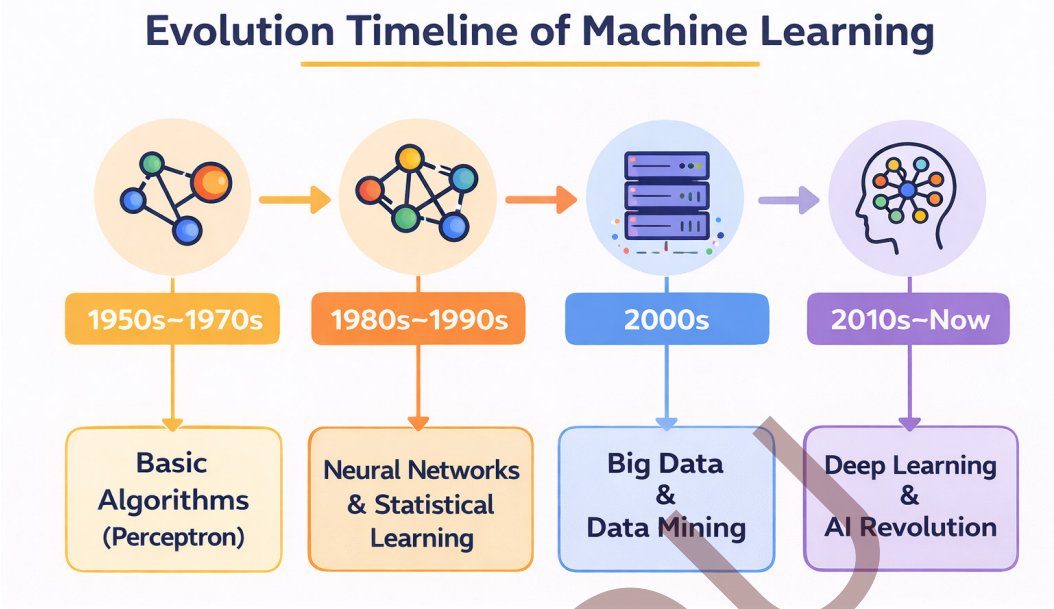


Fig. 1.1.3 Evolution Timeline of Machine Learning

1.1.4 Real-World Applications of Machine Learning

Machine Learning is widely used in real-world applications that we encounter daily. One of the most common applications is email spam filtering, where ML algorithms classify emails as spam or not spam based on patterns learned from previous emails. This helps users avoid unwanted messages and improves email security.

Voice assistants such as Siri and Alexa use Machine Learning to understand human speech, process commands, and provide responses. For example, when a user asks for weather updates, the system processes the speech, interprets the request, and provides accurate information.

Recommendation systems used by platforms like Netflix and Amazon analyze user behavior to suggest relevant content or products. For instance, if a user frequently watches action movies, the system recommends similar movies based on viewing history. This enhances user engagement and satisfaction.

Face recognition systems are commonly used in smartphones for security purposes. These systems analyze facial features and match them with stored data to verify identity. Autonomous vehicles use Machine Learning to detect objects, recognize traffic signals, and make driving decisions, ensuring safe navigation.

Machine Learning improves efficiency, enhances user experience, and enables automation across various industries. Its applications continue to grow as technology advances.

SGOU

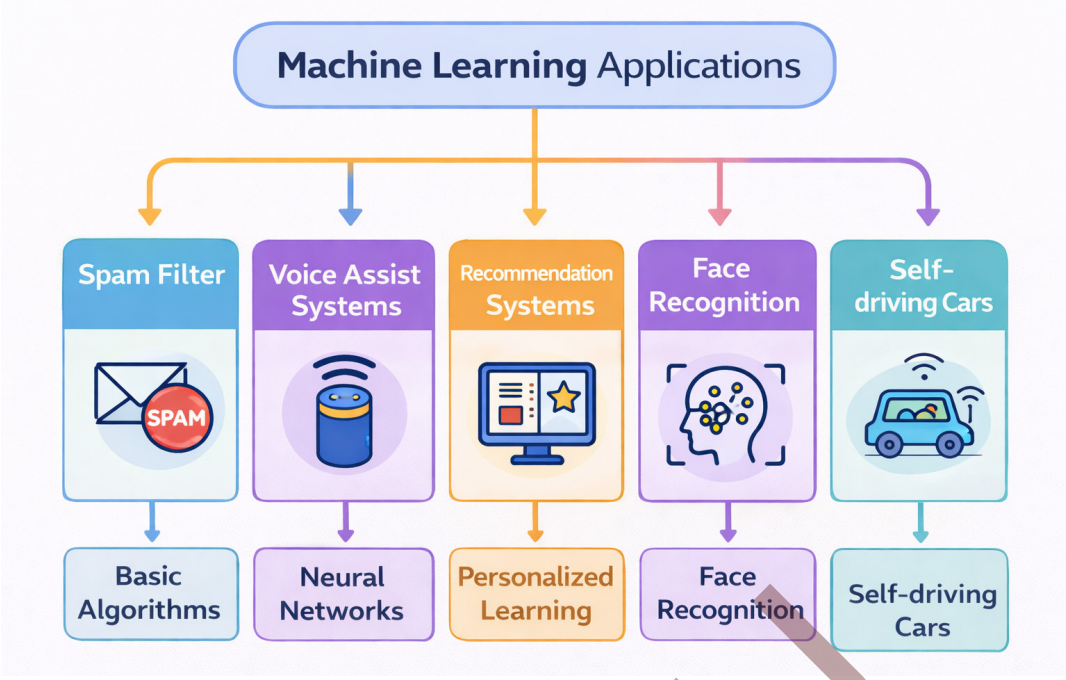


Fig. 1.1.4 Real-World Applications of ML

1.1.5 AI vs Machine Learning vs Deep Learning

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are closely related but differ in scope and complexity. Artificial Intelligence is the broadest concept, aiming to create intelligent systems that can perform tasks requiring human intelligence, such as chatbots that interact with users. Machine Learning is a subset of AI that focuses on enabling machines to learn from data and improve their performance, such as spam filters that classify emails. Deep Learning is a subset of Machine Learning that uses multi-layered neural networks to process complex data, such as recognizing objects in images or understanding speech.

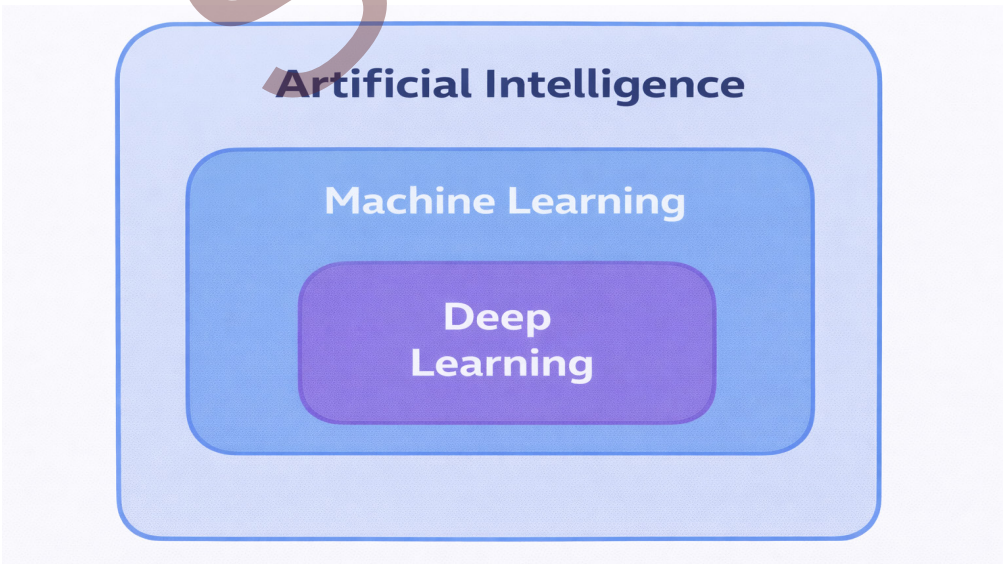


Fig. 1.1.5 Relationship between AI, ML, and DL

SGOU

In simple terms, AI represents the overall concept of intelligent systems, Machine Learning is a method used to achieve this by learning from data, and Deep Learning is an advanced technique that handles highly complex tasks using neural networks.

Table 1.1.1 Comparison Table of AI, Machine Learning, Deep Learning

Aspect	Artificial Intelligence	Machine Learning	Deep Learning
Definition	Broad field of intelligent systems	Subset of AI	Subset of ML
Approach	Rule-based + learning	Data-driven	Neural networks
Complexity	High	Medium	Very High
Example	Chatbots answering queries	Spam email filtering	Face recognition systems

Summerised Overview

Machine Learning is a powerful subset of Artificial Intelligence that enables systems to learn from data and improve their performance without explicit programming. Unlike traditional systems that rely on predefined rules, Machine Learning models learn patterns from data and make predictions or decisions based on those patterns.

The scope of Machine Learning is vast, covering fields such as healthcare, finance, education, business, and transportation. It plays a crucial role in automating processes, improving accuracy, and handling large-scale data efficiently. Over time, Machine Learning has evolved from simple algorithms like perceptrons to advanced deep learning models capable of solving complex problems.

Machine Learning is widely used in real-world applications such as spam filtering, voice assistants, recommendation systems, face recognition, and self-driving cars. These applications enhance user experience and improve efficiency in various domains.

Artificial Intelligence, Machine Learning, and Deep Learning are interconnected concepts, where AI is the broader field, ML is a subset that focuses on learning from data, and DL is an advanced technique using neural networks for complex tasks.

SGOU

Assignments

1. Explain the concept of Machine Learning with suitable examples.
2. Discuss the scope of Machine Learning in different fields.
3. Describe the evolution of Machine Learning in detail.
4. Explain real-world applications of Machine Learning with examples.
5. Differentiate between Artificial Intelligence, Machine Learning, and Deep Learning.

Suggested Reading

1. Alpaydin, E. (2016). *Machine learning: The new AI*. MIT Press.
2. Burkov, A. (2019). *The hundred-page machine learning book*. Andriy Burkov.
3. Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.
4. Theobald, O. (2017). *Machine learning for absolute beginners*. Scatterplot Press.

SGOU

Reference

1. Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.
2. Alpaydin, E. (2016). *Machine learning*. MIT Press.
3. Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
4. Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

2 UNIT Types of Machine Learning

Learning Outcomes

After completion of this unit, the learner will be able to:

1. understand the types of Machine Learning techniques:- supervised, unsupervised, and reinforcement learning
2. differentiate between labeled, unlabeled, and feedback-based learning approaches
3. identify real-world applications of each type of Machine Learning
4. compare the advantages and limitations of different Machine Learning techniques

SGOU

Background

Machine Learning has evolved as a core area of Artificial Intelligence, enabling systems to learn from data without explicit programming. With the growth of digital data in sectors like healthcare, finance, education, and transportation, traditional programming approaches became insufficient to handle complex patterns and predictions.

The need to automate decision-making and extract meaningful insights from large datasets led to the development of different learning paradigms. These paradigms: supervised, unsupervised, and reinforcement learning—address different types of problems. Understanding these categories is essential because each type is designed to solve specific real-world challenges, such as prediction, pattern discovery, and decision-making.

Keywords

Machine Learning, Supervised Learning, Unsupervised Learning, Reinforcement Learning, Classification, Regression, Clustering, Agent, Environment, Reward, Prediction, Pattern Recognition

Discussion

1.2.1 Overview of Machine Learning Types

Machine Learning is broadly classified into three main types based on how models learn from data: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. This classification is important because different real-world problems require different learning approaches.

In supervised learning, the system learns from labeled data where both input and output are known. In unsupervised learning, the system works with unlabeled data and discovers hidden patterns or structures. In reinforcement learning, the system learns by interacting with its environment and improving through feedback in the form of rewards or penalties.

For example, when predicting student marks based on previous data, supervised learning is used because the output (marks) is already known during training. When grouping customers based on purchasing behavior, unsupervised learning is used because there are no predefined labels. When training a robot or a game-playing system, reinforcement learning is used because the system learns through trial and error.

These three types together form the foundation of Machine Learning and are widely used in industries such as healthcare, finance, education, and transportation.

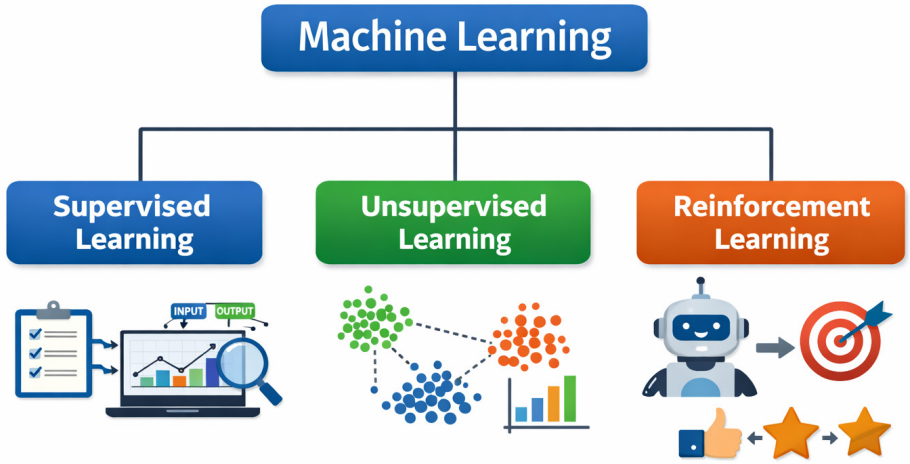


Fig. 1.2.1 Types of Machine Learning

SGOU

1.2.2 Supervised Learning

Supervised Learning is one of the most widely used Machine Learning techniques. In this method, the model is trained using labeled data, meaning each input is associated with a correct output. The goal is to learn a mapping function that can predict outputs for new, unseen inputs.

During training, the model continuously compares its predictions with actual outputs and adjusts itself to minimize errors. This process is called learning, and it improves the model's accuracy over time.

For example, in email spam detection, the system is trained using thousands of emails labeled as "spam" or "not spam." It learns patterns such as suspicious words, links, or sender addresses. Once trained, it can classify new emails automatically.

Another example is house price prediction. The model is trained using past data such as location, size, number of rooms, and price. Based on this, it can estimate the price of a new house.

In healthcare, supervised learning is used to predict diseases. For instance, based on patient symptoms and medical history, the model can predict whether a patient has a particular disease.

1.2.2.1 Types of Supervised Learning

Supervised learning is mainly divided into classification and regression.

Classification is used when the output is categorical. For example, predicting whether a student will pass or fail, or identifying whether an email is spam or not.

Regression is used when the output is continuous. For example, predicting temperature, sales revenue, or house prices.

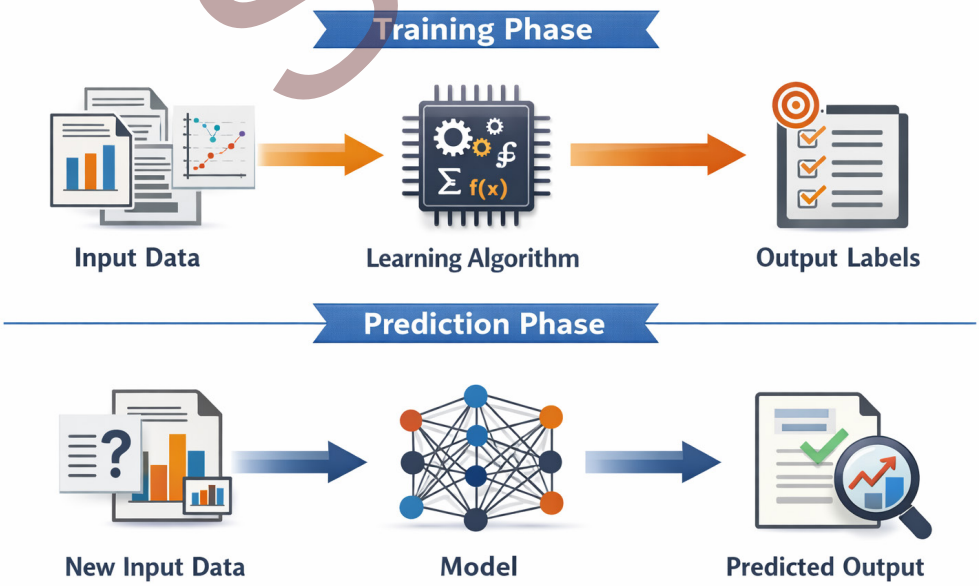


Fig. 1.2.2 Supervised Learning Process

SGOU

Examples of Supervised Learning

- Credit card fraud detection (Fraud / Not Fraud)
- Image classification (Cat / Dog)
- Weather prediction (Temperature values)
- Stock price prediction
- Loan approval systems

Advantages of Supervised Learning

Supervised learning provides high accuracy when trained with good quality data. It is easy to evaluate performance using metrics such as accuracy and error rate. It is widely used in real-world applications where prediction is required.

Limitations

It requires a large amount of labeled data, which may be expensive and time-consuming to collect. It may also fail when new data is very different from training data.

1.2.3 Unsupervised Learning

Unsupervised Learning is a type of Machine Learning where the model works with unlabeled data. The system does not know the correct output and must discover patterns, relationships, or structures on its own.

This type of learning is useful for exploratory data analysis. It helps in understanding the data and finding hidden insights.

For example, in customer segmentation, businesses analyze customer data such as age, income, and buying behavior. The system groups customers into different categories without predefined labels, such as premium customers, regular customers, and occasional buyers.

Another example is grouping news articles based on topics such as sports, politics, and technology without prior labeling.

1.2.3.1 Techniques in Unsupervised Learning

Clustering is used to group similar data points together. For example, grouping students based on performance levels.

Association is used to find relationships between items. For example, customers who buy rice also buy curry powder.

SGOU

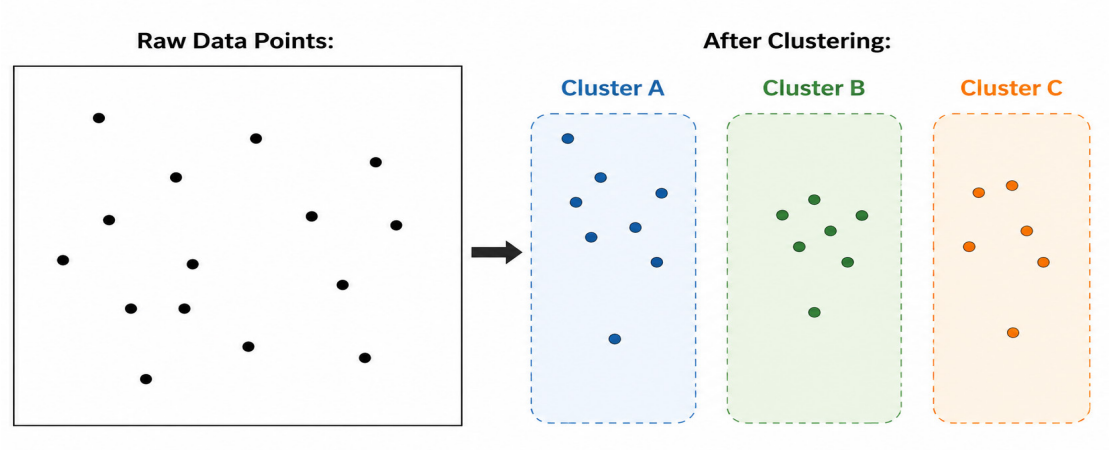


Fig. 1.2.3 Clustering Example

Examples of Unsupervised Learning

- Customer segmentation in marketing
- Social network analysis
- Document grouping
- Image compression
- Anomaly detection (detecting unusual patterns)

Advantages of Unsupervised Learning: It does not require labeled data and can handle large datasets. It helps discover hidden patterns and relationships that are not obvious.

Limitations: The results may not always be accurate and can be difficult to interpret. There is no clear way to evaluate performance.

1.2.4 Reinforcement Learning

Reinforcement Learning is a type of Machine Learning where the system learns by interacting with its environment. The system improves its performance by receiving rewards for correct actions and penalties for incorrect actions.

The learner is called an agent. The agent interacts with the environment by taking actions. Based on the outcome, it receives feedback and updates its strategy.

For example, in a game like chess, the system learns by playing many games. Winning gives a reward, and losing gives a penalty. Over time, the system learns the best moves.

In robotics, machines learn how to perform tasks such as walking or picking objects through repeated trials. In self-driving cars, reinforcement learning helps in decision-making such as braking, accelerating, and turning.

SGOU

Key Components

- Agent – learner or decision-maker
- Environment – surroundings
- Action – steps taken
- Reward – feedback

Working of Reinforcement Learning

1. Agent observes environment
2. Takes action
3. Receives reward or penalty
4. Updates policy (strategy)
5. Repeats process



Fig.1.2.4 Reinforcement Learning Cycle

SGOU

Examples of Reinforcement Learning

- Game AI (Chess, AlphaGo)
- Robot navigation
- Traffic signal optimization
- Recommendation systems (learning user preferences)
- Industrial automation

Advantages of Reinforcement Learning: It is suitable for complex decision-making problems and works well in dynamic environments. It does not require labeled data.

Limitations: It requires a lot of training time and computational power. It is difficult to design and implement.

1.2.5 Comparison of ML Types

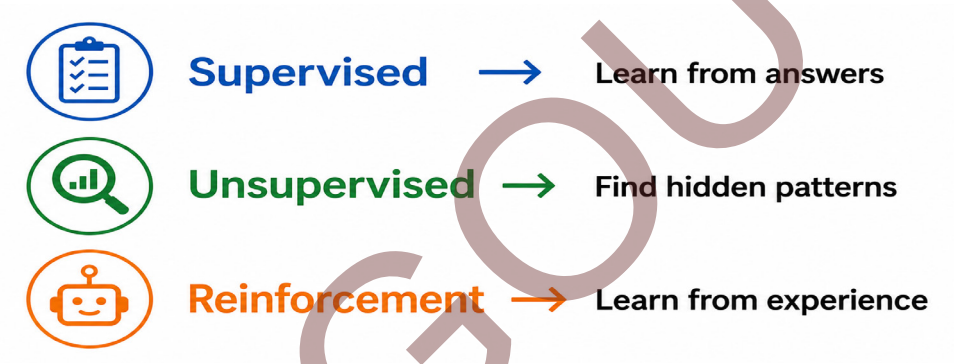


Fig.1.2.5 Types of Machine Learing

The three types of Machine Learning differ in how they learn from data and what their objectives are. Supervised learning focuses on prediction using labeled data. Unsupervised learning focuses on discovering patterns in unlabeled data. Reinforcement learning focuses on decision-making through feedback.

Table 1.2.1 Comparison of Machine Learning Types

Feature	Supervised	Unsupervised	Reinforcement
Data	Labeled	Unlabeled	Feedback-based
Goal	Predict output	Find patterns	Maximize reward
Learning Style	Guided	Self-discovery	Trial & error
Example	Spam detection	Customer grouping	Game AI

SGOU

Summerised Overview

Machine Learning is broadly classified into three main types based on how systems learn from data: supervised, unsupervised, and reinforcement learning. Supervised learning involves training models using labeled data to make accurate predictions, making it suitable for tasks like classification and regression. Unsupervised learning, on the other hand, deals with unlabeled data and focuses on discovering hidden patterns, relationships, or groupings within the data, such as clustering and association. Reinforcement learning differs by allowing an agent to learn through interaction with its environment, using rewards and penalties to improve decision-making over time. Each type serves a unique purpose and is widely applied in real-world domains such as healthcare, finance, marketing, and automation, depending on the nature of the problem and available data.

Assignments

1. Define Machine Learning and explain its main types.
2. Differentiate between supervised and unsupervised learning with examples.
3. What is reinforcement learning? Explain with a real-life example.
4. Explain classification and regression in supervised learning.
5. What is clustering? Give an example of its application.

SGOU

Suggested Reading

1. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow Géron, A. (2022). *Hands-on machine learning with Scikit-learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

2. Introduction to Machine Learning with Python, Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.

3. Machine Learning Yearning, Ng, A. (2018). *Machine learning yearning*. DeepLearning.AI.

4. Data Mining: Concepts and Techniques, Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

Reference

1. Pattern Recognition and Machine Learning, Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

2. Machine Learning: A Probabilistic Perspective, Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

3. The Elements of Statistical Learning, Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

4. Deep Learning, Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

3
UNIT

Data Preprocessing

Learning Outcomes

After completion of this unit, the learner will be able to:

- Understand the concept and importance of data preprocessing
- Identify common data issues such as missing values, noise, and inconsistencies
- Apply data cleaning and transformation techniques effectively
- Perform data normalization and feature selection methods
- Split data into training and testing sets for model building

SGOU

Background

In the modern data driven world, large volumes of data are generated from various sources such as databases, sensors, and online platforms. However, this raw data is often incomplete, inconsistent, and unstructured, making it unsuitable for direct use in analysis or machine learning models. The quality of results produced by any system depends heavily on the quality of input data. Therefore, it becomes essential to process and refine this raw data before applying analytical techniques. This necessity gives rise to the concept of data preprocessing, which prepares data for accurate and efficient analysis.

This unit focuses on the fundamental techniques involved in data preprocessing, including data cleaning, transformation, normalization, feature selection, and data splitting. Each of these steps plays a vital role in improving data quality and ensuring better model performance. By understanding and applying these techniques, learners can handle real world data effectively and build reliable machine learning models. This unit provides a strong foundation for further study in data science, analytics, and machine learning.

Keywords

Data Preprocessing, Data Cleaning, Transformation, Normalization, Feature Selection, Data Splitting, Missing Values, Outliers, Encoding, Scaling, Overfitting

Discussion

1.3.1 Introduction to Data Preprocessing

Data preprocessing is one of the most important stages in the machine learning process. It refers to the process of converting raw, incomplete, inconsistent, and unorganized data into a clean, structured, and meaningful format that can be effectively used for data analysis and machine learning. Since the quality of a machine learning model depends largely on the quality of the input data, preprocessing plays a crucial role in producing accurate and reliable results.

In real-world applications, data is collected from a variety of sources such as databases, sensors, websites, mobile applications, social media platforms, and user inputs. This collected data is often imperfect and may contain missing values, duplicate records, incorrect entries, inconsistent formats, noise, and irrelevant information. If such data is directly supplied to a machine learning algorithm, it can produce inaccurate predictions, increase computational cost, and reduce the overall performance of the model.

Data preprocessing addresses these problems by improving the quality of the dataset before it is used for training a machine learning model. It involves identifying and correcting errors, handling missing values, transforming data into a suitable format, scaling numerical values, selecting important features, and dividing the dataset into training and testing subsets. These operations ensure that the data is consistent, complete, and suitable for effective model development.

A well preprocessed dataset offers several advantages. It improves the accuracy, efficiency, and reliability of machine learning models. It also reduces the training time by eliminating unnecessary or redundant information and helps machine learning algorithms identify meaningful patterns more effectively. In addition, preprocessing minimizes the chances of biased predictions and improves the ability of the model to generalize to new, unseen data.

In simple terms, data preprocessing acts as the foundation of the machine learning pipeline. Just as a strong foundation is essential for constructing a stable building, high-quality preprocessed data is essential for building accurate and reliable machine learning models.

SGOU

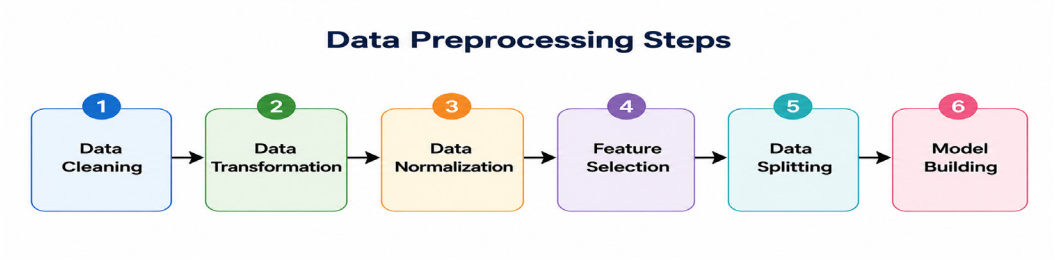


Fig. 1.3.1 Steps of data preprocessing

Figure 1.3.1 illustrates the sequence of activities involved in preparing raw data for machine learning. The process begins with Data Cleaning, where errors, duplicate records, missing values, and inconsistencies are identified and corrected to improve data quality. Once the data has been cleaned, it undergoes Data Transformation, where it is converted into a suitable format using techniques such as encoding, aggregation, and discretization.

The transformed data is then subjected to Data Normalization, which scales numerical attributes to a common range so that all features contribute equally during model training. Next, Feature Selection identifies the most relevant features while eliminating redundant or irrelevant attributes. This improves model efficiency and reduces computational complexity. After selecting the important features, the dataset is divided into training, validation (optional), and testing subsets through Data Splitting. Finally, the prepared data is used in Data Modeling, where machine learning algorithms learn patterns from the training data and generate predictive models that can be evaluated and deployed for real-world applications.

Consider an online shopping company that wants to recommend products to its customers. The customer database may contain duplicate accounts, missing addresses, inconsistent product names, and incomplete purchase records. Before building a recommendation system, the company must clean the data, convert categorical values into numerical form, normalize purchase amounts, select relevant customer attributes, and split the data into training and testing sets. Only after completing these preprocessing steps can a machine learning model generate accurate and personalized product recommendations.

1.3.2 Data Cleaning

Data cleaning is the first and one of the most important steps in the data preprocessing process. It involves identifying, correcting, or removing errors, inconsistencies, and inaccuracies present in a dataset. Since data collected from real-world sources is often incomplete or imperfect, it must be cleaned before it can be used for analysis or machine learning. High-quality data enables machine learning algorithms to learn meaningful patterns and produce accurate predictions.

Real world datasets may contain several types of problems, including missing values, duplicate records, incorrect data entries, outliers, and inconsistent data formats. If these issues are not addressed, they can reduce the accuracy of machine learning models, increase training time, and produce misleading results. Therefore, data cleaning is an essential step in ensuring that the dataset is reliable, consistent, and suitable for further processing.

SGOU

Handling Missing Values

Missing values occur when information is unavailable for one or more attributes in a dataset. They may arise due to data entry mistakes, sensor failures, incomplete surveys, or communication errors during data collection.

Several techniques are available for handling missing values. If only a small number of records contain missing values, those records may be removed. Alternatively, missing values can be replaced using statistical measures such as the mean, median, or mode. For more complex datasets, interpolation methods or machine learning-based imputation techniques can be used to estimate the missing values.

Removing Duplicate Records

Duplicate records occur when the same information is entered more than once. Such duplicates may result from repeated data entry, system synchronization errors, or merging data from multiple sources.

Keeping duplicate records can bias statistical analysis and lead to incorrect predictions. Therefore, duplicate entries should be identified and removed to ensure that each record in the dataset represents unique information.

Handling Outliers

An outlier is a data value that differs significantly from the majority of observations in a dataset. Outliers may occur due to measurement errors, data entry mistakes, or genuine rare events.

For example, if the marks of most students range between 40 and 90, a recorded mark of 1000 is clearly an outlier. Such abnormal values can distort statistical analysis and negatively affect machine learning algorithms.

Outliers can be handled by removing incorrect observations, replacing them with suitable values, or applying techniques such as capping, transformation, or normalization depending on the nature of the data.

Correcting Inconsistent Data

Datasets often contain inconsistent representations of the same information. For example, the gender attribute may be recorded as "Male", "M", "male", or "MALE". Similarly, dates may appear in different formats.

Standardizing these values into a common format improves data consistency and prevents machine learning algorithms from interpreting identical values as different categories.

After completing all these activities, the dataset becomes clean, consistent, and reliable. Such high-quality data forms the foundation for effective machine learning and accurate predictive modeling.

SGOU

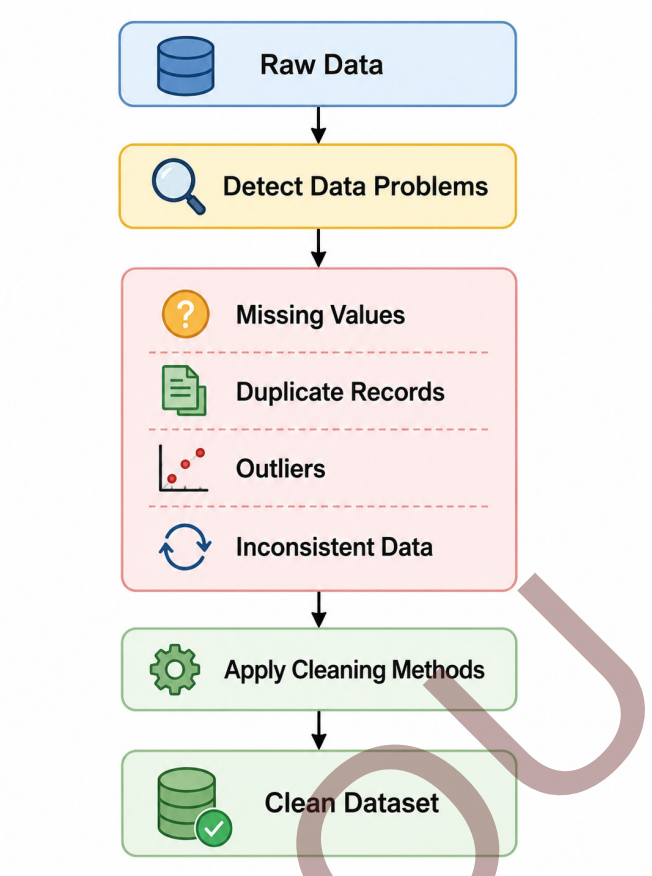


Fig 1.3.2 Data Cleaning Process

Figure 1.3.2 illustrates the data cleaning process. The raw dataset is first examined to identify common data quality problems such as missing values, duplicate records, outliers, and inconsistent data formats. Appropriate cleaning techniques are then applied to correct these issues, resulting in a clean and reliable dataset that is ready for further preprocessing.

Table 1.3.1 Common Data Cleaning Problems and Solutions

Data Quality Issue	Description	Common Solution
Missing Values	Some attribute values are unavailable.	Remove records or replace using mean, median, mode, or interpolation.
Duplicate Records	Same record appears multiple times.	Detect and remove duplicate entries.
Outliers	Extremely large or small values.	Remove, cap, transform, or normalize the values.
Inconsistent Data	Same information represented in different formats.	Standardize values into a common format.

SGOU

1.3.3 Data Transformation

Data transformation is the process of converting cleaned data into a suitable format and structure that can be effectively processed by machine learning algorithms. After completing the data cleaning process, the dataset may still contain values in different formats, categorical information, inconsistent units, or unstructured data. These forms of data cannot be directly processed by most machine learning algorithms. Therefore, transformation is required to convert the data into a standardized and machine-readable format.

Machine learning algorithms generally work with numerical data. However, real-world datasets often contain categorical attributes such as gender, location, product category, and customer type. Data transformation converts such information into a numerical representation while preserving the meaning of the original data. It also helps in reducing complexity, improving consistency, and preparing the dataset for efficient model training.

1.3.3.1 Techniques of Data Transformation

1. Encoding

Encoding is a transformation technique used to convert categorical data into numerical values. Since most machine learning algorithms cannot directly process textual information, categorical attributes must be represented in numerical form.

For example, a gender attribute containing values Male and Female can be converted into numerical values such as:

Gender	Encoded Value
Male	1
Female	0

Another commonly used technique is One-Hot Encoding, where each category is represented as a separate binary column.

For example:

Colour	Red	Blue	Green
Red	1	0	0
Blue	0	1	0
Green	0	0	1

One-hot encoding is useful when categories do not have any natural order.

SGOU

2. Aggregation

Aggregation is the process of combining multiple data values into a single summarized value. It helps reduce the size of the dataset and makes patterns easier to identify.

For example, daily sales records of a company can be combined to generate monthly or yearly sales information. This transformed data helps organizations analyze long-term trends and make better decisions.

3. Discretization

Discretization is the process of converting continuous numerical values into categories or intervals. It simplifies complex numerical data and improves interpretability.

For example, age values can be grouped into categories:

Age Range	Category
0–18	Young
19–60	Adult
Above 60	Senior

Discretization is useful in applications where categories are easier to interpret than exact numerical values.

4. Feature Construction

Feature construction involves creating new features from existing attributes to improve the performance of a machine learning model. A carefully designed feature can provide additional information and help the model discover important patterns.

For example, in a student performance dataset, separate attributes such as internal marks and attendance percentage can be combined to create a new feature such as overall performance score.

5. Normalization and Standardization

Data transformation also includes adjusting the scale and distribution of numerical features. Different attributes may have different units and ranges, which can affect model performance. Techniques such as normalization and standardization convert values into a consistent scale.

For example, age may range from 0 to 100, while income may range from thousands to lakhs. Transforming these values ensures that one feature does not dominate another during model training.

SGOU

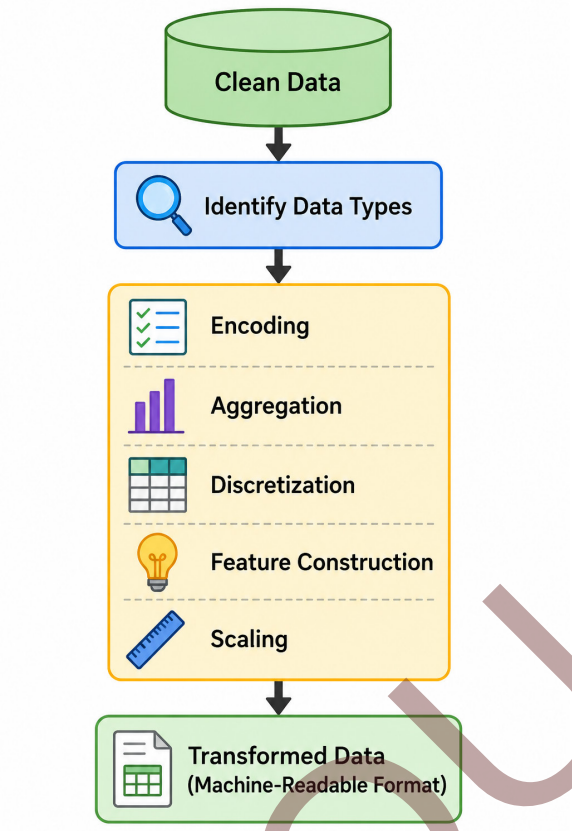


Fig 1.3.3 Data Transformation Process

Figure 1.3.3 represents the process of converting cleaned data into a suitable format for machine learning. Different transformation techniques are applied depending on the type and characteristics of the dataset. The final transformed dataset becomes easier for algorithms to process and analyze.

Consider an online shopping platform that collects customer information such as age, gender, location, purchase history, and spending amount. Before developing a recommendation system, the company needs to transform this data. Gender and location are converted into numerical values using encoding techniques, purchase records are aggregated to identify buying patterns, and spending amounts are normalized to ensure equal contribution of features. After transformation, the dataset becomes suitable for training a machine learning model that can recommend products based on customer preferences.

1.3.4 Data Normalization

Data normalization is a data transformation technique used to adjust the values of numerical features into a common scale without changing the original meaning of the data. In real-world datasets, different attributes may have different ranges and measurement units. For example, a student's age may range from 18 to 60, while annual income may range from thousands to lakhs. When such features are used together in machine learning algorithms, attributes with larger values may dominate the learning process. Normalization helps overcome this problem by bringing all numerical values into a similar range.

SGOU

Normalization improves the performance and accuracy of machine learning models, especially algorithms that depend on distance calculations, such as K-Nearest Neighbour (KNN), Support Vector Machine (SVM), and clustering algorithms. It also helps algorithms converge faster during the training process.

Need for Data Normalization

Machine learning algorithms process numerical values and often assume that all input features contribute equally to the model. However, datasets may contain features with different scales and ranges. For example:

Feature	Range
Age	18–60
Salary	20,000–1,00,000
Experience	1–30 years

In this case, salary values are much larger compared to age and experience. Without normalization, the model may give more importance to salary rather than considering all features equally. Normalization converts these values into a standard range, allowing fair contribution from all attributes.

Techniques of Data Normalization

1. Min-Max Normalization

Min-Max normalization transforms the values of a feature into a fixed range, usually between 0 and 1. It preserves the relationship between the original values while reducing the scale difference.

The formula for Min-Max normalization is:

$$X' = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

Where

- **X'** = Normalized value
- **X** = Original value
- **Xmin** = Minimum value of the feature
- **Xmax** = Maximum value of the feature

SGOU

Example

Consider the age values:

Original Age	Normalized Age
20	0.0
40	0.5
60	1.0

After normalization, all values are converted into the range between 0 and 1.

Applications: Min-Max normalization is commonly used in image processing, neural networks, and algorithms where a fixed input range is required.

2. Z-Score Normalization (Standardization)

Z-score normalization transforms data based on the mean and standard deviation of a feature. It converts the values so that the dataset has a mean of 0 and a standard deviation of 1.

The formula is:

$$X' = \frac{X - \mu}{\sigma}$$

Where:

- **X'** = Standardized value
- **X** = Original value
- **μ** = Mean of the dataset
- **σ** = Standard deviation

Example

If the average salary of employees is $50,000 and an employee's salary is $60,000, standardization represents how far the salary is from the average value.

Z-score normalization is useful when data follows a normal distribution and is commonly applied in regression, classification, and clustering algorithms.

3. Decimal Scaling Normalization

Decimal scaling normalization transforms data by moving the decimal point of values. The number of decimal places shifted depends on the maximum absolute value in the dataset.

The formula is:

$$X' = \frac{X}{10^j}$$

SGOU

Where **j** is the smallest integer such that all normalized values fall within the range of -1 and 1.

Example

Original values:

Original Value	Normalized Value
500	0.5
800	0.8
900	0.9

Decimal scaling reduces the magnitude of values while maintaining their relative differences.

4. Robust Scaling

Robust scaling is a normalization technique that uses the median and interquartile range instead of mean and standard deviation. It is effective when the dataset contains outliers.

The formula is:

$$X' = \frac{X - \text{Median}}{IQR}$$

Where, IQR is Interquartile Range

This method reduces the effect of extreme values and is suitable for datasets containing unusual observations.

1.3.5 Feature Selection

Feature selection is the process of identifying and selecting the most relevant features (attributes or variables) from a dataset for building a machine learning model. Real-world datasets often contain many features, but not all of them contribute equally to the prediction process. Some features may be irrelevant, redundant, or highly correlated with other features. Including such unnecessary features can increase computational cost, reduce model accuracy, and make the model more complex. Feature selection helps overcome these problems by retaining only the important features that influence the target variable.

The primary objective of feature selection is to improve the performance of machine learning models while reducing the dimensionality of the dataset. By eliminating irrelevant features, feature selection reduces training time, improves model interpretability, minimizes overfitting, and enhances prediction accuracy. It is widely used in domains such as healthcare, finance, marketing, image processing, and text mining.

SGOU

1.3.5.1 Techniques of Feature Selection

1. Filter Methods

Filter methods are feature selection techniques t hat evaluate the relevance of each feature using statistical measures before applying any machine learning algorithm. These methods work independently of the learning model and rank features based on their relationship with the target variable. Features with higher scores are selected, while those with lower scores are removed.

Filter methods are computationally efficient and suitable for handling large datasets because they do not require repeated model training. However, they do not consider interactions between features and may sometimes overlook the best combination of attributes.

Common Filter Techniques

- **Correlation Coefficient:** Measures the strength of the relationship between a feature and the target variable. Features with high correlation are usually selected.
- **Chi-Square Test:** Measures the dependence between categorical features and the target variable. Features with higher Chi-Square values are considered more important.
- **Information Gain:** Measures how much information a feature provides about the target variable. Features with higher information gain are selected.
- **Mutual Information:** Measures the amount of information shared between a feature and the target variable, capturing both linear and non-linear relationships.

Example

In a student performance dataset containing Student ID, Name, Attendance, Internal Marks, and Final Grade, Attendance and Internal Marks have a strong relationship with the Final Grade. Student ID and Name have little or no influence and are therefore removed.

2. Wrapper Methods

Wrapper methods evaluate different combinations of features by repeatedly training and testing a machine learning model. Instead of evaluating features individually, wrapper methods determine which subset of features produces the best model performance. The subset with the highest prediction accuracy is selected.

Since multiple models must be trained, wrapper methods are computationally expensive and require more processing time than filter methods. However, they generally provide higher accuracy because they consider feature interactions.

Common Wrapper Techniques

a. Forward Selection

Forward selection begins with an empty set of features. Features are added one at a time based on their contribution to model performance. At each step, the feature that

SGOU

produces the greatest improvement is selected. The process continues until adding more features does not significantly improve the model.

Example

A loan approval dataset contains Age, Income, Occupation, Address, Credit Score, and Marital Status. Initially, no features are selected. Credit Score is added first because it gives the highest accuracy. Income is added next, followed by Age. Address is not included because it does not improve the model.

b. Backward Elimination

Backward elimination starts with all available features. The least important feature is removed one at a time while evaluating model performance after each removal. The process continues until removing any additional feature decreases the model's accuracy.

Example: A medical dataset contains Age, Weight, Blood Pressure, Cholesterol, Patient ID, and Phone Number. Patient ID and Phone Number are removed first because they contribute little to disease prediction.

c.Recursive Feature Elimination (RFE)

Recursive Feature Elimination repeatedly trains a machine learning model and ranks features according to their importance. The least important feature is removed in each iteration until the desired number of features remains.

RFE is commonly used with algorithms such as Support Vector Machines (SVM) and Random Forests.

Example: An employee performance dataset contains ten features. After each iteration, the least important feature is removed until only the five most significant features remain.

3. Embedded Methods

Embedded methods perform feature selection during the model training process. Unlike filter methods, which work before model training, and wrapper methods, which repeatedly train models, embedded methods integrate feature selection directly into the learning algorithm.

These methods provide a balance between computational efficiency and prediction accuracy because feature selection and model training occur simultaneously.

Common Embedded Techniques

a. Decision Tree

Decision Trees automatically select important features by choosing attributes that best split the data at each decision node. Features that contribute more to reducing impurity are considered more important.

SGOU

Example: In a customer purchase dataset, a Decision Tree may first split customers based on Income, indicating that it is the most important feature, followed by Age and Purchase History.

b.Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees. It calculates feature importance by measuring how much each feature improves the predictive performance across all trees. Features with higher importance scores are selected.

Example: In a fraud detection dataset, Transaction Amount and Transaction Location receive higher importance scores than Customer ID.

c. LASSO (Least Absolute Shrinkage and Selection Operator)

LASSO is a regression-based feature selection technique that adds a penalty to the model for using too many features. During training, the coefficients of less important features are reduced toward zero. Features whose coefficients become zero are automatically removed from the model.

LASSO is especially useful for high-dimensional datasets containing many input variables.

Example: A housing price prediction dataset contains 30 features. After applying LASSO, only 12 important features remain, while the remaining 18 features receive zero coefficients and are eliminated.

Table 1.3.2 Comparison of Feature Selection Techniques

Technique	Working Principle	Advantages	Limitations
Filter Method	Uses statistical measures to rank features before model training.	Fast, simple, suitable for large datasets.	Ignores interaction among features.
Wrapper Method	Evaluates different feature subsets using a machine learning model.	High prediction accuracy.	Computationally expensive and time-consuming.
Embedded Method	Performs feature selection during model training.	Balances speed and accuracy.	Depends on the learning algorithm used.

SGOU

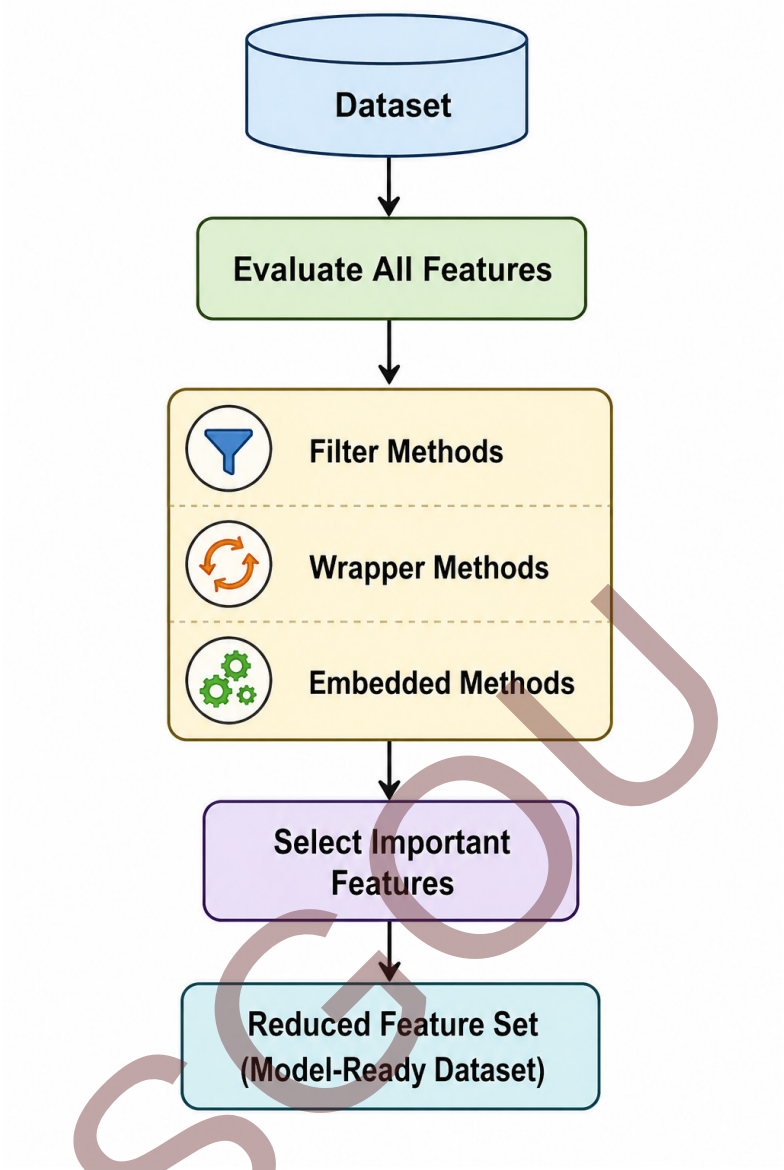


Fig 1.3.5 Feature Selection Process

Figure 1.3.5 illustrates the feature selection process. The dataset is first analyzed to evaluate the importance of each feature. Appropriate feature selection methods are then applied to identify the most relevant attributes. The resulting reduced feature set contains only the important features required for building an efficient machine learning model.

1.3.6 Data Splitting

Data splitting is the process of dividing a dataset into separate subsets for training, validating, and testing a machine learning model. After completing data preprocessing and feature selection, the prepared dataset cannot be directly used for evaluation because doing so may produce misleading results. Instead, the dataset is divided into different parts so that the model can be trained on one subset and evaluated on another unseen subset. This helps measure how well the model performs on new data and prevents overfitting.

SGOU

The primary objective of data splitting is to ensure that the machine learning model learns meaningful patterns from the training data while being evaluated fairly on unseen data. A properly split dataset improves the reliability of model evaluation and helps estimate how well the model will perform in real-world applications. Data splitting is an essential step in supervised machine learning and is widely used in applications such as healthcare, finance, recommendation systems, and image recognition.

Techniques of Data Splitting

1. Train-Test Split

The Train-Test Split is the simplest and most widely used data splitting technique. The dataset is divided into two subsets: the training set and the testing set. The training set is used to train the machine learning model, while the testing set is used to evaluate its performance on unseen data.

A common practice is to allocate 80% of the data for training and 20% for testing, although ratios such as 70:30 or 75:25 are also commonly used depending on the size of the dataset.

Example:

A dataset contains 10,000 customer records.

Dataset	Number of Records
Training Set (80%)	8,000
Testing Set (20%)	2,000

The model learns from the 8,000 training records and its performance is evaluated using the remaining 2,000 testing records.

2. Train-Validation-Test Split

In this technique, the dataset is divided into three subsets:

- **Training Set** – Used to train the model.
- **Validation Set** – Used to tune model parameters and select the best model.
- **Testing Set** – Used for the final evaluation of the trained model.

The validation set helps compare different models or parameter settings without using the testing data. This ensures that the testing set remains completely unseen until the final evaluation.

A commonly used split is:

- Training: 70%
- Validation: 15%
- Testing: 15%

SGOU

Example:

A dataset containing 20,000 student records may be divided as follows:

Dataset	Number of Records
Training Set	14,000
Validation Set	3,000
Testing Set	3,000

The validation set is used to identify the best-performing model before final testing.

3. Cross-Validation

Cross-validation is a technique used to obtain a more reliable estimate of model performance. Instead of dividing the dataset only once, the data is split into multiple equal-sized parts called folds. The model is trained and tested multiple times so that each fold is used once as the testing set while the remaining folds are used for training.

The most common approach is K-Fold Cross-Validation, where the dataset is divided into K equal folds. The final model performance is calculated by averaging the results obtained from all folds.

Example:

In **5-Fold Cross-Validation**, the dataset is divided into five equal parts.

- Iteration 1: Fold 1 → Testing, Remaining → Training
- Iteration 2: Fold 2 → Testing, Remaining → Training
- Iteration 3: Fold 3 → Testing, Remaining → Training
- Iteration 4: Fold 4 → Testing, Remaining → Training
- Iteration 5: Fold 5 → Testing, Remaining → Training

The average accuracy of all five iterations represents the overall model performance.

Cross-validation is especially useful when working with small datasets because it allows every data instance to be used for both training and testing.

4. Stratified Sampling

Stratified sampling is a data splitting technique used when the dataset contains imbalanced classes. Instead of selecting records randomly, the data is divided so that each subset contains approximately the same proportion of each class as the original dataset.

This technique ensures that both the training and testing sets fairly represent all categories, reducing bias during model evaluation.

SGOU

Example:

Suppose a dataset contains:

- 90% Healthy patients
- 10% Disease patients

A random split may result in very few disease cases in the testing set. Using stratified sampling ensures that both the training and testing datasets maintain the same 90:10 class distribution.

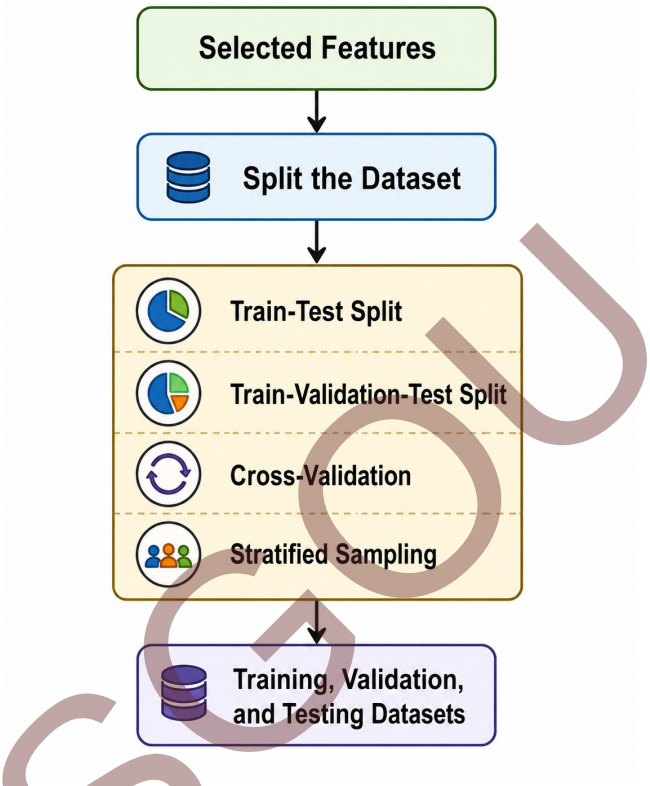


Fig 1.3.6 Data Splitting Process

Figure 1.3.6 illustrates the process of dividing the prepared dataset into different subsets for machine learning. Depending on the dataset size and application, different data splitting techniques are used to create training, validation, and testing datasets. These subsets help train the model, optimize its performance, and evaluate its ability to make accurate predictions on unseen data.

Consider a bank developing a machine learning model to detect fraudulent credit card transactions. After preprocessing and selecting the most relevant features, the dataset is divided into training, validation, and testing subsets. The training set is used to teach the model to distinguish between genuine and fraudulent transactions. The validation set helps select the best model and optimize its parameters. Finally, the testing set evaluates the model using previously unseen transaction records to measure its real-world performance. Since fraudulent transactions are much fewer than genuine ones, stratified sampling is used to ensure that both the training and testing datasets contain the same proportion of fraud cases, resulting in a fair and reliable evaluation.

SGOU

1.3.7 Data Modeling

Data modelling is the process of building a machine learning model using the prepared training data to learn patterns, relationships, and trends. After completing data preprocessing, feature selection, and data splitting, the training dataset is provided to a machine learning algorithm. The algorithm analyzes the input data and learns how different features are related to the target variable. The resulting model can then be used to make predictions or decisions on new, unseen data.

The primary objective of data modelling is to develop a model that accurately captures the underlying patterns in the dataset and generalizes well to new data. Different machine learning algorithms are suitable for different types of problems, such as classification, regression, clustering, and recommendation systems. Selecting an appropriate model depends on the nature of the dataset, the problem being solved, and the desired level of accuracy.

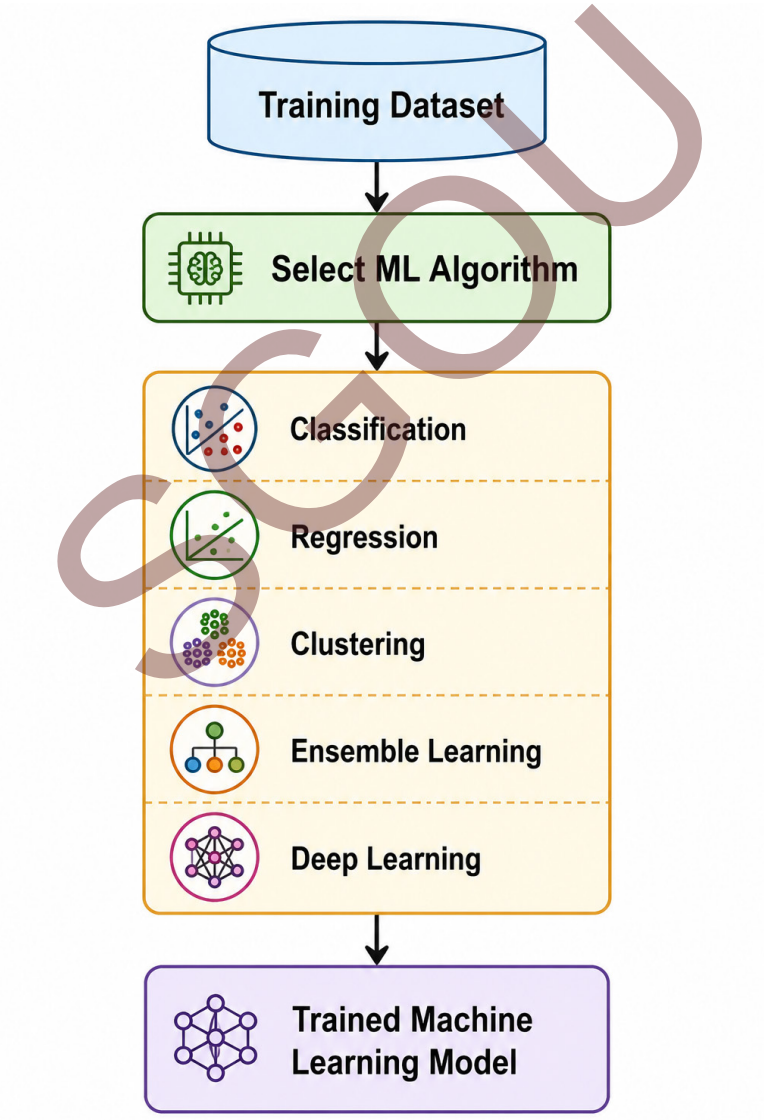


Fig. 1.3.7: Data Modelling Process

Figure 1.3.7 illustrates the data modelling process. The prepared training dataset is provided to a suitable machine learning algorithm based on the problem type. The algorithm learns patterns and relationships from the data to build a trained model. This trained model is then ready for evaluation and prediction using unseen data.

Techniques of Data Modelling

A. Classification Models

Classification models are used when the target variable consists of predefined categories or classes. The objective is to assign each input instance to one of the available classes based on the learned patterns from the training data.

Common classification algorithms include:

- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- Naïve Bayes
- Logistic Regression
- K-Nearest Neighbour (KNN)

1. Decision Tree

A Decision Tree is a flowchart-like algorithm that makes decisions by asking a series of questions about the data. Each question divides the data into smaller groups until a final prediction is reached.

Example:

To predict whether a student will pass or fail, the model may ask:

- Is attendance greater than 75%?
- Are internal marks above 40?
- If yes, predict Pass; otherwise, predict Fail.

2. Random Forest

Random Forest is a collection of many decision trees. Each tree makes its own prediction, and the final result is chosen based on the majority vote. Combining many trees usually produces more accurate and reliable predictions.

Example: If 95 out of 100 trees predict that an email is spam, the final prediction is Spam.

3. Support Vector Machine (SVM)

Support Vector Machine (SVM) separates data into different classes by finding the best boundary between them. It tries to maximize the distance between the classes, making classification more accurate.

SGOU

Example: An SVM separates healthy and diseased patients using a boundary based on medical test results.

4. Naïve Bayes

Naïve Bayes is a probability-based algorithm. It calculates the probability that a data item belongs to each class and predicts the class with the highest probability.

Example: An email containing words such as "Offer" and "Free" has a higher probability of being classified as Spam.

5. Logistic Regression

Logistic Regression predicts the probability that data belongs to a particular class. Based on the probability value, it assigns the data to one of the predefined categories.

Example: A bank predicts whether a customer is likely to repay a loan (Yes or No).

6. K-Nearest Neighbour (KNN)

KNN classifies a new data item by looking at the K most similar data points (nearest neighbours). The new item is assigned to the class that is most common among these neighbours.

Example: If most nearby customers purchased a product, a new customer with similar characteristics is also predicted to purchase it.

Regression Models

Regression models are used when the target variable is continuous rather than categorical. These models predict numerical values by learning the relationship between input features and the target variable.

Common regression algorithms include:

- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Random Forest Regression
- Support Vector Regression (SVR)

1. Linear Regression

Linear Regression predicts a numerical value by fitting a straight line that represents the relationship between input variables and the output.

Example: Predicting a house price based on its area.

2. Polynomial Regression

Polynomial Regression is similar to Linear Regression but fits a curved line instead of a straight line. It is useful when the relationship between variables is not linear.

SGOU

Example: Predicting crop production based on rainfall where the relationship is curved rather than straight.

3. Decision Tree Regression

Decision Tree Regression divides the data into smaller groups using decision rules and predicts a numerical value for each group.

Example: Estimating the selling price of a used car based on its age and mileage.

4. Random Forest Regression

Random Forest Regression combines the predictions of many decision tree regression models and calculates the average result to improve prediction accuracy.

Example: Predicting the market value of a house using multiple decision trees.

5. Support Vector Regression (SVR)

Support Vector Regression predicts numerical values by finding the best-fit line while allowing only small prediction errors.

Example: Predicting future electricity consumption based on historical usage.

C. Clustering Models

Clustering models group similar data instances into clusters without using predefined class labels. Since there are no target labels, clustering is an unsupervised learning technique.

Common clustering algorithms include:

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN

1. K-Means Clustering

K-Means groups similar data into a predefined number (K) of clusters. The algorithm repeatedly assigns data points to the nearest cluster until stable groups are formed.

Example: Grouping customers into five different purchasing categories.

2. Hierarchical Clustering

Hierarchical Clustering builds clusters step by step by combining similar data points into larger groups. The result is a tree-like structure called a dendrogram.

Example: Grouping similar plant species based on their characteristics.

3. DBSCAN

DBSCAN groups data based on the density of data points. It forms clusters where many points are close together and identifies isolated points as outliers.

SGOU

Example: Detecting unusual financial transactions in banking data.

D. Ensemble Models

Ensemble models combine the predictions of multiple machine learning models to improve overall accuracy, robustness, and reliability. Instead of relying on a single model, ensemble methods aggregate the outputs of several models to produce better predictions.

Common ensemble algorithms include:

- Random Forest
- AdaBoost
- Gradient Boosting
- XGBoost

1. Random Forest

Random Forest creates many decision trees using different samples of the data. Each tree makes a prediction, and the final prediction is based on the majority vote (classification) or average (regression).

Example: Predicting whether a loan application should be approved.

2. AdaBoost

AdaBoost builds several simple models one after another. Each new model focuses more on the mistakes made by the previous model, gradually improving overall accuracy.

Example: Improving customer churn prediction.

3. Gradient Boosting

Gradient Boosting also builds models sequentially, but each new model tries to reduce the prediction errors of the previous models using mathematical optimization.

Example: Predicting customer demand in online shopping.

4. XGBoost

XGBoost (Extreme Gradient Boosting) is an advanced and optimized version of Gradient Boosting. It is faster, more accurate, and can efficiently handle large datasets.

Example: Fraud detection in online banking.

E. Deep Learning Models

Deep learning models use artificial neural networks with multiple hidden layers to learn complex patterns from large datasets. These models are particularly effective for tasks involving images, speech, text, and video.

SGOU

Common deep learning models include:

- Artificial Neural Networks (ANN)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)

1. Artificial Neural Network (ANN)

An Artificial Neural Network is made up of interconnected nodes called neurons. It learns by passing data through multiple layers and adjusting connections to improve predictions.

Example: Predicting student performance from attendance and marks.

2. Convolutional Neural Network (CNN)

A CNN is specially designed for image processing. It automatically detects important features such as edges, shapes, and patterns in images.

Example: Recognizing handwritten digits or identifying faces in photographs.

3. Recurrent Neural Network (RNN)

An RNN processes sequential data by remembering information from previous inputs. It is suitable for data where the order of information is important.

Example: Predicting the next word while typing a sentence.

4. Long Short-Term Memory (LSTM)

LSTM is a special type of RNN that can remember information for a longer period. It overcomes the memory limitations of standard RNNs and is widely used for sequence prediction.

Example: Weather forecasting or stock price prediction using historical data.

SGOU

Summerised Overview

Data preprocessing is a fundamental stage in the machine learning pipeline that prepares raw data for model development. Real-world data collected from sources such as databases, sensors, websites, and mobile applications is often incomplete, inconsistent, noisy, or contains redundant information. If such data is directly used for machine learning, it can reduce model accuracy and lead to unreliable predictions. Data preprocessing improves data quality by cleaning, transforming, and organizing the dataset into a suitable format for analysis.

The preprocessing pipeline begins with Data Cleaning, where missing values, duplicate records, outliers, and inconsistencies are identified and corrected to ensure data quality. The cleaned data then undergoes Data Transformation, which converts it into a machine-readable format using techniques such as encoding, aggregation, discretization, and feature construction. Data Normalization follows by scaling numerical attributes to a common range so that all features contribute equally during model training.

After normalization, Feature Selection identifies the most relevant attributes while removing redundant or irrelevant features, reducing dimensionality and improving model efficiency. The selected dataset is then divided into training, validation, and testing datasets through Data Splitting, enabling fair model training and evaluation. Finally, Data Modelling applies appropriate machine learning algorithms, such as classification, regression, clustering, ensemble learning, or deep learning, to learn patterns from the training data and generate predictive models.

Overall, data preprocessing transforms raw and unstructured data into a clean, consistent, and well-organized dataset. By improving data quality and reducing complexity, it enhances the accuracy, efficiency, and reliability of machine learning models, making it one of the most important stages in the machine learning process.

SGOU

Assignments

1. Define data preprocessing and explain its importance.
2. Describe data cleaning techniques in detail.
3. Explain data transformation methods with examples.
4. What is normalization? Explain its methods.
5. Discuss feature selection techniques.
6. Explain data splitting and its significance.
7. Explain data modeling in detail.

Suggested Reading

1. Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

2. Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (3rd ed.). Packt Publishing.

3. McKinney, W. (2022). *Python for Data Analysis* (3rd ed.). O'Reilly Media.

4. Aggarwal, C. C. (2015). *Data Mining: The Textbook*. Springer.

Reference

1. Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.

2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2nd ed.). Springer.

3. Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection*. CRC Press.

4. Tan, P.-N., Steinbach, M., & Kumar, V. (2019). *Introduction to Data Mining* (2nd ed.). Pearson.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

4 UNIT

Evaluation Metrics

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the importance of evaluation metrics in machine learning
- describe the Confusion Matrix and classification evaluation metrics
- differentiate between bias and variance
- distinguish between overfitting and underfitting
- dxplain the concept of regularization and its types

Background

Machine learning models are widely used to make predictions and support decision-making in various fields such as healthcare, finance, education, agriculture, and e-commerce. However, building a machine learning model alone is not sufficient. It is equally important to evaluate how well the model performs and whether it can make accurate predictions for new data. A model that performs well during training may not always produce reliable results in real-world situations. Therefore, understanding how to measure and improve model performance is an essential part of machine learning.

Several evaluation metrics are available to assess the performance of classification models. The Confusion Matrix provides the foundation for calculating important performance measures such as Accuracy, Precision, Recall, and F1-Score. The Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC) help compare the effectiveness of different classification models. In addition, concepts such as bias, variance, overfitting, and underfitting explain the factors that influence model performance. Techniques such as regularization help overcome these challenges by controlling model complexity and improving its ability to generalize to unseen data.

SGOU

Keywords

Confusion Matrix, Accuracy, Precision, Recall, Overfitting

Discussion

In the previous unit, data preprocessing techniques such as data cleaning, transformation, normalization, feature selection, and data splitting were discussed. These techniques help prepare the dataset for building a machine learning model. Once the model is trained, it is important to evaluate its performance to determine how well it makes predictions.

Model evaluation is the process of measuring the performance of a machine learning model. It helps determine whether the model can make accurate predictions and perform well on unseen data. For classification problems, commonly used evaluation metrics include Accuracy, Precision, Recall, F1-Score, Receiver Operating Characteristic (ROC) Curve, and Area Under the Curve (AUC). The Confusion Matrix is the foundation for calculating many of these evaluation metrics.

A machine learning model should perform well not only on the training data but also on new data. Models may suffer from overfitting or underfitting, which affect their prediction accuracy. The concepts of bias and variance help explain these problems, while regularization techniques are used to improve the performance and generalization of machine learning models.

This unit discusses the Confusion Matrix, evaluation metrics such as Accuracy, Precision, Recall, F1-Score, ROC Curve, and AUC, and the concepts of Bias, Variance, Overfitting, Underfitting, and Regularization. Understanding these concepts helps learners evaluate machine learning models and select suitable models for different applications.

1.4.1 Evaluation Metrics

Evaluation metrics are used to measure the performance of a machine learning model. After a model is trained, it is important to evaluate how accurately it makes predictions on new data. These metrics help determine whether the model is suitable for a particular task and allow comparison between different machine learning models.

The choice of an evaluation metric depends on the type of machine learning problem. For classification problems, commonly used evaluation metrics include Accuracy, Precision, Recall, F1-Score, Receiver Operating Characteristic (ROC) Curve, and Area Under the Curve (AUC). These metrics provide different ways of measuring the performance of a classification model.

SGOU

Most classification evaluation metrics are calculated using the Confusion Matrix, which compares the predicted class labels with the actual class labels. The confusion matrix provides the information required to calculate metrics such as Accuracy, Precision, Recall, and F1-Score.

Understanding evaluation metrics helps in selecting the most appropriate machine learning model for a given application and improving its overall performance.

1.4.2 Confusion Matrix

A Confusion Matrix is a table used to evaluate the performance of a classification model. It compares the actual class labels with the predicted class labels generated by the machine learning model. By comparing these values, the confusion matrix provides a clear summary of the prediction results and helps determine how well the model performs.

The confusion matrix is one of the most widely used tools for evaluating classification models. It provides information about the number of correctly classified instances as well as the number of incorrectly classified instances. This information helps in understanding the performance of the model and identifying the types of prediction errors.

A confusion matrix is mainly used for binary classification problems, where the output belongs to one of two classes, such as Positive/Negative, Yes/No, Spam/Not Spam, or Fraud/Not Fraud. It can also be extended to multiclass classification problems.

A confusion matrix consists of four important components:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

These four components form the basis for calculating several evaluation metrics such as Accuracy, Precision, Recall, and F1-Score, which are discussed in the following sections. The general structure of a confusion matrix is shown in Figure 1.4.1.

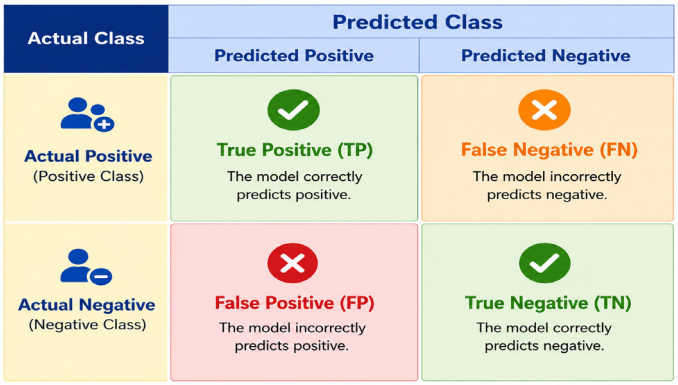


Fig. 1.4.1 Structure of a confusion matrix

SGOU

From the confusion matrix, the number of correct and incorrect predictions made by the classification model can be easily identified. A model with a higher number of True Positives (TP) and True Negatives (TN) and a lower number of False Positives (FP) and False Negatives (FN) is generally considered to have better classification performance.

The four components of the confusion matrix are explained below.

1. True Positive (TP)

A True Positive (TP) is a case in which the machine learning model correctly predicts a positive class. In other words, the actual class is positive, and the predicted class is also positive.

Example:
A medical diagnosis model predicts that a patient has a disease, and the patient is actually suffering from the disease. This is a True Positive prediction.

2. True Negative (TN)

A True Negative (TN) is a case in which the machine learning model correctly predicts a negative class. Here, both the actual class and the predicted class are negative.

Example:
A spam email classifier predicts that an email is not spam, and the email is actually not spam. This is a True Negative prediction.

3. False Positive (FP)

A False Positive (FP) is a case in which the machine learning model incorrectly predicts a negative instance as positive. This type of error is also known as a Type I Error.

Example:
A medical diagnosis model predicts that a healthy person has a disease, even though the person is healthy. This is a False Positive prediction.

4. False Negative (FN)

A False Negative (FN) is a case in which the machine learning model incorrectly predicts a positive instance as negative. This type of error is also known as a Type II Error.

Example:
A medical diagnosis model predicts that a patient does not have a disease, even though the patient is actually suffering from the disease. This is a False Negative prediction.

1.4.3 Accuracy

Accuracy is one of the most commonly used evaluation metrics for measuring the performance of a classification model. It indicates the proportion of correctly classified instances out of the total number of instances. In other words, accuracy measures how often the machine learning model makes correct predictions.

SGOU

Accuracy is calculated using the values obtained from the Confusion Matrix. It considers both correctly predicted positive instances (True Positives) and correctly predicted negative instances (True Negatives).

The formula for calculating accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

where:

- **TP (True Positive):** Correctly predicted positive instances
- **TN (True Negative):** Correctly predicted negative instances
- **FP (False Positive):** Incorrectly predicted positive instances
- **FN (False Negative):** Incorrectly predicted negative instances

Example

Consider a disease prediction model tested on 100 patients. The confusion matrix is given below.

Table 1.4.1 Confusion Matrix for Disease Prediction Model

Actual Class	Predicted Positive	Predicted Negative
Actual Positive	45	5
Actual Negative	10	40

Here,

- TP = 45
- TN = 40
- FP = 10
- FN = 5

Using the accuracy formula,

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$= \frac{45+40}{45+40+10+5}$$

$$= \frac{85}{100} = 0.85$$

Accuracy = 85%

Therefore, the classification model correctly predicts 85 out of 100 instances.

SGOU

1.4.4 Precision

Precision is an evaluation metric used to measure the accuracy of positive predictions made by a classification model. It indicates how many of the instances predicted as positive are actually positive. In other words, precision measures the reliability of the model's positive predictions.

Precision is especially important in applications where false positive predictions should be minimized. For example, in email spam detection, a genuine email incorrectly classified as spam may result in the loss of important information. Therefore, a high precision value indicates that the model makes fewer false positive predictions.

Precision is calculated using the values obtained from the Confusion Matrix.

$$\text{Precision} = \frac{TP}{TP+FP} \times 100\%$$

where:

- **TP (True Positive):** Correctly predicted positive instances
- **FP (False Positive):** Incorrectly predicted positive instances

Example,

Consider the confusion matrix shown in Table 1.4.1. Calculate the Precision of the classification model.

From Table 1.4.1,

- TP = 45
- FP = 10

Using the precision formula,

$$Precision = \frac{TP}{TP+FP}$$

$$= \frac{45}{45+10}$$

$$= \frac{45}{55} = 0.818$$

$$Precision = 81.8\%$$

The precision of the model is 81.8%. This means that 81.8% of the instances predicted as positive are actually positive.

SGOU

1.4.5 Recall

Recall is an evaluation metric used to measure the ability of a classification model to identify all the actual positive instances. It indicates how many of the actual positive instances are correctly predicted by the model. In other words, recall measures the completeness of the model in identifying positive cases.

Recall is particularly important in applications where False Negatives (FN) should be minimized. For example, in disease diagnosis, failing to identify a patient who actually has a disease may lead to serious consequences. Therefore, a high recall value indicates that the model correctly identifies most of the actual positive instances.

Recall is calculated using the values obtained from the Confusion Matrix.

$$\text{Recall} = \frac{TP}{TP + FP} \times 100\%$$

where:

- **TP (True Positive):** Correctly predicted positive instances
- **FN (False Negative):** Incorrectly predicted negative instances

Example

Using the confusion matrix shown in Table 1.4.1, calculate the Recall of the classification model.

From Table 1.4.1,

- TP = 45
- FN = 5

Using the recall formula,

$$Recall = \frac{TP}{TP+FN}$$

$$= \frac{45}{45+5}$$

$$= \frac{45}{50} = 0.90$$

Recall = 90%

The recall of the model is 90%. This means that 90% of the actual positive instances are correctly identified by the classification model.

SGOU

1.4.6 F1-Score

The F1-Score is an evaluation metric that combines Precision and Recall into a single measure. It is used to evaluate the overall performance of a classification model, especially when the dataset is imbalanced. The F1-Score provides a balance between precision and recall by considering both metrics equally.

A high F1-Score indicates that the model has both high precision and high recall. If either precision or recall is low, the F1-Score also becomes low. Therefore, it is considered a more reliable evaluation metric than accuracy when the classes are not equally distributed.

The F1-Score is calculated using the values of Precision and Recall.

$$\text{F1 - Score} = \frac{2\times(Precision \times Recall)}{Precision+Recall} \times 100\%$$

where:

Precision is the proportion of correctly predicted positive instances among all predicted positive instances.

Recall is the proportion of correctly predicted positive instances among all actual positive instances.

Example

Using the values calculated in the previous examples:

Precision = 81.8% = 0.818

Recall = 90% = 0.90

Using the F1-Score formula,

$$\text{F1- Score} = \frac{2\times(Precision \times Recall)}{Precision+Recall} \times 100\%$$

$$= \frac{2\times(0.818 \times 0.90)}{0.818+0.90} \times 100\%$$

$$= \frac{2\times 0.7362}{1.718}$$

$$= \frac{1.4724}{1.718}$$

$$= 0.857$$

F1-Score = 85.7%

The F1-Score of the model is 85.7%. This indicates that the model has a good balance between Precision and Recall, making it suitable for classification tasks where both false positives and false negatives are important.

SGOU

1.4.7 Receiver Operating Characteristic (ROC) Curve

The Receiver Operating Characteristic (ROC) Curve is a graphical tool used to evaluate the performance of a binary classification model. It shows how well a machine learning model can distinguish between positive and negative classes.

Unlike Accuracy, Precision, Recall, and F1-Score, which provide a single numerical value, the ROC Curve provides a graphical representation of the model's performance. It helps compare different classification models and identify the model with better prediction performance.

The ROC Curve is plotted by comparing the True Positive Rate (TPR) with the False Positive Rate (FPR) at different classification thresholds.

1.True Positive Rate (TPR)

The True Positive Rate (TPR), also known as Recall or Sensitivity, measures the proportion of actual positive instances that are correctly identified by the model.

The formula for TPR is:

$$TPR = \frac{TP}{TP + FN}$$

where,

- TP = True Positive
- FN = False Negative

2.False Positive Rate (FPR)

The False Positive Rate (FPR) measures the proportion of actual negative instances that are incorrectly classified as positive.

The formula for FPR is:

$$FPR = \frac{FP}{FP + TN}$$

where,

- FP = False Positive
- TN = True Negative

Characteristics of the ROC Curve

- The X-axis represents the False Positive Rate (FPR).
- The Y-axis represents the True Positive Rate (TPR) or Recall.
- Each point on the ROC curve represents the model's performance at a different classification threshold.
- A curve closer to the top-left corner indicates better classification performance.
- The diagonal line represents the performance of a random classifier.

SGOU

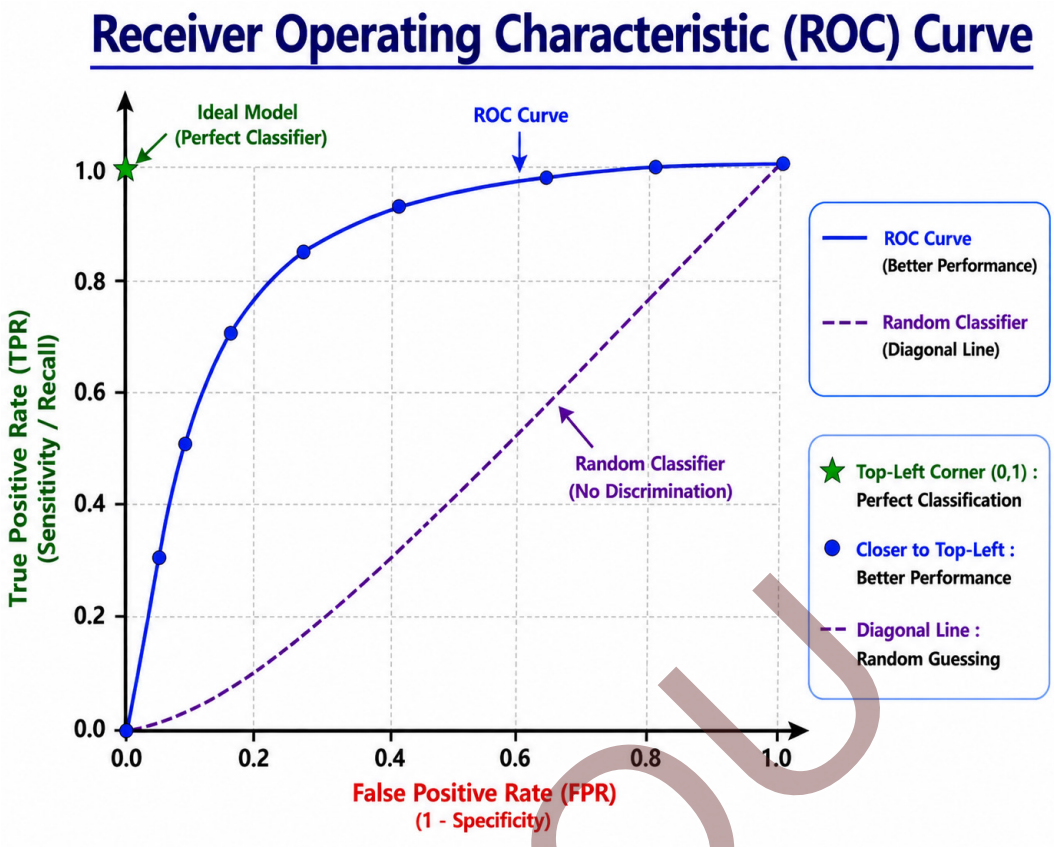


Fig. 1.4.2 ROC Characteristic Curve

Figure 1.4.2 shows the Receiver Operating Characteristic (ROC) Curve of a classification model.

The horizontal axis (X-axis) represents the False Positive Rate (FPR), while the vertical axis (Y-axis) represents the True Positive Rate (TPR) or Recall.

The Solid curve represents the ROC curve of the classification model. Each point on the curve corresponds to a different classification threshold. As the threshold changes, the values of TPR and FPR also change, producing different points on the ROC curve.

The dashed diagonal line represents the performance of a random classifier. If a model's ROC curve lies close to this line, its predictions are almost the same as random guessing.

The star at the top-left corner (0,1) represents the ideal classifier. At this point, the model correctly identifies all positive instances (TPR = 1) without making any false positive predictions (FPR = 0).

A good classification model has an ROC curve that bends towards the top-left corner. The closer the curve is to this corner, the better the model can distinguish between positive and negative classes.

SGOU

Interpretation of the ROC Curve

The ROC Curve can be interpreted as follows:

- A curve closer to the top-left corner indicates excellent model performance.
- A curve lying near the diagonal line indicates that the model performs like random guessing.
- The farther the ROC curve is above the diagonal line, the better the model distinguishes between the two classes.

1.4.8 Area Under the Curve (AUC)

The Area Under the Curve (AUC) is a performance measure used to evaluate a binary classification model. It represents the area under the Receiver Operating Characteristic (ROC) Curve. Unlike the ROC Curve, which provides a graphical representation of model performance, the AUC summarizes the performance into a single numerical value.

The AUC value indicates how well a model can distinguish between positive and negative classes. A higher AUC value means the model has a better ability to classify instances correctly.

The value of AUC always lies between 0 and 1.

- **AUC = 1.0** indicates a **perfect classifier**.
- **AUC = 0.5** indicates a **random classifier**, which performs no better than random guessing.
- **AUC < 0.5** indicates that the model performs worse than random guessing.

Thus, a classification model with a higher AUC value is generally considered to have better predictive performance.

Interpretation of AUC Values

Table 1.4.2 shows the interpretation of different AUC values.

Table 1.4.2 Interpretation of AUC Values

AUC Value	Model Performance
1.0	Perfect classifier
0.9 – 0.99	Excellent
0.8 – 0.89	Good
0.7 – 0.79	Fair
0.6 – 0.69	Poor
0.5	Random classifier

SGOU

Example,

Consider two machine learning models evaluated using the ROC Curve.

- **Model A:** AUC = **0.95**
- **Model B:** AUC = **0.78**

Since **0.95 > 0.78**, **Model A** has better classification performance because it can distinguish between positive and negative classes more accurately.

1.4.9 Bias and Variance

Bias and Variance are two important concepts used to evaluate the performance of a machine learning model. They help us understand why a model performs well or poorly on new data.

A good machine learning model should learn the important patterns from the training data without memorizing them. If a model is too simple, it may not learn enough from the data. On the other hand, if it is too complex, it may learn unnecessary details and perform poorly on unseen data. Bias and Variance help explain these situations.

A successful machine learning model maintains a balance between Bias and Variance so that it can make accurate predictions on new data.

1.4.9.1 Bias

Bias is the error that occurs when a machine learning model is too simple to learn the underlying patterns in the data. Such a model does not learn enough from the training data and therefore makes inaccurate predictions.

A model with high bias gives similar predictions for different inputs because it ignores important information in the data. As a result, it performs poorly on both the training data and the testing data. This problem is known as underfitting.

Characteristics of High Bias

- The model is too simple.
- Learn only a few patterns from the training data.
- Ignore important features.
- Produces high prediction errors.
- Results in underfitting.

SGOU

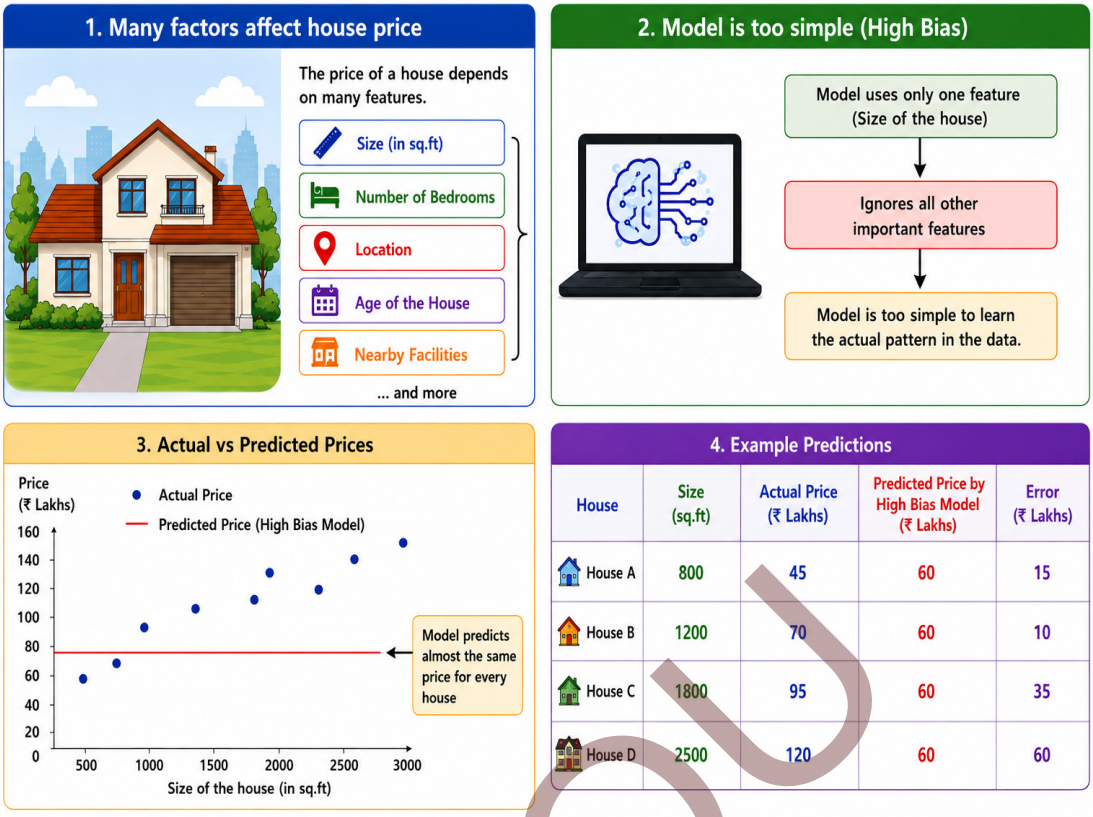


Fig. 1.4.3 Example of High Bias in Machine Learning

The Figure 1.4.3 explains the concept of high bias using a house price prediction problem.

In this example, the objective is to predict the price of a house. Normally, house prices depend on several factors such as size of the house, number of bedrooms, location, age of the house, and nearby facilities. All these features together help the model make accurate predictions.

In Step 1, the figure shows the different features that influence the price of a house.

In Step 2, the machine learning model is too simple because it uses only one feature, the size of the house and ignores all the other important features. Since the model does not consider complete information, it cannot learn the actual relationship between the input features and the house price.

In Step 3, the graph compares the actual house prices with the predicted prices. The blue points represent the actual prices of different houses, while the red horizontal line represents the predictions made by the model. It can be observed that the model predicts almost the same price for every house, regardless of its actual value. This indicates that the model has failed to learn the data properly.

In Step 4, the prediction table further confirms this behaviour. Although the actual prices of the houses are different, the model predicts ₹60 lakhs for all the houses. This results in large prediction errors, especially for houses with much higher or lower actual prices.

SGOU

This example clearly shows that a high bias model is too simple. It ignores important information, fails to learn the underlying patterns in the training data, and therefore produces poor predictions. Such a model suffers from underfitting.

1.4.9.2 Variance

Variance is the error that occurs when a machine learning model is too complex and learns not only the important patterns but also the unnecessary details (noise) present in the training data. As a result, the model performs very well on the training data but fails to make accurate predictions on new or unseen data.

A model with high variance is highly sensitive to changes in the training data. Even a small change in the input data can lead to different predictions. This problem is known as overfitting.

Characteristics of High Variance

- The model is too complex.
- Learn both patterns and noise from the training data.
- Performs very well on training data.
- Performs poorly on unseen (testing) data.
- Results in overfitting.

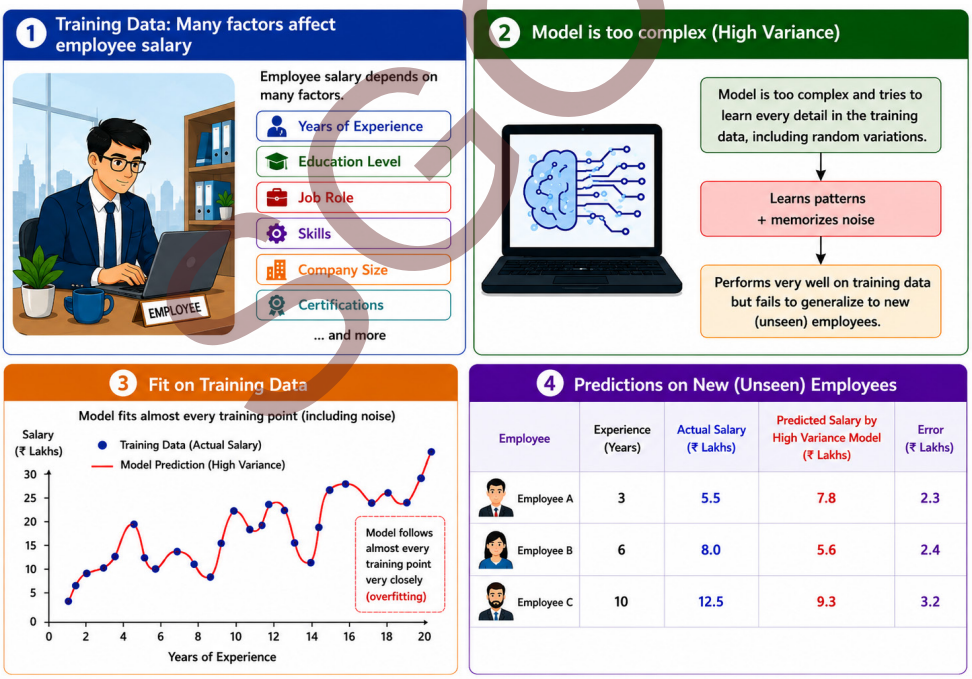


Fig. 1.4.4 Example of High Variance in Employee Salary Prediction here

Figure 1.4.4 illustrates the concept of high variance using an employee salary prediction problem.

In Step 1, the training dataset contains information about different employees, including their years of experience, education level, job role, skills, company size, and certifications. These features are used to predict the salary of each employee.

In Step 2, the machine learning model becomes too complex and attempts to learn every detail in the training data, including random variations and noise. Instead of learning the general relationship between employee characteristics and salary, the model memorizes the training data.

In Step 3, the graph shows that the prediction curve passes very closely through almost every training data point. Although this results in excellent performance on the training data, the prediction curve becomes highly irregular and fails to represent the actual relationship between years of experience and salary.

In Step 4, when the trained model is used to predict the salaries of new employees, the prediction errors become large because the model cannot generalize well to unseen data.

This example demonstrates that a high variance model memorizes the training data instead of learning the underlying pattern. Consequently, it performs exceptionally well on the training data but poorly on new data, resulting in overfitting.

1.4.10 Overfitting

Overfitting is a condition in which a machine learning model learns the training data too closely. Instead of learning only the important patterns, it also learns the random noise and unnecessary details present in the training data. As a result, the model performs very well on the training data but gives poor results when tested with new or unseen data.

An overfitted model has high training accuracy but low testing accuracy because it cannot generalize its learning to new data. Overfitting usually occurs when the model is too complex or when it is trained for a long time.

For example, consider a machine learning model developed to predict house prices. During training, the model learns the prices of the houses in the training dataset very accurately. However, instead of learning the general relationship between house features and price, it memorizes the training data.

When the model is used to predict the price of a new house, its prediction may be incorrect because the new house is different from those in the training dataset. Therefore, the model performs poorly on unseen data. This situation is known as overfitting.

Characteristics of Overfitting

- The model is too complex and tries to learn every detail present in the training data.
- Learn both useful patterns and noise, instead of learning only the important relationships in the data.
- Performs exceptionally well on the training data because it memorizes the training examples.

SGOU

- Produces poor results on testing or unseen data, as it cannot generalize to new data.
- Has high variance, meaning small changes in the input data can significantly affect the model's predictions.
- Shows poor generalization ability, making it less reliable for real-world applications.

Effects of Overfitting

- Overfitting can reduce the effectiveness of a machine learning model in the following ways:
- Poor performance on unseen data, resulting in inaccurate predictions.
- Low generalization ability, making the model unsuitable for new data.
- High prediction errors when tested with unseen data.
- Unreliable performance in real-world applications.
- Sensitive to small changes in the input data.

1.4.11 Underfitting

Underfitting is a condition in which a machine learning model is too simple to learn the important patterns present in the training data. Since the model does not learn enough from the data, it produces poor predictions for both the training data and new or unseen data.

An underfitted model has low training accuracy as well as low testing accuracy because it fails to capture the relationship between the input features and the output. Underfitting usually occurs when the model is too simple or when it is not trained sufficiently.

For example, consider a machine learning model developed to predict house prices. During training, the model uses only the size of the house and ignores other important features such as location, number of bedrooms, age of the house, and nearby facilities.

Since the model considers only limited information, it cannot learn the actual relationship between the house features and the price. As a result, it produces inaccurate predictions not only for the training data but also for new houses. This situation is known as underfitting.

Characteristics of Underfitting

- The model is too simple.
- Learn only a few patterns from the training data.
- Ignores important features in the dataset.
- Performs poorly on both training and testing data.

SGOU

- Has poor generalization ability.
- Usually associated with high bias.

Effects of Underfitting

Underfitting affects the performance of a machine learning model in the following ways:

- Produces poor predictions on both training and testing data.
- Increases prediction errors.
- Reduces the overall accuracy of the model.
- Fails to learn the important patterns in the data.

1.4.12 Regularization

Regularization is a technique used in machine learning to reduce overfitting by controlling the complexity of a model. It prevents the model from learning unnecessary details or noise present in the training data and helps it focus on the important patterns.

A machine learning model should perform well not only on the training data but also on new or unseen data. However, if the model becomes too complex, it may memorize the training data instead of learning the actual relationship between the input features and the output. As a result, the model performs well on the training data but poorly on unseen data. This problem is known as overfitting.

Regularization addresses this problem by limiting the complexity of the model. It encourages the model to learn the essential patterns in the data while ignoring unnecessary details or noise. Consequently, the model becomes more reliable and is able to make accurate predictions on new data.

Need for Regularization

- Regularization is important because it:
- Reduces overfitting.
- Controls the complexity of the model.
- Improves the model's ability to generalize.
- Produces more reliable predictions.
- Improves performance on unseen data.

SGOU

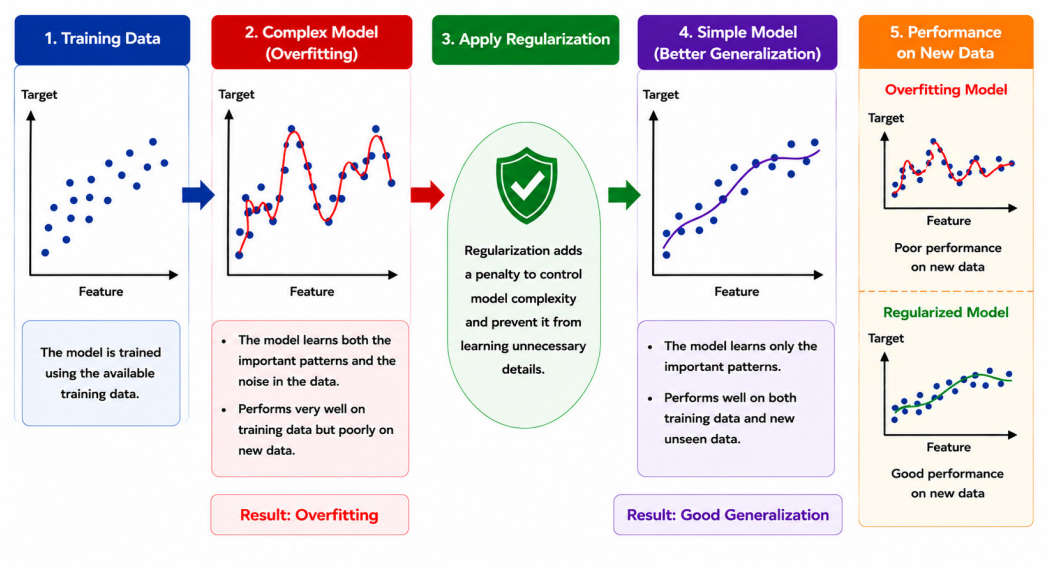


Fig.1.4.5 Effect of Regularization on Model Performance

1.4.12.1 Types of Regularization

Regularization techniques are used to reduce overfitting by controlling the complexity of a machine learning model. The two most commonly used types of regularization are L1 Regularization and L2 Regularization. Both techniques add a penalty during model training to prevent the model from becoming too complex. However, they differ in the way the penalty is applied and how they affect the model.

Types of Regularization

- L1 Regularization (Lasso Regression)
- L2 Regularization (Ridge Regression)

1. L1 Regularization (Lasso Regression)

L1 Regularization, also known as Lasso Regression, adds a penalty based on the absolute values of the model weights during training.

This penalty reduces the importance of less useful features. In some cases, the weights of these features become zero, which means they are automatically removed from the model. As a result, the model becomes simpler and focuses only on the most important features.

Characteristics of L1 Regularization

- Uses the absolute value of the weights as the penalty.
- Can reduce some weights to zero.
- Removes less important features.
- Produces a simpler model.
- Useful for feature selection.

SGOU

2. L2 Regularization (Ridge Regression)

L2 Regularization, also known as Ridge Regression, adds a penalty based on the square of the model weights during training.

Instead of removing features, L2 Regularization reduces the values of the weights. This prevents any single feature from having too much influence on the model while keeping all the features. As a result, the model becomes more stable and performs better on unseen data.

Characteristics of L2 Regularization

- Uses the square of the weights as the penalty.
- Reduces the values of large weights.
- Keeps all features in the model.
- Produces a stable model.
- Improves generalization.

Summerised Overview

Machine learning models must be evaluated to determine how accurately and reliably they make predictions before they are applied to real-world problems. Evaluation metrics provide a systematic way to assess model performance and compare different classification models. The Confusion Matrix summarizes prediction results using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values, which form the basis for calculating important evaluation metrics. Accuracy measures the overall correctness of predictions, Precision measures the proportion of correctly predicted positive instances, Recall measures the ability of a model to identify actual positive instances, and the F1-Score provides a balanced measure by combining precision and recall. The Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC) are used to evaluate the performance of classification models and compare their ability to distinguish between different classes. The concepts of bias and variance explain the two major sources of prediction error in machine learning models. High bias results in underfitting, where the model fails to learn the underlying patterns in the data, whereas high variance results in overfitting, where the model memorizes the training data and performs poorly on unseen data. Understanding the characteristics, causes, effects, and methods for reducing overfitting and underfitting helps in developing models with better prediction capability. Regularization is an important technique used to control model complexity and reduce overfitting. The commonly used regularization methods, L1 (Lasso) Regularization and L2 (Ridge) Regularization, improve the generalization ability of machine learning models by controlling feature weights. Together, these concepts provide a strong foundation for evaluating classification models, interpreting their performance, identifying learning problems, and developing accurate, reliable, and efficient machine learning solutions.

SGOU

Assignments

1. Explain the Confusion Matrix with a neat diagram. Describe how Accuracy, Precision, Recall, and F1-Score are calculated and discuss their significance in evaluating classification models.

2. Explain the Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC). Discuss how they are used to evaluate and compare the performance of classification models.

3. Explain the concepts of Bias and Variance in machine learning. Discuss how they influence model performance with suitable examples.

4. Differentiate between Overfitting and Underfitting. Illustrate your answer with suitable examples.

5. What is Regularization? Explain the need for regularization and discuss the different types of regularization.

Suggested Reading

1. James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: With Applications in Python*. Springer, 2023.

2. Lindholm, Andreas, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

3. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. 2nd ed., Springer, 2021.

SGOU

Reference

1. Mitchell, Tom M., and Vineeth N. Balasubramanian. *Machine Learning*. 2nd ed., McGraw Hill, 2026.

2. Murphy, Kevin P. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

3. Bishop, Christopher M., and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2024.

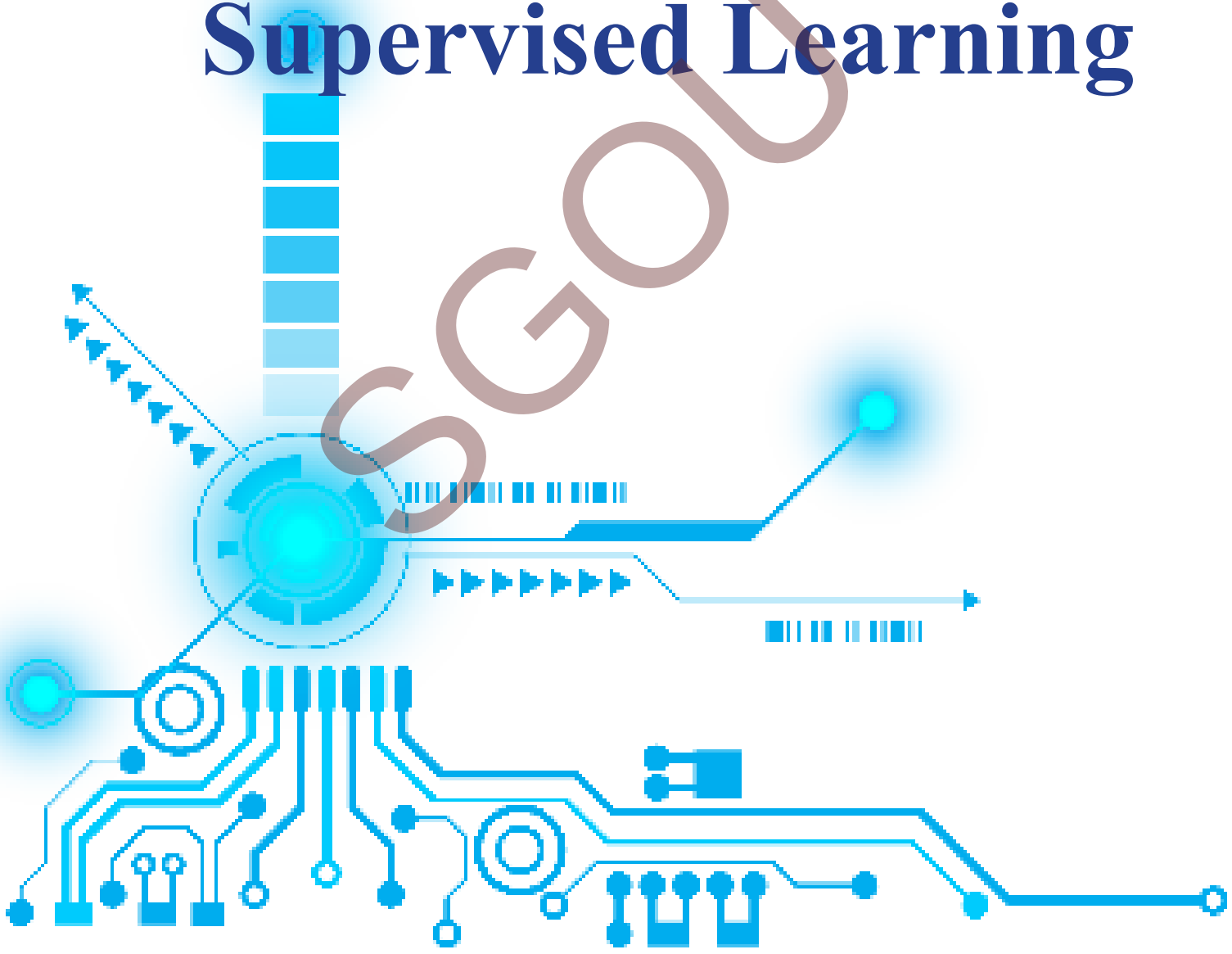
Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

BLOCK 2
Supervised Learning

SGOU



1 UNIT Classification Algorithms

Learning Outcomes

After completion of this unit, the learner will be able to:

- define the concept of classification in supervised machine learning
- describe the working of a classification model
- identify the main classification algorithms: k-NN, Decision Tree, Naive Bayes, SVM, and Ensemble Methods
- explain the basic working of the k-NN, Decision Tree, and Naive Bayes algorithms

Background

Imagine receiving hundreds of emails and having to decide which ones are Spam and which ones are Not Spam. A bank may need to identify whether a transaction is Fraudulent or Legitimate, while an online platform may need to determine the category of a new product or message. In all these situations, the system needs to make a classification decision.

But how can a machine learning system make such decisions?

The process begins with labelled data. In supervised learning, a model is provided with examples for which the correct class is already known. By studying these examples, the model identifies patterns and relationships between the input features and their corresponding class labels. The trained model can then use these learned patterns to assign an appropriate class to new or unseen data.

In this unit, you will study the concept of classification and understand how classification models make predictions. You will study important classification algorithms such as k-Nearest Neighbour (k-NN), Decision Trees, Naive Bayes,

SGOU

Support Vector Machine (SVM), and Ensemble Methods. You will see how k-NN uses neighbouring data points, how a Decision Tree makes decisions through a sequence of conditions, and how Naive Bayes uses probability for classification. You will also be introduced to SVM and Ensemble Methods, while their detailed concepts and techniques will be discussed in a later unit.

Through simple examples, illustrations, and step-by-step explanations, this unit will help you understand how different classification algorithms use data to arrive at a class decision and provide a foundation for studying more advanced classification techniques.

Keywords

Classification , Supervised Learning, k-Nearest Neighbour (k-NN), Decision Tree, Naive Bayes

Discussion

2.1.1 Introduction to Classification

Classification is one of the most widely used techniques in supervised machine learning. It is used to predict the category or class to which a given data item belongs. In classification, a machine learning model learns from a set of labelled training data, where each data instance is associated with a predefined class label. After learning the relationship between the input features and the corresponding class labels, the model predicts the class of new or unseen data.

The output of a classification model is a predefined category or class label. For example, a classification model can determine whether an email is spam or not spam, whether a bank transaction is fraudulent or legitimate, or whether a medical test result is positive or negative.

During the training process, the classification algorithm identifies patterns and relationships in the labelled data. These learned patterns are then used to classify new data accurately. Classification plays an important role in many real-world applications, including healthcare, finance, e-commerce, cybersecurity, and image recognition, where accurate decision-making is essential.

SGOU

Examples of Classification

S. No.	Application	Input	Predicted Class
1.	Email Filtering	Email message	Spam / Not Spam
2.	Disease Diagnosis	Patient symptoms and test results	Disease Present / Disease Absent
3.	Loan Approval	Applicant details (income, credit score, employment, etc.)	Approved / Rejected
4.	Sentiment Analysis	Customer review or feedback	Positive / Negative / Neutral
5.	Fraud Detection	Transaction details (amount, location, time, etc.)	Fraudulent / Legitimate
6.	Face Recognition	Facial image	Person Identified / Unknown

Fig. 2.1.1 Examples of classification

2.1.2 Working of a Classification Model

A classification model follows a series of steps to predict the class label of new data. Initially, a labelled dataset containing input features and their corresponding class labels is collected. The selected classification algorithm learns the relationship between the input features and the class labels during the training process. After training, the model is tested using new or unseen data. Based on the patterns learned during training, the model predicts the appropriate class for each new data instance.

Steps Involved

1. Collect labelled training data.
2. Select a suitable classification algorithm.
3. Train the classification model.
4. Provide new or unseen data.
5. Predict the class label.
6. Evaluate the model performance.

SGOU

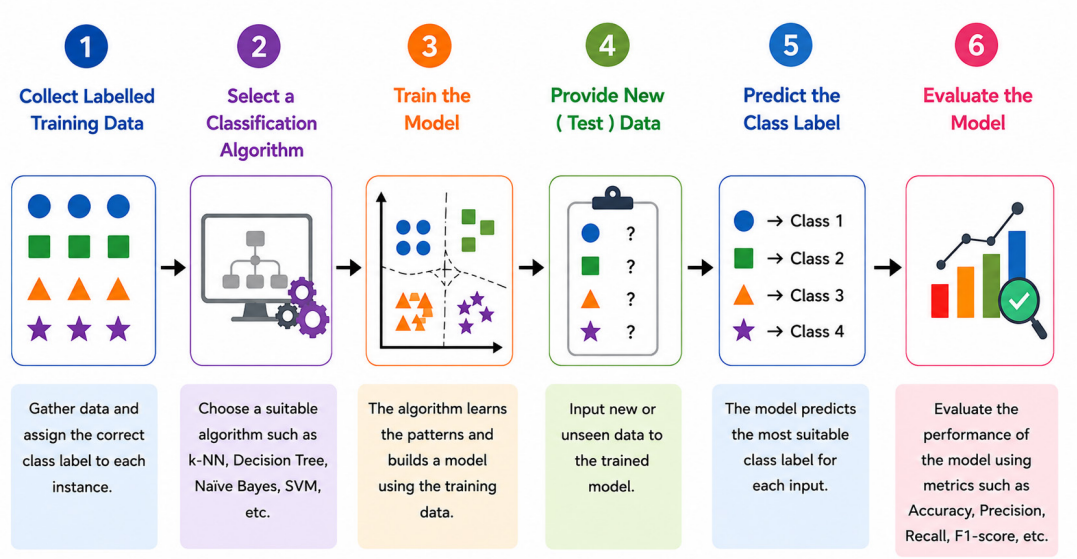


Fig. 2.1.2 working of a classification model

Figure 2.1.2 illustrates the basic workflow of a classification model. The process begins with labelled training data, which is used to train a classification algorithm. During training, the algorithm learns the relationship between the input features and the corresponding class labels and builds a classification model. When new or unseen data is provided, the trained model analyzes its features and predicts the most appropriate class. Finally, the model's predictions can be evaluated to determine its accuracy and reliability.

2.1.3 Classification Algorithms

Classification algorithms are supervised machine learning techniques used to assign data instances to predefined categories or classes based on patterns learned from labelled training data. Different classification algorithms adopt different learning strategies to perform classification tasks. Figure 2.1.3 provides an overview of the major classification algorithms covered in this block. In this unit, the discussion focuses on k-Nearest Neighbour (k-NN), Decision Trees, and Naïve Bayes, while Support Vector Machine (SVM) and Ensemble Methods are introduced here as part of the overall classification framework and are discussed in detail in Unit 3.

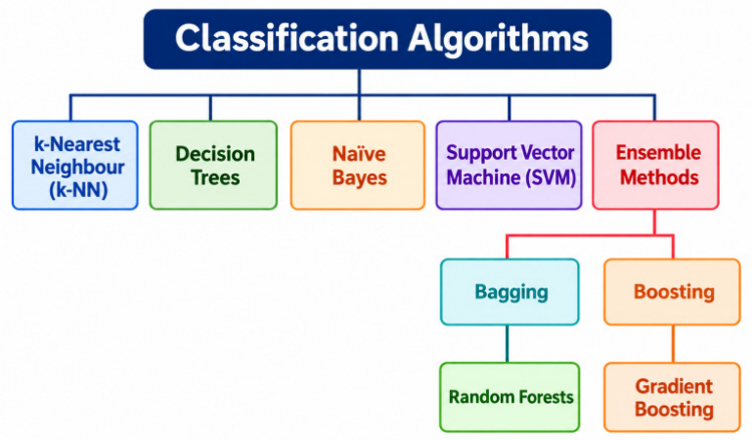


Fig. 2.1.3 classification algorithms

SGOU

2.1.3.1 k-Nearest Neighbour (k-NN) algorithm

The k-Nearest Neighbour (k-NN) algorithm is one of the simplest and most widely used supervised machine learning algorithms for solving classification problems. It classifies a new data instance by comparing it with previously labelled data stored in the training dataset. The algorithm is based on the principle that similar data instances are likely to belong to the same class.

The letter 'k' represents the number of nearest neighbours considered for making a prediction. When a new data instance is given as input, the algorithm identifies the k closest neighbours from the training dataset using a suitable distance measure. The new data instance is then assigned to the class that appears most frequently among these neighbours.

Unlike many machine learning algorithms, the k-NN algorithm does not build a model during the training phase. Instead, it stores the labelled training data and uses it to classify new data whenever a prediction is required. Since learning takes place only at the time of prediction, k-NN is known as a lazy learning or instance-based learning algorithm.

To identify the nearest neighbours, the algorithm calculates the distance between the new data point and the data points in the training dataset. The Euclidean distance is the most commonly used distance measure for this purpose. The performance of the k-NN algorithm depends on factors such as the value of k, the distance measure used, and the quality of the training data.

Characteristics of k-Nearest Neighbour (k-NN)

The important characteristics of the k-NN algorithm are:

- It is a supervised learning algorithm.
- It is mainly used for classification problems.
- It classifies data based on the nearest neighbouring instances.
- It stores the training data instead of building a model.
- It is a lazy learning and instance-based learning algorithm.
- It predicts the class using majority voting among the nearest neighbours.
- It is simple to understand and easy to implement.

Working of the k-Nearest Neighbour (k-NN) Algorithm

The k-Nearest Neighbour (k-NN) algorithm classifies a new data instance by comparing it with the labelled data available in the training dataset. It identifies the k nearest neighbours based on the distance between the new data point and the existing data points. The class that appears most frequently among these neighbours is assigned to the new data instance. The working of the k-NN algorithm is explained below.

Step 1: Prepare the Training Dataset

Collect a labelled training dataset in which each data instance belongs to a predefined class. The dataset consists of input features and their corresponding class labels. This

SGOU

dataset serves as the reference for classifying new data.

Step 2: Select the Value of k

Choose the value of k, which represents the number of nearest neighbours considered for classification. The value of k is selected before the prediction process begins. Common values are 3, 5, or 7. Choosing an appropriate value of k is important because it directly affects the accuracy of the classification.

Step 3: Calculate the Distance

When a new data instance is provided, the algorithm calculates its distance from every data point in the training dataset. The Euclidean distance is the most commonly used distance measure because it computes the straight-line distance between two data points.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where,

d = Euclidean distance

(x_1, y_1) = Coordinates of the first data point

(x_2, y_2) = Coordinates of the second data point

The smaller the distance, the more similar the two data points are.

Step 4: Identify the k Nearest Neighbours

Arrange all the calculated distances in ascending order and select the k data points with the smallest distances. These selected data points are called the k nearest neighbours because they are closest to the new data instance.

Step 5: Perform Majority Voting

Observe the class labels of the selected neighbours. Count the number of neighbours belonging to each class. The class with the highest frequency (majority) among the selected neighbours is considered the predicted class.

Step 6: Assign the Class Label

Assign the majority class obtained from the voting process to the new data instance. The new data point is now classified into one of the predefined classes.

SGOU

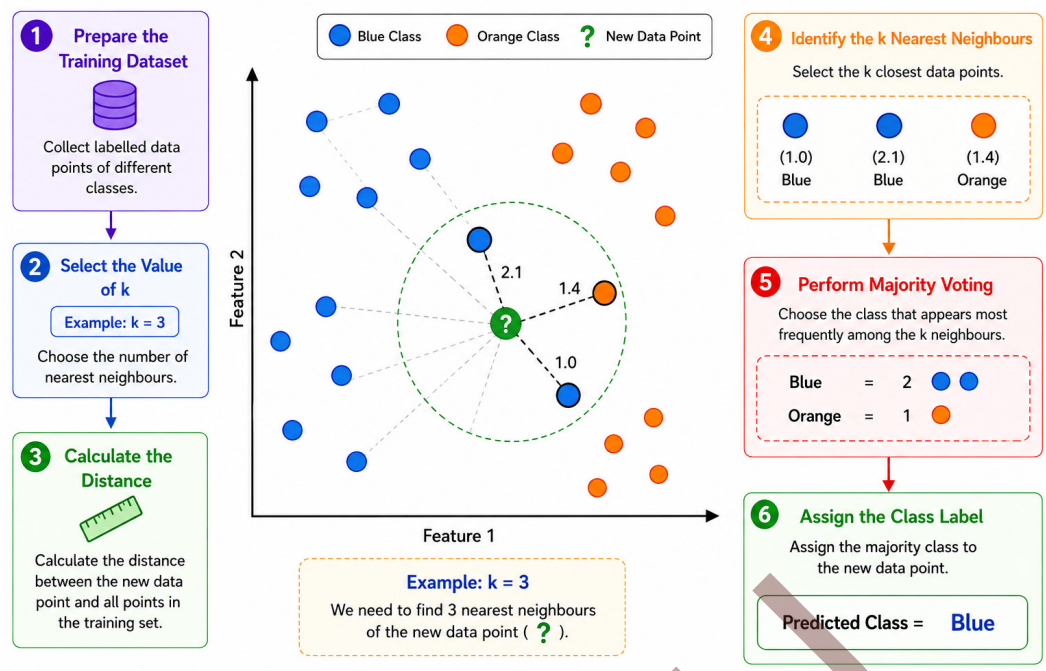


Fig. 2.1.4 working of k-Nearest Neighbour algorithm

Figure 2.1.4 illustrates the step-by-step working of the k-Nearest Neighbour (k-NN) algorithm. The process begins by preparing a labelled training dataset, followed by selecting an appropriate value of k, which specifies the number of nearest neighbours to be considered for classification. When a new data point is introduced, the algorithm calculates its distance from every data point in the training dataset using a suitable distance measure, such as the Euclidean distance. It then identifies the k nearest neighbours based on the smallest distance values. The class labels of these neighbours are examined, and majority voting is performed to determine the most frequent class. Finally, the new data point is assigned to this majority class. In the illustrated example, the three nearest neighbours consist of two Blue data points and one Orange data point. Since the majority of the neighbours belong to the Blue class, the new data point is classified as Blue. This simple procedure enables the k-NN algorithm to classify new data accurately based on the similarity between data instances.

2.1.3.2 Decision Tree

A Decision Tree is one of the most widely used supervised machine learning algorithms for classification and regression tasks. It is commonly used for classification because it is simple to understand, easy to interpret, and represents the decision-making process in a visual manner.

The algorithm is called a Decision Tree because its structure resembles an upside-down tree. Starting from a root node, the algorithm makes a series of decisions based on the values of different features. Each decision creates a branch, leading to another decision or to a final outcome. This process continues until a leaf node is reached, which represents the predicted class.

SGOU

A Decision Tree classifies a new data instance by asking a sequence of questions about its features and following the corresponding branches. During training, the algorithm repeatedly selects the most suitable feature to divide the dataset into smaller and more homogeneous groups. This process continues until the data in each group belongs to a single class or a stopping condition is reached.

Components of a Decision Tree

A Decision Tree is composed of several components that work together to classify data. Each component has a specific role in the decision-making process. Understanding these components helps in interpreting how a Decision Tree makes predictions.

1. Root Node

The root node is the topmost node of the Decision Tree and represents the entire training dataset. It is the starting point of the decision-making process. The algorithm selects the most suitable feature at the root node to divide the dataset into smaller subsets.

2. Decision Node

A decision node is an internal node that represents a test or condition on a particular feature. Based on the outcome of the test, the data is divided into different branches. A Decision Tree may contain multiple decision nodes, each helping to further classify the data.

3. Branch

A branch is a connection between two nodes. It represents the outcome of a decision or test performed at a decision node. Each branch directs the data to the next decision node or to a leaf node based on the feature value.

4. Leaf Node

A leaf node, also known as a terminal node, is the final node of the Decision Tree. It represents the predicted class or final decision. Once a leaf node is reached, no further splitting or decision-making takes place.

Working of Decision Tree algorithm

A Decision Tree classifies a new data instance by evaluating a sequence of conditions from the root node to a leaf node. At each decision node, the algorithm tests the value of a feature and follows the appropriate branch based on the outcome of the test. The process continues until it reaches a leaf node, which represents the final decision or predicted class. The working of a Decision Tree can be understood using the following example.

SGOU

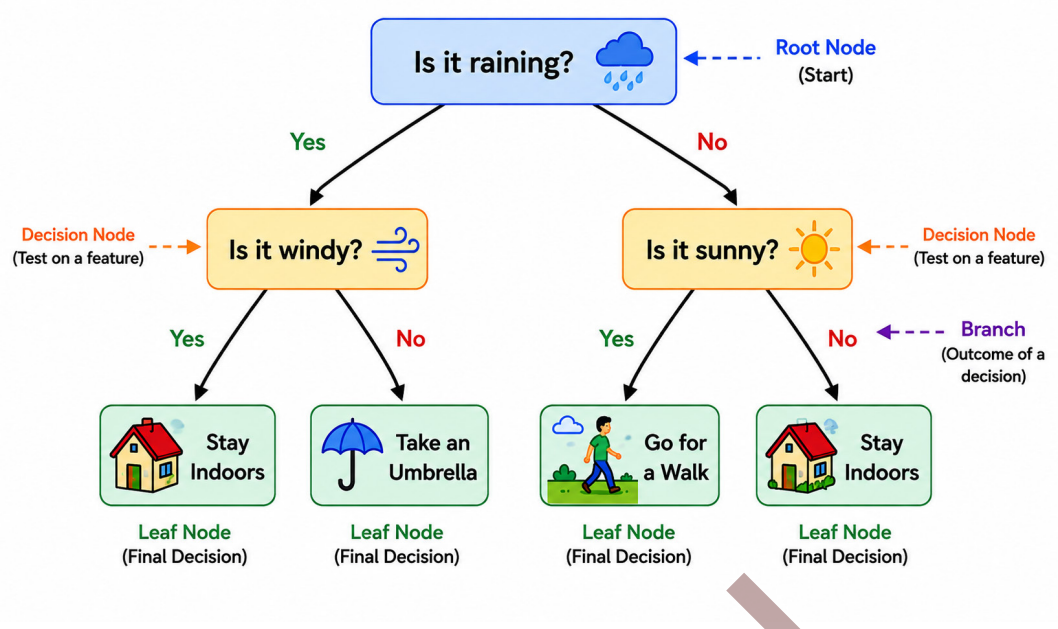


Fig. 2.1.5 Example of Decision Tree

The working of the Decision Tree shown in Figure 1.5 can be explained as follows:

Step 1: Start at the Root Node

The decision-making process begins at the root node, which asks the first question: "Is it raining?" This is the first condition used to evaluate the weather.

Step 2: Follow the Appropriate Branch

If the answer is Yes, the algorithm follows the Yes branch to the next decision node. If the answer is No, it follows the No branch to another decision node.

Step 3: Evaluate the Next Decision

If the answer to "Is it raining?" is Yes, the algorithm asks the next question: "Is it windy?"

If the answer is Yes, the algorithm reaches the leaf node "Stay Indoors."

If the answer is No, the algorithm reaches the leaf node "Take an Umbrella."

Step 4: Evaluate the Alternative Path

If the answer to "Is it raining?" is No, the algorithm asks another question: "Is it sunny?"

If the answer is Yes, the algorithm reaches the leaf node "Go for a Walk."

If the answer is No, the algorithm reaches the leaf node "Stay Indoors."

Step 5: Reach the Final Decision

The algorithm continues following the appropriate branches until it reaches a leaf node. The leaf node represents the final decision or predicted class. In this example, the possible outcomes are Stay Indoors, Take an Umbrella, and Go for a Walk.

SGOU

2.1.3.3 Naive Bayes Algorithm

Naive Bayes is a supervised machine learning algorithm used for classification. It is based on Bayes' theorem, which uses the probability of different classes to predict the most likely class for a given data instance.

The algorithm examines the available features of a data instance and calculates the probability of the instance belonging to each possible class. The class with the highest probability is selected as the predicted class.

The algorithm is called "Naive" because it assumes that the features are independent of each other when making a prediction. Although this assumption may not always be completely true in real-world data, the algorithm can still provide good classification results in many applications.

Naive Bayes is particularly useful when dealing with large datasets and multiple features. It is also computationally efficient and relatively simple to implement. It is widely used in applications such as spam email detection, text classification, sentiment analysis, document classification, and medical diagnosis.

For example, in spam email detection, the algorithm can examine features such as the presence of certain words or phrases in an email and calculate whether the email is more likely to belong to the Spam or Not Spam class. The class with the higher probability is assigned to the email.

Working of the Naive Bayes Algorithm

The Naive Bayes algorithm classifies a new data instance by calculating the probability of that instance belonging to each possible class. It uses the available features and Bayes' theorem to determine the most likely class. The working of Naive Bayes can be understood using a simple spam email classification example.

Example: Spam Email Classification

Consider a system that classifies an email as either **Spam** or **Not Spam** based on the words(free, win, offer) present in the email.

Step 1: Prepare the Training Dataset

The algorithm starts with a labelled dataset containing previously classified emails. Each email is labelled as either Spam or Not Spam.

Step 2: Identify the Classes

The possible classes in this example are:

- **Spam**
- **Not Spam**

These are the classes that the algorithm will use to classify a new email.

SGOU

Step 3: Identify the Features

The algorithm examines features of the emails, such as the presence of particular words. For example, words such as **"offer"**, **"free"**, and **"prize"** may be considered features.

Step 4: Calculate the Prior Probability

The algorithm calculates the probability of each class based on the training data.

For example:

$$P(Spam) = \frac{Number\ of\ Spam\ emails}{Total\ number\ of\ Emails}$$

$$P(Not\ Spam) = \frac{Number\ of\ Not\ Spam\ emails}{Total\ number\ of\ Emails}$$

These probabilities indicate how likely each class is before considering the features of the new email.

Step 5: Calculate the Feature Probabilities

The algorithm calculates how likely each feature is to occur within each class.

The probability of a feature occuring in a class can be calculated using the following formula:

$$P\ (Feature|\ Class) = \frac{Number\ of\ emails\ in\ the\ class\ containing\ the\ feature}{Total\ number\ of\ emails\ in\ the\ class}$$

For example:

$$P\ (Free\ |\ Spam) = \frac{Spam\ emails\ containing\ "Free"}{Total\ number\ of\ spam\ emails}$$

represents the probability of the word "Free" appearing in a Spam email.

Similarly,

$$P\ (Free\ |\ Not\ Spam) = \frac{Not\ Spam\ emails\ containing\ "Free"}{Total\ number\ of\ Not\ Spam\ emails}$$

represents the probability of the word "Free" appearing in a Not Spam email.

The same calculation can be performed for other features such as "Offer" and "Prize".

Step 6: Calculate the Probability for Each Class

For a new email, the algorithm combines the prior probability with the feature probabilities to determine the probability that the email belongs to each class.

SGOU

The Naive Bayes algorithm Calculates the Probability of an email belonging to each class using the following formula:

$$P(C|_X) \propto P(C) \prod_{i=1}^{n} P(x_i \mid C)$$

Where:

- $P(C \mid X)$: Probability that the email belongs to class C, given its features.
- $P(C)$: Prior probability of the class (Spam or Not Spam).
- $P(x_i \mid C)$: Probability of a feature occurring in that class
- n: Total number of features in the email.
- $\prod$: Multiply the probabilities of all features.

Example: For an email containing the words "Free", "Offer", and "Prize":

$$Spam\ Score\ \alpha\ P(Spam)\ X\ P(Free \mid Spam)\ X\ P(Offer \mid Spam)\ X\ P(Prize \mid Spam)$$

The algorithm calculates a similar score for Not Spam.

Step 7: Compare the Probabilities

The calculated probabilities for the two classes are compared. The class with the higher probability is considered more likely to be the correct class.

Step 8: Assign the Class

Finally, the new email is assigned to the class with the highest probability. For example, if the calculated probability of Spam is higher than that of Not Spam, the email is classified as Spam.

2.1.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm used mainly for classification. It classifies data by finding a suitable boundary that separates data points belonging to different classes.

SGOU

SVM aims to find the best possible separation between classes so that new data instances can be classified accurately. It is particularly effective when the classes are clearly separable and can also handle complex classification problems using suitable techniques.

SVM can be used for binary classification, where the data is divided into two classes, and it can also be extended to handle multiple classes.

SVM is widely used in applications such as text classification, image classification, handwriting recognition, and pattern recognition. Its ability to work with high-dimensional data makes it useful in many machine learning applications.

The main concepts associated with SVM include hyperplanes, margins, and support vectors. These concepts help the algorithm identify the most suitable boundary for separating different classes.

Note: The detailed concepts of hyperplanes, margins, support vectors, and the Kernel Trick, along with the working of SVM, are discussed in Unit 3.

2.1.5 Ensemble Methods

Ensemble Methods are supervised machine learning techniques that combine the predictions of multiple models to produce a more reliable and accurate prediction. Instead of depending on a single model, ensemble methods use several models and combine their results to make the final decision.

The individual models used in an ensemble are often called base learners or weak learners. By combining their predictions, ensemble methods can improve the overall performance of the classification model and reduce the effect of errors made by individual models.

Ensemble methods are widely used in classification problems because they can improve accuracy, stability, and generalization. They are useful in applications such as fraud detection, medical diagnosis, customer classification, and risk prediction.

Common ensemble techniques include Bagging, Boosting, Random Forests, and Gradient Boosting. These methods differ in the way multiple models are trained and how their predictions are combined.

Note: The detailed concepts and working of Bagging, Boosting, Random Forests, and Gradient Boosting are discussed in Unit 3.

SGOU

Summerised Overview

Classification is an important supervised machine learning technique used to assign data instances to predefined classes based on patterns learned from labelled data. A classification model is trained using features, training data, and class labels and then uses the learned patterns to classify new or unseen data. The basic idea is to examine the characteristics of a new data instance and determine the class to which it most likely belongs. k-Nearest Neighbour (k-NN) performs classification by finding the data points that are closest to a new instance. It uses a distance measure to identify the nearest neighbours and applies majority voting to determine the final class. A Decision Tree follows a sequence of conditions to classify data. The process starts at the root node, moves through decision nodes and branches according to the feature values, and finally reaches a leaf node, which represents the predicted class. Naive Bayes uses probability and Bayes' theorem to determine the most likely class for a new data instance. It calculates the probability of different classes based on the available features and selects the class with the highest probability. Support Vector Machine (SVM) is another classification approach that separates data belonging to different classes using a suitable decision boundary. It is useful for classification problems involving multiple features and different types of data. Ensemble Methods use multiple models together rather than depending on a single model. Their predictions are combined to obtain a more reliable and accurate final prediction. Important ensemble approaches include Bagging, Boosting, Random Forests, and Gradient Boosting. Thus, the main classification approaches to remember are k-NN, Decision Tree, Naive Bayes, SVM, and Ensemble Methods. Each approach uses a different strategy to identify patterns in data and assign new instances to appropriate classes.

SGOU

Assignments

1. Explain the concept of classification in supervised machine learning. Describe the working of a classification model with suitable examples.

2. Explain the working of the k-Nearest Neighbour (k-NN) algorithm step by step. Discuss how distance calculation and majority voting are used to classify a new data instance with a suitable example.

3. Explain the working of a Decision Tree algorithm with a suitable example. Describe the role of the root node, decision nodes, branches, and leaf nodes in the classification process.

4. Explain the working of the Naive Bayes algorithm. Describe how prior probability and feature probabilities are used to classify a new data instance with a suitable example.

5. Explain the basic concepts of Support Vector Machine (SVM) and Ensemble Methods as classification approaches. Describe their purpose and list the major techniques associated with each approach.

Suggested Reading

1. Raschka, Sebastian, Yuxi Liu, and Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*. Packt Publishing, 2022.

2. Müller, Andreas C., and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 2nd ed., O'Reilly Media, 2025.

3. Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Reference

1. Chatterjee, Indranath. *Machine Learning and Its Application: A Quick Guide for Beginners*. Bentham Science Publishers, 2021.

2. Wang, Ruye, and Harvey Mudd College. *Introduction to Machine Learning: From Math to Code*. Cambridge University Press, 2025.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

2 UNIT

Regression Algorithms

Learning Outcomes

After completion of this unit, the learner will be able to:

- describe the working of Linear, Logistic, and Polynomial Regression
- explain the role of the Sigmoid Function in Logistic Regression
- describe the use of polynomial equations for modelling non-linear relationships
- explain MAE, RMSE, and R^2 as regression evaluation metrics

Background

Have you ever wondered how a machine learning model can predict the price of a house, estimate product sales, or forecast a student's marks? Such predictions involve numerical values. Machine learning uses regression techniques to learn patterns from existing data and make predictions for new situations.

The relationship between input and output is not always the same. Sometimes the data follows a simple straight-line pattern, while in other cases the relationship may be curved or non-linear. In some prediction problems, the model may also need to determine the probability of an outcome belonging to a particular class. These different situations require different approaches.

In this unit, you will explore Linear Regression, Logistic Regression, and Polynomial Regression and understand how each technique learns from data and produces predictions. You will also explore the Sigmoid Function, which enables Logistic Regression to convert calculated values into probabilities.

But making a prediction is only part of the task. An important question remains: How accurate is the prediction? To answer this, the unit introduces Mean Absolute

SGOU

Error (MAE), Root Mean Squared Error (RMSE), and R-Squared (R^2), which help in understanding and evaluating the performance of regression models.

Keywords

Regression, Linear Regression, Logistic Regression, Polynomial Regression, Evaluation Metrics

Discussion

2.2.1 Introduction to Regression

In many real-world situations, the goal of machine learning is not to predict a category, but to predict a numerical value. For example, we may want to predict the price of a house, the sales of a product, the temperature of a day, or the marks a student may obtain. Such prediction problems are handled using regression.

Regression is defined as a supervised machine learning technique used to predict a continuous numerical value based on one or more input features. The model learns the relationship between the input features and the known numerical outcomes from labelled training data. After learning this relationship, the model can estimate the value for a new or unseen data instance.

For example, suppose a model is trained using information about students' hours of study and their corresponding exam scores. After learning from the available data, the model can predict the expected score of a student based on the number of hours studied.

Similarly, a regression model can be used to:

- Predict the price of a house based on its size and location.
- Predict monthly sales based on previous sales information.
- Predict temperature based on weather-related features.
- Predict salary based on experience and other relevant factors.

2.2.1.1 Types of Regression Algorithms

Regression problems can be solved using different algorithms, depending on the type of relationship between the input features and the output value. The major regression algorithms covered here are Linear Regression, Logistic Regression, and Polynomial Regression. Each algorithm uses a different approach to learn patterns from data and make predictions, as shown in Figure 2.2.1.

SGOU

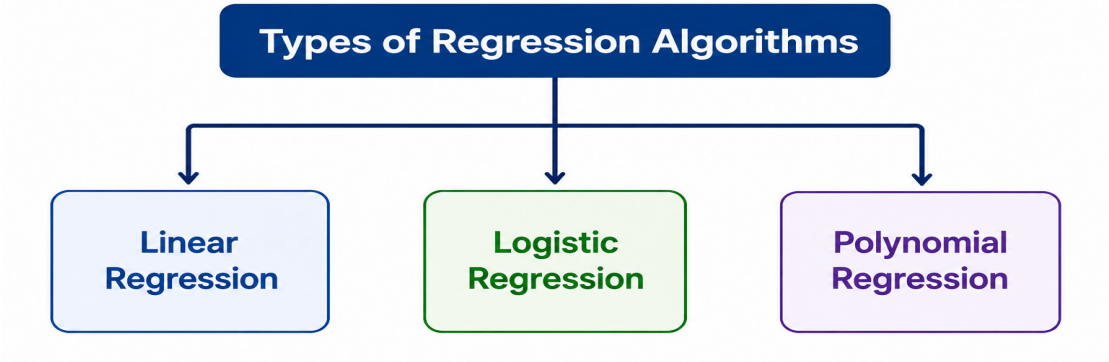


Fig. 2.2.1 Types of Regression Algorithms

2.2.2 Linear Regression

Linear Regression is a fundamental supervised learning algorithm used to predict a continuous numerical value by modelling the relationship between an input variable and an output variable. It is particularly suitable when the relationship between the variables can be reasonably represented by a straight line.

For example, there may be a relationship between the area of a house and its selling price. In general, houses with larger areas may have higher prices. Similarly, the sales of a product may be related to advertising expenditure. Linear Regression uses the available training data to identify and represent such relationships.

The algorithm analyses the training data and determines a best-fit straight line that represents the relationship between the input and output variables. This line can then be used to estimate the output value corresponding to a new input.

For example:

House Area → Linear Regression Model → Predicted House Price

Here, house area is the input variable, while house price is the output variable. After learning the relationship from existing house data, the model can predict the expected price of a house based on its area.

Linear Regression involves two important variables:

- Independent Variable: The input variable used to make the prediction.
- Dependent Variable: The continuous numerical value that is predicted.

Working of Linear Regression

Linear Regression works by analysing the relationship between the input and output variables and identifying a best-fit straight line through the available data points. The learned line is then used to predict the output value for a new input. The working of Linear Regression can be understood from Figure 2.2.2.

SGOU

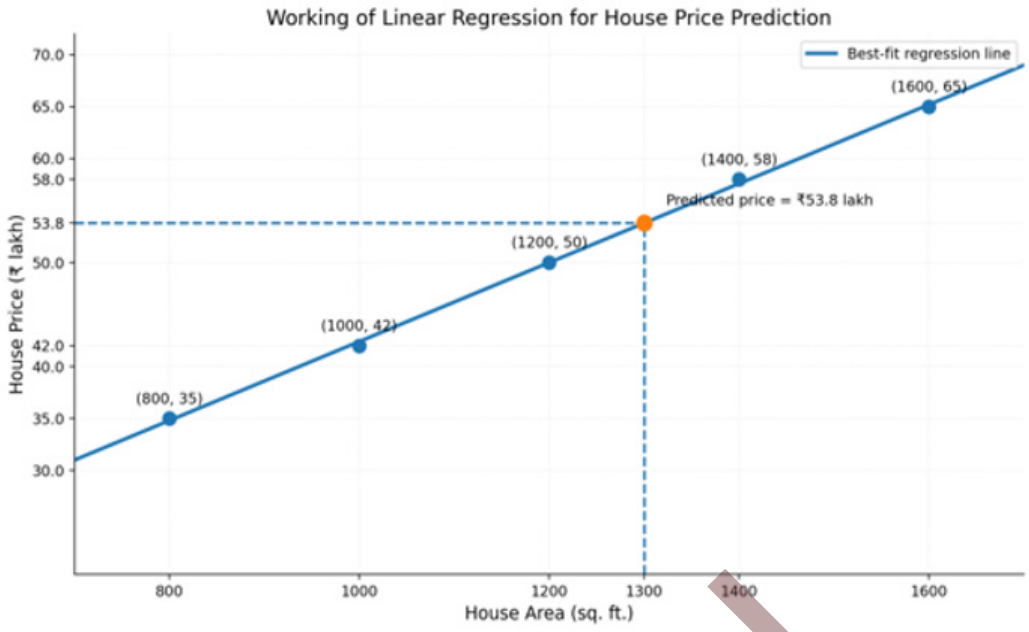


Fig. 2.2.2 Linear Regression for House Price Prediction

Step 1: Plot the Training Data

The available training data is represented as data points on the graph. In the given example, house area is represented on the X-axis, while house price is represented on the Y-axis. Each point represents the observed area and corresponding price of a house.

Step 2: Analyse the Relationship

The algorithm analyses the plotted data points to identify the relationship between house area and house price. In the figure, the data points show a general upward trend, indicating that house price tends to increase as house area increases.

Step 3: Determine the Best-Fit Line

Linear Regression determines a best-fit straight line that represents the overall relationship between the data points. The line provides an estimate of the house price for different house areas.

Step 4: Provide a New Input

After learning the relationship from the training data, a new input can be given to the model. For example, consider a new house with an area of 1300 sq. ft.

Step 5: Predict the Output

The new input value, 1300 sq. ft., is located on the X-axis. By moving vertically from this point until it meets the regression line, the corresponding value on the Y-axis gives the predicted house price.

Step 6: Obtain the Final Prediction

The value estimated from the regression line is taken as the predicted output for the new house. Thus, the model uses the relationship learned from the existing data to estimate the price of a house for a given area.

SGOU

2.2.3 Logistic Regression

Logistic Regression is a supervised machine learning algorithm mainly used for classification problems. Unlike Linear Regression, which predicts a continuous numerical value, Logistic Regression predicts the probability of an instance belonging to a particular class.

It is commonly used when the output has two possible categories, such as Yes/No, True/False, Pass/Fail, or Disease/No Disease.

For example, a Logistic Regression model can be used to predict whether a student is likely to Pass or Fail based on factors such as study hours, attendance, and internal marks. The model analyses the available labelled training data and learns the relationship between the input features and the class outcome.

The predicted probability is converted into a class using a suitable decision threshold, commonly 0.5. If the predicted probability is greater than or equal to 0.5, the instance may be classified as one class; otherwise, it is classified as the other class.

Working of Logistic Regression

Logistic Regression works by calculating the relationship between the input features and the target class and then converting the calculated value into a probability between 0 and 1. This conversion is performed using a mathematical function called the Sigmoid Function.

The Sigmoid Function produces an S-shaped curve, known as the Sigmoid Curve. The curve gradually changes from values close to 0 to values close to 1. This property makes it suitable for representing probabilities in classification problems.

The Sigmoid Function is given by:

$$P = \frac{1}{1+e^{-z}}$$

where:

- P : predicted probability
- z : value calculated from the input features
- e : mathematical constant approximately equal to 2.718

SGOU

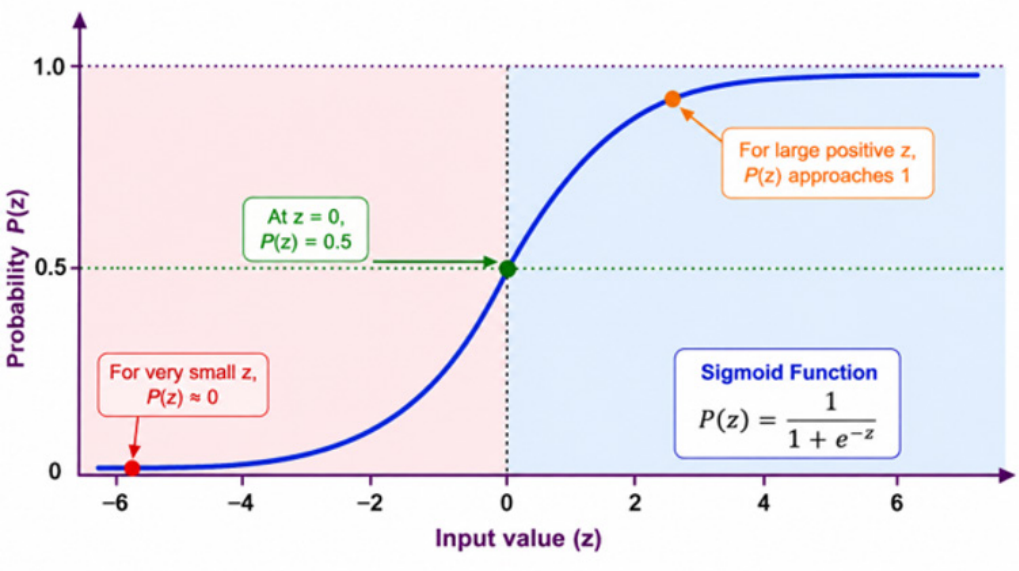


Fig. 2.2.3 Sigmoid Function Curve

The horizontal axis represents the calculated inpu value z, while the vertical axis represents the predicted robability. As the value of z increases, the probability gradually approaches 1. When z decreases, the probability approaches 0.

The working of Logistic Regression can be understood through the following steps:

Step 1: Provide the Input Data

Consider a bank that wants to predict whether a loan application should be approved or rejected. The model receives information about the applicant as input.

For example, the input feature can be monthly income. Other factors such as age, existing loan amount, credit history, and employment status may also be considered.

Step 2: Calculate the Linear Combination

The Logistic Regression model combines the input values using coefficients learned from previous loan applications. For a single input feature, the calculation is represented as:

$$z = b_0 + b_1 x$$

$z =$ intermediate value calculated by the model

$b_0 =$ intercept

$b_1 =$ coefficient associated with the input

$x =$ input value

SGOU

Suppose the model has learned:

$$b_0 = -2$$

and

$$b_1 = 0.00004$$

For an applicant with a monthly income of ₹50,000:

$$z = -2 + (0.00004\ X\ 50000)$$

$$z = -2 + 2 = 0$$

Note: The values of b_0 and $b1$ are learned automatically from the training data during the model training process.

Step 3: Apply the Sigmoid Function

This calculated value Z obtained in the previous step is passed through the Sigmoid Function to convert it into a probability between 0 and 1.

$$P = \frac{1}{1 + e^{-z}}$$

Since $Z = 0$:

$$P = \frac{1}{1 + e^{0}} = 0.5$$

Thus, the model estimates a 50% probability that the applicant belongs to the Approved class.

Step 4: Compare with the Decision Threshold

A decision threshold of 0.5 is commonly used to convert the predicted probability into a final class.

$$if\ P \geq 0.5 \rightarrow Approved$$

$$if\ P < 0.5 \rightarrow Rejected$$

SGOU

In this example:

$$P = 0.5$$

Since the probability is equal to the threshold, the application is classified as Approved according to the given rule.

However, the applicant is exactly at the classification boundary, meaning the model does not strongly favour either Approved or Rejected.

Step 5: Obtain the Final Prediction

The model gives the final result based on the calculated probability.

In this example, the probability is 0.50. Since it is equal to the decision threshold, the application is classified as Approved.

Thus, the model predicts that the loan application is Approved.

2.2.4 Polynomial Regression

Polynomial Regression is a supervised machine learning algorithm used to predict a continuous numerical value when the relationship between the input and output variables is curved or non-linear.

Unlike Linear Regression, which represents the relationship using a straight line, Polynomial Regression uses a polynomial equation to represent the pattern in the data.

It adds higher-order terms such as x^2, x^3, and so on to capture the curved relationship.

For example, the relationship between temperature and ice cream sales may not increase at a constant rate. As temperature increases, ice cream sales may increase more rapidly. Polynomial Regression can represent such a curved pattern more effectively.

The general form of a polynomial regression equation is:

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$$

where y is the predicted value, x is the input variable, and $b_0, b_1, \ldots, b_n$ are the coefficients learned from the training data.

SGOU

Working of Polynomial Regression

The working of Polynomial Regression can be understood using the relationship between temperature and ice cream sales. Consider the following illustrative data:

Table 2.2.1 Temperature & Ice cream sales data

Temperature (°C)	Ice Cream Sales (units)
10	20
15	35
20	60
25	105
30	180
35	275
40	380

Here, temperature is the input variable, while ice cream sales is the continuous output variable.

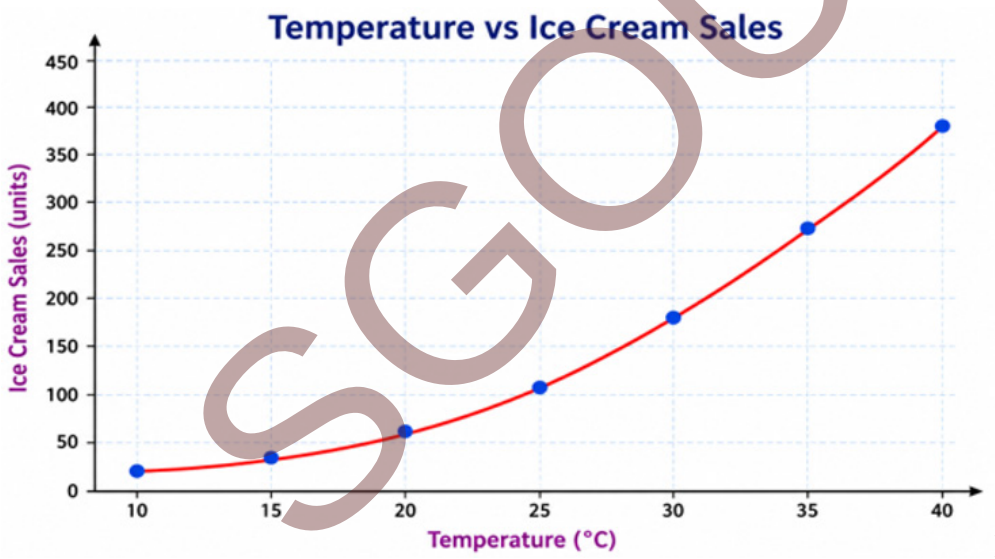


Fig. 2.2.4 Temperature vs Ice cream Sales data

Step 1: Provide the Training Data

The model is first provided with historical data containing temperature values and the corresponding ice cream sales.

For example, the data shows that sales are generally higher when the temperature is higher.

Step 2: Observe the Relationship

The model examines the relationship between temperature and sales.

In this example, sales do not increase at a constant rate. For instance, the increase in sales from 10°C to 15°C is smaller than the increase from 30°C to 35°C.

SGOU

Therefore, the relationship appears to be curved rather than a straight line.

Step 3: Add Polynomial Terms

To represent this curved relationship, Polynomial Regression introduces higher-order terms such as x^2, x^3, and so on.

For a second-degree polynomial, the equation is:

$$y = b_0 + b_1 x + b_2 x^2$$

Here:

- x = temperature
- y = predicted ice cream sales
- b_0 = intercept
- b_1 and b_2 = coefficients learned from the training data

Step 4: Learn the Coefficients

The model uses the training data to determine suitable values for b_0, b_1, and b_2.

Note: The coefficients are learned automatically during the training process so that the polynomial curve represents the observed data as closely as possible.

For the illustrative data above, a fitted quadratic equation is approximately:

$$y = 0.4095x^2 - 8.4762x + 65.7143$$

Step 5: Form the Polynomial Curve

The learned equation produces a curved line rather than a straight line. This curve represents the relationship between temperature and ice cream sales.

The curve can follow the pattern in the training data more effectively because it can represent the increasing rate of sales as temperature rises.

Step 6: Make a Prediction

After learning the relationship between temperature and ice cream sales, the model can predict ice cream sales for a new temperature value.

SGOU

For example, when the temperature is 32°C, the trained model uses the learned polynomial equation to predict the expected ice cream sales.

Step 7: Interpret the Result

The predicted value represents the estimated number of ice cream units that may be sold at the given temperature.

Therefore, Polynomial Regression is useful when the available data shows a curved relationship between the input and the continuous output.

2.2.5 Evaluation Metrics for Regression

Regression evaluation metrics are used to measure how accurately a regression model predicts continuous numerical values. The predicted values are compared with the actual values, and the difference between them is used to evaluate the model's performance.

The important evaluation metrics are:

1. Mean Absolute Error (MAE)
2. Root Mean Squared Error (RMSE)
3. R Squared (R^2) / Coefficient of Determination

2.2.5.1 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a measure used to determine how close the predicted values are to the actual values. It calculates the average difference between the predicted and actual values, where a lower MAE indicates that the model's predictions are closer to the actual values.

The formula for MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- n = number of observations
- y_i = actual value
- $\hat{y}_i$ = predicted value
- $|y_i - \hat{y}_i|$ = absolute difference between the actual and predicted values

SGOU

Example

Consider a model that predicts ice cream sales based on temperature. The actual and predicted sales for three days are given below.

Table 2.2.2 Ice cream sales data with Absolute Error

Day	Actual Sales	Predicted Sales	Absolute Error
1	100	90	10
2	150	160	10
3	200	180	20

Step 1: Calculate the Difference

Find the difference between the actual sales and predicted sales for each day.

Day 1:

$100-90=10$

Day 2:

$150-160=-10$

Day 3:

$200-180=20$

Step 2: Take the Absolute Values

MAE uses the absolute value, so negative signs are removed.

$|10| = 10$

$|-10| = 10$

$|20| = 20$

Therefore, the absolute errors are: 10, 10, and 20

Step 3: Calculate the Average Error

Add the absolute errors and divide by the total number of observations.

$$MAE = \frac{10+10+20}{3} = \frac{40}{3}$$

$$MAE = 13.33$$

SGOU

Step 4: Interpret the Result

The calculated MAE is 13.33 units.

This means that the model's predicted ice cream sales differ from the actual sales by approximately 13.33 units on average.

2.2.5.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a regression evaluation metric used to measure the difference between the actual values and the predicted values. It gives more importance to larger errors because the errors are squared during the calculation.

The formula for RMSE is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

- n = number of observations
- y_i = actual value
- $\hat{y}_i$ = predicted value
- $(y_i - \hat{y}_i)^2$ = squared error

Example

Consider the same ice cream sales example:

Table 2.2.3 Ice cream sales data with Squared Error

Day	Actual Sales	Predicted Sales	Error	Squared Error
1	100	90	10	100
2	150	160	-10	100
3	200	180	20	400

Step 1: Calculate the Error

100−90=10

150−160=−10

200−180=20

SGOU

Step 2: Square the Errors

$10^2 = 100$

$(-10)^2 = 100$

$20^2 = 400$

Step 3: Calculate the Mean Squared Error(MSE)

$$MSE = \frac{100 + 100 + 400}{3}$$

$$MSE = \frac{600}{3} = 200$$

Step 4: Take the Square Root

$$RMSE = \sqrt{200}$$

$$RMSE \approx 14.14$$

Therefore, the RMSE is approximately 14.14 units.

In this example, the RMSE of 14.14 units means that the model's prediction errors are approximately 14.14 un[illegible] with larger errors having a greater effect on the result.

2.2.5.3 R-Squared (R^2)/Coefficient of Determination

R-Squared (R^2) is a regression evaluation metric used to measure how well the model explains the variation in the actual output values.

In simple terms, R^2 tells us how well the regression model fits the given data. Its value generally ranges from 0 to 1, where a value closer to 1 indicates a better fit.

The formula is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

SGOU

where:

- n = number of observations
- y_i = actual value
- $\hat{y}_i$ = predicted value
- $\bar{y}$ = mean of the actual values

Example

Consider a model that predicts **ice cream sales** based on temperature. Suppose the model produces an R^2 value of:

$$R^2=0.90$$

This means that the model explains approximately 90% of the variation in ice cream sales based on the input data used in the model.

The remaining 10% of the variation may be due to other factors that are not considered by the model, such as holidays, weather conditions, customer preferences, or special offers.

In general:

- R^2 close to 1 → The model fits the data well.
- R^2 close to 0 → The model explains very little of the variation in the output.
- A higher R^2 generally indicates a better fit, but it should not be used alone to judge model performance.

SGOU

Summerised Overview

Regression is a supervised machine learning technique used to predict continuous numerical values from one or more input features. It learns the relationship between input variables and known outcomes from labelled training data and uses this relationship to predict values for new data. Common applications include predicting house prices, product sales, temperature, salary, and other numerical outcomes. The unit introduces three important regression algorithms: Linear Regression, Logistic Regression, and Polynomial Regression.

Linear Regression is used when the relationship between the input and output can be represented approximately by a straight line. It identifies a best-fit line from the training data and uses it to predict a continuous value, such as house price based on house area. The input is called the independent variable, while the predicted value is the dependent variable. Logistic Regression is mainly used for classification rather than continuous-value prediction. It calculates a probability between 0 and 1 using the Sigmoid Function and applies a decision threshold, commonly 0.5, to assign an instance to a class such as Approved/Rejected or Pass/Fail.

Polynomial Regression is useful when the relationship between input and output is curved or non-linear. It introduces higher-order terms such as x^2 and x^3 to represent the pattern more effectively. Finally, regression models are evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-Squared (R^2). MAE measures the average prediction error, RMSE gives greater importance to larger errors, and R^2 indicates how well the model explains variation in the output.

Assignments

1. Explain the working of Linear Regression using the house price prediction example. Describe the steps involved in obtaining the final prediction.

2. Explain Logistic Regression as a classification algorithm. Describe the role of the Sigmoid Function and decision threshold in obtaining the final classification.

3. Explain Polynomial Regression with the help of the temperature and ice cream sales example. Describe how polynomial terms help the model represent a curved relationship.

4. Explain Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Using a suitable example, describe how these metrics are calculated and interpreted.

5. Explain R-Squared (R^2) as an evaluation metric for regression models. Describe what its value indicates about the model's fit and explain the interpretation of R^2 values close to 0 and 1.

SGOU

Reference

1. Theodoridis, Sergios. *Machine Learning: From the Classics to Deep Networks, Transformers, and Diffusion Models*. 3rd ed., Academic Press, 2024.

2. Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. 6th ed., Wiley, 2021.

3. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in Python*. Springer, 2023.

4. Murphy, Kevin P. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

Suggested Reading

1. Sridhar, S., and M. Vijayalakshmi. *Machine Learning*. 2nd ed., Oxford University Press, 2026.

2. Witten, Ian H., Eibe Frank, Mark A. Hall, Christopher J. Pal, and James Foulds. *Data Mining: Practical Machine Learning Tools and Techniques*. 5th ed., Elsevier, 2025.

3. Thareja, Reema. *Data Science and Machine Learning Using Python*. 2nd ed., McGraw Hill, 2025.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

3 UNIT

SVM and Ensemble Method

Learning Outcomes

After completion of this unit, the learner will be able to:

- understand the basic idea of SVM, hyperplanes, and ensemble methods.
- explain how the kernel trick helps in solving non linear problems.
- describe how bagging, boosting, Random Forest, and Gradient Boosting work.
- apply these methods to solve simple machine learning problems.

Background

Support Vector Machines and ensemble methods are important techniques in modern machine learning used for solving classification and regression problems. SVM focuses on finding an optimal hyperplane that can effectively separate data points into different classes. However, many real world datasets are complex and not linearly separable, which makes the kernel trick essential for transforming data into higher dimensions. Ensemble methods such as bagging and boosting improve model performance by combining multiple models instead of relying on a single one. Bagging helps reduce variance by training models independently on different subsets of data, while boosting focuses on correcting errors made by previous models. Techniques like Random Forest further enhance bagging by using multiple decision trees with random feature selection. Gradient Boosting builds models sequentially to minimize prediction errors and achieve high accuracy. Studying these methods is important because they provide powerful tools to handle complex and large datasets. They are widely used in applications such as fraud detection, medical diagnosis, and recommendation systems. Understanding these concepts

SGOU

helps in building accurate, reliable, and efficient machine learning models for real world problems.

Keywords

Hyperplane, Margin, Support vectors, Kernel trick, Feature mapping, Bagging, Boosting, Random forest

Discussion

2.3.1 What is a Hyperplane?

Consider the 2D case first. When you draw a straight line to separate data, that line is a hyperplane in two dimensions. If we move to 3D, the hyperplane becomes a flat sheet (a plane) that divides space. If we move to higher dimensions such as four, five, or even hundreds of features, the hyperplane becomes a general mathematical surface that still divides data into regions.

So a hyperplane is nothing but a decision boundary.

A typical hyperplane equation looks like:

$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = 0$

This is the dividing boundary. The sign of the left-hand expression decides whether a point belongs to Class 1 or Class 2.

In simpler words, the hyperplane tells the SVM where one class ends and the other begins.

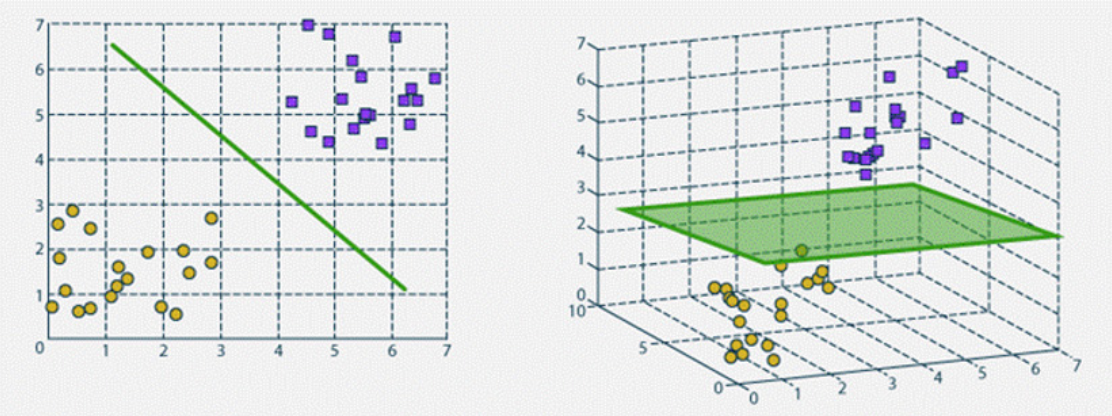


Fig. 2.3.1 Hyperpanes in R^2 and R^3

SGOU

Fig. 2.3.1 illustrates how hyperplanes separate two classes of data in two dimensional and three dimensional feature spaces.

Left diagram: Hyperplane in R^2

The left diagram represents data in a two dimensional feature space, where each data point is described by two features plotted along the X and Y axes. The yellow circles and purple squares represent two different classes. The green straight line is the hyperplane that separates these classes. In two dimensional space, a hyperplane is a straight line. A suitable hyperplane helps the classifier distinguish between the two classes based on their feature values.

Right diagram: Hyperplane in R^3

The right diagram represents data in a three dimensional feature space, where each data point is described by three features along the X, Y and Z axes. The yellow circles and purple squares represent two different classes. The green flat surface is the hyperplane that separates the classes. In three dimensional space, a hyperplane is a two dimensional plane. This plane separates the data points according to their positions in the feature space.

Key difference: In R^2, a hyperplane is a line, whereas in R^3, it is a plane. In higher dimensional feature spaces, a hyperplane has one dimension less than the feature space. In SVM, the classifier seeks a separating hyperplane that maximizes the margin between the classes when a suitable linear separation is possible.

2.3.1.1 What makes one hyperplane better than another?

Imagine drawing many possible lines separating the two classes on paper. All these lines do the job, but some may be very close to the data points, while others may pass right between them with plenty of space.

If a line sits too close to the data, even a small disturbance might cause misclassification. But if a line travels through the center of the empty space, giving equal distance from points of both classes, it is far more reliable.

This equal empty space on both sides is called the margin.

2.3.1.2 Understanding Margin: The "Safety Zone" of SVM

Margin is the distance between the hyperplane and the closest data points. These closest data points are very important. They sit right on the edge of the boundary's safe zone. They are called support vectors.

So support vectors define the margin.

A wider margin means the classifier has a greater safety buffer. SVM tries to maximize this margin. That is why the method is known as a maximum-margin classifier.

SGOU

Think of the margin like this:

If you are driving on a road with a narrow lane, you are more likely to hit the side if you drift a little. But if the lane is wider, you feel safer even if you move slightly to the left or right.

See fig.2.3.2

Similarly, SVM chooses the boundary that gives the largest possible safety lane.

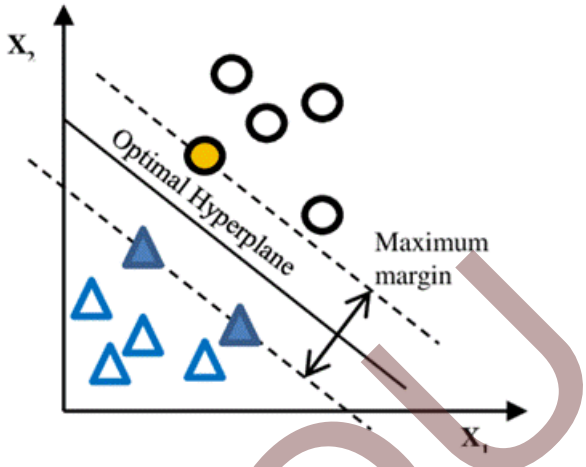


Fig. 2.3.2 Optimal hyperplane and maximum margin

2.3.1.3 Soft Margin vs Hard Margin SVM

In real data, perfect separation is rare. There may be outliers or overlapping cases. So SVM introduces two ideas.

A hard margin insists that every point must be perfectly classified with no mistakes. This is possible only when the data is clean and well-separated. Even a single outlier ruins hard margin SVM.

A soft margin allows some misclassification. It accepts that data is noisy. A penalty variable controls how much error is allowed. The classifier balances between maximizing the margin and minimizing error.

Soft margin SVM is used almost everywhere in practical machine learning.

2.3.1.4 What are Support Vectors?

If you take away all data points except the ones lying closest to the boundary, SVM would still give exactly the same hyperplane. These close points "support" the position of the boundary. So these are the support vectors.

Even if you remove other points farther away, SVM doesn't change its decision. That is why SVM is efficient. It does not depend on the full dataset for decision boundaries, only on critical points near the edge.

SGOU

Support vectors are the heroes of SVM. They decide how the boundary is drawn. Fig. 2.3.3 shows the support vectors in SVM

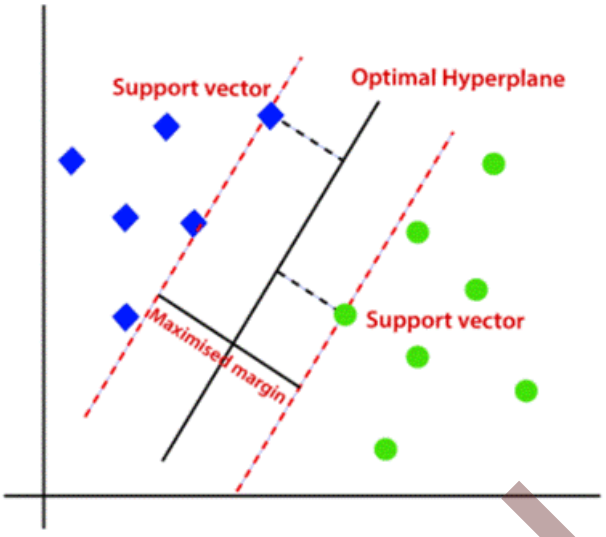


Fig. 2.3.3 Support vectors

2.3.2 Support Vector Machine (SVM)

A Support Vector Machine is a machine learning technique that tries to find the best possible boundary that divides data into classes. Many algorithms can draw boundaries, but SVM tries to draw the most reliable and confident one. Instead of just separating the groups, it looks for the boundary that stays as far away as possible from the nearest points of both groups.

2.3.2.1 Linear SVM – When Data is Linearly Separable

Let us begin with the simple case. If the data is arranged in such a way that a straight line (or hyperplane) can easily separate it, SVM draws the best line that creates the maximum margin.

The steps of thinking are simple:

1. Look at the data
2. Identify the best boundary
3. Maximize the margin
4. Use support vectors to lock the boundary in place
5. Classify new points using the hyperplane equation

In a linear SVM, everything is clean and intuitive. But the real world is rarely so cooperative.

SGOU

Imagine a situation where your data looks like a doughnut. One class is sitting right at the center like a solid circle, and the other class forms a ring around it. No matter how you draw a straight line, you can never cleanly separate the inner points from the outer ones. Even if you try dozens of lines, shifting or rotating them, nothing works.

Does that mean SVM fails completely?

Not at all. This is where SVM uses one of the most elegant ideas in machine learning: the kernel trick.

To understand the algorithm fully, let us walk through the steps one by one.

Step 1: Representing the data in a feature space

Every data point is represented as a point in space. For example, with two features, each point becomes a dot in a 2D plane; with three features, a dot in 3D, and so on.The algorithm views the dataset as a cloud of points.

Step 2: Finding a boundary that separates the classes

The first task is to check whether a line (or hyperplane) can separate the classes. Several boundaries may work, but SVM searches for the best one—not just any separating line, but the one that provides the strongest separation. This leads to the concept of a maximum-margin hyperplane.

Step 3: Identifying the support vectors

As the algorithm evaluates the dataset, it looks for the points that lie closest to the boundary. These are the most important points, called support vectors, because they determine exactly where the boundary should be placed.Even if all other points disappeared, the support vectors alone would keep the same boundary intact.

Step 4: Maximizing the margin

Now comes the heart of the SVM algorithm — margin maximization.

SVM aims to make the margin (the distance between the support vectors and the boundary) as large as possible.
This leads to the objective function used during training:

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

This equation expresses the tradeoff SVM tries to balance:

- The term (1 / margin) encourages a large margin, because a larger margin makes the value smaller.
- The term $\lambda \sum$ penalty controls how much misclassification is allowed (λ is a tuning parameter).

So the algorithm tries to minimize this objective function. Minimizing it automatically leads to maximizing the margin while keeping errors within acceptable limits.

SGOU

This is why SVM models are both confident and stable when predicting new data.

Step 5: Handling noisy or imperfect data (Soft Margin and Objective Function use)

Real-world data is rarely perfect. Some points lie very close to the boundary or even on the wrong side.
To handle this, SVM uses soft margins.

The objective function shown earlier plays an important role here:

Objective Function = (1 / margin) + $\lambda \sum$ penalty

Here:

- The penalty term measures how much each misclassified point "violates" the margin.
- The parameter λ controls how strictly the boundary should fit the data.
  A high $\lambda \rightarrow$ fewer errors allowed
  A low $\lambda \rightarrow$ more flexibility
- Thus the objective function helps SVM balance two goals: maximizing margin and minimizing mistakes.

Step 6: Handling Non-linear Data using the Kernel Trick

If the data cannot be separated by a straight line like a ring or spiral shape, SVM uses kernels.

Look at a curved line on a piece of paper. If you cannot straighten it in 2D, lift it into 3D space. Once you lift it, suddenly the curve might become straight.

Similarly, data that is not linearly separable in its original space might become separable when transformed into a higher-dimensional space.

This transformation is called feature mapping.

Instead of trying to draw a curved boundary in the original space, we lift the data into a higher dimension where a straight boundary is enough.

But doing the complete mathematical transformation for every data point is expensive. It makes the algorithm slow and sometimes impossible to compute.

To grasp the kernel trick, let's use a simple thought experiment. You have two circles—one inside another. In 2D, you cannot draw a straight line to separate them.

That means *linear* SVM cannot help you here.

Instead of struggling in 2D, imagine lifting the data into 3D.

Let's say we add a new dimension:

$Z = x^2 + y^2$

SGOU

Now, think what happens:

- The inner circle (small radius) becomes a set of low z-values
- The outer ring (large radius) becomes high z-values

In the 3D space, they sit at different heights.

Now a simple flat plane,which is a linear separator in 3D, can separate them perfectly. The kernel trick allows us to draw a straight line in a higher-dimensional space without ever going there. See fig.2.3.4

A kernel is just a function.
Instead of computing the actual coordinates in the higher dimension, it directly computes:

$$K(x_i, x_j)=\phi(x_i).\phi(x_j)$$

Here:

ϕ = the imaginary high-dimensional mapping

K = kernel function

We never compute $\phi(x)$—the kernel handles everything.

In this transformed space, the same objective function is optimized, But the margin is now computed using kernels, which means the boundary is no longer a straight line. It can curve smoothly to follow the pattern of the data.

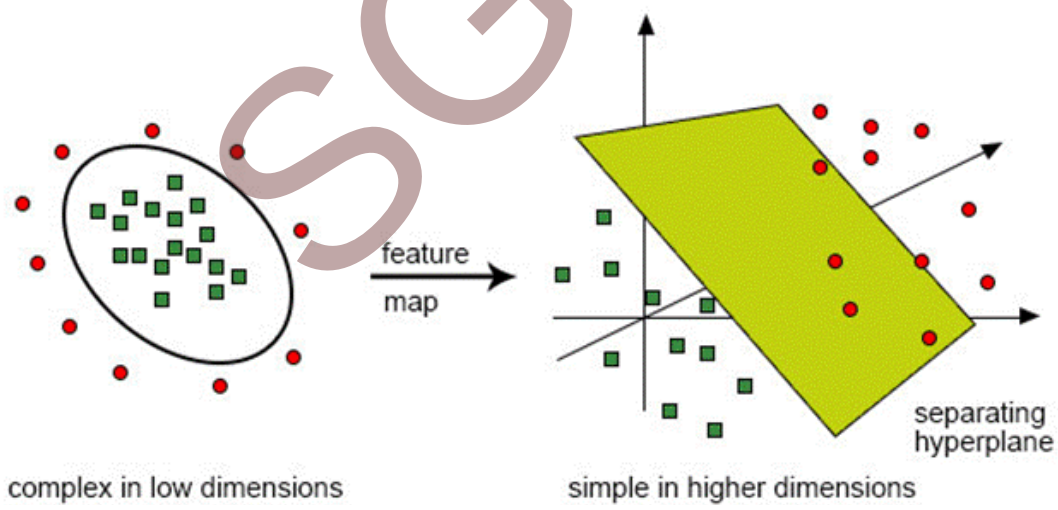


Fig. 2.3.4 Kernel method

Step 7: Final Decision Function

Once the best hyperplane is found (by minimizing the objective function), SVM constructs a decision function.
This function checks whether a new point lies on one side of the boundary or the other.

SGOU

One side → Class A
Other side → Class B

This is how the algorithm performs classification.

SVM is an algorithm that finds the best possible boundary between classes by maximizing the margin and minimizing errors, using the objective function (1 / margin) + λ ∑ penalty, and uses kernels to handle complex non-linear patterns.

2.3.2.2 Types of Kernels used in Non-linear SVM

Several kernels are used, each with its own strengths.

One popular kernel is the Polynomial Kernel.
If you have data that requires curved boundaries of polynomial shape, this kernel does the job. It allows the model to consider interactions between features. For example, rather than just x_1, it can consider x_1^2 or x_1x_2, creating flexibility.

Another widely used kernel is the RBF, or Radial Basis Function kernel. This is extremely powerful because it creates a boundary that depends on distance. If data forms circular, elliptical, or irregular clusters, the RBF kernel handles them nicely. When you hear about SVM for complex image data or sensor patterns, it is often the RBF kernel working behind the scenes.

There are other kernels too, but these two cover most practical applications.

2.3.3 Kernel Trick in Support Vector Classification

Support Vector Machines (SVMs) are widely recognized as a robust and flexible approach for solving classification problems. One of the most important features that extends their effectiveness, especially when handling non-linear datasets—is the Kernel Trick. This concept allows SVMs to operate in higher-dimensional spaces without explicitly computing the transformation, making complex data easier to separate. This article explores the fundamental idea behind the Kernel Trick, including its purpose, how it works, and its real-world applications.

2.3.3.1 Linear vs. Non-Linear Problems

When data is linearly separable, identifying a hyperplane that clearly divides the classes is relatively simple. However, in many practical scenarios, data exhibits non-linear patterns, making it impossible to separate classes using a straight line or flat surface. In such situations, advanced techniques like the kernel trick are used to handle the complexity.

2.3.3.2 Concept of Feature Mapping

To address non-linear data, a common strategy is to transform the original input data into a higher-dimensional space. This process, known as feature mapping, projects the data into a new feature space where the classes can be separated more easily using a linear boundary.

SGOU

For example, imagine a dataset in two dimensions where the classes cannot be separated using a straight line. If these same data points are transformed into a three-dimensional space, it may become possible to separate them using a flat surface (plane), illustrating how higher-dimensional representations can simplify classification.

2.3.3.3 What is the Kernel Trick?

The kernel trick is a powerful technique used in Support Vector Machines (SVMs) to handle non-linear classification problems while still using a linear decision boundary. It works by applying a kernel function that implicitly projects the original data into a higher-dimensional space where the classes can be linearly separated. This approach is efficient because it avoids explicitly computing the transformed coordinates in that high-dimensional space.

2.3.3.4 Types of Kernel Functions

Different kernel functions are available, each suited to particular data characteristics:

- **Linear Kernel:** Assumes the data is already linearly separable, so no transformation is required.
- **Polynomial Kernel:** Transforms the data into a polynomial feature space, allowing the model to capture interactions among features.
- **Radial Basis Function (RBF) Kernel:** Also called the Gaussian kernel, it is effective for modeling complex and non-linear decision boundaries by considering distances between data points.
- **Sigmoid Kernel:** Resembles the behavior of neural networks by applying a sigmoid function, enabling certain non-linear patterns to be captured.

2.3.3.5 How Does the Kernel Trick Work?

The kernel trick is based on computing inner (dot) products between vectors. In SVMs, the decision function depends on these dot products in the input space. Instead of calculating them directly, a kernel function replaces the dot product with a non-linear computation that effectively represents the dot product in a higher-dimensional space. This allows the model to work in an expanded feature space without explicitly performing the transformation, thereby saving both time and computational resources.

Mathematically, it is represented as:

$K(x,y)=\phi(x).\phi(y)$

where:

- x and y are input vectors in the original space
- ϕ is the function that maps data into a higher-dimensional space

The kernel trick greatly improves the flexibility and effectiveness of Support Vector Machines. By enabling them to manage non-linear data efficiently, it expands their applicability to a wide range of complex classification problems. As computing

SGOU

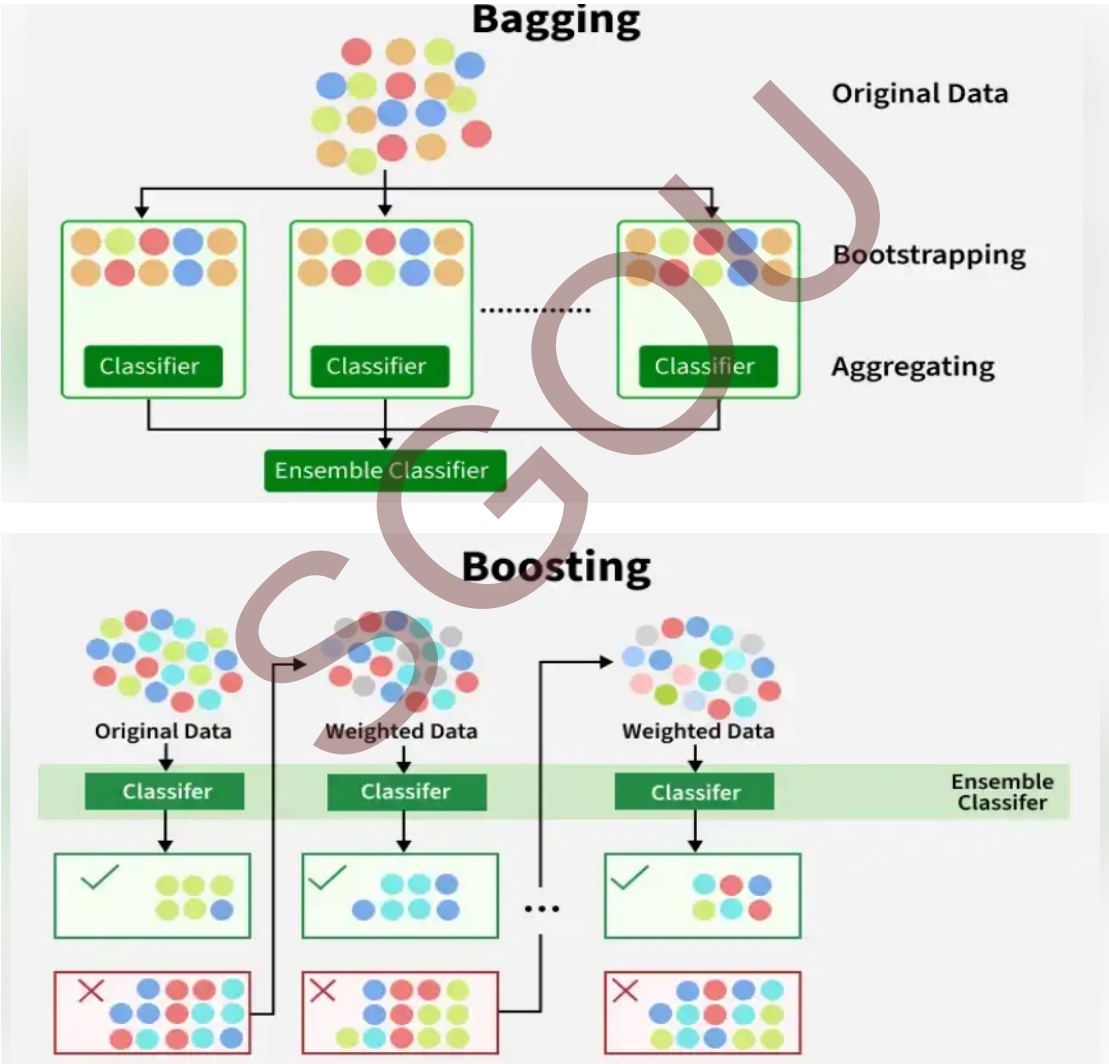
capabilities continue to improve, kernel-based SVMs are expected to play an even more important role in solving real-world challenges across diverse fields.

2.3.4 Bagging vs. Boosting

Bagging and Boosting are ensemble learning methods designed to enhance the accuracy and stability of predictive models by combining the outputs of multiple learners. The key distinction between them lies in how the models are trained:

Bagging (Bootstrap Aggregating): Aims to reduce variance by training several models independently on different subsets of the data and then aggregating their results.

Boosting: Focuses on reducing bias by training models in a sequential manner, where each new model gives more attention to the errors made by the previous ones.



2.3.5 Understanding Bagging and Boosting

Bagging, short for Bootstrap Aggregating, is an ensemble technique that helps reduce the variance of a model by training multiple versions of it on different randomly generated subsets of the original dataset. These subsets are formed through bootstrapping, a process in which data points are sampled with replacement, allowing some observations

SGOU

to appear more than once while others may be omitted. See fig. 2.3.5

Each model is built independently of the others, and their outputs are later combined—using majority voting in classification tasks or averaging in regression problems. Because of this independence, Bagging is particularly effective when applied to high-variance models such as decision trees.

In essence, Bagging relies on random sampling with replacement, trains models in parallel, and aggregates their predictions to produce a more stable and accurate result. It is widely used to reduce overfitting and is commonly applied in methods like Random Forest, as well as in real-world applications such as credit risk assessment and medical diagnosis systems.

2.3.4.2 Understanding Boosting

Boosting is an ensemble learning approach that enhances model performance by building models in a sequential manner. In this process, each new model concentrates more on the instances that were incorrectly predicted by earlier models, allowing the overall system to gradually improve its handling of challenging cases.

This technique is particularly useful for reducing bias and can achieve strong performance even when individual models are relatively simple or weak learners. In Boosting, models are trained one after another, misclassified samples are assigned higher importance, and the final prediction is obtained through a weighted combination of all models.

Overall, Boosting trains models sequentially, emphasizes difficult data points, aggregates results using weighted voting, and helps in minimizing bias. It is widely applied in areas such as fraud detection.See fig. 2.3.5

2.3.4.3 Difference Between Bagging and Boosting

The distinction between Bagging and Boosting can be clearly understood by comparing their key characteristics:

Table 2.3.1 Features of Bagging and Boosting

Feature	Bagging	Boosting
Training approach	Builds multiple models independently	Builds models sequentially, one after another
Primary objective	Aims to reduce variance	Aims to reduce bias
Error handling	Treats all data points equally	Gives more importance to misclassified samples
Overfitting tendency	Less prone to overfitting	More likely to overfit if not controlled
Parallelization	Supports parallel training of models	Does not support parallel training due to sequential dependency
Robustness to noise	Relatively robust to noise and outliers	More sensitive to noise and outliers

SGOU

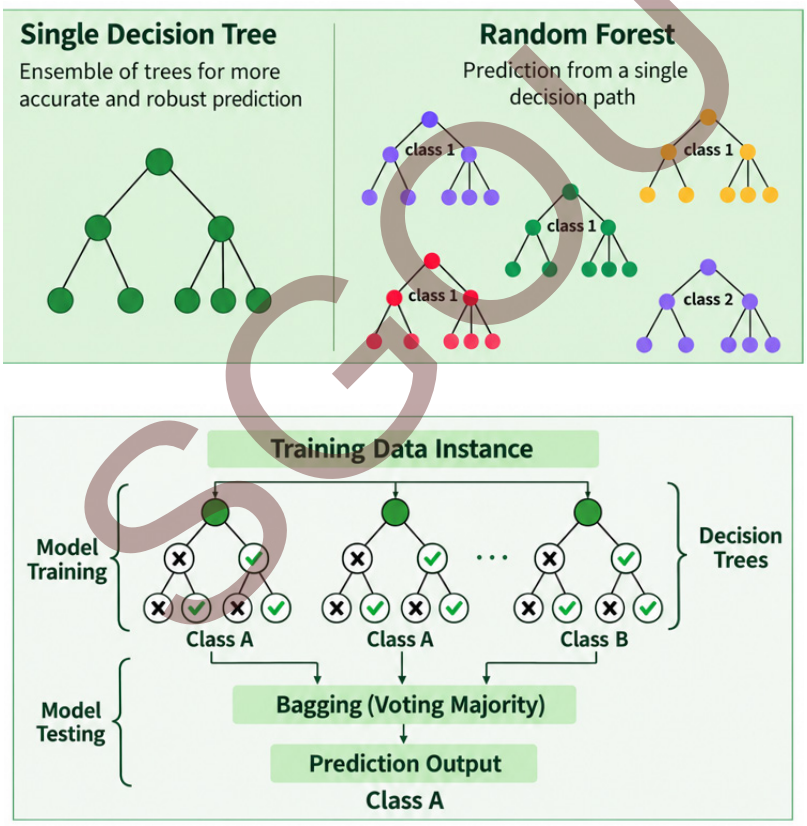
2.3.4.4 When to Choose Each Technique

Bagging: Prefer this method when your model suffers from overfitting and exhibits high variance, which is often the case with models like decision trees. It helps stabilize predictions by reducing variability.

Boosting: Opt for boosting when your model is underfitting and lacks the ability to capture complex patterns. It incrementally improves performance by focusing on difficult cases, leading to higher overall accuracy.

2.3.5 Random Forest

Random Forest is an ensemble learning algorithm that enhances predictive performance by combining multiple decision trees. Each tree is trained on a randomly selected subset of the data and features, ensuring diversity among the models. The final prediction is obtained by aggregating the outputs of all trees, using majority voting for classification tasks and averaging for regression tasks. This collaborative approach improves overall accuracy while minimizing errors and overfitting.See fig. 2.3.6



2.3.6 Random Forest

2.3.5.1 Working of the Random Forest Algorithm

- **Building Multiple Decision Trees:** The algorithm constructs a large number of decision trees, each trained on a different randomly selected subset of the dataset. As a result, every tree learns slightly different patterns.
- **Selecting Random Features:** While creating each tree, only a random subset of features is considered at each split instead of using all available features. This

SGOU

randomness increases diversity among the trees.

- **Individual Predictions:** Each decision tree independently produces its own prediction based on the data it was trained on.
- **Aggregating Results:** The final output is obtained by combining the predictions of all trees. In classification tasks, the result is determined by majority voting among the trees, while in regression, the predictions are averaged.
- **Effectiveness:** The use of random sampling and feature selection helps reduce overfitting and improves the reliability and accuracy of the overall model.

2.3.5.2 Key Features of Random Forest

- **Handles Missing Values:** The algorithm can manage datasets with missing entries, reducing the need for extensive data preprocessing.
- **Identifies Important Features:** It provides insights into which features contribute most to predictions, helping in better data understanding and feature selection.
- **Efficient with Large and Complex Data:** Random Forest performs well on large datasets with numerous features, maintaining accuracy without significant performance issues.
- **Versatile in Applications:** It can be applied to both classification tasks (predicting categories) and regression tasks (predicting numerical values).

2.3.5.3 Assumptions of Random Forest

- **Independent Tree Decisions:** Each tree in the forest makes predictions independently without depending on other trees.
- **Use of Random Sampling:** Trees are trained on randomly selected subsets of data and features to minimize errors and increase diversity.
- **Requirement of Sufficient Data:** A large dataset is necessary to ensure that individual trees capture varied patterns effectively.
- **Improved Accuracy through Aggregation:** Combining predictions from multiple diverse trees leads to a more accurate and stable overall result.

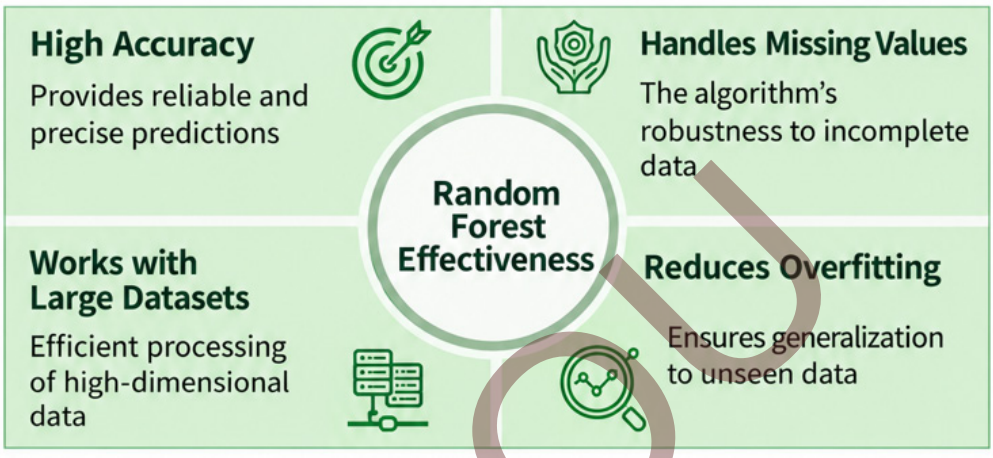
2.3.5.4 Steps for Implementing Random Forest for Classification Tasks

In this example, the goal is to predict whether a passenger survived the Titanic disaster using a Random Forest classifier.

- Begin by importing essential libraries such as pandas and scikit-learn.
- Load the Titanic dataset into your working environment.

SGOU

- Clean the data by removing records where the target variable (*Survived*) is missing.
- Choose relevant input features like passenger class, gender, age, and others, and convert categorical values such as *Sex* into numerical form.
- Handle missing values in the *Age* column by replacing them with the median value.
- Divide the dataset into training and testing subsets, then train the Random Forest model using the training data.
- Finally, make predictions on the test set, evaluate the model's accuracy, and display a few sample prediction results.



2.3.7 Effectiveness of Random Forest

2.3.5.5 Advantages of Random Forest

- Random Forest delivers high predictive accuracy, even when working with large and complex datasets.
- It can effectively manage missing values without significantly affecting performance.
- The algorithm does not require data normalization or standardization, making preprocessing simpler.
- By aggregating multiple decision trees, it minimizes the chances of overfitting and improves model stability. See fig. 2.3.7

2.3.5.6 Limitations of Randomorest

- It can be resource-intensive, particularly when a large number of trees are used, leading to higher computational cost.
- The model is less interpretable compared to simpler approaches such as a single decision tree, making it harder to understand the decision-making process.

SGOU

2.3.6 Gradient Boosting

Gradient Boosting is an ensemble learning method where models are constructed sequentially, and each new model attempts to correct the errors made by previous ones. This is accomplished by minimizing a chosen loss function, such as mean squared error for regression or cross entropy for classification, using a gradient descent approach. At every iteration, the algorithm calculates the gradient of the loss with respect to the current predictions and trains a weak learner to approximate this gradient. The predictions from this learner are then added to the overall ensemble, and the process continues until a specified stopping condition is reached.

2.3.6.1 Shrinkage and Model Complexity

A key concept in Gradient Boosting is shrinkage, which regulates the contribution of each new model using a parameter known as the learning rate (η).

- **Smaller learning rates:** Limit the influence of individual trees, helping to reduce overfitting, but require a larger number of trees to achieve high performance.
- **Larger learning rates:** Increase the contribution of each tree, allowing faster learning but potentially causing overfitting.

This creates a trade-off between the learning rate and the number of trees, where lower learning rates typically demand more trees for optimal results.

2.3.6.2 Working of Gradient Boosting

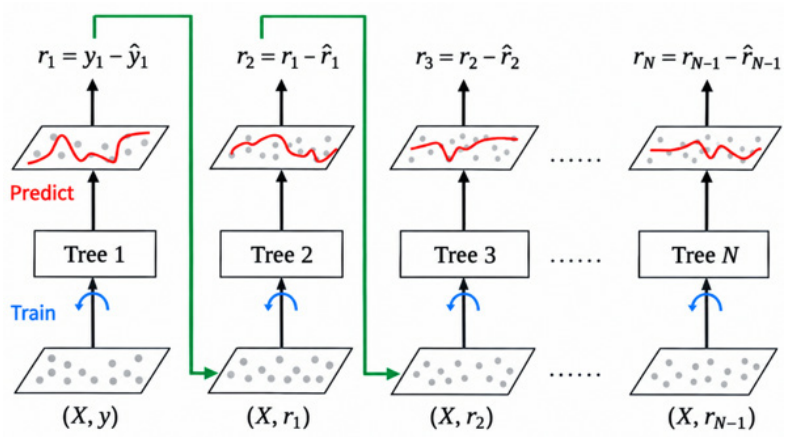
- **Sequential Learning Process**

The model is built as a sequence of trees, where each tree is designed to improve upon the mistakes of the one before it. In the initial step, the first tree is trained using the original input data x along with the actual target values y. It generates predictions, which are then compared with the true values to calculate the errors.

- **Residuals Calculation**

In the next step, the second tree is trained using the same input features x, but instead of the original labels, it uses the errors produced by the first tree as targets. In this way, the second tree learns to model the mistakes made by the first. This process is repeated throughout the ensemble, with each new tree focusing on predicting the residual errors of the preceding tree.

SGOU



2.3.8 Gradient Boosted Trees

♦ **Shrinkage**

After training each tree, its output is scaled down by multiplying it with the learning rate η, which typically lies between 0 and 1. This step controls the influence of each tree, helping to prevent overfitting by ensuring that no single model dominates the final prediction. See fig. 2.3.8

Once all the trees are built, the final prediction is obtained by adding together the contributions from each tree:

$$y_{pred} = y_1 + \eta r_1 + \eta r_2 + \ldots + \eta r_N$$

Here, $r_1, r_2, \ldots, r_N$ represent the residuals (errors) predicted by each successive tree, and their scaled contributions collectively form the final model output.

2.3.6.3 Difference Between AdaBoost and Gradient Boosting

The following points highlight the key differences between AdaBoost and Gradient Boosting:

♦ **Weight Update Strategy:**

AdaBoost increases the weights of misclassified samples so that subsequent learners focus more on those difficult cases. In contrast, Gradient Boosting improves predictions by minimizing a loss function using the negative gradient.

♦ **Base Learners:**

AdaBoost typically relies on very simple models, such as decision stumps (single split trees). Gradient Boosting, however, can employ a variety of base learners, including more complex decision trees or even linear models.

♦ **Sensitivity to Noise:**

AdaBoost is more prone to the effects of noisy data and outliers because it heavily emphasizes misclassified points. Gradient Boosting is comparatively more robust, as it refines predictions gradually through gradient based updates.

SGOU

- **Optimization Approach:**

AdaBoost does not explicitly optimize a defined loss function and mainly focuses on reducing classification errors. Gradient Boosting, on the other hand, directly minimizes a differentiable loss function.

- **Boosting Mechanism:**

In AdaBoost, models are trained sequentially with a focus on reweighting training samples. In Gradient Boosting, models are also trained sequentially, but each new model learns from the residual errors using a gradient descent method.

- **Interpretability:**

AdaBoost models are generally easier to interpret due to their use of simple weak learners. Gradient Boosting models can become more complex and harder to interpret, especially when deeper trees are used.

- **Use Cases:**

AdaBoost is more suitable for relatively clean datasets with fewer outliers, whereas Gradient Boosting is better suited for complex problems and scenarios involving different types of loss functions.

2.3.6.4 Implementing Gradient Boosting for Classification and Regression

The following examples illustrate how Gradient Boosting can be applied to both classification and regression tasks. Before exploring the examples, it is important to understand some key parameters used in the model.

- **n_estimators:** Determines the number of trees to be created. Increasing this value can improve performance, but it also raises computational cost.
- **learning_rate:** Controls the contribution of each tree by scaling its output, helping to manage overfitting.
- **random_state:** Ensures consistent and reproducible results by fixing randomness in the model.
- **max_features:** Limits the number of features considered for splitting at each node, which helps reduce overfitting and encourages model diversity.

With these parameters in mind, we can proceed to build Gradient Boosting models.

Example 1: Classification

In this example, a Gradient Boosting Classifier is used to recognize handwritten digits from a dataset. The process involves importing required libraries, setting a seed value for reproducibility, loading the dataset, splitting it into training and testing sets, training the model, and finally evaluating its accuracy.

SGOU

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_digits
SEED = 23
X, y = load_digits(return_X_y=True)
train_X, test_X, train_y, test_y = train_test_split(
  X, y, test_size=0.25, random_state=SEED
)
gbc = GradientBoostingClassifier(
  n_estimators=300,
  learning_rate=0.05,
  random_state=100,
  max_features=5
)
gbc.fit(train_X, train_y)
pred_y = gbc.predict(test_X)
acc = accuracy_score(test_y, pred_y)
print("Gradient Boosting Classifier accuracy is : {:.2f}".format(acc))
```

Output:

Gradient Boosting Classifier accuracy is : 0.98

Example 2: Regression

Here, a Gradient Boosting Regressor is applied to the diabetes dataset to predict continuous target values. The steps are similar: import libraries, prepare the dataset, train the model, and evaluate its performance using Root Mean Square Error (RMSE).

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.datasets import load_diabetes
SEED = 23
X, y = load_diabetes(return_X_y=True)
train_X, test_X, train_y, test_y = train_test_split(
  X, y, test_size=0.25, random_state=SEED
)
gbr = GradientBoostingRegressor(
```

SGOU

```
    loss='absolute_error',
    learning_rate=0.1,
    n_estimators=300,
    max_depth=1,
    random_state=SEED,
    max_features=5
)
gbr.fit(train_X, train_y)
pred_y = gbr.predict(test_X)
test_rmse = mean_squared_error(test_y, pred_y) ** 0.5
print("Root mean Square error: {:.2f}".format(test_rmse))
```

Output:

Root mean Square Error: 56.39

Gradient Boosting is a powerful and widely adopted machine learning method for both classification and regression. By building models sequentially and continuously correcting previous errors, it achieves high predictive accuracy and strong performance across a variety of real-world problems.

Summerised Overview

This section introduces important machine learning concepts and algorithms used for classification and prediction. A hyperplane is a decision boundary that separates classes in a feature space, represented by a line in two dimensions and a plane in three dimensions. Support Vector Machines (SVMs) identify a suitable separating hyperplane by maximizing the margin between the nearest data points of different classes. These nearest points are called support vectors because they determine the position of the decision boundary. Hard margin SVM requires perfect separation, whereas soft margin SVM allows certain violations to handle noisy and overlapping data. Linear SVM is suitable for linearly separable data, while non linear SVM uses kernel functions to model complex decision boundaries. The kernel trick enables SVMs to operate in higher dimensional feature spaces without explicitly calculating the transformed coordinates. Common kernel functions include linear, polynomial, Radial Basis Function (RBF) and sigmoid kernels. Ensemble learning improves predictive performance by combining multiple learning models. Bagging trains models independently on bootstrap samples and combines their predictions to reduce variance and improve stability. Boosting builds models sequentially, with subsequent models focusing on errors or optimizing a loss function to improve overall predictions. Random Forest is a bagging based ensemble

SGOU

algorithm that combines multiple decision trees using random sampling and feature selection. Its final prediction is obtained through majority voting for classification or averaging for regression. Gradient Boosting builds successive weak learners that improve the ensemble by minimizing a specified loss function, with the learning rate controlling each learner's contribution. Understanding these concepts and algorithms helps learners select suitable classification and regression techniques for practical applications, evaluate their strengths and limitations, and develop effective machine learning solutions.

Assignments

1. Explain the concept of a hyperplane in Support Vector Machines and describe its role as a decision boundary in different dimensions.
2. Discuss the concept of margin in SVM and explain why maximizing the margin leads to better classification performance.
3. Differentiate between hard margin and soft margin SVM with suitable examples and explain their practical significance.
4. Describe the role of support vectors in SVM and explain why they are considered the most important data points.
5. Explain the working of Support Vector Machine step by step, including margin maximization and the objective function.
6. Discuss the kernel trick in SVM and explain how it helps in handling non linear data with suitable examples.
7. Compare bagging and boosting techniques in ensemble learning and explain when each method should be used.
8. Explain the working of Random Forest and Gradient Boosting algorithms and discuss how they improve model accuracy and reduce errors.

SGOU

Reference

1. Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.
2. Alpaydin, E. (2016). *Machine learning*. MIT Press.
3. Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
4. Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

Suggested Reading

1. Alpaydin, E. (2016). *Machine learning: The new AI*. MIT Press.
2. Burkov, A. (2019). *The hundred-page machine learning book*. Andriy Burkov.
3. Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.
4. Theobald, O. (2017). *Machine learning for absolute beginners*. Scatterplot Press.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

4 UNIT Model Tuning and Cross-validation

Learning Outcomes

After completion of this unit, the learner will be able to:

- understand the basic idea of hyperparameter tuning and cross validation
- explain how grid search and k fold cross validation work
- apply tuning techniques to improve a machine learning model
- choose suitable hyperparameters to get better accuracy

Background

Model tuning and cross validation are essential concepts in machine learning that help improve the performance and reliability of predictive models. Machine learning models often depend on hyperparameters, which must be set before training and can significantly influence the results. Without proper tuning, models may either overfit the training data or underfit and fail to capture important patterns. Techniques such as grid search allow systematic exploration of different parameter combinations to find the best configuration. Similarly, k fold cross validation provides a reliable way to evaluate model performance by using different portions of the data for training and testing. These methods help ensure that the model generalizes well to unseen data rather than just memorizing the training dataset. Studying these techniques is important for building accurate and robust machine learning systems. They also help in making better use of available data, especially when datasets are limited. In real world applications, properly tuned models can lead to better decision making and improved outcomes. Therefore, understanding model tuning and cross validation is necessary for anyone working in data science and machine learning.

SGOU

Keywords

grid search, randomized search, bayesian optimization, k fold cross validation, model performance, overfitting

Discussion

Hyperparameter Tuning

Hyperparameter tuning refers to the process of choosing the most suitable values for a model's hyperparameters. These parameters are defined before training begins and govern how the learning process operates. Proper tuning enables the model to capture meaningful patterns more effectively, reduce the chances of overfitting or underfitting, and improve its performance on new, unseen data.

2.4.1 Techniques for Hyperparameter Tuning

Machine learning models often include multiple hyperparameters, and identifying the optimal combination can be viewed as a search problem. Among the various approaches available, the widely used strategies for hyperparameter tuning are discussed below.

2.4.1.1 Grid SearchCV

Grid SearchCV is an exhaustive hyperparameter tuning method that evaluates all possible combinations of predefined hyperparameter values to identify the configuration that produces the best model performance. Although it provides a systematic approach to hyperparameter optimization, it can be computationally expensive and time-consuming when the number of parameters and their possible values increases. See fig. 2.4.1.

It typically follows these steps:

- **Define the parameter grid:** Specify the possible values for each hyperparameter to be evaluated.
- **Train the model:** Evaluate the model using every possible combination of the specified hyperparameter values.
- **Perform cross-validation:** Assess the performance of each combination using cross-validation.
- **Select the best configuration:** Identify the combination that achieves the highest performance score.

SGOU

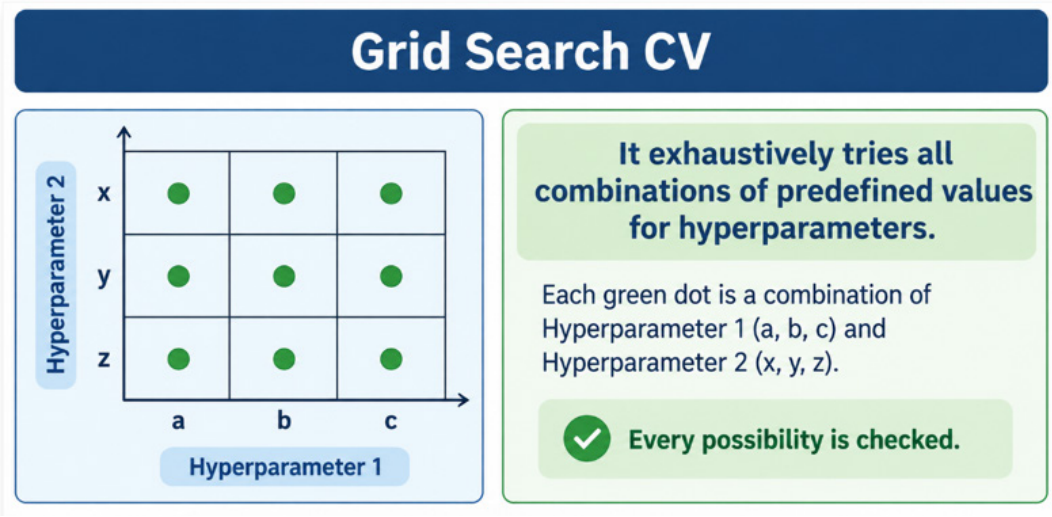


Fig. 2.4.1 Grid Search CV

For example, consider tuning two hyperparameters, C and α, for a machine learning model using the following values:

- C = [0.1, 0.2, 0.3, 0.4, 0.5]
- α =[0.1, 0.2, 0.3, 0.4]

The performance scores obtained for different combinations of these hyperparameters are shown in the following table.

Table 2.4.1 Performance scores obtained for different combinations of hyperparameters, C and α

C	α= 0.1	α=0.2	α=0.3	α=0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	**0.726**	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

In this example, GridSearchCV evaluates all 5×4=20 possible combinations of the hyperparameters. The table presents the corresponding performance scores for each combination.

The best combination is C=0.3 and α=0.2, which achieves the highest performance score of 0.726. Therefore, this combination is selected as the optimal configuration among the tested values.

GridSearchCV helps identify the most suitable hyperparameter configuration through systematic evaluation. However, its computational cost increases as the number of hyperparameters and their candidate values grows.

SGOU

Example: Tuning Logistic Regression Using GridSearchCV

The example below demonstrates how GridSearchCV can be applied in practice. In this case:

- A sample dataset is created using the make_classification function.
- A set of possible values for the hyperparameter C is defined on a logarithmic scale.
- GridSearchCV evaluates every combination specified in the param_grid using 5 fold cross validation.
- Finally, it identifies and returns the optimal value of C along with the corresponding validation score

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
import numpy as np
from sklearn.datasets import make_classification
X, y = make_classification(
n_samples=1000, n_features=20, n_informative=10, n_classes=2, random_state=42)
c_space = np.logspace(-5, 8, 15)

param_grid = {
   'C': c_space,
   'penalty': ['l1', 'l2']
}
logreg = LogisticRegression(solver='liblinear')
logreg_cv = GridSearchCV(logreg, param_grid, cv=5)
logreg_cv.fit(X, y)
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
```

This example demonstrates how GridSearchCV can be applied to optimize the performance of a Logistic Regression model.

First, a synthetic dataset is generated using the make_classification function, which creates a binary classification problem with a specified number of samples and features. Next, a range of values for the hyperparameter C is defined using a logarithmic scale,

SGOU

allowing the model to explore both very small and very large values.

A parameter grid is then created, which includes different values of C along with two types of regularization penalties (l1 and l2). The Logistic Regression model is initialized with the liblinear solver, as it supports both penalty types.

GridSearchCV is then used with 5 fold cross validation to train and evaluate the model across all possible combinations of the specified parameters. After training, it identifies the best performing parameter combination and provides the corresponding evaluation score.

Output:

Optimal parameters: {'C': 0.006105402296585327}

Best cross validation score: 0.853

This example shows how GridSearchCV systematically searches through parameter combinations to select the most effective configuration for improving model performance.

2.4.1.2 Randomized Search CV

RandomizedSearchCV, as the name implies, selects random combinations of hyperparameters from predefined ranges instead of exhaustively evaluating every possible combination like GridSearchCV. This makes it more efficient, especially when dealing with large search spaces. See fig.2.4.2

- In each iteration, it samples a new random set of hyperparameter values.
- The model is trained and evaluated for each selected combination.
- After a specified number of iterations, the best-performing configuration is identified.

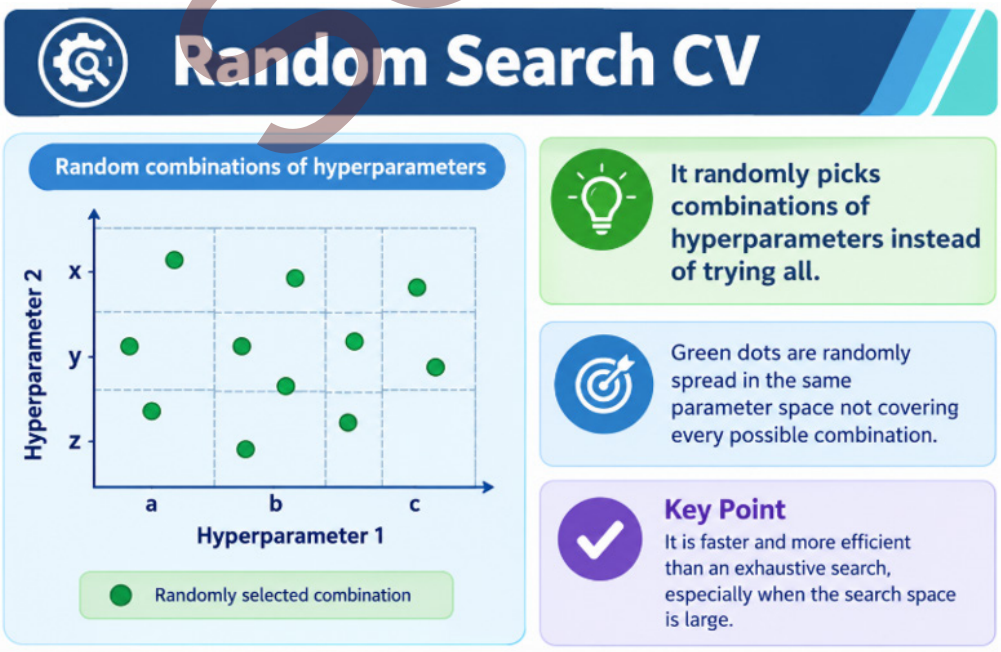


Fig. 2.4.2 Random Search CV

SGOU

Example: Tuning a Decision Tree Using RandomizedSearchCV

The following example demonstrates the use of RandomizedSearchCV. In this case:

- A range of values is defined for different hyperparameters such as *max_depth* and *min_samples_leaf*.
- Random combinations of these parameters are selected and evaluated using 5 fold cross validation.
- Finally, the method outputs the optimal parameter combination along with its corresponding performance score.

```
import numpy as np

from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=20, n_informative=10, n_classes=2, random_state=42)

from scipy.stats import randint

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import RandomizedSearchCV


param_dist = {
    "max_depth": [3, None],
    "max_features": randint(1, 9),
    "min_samples_leaf": randint(1, 9),
    "criterion": ["gini", "entropy"]
}


tree = DecisionTreeClassifier()

tree_cv = RandomizedSearchCV(tree, param_dist, cv=5)

tree_cv.fit(X, y)


print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))

print("Best score is {}".format(tree_cv.best_score_))
```

SGOU

Output:

Tuned Decision Tree Parameters: {'criterion': 'entropy', 'max_depth': None, 'max_features': 6, 'min_samples_leaf': 6}
Best score is 0.8

A score of 0.842 means the model performed with an accuracy of 84.2% on the validation set with following hyperparameters.

2.4.1.3 Bayesian Optimization

Unlike Grid Search and Random Search, which explore hyperparameter combinations without considering past outcomes, Bayesian Optimization follows a more intelligent and guided approach. It treats hyperparameter tuning as an optimization problem and uses previous evaluation results to decide which configurations to test next. See fig. 2.4.3

- It first constructs a probabilistic model, known as a surrogate function, to estimate model performance based on different hyperparameter values.
- After each evaluation, this model is updated with new results.
- The updated model is then used to select the most promising set of hyperparameters for the next trial.
- This process continues iteratively until an optimal or near optimal configuration is found.

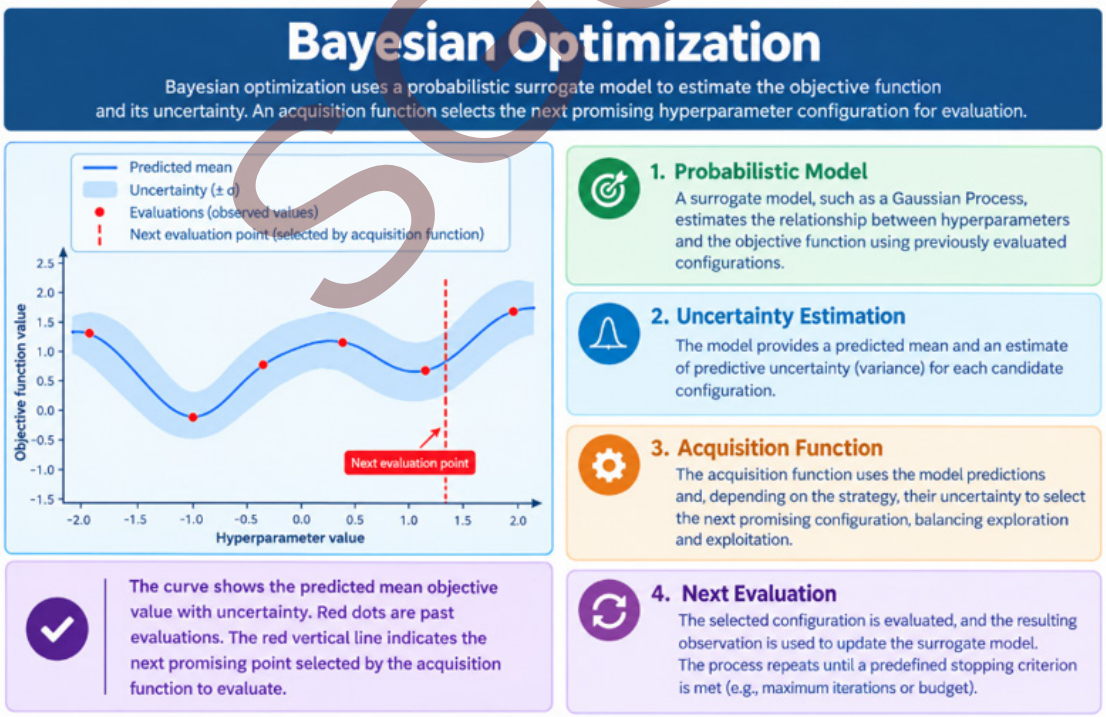


Fig. 2.4.3 Bayesian Optimization

The surrogate model represents the relationship between hyperparameters (x) and the resulting performance score (y)

P(score(y)| hyperparameters(x))

By continuously refining this relationship, Bayesian Optimization becomes more efficient in selecting better parameter combinations over time. Common surrogate models used include:

- Gaussian Processes
- Random Forest Regression
- Tree structured Parzen Estimators (TPE)

Comparing Randomized Search and Grid Search for Hyperparameter Estimation in Scikit Learn.

Hyperparameters are configuration settings that control how a machine learning model behaves and performs. Unlike model parameters, they are not learned from the data during training but must be defined beforehand. Determining the best values for these settings is known as hyperparameter optimization, which plays a crucial role in building effective machine learning models.

There are several approaches to hyperparameter tuning, but two of the most widely used techniques are grid search and randomized search. These methods help identify the most suitable parameter combinations for improving model performance.

2.4.1.4 Grid Search for Hyperparameter Optimization

Grid search is a systematic approach in which a set of possible values is defined for each hyperparameter. The algorithm then trains and evaluates a model for every possible combination of these values. For example, if two hyperparameters such as alpha and beta are to be tuned, a list of candidate values is specified for each. The method then tests all combinations formed from these lists and selects the one that produces the best performance.

To obtain reliable results, grid search is often combined with cross validation. In this technique, the training data is divided into multiple subsets, and the model is trained several times, each time using a different subset for validation. This provides a more robust estimate of performance and helps reduce the risk of overfitting.

Example: Implementing Grid Search in Scikit Learn

The following example shows how GridSearchCV can be used to determine the best hyperparameter values for a model.

```
from sklearn.model_selection import GridSearchCV

# Define the hyperparameters and their possible values

param_grid = {
```

SGOU

```
    'alpha': [0.01, 0.1, 1.0, 10.0],
    'beta': [0.01, 0.1, 1.0, 10.0]
}
# Create a model
model = SomeModel()
# Apply grid search to find the best hyperparameters
grid_search = GridSearchCV(model, param_grid)
grid_search.fit(X, y)
# Display the optimal hyperparameter values
print(grid_search.best_params_)
```

In this example, a dictionary named param_grid is defined to specify different possible values for the hyperparameters alpha and beta. The GridSearchCV class from the sklearn.model_selection module is then used to evaluate all combinations of these values. The fit method trains the model for each combination, and the best_params_ attribute provides the set of hyperparameters that yields the best performance.

Limitation of Grid Search

A major drawback of grid search is its high computational cost. When there are many hyperparameters or a large range of values for each, the total number of combinations increases rapidly. As a result, the method requires training a large number of models, which can make it slow and impractical for complex problems.

2.4.1.5 Advantages of Hyperparameter Tuning

- Identifying the best hyperparameter settings can greatly improve model accuracy and reliability.
- It helps balance the model by reducing both overfitting and underfitting.
- Proper tuning enhances the model's ability to generalize well to new, unseen data.
- It ensures more efficient use of computational resources by avoiding unnecessary trials.
- Well tuned models are often simpler and easier to interpret.

2.4.1.6 Challenges of Hyperparameter Tuning

- Large hyperparameter spaces increase the number of possible combinations, making the tuning process computationally intensive and time consuming, especially for complex models.

SGOU

- Incorporating prior knowledge can help reduce the search space and improve efficiency.
- Adjusting hyperparameters dynamically during training, such as using learning rate schedules or early stopping, can further enhance model performance.

2.4.2 K Fold Cross Validation in Machine Learning

K Fold Cross Validation is a statistical method used to evaluate the performance of a machine learning model by splitting the dataset into K equal parts, known as folds. In each iteration, the model is trained on K minus 1 folds and tested on the remaining fold. This procedure is repeated K times so that every fold serves as the test set exactly once. The final performance is calculated by averaging the results from all iterations, providing a reliable estimate of how well the model generalizes to unseen data. See fig. 2.4.4

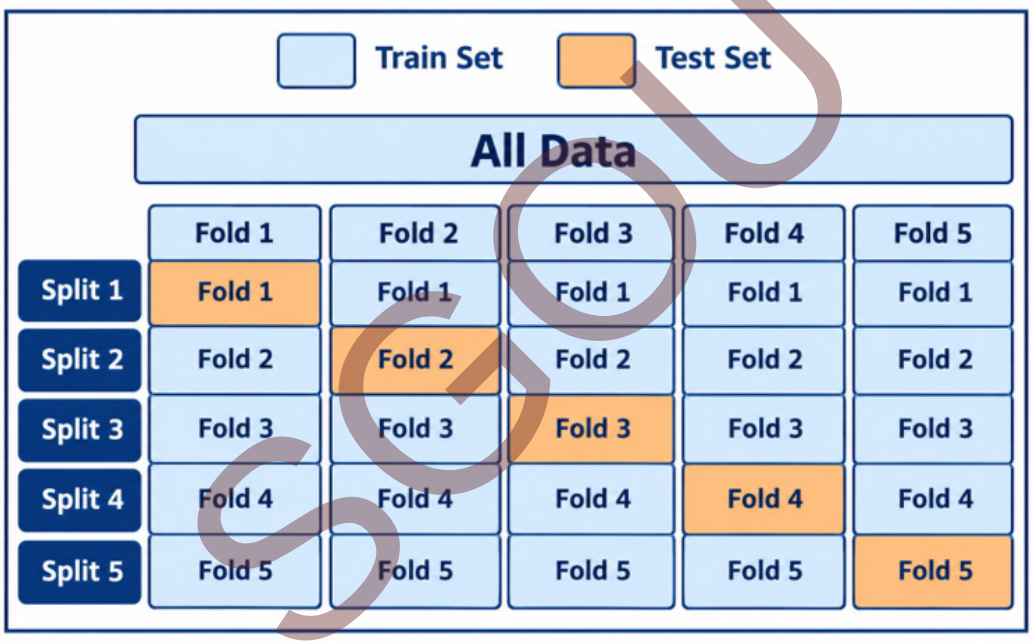


Fig. 2.4.4 K-Fold train and test split

2.4.2.1 What is Cross Validation?

Cross validation is used for several important reasons:

- Prevents Overfitting: It helps ensure that the model performs well not only on the training data but also on new, unseen data.
- Offers Reliable Evaluation: By averaging results across multiple runs, it reduces bias and variability in performance measurements.
- Makes Better Use of Data: It allows the entire dataset to be utilized effectively, which is particularly valuable when the available data is limited.

2.4.2.2 K Fold Cross Validation Steps

The procedure can be described in the following steps:

1. **Divide the Dataset:** Split the entire dataset into K equal subsets, known as folds.

2. **Training and Testing Cycle:** For each fold:

   - Train the model using K minus 1 folds.
   - Use the remaining fold as the test set to evaluate the model's performance.

3. Combining Results

After completing all iterations, the performance measure such as accuracy, precision, or recall is calculated for each fold and then averaged to obtain an overall evaluation of the model.

Let D represent the dataset divided into K folds. For the k-th iteration, the training set consists of all data except the k-th fold, while the test set is the k-th fold itself:

D(k)train=D / Fk

D(k)test=Fk

The overall model performance is then computed as the average of the evaluation metric across all folds:

$$\text{Performance}=\frac{1}{K}\sum_{k=1}^{K} \text{Metric}(Mk,Fk)$$

Selecting the Value of K

The choice of K influences the balance between bias and variance:

- Smaller values of K (such as 2 or 5) reduce computation time but may result in higher variance in performance estimates.
- Larger values of K (such as 10 or equal to the dataset size nnn) provide more stable estimates but require greater computational effort. When K=n, the method becomes Leave-One-Out Cross Validation (LOOCV).

In practice, 10 fold cross validation is commonly preferred as it offers a good balance between computational efficiency and reliable performance estimation.

2.4.2.3 Benefits of K Fold Cross Validation

- **Efficient Use of Data:** Each data point is utilized in both training and testing phases, ensuring maximum use of the dataset.
- **More Reliable Evaluation:** It provides a balanced estimate of model performance, reducing the chances of overfitting or underfitting.

SGOU

- **Suitable for Limited Data:** It is particularly useful when the dataset is small or when collecting data is costly.
- **Adaptation for Time Series:** For time dependent data, a variation called Time Series Cross Validation is applied, where training uses data up to a certain time t, and testing is performed on data from time t+1 onward.

2.4.2.4 Limitations of K Fold Cross Validation

- High Computational Effort: Since the model must be trained multiple times, the process can be slow for large datasets or complex algorithms.
- Risk of Data Leakage: Proper care is required during preprocessing to prevent information from the test set from influencing the training process.
- Challenges with Time Series Data: Standard K Fold may not be suitable for time dependent datasets due to their sequential nature, unless modified approaches such as time based splitting are used.

2.4.2.5 Implementation of K Fold Cross Validation

The following Python example demonstrates how to perform K Fold Cross Validation using the scikit learn library:

Python code:

```
from sklearn.model_selection import KFold, cross_val_score

from sklearn.datasets import load_iris

from sklearn.ensemble import RandomForestClassifier

import numpy as np


# Load the Iris dataset

data = load_iris()

X, y = data.data, data.target


# Define the number of folds

k = 5

kf = KFold(n_splits=k, shuffle=True, random_state=42)


# Initialize the model
```

SGOU

```
model = RandomForestClassifier(random_state=42)

# Apply cross-validation
scores = cross_val_score(
    model, X, y, cv=kf, scoring='accuracy'
)
print(f"Accuracy for each fold: {scores}")

# Calculate the average accuracy
average_accuracy = np.mean(scores)
print(f"Average Accuracy: {average_accuracy:.2f}")
```

Output:

Accuracy for each fold: [1.000, 0.967, 0.933, 0.933, 0.967]

Average Accuracy: 0.96

Explanation:

This example applies five-fold cross-validation to evaluate a Random Forest Classifier using the Iris dataset, which contains 150 samples. The dataset is divided into five approximately equal-sized folds, with 30 samples in each fold.

During each iteration, four folds (120 samples) are used to train the model, while the remaining fold (30 samples) is used to evaluate its performance. This process is repeated five times, ensuring that each fold serves as the test set once.

The cross_val_score() function returns the accuracy score for each fold. The np.mean() function calculates the average of these five scores, which is approximately **0.96 (96%)** in this example.

Thus, K-Fold Cross-Validation provides a more comprehensive assessment of model performance than a single train-test split because it evaluates the model across multiple subsets of the dataset.

SGOU

Summerised Overview

Hyperparameter tuning is the process of selecting the best configuration values that control how a machine learning model learns before training begins. It plays a crucial role in improving model accuracy, reducing overfitting or underfitting, and enhancing generalization to new data. This process can be viewed as a search problem where different combinations of parameters are explored to find the most effective setup. GridSearchCV is a common method that evaluates all possible combinations of predefined parameter values, making it thorough but computationally expensive. In contrast, RandomizedSearchCV selects random combinations from given ranges, making it faster and more efficient for large search spaces. Bayesian Optimization offers a more intelligent approach by learning from previous results and selecting promising parameter combinations iteratively. Hyperparameter tuning significantly improves model reliability, interpretability, and efficient use of computational resources. However, it can be time consuming and computationally intensive, especially with large parameter spaces. To evaluate model performance during tuning, K Fold Cross Validation is widely used, where the dataset is divided into K subsets and the model is trained and tested multiple times. This method provides a reliable estimate of model performance by averaging results across all folds. Overall, combining hyperparameter tuning with cross validation leads to more robust and well generalizing machine learning models.

SGOU

Assignments

1. Explain the concept of hyperparameter tuning in machine learning and discuss its importance in improving model performance.

2. Describe the working of GridSearchCV and explain the steps involved in finding the optimal hyperparameters.

3. Compare GridSearchCV and RandomizedSearchCV in terms of methodology, efficiency, and use cases.

4. Explain how Logistic Regression can be tuned using GridSearchCV with suitable parameters and cross validation.

5. Discuss RandomizedSearchCV and explain why it is preferred for large hyperparameter search spaces.

6. Explain the concept of Bayesian Optimization and how it improves the process of hyperparameter tuning.

7. What is K Fold Cross Validation and how does it help in evaluating machine learning models?

8. Discuss the advantages and limitations of K Fold Cross Validation and explain the impact of choosing different values of K.

Suggested Reading

1. Alpaydin, E. (2016). *Machine learning: The new AI*. MIT Press.

2. Burkov, A. (2019). *The hundred-page machine learning book*. Andriy Burkov.

3. Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

4. Theobald, O. (2017). *Machine learning for absolute beginners*. Scatterplot Press.

Reference

1. Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

2. Alpaydin, E. (2016). Machine learning. MIT Press.

3. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

4. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

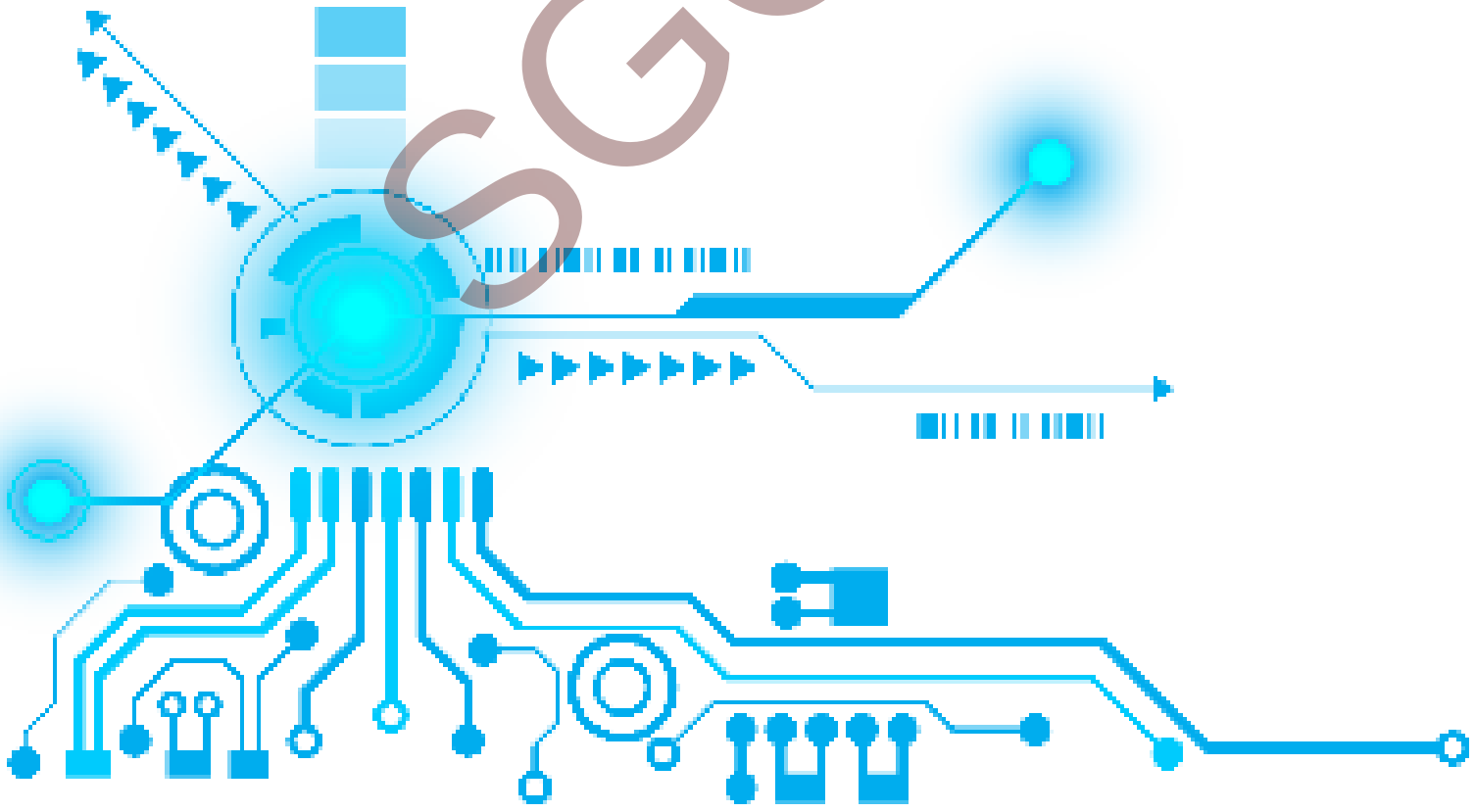
SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

BLOCK 3
Unsupervised Learning



SGOU

1 UNIT

Clustering Techniques

Learning Outcomes

After completion of this unit, the learner will be able to:

- understand the basic concept and importance of clustering techniques.
- explain how k-Means, Hierarchical Clustering, and DBSCAN algorithms work.
- compare different clustering methods and their applications.
- apply clustering techniques to group similar data points.
- evaluate the quality of clusters using different evaluation metrics.

SGOU

Background

Clustering techniques are used to group similar data items together so that patterns in data can be easily understood. In real life, we often do clustering without realizing it. For example, a teacher may group students based on their performance levels such as high, medium, and low. Similarly, in online shopping, customers are grouped based on their buying behavior. Techniques like k-Means divide data into a fixed number of groups, while Hierarchical Clustering creates a tree-like structure of groups, and DBSCAN forms clusters based on dense regions of data.

Different clustering methods work in different ways, so it is important to evaluate how well the clusters are formed. For instance, if we group fruits based on size and color, a good clustering method will clearly separate apples, oranges, and grapes into different groups. Cluster evaluation metrics help measure the quality of these groupings by checking how similar the items are within a cluster and how different they are from other clusters. This helps in selecting the best clustering technique for a given problem and improves decision-making in fields like business, healthcare, and data analysis.

Keywords

Centroid, Dendrogram, Density-Based Clustering, Silhouette Score, Noise Points

Discussion

Clustering techniques are an important part of data analysis used to group similar data points into meaningful clusters without prior labels. This unit introduces popular clustering methods such as k-Means, hierarchical clustering, and DBSCAN, each with different approaches to identifying patterns and structures in data. It also covers cluster evaluation metrics, which help in measuring the quality and effectiveness of the formed clusters. Understanding these techniques enables better data exploration, pattern recognition, and decision-making in various real-world applications.

3.1.1 Introduction to Clustering

Clustering is a way of organizing data by grouping similar items together so that hidden patterns can be discovered easily. Imagine entering a supermarket where items are arranged into sections like fruits, vegetables, and dairy products without any labels given. You can still identify the groups based on similarities. In the same way, clustering algorithms automatically find such groupings in data based on common features. This concept helps learners understand how large amounts of raw data can be structured meaningfully, making it easier to analyze and draw useful conclusions.

3.1.1.1 Definition of Clustering

Clustering (Fig. 3.1.1) is an unsupervised machine learning technique used to group similar data points together without using labelled data. It helps in discovering hidden patterns or natural groupings in a dataset by placing similar items into the same cluster. The main objective is to identify the natural structure present in unlabelled data without any predefined categories or prior knowledge about the groups.

In clustering, data points are assigned to different clusters based on their similarity or distance from one another. Various measures such as Euclidean distance, cosine similarity, or other metrics are used depending on the type of data and the clustering method applied. This approach is useful in organizing complex data, making it easier to analyze patterns and extract meaningful insights.

SGOU

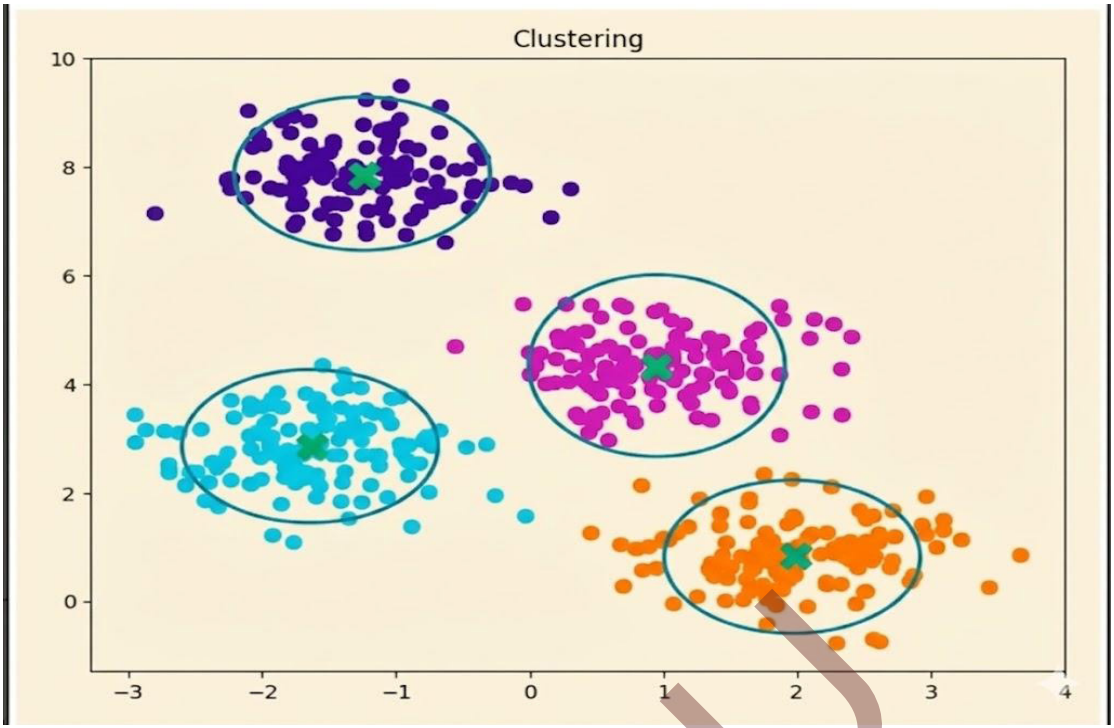


Fig. 3.1.1 Clustering

3.1.1.2 Importance of Clustering in Data Analysis

Clustering plays a key role in data analysis by helping to organize large and complex datasets into smaller, meaningful groups. When data is unstructured or difficult to interpret, clustering makes it easier to understand by highlighting similarities among data points. This allows analysts to quickly identify patterns, trends, or relationships that may not be visible at first glance, improving the overall understanding of the dataset.

It is also valuable for decision-making in many real-world applications. For example, businesses can segment customers to design targeted marketing strategies, while researchers can group similar observations for better study and analysis. By simplifying data and revealing useful insights, clustering supports efficient analysis and helps in making informed decisions across different fields.

3.1.1.3 Types of Clustering (Hard vs Soft Clustering)

Clustering methods can be broadly divided based on how they assign data points to groups. Some methods create strict boundaries between clusters, while others allow flexibility when data points share characteristics with multiple groups. Understanding these types helps in choosing the right approach for different kinds of data.

1. Hard Clustering

Hard clustering follows a strict grouping approach where each data point is placed into one and only one cluster. This method works well when the data can be clearly separated into distinct groups without any ambiguity.

SGOU

Key Features

- Each data point belongs to a single cluster only
- No overlapping between clusters
- Easy to understand and interpret

Example

Consider a classroom where students are grouped based on pass and fail categories. Each student is placed in only one category, making the grouping very clear.

Applications

- Market segmentation for targeted advertising
- Customer categorization for business analysis
- Grouping documents by topic
- Advantages
- Simple and easy to understand
- Clear separation between clusters
- Works well when data groups are distinct
- Less computational complexity compared to soft clustering

Limitations

- Cannot represent situations where data points share properties of multiple groups
- Cannot handle overlapping data
- Not suitable for complex or ambiguous datasets
- May give less accurate results when boundaries are unclear

2. Soft Clustering

Soft clustering provides a flexible grouping method where a data point can belong to more than one cluster. It is useful when the boundaries between groups are not clearly defined.

Key Features

- Data points can belong to multiple clusters
- Membership is expressed in degrees or probabilities
- Captures uncertainty in data

SGOU

Example

In movie recommendation systems, a film can belong partly to both "comedy" and "romance" categories, showing characteristics of both.

Applications

- Customer behavior analysis
- Medical diagnosis with overlapping symptoms
- Pattern recognition tasks
- Advantages
- Allows a data point to belong to multiple clusters
- Handles overlapping data effectively
- Provides more realistic grouping in complex datasets
- Useful for real-world problems with unclear boundaries

Limitations

- Higher computational cost
- More complex compared to hard clustering
- Requires careful selection of parameters
- May produce less clear cluster boundaries

In summary, hard and soft clustering represent two different ways of assigning data points to groups. Hard clustering provides clear and definite groupings where each data point belongs to only one cluster, making it simple and easy to interpret. In contrast, soft clustering offers flexibility by allowing data points to belong to multiple clusters with varying degrees of membership, which is useful when data is complex or overlapping. The choice between the two depends on the nature of the data and the level of precision required for the analysis.

3.1.1.4 Clustering Methods

Clustering techniques can also be classified based on the way clusters are formed. Different methods use different strategies such as distance, density, or probability to group data points. Some of the methods are explained below.

1. Centroid-Based Clustering

Centroid-based clustering is a technique where clusters are formed by identifying central points, known as centroids, and assigning nearby data points to these centers based on similarity or distance. Each cluster is represented by its centroid, which acts as the average position of all points in that group. Common algorithms used in this approach include k-Means, which repeatedly updates cluster centers to minimize variation within

SGOU

clusters, and k-Medoids, which selects actual data points as centers, making it more robust to noise and outliers.

Advantages

- Fast and efficient
- Works well with large datasets

Limitations

- Number of clusters must be decided in advance
- Sensitive to outliers
- Not suitable for irregular shapes

2. Density-Based Clustering

Density-based clustering groups data by identifying regions where points are closely packed together, forming clusters, while points that lie in sparse areas are considered noise or outliers. This approach is effective for discovering clusters of different shapes and handling irregular data distributions. Common algorithms used in this method include DBSCAN, which creates clusters based on density conditions, and OPTICS, which extends this idea to better manage datasets with varying densities.

Advantages

- Can find clusters of different shapes
- Works well with noisy data
- No need to specify cluster count initially

Limitations

- Parameter selection can be difficult
- Performance may vary with uneven density

3. Connectivity-Based Clustering (Hierarchical Clustering)

Connectivity-based (hierarchical) clustering forms groups in a step-by-step manner and represents their relationships using a tree-like structure, making it easy to see how clusters are organized at different levels. This method can follow two main approaches: Agglomerative clustering, which starts with individual data points and gradually merges them into larger clusters (bottom-up), and divisive clustering, which begins with a single cluster and splits it into smaller groups (top-down).

Advantages

- Easy to visualize using dendrograms
- Does not require predefined cluster count

SGOU

Limitations

- Time-consuming for large datasets
- Cannot undo merging or splitting decisions

4. Distribution-Based Clustering

Distribution-based clustering is a method that assumes the data is generated from underlying statistical distributions and groups data points according to their probability of belonging to each distribution. Instead of assigning points strictly, it estimates how likely each point fits into a cluster. A commonly used algorithm in this approach is Gaussian Mixture Model (GMM), which models the data as a combination of multiple Gaussian distributions and assigns membership based on likelihood.

Advantages

- Handles complex cluster shapes
- Provides probability-based grouping

Limitations

- Requires number of clusters in advance
- Computationally intensive

5. Fuzzy Clustering

Fuzzy clustering is a type of clustering where each data point can belong to multiple clusters with different degrees of membership instead of being assigned to just one group. It is useful when cluster boundaries are not clearly defined, allowing more flexible and realistic grouping of data. A commonly used algorithm for this method is Fuzzy C-Means, which assigns membership values to data points and updates them iteratively to improve cluster formation.

Advantages

- Suitable for uncertain and overlapping data
- Provides more realistic grouping

Limitations

- Requires careful parameter tuning
- Slightly slower than simple methods

Clustering methods provide different ways to organize and understand data by grouping similar items based on various principles such as distance, density, connectivity, and probability. Each method has its own strengths and is suitable for specific types of data and applications. While some techniques work best for clearly separated groups, others are more effective in handling complex, overlapping, or noisy data. Therefore, selecting

SGOU

an appropriate clustering method and evaluating its results are important steps to ensure meaningful and accurate analysis in real-world scenarios.

3.1.1.5 Applications of Clustering

Clustering is widely used in many fields to organize large amounts of data into meaningful groups. By identifying similarities among data points, it helps in discovering hidden patterns and improving decision-making. Its ability to work with unlabeled data makes it especially useful in real-world situations where predefined categories are not available. Some applications are:

1. **Customer segmentation:** Businesses group customers based on purchasing behavior, preferences, or demographics. This helps companies design targeted marketing strategies and improve customer satisfaction.

2. **Anomaly detection:** Clustering helps identify unusual patterns or outliers in data. It is commonly used in fraud detection, network security, and fault detection systems.

3. **Image segmentation:** In computer vision, clustering is used to divide an image into meaningful regions. This helps in object recognition, medical imaging, and video analysis.

4. **Recommendation systems:** Clustering groups users or products based on similarities. This is used in platforms like e-commerce and streaming services to provide personalized recommendations.

5. **Document classification:** Large collections of documents can be grouped based on similar topics or keywords. This makes information retrieval and organization easier.

6. **Market basket analysis:** Retailers use clustering to find products that are often purchased together. This helps in product placement and cross-selling strategies.

7. **Healthcare and medical analysis:** Patients can be grouped based on symptoms, medical history, or treatment responses. This supports better diagnosis and personalized treatment plans.

8. **Social network analysis:** Clustering helps identify communities or groups of users with similar interests or interactions in social networks, improving targeted communication and analysis.

Clustering plays a significant role in extracting useful insights from data across various domains. By grouping similar data points, it simplifies complex datasets and supports better analysis and decision-making. Its wide range of applications highlights its importance as a powerful tool in data analysis and machine learning.

SGOU

3.1.1.6 Selection Criteria for Clustering Algorithms

Choosing the right clustering algorithm is important to obtain meaningful and accurate results from data. Since different algorithms work based on different principles, the selection depends on the nature of the dataset, the problem requirements, and the expected output. A proper choice ensures better grouping, improved interpretation, and effective decision-making. Key selection criteria are:

1. **Nature of data:** The type of data (numerical, categorical, or mixed) plays a major role in selecting an algorithm. For example, k-Means works well with numerical data, while other methods may be needed for categorical data.

2. **Shape of clusters:** If clusters are spherical and well-separated, centroid-based methods are suitable. For irregular or complex shapes, density-based methods like DBSCAN are more effective.

3. **Size of dataset:** For large datasets, faster algorithms like k-Means are preferred. Some methods like hierarchical clustering may not perform efficiently with very large data.

4. **Presence of noise and outliers:** If the dataset contains noise or outliers, density-based methods are better as they can identify and ignore such points.

5. **Number of clusters:** Some algorithms require the number of clusters to be specified in advance (e.g., k-Means), while others can determine clusters automatically (e.g., DBSCAN).

6. **Scalability:** The algorithm should be able to handle increasing data size efficiently without significant performance loss.

7. **Interpretability:** The results should be easy to understand and explain. Simple methods are preferred when clarity is important.

8. **Computational Complexity:** Time and resource requirements should be considered, especially for large-scale applications.

Selecting an appropriate clustering algorithm depends on multiple factors such as data type, cluster shape, dataset size, and computational requirements. There is no single best method for all situations, so understanding these criteria helps in choosing the most suitable technique for accurate and meaningful data analysis.

3.1.2 k-Means Clustering

k-Means clustering is a popular unsupervised machine learning technique used to group similar data points into clusters without requiring labeled data. It helps in discovering hidden patterns and identifying natural groupings within a dataset, especially when the data is unorganized. This method is widely applied in areas such as customer segmentation, image compression, and pattern recognition to bring structure to raw data.

SGOU

The algorithm works by dividing the dataset into a predefined number of clusters (k), where each cluster is represented by a central point called a centroid. Data points are assigned to clusters based on their distance from these centroids, usually using distance measures like Euclidean distance. The centroids are updated repeatedly by calculating the average of points in each cluster, and the process continues until the clusters become stable.

For example, in a retail business, customers can be grouped based on their purchasing behavior. If k = 3, the algorithm might create clusters such as frequent buyers, occasional buyers, and low-activity customers. Each cluster represents a group with similar characteristics, helping businesses understand customer patterns and make better marketing decisions.

3.1.2.1 Working Principle of k-Means

k-Means clustering works by dividing a dataset into a fixed number of clusters based on similarity among data points. Consider a dataset where each item is represented by features (values), often in the form of vectors. The aim is to group these items into clusters such that similar items are placed together.

The number of clusters is denoted by k, which must be specified before applying the algorithm. The method measures similarity using distance metrics, most commonly Euclidean distance, to determine how close a data point is to a cluster center (centroid). The algorithm follows an iterative approach where data points are assigned to the nearest centroid, and the centroids are updated repeatedly until the grouping stabilizes.

The main objective of k-Means is to minimize the distance between data points within the same cluster while maximizing the distance between different clusters. This ensures that each cluster is compact and well-separated from others.

3.1.2.2 Steps Involved in k-Means Algorithm

The k-Means algorithm follows a systematic and iterative process to group data points into meaningful clusters. It begins with an initial setup and repeatedly refines the grouping by assigning data points to clusters and updating their centers. Each step plays an important role in improving the accuracy of clustering and ensuring that similar data points are grouped together. Understanding these steps helps in clearly knowing how the algorithm works and how clusters are formed in a structured manner. The k-Means algorithm proceeds through the following steps:

1. Initialization: The process starts by selecting k initial centroids randomly from the dataset. These centroids act as the starting points for forming clusters (Fig. 3.1.2).

SGOU

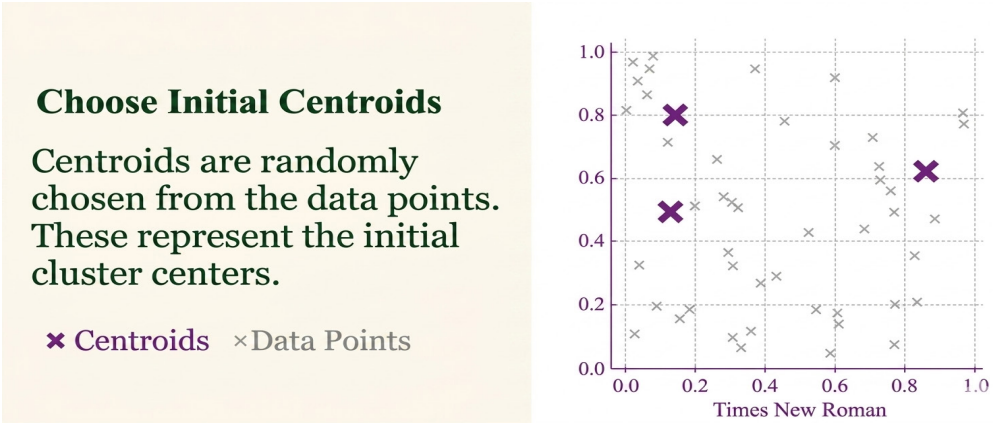


Fig. 3.1.2 Selecting k initial centroids

2. Assignment Step: Each data point is assigned to the nearest centroid based on a distance measure (commonly Euclidean distance). This step groups data points into k clusters (Fig. 3.1.3).

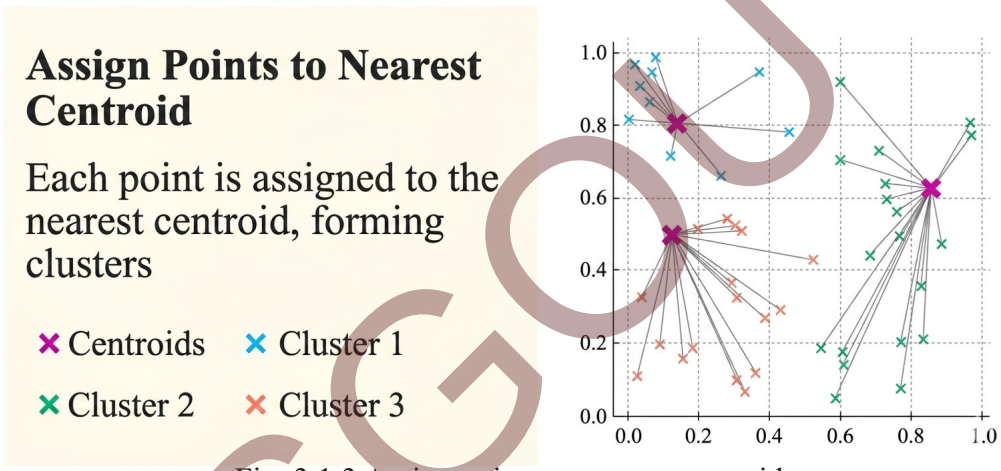


Fig. 3.1.3 Assign points to nearest centroid

3. Update Step: After assigning the points, the centroid of each cluster is recalculated by taking the average (mean) of all data points in that cluster. This updates the cluster centers (Fig. 3.1.4).

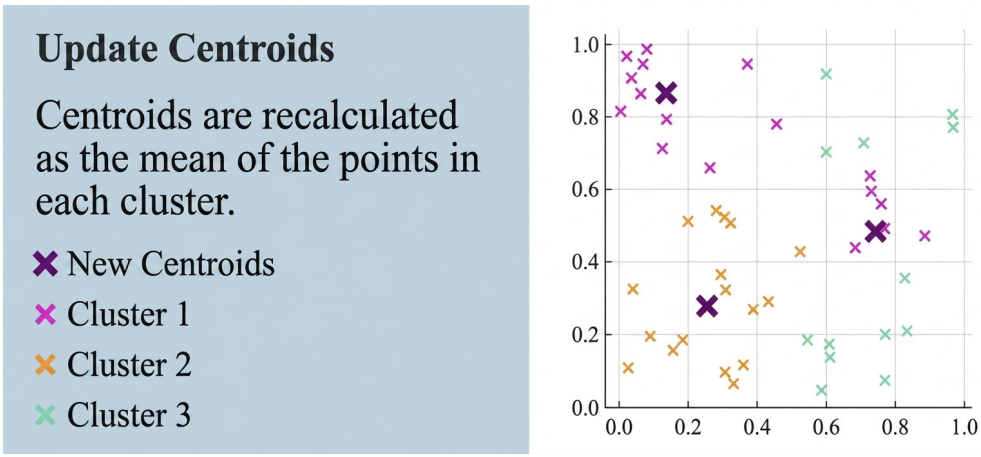


Fig.. 3.1.4 Update centroids

SGOU

4. Iteration (Repeat Process): The assignment and update steps are repeated multiple times (Fig. 3.1.5). With each iteration,

Data points may shift to different clusters

Centroids are recalculated again

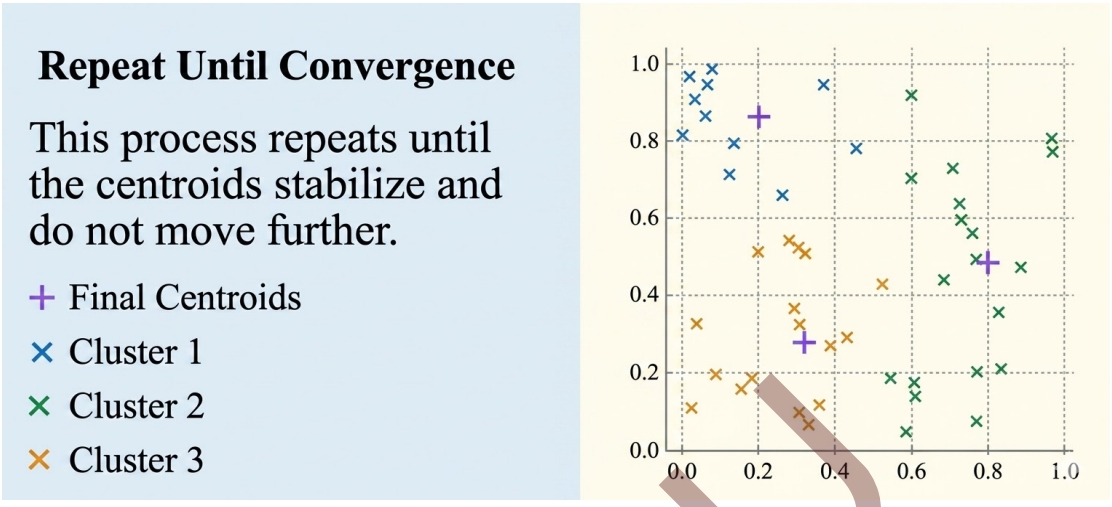


Fig. 3.1.5 Repeat until convergence

5. Convergence (Stopping Condition): The algorithm stops when,

The centroids do not change significantly, or

The cluster assignments remain the same, or

A maximum number of iterations is reached.

6. Final Clusters: At the end of the process, stable clusters are formed where data points within each cluster are more similar to each other than to those in other clusters.

3.1.2.3 Choosing the Value of k (Elbow Method)

Selecting the correct number of clusters (**k**) is a crucial step in k-Means clustering because it directly affects the quality and usefulness of the results. If the value of k is too small, important patterns in the data may be missed. On the other hand, if k is too large, the data may be divided into too many small and less meaningful groups. Therefore, a proper method is needed to determine the optimal value of k.

The Elbow Method is a commonly used technique to find the best value of k. In this method, k-Means clustering is applied for different values of k (for example, from 1 to 10). For each value of k, a metric called Within-Cluster Sum of Squares (WCSS) is calculated, which measures how closely the data points are grouped within a cluster. Lower WCSS values indicate tighter and better clusters.

The values of k are plotted on the x-axis and the corresponding WCSS values on the

SGOU

y-axis. As k increases, WCSS decreases because clusters become smaller and more compact. However, after a certain point, the rate of decrease slows down significantly, forming a bend or "elbow" shape in the graph. This point is considered the optimal value of k because it represents a balance between minimizing error and avoiding unnecessary clusters.

For example, if we are grouping customers and observe that the graph shows a clear bend at k = 3, then choosing three clusters would be appropriate. This ensures meaningful grouping without overcomplicating the model.

3.1.2.4 Advantages and Limitations of k-Means

k-Means is one of the most commonly used clustering algorithms due to its simplicity and efficiency. It is widely applied in various fields because it provides quick and understandable results when grouping data.

1. **Simple and easy to implement:** The algorithm is straightforward and easy to understand, making it suitable for beginners and practical applications.

2. **Fast and efficient:** k-Means works quickly even with large datasets, making it useful for real-time data analysis.

3. **Scalable to large data:** It can handle large volumes of data efficiently without significant performance issues.

4. **Easy to interpret results:** The clusters formed are clear and easy to analyze, which helps in decision-making.

5. **Works well with separated data:** It performs effectively when clusters are distinct and clearly separated.

6. **Widely applicable:** Used in many areas such as customer segmentation, image processing, and pattern recognition.

Overall, k-Means is a powerful and practical clustering method due to its simplicity, speed, and effectiveness, especially when the data is well-structured and clearly separable.

Despite its advantages, k-Means has certain limitations that can affect its performance in complex datasets. Understanding these limitations helps in choosing the right method for specific problems.

1. **Requires predefined value of k:** The number of clusters must be specified in advance, which may not always be known.

2. **Sensitive to initial centroids:** Different initial choices of centroids can lead to different clustering results.

3. **Affected by outliers and noise:** Outliers can significantly influence the position of centroids and distort clusters.

SGOU

4. **Not suitable for non-spherical clusters:** It assumes clusters are circular or spherical, which is not always true in real data.

5. **May converge to local optimum:** The algorithm may not always find the best possible clustering solution.

6. **Works only with numerical data:** k-Means is mainly suitable for numerical data and not ideal for categorical data.

In conclusion, while k-Means is efficient and easy to use, its limitations make it less suitable for complex, noisy, or irregular datasets. Careful consideration is required before applying it to ensure accurate results.

3.1.2.5 Applications of k-Means

k-Means clustering is widely used in various fields to group data into meaningful clusters based on similarity. Its simplicity and efficiency make it suitable for handling large datasets and discovering hidden patterns in unstructured data.

1. **Customer segmentation:** Businesses use k-Means to group customers based on purchasing behavior, preferences, or demographics. This helps in targeted marketing and personalized services.

2. **Image compression:** k-Means reduces the number of colors in an image by grouping similar colors together, thereby reducing image size without major loss of quality.

3. **Document clustering:** It is used to organize large collections of documents into groups based on similar topics or keywords, making information retrieval easier.

4. **Recommendation systems:** k-Means helps group users or products with similar characteristics, which is useful in providing personalized recommendations in platforms like e-commerce or streaming services.

5. **Anomaly detection:** It can identify unusual data points that do not fit into any cluster properly, helping detect fraud or abnormal behavior.

6. **Market basket analysis:** Retailers use k-Means to group products that are often purchased together, helping in better product placement and sales strategies.

7. **Healthcare analysis:** Patients can be grouped based on symptoms or medical history, aiding in diagnosis and treatment planning.

8. **Social network analysis:** k-Means helps identify groups of users with similar interests or interactions in social media platforms.

k-Means clustering plays an important role in data analysis by simplifying complex datasets into meaningful groups. Its wide range of applications across different domains highlights its usefulness in solving real-world problems effectively.

SGOU

3.1.3 Hierarchical Clustering

Hierarchical clustering is a method of grouping data points into clusters based on their similarity by building a hierarchy of clusters. It is an unsupervised learning technique that organizes data into a tree-like structure, showing how clusters are related at different levels. Instead of dividing data into a fixed number of clusters at the beginning, it creates a nested grouping that can be viewed at various levels of detail.

This approach works by either gradually merging smaller clusters into larger ones or by splitting a large cluster into smaller groups. The relationships between data points are represented using a structure called a dendrogram, which helps in visualizing how clusters are formed step by step. It is useful when we want to understand the structure and relationships within the data rather than just forming groups.

For example, consider grouping animals based on their characteristics. At a lower level, animals may be grouped into cats and dogs, and at a higher level, both groups may be combined into a broader category like mammals. This step-by-step grouping shows how data points are related, making hierarchical clustering useful for analysis in areas like biology, document classification, and customer segmentation.

3.1.3.1 Types of Hierarchical Clustering

Hierarchical clustering methods are divided into two types based on how clusters are formed. One method builds clusters step by step by combining smaller groups, while the other divides a large group into smaller clusters.

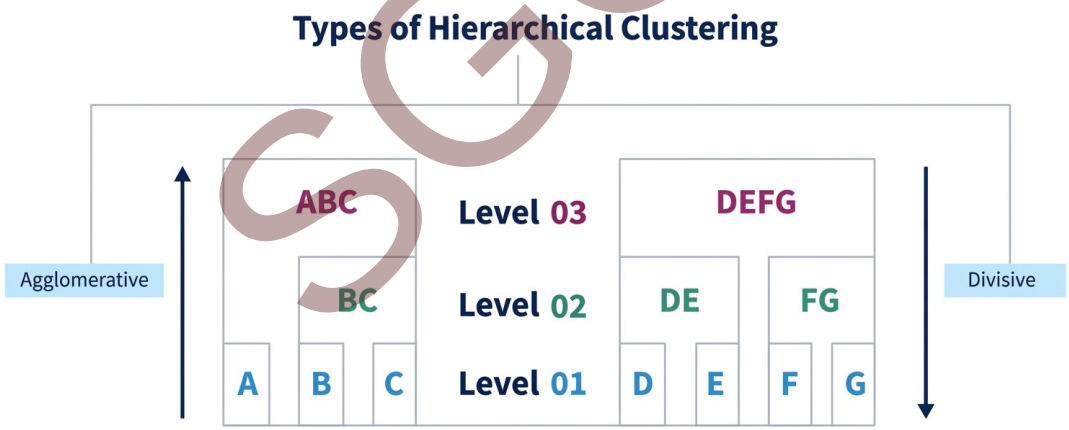


Fig. 3.1.6 Types of Hierarchical Clustering

A. Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is also called the bottom-up approach. In this method, each data point initially acts as an individual cluster. The algorithm then repeatedly merges the most similar clusters until all data points are combined into a single cluster containing the entire dataset.

SGOU

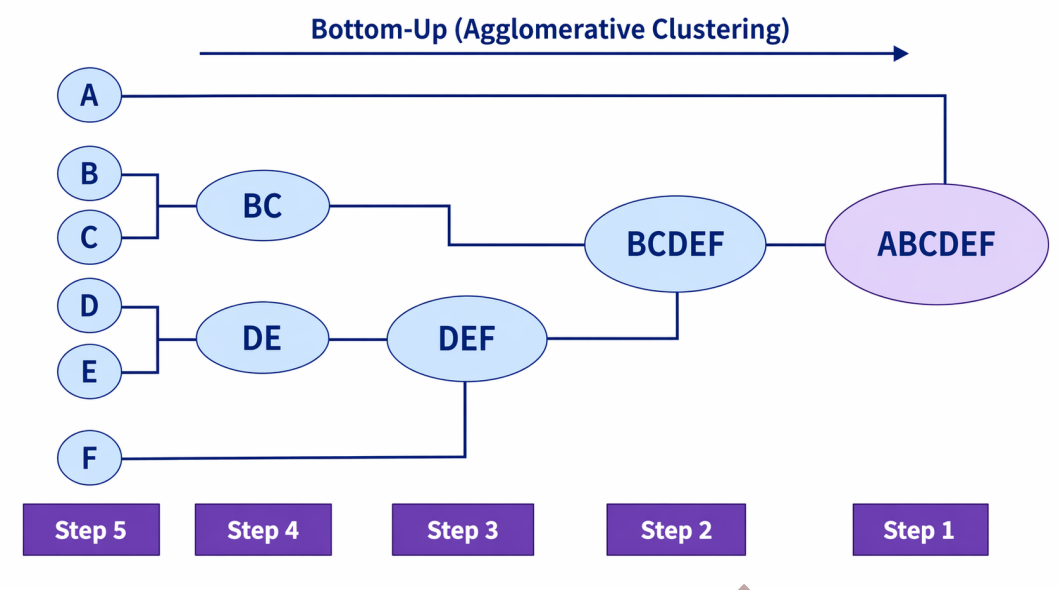


Fig. 3.1.7 Bottom-up approach

Workflow of Agglomerative Clustering

The workflow of agglomerative clustering describes the step-by-step process of gradually merging individual data points into larger clusters based on their similarity.

1. **Start with Individual Clusters:** Each data point is treated as a separate cluster. For example, if there are 5 data points, we begin with 5 clusters.

2. **Calculate Distances:** The distance between every pair of clusters is computed. Initially, this is simply the distance between individual data points.

3. **Merge Closest Clusters:** The two clusters with the smallest distance are combined into a single cluster.

4. **Update Distances:** After merging, the number of clusters reduces. Distances between the new cluster and the remaining clusters are recalculated.

5. **Repeat the Process:** Steps of merging and updating continue until only one cluster remains or the desired number of clusters is achieved.

6. **Dendrogram Representation:** The entire merging process is represented using a dendrogram, which shows how clusters are formed at different levels.

Consider a group of five cities based on their temperature values (Fig. 3.1.6). Initially, each city is treated as a separate cluster. The algorithm first finds the two cities with the most similar temperatures and groups them together. Next, it compares this new cluster with the remaining cities and again merges the closest ones. This process continues step by step, combining the most similar clusters at each stage, until all cities are grouped into a single cluster. The entire process can be visualized using a dendrogram, which shows how clusters are merged at different levels.

SGOU

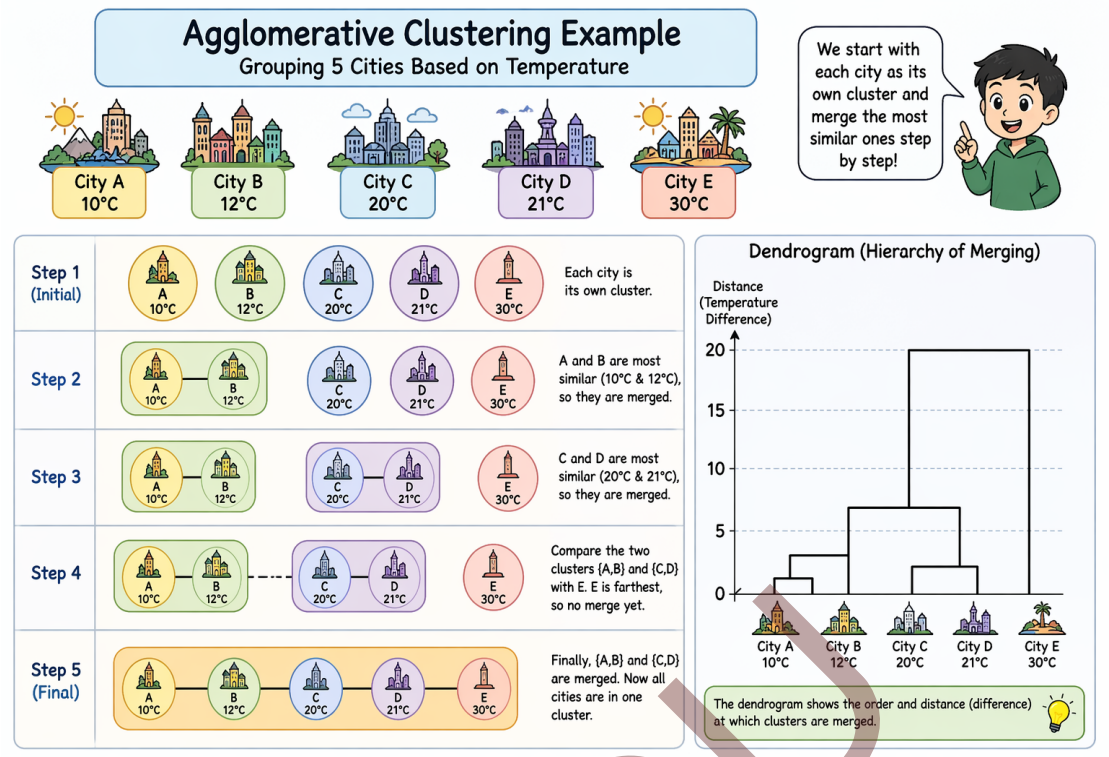


Fig. 3.1.8 Hierarchical Agglomerative Clustering

Agglomerative clustering is a bottom-up hierarchical approach where clusters are formed by merging smaller groups step by step. It is widely used for its simplicity and ability to represent data relationships. Some advantages are:

1. **Easy to understand and implement:** The step-by-step merging process is simple and intuitive.
2. **No need to predefine number of clusters:** Clusters can be chosen later by cutting the dendrogram at a desired level.
3. **Provides hierarchical structure:** Shows relationships between data points at different levels.
4. **Flexible with distance measures:** Can use different linkage criteria like single, complete, and average.

Agglomerative clustering is useful for understanding data structure, but its performance limitations make it less suitable for large or noisy datasets. Some limitations are:

1. **High computational cost:** Not suitable for very large datasets due to time complexity.
2. **Irreversible merging:** Once clusters are merged, the decision cannot be undone.
3. **Sensitive to noise and outliers:** Outliers can affect cluster formation.
4. **Memory intensive:** Requires storing distance matrix, which consumes more memory.

SGOU

Agglomerative clustering is an effective bottom-up approach that builds clusters gradually by merging similar data points. It provides a clear hierarchical structure of data, making it useful for understanding relationships and patterns. Although it can be computationally expensive for large datasets, it is widely used for its simplicity and ability to represent data in a meaningful and interpretable way.

B. Hierarchical Divisive Clustering

Divisive clustering is known as the top-down approach. It begins with all data points grouped into a single cluster and then divides this cluster into smaller clusters step by step until each data point becomes its own cluster or a stopping condition is reached.

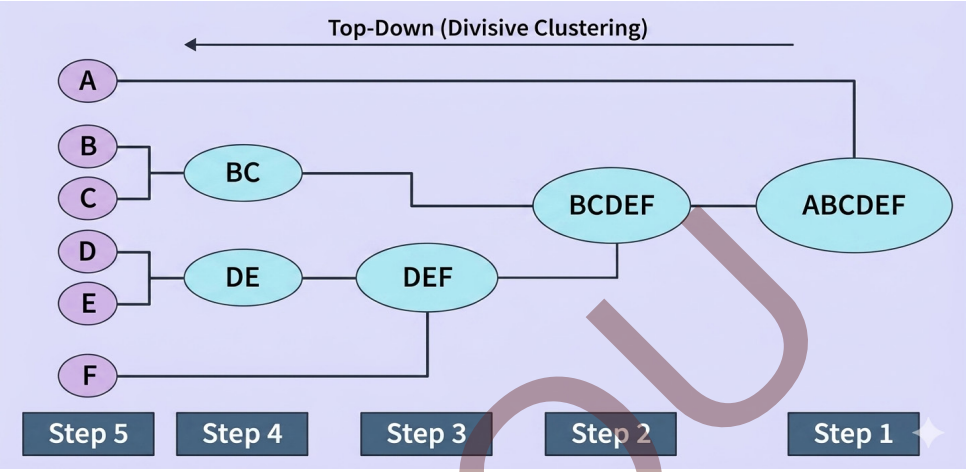


Fig. 3.1.9 Top-down approach

Workflow of Divisive Clustering

The workflow of divisive clustering describes the step-by-step process of splitting a single large cluster into smaller clusters based on differences among data points.

1. **Start with one large cluster:** All data points are initially placed in a single cluster.
2. **Split the cluster:** The cluster is divided into two smaller clusters. This is usually done by identifying the most dissimilar data points and separating them.
3. **Repeat splitting:** The process continues by selecting clusters with high dissimilarity and dividing them further.
4. **Recursive division:** Each newly formed cluster is split again into smaller clusters.
5. **Stopping condition:** The process stops when each data point forms its own cluster or when a predefined number of clusters is reached.

Consider a dataset of employees based on their performance scores (Fig. 3.1.10). Initially, all employees are placed in one large cluster. The algorithm then divides this group into two smaller clusters, such as high performers and low performers, based on differences in scores. Next, each of these clusters is further split into more specific groups, such as very high, average, and below-average performers. This process continues step by step until smaller and more meaningful clusters are formed or until a desired number of clusters is reached.

SGOU

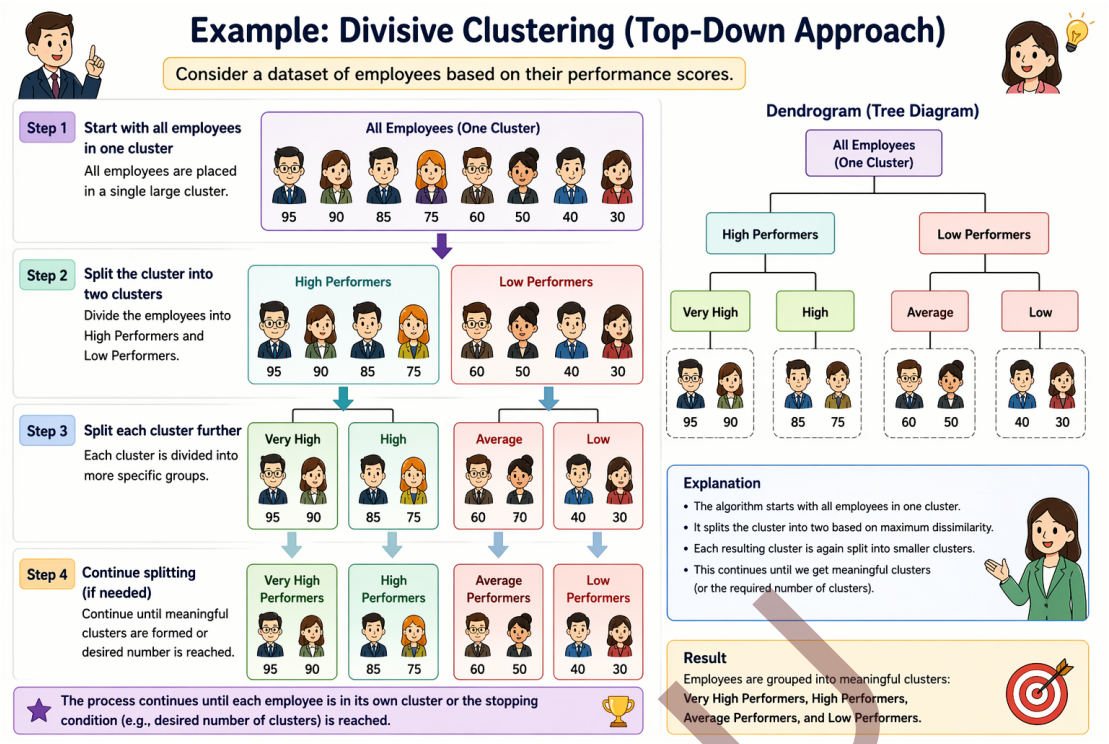


Fig. 3.1.10 Hierarchical Divisive Clustering

Divisive clustering is a top-down hierarchical method that starts with one cluster and splits it into smaller clusters. It helps in identifying differences within data from a broader perspective. Some advantages are:

1. **Captures global structure first:** Begins with the overall dataset, giving a clear top-level view.
2. **Produces meaningful clusters:** Splitting can lead to well-separated groups.
3. **Less affected by early local decisions:** Initial grouping considers the entire dataset.
4. **Useful for smaller datasets:** Effective when data size is manageable.

Divisive clustering provides a top-down view of data structure, but its complexity and computational cost limit its use in large-scale applications. Some limitations are:

1. **Computationally expensive:** Requires complex calculations for splitting clusters.
2. **Difficult to implement:** More complex compared to agglomerative clustering.
3. **Requires splitting strategy:** Choosing how to divide clusters is not straightforward.
4. **Less commonly used:** Limited practical implementations and tools.

Divisive clustering is a top-down approach that starts with a single cluster and gradually breaks it into smaller groups. It helps in clearly identifying differences within data and forming well-defined clusters. Although it is less commonly used due to higher

SGOU

computational complexity, it is useful when the overall structure of the data needs to be explored from a broader perspective.

3.1.3.2 Linkage Criteria (Single, Complete, Average)

In hierarchical clustering, linkage criteria are used to decide how the distance between clusters is calculated. When clusters are merged (in agglomerative clustering) or split (in divisive clustering), the algorithm needs a rule to measure similarity between groups of data points. Different linkage methods produce different cluster shapes and results.

1. Single Linkage (Nearest Neighbor Method)

Single linkage is a method used in hierarchical clustering to measure the distance between two clusters by considering the **minimum distance** between any pair of data points, where one point belongs to each cluster. It focuses on the closest points when deciding which clusters to merge. Some key points are:

- Uses the shortest distance between clusters
- Merges clusters with the nearest data points
- Also called the nearest neighbor method

Example

If Cluster A has points {2, 6} and Cluster B has points {9, 12}, the distance is calculated using the closest pair (6 and 9).

Advantages

- Can detect clusters with irregular shapes
- Simple to understand and implement

Limitations

- Sensitive to noise and outliers
- May form long, chain-like clusters (chaining effect)

2. Complete Linkage (Farthest Neighbor Method)

Complete linkage is a method in hierarchical clustering where the distance between two clusters is defined as the **maximum distance** between any pair of data points, with one point from each cluster. It considers the farthest points when deciding which clusters to merge, leading to more compact groups. Some key points are:

- Uses the largest distance between clusters
- Merges clusters based on farthest data points
- Also called the *farthest neighbor method*

SGOU

Example

If Cluster A has points {2, 6} and Cluster B has points {9, 12}, the distance is calculated using the farthest pair (2 and 12).

Advantages

- Produces compact and well-separated clusters
- Reduces chaining effect compared to single linkage

Limitations

- Sensitive to outliers
- May break larger natural clusters into smaller ones

3. Average Linkage (Group Average Method)

Average linkage is a hierarchical clustering method where the distance between two clusters is calculated as the **average of all pairwise distances** between data points in the two clusters. Instead of focusing only on the nearest or farthest points, it considers all points, providing a balanced measure of similarity. Some key points are:

- Uses the average distance between all pairs of points
- Provides a balance between single and complete linkage
- Also called the *group average method*

Example

If Cluster A has points {2, 6} and Cluster B has points {9, 12}, distances are calculated between all pairs (2–9, 2–12, 6–9, 6–12), and their average is taken as the cluster distance.

Advantages

- Less sensitive to noise compared to single linkage
- Produces more balanced clusters

Limitations

- Computationally more expensive
- May not perform well for very complex cluster shapes

Linkage criteria play a crucial role in hierarchical clustering by determining how clusters are formed and merged. Single linkage focuses on the nearest points, complete linkage considers the farthest points, and average linkage provides a balanced approach. The choice of linkage method affects the final clustering result, so it should be selected based on the nature of the dataset and the desired outcome.

SGOU

3.1.3.3 Dendrogram Representation

A dendrogram is a graphical representation used in hierarchical clustering to show how data points are grouped together step by step. It looks similar to a tree diagram and helps in understanding the structure and relationships between clusters.

A dendrogram can be compared to a family tree for clusters. At the bottom, each data point starts as its own group, and as we move upward, similar data points or clusters are gradually merged. This visual representation makes it easy to track how clusters are formed at different stages.

- At the bottom level, individual data points such as P, Q, R, S, and T are all separate clusters.
- As we move upward, the most similar data points are merged first to form small clusters.
- These clusters are then combined with other nearby clusters step by step.
- The lines connecting points or clusters show the order in which merging happens.
- The height of the connection (link) indicates the distance or dissimilarity between clusters.
- A shorter vertical line means the clusters are very similar, while a longer line indicates less similarity.

Key Features

- Some key features of dentrogram are:
- Visualizes hierarchical relationships between clusters.
- Helps in understanding how clusters are formed progressively.
- Useful for deciding the number of clusters by cutting the tree at a certain level.

Example

Consider six data points: P, Q, R, S, T and U. Initially, each point is separate. Suppose P and Q, T and U are very similar, so they merge first at a lower level. Then S joins the T,U group, then R follows at higher levels. The dendrogram clearly shows this merging process and the similarity between points.

SGOU

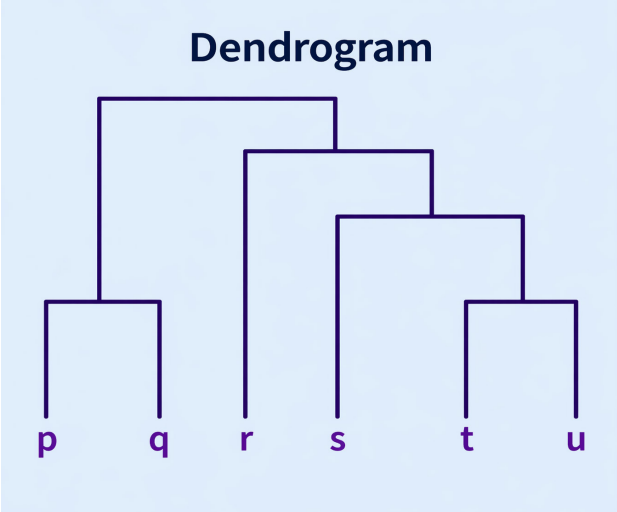


Fig. 3.1.11 Dendrogram

Dendrogram representation is a powerful tool in hierarchical clustering that visually explains how clusters are formed and related. It not only shows the merging process but also helps in selecting the appropriate number of clusters by analyzing the structure of the tree.

3.1.3.4 Advantages and Limitations of Hierarchical Clustering

Hierarchical clustering offers several important advantages that make it a powerful and widely used technique for analyzing and understanding the structure of data. Some important advantages are:

1. **No need to predefine number of clusters:** Unlike K-means, hierarchical clustering does not require specifying the number of clusters beforehand. The dendrogram helps decide the optimal number of clusters by visually cutting the tree at a desired level.

2. **Easy to understand and interpret:** The dendrogram provides a clear visual representation of how clusters are formed step by step, making the results intuitive and easy to interpret.

3. **Works well with small datasets:** It is effective for small to medium-sized datasets where detailed cluster relationships are important.

4. **Captures hierarchical relationships:** It shows nested group structures, meaning we can see both small sub-clusters and larger clusters within the same model.

5. **Flexible distance metrics:** Different distance measures (Euclidean, Manhattan, cosine, etc.) and linkage methods (single, complete, average) can be used depending on the problem.

6. **Deterministic results:** Unlike K-means, hierarchical clustering does not depend on random initialization, so it produces consistent results for the same dataset.

SGOU

Hierarchical clustering, despite its advantages, has several limitations that can affect its efficiency, scalability, and accuracy in practical applications. Some key points are:

1. **High computational complexity:** It requires more time and memory, especially for large datasets, making it inefficient for big data applications.

2. **Not suitable for large datasets:** Because of its $O(n^2)$ or higher complexity, it becomes very slow as the dataset size increases.

3. **No possibility of backtracking:** Once clusters are merged or split, the algorithm cannot undo the decision, which may lead to suboptimal clusters.

4. **Sensitive to noise and outliers:** Outliers can significantly affect the clustering structure and may distort the dendrogram.

5. **Difficulty in choosing cutoff level:** Deciding where to "cut" the dendrogram to form final clusters can be subjective and may lead to different interpretations.

6. **Less efficient for dynamic data:** It is not suitable for datasets that change frequently, as the entire clustering process must be recomputed.

3.1.3.5 Applications of Hierarchical Clustering

Hierarchical clustering is widely used in different fields to group similar data and understand relationships among data points. Its ability to form a tree-like structure (dendrogram) makes it especially useful for analyzing complex datasets.

1. **Bioinformatics and Genetics:** Hierarchical clustering is used to classify genes and proteins based on similarities in their expression patterns. It helps researchers understand gene functions, identify disease patterns, and study evolutionary relationships.

2. **Market Segmentation:** Businesses use hierarchical clustering to group customers based on purchasing behavior, preferences, or demographics. This helps in targeted marketing, personalized recommendations, and improving customer satisfaction.

3. **Document and Text Clustering:** It is used in organizing large collections of documents by grouping similar texts together. Applications include search engines, topic identification, and automatic document classification.

4. **Image Segmentation:** Hierarchical clustering helps in dividing images into meaningful regions based on pixel similarity. It is used in medical imaging, object detection, and pattern recognition.

5. **Social Network Analysis:** It is applied to identify communities or groups within social networks. This helps in understanding relationships, influence patterns, and information flow among users.

6. **Anomaly Detection:** Hierarchical clustering can be used to detect outliers or unusual data points that do not fit into any cluster. This is useful in fraud detection, network security, and fault diagnosis.

SGOU

Hierarchical clustering is a versatile technique with applications across various domains such as biology, business, and computer science. Its ability to reveal hidden patterns and relationships makes it valuable for data analysis, although its use is often limited to smaller datasets due to computational complexity.

3.1.4 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a powerful clustering algorithm used in data mining and machine learning to discover patterns in datasets. It is a density-based clustering algorithm that groups data points that are closely packed together and marks outliers as noise based on their density in the feature space. Instead of requiring a predefined number of clusters, DBSCAN automatically identifies clusters as dense regions in the data space separated by areas of lower density. This makes it especially useful when dealing with complex datasets where the structure is not clearly defined.

Unlike traditional methods such as K-Means or hierarchical clustering, which assume clusters are compact and spherical, DBSCAN performs well in handling real-world data irregularities. For example, consider a dataset representing locations of restaurants in a city: DBSCAN can group densely located restaurants into clusters (busy commercial areas) while identifying isolated restaurants as noise. It can also detect clusters of arbitrary shapes, such as curved or uneven patterns, making it more flexible and practical for real-world applications like geographic data analysis, fraud detection, and image processing.

3.1.4.1 Key Concepts (Epsilon, MinPts, Core Points, Border Points, Noise)

DBSCAN is highly effective in identifying arbitrary-shaped clusters, meaning the clusters it forms are not limited to simple circular or convex shapes. This makes it suitable for real-world datasets where patterns may be irregular or complex. Another key strength of DBSCAN is its ability to handle noise and outliers. Unlike many clustering algorithms, it does not force every data point into a cluster; instead, it clearly identifies and separates noise points that do not belong to any meaningful group. Some key concepts of DBSCAN are:

1. **Epsilon (ε):** Epsilon defines the radius or maximum distance within which neighboring data points are considered part of the same cluster. It determines how close points should be to each other to be grouped together. A smaller ε may result in many small clusters or noise, while a larger ε may merge distinct clusters into one.

2. **MinPts (Minimum Points):** MinPts specifies the minimum number of data points required within the ε-radius to form a dense region. It helps in determining whether a group of points is dense enough to be considered a cluster. Typically, higher values of MinPts lead to more robust clusters but may also increase the number of points classified as noise.A general rule of thumb is to set **MinPts ≥ D + 1**, where

SGOU

D is the number of dimensions in the dataset. For most practical cases, a minimum value of **MinPts = 3** is recommended, although higher values may be used for more complex datasets to improve clustering quality.

3. **Core Points:** A core point is a data point that has at least MinPts number of points (including itself) within its ε-neighborhood. Core points form the backbone of clusters and are located in dense regions of the dataset.

4. **Border Points:** Border points are data points that lie within the ε-neighborhood of a core point but do not have enough neighboring points to be considered core points themselves. They are part of a cluster but are located at its edges.

5. **Noise (Outliers):** Noise points are those data points that do not belong to any cluster. They are not within the ε-neighborhood of any core point and are considered outliers. DBSCAN effectively identifies and separates these points from meaningful clusters.

3.1.4.2 Working of DBSCAN Algorithm

The working of DBSCAN algorithm is based on grouping data points according to their density and connectivity. It identifies clusters by expanding from core points and connecting nearby points within a specified radius, while separating sparse points as noise. DBSCAN works by categorizing data points into three types:

- **Core points**: Points that have a sufficient number of neighbors within a specified radius (ε).
- **Border points**: Points that are close to core points but do not have enough neighbors to be core points themselves.
- **Noise points**: Points that do not belong to any cluster and are considered outliers.

By iteratively expanding clusters from core points and connecting density-reachable points, DBSCAN forms clusters without relying on rigid assumptions about their shape or size. This makes it highly effective for discovering complex and irregular patterns in data.

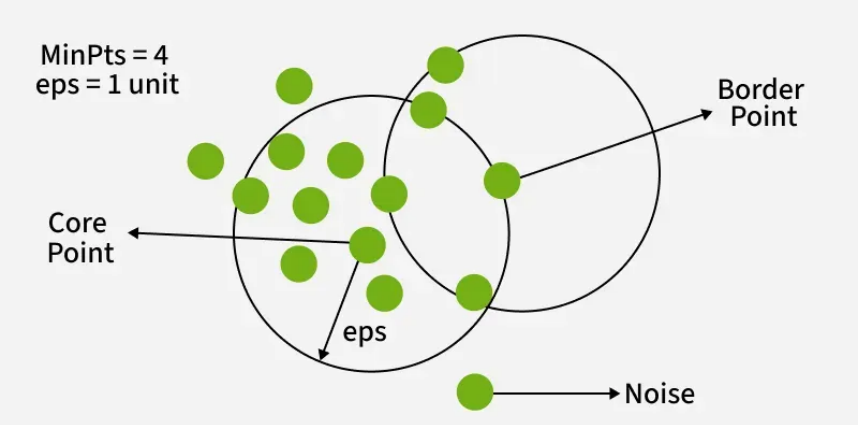


Fig. 3.1.12: DBSCAN Algorithm

SGOU

Steps in the DBSCAN Algorithm

1. **Identify Core Points:** For each point in the dataset, count the number of points within its ε-neighborhood. If this count is greater than or equal to MinPts, mark the point as a core point.

2. **Form Clusters:** For each core point that is not yet assigned to any cluster, create a new cluster. Then, recursively find all points that are density-reachable (i.e., within the ε radius of the core point) and add them to the cluster.

3. **Density Connectivity:** Two points *a* and *b* are said to be density-connected if there exists a chain of points between them such that each point lies within the ε radius of the next, and at least one point in the chain is a core point. This chaining process ensures that all points in a cluster are connected through dense regions.

4. **Label Noise Points:** After processing all data points, any point that does not belong to any cluster is labeled as noise. These points are treated as outliers and excluded from cluster formations.

Thus, the DBSCAN algorithm effectively forms clusters by exploring dense regions, connecting density-reachable points, and distinguishing noise, making it suitable for analyzing complex and irregular datasets.

3.1.4.3 Advantages of DBSCAN

1. **No Need to Specify Number of Clusters:** Unlike algorithms such as K-Means, DBSCAN does not require the number of clusters to be defined in advance. It automatically determines clusters based on data density.

2. **Detects Arbitrary-Shaped Clusters:** DBSCAN can identify clusters of any shape (e.g., circular, elongated, or irregular), making it suitable for real-world datasets.

3. **Effective Handling of Noise and Outliers:** It clearly identifies and separates noise points instead of forcing them into clusters, improving clustering accuracy.

4. **Works Well with Spatial Data:** DBSCAN is highly effective for geographical and spatial datasets where clusters are formed based on proximity.

5. **Robust to Outliers:** Since noise points are excluded from clusters, the algorithm is less affected by extreme values.

6. **No Assumption of Cluster Shape:** Unlike some clustering methods, DBSCAN does not assume clusters to be spherical or evenly sized.

7. **Ability to Discover Meaningful Patterns:** It can reveal hidden structures in complex datasets where traditional methods may fail.

8. **Flexible Cluster Formation:** Clusters are formed based on density connectivity, allowing natural grouping of data points.

SGOU

Overall, DBSCAN is a powerful clustering algorithm that provides flexibility, handles noise effectively, and works well with complex and irregular datasets, making it widely used in practical applications.

3.1.4.4 Limitations of DBSCAN

1. **Sensitive to Parameter Selection (ε and MinPts):** Choosing appropriate values for ε (epsilon) and MinPts is difficult. Incorrect values can lead to poor clustering results, such as too many noise points or merged clusters.

2. **Difficulty with Varying Density Clusters:** DBSCAN struggles when clusters have different densities. A single ε value may not work well for all clusters, causing some clusters to be missed or incorrectly merged.

3. **Not Suitable for High-Dimensional Data:** In high-dimensional spaces, distance measures become less meaningful (curse of dimensionality), making it harder for DBSCAN to identify dense regions accurately.

4. **Performance Issues with Large Datasets:** Although efficient with indexing techniques, DBSCAN can become computationally expensive for very large datasets, especially without optimization.

5. **Difficulty in Choosing ε Automatically:** Methods like k-distance graphs help, but selecting the correct ε still requires experience and experimentation.

6. **Border Point Ambiguity:** A border point may be reachable from multiple clusters, but DBSCAN assigns it to only one, which may not always be correct.

7. **Sensitive to Distance Metric:** The results depend on the chosen distance measure (e.g., Euclidean distance). Different metrics can produce different clustering outcomes.

While DBSCAN is powerful for detecting clusters and handling noise, its effectiveness depends on proper parameter tuning and may be limited in handling complex datasets with varying densities or high dimensions.

3.1.4.5 Applications of DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is widely used to identify clusters of different shapes and detect noise or outliers in real-world datasets. Its ability to discover dense groups without requiring the number of clusters in advance makes it useful in various applications.

1. **Spatial Data Analysis (GIS):** DBSCAN is widely used in geographic information systems to identify clusters such as urban areas, earthquake zones, or regions with high population density.

2. **Image Processing:** It helps in image segmentation by grouping similar pixels together and separating noise, improving object detection and pattern recognition.

SGOU

3. **Anomaly and Fraud Detection:** DBSCAN is effective in detecting unusual patterns such as fraudulent transactions or network intrusions by identifying points that do not belong to any cluster.

4. **Market Analysis and Customer Segmentation:** Businesses use DBSCAN to group customers based on purchasing behavior, helping in targeted marketing and personalized recommendations.

5. **Social Network Analysis:** It can identify communities or groups of users with similar interests or interactions within social networks.

6. **Clustering in Astronomy:** DBSCAN is used to detect star clusters and galaxy formations by analyzing spatial data in astronomical datasets.

7. **Biological Data Analysis:** In bioinformatics, it is applied to cluster gene expression data and identify patterns in biological datasets.

8. **Traffic and Transportation Analysis:** DBSCAN helps in identifying traffic congestion zones and analyzing movement patterns of vehicles or people.

9. **Document Clustering and Text Mining:** It groups similar documents or topics together in large text datasets, useful in search engines and recommendation systems.

DBSCAN is widely applied across various domains due to its ability to detect clusters of arbitrary shapes and handle noise effectively, making it suitable for complex real-world data analysis.

3.1.5 Comparison of k-Means, Hierarchical and DBSCAN

K-Means, Hierarchical Clustering, and DBSCAN are three widely used clustering techniques in data mining and machine learning, each based on a different approach to grouping data. K-Means follows a centroid-based method, Hierarchical clustering builds a tree-like structure of clusters, and DBSCAN groups data based on density. Comparing these algorithms is important to understand their strengths, limitations, and suitability for different types of datasets, such as handling noise, cluster shapes, and scalability.

Table 3.1.1 Comparison of k-Means, Hierarchical and DBSCAN

Feature	K-Means Clustering	Hierarchical Clustering	DBSCAN
Basic Approach	Partition-based (centroid method)	Tree-based (dendrogram structure)	Density-based
Number of Clusters	Must be specified in advance (K value)	Not required initially (can be decided later)	Not required

SGOU

Cluster Shape	Works best with spherical (convex) clusters	Can handle various shapes (limited flexibility)	Can detect arbitrary-shaped clusters
Handling Noise/ Outliers	Poor (sensitive to outliers)	Limited handling of noise	Excellent (explicitly detects noise)
Scalability	Efficient for large datasets	Less efficient for large datasets	Moderate (can be optimized with indexing)
Parameter Requirement	Requires K (number of clusters)	Requires distance metric and linkage method	Requires ε (epsilon) and MinPts
Cluster Formation	Based on distance from centroids	Based on merging/ splitting clusters	Based on density and connectivity
Sensitivity	Sensitive to initial centroid selection	Sensitive to linkage criteria	Sensitive to ε and MinPts
Handling Varying Density	Poor	Moderate	Poor (struggles with varying density)
Interpretability	Easy to understand	Visual (dendrogram helps interpretation)	Slightly complex

Each clustering technique has its strengths and limitations. The choice depends on the nature of the dataset, presence of noise, cluster shape, and whether the number of clusters is known beforehand.

3.1.6 Cluster Evaluation Metrics

luster evaluation metrics are used to measure the quality and effectiveness of clustering results. Since clustering is an unsupervised learning technique (no predefined labels), these metrics help determine how well the data points are grouped. Evaluation can be done using **internal metrics** (based on the dataset itself), **external metrics** (using ground truth labels if available), and **relative metrics** (to compare different clustering results). These metrics ensure that clusters are meaningful, well-separated, and compact. Importance of cluster evaluation:

1. **Assess Quality of Clusters:** Helps determine whether the formed clusters are meaningful and correctly represent the structure of the data.
2. **Measure Compactness and Separation:** Ensures that data points within a cluster are similar (high cohesion) and different from points in other clusters (high separation).
3. **Compare Different Algorithms:** Allows comparison between clustering techniques like K-Means, Hierarchical, and DBSCAN to select the best method.

SGOU

4. **Parameter Tuning:** Helps in selecting optimal parameters (e.g., K in K-Means, ε and MinPts in DBSCAN) for better clustering performance.

5. **Validate Results Without Labels:** Since clustering is unsupervised, evaluation metrics provide a way to validate results even when true labels are not available.

6. **Improve Decision Making:** Reliable cluster evaluation leads to better insights and more accurate decision-making in applications like marketing, healthcare, and fraud detection.

Cluster evaluation is essential for ensuring that clustering results are accurate, reliable, and useful for real-world applications, helping in selecting the best model and improving overall analysis.

3.1.6.1 Internal Evaluation Metrics

Internal evaluation metrics assess the quality of clustering based only on the dataset itself, without using any external labels. These metrics focus on how well the clusters are formed by measuring cohesion (compactness within clusters) and separation (distance between clusters).

1. Silhouette Score: The Silhouette Score measures how similar a data point is to its own cluster compared to other clusters.

- Value ranges from -1 to +1
- +1 → Point is well matched to its cluster and far from others
- 0 → Point lies between clusters
- -1 → Point may be assigned to the wrong cluster

Interpretation: Higher silhouette score indicates better-defined and well-separated clusters.

2. Davies-Bouldin Index (DBI): The Davies-Bouldin Index evaluates the average similarity between clusters based on their compactness and separation.

- Value ranges from 0 to ∞
- Lower value indicates better clustering

Interpretation: Clusters with low intra-cluster distance (compact) and high inter-cluster distance (well-separated) result in a lower DBI.

3. Within-Cluster Sum of Squares (WCSS): WCSS measures the total variance within each cluster by calculating the sum of squared distances between data points and their respective cluster centroids.

WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2

- Lower WCSS indicates tighter (more compact) clusters
- Commonly used in the Elbow Method to determine the optimal number of clusters

SGOU

Interpretation: As the number of clusters increases, WCSS decreases, but the goal is to find a balance where adding more clusters does not significantly improve compactness.

Internal evaluation metrics like Silhouette Score, Davies-Bouldin Index, and WCSS are essential tools for assessing clustering performance, helping to ensure that clusters are both compact and well-separated without relying on external information.

3.1.6.2 External Evaluation Metrics

External evaluation metrics are used to assess the quality of clustering results by comparing them with ground truth labels (actual class labels). These metrics measure how well the clusters formed by an algorithm match the true grouping of the data.

1. Adjusted Rand Index (ARI): The Adjusted Rand Index measures the similarity between the predicted clustering and the true labels, while adjusting for chance grouping.

- Value ranges from -1 to 1
- 1 → Perfect agreement between clustering and true labels
- 0 → Random labeling (no agreement)
- Negative values → Worse than random clustering

Interpretation: A higher ARI indicates that the clustering result closely matches the actual classification of the data.

2. Normalized Mutual Information (NMI): Normalized Mutual Information measures the amount of shared information between the predicted clusters and the true labels.

- Value ranges from 0 to 1
- 1 → Perfect correlation between clusters and true labels
- 0 → No mutual information (independent assignments)

Interpretation: Higher NMI values indicate better clustering performance, as more information is shared between predicted and actual groupings.

External evaluation metrics like ARI and NMI provide a reliable way to validate clustering results when true labels are available, helping in comparing different algorithms and ensuring accurate grouping.

3.1.6.3 Comparison of Evaluation Metrics

Comparison of evaluation metrics helps in understanding the strengths, limitations, and suitability of different measures used to assess the quality of clustering results.

SGOU

Table 3.1.2 Comparison of Evaluation Metrics

Metric	Type	Range	Ideal Value	Key Idea	Advantages	Limitations
Silhouette Score	Internal	-1 to 1	Higher (close to 1)	Measures cohesion and separation	Easy to interpret; no labels needed	Less effective for complex or varying density clusters
Davies-Bouldin Index (DBI)	Internal	0 to ∞	Lower (close to 0)	Ratio of intra-cluster to inter-cluster distance	Simple computation; compares cluster similarity	Sensitive to noise and cluster shape
WCSS	Internal	0 to ∞	Lower	Measures compactness within clusters	Useful for Elbow Method; simple	Always decreases with more clusters; not sufficient alone
Adjusted Rand Index (ARI)	External	-1 to 1	Higher (close to 1)	Measures similarity with true labels (adjusted for chance)	Accurate comparison with ground truth	Requires labeled data
Normalized Mutual Information (NMI)	External	0 to 1	Higher (close to 1)	Measures shared information between clusters and labels	Works well for comparing clustering results	Requires true labels; may not capture structure fully

Different evaluation metrics serve different purposes, and no single metric is sufficient in all cases. Internal metrics help assess structure without labels, while external metrics validate results when true labels are available. Choosing the right metric depends on the dataset and the objective of clustering.

3.1.6.4 Challenges in Cluster Evaluation

Challenges in cluster evaluation arise due to the absence of true labels, variability in data structures, and limitations of evaluation metrics in accurately assessing clustering quality.

1. **Lack of Ground Truth Labels:** In most clustering problems, true class labels are not available, making it difficult to objectively evaluate the correctness of clusters.

SGOU

2. **Subjectivity in Results:** Different evaluation metrics may produce different results for the same clustering, leading to ambiguity in deciding the best clustering outcome.

3. **Choosing the Right Metric:** Selecting an appropriate evaluation metric depends on the dataset and objective, which can be confusing and may affect the final interpretation.

4. **Sensitivity to Parameters:** Evaluation results can vary significantly based on algorithm parameters (e.g., K in K-Means, ε and MinPts in DBSCAN), making consistent evaluation challenging.

5. **Handling Complex Cluster Shapes:** Some metrics may not perform well for clusters with irregular shapes or varying densities, leading to misleading results.

6. **Scalability Issues:** Computing evaluation metrics for very large datasets can be time-consuming and computationally expensive.

7. **Noise and Outliers:** Presence of noise can distort evaluation results, especially for metrics that assume clean and well-separated clusters.

8. **Curse of Dimensionality:** In high-dimensional data, distance measures become less meaningful, affecting both clustering and its evaluation.

Cluster evaluation is a complex task due to the absence of labels, varying data structures, and metric limitations, requiring careful selection of methods and interpretation of results.

Summerised Overview

Clustering is an important unsupervised learning technique used to group similar data points into meaningful clusters based on their characteristics. Among the widely used clustering methods, K-Means, Hierarchical Clustering, and DBSCAN differ in their approach and application. K-Means is a centroid-based method that partitions data into a predefined number of clusters, making it simple and efficient for large datasets. Hierarchical clustering builds a tree-like structure (dendrogram) that shows relationships between data points and does not require a predefined number of clusters. DBSCAN, a density-based method, groups data based on density, allowing it to detect clusters of arbitrary shapes and effectively identify noise or outliers. Each technique has its strengths and limitations. K-Means is fast but sensitive to outliers and assumes spherical clusters. Hierarchical clustering provides better visualization and flexibility but is computationally expensive. DBSCAN is powerful for real-world datasets with irregular patterns but requires careful parameter selection and struggles with varying densities.

SGOU

To evaluate the performance of these clustering methods, cluster evaluation metrics are used. Internal metrics such as Silhouette Score, Davies-Bouldin Index, and WCSS assess clustering quality based on compactness and separation without using labels. External metrics like Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) compare clustering results with true labels when available. However, cluster evaluation faces challenges such as lack of ground truth, sensitivity to parameters, and difficulty in handling complex data structures.Overall, understanding different clustering techniques and their evaluation methods is essential for selecting the most suitable approach and ensuring meaningful and accurate data analysis in real-world applications.

Assignments

1. Explain the working principles of K-Means, Hierarchical Clustering, and DBSCAN. Compare their approaches, advantages, and limitations with suitable examples.

2. Implement K-Means and DBSCAN algorithms using Python. Apply them to a sample dataset and compare the clustering results using visualizations.

3. Construct a dendrogram for a given dataset and explain how clusters are formed at different levels. Discuss the impact of different linkage methods (single, complete, average).

4. Evaluate clustering results using internal metrics (Silhouette Score, Davies-Bouldin Index, WCSS) and external metrics (ARI, NMI). Analyze which algorithm performs better for a given dataset.

5. Select a real-world problem (e.g., customer segmentation, fraud detection, or geographic clustering) and apply an appropriate clustering technique. Justify the choice of algorithm and evaluate the results using suitable metrics.

SGOU

Suggested Reading

1. Introduction to Data Mining (Clustering Concepts)
   https://www-users.cse.umn.edu/~kumar/dmbook/index.php

2. Scikit-learn Clustering Documentation
   https://scikit-learn.org/stable/modules/clustering.html

3. DBSCAN Explained (Towards Data Science)
   https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556

4. K-Means Clustering (GeeksforGeeks)
   https://www.geeksforgeeks.org/k-means-clustering-introduction/

5. Hierarchical Clustering (GeeksforGeeks)
   https://www.geeksforgeeks.org/hierarchical-clustering/

Reference

1. Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

2. Tan, P. N., Steinbach, M., & Kumar, V. (2019). *Introduction to data mining* (2nd ed.). Pearson.

3. Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer.

4. Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice Hall.

5. Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

2 UNIT Dimensionality Reduction

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the basic concept and need for dimensionality reduction
- describe the working principles of PCA and SVD
- explain the basic concepts and applications of t-SNE and LDA
- compare different dimensionality reduction techniques
- identify real-world applications of dimensionality reduction techniques

Background

Modern datasets often contain a large number of features, making them difficult to analyze, visualize, and process efficiently. High-dimensional data may contain redundant or irrelevant information, which can increase computational complexity and affect the performance of machine learning algorithms. Dimensionality reduction is used to reduce the number of features while preserving the important information present in the data.

Several techniques are available for dimensionality reduction, each with a different approach. Principal Component Analysis (PCA) identifies directions of maximum variance, while Singular Value Decomposition (SVD) decomposes a matrix into simpler components. t-Distributed Stochastic Neighbor Embedding (t-SNE) focuses on preserving local relationships and is mainly used for visualizing high-dimensional data. Linear Discriminant Analysis (LDA) uses class labels to find directions that provide better separation between classes.

Understanding these techniques helps learners select suitable methods for different data analysis and machine learning tasks. This unit introduces the basic concepts, working principles, advantages, limitations, and applications of PCA, SVD, t-SNE, and LDA, providing a foundation for handling high-dimensional datasets in real-world applications.

SGOU

Keywords

Dimensionality Reduction, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), t-SNE, Linear Discriminant Analysis (LDA)

Discussion

Dimensionality reduction is a technique used to reduce the number of features in a dataset while preserving important information. It helps simplify complex datasets and makes data analysis and machine learning more efficient. PCA and SVD are linear techniques commonly used for data transformation and feature reduction. t-SNE is mainly used to visualize high-dimensional data, while LDA focuses on separating different classes. This unit introduces these techniques, their basic principles, advantages, limitations, and practical applications.

3.2.1 Introduction to Dimensionality Reduction

Modern datasets often contain hundreds or even thousands of features, making them difficult to analyze and visualize. Dimensionality Reduction provides a way to simplify such complex data by reducing the number of features while preserving the most important information. It helps machine learning models work faster, reduces unnecessary information, and makes hidden patterns easier to discover. Techniques such as PCA, SVD, t-SNE, and LDA offer different ways to transform high-dimensional data into a simpler form. Through this topic, complex data can be simplified, visualized, and made more useful for real-world decision-making.

3.2.1.1 Definition and Concept

Dimensionality Reduction is the process of reducing the number of features or dimensions in a dataset while preserving the most important and relevant information. High-dimensional datasets may contain many features, some of which may be redundant, irrelevant, or strongly correlated, making data analysis more complex and computationally expensive. Dimensionality reduction simplifies such datasets by representing them using a smaller number of meaningful variables.

It can be achieved in two main ways:

- Removing redundant or irrelevant features from the original dataset.
- Creating new features by combining or transforming the original features.

The main goal of dimensionality reduction is to obtain a compact and meaningful representation of the data with minimum loss of important information. This makes the data easier to analyze, visualize, store, and process using machine learning algorithms.

SGOU

3.2.1.2 Importance in Data Analysis and Machine Learning

Dimensionality reduction is important in Data Analysis and Machine Learning because it simplifies datasets by reducing the number of features while retaining important information. It helps make data easier to process, understand, visualize, and use for building machine learning models.

1. **Improved Model Performance:** Removing unnecessary or redundant features can reduce overfitting and help models focus on important patterns.

2. **Faster Computation:** Fewer features require less processing and memory, which can make training and prediction faster.

3. **Better Visualization:** High-dimensional data can be reduced to 2D or 3D, making patterns, relationships, and clusters easier to visualize.

4. **Noise Reduction:** Irrelevant features and unnecessary variations can be removed, helping the model focus on meaningful information.

5. **Reduced Redundancy:** Highly correlated or duplicate information can be combined into fewer dimensions, making the dataset more compact.

6. **Easier Interpretation:** A simpler dataset is generally easier to understand and analyze, especially when dealing with a large number of features.

7. **Reduced Storage Requirements:** Using fewer features can reduce the amount of storage space required for large datasets.

8. **Handling Multicollinearity:** Dimensionality reduction can reduce the effect of highly correlated features, which can be problematic for some machine learning and statistical methods.

Dimensionality reduction helps make complex datasets smaller, faster, clearer, and easier to analyze. It is therefore widely used as a preprocessing technique in machine learning and data analysis.

3.2.1.3 Curse of Dimensionality

The Curse of Dimensionality refers to the problems that arise when the number of features or dimensions in a dataset becomes very large. As dimensionality increases, data analysis and machine learning algorithms may become more difficult, slower, and less effective. High-dimensional data requires more computational resources and may contain unnecessary or redundant information. Key issues are:

- **Data Sparsity:** As the number of dimensions increases, data points become more widely scattered, making it difficult to identify meaningful patterns.

- **Increased Computational Cost:** A larger number of features requires more memory, processing time, and computational resources.

SGOU

- **Difficulty in Measuring Distance:** In high-dimensional spaces, measures such as distance and similarity may become less useful for identifying nearby or similar data points.
- **Higher Risk of Overfitting:** With too many features, a model may learn noise and random variations instead of the actual patterns in the data.
- **Difficulty in Visualization:** Data with many dimensions cannot be easily represented or visualized using traditional graphs.
- **Reduced Model Efficiency:** Some machine learning algorithms may perform poorly when they have to process a very large number of features.

The Curse of Dimensionality shows why it is important to manage the number of features in a dataset. Dimensionality reduction techniques such as PCA, SVD, t-SNE, and LDA can help simplify high-dimensional data, reduce computational complexity, and improve the effectiveness of data analysis and machine learning models.

3.2.1.4 Benefits of Dimensionality Reduction

Dimensionality reduction offers several benefits when working with datasets containing a large number of features. By reducing the number of dimensions, it helps simplify the structure of data and makes computational and analytical tasks more manageable. The major benefits are discussed below.

1. **Reduces Storage Requirements:** A dataset with fewer features requires less storage space and memory. This is particularly useful when handling large datasets such as images, videos, sensor data, and text collections.

2. **Speeds Up Learning Algorithms:** Machine learning algorithms have fewer variables to process after dimensionality reduction. This can reduce training time, processing requirements, and computational cost.

3. **Improves Model Generalization:** Removing irrelevant or redundant features can reduce the possibility of overfitting. The resulting model can focus on meaningful patterns and may perform better on previously unseen data.

4. **Reduces Multicollinearity:** Datasets may contain features that are highly correlated with each other. Dimensionality reduction can represent related features using fewer components, thereby reducing multicollinearity and improving the stability of some models.

5. **Enhances Data Visualization:** High-dimensional data is difficult to represent using conventional graphs. Reduction to two or three dimensions allows patterns, clusters, trends, and relationships to be visualized more effectively.

6. **Reduces Noise:** Some dimensions may contain irrelevant information or random variations. Removing less important dimensions can reduce noise and help highlight significant patterns in the data.

SGOU

7. **Simplifies Data Analysis:** Working with fewer dimensions makes complex datasets easier to explore, interpret, and process. It can also make relationships among important variables easier to identify.

Dimensionality reduction provides an effective way to manage complex, high-dimensional datasets. It can reduce storage and computational requirements, minimize redundancy and noise, improve model generalization, and support clearer visualization. Therefore, it is an important preprocessing technique in data analysis and machine learning.

3.2.1.5 Types: Feature Selection vs Feature Extraction

Dimensionality reduction can be broadly achieved through two approaches: Feature Selection and Feature Extraction. Both methods aim to reduce the number of dimensions in a dataset, but they differ in how the features are handled. Feature selection retains selected original features, whereas feature extraction creates new features from the existing ones (Fig. 3.2.1).

1. Feature Selection

Feature Selection is the process of choosing a relevant subset of features from the original dataset while removing features that are unnecessary, irrelevant, or redundant. The selected features remain unchanged and retain their original meaning. Some key characteristics are:

- Selects only the most relevant features.
- Removes irrelevant and redundant variables.
- Does not create new features.
- Reduces the number of inputs given to a machine learning model.
- Makes the selected variables easier to understand.

Common Feature Selection Methods

a. **Filter Methods:** Features are selected using statistical measures or relationships within the data. Examples include correlation, chi-square test, and mutual information.

b. **Wrapper Methods:** Different subsets of features are evaluated using a machine learning model, and the subset producing the best performance is selected.

c. **Embedded Methods:** Feature selection takes place during the model-training process. Examples include LASSO regression and decision-tree-based feature importance.

Advantages

Feature selection provides several benefits by identifying and retaining the most relevant original features while removing unnecessary variables. Some key points are:

SGOU

- Improves model performance by removing irrelevant and redundant features.
- Reduces computational cost by decreasing the number of input variables.
- Reduces overfitting by limiting unnecessary information used by the model.
- Improves interpretability because the selected features retain their original meaning.

2. Feature Extraction

Feature Extraction is the process of transforming the original features into a smaller set of new features that represent the important characteristics of the original data. The new features may be combinations or transformations of several original variables. Some key characteristics are:

- Creates new features from existing features.
- Combines information from multiple variables.
- Reduces the dimensionality of the dataset.
- Can reveal important patterns that may not be obvious in the original features.
- The resulting features may not have the same direct meaning as the original variables.

Examples of Feature Extraction

Common techniques include:

- **Principal Component Analysis (PCA)**: Creates principal components that capture major variations in the data.
- **Singular Value Decomposition (SVD):** Represents data using a smaller number of important components.
- **t-SNE**: Creates low-dimensional representations mainly for visualizing complex data.
- **Linear Discriminant Analysis (LDA)**: Creates discriminant components that emphasize differences between classes.

Advantages

Feature extraction provides several benefits by transforming multiple original features into a smaller set of meaningful components that represent important information in the dataset. Some key points are:

- Reduces Dimensionality by representing data using fewer features.
- Captures Important Patterns by combining information from multiple original features.
- Reduces Redundancy by representing correlated variables through new components.
- Supports Visualization by transforming high-dimensional data into 2D or 3D representations.

SGOU

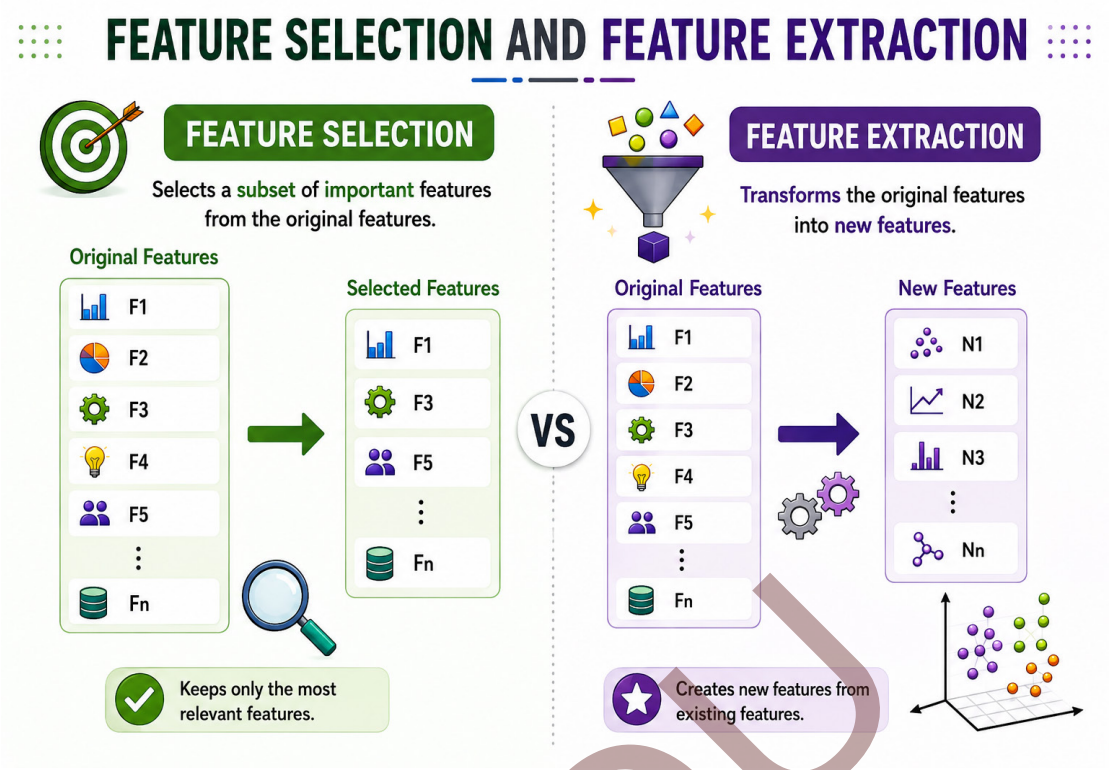


Fig. 3.2.1 Feature Selection and Feature Extraction

Feature Selection vs Feature Extraction

Feature selection and feature extraction are two important approaches to dimensionality reduction. While feature selection retains the most relevant original features, feature extraction transforms the original features into a smaller set of new features. Table 3.2.1 shows the key differences between feature selection and feature extraction based on their approach, treatment of original features, interpretability, purpose, and commonly used techniques.

Table 3.2.1: Feature Selection vs Feature Extraction

Aspect	Feature Selection	Feature Extraction
Basic Idea	Selects a subset of existing features	Creates new features from existing features
Original Features	Retained without modification	Transformed or combined
New Variables	No new variables are created	New variables are created
Interpretability	Generally high	Usually lower
Main Purpose	Removes irrelevant or redundant features	Creates a compact representation of data
Examples	Filter, Wrapper, and Embedded methods	PCA, SVD, t-SNE, and LDA

SGOU

Dimensionality reduction simplifies high-dimensional datasets while preserving the most important information. It helps reduce computational complexity, improve model performance, and make data easier to visualize and analyze. Understanding feature selection, feature extraction, and the curse of dimensionality provides a foundation for learning advanced techniques such as PCA, SVD, t-SNE, and LDA.

3.2.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used statistical technique for dimensionality reduction. It reduces the number of features in a dataset while preserving as much important information as possible. In many real-world datasets, several features may be correlated and contain similar information, resulting in redundancy. PCA addresses this problem by transforming the original correlated features into a new set of uncorrelated variables called principal components.

The main idea of PCA is to re-orient the data along new axes that represent the most important patterns in the dataset.

Key ideas of PCA

- **Linear Combination:** Each principal component is a linear combination of the original features.
- **Orthogonal Components:** The principal components are mutually orthogonal and therefore uncorrelated.
- **Variance:** Each component captures a certain amount of variation present in the data.
- **Component Ranking:** Principal components are arranged according to the amount of variance they capture.
- **Dimensionality Reduction:** The first few important components can be selected to represent the original dataset using fewer dimensions.

Intuitive Example

Consider a dataset containing two features plotted on a graph. Suppose most of the data points are distributed diagonally rather than horizontally or vertically. The original x-axis and y-axis may not represent the data efficiently.

PCA rotates the coordinate system to create new axes:

- **First Principal Component (PC1):** Follows the direction in which the data has the greatest spread and captures the maximum variance.
- **Second Principal Component (PC2):** Is perpendicular to PC1 and captures the remaining variation.

If PC1 contains most of the important information, PC2 may be removed, reducing the dataset from two dimensions to one dimension with relatively little information loss.

SGOU

3.2.2.1 Variance Maximization Principle

The fundamental principle of PCA is variance maximization. Variance describes how widely data points are distributed around their mean. A direction with greater variance generally provides more information about the structure and differences present in the dataset.

PCA identifies new directions called principal components and arranges them according to the amount of variance they capture.

- **First Principal Component (PC1):** Captures the **maximum possible variance** in the dataset.
- **Second Principal Component (PC2):** Captures the next highest variance while remaining **orthogonal to PC1**.
- **Subsequent Components:** Each following component captures the highest remaining variance while being orthogonal to the previously selected components.

Therefore, the first few principal components usually contain most of the important information, while later components contribute relatively less.

Importance of Variance Maximization

The variance maximization principle helps PCA:

- Retain the most important information in the first few components.
- Reduce the number of dimensions with relatively little information loss.
- Identify the major patterns and directions of variation in the dataset.
- Separate significant variations from less important variations that may sometimes represent noise.

Mathematically, PCA finds a direction vector such that the variance of the projected data is maximized. The direction satisfying this condition becomes a principal component.

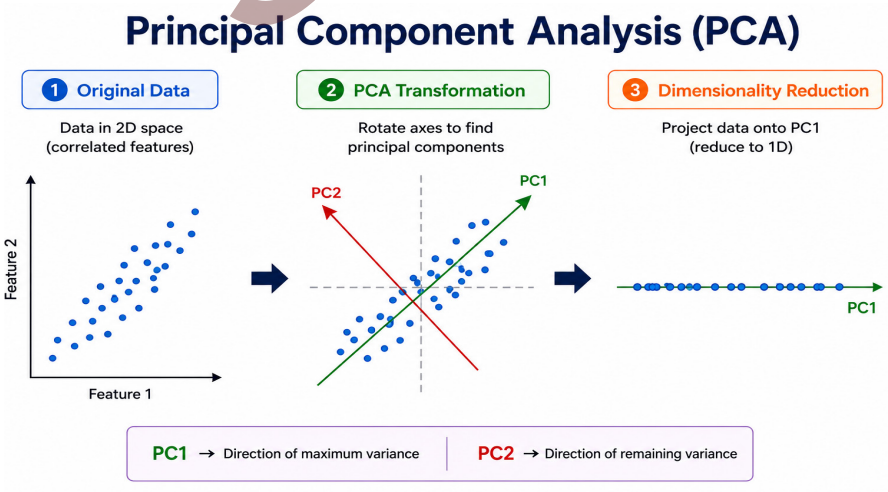


Fig. 3.2.2 Principal Component Analysis (PCA)

3.2.2.2 Geometric Interpretation

Geometrically, PCA can be understood as rotating the coordinate system and projecting data onto a lower-dimensional space. Instead of representing data using the original axes, PCA creates new axes that are better aligned with the distribution of the data.

For example, consider data with three dimensions. If most of the meaningful variation lies on a particular plane, PCA can project the 3D data onto that 2D plane, reducing the dimensionality while retaining most of the important information.

The geometric process can be summarized as:
Original Data → Rotate Axes → Identify Maximum-Variance Directions → Project Data → Reduced-Dimensional Representation

This projection simplifies the dataset while attempting to minimize the loss of important information.

Simple Representation

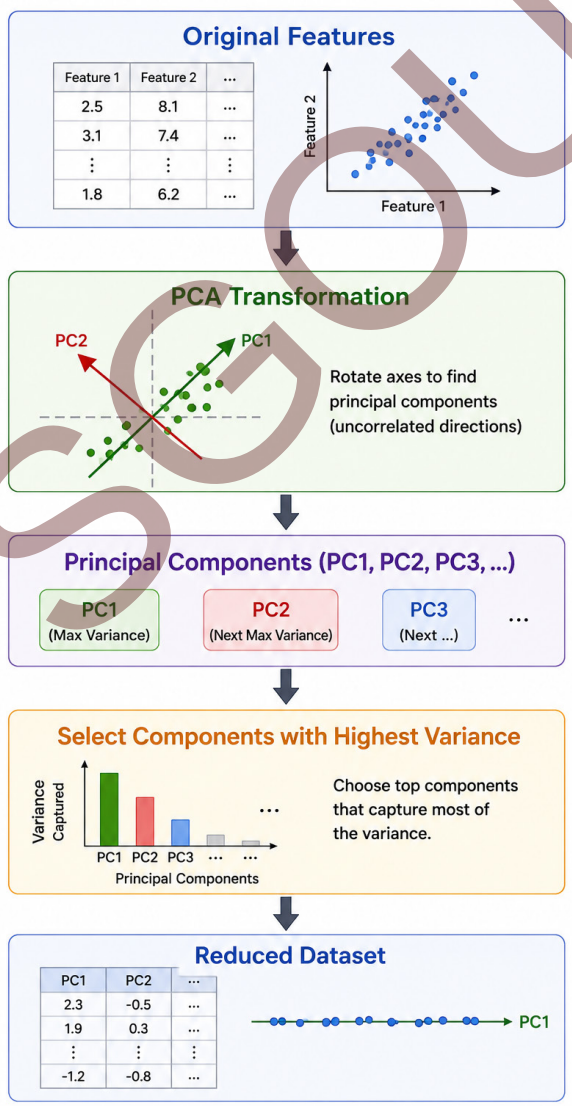


Fig. 3.2.3 Simple representation of PCA

SGOU

3.2.2.3 Key Characteristics of PCA

PCA has several important characteristics that make it useful for reducing the dimensionality of complex datasets.

- **Produces Uncorrelated Components:** PCA transforms the original features into principal components that are uncorrelated with each other.
- **Ranks Components by Variance:** Principal components are arranged according to the amount of variance they capture, with the most informative component placed first.
- **Works with Linear Relationships:** PCA is a linear technique and is most effective when the important relationships among features can be represented linearly.
- **Sensitive to Feature Scaling:** Features with larger numerical values can have a greater influence on PCA. Therefore, **standardization is usually required** when features are measured on different scales.
- **Reduces Dimensionality:** Only the most important principal components need to be retained, allowing the dataset to be represented using fewer dimensions.
- **Creates New Features:** The principal components are new variables formed from combinations of the original features.

3.2.2.4 Simple Example (Conceptual)

Consider a dataset containing two features:

- Height
- Weight

Height and weight are often related, so they may contain overlapping information. PCA can transform these two correlated features into two principal components:

- **PC1:** Captures the largest variation and may represent the overall **body-size pattern**.
- **PC2:** Captures the remaining variation that is not represented by PC1.

If PC1 captures most of the important variation, PC2 can be removed, allowing the original two features to be represented using a single principal component.

Height + Weight → PCA → PC1 + PC2 → Select PC1 → Reduced Representation

PCA simplifies high-dimensional data by creating new orthogonal components that capture the major variations in the dataset. Its variance maximization principle ensures that the most informative directions are selected first, making PCA useful for dimensionality reduction, visualization, noise reduction, and machine learning.

SGOU

3.2.2.5 Mathematical Foundation of PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique used to transform a dataset with many correlated features into a smaller set of new features called principal components. PCA is based on mathematical concepts such as covariance, eigenvalues, and eigenvectors.

The main idea of PCA is to find new directions in the data along which the maximum amount of variation (variance) is preserved. The mathematical foundation of PCA helps determine which directions are most important and which components can be removed without losing significant information.

1. Covariance Matrix

The **covariance matrix** is used to understand how different variables in a dataset vary with each other. It shows whether two features tend to increase or decrease together.

- **Positive covariance:** Both variables tend to increase or decrease together.
- **Negative covariance:** One variable tends to increase when the other decreases.
- **Zero covariance:** There is no linear relationship between the variables.

For example, if study hours and exam marks generally increase together, they may have positive covariance.

The covariance matrix is important in PCA because it helps identify relationships, correlations, and redundancy among features.

2. Eigenvalues and Eigenvectors

PCA calculates the eigenvalues and eigenvectors of the covariance matrix.

Eigenvectors

Eigenvectors represent the directions of the principal components. They indicate the new directions in which the data will be projected. In simple terms, an eigenvector tells PCA which direction to look at in the data.

Eigenvalues

Eigenvalues represent the amount of variance captured along each eigenvector. A larger eigenvalue means that the corresponding direction contains more information or variation in the dataset.

Therefore:

- Higher eigenvalue → More variance → More important component
- Lower eigenvalue → Less variance → Less important component

SGOU

3. Principal Components

Principal components (PCs) are new variables created from the original features through linear combinations. They transform the original feature space into a new coordinate system where the most important patterns in the data are represented by the first few components.

The components are arranged according to the amount of variance they capture:

- **PC1** → Captures the highest variance
- **PC2** → Captures the second-highest variance
- **PC3** → Captures the third-highest variance
- **PCn** → Captures progressively smaller amounts of variance

The principal components are also uncorrelated with each other, which helps reduce redundancy in the dataset.

3.2.2.6 Steps in PCA Algorithm

PCA is performed through a sequence of mathematical steps to transform the original dataset into a smaller set of principal components while preserving the maximum possible variance.

1. Data Standardization

The first step is to standardize the data so that all features are on a comparable scale.

- Convert each feature to zero mean and unit variance.
- Prevent features with larger numerical values from dominating the PCA results.
- Ensures a fair comparison among different features.

2. Covariance Matrix Computation

The covariance matrix is calculated to identify how the features vary with respect to each other.

- Measures relationships between variables.
- Helps identify correlated and redundant features.
- Provides the mathematical basis for finding the principal directions.

3. Eigen Decomposition

The covariance matrix is decomposed to obtain its eigenvalues and eigenvectors.

- Eigenvectors represent the directions of the principal components.
- Eigenvalues indicate the amount of variance captured by each direction.
- Components are ranked according to their eigenvalues.

SGOU

4. Feature Vector Formation

The eigenvectors are arranged in descending order based on their corresponding eigenvalues.

- Select the top k eigenvectors.
- These eigenvectors represent the most important directions in the dataset.
- Combine them to form a feature vector or projection matrix.

5. Transform the Data

The original standardized data is projected onto the selected principal components.

- Multiply the standardized data by the projection matrix.
- Convert the original high-dimensional data into a lower-dimensional representation.
- The resulting dataset contains the selected principal components.

Thus, PCA transforms high-dimensional data into a smaller set of principal components while preserving the most important information and reducing redundancy among features.

PCA Algorithm Flow

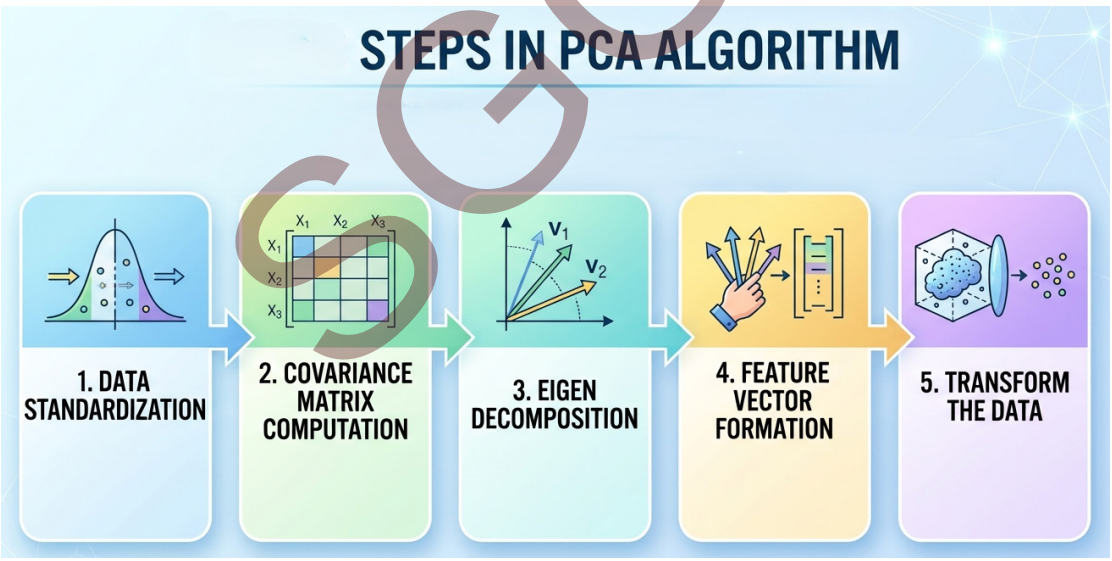


Fig. 3.2.4: PCA Algorithm

3.2.2.7 Advantages and Limitations of PCA

Advantages of PCA

PCA offers several advantages in data analysis by reducing the number of features while retaining the most important information in the dataset.

SGOU

1. **Reduces Dimensionality:** PCA reduces the number of features while retaining most of the important information in the dataset.

2. **Reduces Information Redundancy:** It transforms correlated features into uncorrelated principal components, reducing redundancy.

3. **Improves Computational Efficiency:** Working with fewer features can reduce computation time and memory requirements, especially for large datasets.

4. **Useful for Data Visualization:** PCA can reduce high-dimensional data to two or three dimensions, making it easier to visualize patterns and clusters.

5. **Helps Handle Multicollinearity:** ince principal components are uncorrelated, PCA can reduce problems caused by highly correlated predictor variables.

Thus, PCA improves data efficiency, reduces redundancy, and makes high-dimensional data easier to analyze and visualize.

Limitations of PCA

Although PCA is useful for dimensionality reduction, it has certain limitations that should be considered before applying it to a dataset.

1. **Loss of Interpretability:** Principal components are combinations of the original features, so they may be difficult to interpret in their original meaning.

2. **Sensitive to Feature Scaling:** Features with larger numerical scales can dominate the results. Therefore, standardization is usually required before applying PCA.

3. **Assumes Linear Relationships:** PCA identifies linear directions of maximum variance and may not effectively capture complex non-linear relationships.

4. **Possible Information Loss:** When only a few components are retained, some information contained in the discarded components is lost.

5. **Sensitive to Outliers:** Extreme values can strongly influence the covariance matrix and may affect the resulting principal components.

Therefore, PCA should be used carefully when interpretability, feature scaling, non-linear relationships, or outliers are important.

3.2.2.8 Applications of PCA

Principal Component Analysis (PCA) is widely used in data analysis and machine learning to reduce the number of features, remove redundancy, and identify important patterns in high-dimensional datasets.

1. **Data Visualization:** PCA reduces high-dimensional datasets to two or three principal components, making complex data easier to visualize using graphs and plots.

SGOU

2. **Image Processing:** PCA can reduce the number of image features while preserving important visual information. It is used in image compression, face recognition, and image analysis.

3. **Machine Learning:** PCA is used as a preprocessing technique to reduce the number of input features before applying classification or regression algorithms.

4. **Feature Reduction:** PCA helps remove redundant and highly correlated features, producing a smaller set of representative variables.

5. **Pattern Recognition:** PCA helps identify important patterns and structures in large datasets, making it useful in pattern recognition and classification tasks.

6. **Anomaly Detection:** PCA can identify unusual observations by examining how far data points deviate from the main principal-component space.

7. **Signal Processing:** PCA is used to reduce noise and extract important information from complex signals by retaining the components that contain the most significant variation.

8. **Bioinformatics**: PCA is applied to high-dimensional biological data, such as gene expression datasets, to identify patterns and relationships among samples.

Thus, PCA is useful in many fields where datasets contain a large number of correlated features, helping simplify data while preserving its most important patterns.

3.2.3 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a powerful mathematical technique used for matrix factorization. It decomposes a matrix into three simpler matrices that reveal important structural information contained in the original data.

SVD is applicable to square as well as rectangular matrices, making it particularly useful for high-dimensional datasets. It is widely used in dimensionality reduction, data compression, image processing, information retrieval, recommendation systems, natural language processing, and machine learning.

The basic idea of SVD is to represent a complex matrix using a set of important directions and corresponding weights. By retaining only the most significant components, the original matrix can be approximated using fewer dimensions.

Matrix Factorization

Matrix factorization is the process of representing a complex matrix as the product of simpler matrices. SVD factorizes a matrix into three matrices. Each matrix provides different information about the original data. This makes the data easier to analyze, process, compress, and transform.

SGOU

Let A be an m×n matrix. SVD decomposes A as:

$A=U\Sigma V^T$

where:

- A is the Original matrix
- U is the Left singular vector matrix
- Σ is the Diagonal matrix of singular values
- V^T is the Transpose of the right singular vector matrix

This factorization separates the information contained in the original matrix into different mathematical components.

3.2.3.1 Components of SVD

Singular Value Decomposition (SVD) decomposes an original matrix (A) into three main matrices. Each matrix has a specific role in representing the structure and information contained in the original data.

1. U – Left Singular Vector Matrix

The U matrix contains the left singular vectors.

- Represents important directions related to the rows or observations of the matrix.
- Its columns are orthogonal unit vectors.
- Describes the structure of the data from the observation or row perspective.

2. Σ – Singular Value Matrix

The **Σ** matrix is a diagonal matrix containing the singular values.

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

The singular values are arranged from largest to smallest:

$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq 0$

They indicate the importance of the corresponding components.

- Large singular value → More important component
- Small singular value → Less important component
- Zero singular value → No contribution from that component

SGOU

3. V^T – Right Singular Vector Matrix

The **V^T matrix** is the transpose of the right singular vector matrix (V).

- Represents important directions related to the columns or features of the matrix.
- Shows how the original features contribute to different components.
- The columns of (V) are orthogonal unit vectors.

Table 3.2.2 Description of Components of SVD

Component	Description	Main Role
U	Left singular vectors	Represents row/observation directions
Σ	Singular values	Indicates the importance of components
V^T	Right singular vectors	Represents column/feature directions

In summary, U represents the row directions, Σ represents the importance of the components, and V^T represents the feature directions of the original matrix.

3.2.3.2 Relationship between PCA and SVD

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are closely related mathematical techniques used for dimensionality reduction and extracting important patterns from high-dimensional data. SVD provides an efficient way to compute PCA directly from the data matrix.

1. **PCA can be computed using SVD:** SVD provides an efficient mathematical method for calculating PCA, especially for large datasets.
2. **Both identify important directions:** Both PCA and SVD identify directions that capture the most significant variation or information in the data.
3. **Both support dimensionality reduction:** PCA and SVD can reduce high-dimensional data by retaining only the most important components.
4. **SVD operates directly on the data matrix:** SVD decomposes the centered data matrix directly as $X=U\Sigma V^T$, without explicitly forming the covariance matrix.
5. **PCA commonly uses the covariance matrix:** Traditional PCA calculates the covariance matrix and then finds its eigenvalues and eigenvectors to determine the principal components.
6. **Singular values correspond to PCA variance:** The squared singular values

SGOU

obtained from SVD are related to the variance captured by PCA components. Therefore, larger singular values indicate more important principal components.

Thus, PCA and SVD are closely connected, with SVD providing an efficient mathematical approach for obtaining PCA components while reducing computational complexity and preserving important information.

3.2.3.3 Advantages and Limitations of SVD

Advantages of SVD

Singular Value Decomposition (SVD) offers several advantages for analyzing, simplifying, and processing large and high-dimensional datasets.

1. **Dimensionality Reduction**: Reduces the number of dimensions while retaining important information.
2. **Data Compression**: Represents large matrices using fewer significant components, reducing storage requirements.
3. **Noise Reduction**: Less important singular components can be removed to reduce noise.
4. **Works with Rectangular Matrices**: Can be applied to both square and rectangular matrices.
5. **Handles High-Dimensional Data**: Useful for extracting important patterns from complex datasets.
6. **Wide Range of Applications:** Used in image processing, recommendation systems, NLP, information retrieval, and machine learning.

Thus, SVD is a versatile technique that simplifies complex data, reduces computational requirements, and preserves the most important information.

Limitations of SVD

Despite its usefulness, SVD has certain limitations that should be considered when applying it to large or complex datasets.

1. **Computationally Expensive:** Full SVD can require considerable computational resources for very large matrices.
2. **Difficult to Interpret**: Singular vectors may not have a clear real-world meaning.
3. **Information Loss**: Removing smaller singular values may result in the loss of some information.
4. **Choice of Components**: Selecting an inappropriate number of components can affect the quality of the reduced representation.

SGOU

5. **Memory Requirements**: Storing and processing large matrices can require significant memory.

6. **Sensitive to Data Representation**: Results can be affected by preprocessing, scaling, and centering of the data.

Therefore, SVD should be applied carefully by considering the dataset size, preprocessing requirements, desired accuracy, and number of components to be retained.

3.2.3.4 Applications of SVD

Singular Value Decomposition (SVD) is widely used in data analysis, machine learning, and computational applications to simplify complex matrices, reduce dimensionality, and extract important patterns from high-dimensional data.

1. **Dimensionality Reduction:** SVD reduces the number of features by retaining the most significant singular components while removing less important information.

2. **Image Compression:** Images can be represented as matrices of pixel values. SVD can retain the most important components to represent the image using less storage space.

3. **Recommendation Systems:** SVD can identify hidden relationships between users and products or movies, helping systems provide personalized recommendations.

4. **Information Retrieval:** SVD is used to identify relationships between terms and documents and is an important technique in Latent Semantic Analysis (LSA).

5. **Natural Language Processing:** SVD can reduce the dimensionality of document-term matrices and help identify underlying patterns in text data.

6. **Noise Reduction:** Less significant singular components can be removed to reduce noise while preserving the important structure of the original data.

7. **Data Compression:** Large matrices can be represented using a smaller number of significant components, reducing storage and computational requirements.

8. **Machine Learning:** SVD can be used as a preprocessing technique to simplify high-dimensional datasets before applying machine learning algorithms.

Thus, SVD is a versatile technique that supports dimensionality reduction, compression, pattern extraction, and efficient data processing across a wide range of applications.

3.2.4 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique mainly designed for visualizing high-dimensional data in two or three dimensions. It converts complex relationships in high-dimensional data into a lower-dimensional representation while attempting to preserve the relationships between nearby data points.

SGOU

Non-Linear Dimensionality Reduction

Unlike PCA, which mainly captures linear relationships and global variance, t-SNE is designed to capture complex and non-linear patterns in the data.

t-SNE focuses primarily on preserving local structures, meaning that data points that are close to each other in the original high-dimensional space are generally placed close together in the lower-dimensional representation. This makes t-SNE particularly useful for identifying clusters, groups, and hidden patterns in complex datasets.

In summary, t-SNE transforms high-dimensional data into a low-dimensional visual representation while preserving important local relationships among data points.

3.2.4.1 Working Principle of t-SNE

The t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm works by converting the relationships between data points in a high-dimensional space into probabilities and then creating a low-dimensional representation that preserves these relationships as closely as possible. It mainly focuses on maintaining local similarities between nearby data points.

Steps in the Working Principle

1. **Calculate Pairwise Similarities:** t-SNE measures the similarity between pairs of data points in the high-dimensional space. Nearby points are assigned higher similarity, while distant points receive lower similarity.

2. **Convert Similarities into Probabilities:** The similarities are converted into probability distributions. A high probability indicates that two data points are likely to be neighbors.

3. **Create a Low-Dimensional Representation:** The data points are initially placed in a two- or three-dimensional space, usually with random or suitable initial positions.

4. **Calculate Low-Dimensional Similarities:** t-SNE calculates similarities between points in the low-dimensional space using a Student's t-distribution.

5. **Minimize the Difference:** t-SNE compares the high-dimensional and low-dimensional probability distributions and minimizes their difference using Kullback–Leibler (KL) divergence.

6. **Iteratively Update the Points:** The positions of the low-dimensional points are repeatedly adjusted so that similar points move closer together and dissimilar points move farther apart.

7. **Generate the Final Visualization:** After several iterations, the resulting 2D or 3D representation reveals clusters, groups, and local patterns in the original high-dimensional data.

SGOU

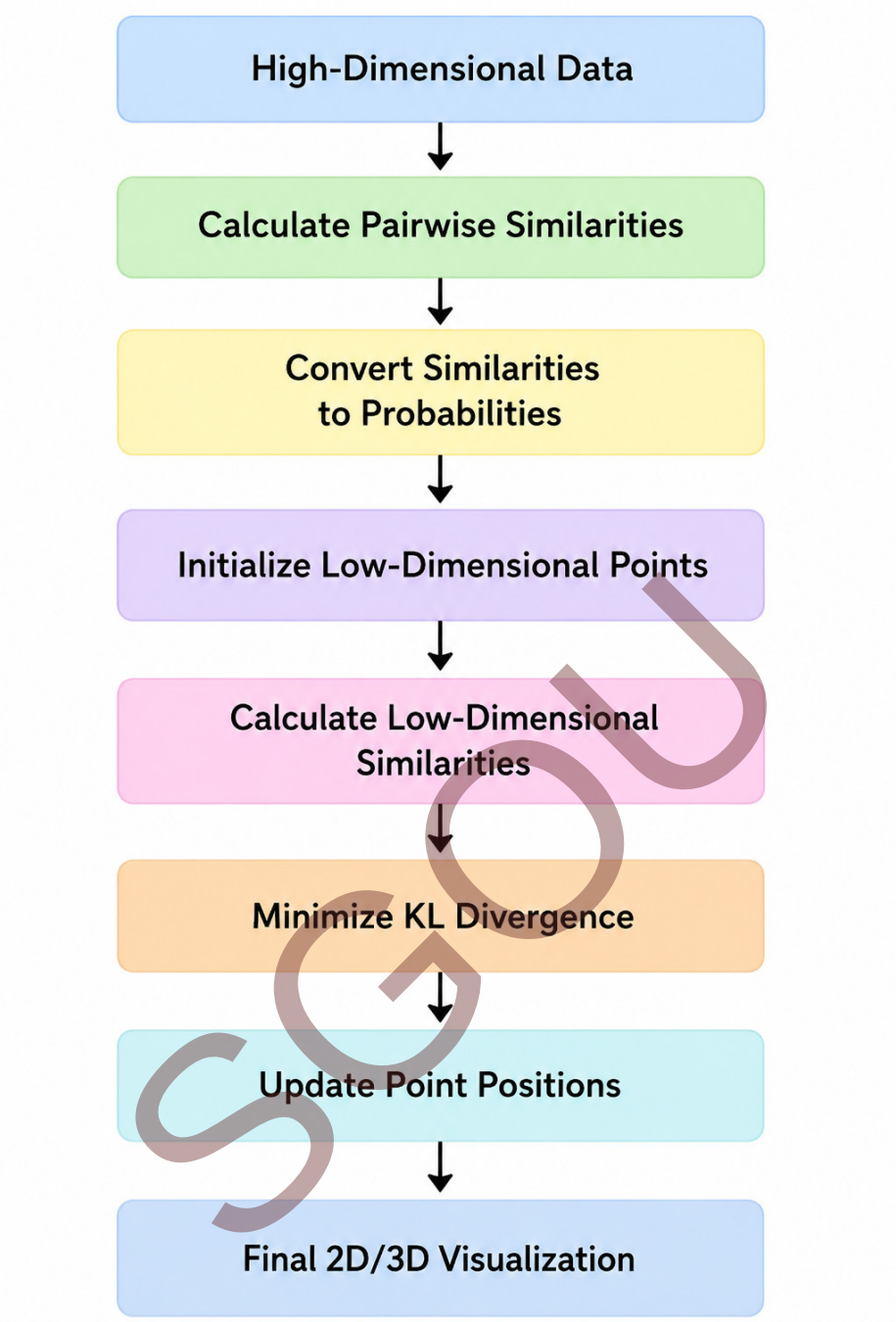


Fig. 3.2.5 Working flow of t-SNE

Thus, t-SNE works by preserving the probability-based relationships between nearby points, allowing complex high-dimensional data to be represented visually in a lower-dimensional space.

3.2.4.2 Key Parameters in t-SNE

The performance and quality of t-SNE depend on several important parameters. These parameters control how local relationships are identified and how the low-dimensional representation is optimized.

1. **Perplexity:** Perplexity controls the approximate number of nearest neighbors considered for each data point. A low value focuses more on very local relationships, while a higher value considers a broader neighborhood.

2. **Learning Rate:** The learning rate controls the size of the updates made during optimization. A suitable learning rate helps the algorithm reach a stable and meaningful representation.

3. **Iterations:** Iterations specify the number of optimization steps performed by t-SNE. More iterations generally give the algorithm more time to improve the arrangement of points.

Thus, selecting suitable values for perplexity, learning rate, and iterations is important for obtaining a clear and meaningful t-SNE visualization.

3.2.4.3 Advantages and Limitations of t-SNE

Advantages of t-SNE

t-SNE provides several advantages for exploring and visualizing complex, high-dimensional datasets, particularly when the data contains non-linear relationships and distinct groups.

1. **Captures Non-Linear Relationships:** Identifies complex patterns that linear techniques such as PCA may not capture effectively.

2. **Preserves Local Structure:** Keeps similar or nearby data points close together in the reduced representation.

3. **Effective for Visualization:** Converts high-dimensional data into 2D or 3D representations that are easy to visualize.

4. **Reveals Clusters**: Helps identify groups and clusters that may be difficult to observe in the original dataset.

5. **Useful for Exploratory Data Analysis**: Helps researchers understand the structure and distribution of complex datasets.

6. **Works with High-Dimensional Data**: Can handle datasets containing a large number of features.

Thus, t-SNE is a powerful tool for visualizing high-dimensional data and discovering local patterns and clusters.

Limitations of t-SNE

Although t-SNE is highly effective for visualization, it has several limitations related to computational cost, parameter selection, and interpretation of the resulting visualization.

1. **Computationally Expensive**: It can require significant processing time for large datasets.

SGOU

2. **Sensitive to Parameters**: Results can vary depending on perplexity, learning rate, and number of iterations.

3. **Mainly Preserves Local Structure**: It may not accurately preserve global relationships or distances between distant clusters.

4. **Difficult to Interpret Distances**: The distance between clusters in a t-SNE plot does not necessarily represent their true distance in the original data.

5. **Non-Deterministic Results**; Different initializations or settings can produce different visualizations.

6. **Not Ideal for Direct Feature Reduction**: t-SNE is primarily designed for visualization rather than creating features for further predictive modeling.

Therefore, t-SNE should be used mainly as a visualization and exploratory analysis tool, with careful consideration of its parameters and interpretation.

3.2.4.4 Applications of t-SNE

t-SNE is widely used for visualizing and exploring high-dimensional datasets, especially when the data contains complex, non-linear relationships and hidden clusters.

1. **Data Visualization:** Converts high-dimensional data into 2D or 3D representations, making complex datasets easier to understand.

2. **Cluster Analysis:** Helps identify and visualize natural groups or clusters within a dataset.

3. **Image Analysis:** Used to visualize patterns and similarities among images and image features, such as handwritten digits or facial images.

4. **Natural Language Processing:** Helps visualize relationships among words, documents, or word embeddings in high-dimensional spaces.

5. **Bioinformatics:** Used to explore high-dimensional biological data, such as gene expression and single-cell datasets, and identify groups of similar samples.

6. **Machine Learning:** Helps visualize learned feature representations and understand how different classes are distributed in a model's feature space.

7. **Anomaly Detection:** Can help identify unusual data points that appear separate from major groups in the visualization.

8. **Exploratory Data Analysis:** Supports the discovery of hidden patterns, relationships, and structures before applying further analytical techniques.

Thus, t-SNE is particularly valuable for exploring complex high-dimensional data visually, helping researchers identify clusters, patterns, and relationships that may not be apparent from the original data.

SGOU

3.2.5 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique used to reduce the number of features while maximizing the separation between different classes. Unlike unsupervised methods such as PCA, LDA uses the class labels of the data to find directions that provide the best discrimination between classes.

Supervised Dimensionality Reduction

LDA is called a supervised dimensionality reduction technique because it requires labeled training data. It considers both the features and their corresponding class labels when finding the reduced representation.

The main objectives of LDA are to:

- Maximize the separation between different classes
- Minimize the variation within each class
- Reduce the number of dimensions
- Improve the performance of classification algorithms

For example, if a dataset contains samples belonging to Class A, Class B, and Class C, LDA finds new directions where the samples from different classes are as far apart as possible while keeping samples from the same class close together.

3.2.5.1 Mathematical Foundation of LDA

The mathematical foundation of LDA is based on two important types of scatter:

1. **Within-Class Scatter:** Measures how widely the data points are distributed within each class.
2. **Between-Class Scatter:** Measures how far the class means are separated from each other.

LDA searches for a projection that maximizes between-class scatter while minimizing within-class scatter.

The objective can be expressed as:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

where:

- w is the projection vector
- S_B is the between-class scatter matrix
- S_W is the within-class scatter matrix

SGOU

Thus, LDA finds the projection directions that provide maximum class separability.

3.2.5.2 Steps in LDA Algorithm

The Linear Discriminant Analysis (LDA) algorithm follows a systematic process to reduce the dimensionality of labeled data while maximizing the separation between different classes.

1. **Collect Labeled Data:** Start with a dataset containing input features and their corresponding class labels.

2. **Calculate Class Means:** Compute the mean vector for each class to identify the central position of each class in the feature space.

3. **Compute Within-Class Scatter Matrix:** Calculate the within-class scatter matrix (S_W) to measure the variation of data points within each class.

4. **Compute Between-Class Scatter Matrix:** Calculate the between-class scatter matrix (S_B) to measure the separation between the different class means.

5. **Solve the Eigenvalue Problem:** Find the eigenvalues and eigenvectors of the matrix. The eigenvectors represent the directions that provide maximum class separation.

6. **Select the Best Discriminant Components:** Arrange the eigenvectors according to their eigenvalues and select the eigenvectors with the largest eigenvalues.

7. **Form the Projection Matrix:** Use the selected eigenvectors to construct a projection matrix for transforming the original data.

8. **Transform the Data:** Project the original data onto the selected discriminant directions to obtain a lower-dimensional representation.

SGOU

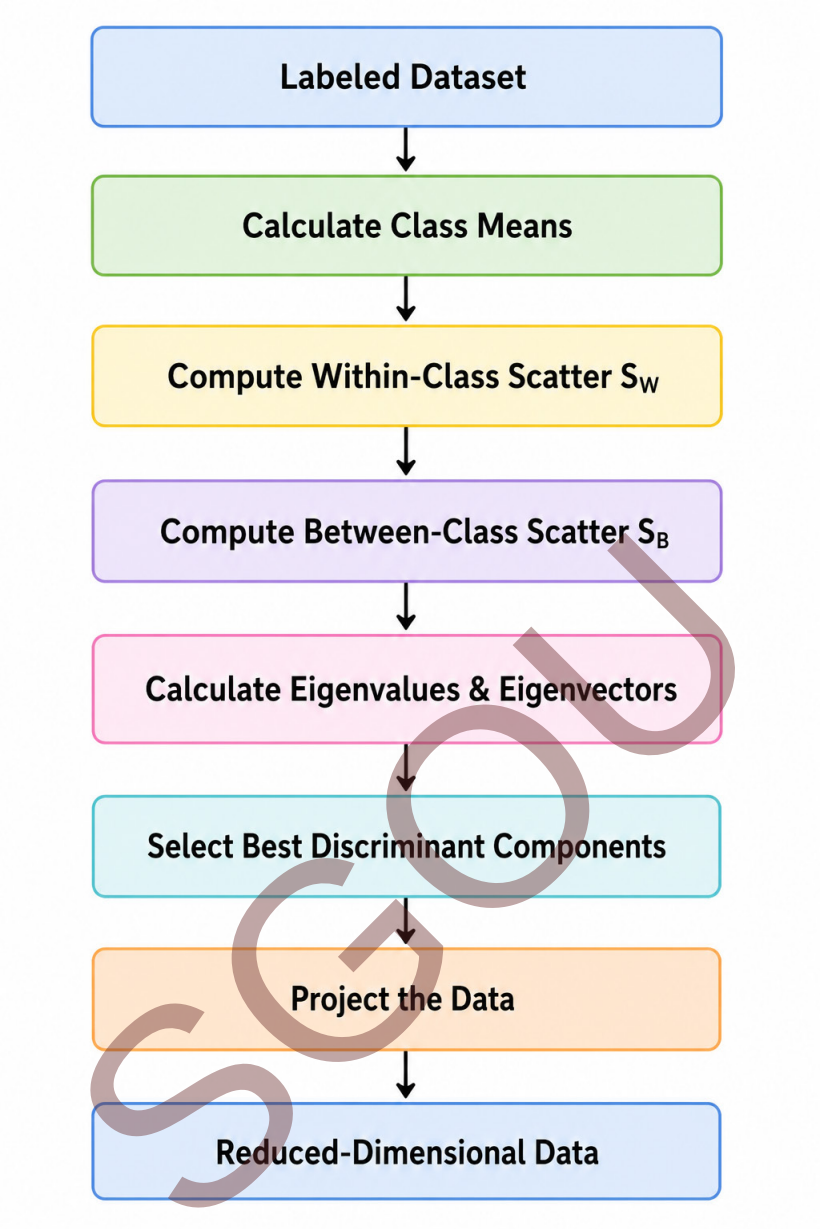


Fig. 3.2.6: Working flow of LDA

Thus, LDA reduces the dimensionality of labeled data by finding projection directions that maximize separation between classes while minimizing variation within each class.

3.2.5.3 Advantages and Limitations of LDA

Advantages of LDA

Linear Discriminant Analysis (LDA) provides several advantages for supervised dimensionality reduction by using class information to create a representation with better class separation.

1. **Reduces Dimensionality**: Reduces the number of features while retaining important class-related information.

2. **Improves Class Separation**: Maximizes the separation between different classes.

3. **Useful for Classification**: Produces features that can improve the performance of classification algorithms.

4. **Uses Class Labels**: Utilizes labeled data to find the most discriminative directions.

5. **Reduces Computational Complexity**: Fewer dimensions can make subsequent analysis and classification more efficient.

6. **Easy to Interpret**: The discriminant directions can provide insight into how features contribute to class separation.

Thus, LDA is an effective technique for reducing dimensionality while preserving information that is important for distinguishing between classes.

Limitations of LDA

Although LDA is useful for supervised dimensionality reduction, it has certain limitations related to its assumptions and the nature of the data.

1. **Requires Labeled Data**: LDA cannot be directly applied when class labels are unavailable.

2. **Assumes Linear Separability**: It works best when classes can be separated using linear boundaries.

3. **Assumes Similar Class Covariance**: Traditional LDA assumes that different classes have similar covariance structures.

4. **Sensitive to Outliers** – Extreme observations can affect class means and scatter matrices.

5. **Limited Number of Components**: For (C) classes, LDA can produce at most (C-1) discriminant components.

6. **May Not Handle Non-Linear Patterns Well**: Complex non-linear relationships may not be captured effectively.

Therefore, LDA is most suitable for labeled datasets where the classes have reasonably distinct and approximately linear structures.

3.2.5.4 Applications of LDA

Linear Discriminant Analysis (LDA) is widely used in classification and supervised dimensionality reduction. It is particularly useful when the objective is to reduce the number of features while maintaining clear separation between different classes.

1. **Classification:** LDA is used to classify data into different categories by finding directions that provide maximum separation between classes.

SGOU

2. **Face Recognition:** LDA can reduce facial features into discriminative components and help distinguish between different individuals.

3. **Pattern Recognition:** It is used to identify and classify patterns in areas such as image, speech, and character recognition.

4. **Medical Diagnosis:** LDA can classify patients into different diagnostic groups based on clinical or laboratory features.

5. **Image Processing:** LDA is used for feature extraction and dimensionality reduction in image classification tasks.

6. **Text Classification:** LDA can help classify documents into predefined categories based on their extracted features.

7. **Customer and Market Classification:** It can be used to classify customers into groups based on purchasing behavior, preferences, or demographic features.

8. **Bioinformatics:** LDA can help classify biological samples based on high-dimensional genetic or molecular features.

Thus, LDA is useful in applications where labeled data and class separation are important, making it valuable for classification, pattern recognition, and supervised dimensionality reduction.

3.2.6 Comparison of Dimensionality Reduction Techniques

Dimensionality reduction techniques are used to reduce the number of features in a dataset while retaining important information. Different techniques use different approaches to identify and preserve useful patterns. PCA, SVD, t-SNE, and LDA are commonly used techniques with different objectives and characteristics.

Table 3.2.3 Comparison of Dimensionality Reduction Techniques

Feature	PCA	SVD	t-SNE	LDA
Type	Linear	Matrix factorization	Non-linear	Linear
Learning Type	Unsupervised	Unsupervised / mathematical decomposition	Unsupervised	Supervised
Main Objective	Maximize variance	Decompose a matrix	Preserve local structure	Maximize class separation
Uses Class Labels	No	No	No	Yes

SGOU

Main Focus	Overall variance	Important matrix components	Local relationships	Between-class separation
Typical Output	Principal components	Singular vectors and values	2D/3D embedding	Discriminant components
Visualization	Useful	Useful	Excellent	Useful
Non-linear Patterns	Not suitable	Not directly	Suitable	Not directly
Classification	Useful as preprocessing	Useful as preprocessing	Limited	Highly suitable
Interpretability	Moderate	Moderate	Low	Moderate
Common Applications	Feature reduction, visualization	Compression, PCA, recommendation systems	Cluster visualization, pattern exploration	Classification, face recognition, diagnosis

Key Differences

- PCA reduces dimensions by finding directions that capture the maximum variance.
- SVD decomposes a matrix into three simpler matrices and can be used to perform PCA efficiently.
- t-SNE is a non-linear technique that mainly preserves local relationships and is particularly useful for visualization.
- LDA is a supervised technique that uses class labels to maximize separation between different classes.

Thus, the choice of dimensionality reduction technique depends on the purpose of the analysis. PCA is suitable for general dimensionality reduction, SVD is useful for matrix decomposition and compression, t-SNE is effective for visualizing complex non-linear patterns, and LDA is appropriate when class labels are available and class separation is important.

3.2.7 Challenges in Dimensionality Reduction

Dimensionality reduction is useful for simplifying high-dimensional datasets, but reducing the number of features can introduce several challenges. The main difficulty is to remove unnecessary information while preserving the important structure and patterns in the data.

1. **Information Loss:** Removing features or components may result in the loss of useful information from the original dataset.

SGOU

2. **Loss of Interpretability:** New reduced features, such as principal components or embeddings, may be combinations of many original features and can be difficult to interpret.

3. **Selection of Optimal Dimensions:** Choosing the appropriate number of components or dimensions is often challenging. Too few dimensions may cause information loss, while too many may reduce the benefits of dimensionality reduction.

4. **Computational Cost:** Some techniques, such as SVD and t-SNE, can require significant computational resources when applied to very large datasets.

5. **Sensitivity to Data Quality:** Missing values, noise, outliers, and inappropriate feature scaling can affect the quality of the reduced representation.

6. **Non-Linear Relationships:** Linear techniques such as PCA and LDA may not effectively capture complex non-linear relationships present in the data.

7. **Parameter Selection:** Techniques such as t-SNE require careful selection of parameters such as perplexity, learning rate, and iterations.

8. **Loss of Global Structure:** Some techniques, particularly t-SNE, focus strongly on local relationships and may not accurately preserve distances or relationships between distant groups.

9. **Dependence on the Dataset:** No single dimensionality reduction technique works equally well for every dataset. The appropriate method depends on the data characteristics and analytical objective.

Thus, dimensionality reduction requires a careful balance between reducing complexity and preserving meaningful information. The choice of technique, number of dimensions, and preprocessing methods should be based on the nature and purpose of the dataset.

3.2.8 Real-world Applications of Dimensionality Reduction

Dimensionality reduction is widely used in real-world applications where datasets contain a large number of features. By reducing unnecessary or redundant features, it helps simplify data, improve computational efficiency, visualize complex information, and support machine learning tasks.

1. **Image and Face Recognition:** Techniques such as PCA and LDA are used to reduce the number of image features while retaining important visual information. This helps in face recognition, object recognition, and image classification.

2. **Medical and Healthcare Analysis:** Medical datasets may contain hundreds or thousands of variables, such as gene expressions, medical images, and patient measurements. Dimensionality reduction helps identify important patterns for disease diagnosis and patient classification.

SGOU

3. **Bioinformatics and Genomics:** Gene expression datasets contain a very large number of features. PCA and t-SNE can help visualize and identify patterns, groups, and relationships among biological samples.

4. **Natural Language Processing (NLP):** Text data can contain thousands of word or document features. Dimensionality reduction is used to create compact representations of text, words, and documents for text classification and semantic analysis.

5. **Recommendation Systems:** SVD can reduce large user-item matrices into smaller representations that capture hidden relationships between users and products, movies, or other items.

6. **Financial Data Analysis:** Financial datasets often contain many correlated variables. PCA can reduce these variables into a smaller number of important factors for risk analysis, portfolio analysis, and market pattern identification.

7. **Customer Segmentation:** Businesses may collect many customer attributes such as purchasing behavior, preferences, and interactions. Dimensionality reduction helps visualize and analyze customer groups before applying clustering techniques.

8. **Anomaly and Fraud Detection:** Reducing high-dimensional transaction data can help identify unusual patterns that may indicate fraudulent or abnormal activities.

9. **Sensor and IoT Data Analysis:** IoT systems generate large amounts of data from multiple sensors. Dimensionality reduction helps compress and simplify this data for monitoring, pattern detection, and predictive analysis.

10. **Visualization of High-Dimensional Data:** Techniques such as PCA and t-SNE can convert high-dimensional data into 2D or 3D representations, allowing researchers and analysts to visually identify clusters, trends, and hidden patterns.

Thus, dimensionality reduction plays an important role in healthcare, finance, image processing, NLP, recommendation systems, IoT, and business analytics, helping organizations extract meaningful information from complex high-dimensional datasets efficiently.

Dimensionality reduction techniques such as PCA, SVD, t-SNE, and LDA are essential tools in modern data analysis. They help simplify complex datasets, improve computational efficiency, and enable better visualization and interpretation. Selecting the appropriate technique depends on the nature of the data and the problem being solved.

SGOU

Summerised Overview

Dimensionality reduction is a technique used to reduce the number of features in high-dimensional datasets while retaining important information. It helps handle complex data, reduce computational requirements, remove redundancy, and improve data visualization and analysis.

Four important techniques are covered: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Linear Discriminant Analysis (LDA). PCA identifies directions of maximum variance, SVD decomposes matrices into simpler components, t-SNE preserves local relationships for effective visualization, and LDA reduces dimensions by maximizing separation between labeled classes.

The mathematical foundations, working principles, advantages, limitations, and applications of these techniques are also discussed. These concepts help in selecting appropriate dimensionality reduction methods for real-world applications in image processing, healthcare, finance, natural language processing, bioinformatics, and machine learning.

SGOU

Assignments

1. Explain the concept and importance of dimensionality reduction. Describe the working steps of Principal Component Analysis (PCA).

2. Explain Singular Value Decomposition (SVD) and its mathematical representation. Discuss its applications in dimensionality reduction.

3. Describe the working principle of t-SNE. Explain its key parameters, advantages, limitations, and applications.

4. Explain Linear Discriminant Analysis (LDA) as a supervised dimensionality reduction technique. Describe its algorithm and applications.

5. Compare PCA, SVD, t-SNE, and LDA based on their objectives, working principles, advantages, limitations, and applications.

Suggested Reading

1. Scikit-learn – Principal Component Analysis (PCA) https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

2. Scikit-learn – Dimensionality Reduction https://scikit-learn.org/stable/modules/decomposition.html

3. Scikit-learn – Linear Discriminant Analysis (LDA) https://scikit-learn.org/stable/modules/lda_qda.html

4. IBM – Dimensionality Reduction https://www.ibm.com/think/topics/dimensionality-reduction

5. Scikit-learn – Manifold Learning and t-SNE https://scikit-learn.org/stable/modules/manifold.html

Reference

1. Rad, J. A., Chakraverty, S., & Parand, K. (2025). *Dimensionality reduction in machine learning*. Elsevier.

2. Ghojogh, B., Crowley, M., Karray, F., & Ghodsi, A. (2023). *Elements of dimensionality reduction and manifold learning*. Springer.

3. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer.

4. Alpaydin, E. (2020). *Introduction to machine learning* (4th ed.). The MIT Press.

5. Prince, S. J. D. (2025). *Understanding machine learning*. Cambridge University Press.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

3 UNIT Association Rule Learning

Learning Outcomes

After completion of this unit, the learner will be able to:

- understand the concept of Association Rule Learning and its purpose in data mining
- describe the meaning of support, confidence, and lift in evaluating association rules
- comprehend the process of identifying frequent itemsets from transactional data
- summarize the basic working of Apriori and FP-Growth algorithms in pattern discovery

Background

Every day, we interact with systems that seem to "understand" our needs—whether it's video platforms suggesting what to watch next or shopping websites recommending products often bought together. Behind these intelligent suggestions lies a powerful idea: patterns hidden within data. What may initially appear as random actions—like buying bread, milk, or baby products—can, when observed across many transactions, reveal meaningful relationships. These patterns are not coincidences but valuable insights that help organizations improve services, personalize experiences, and make informed decisions. In today's data-driven world, uncovering such hidden connections is essential, and this is where Association Rule Learning plays a key role. It provides systematic methods to discover relationships within large datasets, transforming raw data into meaningful knowledge and helping us see the deeper stories data can tell.

SGOU

Keywords

Transactional data, relationships, itemset, support, confidence, lift, apriori, FP-growth

Discussion

3.3.1 Opening Context: Patterns Hidden in everyday decisions

People make many choices every day. They decide what to buy, what to watch, what to search for, which products to add to a cart, and even which services to use next. Each individual choice may seem ordinary. However, when thousands of such choices are collected as data, interesting patterns can begin to appear.

Consider the purchase records of a supermarket. One customer may buy bread and butter, another may buy bread, milk, and eggs, while another may purchase diapers along with baby wipes. Looking at a single transaction does not tell us much. But when a large number of transactions are examined together, a useful question arises:

Do certain items tend to occur together?

Suppose the records show that customers who frequently purchase bread also tend to purchase butter. This does not necessarily mean that buying bread causes a customer to buy butter. It simply indicates that the two items have a tendency to appear together in the observed transactions.

The same idea can be found in many other areas. An online shopping platform may observe that customers who purchase a smartphone often purchase a phone case or screen protector. A streaming platform may find that viewers who watch one type of crime series often watch another. An educational platform may observe that learners who complete a Python course frequently continue with courses related to data analysis.

Such patterns can be valuable for decision-making. A supermarket may place related products closer together, an online store may recommend complementary products, and a learning platform may suggest a suitable next course to a learner.

The important point is that these relationships are not identified simply by looking at one or two records. They are discovered by systematically analysing a large collection of transactions.

This is where Association Rule Learning becomes useful.

Association Rule Learning is a data mining technique used to discover relationships or associations among items or events in a dataset. It identifies patterns such as:

SGOU

If a customer buys X, the customer is likely to buy Y.

For example:

Bread → Butter

This rule suggests an association between bread and butter. To determine whether such a rule is actually useful, it is not enough to know that the two items occur together. We also need to measure how frequently the combination occurs, how often the consequent appears when the antecedent occurs, and whether the relationship is stronger than what might be expected by chance.

These aspects are measured using support, confidence, and lift.

Association Rule Learning is widely used in Market Basket Analysis, where transaction data is analysed to understand purchasing behaviour. Two important algorithms used to discover such associations are Apriori and FP-Growth.

In this unit, we will first understand the basic idea of Association Rule Learning and Market Basket Analysis. We will then learn how support, confidence, and lift are calculated and interpreted. Finally, we will examine how the Apriori and FP-Growth algorithms efficiently discover frequent patterns and association rules from transaction data.

3.3.2 Association Rule Learning

The opening example showed that data can contain relationships that are not immediately visible. The next step is to understand how such relationships can be represented and analysed systematically.

When a large collection of transactions is examined, some items may repeatedly appear together. The objective of Association Rule Learning is to identify such recurring relationships and express them in a form that can be easily understood and evaluated.

3.3.2.1 Concept of Association Rule Learning

Consider the purchase records of an online shopping platform. Thousands of customers may purchase different combinations of products. One customer may purchase a mobile phone and a protective case, another may purchase a laptop and a laptop bag, while another may purchase a mobile phone, earphones, and a charger.

Looking at any one transaction gives only limited information. However, when all the transactions are analysed together, a pattern may become visible. Suppose the records show that customers who purchase a mobile phone frequently purchase a protective case as well.

This relationship can be represented as:

Mobile Phone → Protective Case

The purpose of Association Rule Learning is to discover such relationships from data. It

SGOU

is a data mining technique that identifies items or events that tend to occur together and represents the observed relationship in the form of an association rule.

The idea is not restricted to shopping. A streaming platform may find that viewers who watch one particular series frequently watch another related series. An educational platform may observe that learners who complete a Python course often proceed to a data analysis course. In each case, the available records contain patterns that may not be obvious when individual records are viewed separately.

Association Rule Learning helps bring these patterns to the surface.

An association rule is generally represented as:

$X \rightarrow Y$

Here, X represents one item or a group of items, and Y represents another item or group of items that is associated with X.

For example:

Python Course → Data Analysis Course

The rule suggests that the two courses have an observed association in the learning records. It does not mean that every learner who completes Python will necessarily choose Data Analysis. It simply describes a tendency found in the data.

This distinction is important. Association Rule Learning identifies associations, not causes. If two items frequently occur together, we cannot automatically conclude that one item causes the other.

Once an association has been identified, the next question is: How exactly is such a rule structured?

3.3.2.2 Structure of an Association Rule

An association rule consists of two parts. The item or group of items on the left-hand side is called the antecedent, while the item or group of items on the right-hand side is called the consequent.

For example:

Mobile Phone → Protective Case

Here, Mobile Phone is the antecedent and Protective Case is the consequent.

The antecedent and consequent can also contain multiple items. Consider:

Mobile Phone, Screen Protector → Protective Case

In this case, the antecedent contains two items, Mobile Phone and Screen Protector, while Protective Case is the consequent.

SGOU

The rule can therefore be understood as:

When the antecedent occurs, the consequent tends to occur as well.

Notice that this is still an association rather than a cause-and-effect relationship.

There is also an important difference between:

Bread → Butter

and

Butter → Bread

Although both rules contain the same two items, they are not the same rule. The first asks whether butter tends to occur when bread occurs. The second asks whether bread tends to occur when butter occurs. Since the number of transactions containing bread and butter may differ, the strength of these two rules can also differ.

Association rules therefore provide a structured way of expressing relationships discovered in transaction data. But simply being able to write a rule does not tell us whether that rule is useful.

This leads to another important question: How can we determine whether a discovered association is strong, frequent, or meaningful?

3.3.2.3 Need for Evaluating Association Rules

A dataset can produce a very large number of possible association rules. Some may represent useful patterns, while others may occur only occasionally or may appear strong for reasons that are not immediately obvious.

For example, suppose an online store finds the rule:

Mobile Phone → Charger

If most mobile-phone customers also purchase a charger, the rule may be useful. However, suppose almost every customer of the store purchases a charger, whether or not they purchase a mobile phone. In that situation, a high percentage of mobile-phone buyers purchasing chargers does not necessarily indicate a special relationship between the two products.

Therefore, association rules need to be evaluated rather than accepted simply because they exist.

Three important measures are used to examine these relationships:

- **Support** helps determine how frequently an item or combination of items occurs in the entire dataset.
- **Confidence** helps determine how often the consequent occurs when the antecedent occurs.

SGOU

- **Lift** helps determine whether the observed association is stronger or weaker than what would be expected based on the individual occurrence of the items.

These measures look at the same association from different perspectives. Together, they help distinguish potentially useful rules from patterns that may be too rare or misleading.

We will examine these measures in detail after understanding Market Basket Analysis, where transaction data provides a practical setting for discovering such associations.

3.3.3 Market Basket Analysis

The previous section showed that association rules help identify relationships among items. But where do the data and transactions needed to discover these relationships come from? One of the most common settings is Market Basket Analysis.

Market Basket Analysis is a technique used to examine the products purchased together by customers. The term "market basket" refers to the collection of products that a customer places in a shopping basket during a particular purchase. By analysing many such baskets, patterns in customer purchasing behaviour can be identified.

Consider the billing records of a supermarket. Each customer may purchase a different combination of products. One bill may contain bread, milk, and eggs, while another may contain rice, vegetables, and cooking oil. A third customer may purchase bread, butter, and jam.

A few individual bills may not reveal much. However, suppose thousands of such bills are analysed and the records show that bread and butter frequently occur together. This observation can then be represented using an association rule:

Bread → Butter

The rule does not imply that every person buying bread will buy butter. Instead, it indicates that the two products have a recurring association in the transaction data.

3.3.3.1 Transaction Data

To perform Market Basket Analysis, the purchase records are represented as **transactions**. Each transaction contains the items purchased during one shopping event.

For example:

Transaction ID	Items Purchased
T1	Bread, Milk
T2	Bread, Butter
T3	Milk, Eggs
T4	Bread, Milk, Butter
T5	Bread, Butter

SGOU

Here, T1, T2, T3, T4, and T5 represent individual transactions.

The transaction ID is used to distinguish one purchase from another, while the items show what was included in that particular transaction.

The same idea applies to online shopping. Instead of a physical basket, a transaction may represent the products included in a customer's online order.

For example:

Order ID	Products
O1	Laptop, Mouse
O2	Laptop, Laptop Bag
O3	Mobile Phone, Earphones
O4	Laptop, Mouse, Laptop Bag
O5	Mobile Phone, Screen Protector

The transaction structure remains the same even though the shopping environment is different.

3.3.3.2 Items and Itemsets

Once the transactions are available, the individual products are treated as items. In the supermarket example, Bread, Milk, Butter, and Eggs are individual items. When two or more items are considered together, they form an itemset.

For example:

{Bread, Butter}

is a 2-itemset because it contains two items.

Similarly:

{Bread, Milk, Butter}

is a 3-itemset.

An itemset containing only one item, such as:

{Bread}

is called a 1-itemset.

The order of items in an itemset does not matter. Therefore:

{Bread, Butter}

and

{Butter, Bread}

SGOU

represent the same itemset.

This idea is important because Market Basket Analysis is concerned with whether items occur together within the same transaction, rather than the order in which they were purchased.

For example, if a customer buys bread first and butter later during the same shopping trip, the transaction still contains the itemset:

{Bread, Butter}

3.3.3.3 Discovering Patterns from Transactions

With the transactions and itemsets identified, the next task is to look for combinations that occur repeatedly.

Consider the following transaction dataset:

Transaction	Items
T1	Bread, Milk
T2	Bread, Butter
T3	Milk, Eggs
T4	Bread, Milk, Butter
T5	Bread, Butter
T6	Milk, Butter

Suppose we are interested in the itemset:

{Bread, Butter}

It appears in T2, T4, and T5. Therefore, the combination occurs in three transactions.

Similarly, the itemset:

{Milk, Butter}

appears in T4 and T6.

By examining such combinations across the dataset, it becomes possible to identify recurring purchasing patterns.

However, not every combination that appears in a transaction is necessarily important. A combination occurring only once may not provide enough evidence of a meaningful pattern, particularly when the dataset contains thousands of transactions.

This is why the frequency of an itemset becomes important.

SGOU

An itemset that occurs frequently enough according to a chosen threshold is called a frequent itemset. Frequent itemsets provide the foundation for discovering association rules.

For example, if a supermarket has 10,000 transactions and the combination {Bread, Butter} appears in 2,500 transactions, it may deserve further examination. On the other hand, a combination appearing in only one transaction may not be useful for generalising customer behaviour.

The measure used to determine how frequently an itemset occurs is called support.

Thus, Market Basket Analysis leads naturally to the next concept. Once transaction data has been collected and itemsets have been identified, we need a way to measure how common an itemset is in the complete dataset.

This is the purpose of support.

3.3.4 Support

The previous section showed how Market Basket Analysis uses transaction data to identify combinations of items that occur together. But simply finding that an itemset appears in some transactions is not enough. We need to know **how frequently it occurs in the entire dataset**.

This is where the concept of **support** becomes important.

3.3.4.1 Meaning of Support

Support measures the frequency with which an itemset appears in a transaction dataset. In simple terms, it tells us how common an itemset is among all the transactions.

Consider a supermarket with 10 transactions. Suppose the itemset:

{Bread, Butter}

appears in 4 of those transactions.

The support of the itemset is:

Support = 4 / 10 = 0.4

This can also be expressed as:

Support = 40%

Therefore, 40% of the transactions contain both bread and butter.

Support is calculated for an itemset, not for a rule alone. For example, if we are considering the rule:

Bread → Butter

SGOU

the support of the rule is based on the occurrence of the combined itemset:

{Bread, Butter}

in the complete transaction dataset.

A higher support means that the itemset appears in a larger proportion of transactions. A lower support means that it occurs less frequently.

3.3.4.2 Calculating Support

The general formula for support is:

Support(X) = Number of transactions containing X / Total number of transactions

Here, X represents the itemset whose frequency we want to measure.

Consider the following transactions:

Transaction	Items
T1	Bread, Milk
T2	Bread, Butter
T3	Milk, Eggs
T4	Bread, Milk, Butter
T5	Bread, Butter
T6	Milk, Butter

Suppose we want to find the support of:

{Bread, Butter}

The itemset appears in:

T2

T4

T5

Therefore, it appears in 3 transactions.

The total number of transactions is 6.

So:

Support({Bread, Butter}) = 3 / 6 = 0.5

SGOU

Therefore:

Support({Bread, Butter}) = 50%

This means that half of the transactions contain both bread and butter.

Now consider the itemset:

{Milk, Butter}

It appears in T4 and T6.

Therefore:

Support({Milk, Butter}) = 2 / 6 = 0.333

or approximately:

33.3%

The two itemsets therefore have different levels of support. The combination {Bread, Butter} occurs more frequently than {Milk, Butter} in this dataset.

3.3.4.3 Interpreting Support

Support gives us a measure of the prevalence of an itemset in the dataset.

Suppose an online store has 100,000 orders. If a particular product combination occurs in 25,000 orders, its support is:

25,000 / 100,000 = 0.25 = 25%

This indicates that the combination occurs in one-fourth of all orders.

Support is useful because it helps distinguish common patterns from extremely rare ones. In a large dataset, there may be a huge number of possible item combinations. Many of these combinations may occur only a few times.

For example, a combination of three highly unusual products might appear in only two transactions out of 100,000. Although the combination technically exists in the data, it may not provide enough evidence to represent a general purchasing pattern.

Support therefore helps focus attention on combinations that occur with sufficient frequency.

3.3.4.4 Minimum Support

When mining association rules, it is often necessary to decide how frequent an itemset should be before it is considered important. For this purpose, a minimum support threshold can be specified. Suppose a dataset contains 1,000 transactions and the minimum support is set to 10%.

SGOU

An itemset must therefore appear in at least:

10% of 1,000 = 100 transactions

to satisfy the minimum support requirement.

Consider three itemsets:

Itemset	Number of Transactions	Support	Meets 10% Minimum Support?
{Bread, Butter}	180	18%	Yes
{Milk, Eggs}	120	12%	Yes
{Juice, Cookies}	60	6%	No

The first two itemsets satisfy the minimum support requirement, while the third does not.

An itemset that satisfies the selected minimum support threshold is generally considered a frequent itemset.

The choice of minimum support depends on the dataset and the purpose of the analysis. A very high threshold may eliminate potentially useful patterns, while a very low threshold may produce a very large number of patterns that are difficult to analyse.

Support therefore helps answer one important question:

How frequently does this itemset occur in the dataset?

But frequency alone does not tell us how strongly one item is associated with another. An item may occur frequently simply because it is popular.

For example, suppose 80% of all customers purchase milk. A rule involving milk may have high support simply because milk is common. What we really want to know is whether the presence of another item makes milk more likely to occur.

This leads to the next measure, **confidence**, which examines how often the consequent occurs when the antecedent is present.

3.3.5 Confidence

The previous section showed that support tells us how frequently an itemset occurs in the entire dataset. However, frequency alone cannot tell us how reliable a particular association rule is.

Suppose a supermarket finds that milk appears in 80% of all transactions. A rule such as:

Bread → Milk

may also have a high occurrence simply because milk is already a very common product.

SGOU

Therefore, we need to ask a more specific question:

When a customer buys bread, how often does the customer also buy milk?

The measure used to answer this question is called **confidence**.

3.3.5.1 Meaning of Confidence

Confidence measures how frequently the consequent of an association rule occurs in transactions that contain the antecedent.

In simple terms, confidence tells us how often Y occurs when X occurs in a rule of the form:

$X \rightarrow Y$

For example, consider the rule:

Bread → Butter

If 100 transactions contain bread and 70 of those transactions also contain butter, the confidence of the rule is:

70 / 100 = 0.70

Therefore, the confidence is 70%.

This means that among the transactions containing bread, 70% also contain butter.

A higher confidence indicates that the consequent occurs more frequently among transactions containing the antecedent. However, confidence should not be interpreted as proof that the antecedent causes the consequent. It describes a pattern observed in the transaction data.

3.3.5.2 Calculating Confidence

The formula for confidence is:

Confidence($X \rightarrow Y$) = Support($X \cup Y$) / Support(X)

The same idea can be expressed directly using transaction counts:

Confidence($X \rightarrow Y$) = Number of transactions containing both X and Y / Number of transactions containing X

SGOU

Consider the following transaction dataset:

Transaction	Items
T1	Bread, Milk
T2	Bread, Butter
T3	Milk, Eggs
T4	Bread, Milk, Butter
T5	Bread, Butter
T6	Milk, Butter

Consider the rule:

Bread → Butter

First, identify the transactions containing bread: T1, T2, T4 and T5

Therefore, bread occurs in 4 transactions.

Now identify the transactions containing both bread and butter:

T2

T4

T5

Therefore, the combination occurs in 3 transactions.

The confidence is:

Confidence(Bread → Butter) = 3 / 4

= 0.75

= 75%

Therefore, the confidence of the rule is 75%.

This means that 75% of the transactions containing bread also contain butter.

3.3.5.3 Interpreting Confidence

Confidence can be understood as a conditional measure. It focuses only on the transactions in which the antecedent occurs.

For example, suppose an online store has 1,000 orders. Among these:

200 orders contain a laptop.

SGOU

150 of those 200 orders also contain a laptop bag.

For the rule:

Laptop → Laptop Bag

the confidence is:

150 / 200 = 0.75

or 75%.

The interpretation is straightforward: 75% of the orders containing a laptop also contain a laptop bag.

Notice that the total number of orders is not used directly in this calculation. Confidence focuses on the transactions containing the antecedent.

This is different from support.

If the same 150 orders contain both a laptop and a laptop bag, the support of the itemset is:

150 / 1,000 = 15%

while the confidence of the rule is:

150 / 200 = 75%

The two measures therefore describe different properties of the same data.

Support asks: How common is the combination in all transactions?

Confidence asks: When X occurs, how often does Y occur?

Both measures are useful, but confidence has an important limitation.

3.3.5.4 Limitation of Confidence

A high confidence value does not always indicate a meaningful association.

Consider a supermarket where 90% of all customers purchase milk. Now suppose 90% of customers who purchase bread also purchase milk.

The rule:

Bread → Milk

would have a confidence of 90%.

At first, this may appear to be a very strong relationship. But milk is already purchased by 90% of all customers. Therefore, the high confidence may simply be a result of milk being extremely common rather than a special association between bread and milk.

This shows why confidence should not be considered in isolation.

SGOU

We need a measure that compares the observed association with what would be expected from the individual frequencies of the two items.

That measure is lift. Lift helps answer a deeper question:

Does X actually make Y more likely to occur, or does Y simply occur frequently on its own?

The next section introduces lift and shows how it provides this additional insight.

3.3.6 Lift

The previous section showed that confidence tells us how often the consequent occurs when the antecedent is present. However, confidence alone may sometimes make an association appear stronger than it really is.

Suppose 90% of the customers of an online store purchase a particular type of charging cable. If 90% of customers who purchase mobile phones also purchase that cable, the confidence of the rule:

Mobile Phone → Charging Cable

would be 90%.

At first, this may look like a very strong association. But the charging cable is already purchased by 90% of all customers. Therefore, the high confidence may not indicate a special relationship between mobile phones and charging cables.

This leads to an important question:

Does the consequent occur with the antecedent more often than we would expect based on how common the consequent is overall?

The measure used to answer this question is called **lift**.

3.3.6.1 Meaning of Lift

Lift measures the strength of an association by comparing the observed occurrence of the antecedent and consequent with the occurrence that would be expected if the two items were independent.

In simpler terms, lift tells us whether the presence of X is associated with an increased or decreased likelihood of Y.

For an association rule:

$X \rightarrow Y$

the lift is calculated as:

$\text{Lift}(X \rightarrow Y) = \text{Confidence}(X \rightarrow Y) / \text{Support}(Y)$

It can also be calculated directly using support:

SGOU

$\text{Lift}(X \rightarrow Y) = \text{Support}(X \cup Y) / [\text{Support}(X) \times \text{Support}(Y)]$

The value of lift is interpreted relative to 1.

Lift > 1: X and Y have a positive association.

Lift = 1: X and Y are approximately independent.

Lift < 1: X and Y have a negative association.

The important point is that lift does not simply measure how frequently two items occur together. It compares their observed co-occurrence with what would be expected from their individual frequencies.

3.3.6.2 Calculating Lift

Consider the following transaction dataset:

Transaction	Items
T1	Bread, Milk
T2	Bread, Butter
T3	Milk, Eggs
T4	Bread, Milk, Butter
T5	Bread, Butter
T6	Milk, Butter
T7	Bread, Milk
T8	Milk, Butter
T9	Bread, Butter
T10	Milk, Eggs

Consider the rule:

Bread → Butter

First, identify the transactions containing bread.

Bread occurs in:

T1, T2, T4, T5, T7, T9

Therefore:

Support(Bread) = 6 / 10 = 0.6

SGOU

Now identify the transactions containing butter.

Butter occurs in:

T2, T4, T5, T6, T8, T9

Therefore:

Support(Butter) = 6 / 10 = 0.6

The combination of bread and butter occurs in:

T2, T4, T5, T9

Therefore:

Support(Bread, Butter) = 4 / 10 = 0.4

The confidence of the rule is:

Confidence(Bread → Butter) = 0.4 / 0.6

≈ 0.667

Now calculate lift:

Lift(Bread → Butter) = 0.667 / 0.6

≈ 1.11

Therefore, the lift of the rule is approximately **1.11**.

Since the lift is greater than 1, the data indicates a positive association between bread and butter.

3.3.6.3 Interpreting Lift

The value of lift provides useful information about the nature of the relationship between the antecedent and consequent.

When:

Lift > 1

the two items occur together more often than would be expected if they were independent. This indicates a positive association.

For example, a lift of 1.5 suggests that the consequent occurs with the antecedent at a rate higher than would be expected based on the overall frequency of the consequent.

When:

Lift = 1

SGOU

the occurrence of the antecedent does not provide additional information about the consequent. The two items are approximately independent in the dataset.

When:

Lift < 1

the two items occur together less often than would be expected if they were independent. This indicates a negative association.

For example, consider two products that customers rarely purchase together even though each product is individually popular. Their lift may be less than 1, indicating that the presence of one is associated with a lower likelihood of the other.

Therefore, lift provides information that support and confidence cannot provide on their own.

With these measures understood, we now have the foundation needed to study how association rule mining is performed on large transaction datasets.

The next question is:

How can a computer efficiently discover frequent itemsets from thousands or millions of transactions?

One of the earliest and most widely used approaches for this purpose is the Apriori algorithm.

3.3.7 Apriori Algorithm

The previous sections explained how association rules are evaluated using support, confidence, and lift. But these measures can only be applied after useful patterns have been discovered from the transaction data.

When a dataset contains thousands of transactions and many different items, manually checking all possible combinations is not practical. For example, with just a few items, several pairs and groups can be formed. As the number of items increases, the number of possible combinations grows rapidly.

An algorithm is therefore needed to systematically search the transaction database and identify itemsets that occur frequently enough to be considered useful. One of the earliest and most widely known algorithms for this purpose is the Apriori algorithm.

The Apriori algorithm is a frequent itemset mining algorithm used to discover item combinations that satisfy a specified minimum support threshold.

Its basic approach is to start with individual items and gradually form larger itemsets. At each stage, the support of the candidate itemsets is examined, and only those that satisfy the minimum support requirement are retained.

For example, suppose a transaction dataset contains:

Bread, Milk, Butter, Eggs

SGOU

The algorithm first examines individual items:

{Bread}, {Milk}, {Butter}, {Eggs}

The items that satisfy the minimum support requirement are retained. These frequent 1-itemsets are then used to form possible 2-itemsets, such as:

{Bread, Milk}

{Bread, Butter}

The process can continue to larger itemsets if the required conditions are satisfied.

The general progression is:

1-itemsets → 2-itemsets → 3-itemsets → ...

The important feature of Apriori is that it does not treat every possible combination equally. Information about smaller itemsets is used to decide whether larger combinations are worth examining.

This idea is based on the Apriori principle.

3.3.7.1 Apriori Principle

The Apriori algorithm is based on the observation that:

If an itemset is frequent, all of its non-empty subsets must also be frequent.

This also gives us a useful way to eliminate unnecessary candidates:

If an itemset is not frequent, none of its supersets can be frequent.

Consider the itemset:

{Bread, Milk, Butter}

Its subsets include:

{Bread, Milk}

{Bread, Butter}

{Milk, Butter}

and the individual items.

Suppose:

{Bread, Milk}

does not satisfy the minimum support requirement.

Any larger itemset containing both bread and milk, such as:

{Bread, Milk, Butter}

SGOU

cannot satisfy the minimum support requirement either.

The reason is simple. Every transaction containing bread, milk, and butter must also contain bread and milk. Therefore, the larger itemset cannot occur more frequently than its subset.

This property allows Apriori to prune unnecessary candidates.

For example, suppose a particular 2-itemset is found to be infrequent. There is no need to generate or test 3-itemsets containing that pair because they cannot become frequent.

The Apriori principle therefore helps reduce the search space and forms the foundation of the algorithm.

With this principle in place, the working process of Apriori can be understood step by step.

3.3.7.2 Steps of the Apriori Algorithm

The Apriori algorithm generally follows a repeated process of candidate generation, support counting, and pruning.

Step 1: Set the minimum support

A minimum support threshold is selected.

For example:

Minimum Support = 60%

Only itemsets appearing in at least 60% of the transactions will be retained.

Step 2: Find frequent 1-itemsets

The algorithm scans the transactions and counts how many transactions contain each individual item.

Items satisfying the minimum support become the set of frequent 1-itemsets.

Step 3: Generate candidate itemsets

The frequent itemsets from the previous stage are combined to generate larger candidate itemsets.

For example, frequent 1-itemsets can be combined to create candidate 2-itemsets.

Step 4: Apply pruning

Candidates containing an infrequent subset are removed using the Apriori principle.

This prevents the algorithm from spending time calculating the support of itemsets that cannot possibly become frequent.

SGOU

Step 5: Calculate support

The remaining candidate itemsets are checked against the transaction database, and their support values are calculated.

Step 6: Select frequent itemsets

Candidates whose support meets or exceeds the minimum support threshold are retained as frequent itemsets.

Step 7: Repeat

The frequent itemsets are used to generate larger candidates, and the same process continues until no further frequent itemsets can be generated.

Once the frequent itemsets have been discovered, they can be used to form association rules. These rules can then be evaluated using measures such as confidence and lift.

The process becomes clearer when it is applied to a small transaction dataset.

3.3.7.3 Worked Example of Apriori

Consider the following five transactions:

Transaction	Items
T1	Bread, Milk, Butter
T2	Bread, Milk
T3	Bread, Butter
T4	Milk, Butter
T5	Bread, Milk, Butter

Let the minimum support be **60%**.

Since there are five transactions, an itemset must occur in at least:

60% of 5 = 3 transactions

to be considered frequent.

Step 1: Find Frequent 1-Itemsets

Count the occurrence of each individual item.

Item	Number of Transactions	Support
Bread	4	80%
Milk	4	80%
Butter	4	80%

All three items satisfy the 60% minimum support.

SGOU

Therefore:

L_1 = {Bread}, {Milk}, {Butter}

where L_1 represents the set of frequent 1-itemsets.

Step 2: Generate Candidate 2-Itemsets

The frequent 1-itemsets are combined to form:

{Bread, Milk}

{Bread, Butter}

{Milk, Butter}

Now their support is calculated.

Candidate Itemset	Number of Transactions	Support
{Bread, Milk}	3	60%
{Bread, Butter}	3	60%
{Milk, Butter}	3	60%

All three candidates satisfy the minimum support.

Therefore:

L_2 = {Bread, Milk}, {Bread, Butter}, {Milk, Butter}

Step 3: Generate Candidate 3-Itemsets

The frequent 2-itemsets can be combined to form:

{Bread, Milk, Butter}

This itemset appears in:

T1

T5

Therefore:

Support = 2 / 5 = 40%

Since 40% is less than the required 60%, the itemset is not frequent.

It is therefore discarded.

No larger frequent itemset can be generated from the available frequent 2-itemsets.

The final frequent itemsets are:

SGOU

Frequent 1-itemsets

{Bread}, {Milk}, {Butter}

Frequent 2-itemsets

{Bread, Milk}, {Bread, Butter}, {Milk, Butter}

The example demonstrates how Apriori gradually expands the search while eliminating combinations that do not meet the minimum support requirement.

3.3.7.4 Limitations of Apriori

The algorithm may generate a large number of candidate itemsets when the dataset contains many different items. The number of possible combinations can grow rapidly as the item count increases.

It may also require multiple scans of the transaction database to calculate support for candidates at different levels. These repeated scans can become expensive for large datasets.

The problem becomes more significant when the minimum support threshold is set very low. A low threshold may result in a large number of frequent itemsets, increasing both the number of candidates and the computational effort required to process them.

Thus, Apriori provides a clear and systematic method for frequent itemset mining, but its candidate-generation process can become inefficient for large and complex datasets.

This limitation motivates the need for another approach that can discover frequent patterns without explicitly generating a large number of candidate itemsets. FP-Growth addresses this problem by using a compact tree-based representation of the transaction data.

3.3.8 FP-Growth Algorithm

The Apriori algorithm reduces the search space by using the Apriori principle, but it still relies on generating candidate itemsets and repeatedly checking their support. When the transaction database is large or contains many different items, this process can become computationally expensive.

A different approach is therefore needed—one that can discover frequent patterns without explicitly generating a large number of candidate itemsets.

The **FP-Growth (Frequent Pattern Growth) algorithm** addresses this problem by using a compact tree structure called an **FP-Tree**. Instead of generating candidates level by level, FP-Growth compresses the transaction data and mines frequent patterns directly from this structure.

3.3.8.1 Concept of FP-Growth

FP-Growth is a frequent itemset mining algorithm used to discover frequent patterns from transaction data without explicitly generating candidate itemsets.

SGOU

The basic idea is to represent the transaction database in a compact form called an FP-Tree. Transactions that contain common items can share portions of the same tree. This reduces the need to repeatedly generate and test individual candidate combinations.

Consider an online shopping platform where customers purchase products such as mobile phones, phone cases, screen protectors, and earphones. Different orders may contain some of the same products.

For example:

Order	Products
O1	Mobile Phone, Phone Case, Screen Protector
O2	Mobile Phone, Phone Case
O3	Mobile Phone, Screen Protector
O4	Phone Case, Screen Protector
O5	Mobile Phone, Phone Case, Screen Protector

Several orders contain common products. Instead of treating every order as an entirely separate combination, FP-Growth arranges the products in a common order and stores their shared portions using a tree structure.

The overall idea can be represented as:

Transaction Database → FP-Tree → Frequent Patterns

This is the main difference from Apriori. Apriori repeatedly generates candidate itemsets, whereas FP-Growth compresses the transaction data and mines the resulting structure.

To understand this process, we first need to examine the FP-Tree.

3.3.8.2 FP-Tree

An FP-Tree, or Frequent Pattern Tree, is a compact tree-based representation of transaction data.

The tree stores items as nodes. When different transactions contain the same sequence of frequently occurring items, those transactions can share the corresponding path in the tree.

Consider two online orders:

O1: Mobile Phone, Phone Case, Screen Protector

O2: Mobile Phone, Phone Case

If the products are arranged in the same order, both orders can share the path:

SGOU

Mobile Phone → Phone Case

The first order continues to Screen Protector, while the second stops at Phone Case.

Instead of storing the common products separately, the tree stores the shared path once and uses counts to indicate how many transactions pass through each node.

An FP-Tree generally contains:

- a root node, which represents the starting point;
- item nodes, representing individual products;
- counts, indicating how many transactions pass through each node;
- branches, representing different transaction paths; and
- a header table, which helps locate nodes belonging to the same item.

The header table is particularly useful during mining because the algorithm needs to locate occurrences of a particular item in different parts of the tree.

The FP-Tree therefore provides a compact representation of the original transaction database.

3.3.8.3 Construction of an FP-Tree

The FP-Tree is constructed after identifying the frequency of the items in the transaction database.

Consider the following online orders:

Order	Products
O1	Mobile Phone, Phone Case, Screen Protector
O2	Mobile Phone, Phone Case
O3	Mobile Phone, Screen Protector
O4	Phone Case, Screen Protector
O5	Mobile Phone, Phone Case, Screen Protector

First, count the occurrence of each product.

Product	Frequency
Mobile Phone	4
Phone Case	4
Screen Protector	4

SGOU

In this example, all three products have the same frequency. Therefore, for illustration, let us use the order:

Mobile Phone → Phone Case → Screen Protector

Each order is then rearranged according to this common order.

Order	Ordered Products
O1	Mobile Phone, Phone Case, Screen Protector
O2	Mobile Phone, Phone Case
O3	Mobile Phone, Screen Protector
O4	Phone Case, Screen Protector
O5	Mobile Phone, Phone Case, Screen Protector

The first order creates the initial path:

Root → Mobile Phone → Phone Case → Screen Protector

When O2 is inserted, it shares the existing Mobile Phone and Phone Case nodes:

Root → Mobile Phone → Phone Case

The counts of these shared nodes are increased.

When O3 is inserted, it shares the Mobile Phone node but follows a different branch because Phone Case is absent:

Root → Mobile Phone → Screen Protector

O4 does not contain Mobile Phone, so it creates a path beginning with Phone Case:

Root → Phone Case → Screen Protector

Finally, O5 follows the existing:

Root → Mobile Phone → Phone Case → Screen Protector

path, increasing the relevant counts again.

SGOU

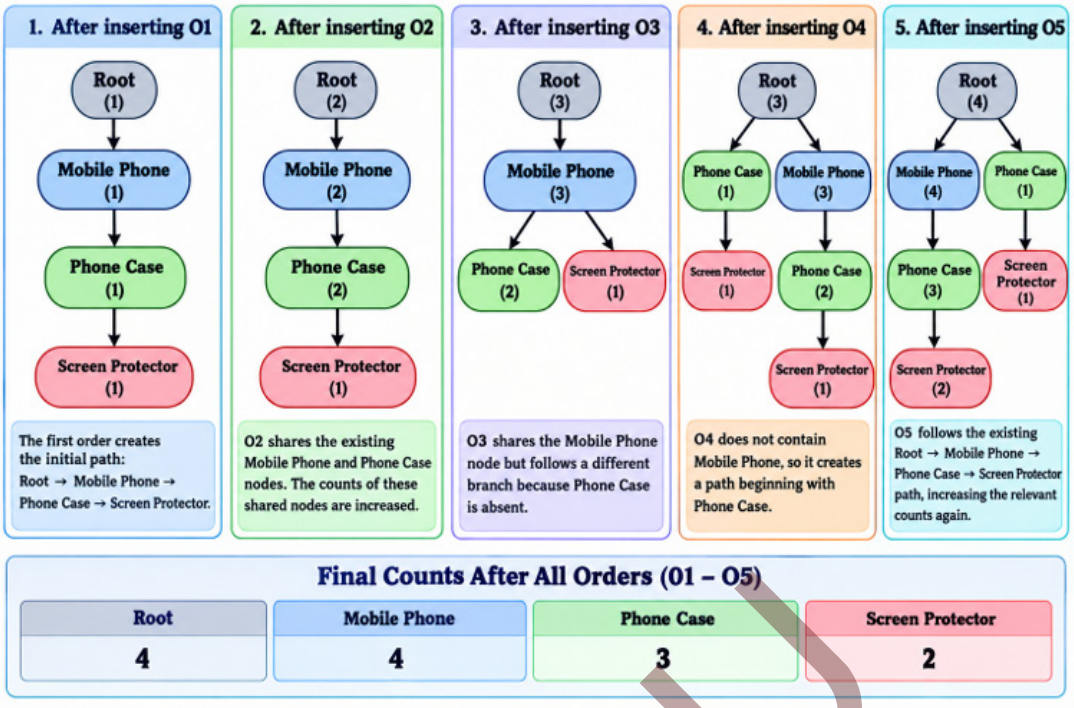


Fig.3.3.1 Step-by-step construction of the Prefix Tree

The important idea is that common portions of orders are shared rather than stored repeatedly.

After the tree has been constructed, the header table maintains links to nodes representing the same product. This allows FP-Growth to locate relevant portions of the tree efficiently.

The resulting structure contains the information required to mine frequent patterns without repeatedly scanning the original transaction database for every candidate.

3.3.8.4 Steps of the FP-Growth Algorithm

The FP-Growth process can be understood through the following steps.

Step 1: Scan the transaction database

The algorithm first scans the database and counts the frequency of each product.

Step 2: Remove infrequent products

Products whose support is below the minimum support threshold are removed.

This prevents products that cannot contribute to a frequent itemset from being included in the tree.

Step 3: Order the frequent products

The remaining products in each transaction are arranged according to a common frequency-based order.

SGOU

Using the same order for all transactions makes it possible for common portions of transactions to share paths.

Step 4: Construct the FP-Tree

The ordered transactions are inserted into the tree. When an order shares a path with an existing order, the corresponding nodes are reused and their counts are updated.

Step 5: Create the header table

The header table records the frequent products and links together the tree nodes representing each product.

Step 6: Mine the FP-Tree

The tree is examined to identify frequent patterns. The algorithm considers products and traces the paths associated with their occurrences.

The relevant paths form a **conditional pattern base** for the product being examined.

Step 7: Construct conditional FP-Trees

A conditional pattern base can be used to construct a smaller **conditional FP-Tree**.

The conditional tree is then mined to identify frequent patterns associated with the selected product.

This process continues recursively until the required frequent patterns have been discovered.

The major difference from Apriori is therefore clear:

Apriori: Generate candidates → Calculate support → Prune candidates

FP-Growth: Build FP-Tree → Mine the tree → Discover frequent patterns

3.3.8.5 Worked Example of FP-Growth

Consider the following five online orders:

Order	Products
O1	Laptop, Laptop Bag, Wireless Mouse
O2	Laptop, Laptop Bag
O3	Laptop, Wireless Mouse
O4	Laptop Bag, Wireless Mouse
O5	Laptop, Laptop Bag, Wireless Mouse

Let the minimum support be 60%.

SGOU

Since there are five orders, a product or product combination must occur in at least:

60% of 5 = 3 orders

to be considered frequent.

First, count the individual products:

Product	Frequency	Support
Laptop	4	80%
Laptop Bag	4	80%
Wireless Mouse	4	80%

All three products satisfy the minimum support threshold.

For simplicity, use the common order:

Laptop → Laptop Bag → Wireless Mouse

The orders are therefore represented as:

Order	Ordered Products
O1	Laptop, Laptop Bag, Wireless Mouse
O2	Laptop, Laptop Bag
O3	Laptop, Wireless Mouse
O4	Laptop Bag, Wireless Mouse
O5	Laptop, Laptop Bag, Wireless Mouse

The first order creates:

Root → Laptop → Laptop Bag → Wireless Mouse

O2 follows:

Root → Laptop → Laptop Bag

and shares the existing Laptop and Laptop Bag nodes.

O3 follows:

Root → Laptop → Wireless Mouse

so it shares Laptop but creates a separate branch for Wireless Mouse.

O4 does not contain Laptop, so it creates:

SGOU

Root → Laptop Bag → Wireless Mouse

O5 follows the existing:

Root → Laptop → Laptop Bag → Wireless Mouse

path, increasing the corresponding counts.

The tree now contains shared paths representing the different online orders.

The frequent combinations can then be identified from this structure.

For example, the itemset:

{Laptop, Laptop Bag}

occurs in O1, O2, and O5.

Therefore:

Support({Laptop, Laptop Bag}) = 3 / 5 = 60%

It satisfies the minimum support requirement.

Similarly:

{Laptop, Wireless Mouse}

occurs in O1, O3, and O5.

Therefore:

Support({Laptop, Wireless Mouse}) = 3 / 5 = 60%

The itemset:

{Laptop Bag, Wireless Mouse}

occurs in O1, O4, and O5.

Therefore:

Support({Laptop Bag, Wireless Mouse}) = 3 / 5 = 60%

The three 2-itemsets are therefore frequent.

The three-product combination:

{Laptop, Laptop Bag, Wireless Mouse}

occurs in O1 and O5.

Therefore:

Support({Laptop, Laptop Bag, Wireless Mouse}) = 2 / 5 = 40%

Since 40% is below the required 60%, this combination is not a frequent itemset.

SGOU

The example demonstrates how FP-Growth can use a compact tree representation to identify frequent product combinations without explicitly generating every possible candidate itemset.

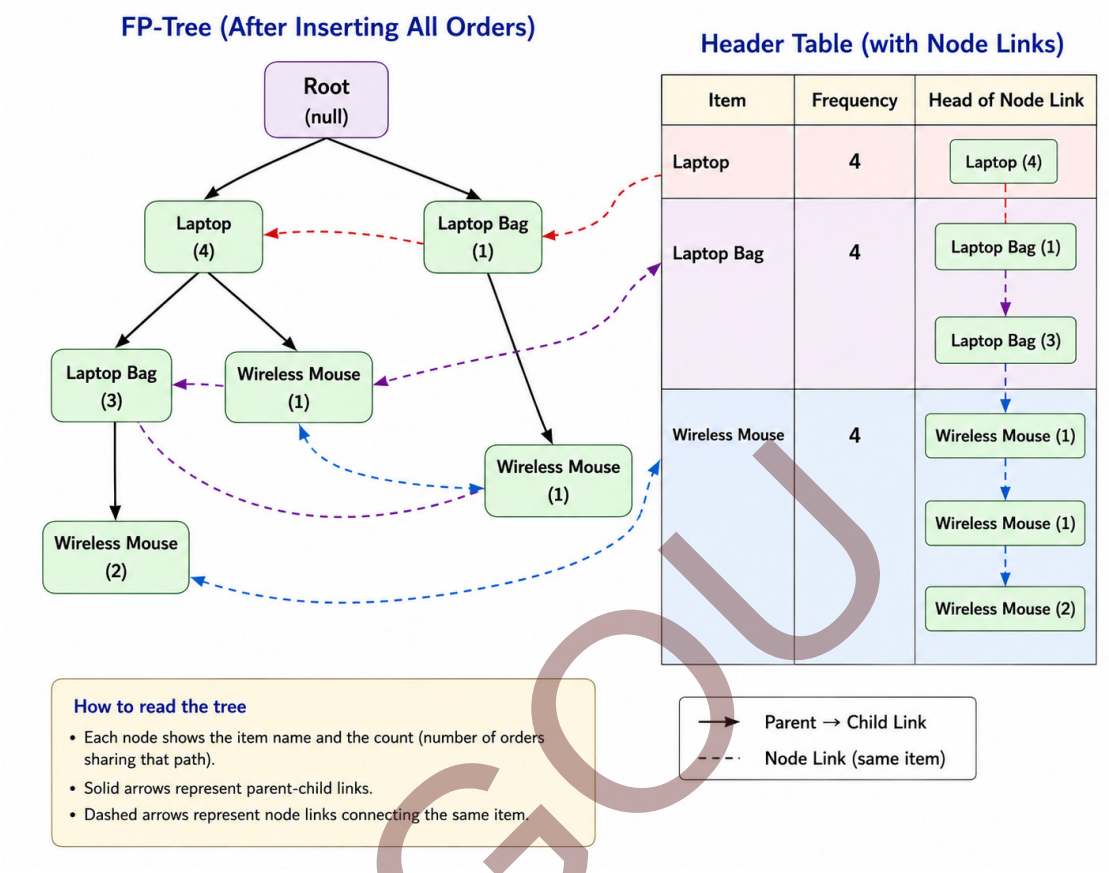


Fig.3.3.2 Illustration of FP-tree Construction

3.3.8.6 Advantages and Limitations of FP-Growth

FP-Growth has several advantages, particularly when working with large transaction datasets.

A major advantage is that it **does not explicitly generate candidate itemsets**. This avoids the potentially large candidate-generation process used by Apriori.

It can also reduce the number of scans of the original transaction database. After the FP-Tree has been constructed, frequent patterns can be mined from the tree structure.

Another advantage is its ability to **compress transaction data**. Orders with common products can share tree paths, reducing repeated representation of the same information.

These features can make FP-Growth more efficient than Apriori for large and dense transaction datasets.

However, FP-Growth also has limitations.

Constructing and maintaining an FP-Tree can be more complex than the straightforward candidate-generation process of Apriori. The tree structure, header table, and

SGOU

conditional pattern bases require a deeper understanding.

The memory requirement can also become significant when the transaction data contains many distinct products or when transactions have relatively few common items and therefore cannot share much of the tree structure.

Therefore, FP-Growth provides a powerful alternative to Apriori, but its implementation can be more complex.

Apriori and FP-Growth ultimately have the same goal—discovering frequent itemsets—but they approach the task differently. Apriori relies on candidate generation and pruning, whereas FP-Growth relies on compact tree representation and pattern growth.

Summerised Overview

Association Rule Learning is a data mining technique used to discover hidden patterns and relationships among items in large datasets. Although individual decisions such as shopping or browsing may seem random, analyzing many such transactions reveals meaningful associations. A common example is Market Basket Analysis, where customer purchase data is examined to identify items that are frequently bought together. These relationships are expressed as rules of the form $X \rightarrow Y$, meaning if X occurs, Y is likely to occur, though this indicates association rather than cause and effect.

To evaluate the usefulness of these rules, three key measures are used: support, confidence, and lift. Support indicates how frequently an itemset appears in the dataset, helping identify common patterns. Confidence measures how reliable a rule is by showing how often Y occurs when X occurs. Lift goes a step further by determining whether the relationship between X and Y is truly meaningful or simply due to chance. Together, these measures help in selecting strong and useful association rules.

Before generating rules, the process begins with frequent itemset mining, which identifies groups of items that commonly occur together. One of the most widely used algorithms for this purpose is the Apriori algorithm, which works by generating candidate itemsets and eliminating those that do not meet a minimum support threshold, based on the principle that all subsets of a frequent itemset must also be frequent. While Apriori is simple and effective for small datasets, it can be inefficient for large datasets due to repeated database scans and large candidate generation. Overall, Association Rule Learning plays a significant role in applications such as recommendation systems, retail analysis, and decision-making processes.

SGOU

Assignments

1. Explain the concept of Association Rule Learning and discuss its importance in real-world applications with suitable examples.

2. Describe the measures **support, confidence, and lift**. Explain how each measure helps in evaluating association rules.

3. Explain the working of the **Apriori algorithm** with a suitable example. Discuss its advantages and limitations.

4. Consider the following transaction dataset:

   T1: Bread, Butter, Milk
   T2: Bread, Diaper, Eggs
   T3: Milk, Diaper, Bread, Butter
   T4: Bread, Milk, Diaper
   T5: Butter, Milk

   a) Calculate support for {Bread} and {Bread, Butter}
   b) Calculate confidence for the rule: Bread → Butter
   c) Calculate lift for the rule: Bread → Butter
   d) Interpret the results

5. A retail store wants to improve product recommendations.

   Based on your understanding of Apriori and FP-Growth algorithms, analyze:

- Which algorithm is more suitable for large datasets and why?
- How does the difference in approach affect performance?

SGOU

Suggested Reading

1. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)* (pp. 487–499). Morgan Kaufmann.

2. Aggarwal, C. C. (2016). *Recommender systems: The textbook*. Springer.

3. Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press.

4. Berry, M. J. A., & Linoff, G. S. (2004). *Data mining techniques: For marketing, sales, and customer relationship management*. Wiley.

5. Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques* (4th ed.). Morgan Kaufmann.

Reference

1. Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

2. Tan, P.-N., Steinbach, M., & Kumar, V. (2018). *Introduction to data mining*. Pearson.

3. Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: An introduction to data mining* (2nd ed.). Wiley.

4. Zaki, M. J., & Meira, W. (2014). *Data mining and analysis: Fundamental concepts and algorithms*. Cambridge University Press.

5. Berson, A., & Smith, S. J. (1997). *Data warehousing, data mining, and OLAP*. McGraw-Hill.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

4 UNIT Reinforcement Learning Basics

Learning Outcomes

After completion of this unit, the learner will be able to:

- understand the basic concept of Reinforcement Learning and its trial-and-error approach.
- explain the components of an agent, environment, state, action, and reward in a reinforcement learning system.
- summarize the concept of Markov Decision Process and its key elements.
- describe the difference between Q-Learning and SARSA in decision-making.

SGOU

Background

Have you ever learned something simply by trying again and again—without anyone teaching you the exact steps? Think about how a child learns to ride a bicycle or how you improve while playing a new game. At first, mistakes are common. But slowly, by observing what works and what does not, better decisions are made. This natural learning process forms the foundation of Reinforcement Learning.

In today's world, machines are expected to make intelligent decisions—whether it is recommending a video, guiding a robot, or helping in medical treatments. But how can a machine learn without explicit instructions? The answer lies in allowing it to interact with its environment and learn from feedback, just like humans do.

Reinforcement Learning is a powerful approach in machine learning where an agent learns by performing actions and receiving rewards or penalties. Over time, the system improves its decisions to achieve the best possible outcome. To formally understand this process, concepts like the Markov Decision Process are used, which help in modeling decision-making situations step by step.

In this section, you will explore how learning happens through interaction, how decisions are evaluated, and how algorithms such as Q-Learning and SARSA help machines improve over time. By the end, you will begin to see how machines can "learn from experience" just like we do in everyday life.

Let us now begin this journey into learning without instructions.

Keywords

Agent, environment, state, action, reward, MDP, Q-learning

Discussion

3.4.1 Introduction to Reinforcement Learning

Learning does not always happen by being shown the correct answer. Sometimes, learning takes place through experience. A child learning to ride a bicycle, for example, tries different movements, loses balance, adjusts the movement and tries again. Gradually, the child learns which movements help maintain balance. The learning happens through repeated interaction and feedback.

Reinforcement Learning (RL) is a type of machine learning in which a system learns to make decisions by interacting with an environment. Instead of being given the correct answer for every situation, the system performs actions and receives feedback in the form of rewards or penalties. Based on this feedback, it gradually learns which actions are more useful.

The system that learns is called the agent, and the surroundings with which it interacts are called the environment. At any particular moment, the agent is in a particular state. It chooses an action, and the environment responds by providing a reward and moving the agent to another state.

The basic interaction can therefore be represented as:

State → Action → Reward → New State → Action → Reward → ...

Consider a robot that has to move from one room to another. The robot acts as the agent, while the building and everything inside it form the environment. The robot's current location represents its state. Moving forward, turning left or turning right are possible actions. If the robot reaches the destination, it may receive a large positive reward. If it hits an obstacle, it may receive a negative reward. After repeated attempts, the robot can learn a route that produces a high total reward.

SGOU

This makes reinforcement learning particularly useful for problems where a sequence of decisions has to be made. The result of one action can affect the next situation, and therefore the agent has to learn not only which individual action is good but also which sequence of actions can lead to a desirable outcome.

3.4.1.1 Reinforcement Learning and other t`ypes of Machine Learning

Reinforcement learning differs from supervised learning and unsupervised learning. In supervised learning, the model learns from labelled examples. For instance, if a system is trained to identify cats and dogs, the training data may contain images along with their correct labels. The model learns the relationship between the input and the expected output.

In reinforcement learning, the correct action is generally not supplied directly. Instead, the agent receives feedback after taking an action. A good action may result in a positive reward, while an undesirable action may result in a negative reward. Through repeated interaction, the agent learns a strategy for making better decisions.

For example, consider a computer program learning to play a game. It may not initially know which moves are good. It tries different moves and observes the results. Winning a game may produce a large positive reward, while losing may produce a negative reward. After playing many games, the program can learn which decisions are more likely to lead to success.

Thus, reinforcement learning can be viewed as learning through interaction, experience and feedback.

3.4.1.2 Basic Components of Reinforcement Learning

A reinforcement learning system consists of several important components. The most fundamental ones are the agent, environment, state, action and reward. Understanding these components makes it much easier to understand algorithms such as Q-Learning and SARSA.

Agent

The agent is the learner or decision-maker in a reinforcement learning system. It observes the current situation, selects an action and learns from the result of that action.

A robot navigating through a building, a computer program playing a game and an automated system controlling traffic signals can all act as agents.

The agent does not necessarily know the best action when learning begins. It gradually develops knowledge by interacting with the environment.

Environment

The environment is the world or system with which the agent interacts. It provides the current state to the agent and responds when the agent performs an action.

SGOU

For a game-playing agent, the game board and the rules of the game form the environment. For a robot, the physical surroundings form the environment. For a traffic-control system, the road network, vehicles and traffic conditions can form part of the environment.

The environment is therefore the source of the feedback from which the agent learns.

State

A state represents the current situation of the agent or the environment.

For a robot moving through a simple grid, its current position can represent its state. In a game, the position of the player, the position of other objects and other relevant information together may form the state.

For example, if a robot is currently at location B, we can represent its current state as:

State = B

The state provides the information needed by the agent to make a decision.

Action

An action is a decision or operation that the agent can perform in a particular state. The available actions depend on the environment.

A robot may be able to move forward, move backward, turn left or turn right. A game-playing agent may be able to move a piece, attack an opponent or defend a position.

The agent selects an action based on its current knowledge and strategy.

Reward

A reward is numerical feedback received by the agent after performing an action. It indicates how desirable or undesirable the result was.

A positive reward generally indicates a desirable outcome, while a negative reward indicates an undesirable outcome. A reward of zero may indicate that the action did not produce any immediate benefit or penalty.

For example, a delivery robot could receive a reward of +10 for successfully delivering a package, −5 for taking an inefficient route and −20 for failing to complete the delivery.

The agent uses these rewards to improve its future decisions.

3.4.1.3 The Agent–Environment Interaction

The relationship between the agent and the environment can be understood as a continuous cycle. The agent observes the current state and chooses an action. The environment then responds to the action by providing a reward and moving to a new state. The agent uses this new experience to update its knowledge.

SGOU

Suppose a robot is standing at the entrance of a building and needs to reach a particular room. The current position of the robot represents its state. It may decide to move forward. After moving, the robot receives information about the new position and may also receive a reward. If the movement takes the robot closer to the destination, the reward may be positive. If it moves into a blocked area, the reward may be negative.

The process continues until the robot reaches the destination or some other stopping condition is reached.

This interaction can be represented as:

Agent → Action → Environment → Reward + New State → Agent

The important point is that the agent learns from the consequences of its actions. It is not simply following a fixed sequence of instructions.

Example: A Simple Maze

Consider an agent placed inside a maze. Its objective is to reach the exit. At the beginning, it does not know which path is correct. It tries different movements and receives feedback.

If the agent moves closer to the exit, it may receive a small positive reward. If it reaches a dead end, it may receive a penalty. Reaching the exit may produce a large positive reward.

After many attempts, the agent begins to recognize which movements are more useful. Eventually, it can learn a policy that guides it towards the exit.

This simple maze contains the major ideas of reinforcement learning: the agent, environment, state, action, reward and learning through repeated interaction.

3.4.1.4 Rewards and Returns

The reward is one of the most important concepts in reinforcement learning because it provides the feedback required for learning. However, an agent is not always interested only in the reward received immediately after an action. It may also need to consider the rewards that can be obtained in the future.

Consider a robot that has two possible routes to a destination. One route provides a small reward immediately but eventually leads to a dead end. Another route provides no immediate reward but eventually leads to a much larger reward. A good learning system should be able to recognize that the second route may be more valuable in the long run.

The sequence of rewards collected by an agent over time is used to calculate the return. In simple situations, the return can be considered as the total reward obtained over a sequence of actions.

For example, suppose an agent receives rewards of +2, +3 and +5 during three successive steps. The total reward is:

2 + 3 + 5 = 10

SGOU

In reinforcement learning, future rewards are often given less importance than immediate rewards. This is controlled by a parameter called the discount factor, represented by γ (gamma).

The discounted return can be expressed as:

$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$

Here, G_t represents the return from time step t, while R represents rewards obtained at later time steps. The value of γ generally lies between 0 and 1.

When γ is closer to 0, immediate rewards are given greater importance. When γ is closer to 1, future rewards become more important.

For example, suppose an agent receives an immediate reward of 5 and a future reward of 10, with $\gamma = 0.9$. The discounted value of these rewards is:

$5 + 0.9 \times 10 = 14$

The agent therefore considers both the immediate reward and the future reward when evaluating its decisions.

3.4.2 Markov Decision Process

Many reinforcement learning problems can be represented using a mathematical framework called a Markov Decision Process (MDP). An MDP provides a structured way to describe situations in which an agent repeatedly makes decisions while interacting with an environment.

An MDP is commonly represented as:

(S, A, P, R, γ)

Here, S represents the set of states, A represents the set of possible actions, P represents the transition probabilities, R represents the reward function and γ represents the discount factor.

States in an MDP

The set of states describes all the possible situations in which the agent can find itself. For example, in a simple navigation problem, the states could represent different locations in a building.

Suppose a delivery robot moves through four locations:

$A \rightarrow B \rightarrow C \rightarrow D$

Each location can be treated as a state. If D is the destination, reaching D may provide a large positive reward.

SGOU

Actions in an MDP

The set of actions represents the choices available to the agent. At a particular state, the agent may have several possible actions.

For example, at location B, the robot may be able to move left, right or forward. The available actions can also depend on the current state. A movement that is possible at one location may not be possible at another.

Transition

A transition describes how the environment changes when an action is performed. For example, if the robot is at State A and chooses the action "Move Right", it may move to State B.

In a deterministic environment, the same action in the same state always produces the same result. In a stochastic environment, the result may involve uncertainty.

For example, a robot may successfully move forward 90% of the time but slip to another position 10% of the time. Such uncertainty can be represented using transition probabilities.

Reward Function

The reward function specifies the feedback associated with states or actions. A successful action may provide a positive reward, while an undesirable action may provide a negative reward.

Discount Factor

The discount factor determines how much importance is given to future rewards. It allows an agent to balance immediate benefits against long-term benefits.

3.4.2.1 Markov Property

The term Markov in Markov Decision Process refers to the Markov property. In simple terms, the Markov property states that the future situation depends on the current state and action rather than requiring the complete history of previous states.

Consider a robot that is currently at location B. To decide its next movement, the important information may be its current location, the available actions and the current environment. It may not need to remember every location through which it travelled before reaching B.

Therefore, the basic idea can be expressed as:

Current State + Action → Next State

This assumption simplifies the decision-making process.

Consider a game in which a player is currently standing at position C. If the relevant information about the game is fully represented by the current state, the agent can make its next decision using that state without storing the complete sequence of moves that led to C.

SGOU

The Markov property is an important assumption in many reinforcement learning models because it allows the problem to be represented using states and transitions.

3.4.2.2 Policy and Value

Once the concepts of state, action and reward are understood, two additional concepts become important: policy and value.

A policy represents the strategy followed by an agent when selecting actions. It tells the agent what action to choose when it encounters a particular state. The policy is usually represented by the symbol π (pi).

Suppose a robot is moving through a small environment. A simple policy may indicate that the robot should move right when it is at A, move right again at B and move down when it reaches C.

The policy can therefore be thought of as a strategy for making decisions.

A value represents how useful a state or action is in terms of expected future rewards. There are two important value functions: the state-value function and the action-value function.

The state-value function, represented by V(s), indicates the expected return from a particular state when a particular policy is followed.

The action-value function, represented by Q(s, a), indicates the expected return for taking action a in state s.

The idea of Q-values can be understood using a simple example. Suppose a robot is at State A and can move either left or right. If the learned Q-value for moving left is 2 and the Q-value for moving right is 8, the agent considers moving right to be more valuable in that state.

Thus, a Q-value can be viewed as a score that indicates how useful an action is in a particular state.

This concept forms the foundation of Q-Learning.

3.4.3 Q-Learning

Q-Learning is one of the most widely studied reinforcement learning algorithms. It is a model-free, off-policy algorithm that learns the value of taking different actions in different states.

The term "model-free" means that the agent does not need to know the complete mathematical model of the environment before learning. Instead, it learns through interaction and experience.

Q-Learning stores information about state-action combinations using Q-values. For small problems, these values can be stored in a structure called a Q-table.

SGOU

Consider a simple environment with three states and four possible actions.

State	Left	Right	Up	Down
A	0	0	0	0
B	0	0	0	0
C	0	0	0	0

Initially, the Q-values may all be set to zero because the agent has no previous experience.

As the agent interacts with the environment, it receives rewards and updates these values. After many experiences, actions that tend to lead to good outcomes will generally acquire higher Q-values.

The main Q-Learning update rule is:

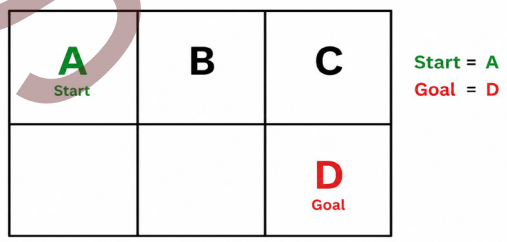
$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$

Here, Q(s,a) is the current value of taking action a in state s. The learning rate α determines how strongly new information changes the existing value. The reward obtained from the action is represented by r, while γ is the discount factor. The expression max Q(s′,a′) represents the highest estimated Q-value among the possible actions in the next state.

Although the equation appears complex, its basic idea is simple. The agent compares its old estimate with a new estimate based on the reward received and the best expected future value. The Q-value is then adjusted towards this new estimate.

3.4.3.1 Q-Learning: A Simple Example

Consider a small grid in which an agent has to reach a destination.



The agent starts at A and wants to reach D. Suppose the agent moves from A to B by choosing the action "Right". The immediate reward for this movement is 0 because the agent has not yet reached the destination.

Assume that the learning rate is $\alpha = 0.5$, the discount factor is $\gamma = 0.9$, the old Q-value for moving right from A is 2 and the highest Q-value available from the next state B is 4.

The Q-Learning update becomes:

SGOU

New Q-value = 2 + 0.5[0 + 0.9(4) − 2]

Therefore:

New Q-value = 2.8

The value has changed from 2 to 2.8 because the agent has received new information from its experience.

This update is repeated every time the agent interacts with the environment. Over many episodes, the Q-values gradually become more useful.

The important idea is not simply the calculation itself. The important idea is that experience changes the agent's knowledge. An action that repeatedly leads to good outcomes tends to receive a higher value, while actions associated with poor outcomes tend to receive lower values.

Eventually, the agent can use the learned Q-values to select actions that are expected to produce better results.

3.4.4 SARSA

SARSA is another important model-free reinforcement learning algorithm. The name SARSA comes from the sequence used in its learning process:

State → Action → Reward → State → Action

This is shortened to:

S → A → R → S → A

SARSA is an on-policy algorithm. It learns the value of actions according to the policy that the agent is actually following.

The SARSA update rule is:

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$

The equation is similar to the Q-Learning equation, but there is an important difference. Q-Learning uses the maximum Q-value among the possible actions in the next state, whereas SARSA uses the Q-value of the actual next action selected by the agent.

Suppose the agent reaches a new state and two actions are available. Their Q-values are:

Action	Q-value
Action 1	8
Action 2	4

Q-Learning considers the maximum value, which is 8.

SGOU

SARSA, however, considers the action that the agent actually chooses. If the agent selects Action 2, SARSA uses the value 4 in its update.

This difference affects how the two algorithms learn their policies.

A simple way to remember the difference is:

Q-Learning: "What is the best action available next?"

SARSA: "What action am I actually going to take next?"

3.4.4.1 Q-Learning and SARSA

Q-Learning and SARSA are closely related algorithms. Both learn from interaction with an environment, both update Q-values and both can be used without having a complete model of the environment.

The main difference lies in how they evaluate the next state.

Q-Learning is an off-policy algorithm. Its update uses the maximum estimated Q-value of the next state, even if the agent's current behaviour policy might select a different action.

SARSA is an on-policy algorithm. Its update uses the Q-value associated with the action that the agent actually selects next.

This difference can become important when exploration is taking place.

Suppose an agent is travelling through an environment where one path is very close to a dangerous area. The agent may occasionally explore different actions instead of always selecting the action with the highest Q-value. SARSA takes the consequences of the actual selected action into account when updating its knowledge, while Q-Learning focuses on the best estimated next action.

Feature	Q-Learning	SARSA
Type	Model-free	Model-free
Policy	Off-policy	On-policy
Next action used in update	Best estimated action	Actual selected action
Main idea	Learn toward the best next action	Learn according to the current behaviour

Neither method is automatically the best choice for every problem. The appropriate method depends on the characteristics of the environment and the behaviour that needs to be learned.

SGOU

3.4.5 Exploration and Exploitation

An important challenge in reinforcement learning is deciding whether the agent should explore new possibilities or exploit the knowledge it has already gained.

Exploration means trying actions that the agent does not know much about. The purpose is to discover whether those actions might produce better results.

Exploitation means choosing an action that the agent already believes is good based on its current knowledge.

Consider a person who knows one restaurant that provides good food. Returning to that restaurant is exploitation because the person is using existing knowledge. Trying a completely new restaurant is exploration because the person is gathering new information.

The same idea applies to reinforcement learning.

Suppose a game-playing agent has learned that one move usually produces a reward of 5. It can continue selecting that move, but there may be another move that it has never tried and that could produce a reward of 20.

If the agent always exploits its current knowledge, it may never discover the better move. On the other hand, if it always explores, it may spend too much time taking actions that produce poor results.

This is called the exploration-exploitation dilemma.

A good reinforcement learning system therefore needs to maintain a suitable balance between exploration and exploitation.

3.4.5.1 Epsilon-Greedy Strategy

One commonly used method for balancing exploration and exploitation is the epsilon-greedy strategy. It uses a parameter called epsilon (ε) to determine how often the agent explores.

With probability ε, the agent chooses an action randomly for exploration. With probability $1 - \varepsilon$, it selects the action with the highest estimated value for exploitation.

For example, suppose $\varepsilon = 0.2$. The agent has approximately a 20% chance of exploring and an 80% chance of exploiting its current knowledge.

Suppose the Q-values for three actions are:

Action	Q-value
Left	3
Right	8
Up	5

SGOU

The action with the highest Q-value is Right. During exploitation, the agent selects Right. During exploration, however, it may select another action to discover whether that action could be better.

At the beginning of learning, the agent usually has very little knowledge. Exploration is therefore important. As learning progresses, the agent has more information about the environment and can gradually rely more heavily on exploitation.

For example, an agent may start with a relatively high exploration probability and reduce it over time. The exact values and strategy depend on the problem.

The important principle is that exploration helps the agent discover, while exploitation helps it use what it has discovered.

3.4.6 Applications of Reinforcement Learning

Reinforcement learning is useful in situations where decisions need to be made repeatedly and the consequences of those decisions provide feedback.

Game Playing

Game playing is one of the well-known applications of reinforcement learning. An agent can learn to play games by observing the game state, selecting moves and receiving feedback based on the outcome.

A game-playing agent may initially make poor decisions. Through repeated games, however, it can learn which actions are more likely to produce successful outcomes. Reinforcement learning has been used in research involving board games, video games and other sequential decision-making tasks.

Robotics

Robotics is another important application area. A robot may need to learn how to move, navigate, grasp objects or avoid obstacles. Instead of manually specifying every possible movement, reinforcement learning can be used to learn suitable behaviour through interaction.

For example, a robotic arm can learn movements required to reach and pick up an object. The system can receive feedback based on the accuracy and efficiency of its movements.

Autonomous Systems

Reinforcement learning can be applied to decision-making problems in autonomous systems. An autonomous vehicle, for example, may need to make decisions related to speed, movement and route selection.

Real-world autonomous systems are safety-critical, so reinforcement learning cannot simply be allowed to explore without restrictions. Simulation, testing, safety constraints and validation are essential before such methods can be used in practical systems.

SGOU

Traffic Management

Traffic signal control is another area where reinforcement learning can be studied. A traffic-control agent can observe traffic conditions such as vehicle density and waiting time and adjust signal timings accordingly.

The objective may be to reduce congestion and waiting time while maintaining efficient traffic flow.

Recommendation Systems

Recommendation systems can also involve sequential decision-making. A system recommends an item, observes the user's response and uses that feedback to improve future recommendations.

For example, a recommendation system may learn that certain types of content receive more positive responses from a particular user and adjust future recommendations accordingly.

Resource Management

Reinforcement learning can also be used for resource allocation and management. In cloud computing, for example, an intelligent system may need to decide how computing resources should be allocated to applications.

The system can consider factors such as workload, response time and resource usage and learn strategies that improve overall performance.

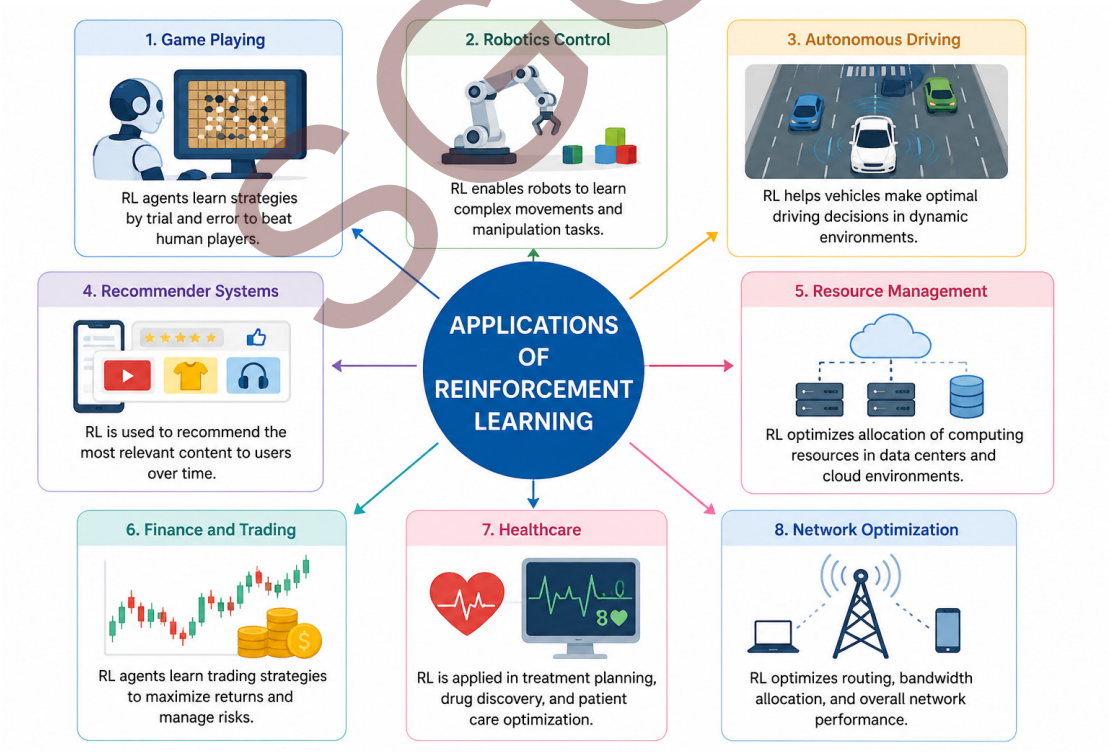


Fig.3.4.1 Applications of reinforcement learning

SGOU

Summerised Overview

Reinforcement Learning is a learning approach where an agent improves its decisions by interacting with an environment and receiving rewards or penalties. Instead of explicit instructions, learning happens through trial and error.

The process involves key elements such as agent, environment, state, action, and reward, which together form a continuous learning cycle. To model this formally, the Markov Decision Process is used, defining how decisions are made step by step.

The agent follows a policy (strategy) and evaluates decisions using value functions. Algorithms like Q-Learning and SARSA help the agent learn optimal actions over time.

A key challenge is balancing exploration (trying new actions) and exploitation (using known best actions). Reinforcement Learning is widely applied in areas such as games, robotics, recommendation systems, and autonomous systems.

Assignments

1. Explain the concept of Reinforcement Learning with a suitable real-life example.
2. Describe the components of a reinforcement learning system (agent, environment, state, action, reward) with clear explanations.
3. Explain the Markov Decision Process and discuss its key elements in detail.
4. Compare Q-Learning and SARSA. Analyze how their decision-making differs in a risky environment.
5. A robot is learning to navigate a maze. Analyze how the balance between exploration and exploitation affects its learning efficiency and final performance.

SGOU

Suggested Reading

1. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*.

2. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

3. Plaat, A. (2022). *Deep reinforcement learning: A textbook*. arXiv.

Reference

1. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

2. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

3. Russell, S., & Norvig, P. (2013). *Artificial intelligence: A modern approach* (3rd ed.). Pearson.

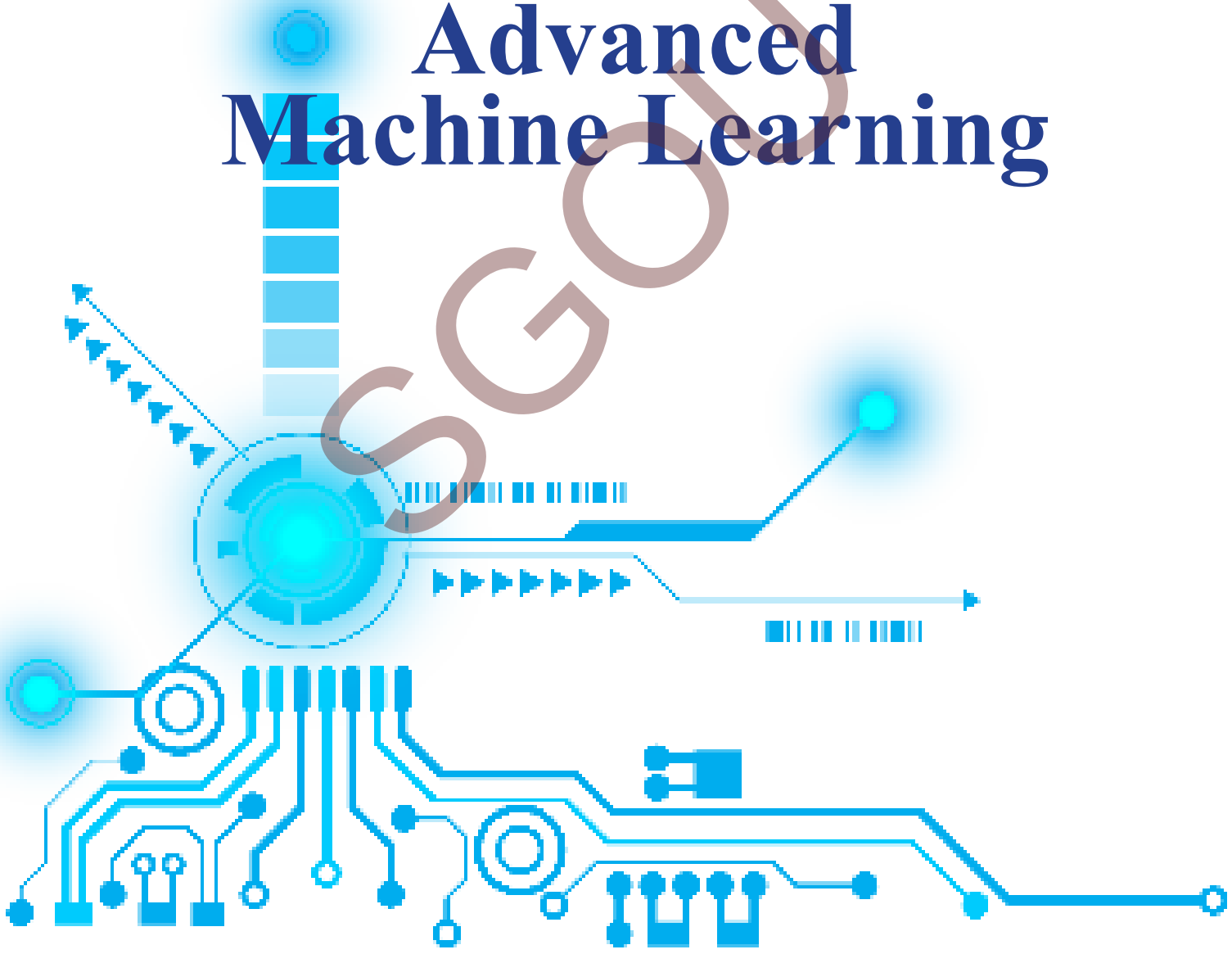
SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

BLOCK 4
Advanced Machine Learning



SGOU

1 UNIT

Neural Network Basic

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the basic concept of neural networks and their working
- describe the structure and function of perceptron and feedforward networks
- discuss the role of activation functions in neural networks
- summarize the process of backpropagation and learning in neural networks.evaluate the quality of clusters using different evaluation metrics

Background

In earlier learning, the focus has been on how computers process data using clearly defined instructions. Concepts such as algorithms, basic programming, and data processing introduced the idea that machines follow step-by-step procedures to produce results. In addition, exposure to topics like data representation, mathematical functions, and simple models would have shown that real-world problems can often be expressed in numerical form.

At the same time, everyday experiences reveal something more powerful than rule-based systems. Humans are able to recognize faces, understand speech, and make decisions even when situations are not clearly defined. This type of learning does not rely on fixed instructions but develops gradually through observation, practice, and correction. Such experiences raise an important question: can machines also learn in a similar way?

This curiosity leads to the concept of learning systems, where machines improve their performance based on data rather than explicit programming. Before reaching neural networks, learners may have encountered basic ideas of pattern recognition,

SGOU

simple classification problems, or even introductory machine learning concepts where systems learn from examples.

Neural networks build upon these ideas by introducing a structured way for machines to learn complex patterns. They combine mathematical modeling with inspiration from biological systems to create models capable of handling tasks that traditional approaches struggle with. Understanding neural networks therefore requires connecting prior knowledge of mathematics, data handling, and logical reasoning with a new way of thinking—learning from data.

This topic serves as a foundation for exploring advanced areas such as deep learning, image recognition, natural language processing, and intelligent systems. By understanding the basics of perceptrons, feedforward networks, activation functions, and backpropagation, learners begin to see how modern intelligent applications are designed and how machines gradually improve through experience.

Keywords

Perceptron model, feed forward neural network, hidden layers, activation function

Discussion

4.1.1 Introduction to Neural Networks

A computer can perform calculations extremely quickly, but traditional computer programs generally require explicit instructions about how a task should be performed. Many real-world problems, however, are difficult to solve by writing a fixed set of rules. Recognizing handwritten digits, identifying objects in images, understanding speech and predicting complex patterns are examples where the relationship between input and output may be difficult to describe using simple rules.

Artificial Neural Networks (ANNs) provide a way to learn such relationships from data. They are computational models inspired loosely by the way biological nervous systems process information. An artificial neural network consists of interconnected computational units called neurons, which process input values and pass information to other neurons as shown in Fig. 4.1.1.

A neural network does not normally learn by storing a collection of explicit rules. Instead, it learns numerical parameters, mainly weights and biases, from training data. These parameters determine how strongly different inputs influence the output.

SGOU

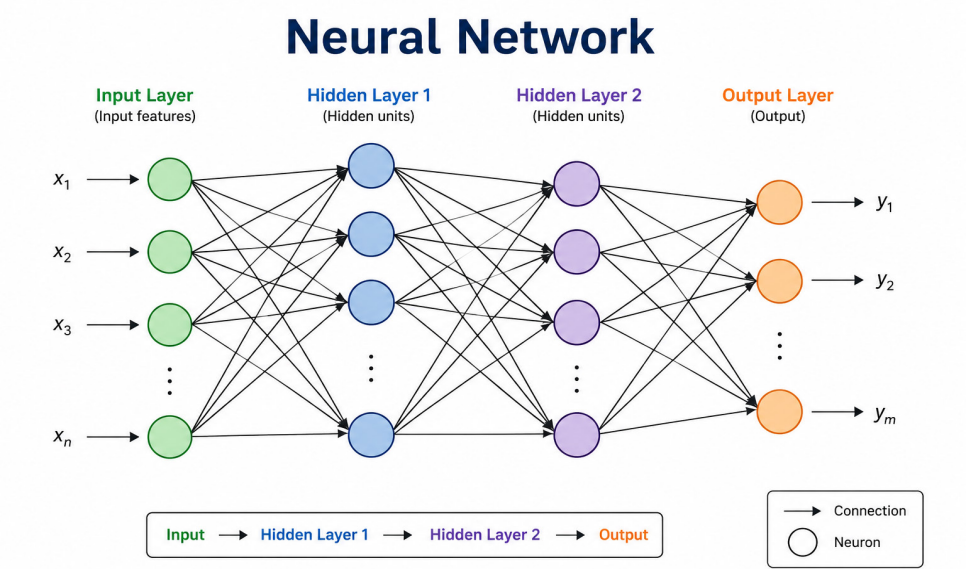


Fig.4.1.1 Neural Network

Consider a simple system that needs to determine whether a student is likely to pass an examination based on study hours and attendance. Instead of manually specifying a rule such as "a student passes if study hours are greater than five and attendance is above 80%," a neural network can learn a relationship between these input variables and the observed outcomes from training examples.

The basic learning process can be viewed as:

Input data → Neural network → Prediction → Error measurement → Parameter adjustment

The network repeatedly adjusts its parameters so that its predictions become closer to the desired outputs. The schematic diagram showing training process is depicted in Fig. 4.1.2

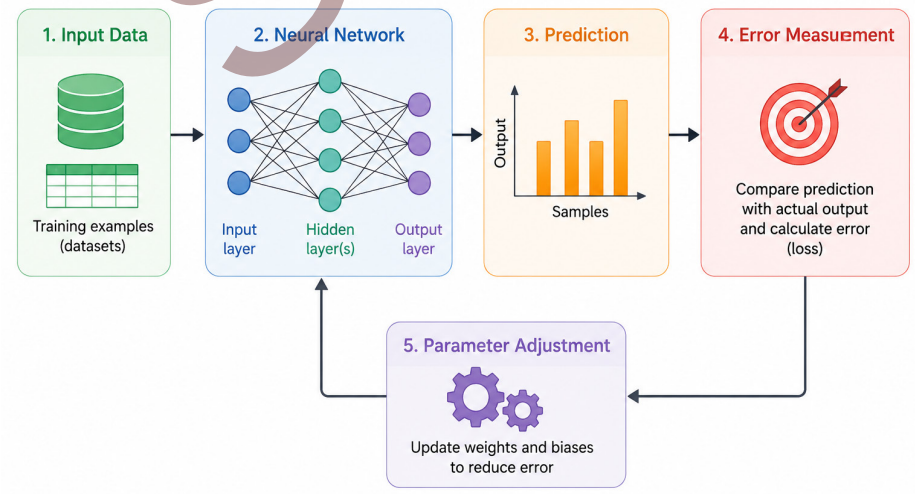


Fig. 4.1.2 Neural Network Training Process

SGOU

Neural networks can contain a single computational layer or many layers. Networks containing several layers between the input and output are generally referred to as **deep neural networks**. The basic concepts introduced in this unit form the foundation for understanding more advanced architectures such as convolutional neural networks and recurrent neural networks.

4.1.2 Artificial Neuron

The fundamental computational unit of a neural network is commonly called an artificial neuron or node. It receives one or more input values, combines them using weights and a bias, and then passes the result through an activation function.

Suppose a neuron receives three inputs:

x_1, x_2 and x_3

Each input is associated with a corresponding weight:

w_1, w_2 and w_3

The neuron first calculates a weighted sum:

$z = w_1x_1 + w_2x_2 + w_3x_3 + b$

Here, b is the bias.

The resulting value z is then passed through an activation function:

$y = f(z)$

where **f** represents the activation function and **y** is the output of the neuron.

The weights determine the contribution of the individual inputs. A larger positive weight means that an input has a stronger positive influence on the weighted sum, while a negative weight gives the input a negative influence.

The bias allows the neuron to shift the activation function and provides additional flexibility when learning a relationship.

A simple example

Suppose a neuron receives:

$x_1 = 2$
$x_2 = 3$

and has:

$w_1 = 0.5$
$w_2 = 0.2$
$b = 1$

SGOU

The weighted sum is:

$z = (0.5 \times 2) + (0.2 \times 3) + 1$

$z = 1 + 0.6 + 1 = 2.6$

The activation function then operates on 2.6 to produce the neuron's output.

This two-stage process is fundamental:

Weighted combination → Activation

4.1.3 Perceptron

The perceptron is one of the simplest forms of an artificial neural network. It was introduced by Frank Rosenblatt in the late 1950s and became an important early model for studying machine learning with artificial neurons.

A basic perceptron receives several input values, calculates their weighted sum with a bias and applies a threshold-based activation function to produce an output.

The weighted sum is:

$z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$

The classical perceptron uses a step function. For example:

$y = 1$, if $z \geq 0$

$y = 0$, if $z < 0$

Thus, the perceptron produces a binary output.

Consider a system that decides whether an applicant satisfies a simple condition based on two features. The inputs could be represented as $\mathbf{x_1}$ and $\mathbf{x_2}$, and the perceptron learns appropriate weights so that examples belonging to one class produce 1 while examples belonging to another class produce 0.

The important point is that the perceptron does not need the weights to be supplied manually. During training, the weights are adjusted based on the errors made by the model.

4.1.3.1 Perceptron Learning Rule

The perceptron learns by comparing its predicted output with the target output. When the prediction is incorrect, the weights and bias are adjusted.

A common form of the perceptron weight update is:

$w_i \leftarrow w_i + \eta(t - y)x_i$

SGOU

where:

w_i is the current weight,

η is the learning rate,

t is the target output,

y is the predicted output, and

x_i is the corresponding input.

The bias can be updated as:

$b \leftarrow b + \eta(t - y)$

The learning rate determines the size of the adjustment.

Consider an example where the target output is 1 but the perceptron predicts 0. The difference $(t - y)$ is positive, so the weights are adjusted in a direction that increases the likelihood of producing the desired output for that input.

If the target is 0 but the perceptron predicts 1, the difference becomes negative, and the weights are adjusted in the opposite direction.

This process is repeated for the training examples.

Important limitation of the perceptron

A single-layer perceptron can learn only linearly separable classification problems. In other words, the classes must be separable by a straight decision boundary in a two-dimensional case, or a hyperplane in higher dimensions.

The classic XOR problem demonstrates this limitation. XOR is not linearly separable, so a single perceptron cannot correctly learn the XOR relationship.

This limitation led to the development and use of networks containing hidden layers and nonlinear activation functions.

4.1.3.2 From Perceptron to Neural Network

A single perceptron is useful for simple classification tasks, but many real-world problems are more complex. A more powerful neural network can be created by connecting many neurons together.

When neurons are organized into layers, the output of one layer becomes the input to the next layer. A basic neural network contains an input layer, one or more hidden layers and an output layer.

The input layer receives the features from the data. The hidden layers transform the information through weighted connections and activation functions. The output layer produces the final prediction.

SGOU

For example, consider a network that receives three features:

x_1, x_2, x_3

These values enter the input layer. They are passed to neurons in a hidden layer, where weighted combinations and activation functions transform the information. The hidden-layer outputs are then passed to the output layer, which produces the final prediction.

The hidden layers are important because they allow the network to learn intermediate representations of the input data. A sufficiently expressive multilayer network can represent relationships that cannot be captured by a single linear decision boundary.

4.1.4 Feedforward Neural Networks

A feedforward neural network is a neural network in which information moves through the network in one direction, from the input layer towards the output layer. There are no connections that send information backward to an earlier layer during the forward computation.

A common feedforward architecture is called a Multilayer Perceptron (MLP). It contains an input layer, one or more hidden layers and an output layer.

Suppose a network is designed to classify whether an email is spam or not spam. The input layer may receive features extracted from the email. These values are processed by one or more hidden layers. The output layer then produces the classification.

During forward propagation, each layer receives the output of the previous layer, performs a weighted calculation and applies an activation function.

For one neuron, the calculation is:

$z = \Sigma w_i x_i + b$

followed by:

$a = f(z)$

The output a becomes an input to neurons in the next layer.

This process continues until the output layer produces the final prediction.

Why is it called feedforward?

The term "feedforward" describes the direction of information flow. Information moves:

Input → Hidden Layer(s) → Output

It does not form a cycle during the forward computation.

This distinguishes feedforward networks from recurrent neural networks, where connections can allow information from previous time steps to influence later computations.

SGOU

4.1.4.1 Layers in a Feedforward Network

The different layers in a feedforward network perform different roles.

Input Layer

The input layer receives the features of the input data. It generally does not perform the same type of learned transformation as the hidden layers. Instead, it provides the input values to the network.

For example, if a model uses age, income and credit score as three input features, the input layer represents these three features.

Hidden Layer

A hidden layer contains neurons that transform the information received from the previous layer. Each neuron calculates a weighted sum, adds a bias and applies an activation function.

A network may contain one hidden layer or several hidden layers. Increasing the number of layers allows the network to learn more complex transformations, although deeper networks also introduce additional training considerations.

Output Layer

The output layer produces the final result of the network.

The structure of the output layer depends on the task. For binary classification, a single output neuron with an appropriate activation function such as sigmoid is commonly used. For multiclass classification, multiple output neurons with a softmax activation are commonly used. For regression, an output neuron with a linear activation is often suitable.

Therefore, the activation function and number of output neurons should be selected according to the type of prediction being performed.

4.1.4.2 Weights and Biases

Weights and biases are the main learnable parameters of a neural network.

A weight determines the strength and direction of the influence of one input on a neuron. Each connection between neurons generally has an associated weight.

Suppose a neuron receives two inputs:

x_1 and x_2

with weights:

w_1 and w_2

The neuron calculates:

$z = w_1x_1 + w_2x_2 + b$

SGOU

Changing the weights changes the contribution of the corresponding inputs.

The bias is an additional learnable parameter. It allows the neuron to shift the value of the weighted sum before applying the activation function.

It is useful to think of the weights as determining how strongly the inputs contribute and the bias as providing an adjustable offset.

During training, the network learns suitable values for both weights and biases.

Initially, these parameters are usually initialized to small values using an appropriate initialization method. They are then updated repeatedly as the network learns from training data.

4.1.5 Activation Functions

If every neuron in a multilayer network used only a linear transformation, stacking multiple layers would still produce an overall linear transformation. This would severely limit the types of relationships the network could learn.

Activation functions introduce nonlinearity into neural networks.

After calculating the weighted sum, a neuron applies an activation function:

$a = f(z)$

The choice of activation function affects how information is transformed and how gradients behave during training.

Several activation functions are commonly used.

Sigmoid Function

The sigmoid function is:

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

Its output lies between 0 and 1.

The sigmoid function is particularly useful for binary classification when the output is interpreted as a probability for one of the classes.

For example, an output of 0.85 can be interpreted as a high estimated probability for the positive class, depending on how the target is defined.

However, sigmoid can suffer from **vanishing gradients** when its inputs become very large or very small. Therefore, it is generally not the default activation for hidden layers in modern deep networks.

Tanh Function

The hyperbolic tangent, or tanh, is given by:

$\tanh(z) = (e^{z} - e^{-z})/(e^{z} + e^{-z})$

SGOU

Its output ranges from −1 to +1.

Unlike sigmoid, tanh is centered around zero. It can be useful in some neural network architectures, although it can also experience vanishing gradients for large positive or negative inputs.

ReLU

The Rectified Linear Unit (ReLU) is defined as:

ReLU(z) = max(0,z)

For positive values, ReLU returns the input itself. For negative values, it returns zero.

ReLU became widely used in hidden layers because it is simple to compute and generally provides stronger gradient behaviour than sigmoid or tanh in many deep learning settings.

However, neurons can sometimes become inactive if they repeatedly receive negative inputs. This is commonly referred to as the dying ReLU problem.

4.1.5.1 Other Activation Functions

Modern neural networks use several other activation functions depending on the architecture and task.

Leaky ReLU is a variation of ReLU that allows a small negative output instead of setting every negative input to zero. It is defined as:

Leaky ReLU(z) = z, if $z > 0$

Leaky ReLU(z) = αz, if $z \leq 0$

where α is a small positive constant.

This allows a small gradient to remain for negative inputs.

Another important activation function is **softmax**, which is commonly used in the output layer of multiclass classification networks. It converts a set of scores into values that sum to 1, allowing them to be interpreted as a probability distribution across classes.

For scores $z_1, z_2, ..., z_k$, the softmax function is:

$$\textbf{softmax}(z_i) = e^{z_i} / \Sigma_j e^{z_j}$$

For example, if a model classifies an image into three categories such as cat, dog and bird, the softmax output might be:

Class	Output
Cat	0.70
Dog	0.20
Bird	0.10

SGOU

The values sum to 1, and the class with the highest output is generally selected as the predicted class.

The choice of activation function therefore depends on where the function is being used and what the network is expected to produce.

4.1.6 Forward Propagation

Once the network has been constructed and its parameters initialized, an input can be passed through the network. This process is called **forward propagation** or **forward pass**.

Consider a network with an input layer, one hidden layer and an output layer. The input values first enter the hidden layer. Each hidden neuron calculates its weighted sum, adds its bias and applies an activation function.

The resulting hidden-layer activations are then passed to the output layer. The output neurons perform their own weighted calculations and apply their activation functions to produce the final prediction.

For example, if the input is represented as:

$\mathbf{x = [x_1, x_2]}$

a hidden neuron may calculate:

$\mathbf{z_1 = w_1x_1 + w_2x_2 + b_1}$

and then:

$\mathbf{a_1 = f(z_1)}$

The output neuron can use this activation along with other hidden-layer outputs to calculate its own value.

The complete process can be summarized as:

Input → Weighted Sum → Activation → Next Layer → Weighted Sum → Activation → Output

Forward propagation(shown in Fig. 4.1.3) is used both during training and during prediction. During training, the output is compared with the target so that the network can calculate how well it performed. During prediction, the trained network simply uses its learned parameters to produce an output.

SGOU

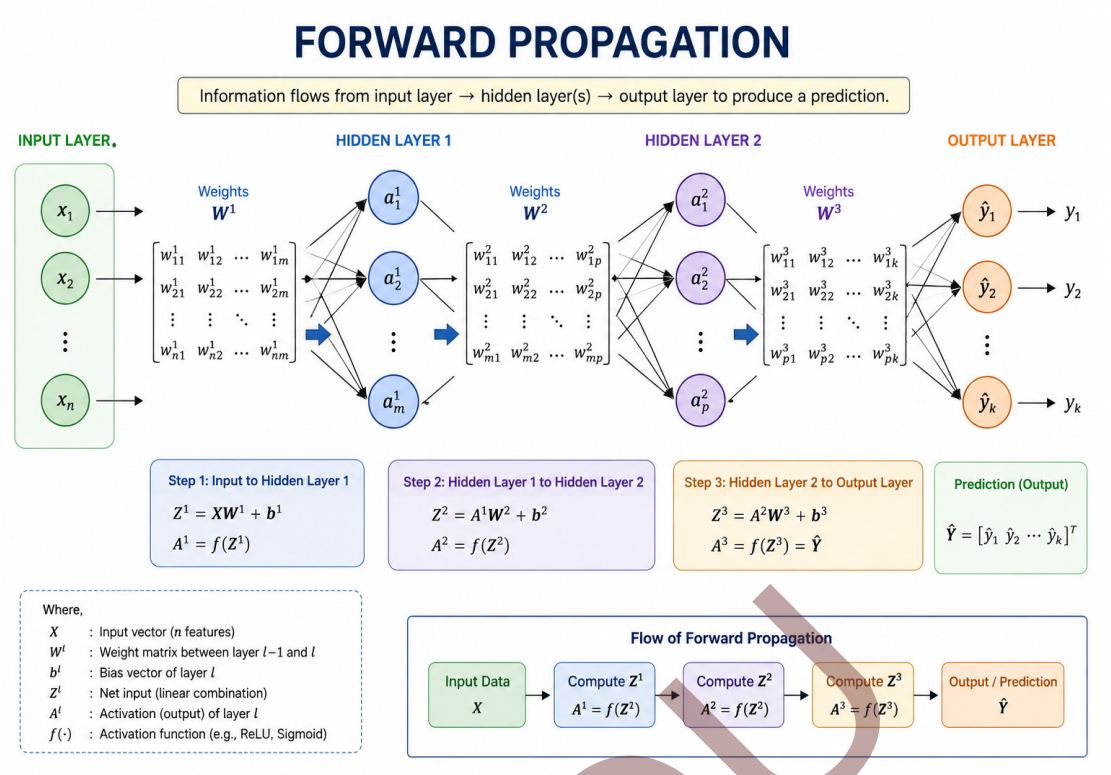


Fig. 4.1.3 Forward Propagation

4.1.7 Loss Function

A neural network needs a way to measure how far its prediction is from the desired output. This is the role of a loss function.

The loss function converts the difference between the prediction and the target into a numerical value. A smaller loss generally indicates that the prediction is closer to the target according to the chosen loss function.

For a simple regression problem, Mean Squared Error (MSE) is commonly used:

$MSE = (1/n)\ \Sigma(y_i - \hat{y}_i)^2$

Here, y_i represents the actual value and $\hat{y}_i$ represents the predicted value.

For classification problems, cross-entropy loss is commonly used. The exact form depends on whether the problem is binary or multiclass.

The loss function is important because it provides the objective that the training process attempts to minimize.

Consider a model predicting the price of a house. If the actual price is ₹50 lakh and the model predicts ₹45 lakh, the prediction contains an error. The loss function measures this error numerically. During training, the network adjusts its parameters so that the loss becomes smaller across the training data.

SGOU

It is important to distinguish between loss and accuracy. Loss measures the discrepancy between predictions and targets according to a mathematical function, whereas accuracy measures the proportion of correctly classified examples in a classification problem. A training process may optimize a loss function even when accuracy is the final evaluation metric.

4.1.8 Backpropagation

A neural network learns by adjusting its weights and biases. The process used to determine how these parameters should be changed is based on backpropagation.

Backpropagation is not itself the complete optimization algorithm. Rather, it is an algorithm for efficiently computing the gradients of the loss with respect to the network's parameters using the chain rule of calculus.

The basic training process consists of two major computational stages.

First, the input is passed through the network to produce a prediction. This is the forward pass. The prediction is then compared with the target using the loss function.

Next, the error information is propagated backward through the network. This is the backward pass, during which gradients are calculated for the weights and biases.

These gradients indicate how changing a parameter would affect the loss.

The overall idea is:

Input → Forward Pass → Prediction → Loss → Backward Pass → Gradients → Parameter Update

Suppose increasing a particular weight would increase the loss. The gradient associated with that weight provides information about this relationship. The optimizer can then adjust the weight in a direction that tends to reduce the loss.

The backward pass makes it possible to calculate these gradients efficiently even when a network contains many layers and parameters.

4.1.9 Gradient Descent and Parameter Updating

Once the gradients have been calculated through backpropagation, an optimization method can use them to update the network's parameters.

One of the simplest optimization methods is gradient descent.

For a parameter such as a weight, the basic update is:

$w \leftarrow w - \eta \, \partial L/\partial w$

Here, w is the current weight, η is the learning rate, L is the loss and $\partial L/\partial w$ represents the gradient of the loss with respect to the weight.

The negative sign indicates that the parameter is adjusted in the direction that tends to decrease the loss.

SGOU

The learning rate controls the size of the update. If the learning rate is too large, the training process may overshoot useful parameter values or become unstable. If it is too small, training may become unnecessarily slow.

Modern neural network training often uses optimizers such as Stochastic Gradient Descent (SGD), Adam and RMSProp. These methods build on gradient-based optimization and can provide improvements in convergence and training efficiency.

The important distinction is:

Backpropagation calculates gradients.

An optimizer uses those gradients to update parameters.

Keeping this distinction clear avoids a common misunderstanding that backpropagation and gradient descent are exactly the same process.

4.1.10 Complete Neural Network Training Process

The concepts discussed in this unit can now be connected into one complete training process.

Training begins with a set of examples containing input data and, for supervised neural network tasks, the corresponding target outputs. The network parameters are initialized, and an input example or batch of examples is passed through the network.

During the forward pass, each neuron calculates a weighted sum, adds a bias and applies an activation function. The information moves through the layers until the output layer produces a prediction.

The prediction is compared with the target using an appropriate loss function. The resulting loss indicates how well the network performed for that training example or batch.

The backward pass then uses backpropagation to calculate the gradients of the loss with respect to the weights and biases. An optimizer such as gradient descent uses these gradients to update the parameters.

This process is repeated over many training examples and multiple passes through the training dataset.

A complete training cycle(Fig.4.1.4) can therefore be represented as:

SGOU

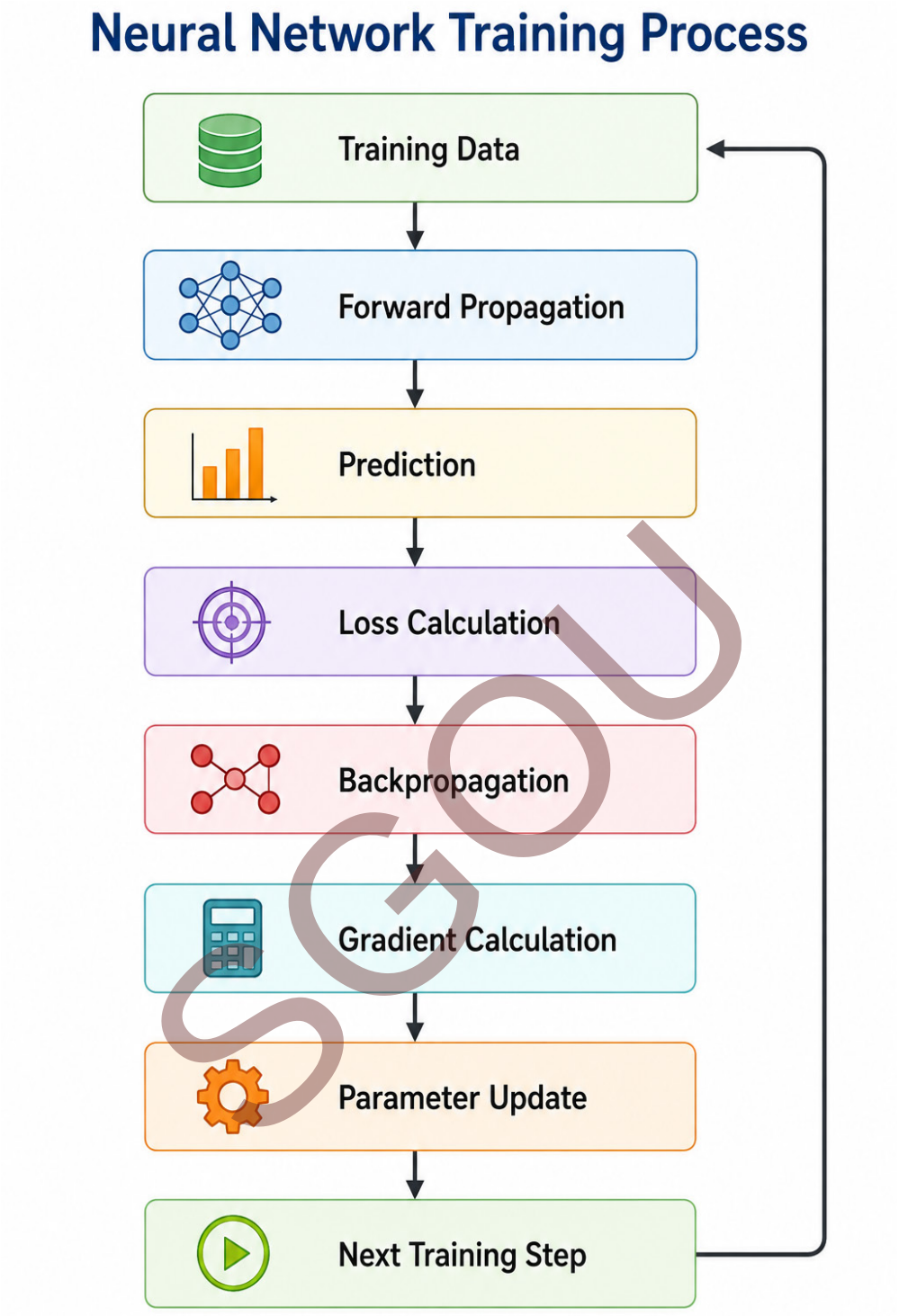


Fig.4.1.4 Neural Network Training process

SGOU

Summerised Overview

Neural networks are computational models inspired by the human brain that enable machines to learn from data rather than follow fixed instructions. They consist of interconnected layers of neurons, where each neuron processes inputs and passes information forward. Starting from the basic perceptron model, which performs simple binary classification, neural networks evolve into more powerful feedforward networks with hidden layers capable of learning complex patterns. These models are widely used in tasks such as image recognition, speech processing, and prediction.

A key aspect of neural networks is their ability to learn through a process called backpropagation, where errors are calculated and used to adjust weights for better performance. Activation functions introduce non-linearity, allowing the network to model real-world complexities. By continuously improving through training, neural networks become capable of making accurate predictions and decisions. Understanding these fundamental concepts provides a strong foundation for exploring advanced areas of artificial intelligence and deep learning.

SGOU

Assignments

1. Explain the concept of a neural network and describe its basic structure with suitable examples.

2. Describe the working of a perceptron model and discuss its limitations in solving complex problems.

3. Explain feedforward neural networks and discuss the role of hidden layers and activation functions in learning.

4. Compare the perceptron and multi-layer feedforward neural network in terms of capability and limitations. Analyze why multiple layers improve performance.

5. Analyze the role of backpropagation in neural network learning. How does the learning rate affect the performance of the model?

Suggested Reading

1. Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook*. Springer International Publishing.

2. Stevens, C. J. (2001). Neural networks. In *Reader's guide to the social sciences* (pp. 1116–1117). Fitzroy Dearborn.

3. Wasserman, P. D., & Oetzel, R. M. (1990). *NeuralSource: The bibliographic guide to artificial neural networks*. Van Nostrand Reinhold.

Reference

1. Haykin, S. S. (2009). *Neural networks and learning machines* (3rd ed.). Prentice Hall.

2. Aggarwal, C. C. (2023). *Neural networks and deep learning*. Springer.

3. Alavala, C. R. (n.d.). *Fuzzy logic and neural networks: Basic concepts and application*. New Age International.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

2 UNIT Deep Learning Architectures

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the basic concepts of Deep Learning and Artificial Neural Networks.
- describe the architecture and working of Convolutional Neural Networks.
- explain the working and applications of Recurrent Neural Networks.
- describe the structure and working of Long Short-Term Memory networks.
- compare CNN, RNN and LSTM and identify their applications in Computer Vision and Natural Language Processing.

SGOU

Background

Deep Learning is widely used in modern applications such as image recognition, speech processing, language processing, medical image analysis and intelligent systems. Before studying deep learning architectures, learners should have a basic understanding of Artificial Intelligence, Machine Learning, neural networks and basic mathematics. Knowledge of programming concepts and basic data handling is also useful for understanding how deep learning models process data.

Consider a mobile phone with a face recognition system. The system receives an image and identifies important visual features such as edges, shapes, textures and facial patterns. A Convolutional Neural Network (CNN) is suitable for this task because it can learn spatial features from images. Similarly, when a system processes a sentence, the order of words becomes important. Recurrent Neural Networks (RNNs) are designed to process sequential information, while Long Short-Term Memory (LSTM) networks are designed to handle long-term dependencies more

effectively. Studying these architectures helps learners understand how deep learning techniques are selected according to the nature of the data and the problem.

Keywords

Deep Learning, Artificial Neural Network, CNN, Convolution, Kernel, Feature Map, ReLU, Pooling, RNN, Hidden State, LSTM, Cell State, Forget Gate, Input Gate, Output Gate, Computer Vision, Natural Language Processing, Sequence Processing.

Discussion

4.2.1 Introduction

Deep Learning is an important area of Artificial Intelligence and Machine Learning that uses artificial neural networks with multiple layers to learn complex patterns from large amounts of data. Deep learning techniques are widely used in applications such as image recognition, speech processing, natural language processing, medical image analysis, recommendation systems, and autonomous systems. Different types of deep learning architectures are designed to handle different kinds of data and learning problems.

Consider a facial recognition system used in a mobile phone. The system receives an image of a person's face and needs to identify important visual patterns such as edges, shapes, textures, and facial features. A Convolutional Neural Network (CNN) is particularly suitable for this type of task because it can automatically learn spatial features from images. Similarly, when a system processes a sentence such as "The learner is reading a book", the order of the words is important. Recurrent Neural Networks (RNNs) are designed to process sequential data, while Long Short-Term Memory (LSTM) networks improve the ability of RNNs to remember important information over longer sequences.

This unit introduces the fundamental concepts and architectures of CNNs, RNNs, and LSTMs. It also discusses their applications in Computer Vision and Natural Language Processing (NLP). The unit helps learners understand how different deep learning architectures are selected according to the nature of the input data and the problem being solved.

SGOU

4.2.2 Deep Learning

Deep Learning is a branch of Machine Learning that uses neural networks with multiple layers to learn representations and patterns from data.

A traditional Machine Learning system may require humans to identify and provide important features. Deep Learning models can automatically learn useful features from raw or minimally processed data.

For example, in image recognition:

Traditional approach

Image → Feature Extraction → Machine Learning Model → Prediction

Deep Learning approach

Image → Deep Neural Network → Prediction

Deep learning models can learn increasingly complex representations through multiple layers.

4.2.3 Artificial Neural Network

An **Artificial Neural Network (ANN)** is a computational model inspired by the structure and functioning of biological neural systems.

A basic neural network consists of:

1. Input Layer

The input layer receives data.

2. Hidden Layers

Hidden layers process the input and learn patterns.

3. Output Layer

The output layer produces the final prediction or classification.

For example, in an image classification problem, the input layer receives image data, hidden layers learn visual features, and the output layer predicts the class of the image.

SGOU

4.2.4 Need for Different Deep Learning Architectures

Different types of data have different characteristics.

Table 4.2.1 Different types of data and suitable architecture

Data Type	Important Characteristic	Suitable Architecture
Images	Spatial relationships	CNN
Sequential data	Order of information	RNN
Long sequences	Long-term dependencies	LSTM
Text	Sequential and contextual information	RNN/LSTM and other NLP models
Video	Spatial and temporal information	CNN with sequence models

Therefore, selecting an appropriate architecture is an important part of deep learning model development.

4.2.5 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning architecture mainly designed for processing data with a grid-like structure, particularly images.

CNNs are widely used in:

- Image classification
- Object detection
- Face recognition
- Medical image analysis
- Image segmentation
- Handwritten digit recognition
- Video analysis

The major advantage of CNNs is their ability to automatically learn spatial features from images.

4.2.5.1 Basic Architecture of CNN

A typical CNN contains the following components:

Input Image → Convolution Layer → Activation Function → Pooling Layer → Fully Connected Layer → Output

SGOU

The main components are:

1. Convolution Layer

The Convolution Layer is one of the most important components of a CNN.

It uses small filters or kernels to scan the input image and detect features.

A filter may learn to detect:

- Edges
- Corners
- Lines
- Textures
- Shapes

During training, the values of these filters are learned automatically. The convolution operation is depicted as shown in Fig 4.2.1

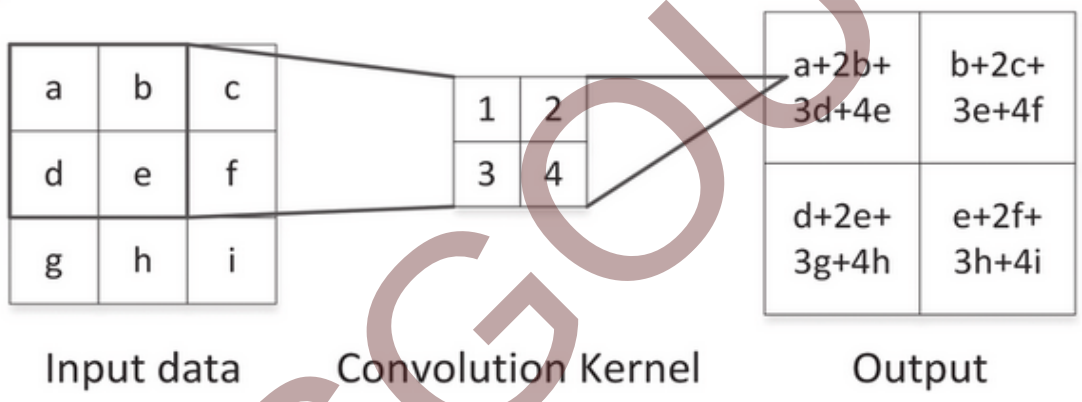


Fig.4.2.1 Convolution operation

Example

Consider an image of a cat.

The initial convolution layers may detect simple features such as edges. Deeper layers can combine these features to detect shapes such as eyes, ears, and eventually the complete structure of the cat.

The output produced by applying a convolution filter to an input is called a Feature Map.

A feature map represents the locations where particular features are detected.

For example, a filter designed to identify vertical edges produces a feature map showing areas of the image containing strong vertical edges.

2. Activation Function

An activation function introduces non-linearity into a neural network.

SGOU

A commonly used activation function in CNNs is ReLU (Rectified Linear Unit).

It is represented as:

ReLU(x) = max(0, x)

ReLU converts negative values into zero while retaining positive values.

Advantages of ReLU

- Simple to calculate.
- Computationally efficient.
- Helps neural networks learn non-linear relationships.
- Reduces some problems associated with very small gradients.

3. Pooling Layer

A Pooling Layer reduces the spatial dimensions of feature maps while retaining important information.

Common types include:

- Max Pooling
- Average Pooling

Max Pooling

Max Pooling selects the maximum value from a region.

For example, if a region contains:

2	**5**
1	**4**

Max Pooling selects:

5

Pooling helps reduce computation and can make the model less sensitive to small changes in the position of features.

4. Fully Connected Layer

The Fully Connected Layer connects learned features to the final classification or prediction.

After several convolution and pooling operations, the extracted features are converted into a suitable form and passed to fully connected layers.

For example, in a cat-versus-dog classification system, the final layer may produce

SGOU

probabilities for:

- Cat
- Dog

The class with the highest probability can be selected as the prediction.

4.2.5.2 Applications of CNN

CNNs are widely used in Computer Vision.

Image Classification

CNNs can classify images into predefined categories.

Example:

Input: Image of a flower
Output: Rose

Face Recognition

CNNs can learn facial features and identify or verify individuals.

Medical Image Analysis

CNNs can analyse X-rays, CT scans, MRI images, and other medical images to assist in identifying abnormalities.

Object Detection

CNN-based systems can identify objects and their locations within an image.

Image Segmentation

CNN-based architectures can divide an image into meaningful regions.

Autonomous Vehicles

Computer vision systems use CNN-based models to identify roads, vehicles, pedestrians, traffic signs, and other objects.

4.2.6 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a neural network architecture designed to process sequential data. The network specifically designed to handle sequential data by maintaining memory of previous inputs through an internal hidden state

Examples of sequential data include:

- Text
- Speech

SGOU

- Time-series data
- Sensor data
- Stock information
- Weather observations

Unlike a basic feed-forward neural network, an RNN maintains information from previous steps while processing the current input.

4.2.6.1 Need for RNN

Consider the sentence:

"The learner went to the library because she needed a book."

Understanding the word "she" requires information from earlier words.

RNNs process information sequentially and use information from previous steps to influence the current output.

Therefore, RNNs are suitable for problems where the order of data is important.

4.2.6.2 Basic Working of RNN

The fundamental processing unit in RNN is a Recurrent Unit. They hold a hidden state that maintains information about previous inputs in a sequence. Recurrent units can "remember" information from prior steps by feeding back their hidden state, allowing them to capture dependencies across time.

Suppose a sequence contains:

$x_1, x_2, x_3, \ldots, x_t$

The RNN processes the elements one at a time.

At each step, the network receives:

Current input

Previous hidden state

The hidden state carries information from previous steps.

A simplified representation is:

Current Input + Previous Hidden State → Current Hidden State → Output

This allows the network to maintain a form of memory about previously processed information as shown in Fig. 4.2.2

SGOU

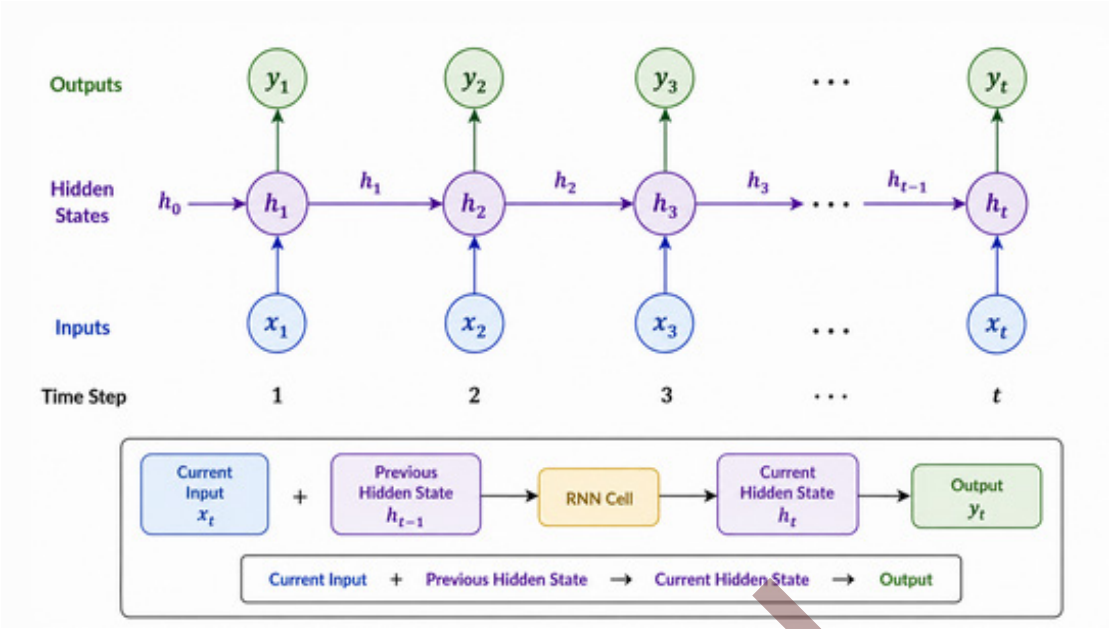


Fig. 4.2.2 Architecture of RNN

4.2.6.3 Applications of RNN

RNNs can be used in:

1. Language modelling
2. Text generation
3. Sentiment analysis
4. Speech processing
5. Time-series prediction
6. Sequence classification
7. Machine translation
8. Handwriting recognition

4.2.6.4 Limitations of RNN

Although RNNs are useful for sequential data, they have limitations.

- **Difficulty with Long-Term Dependencies**

RNNs may have difficulty remembering information from much earlier parts of a long sequence.

SGOU

- **Vanishing Gradient Problem**

During training, gradients can become extremely small as they are propagated through many time steps.

- **Exploding Gradient Problem**

In some situations, gradients can become excessively large.

- **Sequential Processing**

RNNs process sequence elements step by step, which can make training less efficient.

These limitations motivated the development of improved recurrent architectures such as **LSTM**.

4.2.7 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a special type of RNN designed to handle long-term dependencies more effectively.

LSTM networks use a memory mechanism controlled by gates.

The major components are:

1. Forget Gate
2. Input Gate
3. Cell State
4. Output Gate

An LSTM cell maintains information through a **cell state**.

The gates control the information that should:

- Be removed
- Be added
- Be used for output

This allows the network to retain important information for longer periods.

1. Forget Gate

The Forget Gate determines which information from the previous cell state should be discarded.

For example, while processing a long sentence, some earlier information may no longer be relevant. The forget gate helps the network reduce the influence of such information.

SGOU

2. Input Gate

The Input Gate determines which new information should be stored in the cell state.

It helps the network select relevant information from the current input.

3. Cell State

The Cell State acts as a pathway for carrying important information across different time steps.

It is one of the key features that enables LSTM to preserve useful information over longer sequences.

4. Output Gate

The Output Gate determines which information from the current cell state should be used to produce the output.

Thus, the three gates work together to control information flow.

An LSTM, like a basic RNN, has a hidden state, where Ht is the hidden state of the current timestamp and H(t-1) is the hidden state of the previous timestamp. Additionally, C(t-1) and C(t) for the prior and current timestamps, respectively, represent the cell state of LSTM.

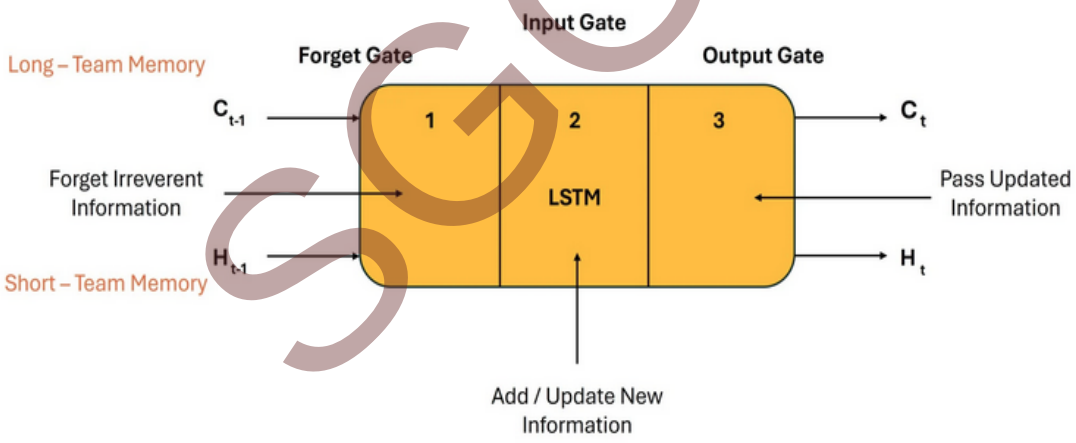


Fig. 4.2.3 Cell state of LSTM.

4.2.7.1 Applications of LSTM

LSTMs are useful in many sequence-processing applications.

- **Sentiment Analysis**

LSTMs can analyse a sequence of words to determine whether a text expresses a positive, negative, or neutral sentiment.

SGOU

- **Language Modelling**

LSTMs can learn relationships between words and predict likely next words.

- **Machine Translation**

LSTMs have been used to process source-language sequences and generate target-language sequences.

- **Speech Recognition**

LSTMs can process sequences of speech-related features.

- **Time-Series Prediction**

LSTMs can learn patterns in sequential numerical data.

- **Text Generation**

LSTMs can generate text based on patterns learned from training data.

4.2.8 CNN, RNN and LSTM Comparison

Table 4.2.2 Comparison of CNN, RNN and LSTM

Feature	CNN	RNN	LSTM
Main purpose	Spatial feature learning	Sequence processing	Long-term sequence processing
Suitable data	Images and spatial data	Sequential data	Long sequential data
Main mechanism	Convolution	Recurrent connections	Memory and gates
Memory mechanism	Not inherently sequential	Hidden state	Cell state and hidden state
Main application	Computer Vision	Sequence processing	NLP and long sequences
Major advantage	Learns spatial features	Handles sequence order	Handles long-term dependencies
Major limitation	Not designed primarily for sequence memory	Difficulty with long dependencies	More complex than basic RNN

SGOU

4.2.9 CNN in Computer Vision

Computer Vision is an area of Artificial Intelligence that enables computers to interpret and understand visual information.

CNNs are particularly important in Computer Vision because images contain spatial relationships.

For example, in an image of a vehicle:

- Early layers may detect edges.
- Intermediate layers may detect shapes.
- Deeper layers may detect wheels, windows, and body structures.
- Final layers may classify the image as a car, bus, or motorcycle.

Thus, CNNs automatically learn increasingly complex visual representations.

4.2.9.1 Major Computer Vision Applications

CNN-based deep learning systems are used in:

- **Image Classification**

Identifying the category of an image.

- **Object Detection**

Identifying objects and their positions.

- **Image Segmentation**

Dividing an image into meaningful regions.

- **Facial Recognition**

Identifying or verifying people using facial features.

- **Medical Image Analysis**

Analysing medical images to assist healthcare professionals.

- **Autonomous Systems**

Identifying objects and environmental conditions for autonomous vehicles and robots.

4.2.10 Deep Learning in Natural Language Processing

Natural Language Processing (NLP) is a field of Artificial Intelligence concerned with enabling computers to process and understand human language.

SGOU

Text is naturally sequential because the order of words can affect meaning.

For example:

“The dog chased the cat.”

and

“The cat chased the dog.”

contain the same words but have different meanings because their order is different.

RNNs and LSTMs can process such sequences and learn relationships between words.

4.2.11 CNN and NLP

Although CNNs are mainly associated with images, they can also be used in NLP.

A one-dimensional convolution can identify local patterns in text.

For example, CNNs can detect important groups of words that may indicate sentiment.

Consider:

“very good product”

A CNN can learn local word patterns that contribute to classification.

Therefore, CNNs can be used for certain NLP tasks such as:

- Text classification
- Sentiment analysis
- Sentence classification

4.2.12 RNN and NLP

RNNs are naturally suitable for NLP because they process text sequentially.

For example:

I → am → learning → deep → learning

The RNN processes each word while maintaining information from previous words.

This helps the model understand relationships between elements in a sequence.

4.2.13 LSTM and NLP

LSTMs improve the ability of recurrent networks to retain useful information over longer sequences.

SGOU

For example, consider:

“The book that the learner borrowed from the library yesterday was interesting.”

Understanding the relationship between different words may require information from earlier parts of the sentence.

LSTM can retain relevant information over multiple time steps and is therefore useful for many NLP applications.

4.2.14 Applications in Vision and NLP

Table 4.2.3 Deep Learning Architectures and Applications

Architecture	Vision Applications	NLP Applications
CNN	Image classification, object detection, face recognition	Text classification, sentiment analysis
RNN	Video and sequence-related tasks	Language modelling, sequence processing
LSTM	Video/time-series applications	Translation, sentiment analysis, text generation



Deep Learning uses neural networks with multiple layers to learn complex patterns from data. Artificial Neural Networks consist of input, hidden and output layers and form the basic foundation for many deep learning architectures.

Convolutional Neural Networks (CNNs) are mainly designed for spatial data such as images. CNNs use convolution layers, activation functions, pooling layers and fully connected layers to learn useful visual features. They are widely used in image classification, object detection, face recognition, image segmentation and medical image analysis.

Recurrent Neural Networks (RNNs) are designed to process sequential data. They use a hidden state to retain information from previous steps. RNNs are useful for text, speech and time-series data, but basic RNNs may have difficulty learning long-term dependencies.

Long Short-Term Memory (LSTM) is an improved recurrent architecture that uses a cell state and gates to control information flow. The forget gate, input gate and output gate help LSTM retain relevant information over longer sequences.

SGOU

CNNs are particularly useful in Computer Vision, while RNNs and LSTMs are useful in Natural Language Processing and other sequential applications. CNNs can also be used for certain NLP tasks. Understanding the characteristics of each architecture helps in selecting an appropriate model for a given application.

Assignments

1. Explain the basic architecture of a Convolutional Neural Network with a suitable example.
2. Describe the working of an RNN and explain its major limitations.
3. Explain the role of the forget gate, input gate, cell state and output gate in an LSTM.
4. Compare CNN, RNN and LSTM based on their architecture, suitable data, advantages and applications.
5. Explain the applications of CNN, RNN and LSTM in Computer Vision and Natural Language Processing.

SGOU

Suggested Reading

1. Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press.
2. Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.
3. Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.
4. Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook*. Springer.
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Reference

1. O'Neill, M. (2023). *Deep learning for computer vision*. Packt Publishing.
2. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2020). *Efficient processing of deep neural networks*. Morgan & Claypool Publishers.
3. Charniak, E. (2019). *Introduction to deep learning*. MIT Press.
4. Bengio, Y. (2016). *Deep learning*. MIT Press.

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

3 UNIT Model Deployment and Case Studies

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the concepts of model saving and model deployment.
- demonstrate the process of loading and deploying a trained machine learning model.
- identify commonly used tools and platforms for model deployment.
- analyse the applications of machine learning models in healthcare, finance, and e-commerce.
- analyse the major challenges and considerations involved in deploying machine learning models.

SGOU

Background

A machine learning model becomes useful in a real-world application only after it is made available for practical use. For example, a recommendation model developed for an online shopping application may be trained using customer purchase history and product interactions. After training, the model must be connected to the website or mobile application so that it can provide recommendations to customers. Similarly, a fraud detection model must be deployed in a financial system to analyse new transactions. Therefore, understanding model saving, loading, deployment tools, and practical applications is important for implementing machine learning systems.

Keywords

Model Deployment, Model Saving, Model Loading, Serialization, Deserialization, Batch Prediction, Real-Time Prediction, Flask, FastAPI, TensorFlow Serving, Docker, ONNX, Cloud Deployment

Discussion

4.3.1 Introduction to Model Deployment

Machine learning development does not end when a model has been trained and evaluated. A trained model must be made available for practical use. **Model deployment** is the process of making a trained machine learning or deep learning model available in a real-world environment so that it can receive new data and produce predictions.

Consider an online shopping application. A recommendation model may be trained using customers' previous purchases, searches, and product interactions. After training, the model needs to be integrated with the website or mobile application. When a customer opens the website, the deployed model can analyse the customer's information and recommend suitable products.

A typical machine learning workflow can be represented as:

Data Collection → Data Preparation → Model Training → Model Evaluation → Model Saving → Model Deployment → Prediction → Monitoring

Deployment is therefore an important stage in the machine learning life cycle.

4.3.2 Model Saving

Model Saving is the process of storing a trained machine learning or deep learning model so that it can be used later without training the model again.

Training a deep learning model may require significant computational resources and time. Once a model has been trained successfully, saving it allows the model to be reused.

For example, suppose a CNN model has been trained to classify medical images. Training the model again every time a prediction is required would be inefficient. Instead, the trained model can be saved and loaded whenever required.

SGOU

4.3.2.1 Common Model Saving Methods

Different frameworks provide different methods for saving models.

1. Pickle

Pickle is a Python mechanism used for serializing Python objects.

It can be used to save certain machine learning models and Python objects.

2. Joblib

Joblib is commonly used for saving machine learning models, particularly models containing large numerical arrays.

3. Keras Model Saving

Keras provides methods for saving deep learning models.

A trained neural network can be saved and loaded later for prediction or further training.

4. ONNX

Open Neural Network Exchange (ONNX) provides a common format for representing machine learning models.

It can help models move between different frameworks and deployment environments.

4.3.3 Model Serialization

Serialization is the process of converting a model or its data into a format that can be stored or transferred.

The stored model can later be reconstructed through a process called deserialization.

The basic process can be represented as:

Trained Model → Serialization → Saved Model

and later:

Saved Model → Deserialization → Loaded Model

Serialization is useful when a model needs to be transferred from the development environment to a deployment environment.

4.3.4 Model Loading

Model Loading is the process of restoring a previously saved model into memory so that it can be used for prediction, evaluation, or further training.

The general process is:

Saved Model → Load Model → Input Data → Prediction

SGOU

For example, a trained image classification model can be saved on a server. When the application starts, the model can be loaded into memory. New images can then be supplied to the model for prediction.

Advantages of Model Loading

- Avoids retraining.
- Makes prediction faster after initial loading.
- Supports application development.
- Allows models to be reused.
- Supports deployment in different environments.

4.3.5 Model Deployment

Model Deployment is the process of integrating a trained machine learning or deep learning model into an environment where it can receive new data and generate predictions.

A deployed model can be used through:

- Web applications
- Mobile applications
- Desktop applications
- Cloud services
- Enterprise software
- IoT systems
- APIs

For example, a fraud detection model can be deployed on a financial server. When a transaction occurs, the system sends relevant transaction information to the model. The model produces a prediction such as **normal transaction** or **potentially fraudulent transaction**.

4.3.5.1 Basic Model Deployment Process

The major steps involved in model deployment are:

Step 1: Train the Model

The model is trained using suitable training data.

Step 2: Evaluate the Model

The model is evaluated using appropriate performance measures.

SGOU

Step 3: Save the Model

The trained model is stored in a suitable format.

Step 4: Prepare the Deployment Environment

Required software, libraries, dependencies, and hardware are prepared.

Step 5: Load the Model

The saved model is loaded into the deployment environment.

Step 6: Connect the Model to an Application

The model is connected to an application or service.

Step 7: Provide New Input

The application sends new data to the model.

Step 8: Generate Prediction

The model processes the input and produces a prediction.

Step 9: Monitor the Model

The deployed model is monitored to ensure that it continues to perform properly.

4.3.5.2 Types of Model Deployment

Models can be deployed in different environments.

1. Local Deployment

The model runs on a local computer.

This approach is useful for development, testing, and demonstrations.

2. Server Deployment

The model runs on a server and provides predictions to applications or users.

3. Cloud Deployment

The model is deployed on a cloud platform and can be accessed through network services.

4. Edge Deployment

The model is deployed close to the location where data is generated.

For example, an AI model may be deployed directly on a mobile phone, camera, or IoT device.

SGOU

4.3.5.3 Batch and Real-Time Deployment

Deployment can also be classified based on how predictions are generated.

Batch Prediction

In **batch prediction**, a large collection of data is processed together.

For example, an e-commerce company may process the previous day's customer transactions and generate recommendations for thousands of customers.

Real-Time Prediction

In **real-time prediction**, the model generates a prediction when a new input is received.

For example, a fraud detection system may analyse a transaction immediately after it is submitted.

Table 4.3.1: Batch Prediction and Real-Time Prediction

Feature	Batch Prediction	Real-Time Prediction
Processing	Data processed in groups	Data processed individually or in small requests
Response	Not immediate	Usually immediate
Suitable for	Large-scale periodic analysis	Time-sensitive applications
Example	Daily product recommendation	Online fraud detection

SGOU

4.3.5.4 Deployment Tools

Several tools and technologies can be used to deploy machine learning and deep learning models.

1. Flask

Flask is a lightweight Python web framework.

A trained model can be connected to a Flask application and exposed through an API.

For example:

User Input → Flask API → ML Model → Prediction → User

Flask is useful for simple machine learning deployment projects and demonstrations.

2. FastAPI

FastAPI is a modern Python framework for developing APIs.

It can be used to create services through which applications send data to a machine learning model and receive predictions.

FastAPI is useful when efficient API development and clear input and output specifications are required.

3. TensorFlow Serving

TensorFlow Serving is a system designed for serving trained TensorFlow models.

It allows applications to send input data to a deployed model and receive predictions.

It is particularly useful for deploying TensorFlow-based deep learning models in production environments.

4. Docker

Docker is a containerization platform.

A machine learning application may depend on a particular version of Python, libraries, frameworks, and system components. Docker can package the application and its dependencies into a container.

The basic idea is:

Model + Application + Dependencies → Container

This helps create a more consistent deployment environment.

5. ONNX

ONNX provides a common model representation that can help models move between supported frameworks and deployment environments.

It is useful when model portability is important.

6. Cloud Platforms

Cloud platforms provide computing resources and services for deploying machine learning models.

Examples include:

- Amazon Web Services
- Microsoft Azure
- Google Cloud
- Other cloud-based machine learning platforms

Cloud deployment can provide scalability and access to computing resources without maintaining all infrastructure locally.

SGOU

4.3.5.5 Model Deployment Challenges

Although deployment makes models useful in practical applications, several challenges must be considered.

1. Model Performance

The deployed model should provide predictions within acceptable time limits.

2. Scalability

The system should be able to handle increasing numbers of users and requests.

3. Security

Model services and data should be protected from unauthorized access.

4. Dependency Management

The required libraries and software versions should be properly maintained.

5. Data Changes

The characteristics of real-world data may change over time.

6. Model Maintenance

Models may need to be updated or retrained when their performance decreases.

7. Monitoring

The deployed system should be monitored for errors, performance issues, and changes in data.

4.3.6 Case Study 1: Healthcare

Machine learning and deep learning have important applications in healthcare.

Consider a medical image classification system that uses a CNN to analyse X-ray images.

Problem

Healthcare professionals may need assistance in identifying possible abnormalities in a large number of medical images.

Solution

A CNN model can be trained using labelled medical images.

The process can be represented as:

Medical Images → Preprocessing → CNN Training → Model Evaluation → Model Saving → Deployment → New Image → Prediction

SGOU

Deployment

The trained model can be deployed on a healthcare server.

When a new image is submitted, the system processes the image and provides a prediction.

Benefits

- Supports analysis of large numbers of images.
- Can provide quick predictions.
- Helps healthcare professionals in decision support.
- Reduces repetitive manual analysis.

Important Considerations

Healthcare applications require special attention to:

- Data privacy
- Security
- Reliability
- Accuracy
- Human oversight
- Interpretability

A model prediction should support professional decision-making rather than automatically replace appropriate clinical judgment.

4.3.7 Case Study 2: Finance

Machine learning is widely used in financial applications.

Consider a fraud detection system.

Problem

Financial institutions may need to identify suspicious transactions among a large number of legitimate transactions.

Solution

A machine learning model can be trained using transaction-related features such as:

- Transaction amount
- Transaction time
- Location
- Transaction frequency
- Account activity
- Previous transaction patterns

SGOU

Deployment

The trained model can be deployed as a real-time prediction service.

The process is:

Transaction → Feature Processing → Deployed Model → Risk Prediction → Action

The model may classify a transaction as:

- Normal
- Suspicious

Benefits

- Faster identification of suspicious transactions.
- Ability to process large amounts of transaction data.
- Supports automated risk assessment.
- Helps reduce financial losses.

Important Considerations

Financial applications require:

- Security
- Data protection
- Reliability
- Fairness
- Monitoring
- Explainability

Interpretability can be particularly important when a financial decision affects a customer.

4.3.8 Case Study 3: E-Commerce

Machine learning is widely used in e-commerce platforms.

Consider a product recommendation system.

Problem

An e-commerce platform may contain thousands or millions of products. Customers may find it difficult to identify products that are relevant to their interests.

Solution

A recommendation model can analyse information such as:

- Previous purchases
- Search history

SGOU

- Product views
- Ratings
- Customer preferences
- Similar customer behaviour

The model can generate personalized product recommendations.

Deployment

The recommendation model can be integrated with the website or mobile application.

The process is:

Customer Activity → Data Processing → Recommendation Model → Recommended Products

Benefits

- Provides personalised recommendations.
- Improves customer experience.
- Helps customers discover relevant products.
- Can support business growth.

Example

If a customer frequently searches for laptops and computer accessories, the recommendation system may suggest suitable laptops, keyboards, mice, or laptop bags.

4.3.9 Comparison of Case Studies

Table 4.3.2: Machine Learning Applications in Different Domains

Feature	Healthcare	Finance	E-Commerce
Example	Medical image analysis	Fraud detection	Product recommendation
Common data	Medical images and patient-related data	Transaction data	Customer activity data
Possible model	CNN	Classification models	Recommendation models
Prediction	Possible abnormality	Fraud risk	Suitable products
Important concern	Privacy and reliability	Security and fairness	Privacy and personalisation
Deployment	Healthcare application/ server	Financial system/API	Website/mobile application

SGOU

Summerised Overview

Model deployment is the process of making a trained machine learning or deep learning model available for practical use. Model saving allows a trained model to be stored and reused without retraining. Serialization converts a model into a form that can be stored or transferred, while deserialization restores the model for use. Model deployment involves preparing the deployment environment, loading the saved model, connecting it with an application, providing new input, generating predictions, and monitoring the deployed system.

Models can be deployed locally, on servers, in cloud environments, or at the edge. Predictions may be generated through batch processing or real-time processing. Tools such as Flask, FastAPI, TensorFlow Serving, Docker, and ONNX support different aspects of model deployment. Deployment also involves challenges such as performance, scalability, security, dependency management, changing data, model maintenance, and monitoring.

Machine learning models are applied in several practical domains. In healthcare, CNNs can assist in medical image analysis. In finance, machine learning models can support fraud detection. In e-commerce, recommendation models can provide personalised product suggestions. These applications also require attention to privacy, security, reliability, fairness, monitoring, and interpretability.

SGOU

Assignments

1. Explain the concept of model deployment and describe the major steps involved in the deployment process.
2. Explain model saving, serialization, and model loading with suitable examples.
3. Compare batch prediction and real-time prediction.
4. Explain any four tools used for machine learning model deployment.
5. Discuss the applications of machine learning in healthcare, finance, and e-commerce.

Suggested Reading

1. Masís, S. (2023). Interpretable machine learning with Python (2nd ed.). Packt Publishing

2. *Huyen, C. (2022). Designing machine learning systems: An iterative process for production-ready applications. O'Reilly Media.*

3. *Burkov, A. (2020). Machine learning engineering. True Positive Inc.*

4. *Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.*

Reference

1. Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

2. Raschka, S., Liu, Y., & Mirjalili, V. (2022). *Machine learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing.

3. Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.

4. Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.

5. Sculley, D., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances *in Neural Information Processing Systems 28*.

SGOU

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

4 UNIT Recent Trends and Research in Machine Learning

Learning Outcomes

After completion of this unit, the learner will be able to:

- explain the concepts of Transfer Learning, GANs, AutoML, and Explainable AI.
- describe the working and applications of Transfer Learning and Generative Adversarial Networks.
- apply the concepts of Transfer Learning and AutoML to suitable machine learning problems.
- analyze the advantages and limitations of GANs, AutoML, and Explainable AI.Describe the difference between Q-Learning and SARSA in decision-making.

SGOU

Background

Machine Learning has developed from traditional algorithms into advanced techniques that can learn complex patterns from large and diverse datasets. As machine learning applications continue to grow, researchers are developing new approaches to reduce training time, generate useful data, automate model development, and make machine learning decisions easier for humans to understand. Transfer Learning, Generative Adversarial Networks (GANs), Automated Machine Learning (AutoML), and Explainable Artificial Intelligence (XAI) are important developments in this area. Understanding these techniques helps learners identify how modern machine learning systems can be developed more efficiently and applied to practical problems.

Recent research also focuses on areas such as Generative AI, Federated Learning, Efficient Machine Learning, Multimodal Learning, Human-Centred AI, healthcare

applications, and cybersecurity. Studying these emerging trends helps learners understand the direction in which machine learning is developing and how new technologies are addressing challenges related to data, computational resources, privacy, transparency, and responsible use of AI.

Keywords

Transfer Learning, Pre-trained Model, Fine-tuning, Generative Adversarial Networks (GANs), Generator, Discriminator, Automated Machine Learning (AutoML), Explainable Artificial Intelligence (XAI), Generative AI,Federated Learning, Multimodal Learning.

Discussion

4.4.1 Introduction

Machine Learning has developed rapidly in recent years. Modern machine learning systems are capable of learning complex patterns from images, text, speech, video, and other forms of data. However, developing an effective machine learning system can require large datasets, considerable computational resources, expert knowledge, and considerable time.

Recent research focuses on reducing these limitations and improving the usefulness of machine learning systems. Transfer Learning allows knowledge learned from one task to be reused for another related task. Generative Adversarial Networks (GANs) can generate new data that resembles the training data. Automated Machine Learning (AutoML) automates several stages of machine learning model development. Explainable Artificial Intelligence (XAI) focuses on making machine learning predictions easier for humans to understand.

These techniques have applications in healthcare, finance, education, computer vision, natural language processing, cybersecurity, e-commerce, and many other areas.

4.4.2 Transfer Learning

Transfer Learning is a machine learning technique in which knowledge learned by a model for one task is reused for another related task.

Normally, a machine learning model is trained from the beginning using a dataset specific to the problem. This may require a large amount of labelled data and computational resources. Transfer Learning provides an alternative by using a **pre-trained model** as the starting point for a new task.

SGOU

Example

Consider a system designed to identify different types of flowers. Instead of training a deep neural network from the beginning, a model that has already learned general image features from a large image dataset can be reused.

The existing model may already know how to identify:

- Edges
- Lines
- Shapes
- Textures
- Basic patterns

The model can then be adapted to identify specific types of flowers.

4.4.2.1 Basic Process of Transfer Learning

The general process can be represented as:

Large Dataset → Pre-trained Model → Adapt Model → New Dataset → Fine-tuning → Prediction

1. Pre-trained Model

A pre-trained model is a model that has already been trained on a large dataset for a particular task or a related task.

The learned knowledge can be reused for another problem.

For example, a deep learning model trained on a large collection of images may be adapted for a specific image classification task.

2. Fine-tuning

Fine-tuning is the process of further training a pre-trained model using a new dataset.

During fine-tuning, some or all of the model parameters may be adjusted according to the new task.

For example, a pre-trained image model can be fine-tuned using medical images to develop a system for classifying medical conditions.

4.4.2.2 Advantages of Transfer Learning

Transfer Learning provides several advantages:

1. Reduces training time
The model does not need to learn everything from the beginning.

SGOU

2. Requires less training data
A smaller dataset may be sufficient for the new task.

3. Reduces computational requirements
Existing knowledge can reduce the amount of training required.

4. Improves model performance
Knowledge from a related task can help the new model learn useful features.

5. Useful for specialized applications
It can be useful when labelled data for a particular application is limited.

4.4.3 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a deep learning architecture designed to generate new data that resembles data from a given training dataset.

GANs were introduced by Ian Goodfellow and his collaborators in 2014.

A GAN consists mainly of two neural networks:
1. Generator
2. Discriminator

These two networks compete with each other during training.

4.4.3.1 Generator

The Generator creates new synthetic data.

For example, when trained using a dataset of human faces, the generator attempts to create new face images that look realistic.

The generator tries to produce data that the discriminator cannot distinguish from real data.

4.4.3.2 Discriminator

The **Discriminator** attempts to distinguish between real data and data generated by the generator.

It receives:
- Real data from the training dataset
- Generated data from the generator

The discriminator then determines whether the input is real or generated.

4.4.3.3 Working of GAN

The basic working of a GAN can be represented as:

Random Input → Generator → Synthetic Data

Real Data + Synthetic Data → Discriminator → Real / Fake

SGOU

The generator attempts to improve its ability to produce realistic data, while the discriminator attempts to improve its ability to identify generated data.

As training continues, the generator can produce increasingly realistic outputs.

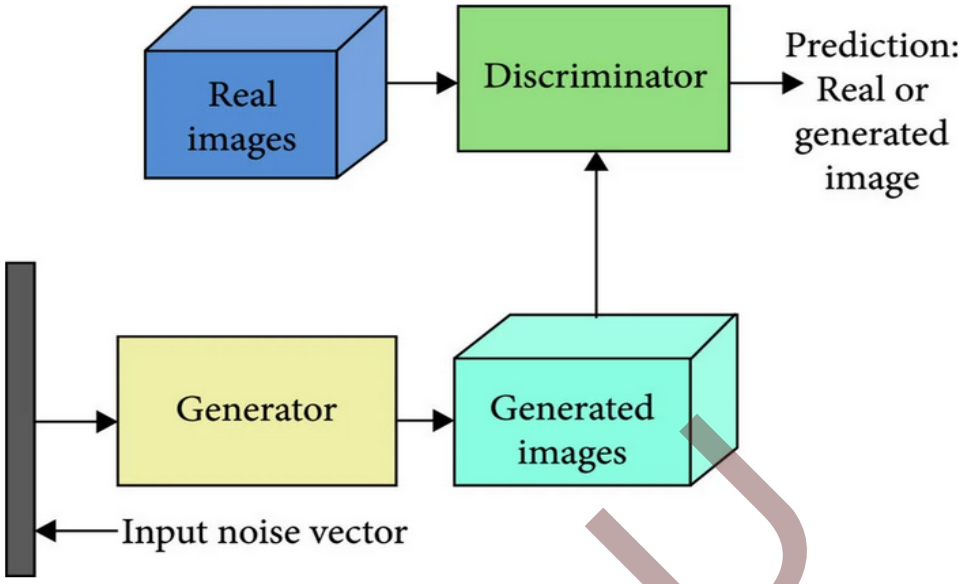


Fig. 4.4.1 Generative Adversarial Network (GAN) Architecture

4.4.3.4 Applications of GANs

GANs can be used in:

1. Image Generation

GANs can generate realistic images based on learned patterns.

2. Image-to-Image Translation

GANs can transform images from one style or representation into another.

3. Data Augmentation

Synthetic data can be generated to supplement limited training data.

4. Image Restoration

GAN-based techniques can be used for image enhancement and restoration.

5. Synthetic Media

GANs can generate synthetic images, videos, and other media.

6. Healthcare

GANs can be used to generate synthetic medical images for research and data augmentation.

SGOU

4.4.3.5 Advantages of GANs

- Can generate realistic synthetic data.
- Useful when training data is limited.
- Can support data augmentation.
- Useful in image generation and transformation.
- Can be applied to different types of generative tasks.

4.4.3.6 Limitations of GANs

GANs also have several challenges:

- Training can be difficult.
- The generator and discriminator must be balanced.
- Training may become unstable.
- The generator may produce limited varieties of outputs.
- Generated content may raise ethical and security concerns.

4.4.4 Automated Machine Learning

Automated Machine Learning (AutoML) refers to techniques and tools that automate parts of the machine learning development process.

Building a machine learning model traditionally involves several manual activities such as:

- Data preprocessing
- Feature selection
- Model selection
- Hyperparameter tuning
- Model evaluation

AutoML attempts to automate some of these activities.

4.4.4.1 Need for AutoML

Machine learning development requires technical knowledge and can be time-consuming.

For example, a developer may need to compare several algorithms and determine suitable parameter values.

SGOU

AutoML can help automate these tasks and allow users to develop effective machine learning models with less manual effort.

4.4.4.2 Core Architecture & Pipeline

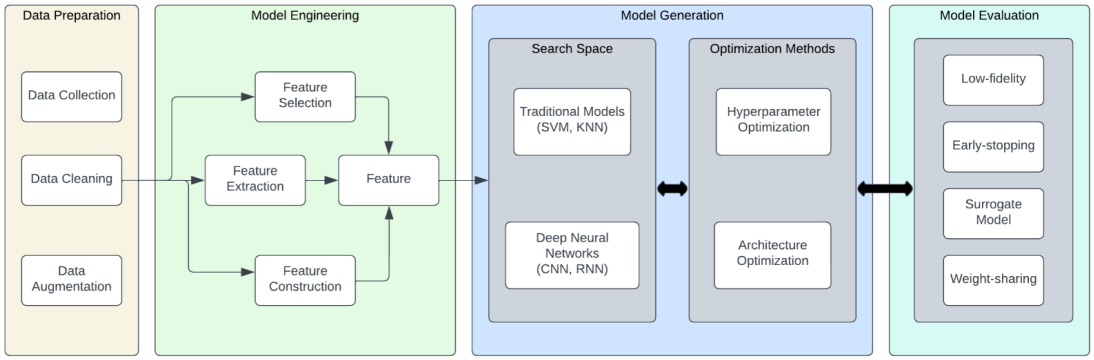


Fig. 4.4.2 Architecture of AutoML

1. Data Preparation & Preprocessing

- **Missing Value Imputation:** Fills null values using mean, median, or predictive models.
- **Encoding Categorical Data:** Automatically applies One-Hot Encoding, Target Encoding, or Label Encoding based on feature cardinality.
- **Scaling & Normalization:** Standardizes numeric variables (e.g., StandardScaler, MinMax).

2. Automated Feature Engineering (AutoFE)

- **Feature Generation:** Creates polynomial features, ratios, date component extractions, or aggregations.
- **Feature Selection:** Filters out low-variance, redundant, or non-informative features using statistical tests or importance scores.

3. Neural Architecture Search (NAS) & Model Selection

- **Search Space Definition:** Specifies candidate algorithms (e.g., XGBoost, LightGBM, Random Forest, ResNet).
- **Neural Architecture Search (NAS):** Automates the design of deep neural network layouts (layers, operations, connections).

4. Hyperparameter Optimization (HPO)

AutoML tests and tunes model settings using optimization strategies:

- **Grid / Random Search:** Simple, baseline exploration.
- **Bayesian Optimization:** Uses probabilistic surrogate models (e.g., Gaussian Processes or Tree-structured Parzen Estimators) to pick optimal parameters.

SGOU

- **Evolutionary Algorithms:** Uses genetic algorithms to mutate and select candidate hyperparameters.

5. Model Evaluation & Ensembling

- **Validation Strategy:** Uses K-Fold Cross-Validation to evaluate models without data leakage.
- **Stacking / Blending:** Combines multiple high-performing models (ensembling) into a single predictor for superior metrics.

4.4.4.3 Advantages of AutoML

- Reduces manual effort.
- Saves development time.
- Helps compare multiple models.
- Can support users with limited machine learning expertise.
- Automates repetitive tasks.
- Supports faster experimentation.

4.4.4.4 Limitations of AutoML

- Automated systems may require considerable computational resources.
- Results still need appropriate evaluation.
- Users need to understand the problem and data.
- Automated model selection does not guarantee the best solution in every situation.
- Interpretability may be difficult for complex models.

4.4.5 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) refers to methods and techniques that help humans understand how an AI or machine learning system reaches its predictions or decisions.

Many modern machine learning models, particularly deep learning models, can be difficult to understand because their decision-making processes involve a large number of parameters.

Such models are sometimes referred to as black-box models.

For example, suppose a machine learning system predicts that a patient has a high risk of a particular disease. The healthcare professional may want to know why the model produced this prediction.

SGOU

XAI attempts to provide understandable information about the factors that influenced the prediction.

4.4.5.1 Need for Explainable AI

Explainability is important because machine learning systems are increasingly used in areas where decisions can have significant consequences.

XAI can help:

- Improve user trust.
- Understand model decisions.
- Identify possible errors.
- Detect bias.
- Support debugging.
- Improve accountability.
- Assist decision-making.

4.4.5.2 Interpretability and Explainability

Interpretability refers to how easily a human can understand the relationship between the input and output of a model.

Explainability refers to methods used to provide understandable reasons or explanations for model predictions.

For example, a simple decision tree may be relatively easy to interpret because its decision rules can be directly examined. A complex deep neural network may require additional explanation techniques.

4.4.5.3 Common Approaches to Explainability

Some commonly used approaches include:

Feature Importance

Feature importance identifies which input features have a significant influence on a model's prediction.

For example, in a loan approval model, factors such as income, credit history, and existing debt may influence the prediction.

Local Explanations

A local explanation attempts to explain the prediction of a particular individual input.

For example, it may explain why one particular loan application was classified as high risk.

SGOU

Global Explanations

A global explanation attempts to describe the general behaviour of a model across a dataset.

4.4.5.4 Benefits of Explainable AI

- Improves transparency.
- Supports user trust.
- Helps identify errors.
- Helps detect unwanted bias.
- Supports regulatory and organizational requirements.
- Makes AI systems easier to evaluate.
- Helps domain experts understand model predictions.

4.4.5.5 Challenges of Explainable AI

Some important challenges are:

- Explanations may be difficult for non-technical users.
- Complex models may be difficult to explain accurately.
- Different explanation methods may produce different interpretations.
- An explanation may simplify a complex model too much.
- There can be a trade-off between model complexity and interpretability.

4.4.6 Comparison of Recent Machine Learning Trends

Table 4.4.1: Comparison of Transfer Learning, GANs, AutoML and XAI

Feature	Transfer Learning	GANs	AutoML	Explainable AI
Main purpose	Reuse learned knowledge	Generate new data	Automate ML development	Explain model decisions
Main components	Pre-trained model and new task	Generator and discriminator	Automated ML processes	Explanation techniques
Major benefit	Reduces training requirements	Generates synthetic data	Reduces manual effort	Improves transparency

SGOU

Major application	Image and NLP tasks	Image and media generation	Model development	High-stakes decision making
Main challenge	Task mismatch	Training instability	Computational cost	Explanation complexity

4.4.7 Current Research Trends in Machine Learning

Machine Learning research continues to develop rapidly. Current research focuses on improving model performance, reducing computational requirements, increasing reliability, and making AI systems more useful and responsible.

Important research directions include the following.

1. Generative AI

Generative AI focuses on systems that can create new content such as:

- Text
- Images
- Audio
- Video
- Computer code

Generative models are becoming increasingly important in creative, educational, business, and scientific applications.

2. Explainable and Responsible AI

Research increasingly focuses on making AI systems:

- Transparent
- Fair
- Reliable
- Safe
- Accountable

This is particularly important when AI is used in healthcare, finance, education, and public services.

3. Efficient Machine Learning

Researchers are developing methods to reduce:

- Computational requirements
- Memory usage
- Energy consumption
- Training time

SGOU

This helps machine learning systems operate effectively on devices with limited resources.

4. Federated Learning

Federated Learning allows machine learning models to be trained across multiple devices or organizations without requiring all training data to be transferred to a central location.

It is useful in situations where data privacy is important.

5. Multimodal Learning

Multimodal Learning combines different types of data such as:

- Text
- Images
- Audio
- Video

A multimodal system can use information from multiple sources to improve understanding and prediction.

6. Human-Centred AI

Research is increasingly focusing on developing AI systems that work effectively with humans rather than simply replacing human decision-making.

7. AI for Healthcare

Machine learning research continues to explore applications such as:

- Medical image analysis
- Disease prediction
- Drug discovery
- Patient risk prediction
- Personalised healthcare

8. AI for Cybersecurity

Machine learning is being studied for:

- Intrusion detection
- Malware detection
- Fraud detection
- Anomaly detection
- Network security

SGOU

Summerised Overview

Machine Learning is continuously evolving to address new challenges in data processing, model development, efficiency, and responsible use of Artificial Intelligence. Transfer Learning enables a pre-trained model to be adapted to a related task, reducing the need for large datasets and extensive training. Generative Adversarial Networks (GANs) use a generator and discriminator to create realistic synthetic data and have applications in image generation, data augmentation, and image restoration.

Automated Machine Learning (AutoML) automates important stages of machine learning development, such as data preprocessing, feature selection, model selection, hyperparameter optimisation, and model evaluation. Explainable Artificial Intelligence (XAI) helps users understand machine learning predictions and improves transparency, trust, and accountability. The unit also introduces current research trends such as Generative AI, Federated Learning, Efficient Machine Learning, Multimodal Learning, Human-Centred AI, AI for Healthcare, and AI for Cybersecurity. These developments show how modern machine learning is becoming more efficient, explainable, privacy-aware, and useful across different application areas.

SGOU

Assignments

1. Explain the concept of Transfer Learning and describe its advantages and applications.
2. Explain the architecture and working of Generative Adversarial Networks.
3. Describe Automated Machine Learning and explain its major components, advantages, and limitations.
4. Explain the need for Explainable AI and discuss its major approaches and benefits.
5. Discuss the important current research trends in Machine Learning and Artificial Intelligence.

6. A hospital has a small collection of labelled medical images and wants to develop an image classification system. Analyse how Transfer Learning can help the hospital develop the system and discuss its advantages and limitations.

Suggested Reading

1. Tomczak, J. M. (2024). *Deep generative modeling*. .Springer.

2. Wang, J., & Chen, Y. (2023). *Introduction to transfer learning: Algorithms and practice*. Springer.

3. Mishra, P. (2023). *Explainable AI recipes: Implement solutions to model explainability and interpretability with Python*.

4. Raschka, S., Liu, Y. H., & Mirjalili, V. (2022). *Machine learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing.

5. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

SGOU

Reference

1. Huisman, M., Plaat, A., & van Rijn, J. N. (2024). Understanding transfer learning and gradient-based meta-learning techniques. *Machine Learning*.

2. Tripathi, S. L., & Mahmud, M. (Eds.). (2023). *Explainable machine learning models and architectures*. Wiley-Scrivener.

3. Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

4. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering,*

Space for Learner Engagement for Objective Questions

Learners are encouraged to develop objective questions based on the content in the paragraph as a sign of their comprehension of the content. The Learners may reflect on the recap bullets and relate their understanding with the narrative in order to frame objective questions from the given text. The University expects that 1 - 2 questions are developed for each paragraph. The space given below can be used for listing the questions.

SGOU

MODEL QUESTION PAPER SETS

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

Model Question Paper (SET- A)

QP CODE: …… Reg. No:.....

Name: ……

MASTER OF COMPUTER APPLICATION, SECOND SEMESTER
END SEMESTER EXAMINATION
DISCIPLINE CORE COURSE

M25CA06DC - Machine Learning

2025-26 - Admission

Time: 3 Hours Max Marks: 70

Section A

Answer any ten of the following questions. Each answer should be in two or three valid points. Each question carries 2 marks.

(10 x 2 = 20)

1. Write any 3 real-world applications of ML.
2. Write about reinforcement learning.
3. Write about confusion matrix.
4. Write about common Filter Methods in feature selection.
5. What is the k-Nearest Neighbours (k-NN) algorithm?
6. Define Logistic Regression and mention its application.
7. What is the kernel trick in Support Vector Machines (SVM)?
8. What is k-Fold Cross-Validation?
9. What is agglomerative clustering? Mention one limitation.
10. Write any two key characteristics of PCA.
11. List the steps involved in the LDA algorithm.
12. Differentiate between Apriori and FP-Growth algorithms.
13. What is the role of hidden layers in a neural network?
14. Define Transfer Learning
15. What is the purpose of the pooling layer in a CNN?

SGOU

Section B

Answer any six of the following questions. Provide adequate valid points with clear explanations, proportionate to the marks allotted. Each question carries 5 marks.

(6 x 5 = 30)

16. Explain about any two evaluation matrices.

17. Write about data preprocessing.

18. Explain the working of a Decision Tree algorithm with a suitable example.

19. Explain Bagging and Boosting techniques used in ensemble learning.

20. Compare K-Means, Hierarchical Clustering, and DBSCAN.

21. What is Singular Value Decomposition (SVD)? Write its mathematical representation.

22. Explain the structure and working of an LSTM network with reference to its gates and cell state

23. Explain the steps involved in deploying a trained machine learning model

24. Describe the architecture and working of a Generative Adversarial Network (GAN)

Section C

Answer any two of the following questions. Provide adequate valid points with clear explanations, proportionate to the marks allotted. Each question carries 10 marks.

(2x10=20)

25. Explain about unsupervised ML techniques.

26. Explain the process of hyperparameter tuning using Grid Search and k-Fold Cross-Validation. Discuss how these techniques help improve the performance of a machine learning model.

27. Explain Reinforcement Learning, its components, and learning cycle.

28. Explain the architecture and working of an Artificial Neural Network (ANN). Discuss the roles of input, hidden, and output layers, weights, biases, and activation functions with a suitable diagram

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

Model Question Paper (SET- B)

QP CODE: …… Reg. No:.....

Name: ……

MASTER OF COMPUTER APPLICATION, SECOND SEMESTER
END SEMESTER EXAMINATION
DISCIPLINE CORE COURSE

M25CA06DC - Machine Learning

2025-26 - Admission

Time: 3 Hours Max Marks: 70

Section A

Answer any ten of the following questions. Each answer should be in two or three valid points. Each question carries 2 marks.

(10 x 2 = 20)

1. Write any three applications of machine learning.
2. Write about LASSO.
3. Write about common wrapper Methods in feature selection.
4. Write the purposes of data cleaning?
5. 5.What is Naive Bayes classification?
6. Define Mean Absolute Error (MAE).
7. What is the purpose of a kernel function in SVM?
8. What is the Random Forest algorithm?
9. Mention any two benefits of dimensionality reduction.
10. What is a dendrogram? Mention its purpose in hierarchical clustering.
11. What is Market Basket Analysis? Mention its purpose.
12. What is the SARSA algorithm?
13. Define an Artificial Neural Network (ANN)
14. What is a Convolutional Neural Network (CNN)
15. What is a Generative Adversarial Network (GAN)

SGOU

Section B

Answer any six of the following questions. Provide adequate valid points with clear explanations, proportionate to the marks allotted. Each question carries 5 marks.

(6 x 5 = 30)

16. Explain different techniques of data transformation.

17. Write in detail about accuracy and precision evaluation metrics.

18. Explain the working of the k-NN algorithm with a suitable example.

19. Differentiate between Linear Regression and Polynomial Regression.

20. What is t-SNE? Explain the advantages of t-SNE.

21. Compare the different clustering evaluation metrics

22. Explain the different layers of a feedforward neural network and their functions

23. Explain the architecture of a Recurrent Neural Network (RNN) and its applications

24. Compare sigmoid, tanh, ReLU, and Leaky ReLU activation functions

Section C

Answer any two of the following questions. Provide adequate valid points with clear explanations, proportionate to the marks allotted. Each question carries 10 marks.

(2x10=20)

25. Explain about supervised ML techniques.

26. Explain the working principles of Support Vector Machines (SVM), including hyperplanes and the kernel trick. Discuss how SVM can handle both linearly separable and non-linearly separable data.

27. Explain the DBSCAN clustering algorithm, including its key concepts, working steps, advantages, and limitations.

28. Explain the architecture of an LSTM network and the functions of its gates

SGOU

സർവ്വകലാശാലാഗീതം

വിദ്യയാൽ സ്വതന്ത്രരാകണം
വിശ്വപൗരരായി മാറണം
ഗ്രഹപ്രസാദമായ് വിളങ്ങണം
ഗുരുപ്രകാശമേ നയിക്കണേ

കൂരിരുട്ടിൽ നിന്നു ഞങ്ങളെ
സൂര്യവീഥിയിൽ തെളിക്കണം
സ്നേഹദീപ്തിയായ്വിളങ്ങണം
നീതിവൈജയന്തി പാറണം

ശാസ്ത്രവ്യാപ്തിയെന്നുമേകണം
ജാതിഭേദമാകെ മാറണം
ബോധരശ്മിയിൽ തിളങ്ങുവാൻ
ജ്ഞാനകേന്ദ്രമേ ജ്വലിക്കണേ

കുരീപ്പുഴ ശ്രീകുമാർ

SGOU

SREENARAYANAGURU OPEN UNIVERSITY

Regional Centres

Kollam
Sreenarayanaguru Open University
Headquarters Building
BSNL Bhavan, Vellayittambalam,
Kollam, Kerala, Pin : 691012
Ph: 9188912554
email:rckollam@sgou.ac.in

Tripunithura
Govt. College
Tirpunithura, Ernakulam
Kerala, Pin : 682301
Ph: 04842927436
email:rcedirector@sgou.ac.in

Thalassery
Govt. Brennen College
Dharmadam, Thalassery
Kannur, Pin : 670106
Ph: 04952990494
email:rctdirector@sgou.ac.in

Pattambi
Sree Neelakanta Govt. Sanskrit College
Pattambi, Palakkad
Kerala, Pin : 679303
Ph: 04662912009
email:rcpdirector@sgou.ac.in

Kozhikode
Govt. Arts and Science College
Meenchantha, Kozhikode,
Kerala, Pin : 673002
Ph: 04952920228
email:rckdirector@sgou.ac.in

SAY NO TO RAGGING

BEFORE YOU EVEN THINK OF RAGGING



Humiliation

Blacklisting

Ruined Career

Download

ANTI RAGGING

App

Don't just stand and watch. Stop Ragging! Show Character

Visit UGC Website i.e. www.ugc.Ac.in & www antinaging.in to UGC Anti Ragging regulations.

Ministry of Education
Government of India

Are You Being Ragged ?

विश्वविद्यालय अनुदान आयोग
University Grants Commission
quality higher education for all

DON'T LET IT BE TOO LATE

SAY NO TO DRUGS

LOVE YOURSELF AND ALWAYS BE HEALTHY



SREENARAYANAGURU OPEN UNIVERSITY

The State University for Education, Training and Research in Blended Format, Kerala

SGOU

MACHINE LEARNING

COURSE CODE: M25CA06DC

SGOU

Sreenarayanaguru Open University
Kollam, Kerala Pin- 691601, email: info@sgou.ac.in, www.sgou.ac.in Ph: +91 474 2966841

ISBN 978-81-181574-2-7