MALLA REDDY COLLEGE OF ENGINEERING FOR WOMEN

Approved by AICTE, New Delhi; Affiliated to JNTU, Hyderabad.

An ISO 9001:2015 Certified Institution

Maisammaguda, Dhulapally, Medchal, Hyderabad -500100,

Telangana.

COURSE FILE

OF

MACHINE LEARNING

YEAR: **III B. Tech** SEMESTER: **I** BRANCH: **CSE**

Regulation R22

ACADEMIC YEAR: 2024-2025

Prepared by

Dr. Syeda Husna Mehanoor

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

R22 B.Tech. CSE Syllabus JNTU Hyderabad **CS601PC: MACHINE LEARNING B. Tech III Year II Sem.**

L T P C

3 0 0 3

Course Objectives:

- To introduce students to the basic concepts and techniques of Machine Learning.
- To have a thorough understanding of the Supervised and Unsupervised learning techniques
- To study the various probability-based learning techniques

Course Outcomes:

- Distinguish between, supervised, unsupervised and semi-supervised learning
- Understand algorithms for building classifiers applied on datasets of non-linearly separable classes
- Understand the principles of evolutionary computing algorithms
- Design an ensembler to increase the classification accuracy

UNIT - I

Learning – Types of Machine Learning – Supervised Learning – The Brain and the Neuron – Design a Learning System – Perspectives and Issues in Machine Learning – Concept Learning Task – Concept Learning as Search – Finding a Maximally Specific Hypothesis – Version Spaces and the Candidate Elimination Algorithm – Linear Discriminants: – Perceptron – Linear Separability – Linear Regression.

UNIT - II

Multi-layer Perceptron– Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

UNIT - III

Learning with Trees – Decision Trees – Constructing Decision Trees – Classification and Regression Trees – Ensemble Learning – Boosting – Bagging – Different ways to Combine Classifiers – Basic Statistics – Gaussian Mixture Models – Nearest Neighbor Methods – Unsupervised Learning – K means Algorithms

UNIT - IV

Dimensionality Reduction – Linear Discriminant Analysis – Principal Component Analysis – Factor Analysis – Independent Component Analysis – Locally Linear Embedding – Isomap – Least Squares Optimization Evolutionary Learning – Genetic algorithms – Genetic Offspring: - Genetic Operators – Using Genetic Algorithms

UNIT - V

Reinforcement Learning – Overview – Getting Lost Example Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – Tracking Methods

TEXT BOOKS:

1. Stephen Marsland, ―Machine Learning – An Algorithmic Perspective, Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014

UNIT - I

Learning – Types of Machine Learning – Supervised Learning – The Brain and the Neuron – Design a Learning System – Perspectives and Issues in Machine Learning – Concept Learning Task – Concept Learning as Search – Finding a Maximally Specific Hypothesis – Version Spaces and the Candidate Elimination Algorithm – Linear Discriminants: – Perceptron – Linear Separability – Linear Regression.

LEARNING

Definition of learning
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience

Examples
i) Handwriting recognition learning problem
• Task T: Recognising and classifying handwritten words within images
• Performance P: Percent of words correctly classified
• Training experience E: A dataset of handwritten words with given classifications

ii) A robot driving learning problem
• Task T: Driving on highways using vision sensors
• Performance measure P: Average distance traveled before an error
• training experience: A sequence of images and steering commands recorded while observing a human driver

iii) A chess learning problem
• Task T: Playing chess
• Performance measure P: Percent of games won against opponents
• Training experience E: Playing practice games against itself

Therefore, a computer program which learns from experience is called a machine learning program or simply a learning program. Such a program is sometimes also referred to as a learner.

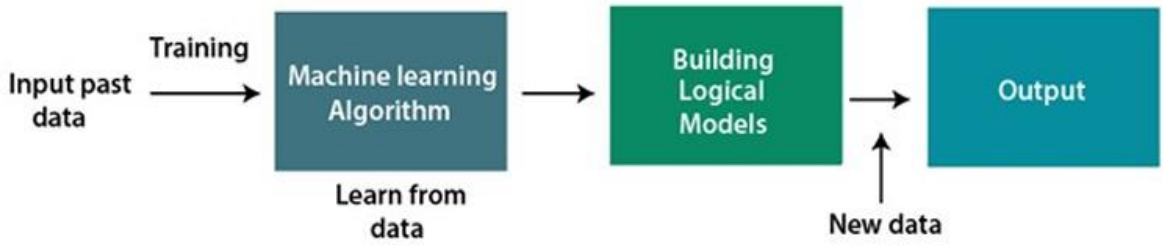
Machine Learning
Machine learning enables a machine to automatically learn from data, prove performance from experiences, and predict things without being explicitly programmed.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately. Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data+ and predict the output.

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term “Machine Learning” in 1959 while at IBM. He defined machine learning as “the field of study that gives computers the ability to learn without being explicitly programmed.” However, there is no universally accepted definition for machine learning. Different authors define the term differently.

How does Machine Learning works

A machine learning system builds prediction models, learns from previous data, and predicts the output of new data whenever it receives it. The amount of data helps to build a better model that accurately predicts the output, which in turn affects the accuracy of the predicted output. Let's say we have a complex problem in which we need to make predictions. Instead of writing code, we just need to feed the data to generic algorithms, which build the logic based on the data and predict the output. Our perspective on the issue has changed as a result of machine learning. The Machine Learning algorithm's operation is depicted in the following block diagram



Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

NEED OF MACHINE LEARNING

- Following are some key points which show the importance of Machine Learning:
- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data

Applications of Machine Learning

Application	Description
Facial Recognition	Image recognition used for face detection, pattern recognition, and personal identification.
Image Recognition	Identifies objects, people, and places in digital images (e.g., Facebook's auto-tagging feature).
Speech Recognition	Converts voice instructions into text (e.g., Google Assistant, Siri, Alexa, Cortana).
Sentiment Analysis	Determines emotions or opinions in text, used in reviews, emails, and decision-making applications.
Self-Driving Cars	Uses unsupervised learning to detect people, objects, and navigate autonomously (e.g., Tesla).
Product Recommendations	Tracks user behavior to suggest products on e-commerce platforms based on past purchases and browsing history.
Fraud Detection	Detects fraudulent transactions using neural networks to identify fake accounts and unauthorized activities.
Chatbots	AI-driven chat assistants provide customer support, collect insights, and enhance user engagement.
Medical Image Analysis	Assists in diagnosing diseases (e.g., cancer detection in X-rays, MRIs, and CT scans).
Natural Language Processing (NLP)	Enables machines to understand and generate human language (e.g., chatbots, translation, text summarization).
Predictive Maintenance	Analyzes sensor data to predict equipment failures, reducing downtime and optimizing maintenance.
Financial Trading	Uses ML models to analyze market data, predict stock trends, and automate trading strategies.

Application	Description
Cybersecurity	Detects anomalies, identifies threats, and prevents cyberattacks such as malware and phishing.

Advantages & Disadvantages of Machine Learning

Advantages	Examples	Explanation
Automation of Complex Tasks	Object detection in self-driving cars	ML algorithms can identify objects like pedestrians, traffic lights, and other vehicles in real-time, enabling autonomous navigation.
Enhanced Accuracy and Precision	Medical image analysis for disease diagnosis	ML models detect subtle patterns in medical images with high accuracy, aiding in early diagnosis.
Improved Efficiency and Scalability	Facial recognition in security systems	ML processes large volumes of images quickly, making it ideal for real-time surveillance.
Adaptability and Continuous Learning	Image classification for product categorization in e-commerce	ML models improve over time by adapting to new data, ensuring accurate categorization of new products.
Uncovering Hidden Insights	Image analysis for market research	ML analyzes large datasets to identify trends and patterns that might not be apparent to human analysts.

Disadvantages	Examples	Explanation
Data Dependency	Facial recognition system trained mainly on Caucasian faces	If training data is biased, incomplete, or noisy, model performance is negatively impacted.
High Computational Costs	Training large-scale image classification models	Deep learning models require significant processing power and time, sometimes taking days or weeks to train.
Lack of Transparency (Black Box Problem)	Unexpected decisions by self-driving cars	Complex ML models make it difficult to interpret decision-making processes, leading to

Disadvantages	Examples	Explanation
		trust issues.
Potential for Bias	AI-driven job recruitment system	Models trained on biased historical data may reinforce existing discrimination in hiring practices.
Security Vulnerabilities	Adversarial attacks on self-driving cars	Malicious actors can manipulate input data (e.g., road signs) to deceive ML models, causing incorrect predictions.

TYPES OF LEARNING

Here are brief definitions for different types of machine learning:

1. **Supervised Learning**: A type of machine learning where the model is trained on labeled data, meaning both input and output are provided. Example: Spam email detection.
2. **Unsupervised Learning**: The model learns patterns and structures from unlabeled data without explicit outputs. Example: Customer segmentation.
3. **Semi-Supervised Learning**: Combines aspects of both supervised and unsupervised learning by using a small amount of labeled data along with a large amount of unlabeled data. Example: Medical diagnosis with limited labeled samples.
4. **Reinforcement Learning**: The model learns through trial and error by interacting with an environment and receiving rewards or penalties. Example: Training an AI to play chess.
5. **Evolutionary Learning**: A type of machine learning inspired by natural selection, where algorithms evolve over generations by selecting the best solutions and applying mutations or crossovers. Example: Genetic algorithms used for optimization problems.

SUPERVISED LEARNING

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

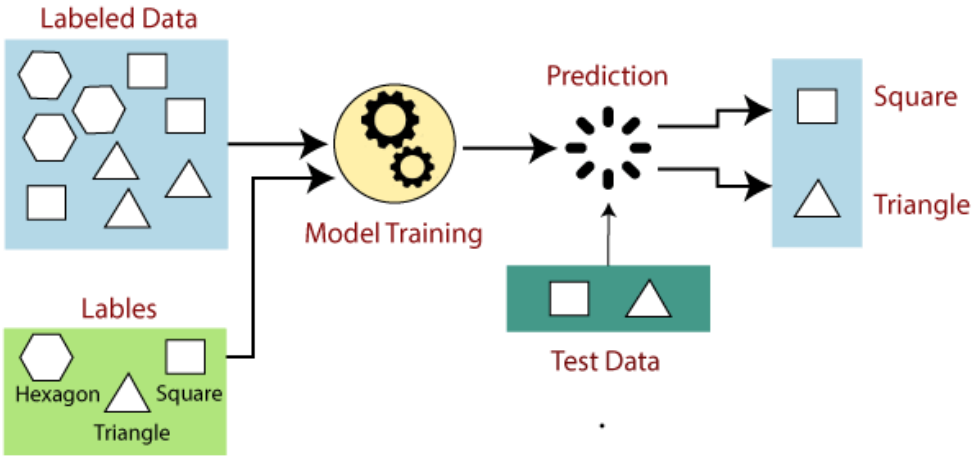
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y). In the real-world, supervised

learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a **Square**.
- If the given shape has three sides, then it will be labelled as a **triangle**.
- If the given shape has six equal sides then it will be labelled as **hexagon**.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

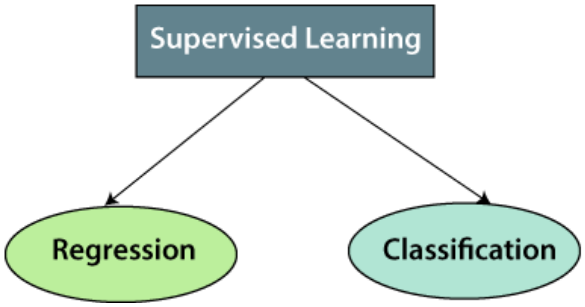
Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset**.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.

- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:



1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc. Below are some popular Classification algorithms which come under supervised learning:

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

THE BRAIN AND THE NEURON

The brain is an amazing system that can handle messy and complicated information (like pictures) and give quick and accurate answers. It's made up of simple building blocks called neurons, which send signals when activated. These signals travel through connections called synapses, creating a huge network of about 100 trillion links. Even as we age and lose neurons, the brain keeps working well.

Each neuron acts like a tiny decision-maker in a massive network of 100 billion neurons. This has inspired scientists to create AI systems that try to copy how the brain learns. The brain learns by changing the strength of its connections i:e plasticity which refers to its ability to change and adapt by modifying the strength of the connections (called synapses) between neurons or forming new connections altogether. This is how the brain learns and remembers things and forming new connections between neurons in the brain. One famous idea, suggested by Donald Hebb in 1949, is that learning happens when neurons that frequently work together strengthen their connection.

Hebb's Rule

Hebb's rule is a simple idea: if two neurons fire at the same time repeatedly, their connection becomes stronger. On the other hand, if they never fire together, their connection weakens and might disappear. This is how the brain learns to associate things.

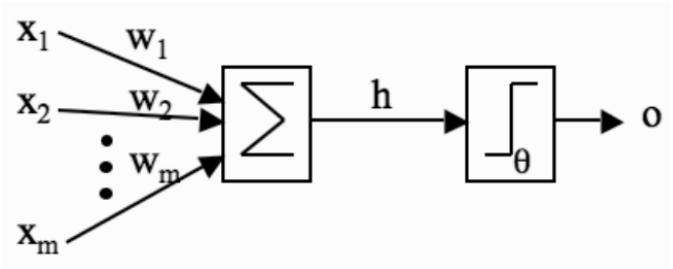
Here's an example: Imagine you always see your grandmother when she gives you chocolate. Neurons in your brain that recognize your grandmother and neurons that make you happy about chocolate will fire at the same time. Over time, their connection strengthens. Eventually, just seeing your grandmother (even in a photo) makes you think of chocolate. This is similar to **classical conditioning**, where Pavlov trained dogs to associate a bell with food. When the bell

and food were paired repeatedly, the dogs began to salivate at the sound of the bell alone because the "bell" neurons and "salivation" neurons became strongly connected.

This idea is called **long-term potentiation** or **neural plasticity**, and it's a real process in our brains that helps us learn and form memories.

McCulloch and Pitts Neurons

Scientists have studied neurons and created a mathematical model of them to simplify understanding. Real neurons are tiny and hard to study, but Hodgkin and Huxley studied large neurons in squids to measure how they work, earning them a Nobel Prize. Later, McCulloch and Pitts created a simplified model of a neuron in 1943 that focused on the essential parts.



Understanding the McCulloch and Pitts Neuron Model

Imagine the neuron model as a simple flowchart with three main parts:

1. **Inputs with Weights**
2. **Summation (Adder)**
3. **Activation Function (Threshold)**
4. **Output**

Here's a breakdown of each part:

1. Inputs with Weights (w_1, w_2, w_3, ...)

- **Inputs (x_1, x_2, x_3, ...):** These are signals coming into the neuron from other neurons. Think of them as messages or pieces of information.
- **Weights (w_1, w_2, w_3, ...):** Each input has a weight that represents the strength or importance of that input. A higher weight means the input has a stronger influence on the neuron's decision to fire.

Example:

- x1=1 (active)
- x2=0 (inactive)
- x3=0.5 (partially active)
- Weights: w1=1, w2=−0.5, w3=−1

2. **Summation (Adder)**

- The neuron adds up all the inputs after they've been multiplied by their respective weights.
- **Formula:** $h=w1x1+w2x2+w3x3+\ldots$

Using the Example:

- $h = (1\times1)+(0\times-0.5)+(0.5\times-1)=1+0+(-0.5)=0.5$
- $h=0.5$

3. **Activation Function (Threshold θ)**

- After summing the inputs, the neuron decides whether to "fire" (send a signal) or not based on a threshold value (θ).
- **Decision Rule:**
  - If $h > \theta$, the neuron fires (output = 1)
  - If $h \leq \theta$, the neuron does not fire (output = 0)

Using the Example:

- Let's set $\theta=0$
- Since $h=0.5>0$, the neuron fires (output = 1)

4. **Output**

- The result of the activation function is the neuron's output, which can be sent to other neurons.

Key Features:

- **Simple Decision-Maker:** Despite its simplicity, this model can perform basic decisions based on input signals.
- **Foundation for Neural Networks:** Multiple such neurons can be connected to form complex networks capable of more advanced computations.
- **Adjusting Weights:** Learning in neural networks involves adjusting these weights to improve decision-making based on data.

Real-World Analogy: Think of the neuron as a light switch system

- **Inputs (x_1, x_2, x_3):** Different sensors detecting things (like motion, light, sound).
- **Weights (w_1, w_2, w_3):** The importance of each sensor in deciding whether to turn on the light.
- **Summation (h):** Adding up the signals from all sensors.
- **Threshold (θ):** The level of combined signals needed to decide to turn the light on.

- **Output:** The light is either on (1) or off (0).

By adjusting the weights, you can make the system more or less sensitive to certain sensors, just like training a neural network to recognize patterns.

Limitations of the McCulloch and Pitts Neuronal Model

The McCulloch and Pitts (M&P) neuron model is a simplified version of how real neurons work. While it has been influential in early neural network models, it has several limitations when compared to actual biological neurons.

1. **Simplified Summing**: In the McCulloch and Pitts model, inputs to the neuron are simply added together in a linear fashion. Real neurons, however, may have non-linear interactions, meaning their inputs don't just add up but interact in more complex ways.
2. **Single Output vs. Spike Train**: The M&P neuron produces just one output, either firing or not firing, based on a threshold. Real neurons, however, send out a series of pulses, called a "spike train," to represent information. So, real neurons don't just decide whether to fire or not—they generate a sequence of signals that encode data.
3. **Changing Thresholds**: In the M&P model, the threshold for firing is constant. In real neurons, the threshold can change depending on the current state of the organism, like how much neurotransmitter is available, which influences the neuron's sensitivity.
4. **Asynchronous vs. Synchronous Updates**: The M&P model updates neurons in a regular, clocked sequence (synchronously). Real neurons don't work this way; they update asynchronously, meaning they fire at different times, influenced by random factors, not just a regular time cycle.
5. **Excitatory and Inhibitory Weights**: The M&P model allows weights (connections between neurons) to change from positive to negative, which isn't seen in real neurons. In the brain, synaptic connections are either excitatory (increase the likelihood of firing) or inhibitory (decrease the likelihood of firing), and they don't switch from one type to the other.
6. **Feedback Loops**: Real neurons can have feedback connections where a neuron connects back to itself. The M&P model typically doesn't include this, although it's a feature in some more advanced models.
7. **Biological Complexity Ignored**: The M&P model focuses on the basic idea of deciding whether a neuron fires or not, leaving out more complex biological factors, such as chemical concentrations or refractory periods (the time it takes for a neuron to reset before firing again).

DESIGN A LEARNING SYSTEM

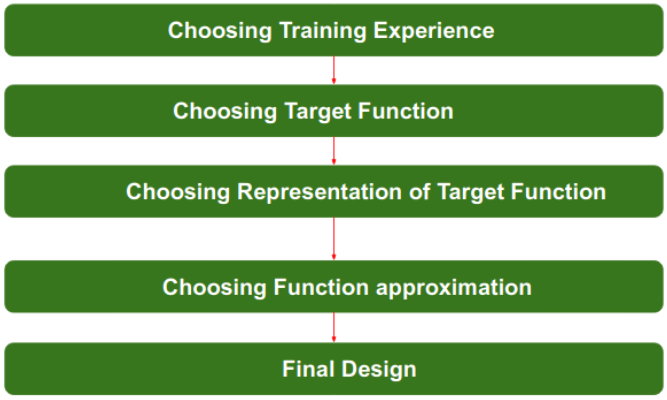
According to Tom Mitchell, "A computer program is said to be learning from experience (E), with respect to some task (T). Thus, the performance measure (P) is the performance at task T, which is measured by P, and it improves with experience E."

Example: In Spam E-Mail detection,

- **Task, T:** To classify mails into Spam or Not Spam.

- **Performance measure, P:** Total percent of mails being correctly classified as being "Spam" or "Not Spam".
- **Experience, E:** Set of Mails with label "Spam"

Steps for Designing Learning System are:



Step 1- Choosing the Training Experience: The very important and first task is to choose the training data or training experience which will be fed to the Machine Learning Algorithm. It is important to note that the data or experience that we fed to the algorithm must have a significant impact on the Success or Failure of the Model. So Training data or experience should be chosen wisely.
Below are the attributes which will impact on Success and Failure of Data:

- The training experience will be able to provide direct or indirect feedback regarding choices. For example: While Playing chess the training data will provide feedback to itself like instead of this move if this is chosen the chances of success increases.
- Second important attribute is the degree to which the learner will control the sequences of training examples. For example: when training data is fed to the machine then at that time accuracy is very less but when it gains experience while playing again and again with itself or opponent the machine algorithm will get feedback and control the chess game accordingly.
- Third important attribute is how it will represent the distribution of examples over which performance will be measured. For example, a Machine learning algorithm will get experience while going through a number of different cases and different examples. Thus, Machine Learning Algorithm will get more and more experience by passing through more and more examples and hence its performance will increase.

Step 2- Choosing target function: The next important step is choosing the target function. It means according to the knowledge fed to the algorithm the machine learning will choose NextMove function which will describe what type of legal moves should be taken. For example: While playing chess with the opponent, when opponent will play then the machine learning algorithm will decide what be the number of possible legal moves taken in order to get success.

Step 3- Choosing Representation for Target function: When the machine algorithm will know all the possible legal moves the next step is to choose the optimized move using any representation i.e. using linear Equations, Hierarchical Graph Representation, Tabular form etc. The NextMove function will move the Target move like out of these move which will provide more success rate. For Example: while playing chess machine have 4 possible moves, so the machine will choose that optimized move which will provide success to it.

Step 4- Choosing Function Approximation Algorithm: An optimized move cannot be chosen just with the training data. The training data had to go through with set of example and through these examples the training data will approximates which steps are chosen and after that machine will provide feedback on it. For Example: When a training data of Playing chess is fed to algorithm so at that time it is not machine algorithm will fail or get success and again from that failure or success it will measure while next move what step should be chosen and what is its success rate.

Step 5- Final Design: The final design is created at last when system goes from number of examples, failures and success, correct and incorrect decision and what will be the next step etc. Example: DeepBlue is an intelligent computer which is ML-based won chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

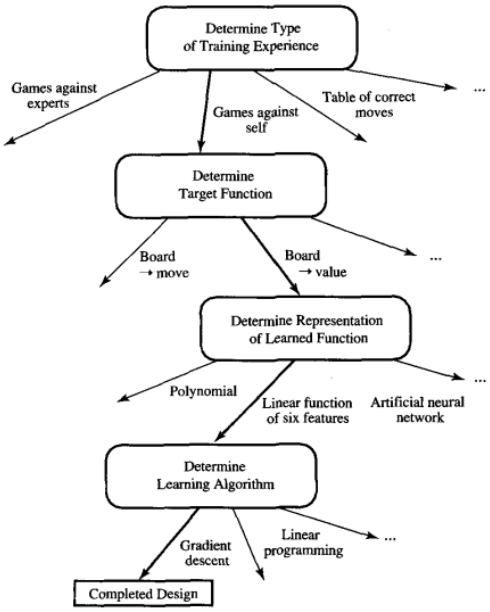


FIGURE 1.2
Summary of choices in designing the checkers learning program.

PERSPECTIVES AND ISSUES IN MACHINE LEARNING

One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner. For example, consider the space of hypotheses that could in principle be output by the above checkers learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for the weight's wo through w6. The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with the available training examples. The LMS algorithm for fitting weights achieves this goal by iteratively tuning the weights, adding a correction to each weight each time the hypothesized evaluation function predicts a value that differs from the training value. This algorithm works well when the hypothesis representation considered by the learner defines a continuously parameterized space of potential hypotheses.

Issues in Machine Learning

Our checkers example raises a number of generic questions about machine learning. The field of machine learning, and much of this book, is concerned with answering questions such as the following:

- What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations?
- How much training data is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?
- When and how can prior knowledge held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct?
- What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
- What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?
- How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

CONCEPT LEARNING TASK

Concept learning is a fundamental task in machine learning that involves training a model to recognize and categorize patterns or concepts from a set of examples or data points. It's like teaching a machine to understand the underlying rules of a specific concept, such as identifying a cat in an image or predicting whether a customer will make a purchase.

Key Concepts

- Target Concept: The underlying rule or pattern that the model aims to learn.
- Training Data: A set of labeled examples used to train the model. Each example consists of an input and its corresponding output (label).
- Hypothesis: A proposed rule or function that the model learns from the training data.
- Generalization: The ability of the model to accurately classify new, unseen data based on the learned concept.

How Concept Learning Works

- Data Preparation: The training data is carefully prepared, ensuring it is representative of the target concept and free from biases.
- Model Selection: An appropriate machine learning algorithm is chosen based on the nature of the data and the desired outcome. Common algorithms include decision trees, support vector machines, and neural networks.
- Training: The model is trained on the labeled data, iteratively adjusting its parameters to minimize errors and improve its ability to predict the target concept.
- Evaluation: The trained model is evaluated on a separate test dataset to assess its generalization performance. Metrics such as accuracy, precision, and recall are used to measure the model's effectiveness.
- Refinement: Based on the evaluation results, the model may be further refined or retrained to improve its performance. Applications of Concept Learning
- Image Recognition: Identifying objects, faces, and scenes in images.
- Natural Language Processing: Understanding and generating human language, such as sentiment analysis and machine translation.
- Fraud Detection: Identifying unusual patterns in financial transactions that may indicate fraudulent activity.
- Medical Diagnosis: Assisting doctors in diagnosing diseases based on patient data.
- Customer Segmentation: Grouping customers based on their behavior and preferences for targeted marketing

CONCEPT LEARNING AS SEARCH

Concept learning involves exploring a hypothesis space to identify the hypothesis that best explains the training examples. This hypothesis space is implicitly defined by the hypothesis representation chosen by the learning algorithm designer. By selecting a specific representation, the designer determines the space of all hypotheses the program can represent and learn.

Example: EnjoySport Learning Task

In the **EnjoySport** learning task, we aim to find a hypothesis (rule) that determines whether the weather conditions are favorable for enjoying sports. Let's break it down step by step.

This represents all possible combinations of weather attributes. The attributes and their possible values are:

1. **Sky**: {Sunny, Cloudy, Rainy} → **3 values**
2. **AirTemp**: {Hot, Cold} → **2 values**
3. **Humidity**: {High, Normal} → **2 values**
4. **Wind**: {Strong, Light} → **2 values**
5. **Water**: {Warm, Cool} → **2 values**
6. **Forecast**: {Same, Change} → **2 values**

Different number of instances possible:

To find the total number of possible weather conditions (**instances in XXX**), multiply the number of possible values for each attribute:

$|X|=3x2x2x2x2x2$

$=3\times32$

$=96$

So, there are **96 distinct weather instances**.

Hypothesis Space (H)

A hypothesis is a rule that classifies instances as positive or negative. Hypotheses can use specific values (e.g., "Sunny") or wildcards (?), which mean "any value is fine." For each attribute:

- There are 4 possible constraints:
  - A specific value (e.g., "Sunny")
  - A wildcard (?)
  - "Ø" (no instances match, meaning the hypothesis is invalid).

Syntactically distinct hypotheses: additionally 2 more values: ?(accepts any values which is most general hypothesis and Ø (reject any values which is more specific hypothesis)

For each of the 6 attributes, there are **4 options**. The total number of syntactically distinct hypotheses is:

1. **Sky**: { Ø, Sunny, Cloudy, Rainy,?}→**5 values**

2. **AirTemp**: { Ø ,Hot, Cold, ?} → **4values**
3. **Humidity**: { Ø ,High, Normal, ?} → **4 values**
4. **Wind**: { Ø ,Strong, Light, ?} → **4 values**
5. **Water**: { Ø ,Warm, Cool, ?} → **4 values**
6. **Forecast**: {Ø ,Same, Change, ?} → **4vvalues**

Including the empty hypothesis (all "Ø"), we get:

|H|=5x4x4x4x4x4=5120

So, we have 5120 Syntactically distinct hypotheses possible

Semantically distinct hypotheses:

Some hypotheses, like those containing only "Ø," classify all instances as negative and are redundant. Removing these, the number of **semantically distinct hypotheses** becomes:

Here Ø is taken as common means

1. **Sky**: {Sunny, Cloudy, Rainy,?}→**4 values**
2. **AirTemp**: { Hot, Cold, ?} → **3 values**
3. **Humidity**: { High, Normal, ?} → **3 values**
4. **Wind**: { Strong, Light, ?} → **3 values**
5. **Water**: { Warm, Cool, ?} → **3 values**
6. **Forecast**: {,Same, Change, ?} → **3 values**

=1+(4×3x3x3x3x3)

=1+(4×243)

=1+972=973

After finding all syntactically and semantically distinct hypothesis we search the best match from all these that matches our learning model (training example).

FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

The FIND-S algorithm is a simple way to find a rule (or hypothesis) that matches all the **positive examples** in a dataset while ignoring the negative ones. It works step by step, starting with a very specific rule and gradually making it more general to include all positive examples. Here's how it works in an easy way:

FIND-S Algorithm

The **FIND-S** algorithm is like starting with the most specific guess and slowly relaxing it until it fits all the examples.

1. **Start small**: Begin with the most specific rule (e.g., "Only this exact weather works").
2. **Fix the rule**: For each good (positive) example, check if your rule matches it:
    - If it does, great—do nothing!
    - If it doesn't, make the rule a bit more general (e.g., "Okay, maybe it works if the wind isn't strong").
3. **Finish**: When you're done, you have a rule that matches all the good examples.

Example:

Imagine you're trying to figure out what kind of weather makes you enjoy playing a sport, using this data:

Sky	Temperature	Humidity	Wind	Water	Forecast	Play Sport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

- Start with the most specific rule: h=(?, ?, ?, ?, ?, ?), which means "no conditions are set yet."
- Look at the first positive example: (Sunny, Warm, Normal, Strong, Warm, Same)
    - Rule becomes: h=(Sunny, Warm, Normal, Strong, Warm, Same)
- Look at the second positive example: (Sunny, Warm, High, Strong, Warm, Same)
    - Update the rule to match both examples:
      h=(Sunny, Warm, ?, Strong, Warm, Same)
- Ignore the negative example.
- Look at the fourth positive example: (Sunny, Warm, High, Strong, Cool, Change)
    - Update the rule again: h=(Sunny, Warm, ?, Strong, ?, ?)

Final rule: (Sunny, Warm, ?, Strong, ?, ?). This means you enjoy playing sports if it's sunny, warm, and windy, regardless of the other conditions.

Properties of FIND-S

1. **Guarantees a Maximally Specific Hypothesis**:
    - In conjunction-based hypothesis spaces like EnjoySport, FIND-S always outputs the most specific consistent hypothesis.
2. **Focus on Positive Examples**:
    - Negative examples are ignored unless they conflict with the current hypothesis, which is unlikely if the hypothesis space contains the true target concept.

Limitations of FIND-S

1. **Uncertainty About the Target Concept**:
    - FIND-S cannot determine if the final hypothesis is the only consistent hypothesis or if others exist.
2. **Preference for Specific Hypotheses**:
    - There's no inherent reason to prefer the most specific hypothesis over more general ones, especially in cases of multiple consistent hypotheses.
3. **Sensitivity to Errors**:
    - Inconsistent or noisy training examples can severely mislead FIND-S, as it relies solely on positive examples and assumes they are error-free.
4. **Multiple Maximally Specific Hypotheses**:
    - In some hypothesis spaces, there may be multiple maximally specific hypotheses, requiring modifications to FIND-S to handle backtracking or branching.
5. **No Accommodation for Non-Conjunctive Hypothesis Spaces**:
    - FIND-S works well only in hypothesis spaces where a single maximally specific consistent hypothesis exists.

Key Points to Remember

- FIND-S focuses only on **positive examples** and ignores negatives.
- It starts very specific and becomes more general as needed.
- It finds one rule that works for the given data but doesn’t guarantee it's the only possible rule.

Why FIND-S Is Simple But Limited

- **Good for Clean Data**: Works well if the data is perfect (no mistakes or noise).
- **Ignores Negatives**: It doesn't use negative examples to refine the rule.
- **May Miss Other Rules**: If there are multiple valid rules, it picks the most specific one but doesn’t explore other options.

In short, FIND-S is like a detective who focuses only on positive clues and tries to make the simplest case for what’s true!

VERSION SPACES AND THE CANDIDATE ELIMINATION ALGORITHM

VERSION SPACES

A **version space** is a set of all hypotheses (rules) that are consistent with the given training data. It represents everything the learner currently knows about the target concept.

Why "Version Space"?

The "version" refers to different possibilities or hypotheses that might explain the data. The space includes:

- Hypotheses that **agree** with all positive examples.
- Hypotheses that **disagree** with all negative examples.

How It Works:

A version space has two boundaries:

1. **Specific boundary (S):** The most specific hypotheses consistent with the data.
2. **General boundary (G):** The most general hypotheses consistent with the data.

The true target concept lies **between these boundaries**.

1. **Initial Version Space:**
    - S=The most specific hypothesis: Only matches exactly one positive example.
    - G=The most general hypothesis: Matches everything.
2. **Update Version Space:**
    - Each **positive example** makes S more general to include it.
    - Each **negative example** makes G more specific to exclude it.

Why Use Version Spaces?

- **Efficient Representation**: Instead of listing all hypotheses, it tracks only the boundaries S and G.
- **Keeps Track of Knowledge**: Helps understand what the learner knows and doesn’t know yet.
- **Flexible Search**: Allows for adding or removing examples to refine the boundaries.

A **version space** is the range of hypotheses consistent with the training data, bounded by the most specific (S) and the most general (G) hypotheses. It narrows down as you process more examples, zeroing in on the true concept.

CANDIDATE ELIMINATION ALGORITHM

It’s a way to learn rules for when something happens (like "Play Sport = Yes") by narrowing down possibilities. The algorithm works by keeping two boundaries:

1. **Specific Boundary (S):** The most specific rule that only fits positive examples.
2. **General Boundary (G):** The most general rule that excludes negative examples.

As we process each example, we update S and G to make them more accurate.

The Data

Sky	Temp	Humidity	Wind	Water	Forecast	Play Sport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Our goal is to find the rule for when "Play Sport" is **Yes**.

Step-by-Step Execution

Start with Initial S **and** G**:**

- **S=⟨Φ,Φ,Φ,Φ,Φ,Φ⟩** "I know nothing; no instance fits."
- **G=⟨?,?,?,?,?,?⟩** "Everything works."

Example 1: Sunny, Warm, Normal, Strong, Warm, Same (Yes)

It’s a **positive example**. We update S and G:

- **S:** Start from nothing and match this example exactly:
  S=⟨Sunny,Warm,Normal,Strong,Warm,Same⟩
- **G:** Stays the same because it already includes everything.
  G=⟨?,?,?,?,?,?⟩

Example 2: Sunny, Warm, High, Strong, Warm, Same (Yes)

It’s another **positive example**. S is too specific, so we generalize it to fit both positive examples:

- Compare each attribute in S to the new example:
  - **Sky = Sunny:** Matches, no change.
  - **Temp = Warm:** Matches, no change.
  - **Humidity = Normal vs High:** Doesn't match, so generalize to ?.
  - **Wind = Strong:** Matches, no change.
  - **Water = Warm:** Matches, no change.
  - **Forecast = Same:** Matches, no change.
- Updated S:
  S=⟨Sunny,Warm,?,Strong,Warm,Same⟩
  (This means Humidity can be anything now.)

  G=⟨?,?,?,?,?,?⟩**:** Still no change.

Example 3: Rainy, Cold, High, Strong, Warm, Change (No)

It's a **negative example**. We update G to exclude this negative example while staying as general as possible.

- S=⟨Sunny,Warm,?,Strong,Warm,Same⟩ keep specific hypothesis same means copy the previous S
  Step 1: Look at G: ⟨?,?,?,?,?,?⟩
  - G is too general—it allows this negative example. We specialize G to exclude it.
- **Step 2:** Create new rules for G by considering each attribute and change G to specific.

  G= {Sunny, warm,?, ?, ?,Same) → change the attributes to more specific.

Example 4: Sunny, Warm, High, Strong, Cool, Change (YES)

It's a **positive example**. S needs to generalize further to fit this example. Compare with previous S.

- Compare S=⟨Sunny,Warm,?,Strong,Warm,Same⟩ to the new example:
  - **Sky = Sunny:** Matches, no change.
  - **Temp = Warm:** Matches, no change.
  - **Humidity = ?:** Already generalized, no change.
  - **Wind = Strong:** Matches, no change.
  - **Water = Warm vs Cool:** Doesn't match, so generalize to ?.
  - **Forecast = Same vs Change:** Doesn't match, so generalize to ?.
- Updated S:
  S=⟨Sunny,Warm,?,Strong,?,?⟩
- **G:** No change—it already matches all positive examples.

Final Results:

1. **Specific Boundary (S):**
   S=⟨Sunny,Warm,?,Strong,?,?⟩
   (This means: Sky must be Sunny, Temp must be Warm, Wind must be Strong. Humidity, Water, and Forecast can be anything.)
2. **General Boundary (G):**
   G=⟨Sunny,warm,?,?,?,?⟩
   (This means: Sky must be Sunny. Everything else can be anything.). Attribute **same** is not considered here.

Why is G=⟨Sunny,?,?,?,?,?⟩

1. **General Boundary (G)** starts very broad (because it initially includes everything).
2. After processing positive examples, G gets **refined** to include only the conditions that must be true for playing sports ("Yes").
3. The **Sky** condition (Sunny) is the only attribute that must always be true in the general rule.
4. The **other attributes** (Temperature, Humidity, Wind, Water, Forecast) can be anything since the general rule still covers all the positive examples we've seen so far.

Why does the Sky have to be Sunny, warm in G?

When we look at all the **positive examples** (the ones where Play Sport = Yes), we find that they all have **Sky = Sunny**. Since G should cover all positive examples, we make **Sky = Sunny** and warm a condition in the rule. But the other attributes (Temperature, Humidity, Wind, etc.) **can vary**, so we leave them as **wildcards (?)**.

Advantages of CEA over Find-S:

1. Improved accuracy: CEA considers both positive and negative examples to generate the hypothesis, which can result in higher accuracy when dealing with noisy or incomplete data.
2. Flexibility: CEA can handle more complex classification tasks, such as those with multiple classes or non-linear decision boundaries.
3. More efficient: CEA reduces the number of hypotheses by generating a set of general hypotheses and then eliminating them one by one. This can result in faster processing and improved efficiency.
4. Better handling of continuous attributes: CEA can handle continuous attributes by creating boundaries for each attribute, which makes it more suitable for a wider range of datasets.

Disadvantages of CEA in comparison with Find-S:

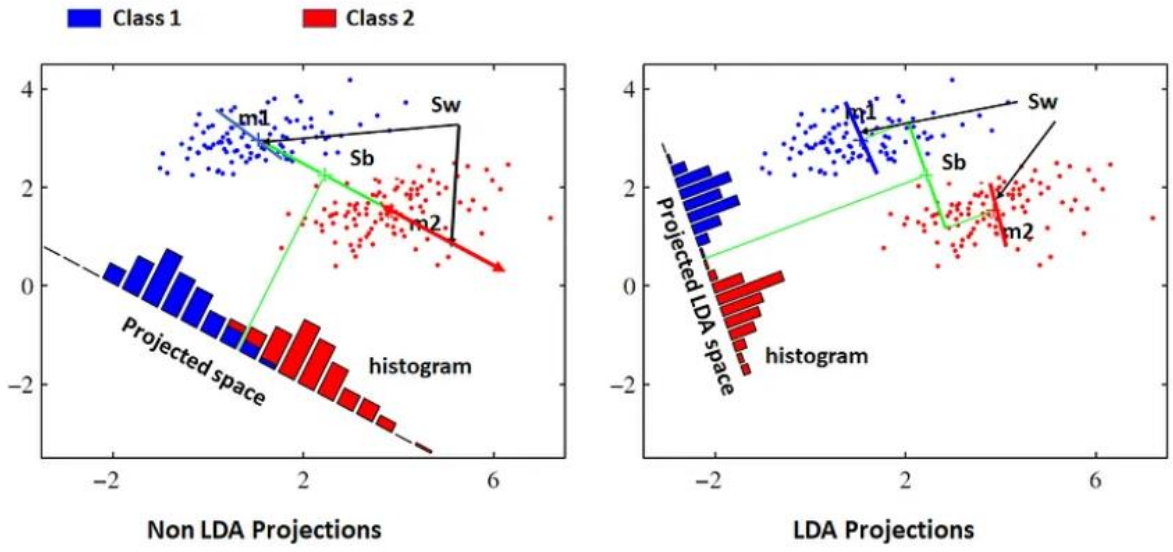
1. More complex: CEA is a more complex algorithm than Find-S, which may make it more difficult for beginners or those without a strong background in machine learning to use and understand.

2. Higher memory requirements: CEA requires more memory to store the set of hypotheses and boundaries, which may make it less suitable for memory-constrained environments.
3. Slower processing for large datasets: CEA may become slower for larger datasets due to the increased number of hypotheses generated.
4. Higher potential for overfitting: The increased complexity of CEA may make it more prone to overfitting on the training data, especially if the dataset is small or has a high degree of noise.

LINEAR DISCRIMINANTS

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

LDA not only helps to reduce the number of features but also ensures that the important class-related information is retained, making it easier for models to differentiate between classes.



What is Linear Discriminant Analysis (LDA)

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

LDA performs two key roles:

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

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

How Does LDA Work?

The core idea of **Linear Discriminant Analysis (LDA)** is to find a new axis that best separates different classes by maximizing the distance between them. LDA achieves this by reducing the dimensionality of the data while retaining the class-discriminative information.

Key Concepts:

1. **Maximizing Between-Class Variance**: LDA maximizes the separation between the means of different classes.
2. **Minimizing Within-Class Variance**: It minimizes the spread (variance) within each class, ensuring that data points from the same class remain close together.
3. **Projection to Lower-Dimensional Space**: LDA projects data onto a new axis or subspace that best separates the classes. For example, in a 3-class problem, LDA can reduce the dimensionality to 2 or even 1 while preserving class-related information.

Working Mechanism:

- **Step 1**: Compute the mean vectors for each class.
- **Step 2**: Calculate the within-class and between-class scatter matrices.
- **Step 3**: Solve for the linear discriminants that maximize the ratio of between-class variance to within-class variance.
- **Step 4**: Project the data onto the new lower-dimensional space.

Advantages & Disadvantages of Using LDA.

Advantages:

1. **Simplicity**: LDA is easy to implement and understand, making it suitable for beginners.
2. **Interpretability**: It provides clear insight into how features contribute to the classification task.
3. **Computational Efficiency**: LDA is computationally less intensive, making it useful for large datasets.

4. **Works Well with Linearly Separable Data**: It performs effectively when the classes are linearly separable.

Disadvantages:

1. **Sensitive to Assumptions**: LDA assumes that features follow a Gaussian distribution, which may not always hold.
2. **Struggles with Non-linear Relationships**: It may not perform well if the data contains complex, non-linear relationships.
3. **Affected by Class Imbalance**: LDA can be biased toward the majority class if the class distribution is imbalanced.
4. **Impact of Outliers**: It is sensitive to outliers, which can affect the model's performance.

Applications of Linear Discriminant Analysis (LDA)

1. Face Recognition: LDA helps extract features from facial images, classifying them based on individuals. It is commonly used in biometric systems to identify or verify users.

2. Disease Diagnosis in Healthcare: LDA is used to analyze medical data for classifying diseases, such as distinguishing between different stages of cancer or predicting the presence of heart disease.

3. Customer Identification in Marketing: In marketing, LDA aids in customer segmentation by grouping customers based on their profiles, enabling targeted campaigns and personalized services.

4. Credit Risk Assessment in Finance: Financial institutions use LDA to assess credit risk by analyzing customer data to predict the likelihood of loan defaults or creditworthiness.

5. Quality Control in Manufacturing: LDA identifies defects in products by analyzing sensor data, ensuring that faulty products are detected early in the production process.

6. Campaign Optimization in Marketing: LDA helps optimize marketing campaigns by analyzing customer interactions and identifying which strategies yield the best results.

PERCEPTRON

Perceptron Are Based on Biological Neurons and Originally proposed in 1957, perceptrons were one of the earliest components of artificial neural networks. The structure of the perceptron is based on the anatomy of neurons. Neurons have several parts, but for our purposes, the most important parts are the dendrites, which receive inputs from other neurons, and the axon, which produces outputs.

Neuron Activation

Neurons "fire" – that is, produce an output – in an all or nothing way. The outputs of a neuron are essentially 0 or 1. On or off. A neuron will "fire" if the input signals at the dendrites are sufficiently large, collectively. If the amount of input signal at the dendrites is high enough, the neuron will "fire" an produce an output. But if the amount of input signal is insufficient, the neuron will not produce a output. Put simply, the neuron sums up the inputs, and if the collective input signals meet a certain threshold, then it will produce an output. If the collective input signals are under the threshold, it will not produce an output. This is, of course, a very simple explanation of how a neuron works (because they are very complex at the chemical level), but it's roughly accurate.

What is Perceptron?

The **Perceptron Learning** is a fundamental concept in machine learning and serves as one of the simplest types of artificial neural networks. It is primarily used for **binary classification tasks** and is based on the idea of learning a linear decision boundary to separate data points into two classes. The perceptron algorithm was introduced by **Frank Rosenblatt** in 1958. It operates on a set of input features and produces an output that is either 1 or −1 (or 0 depending on the implementation). The model is trained iteratively, adjusting its weights based on the error between predicted and actual labels.

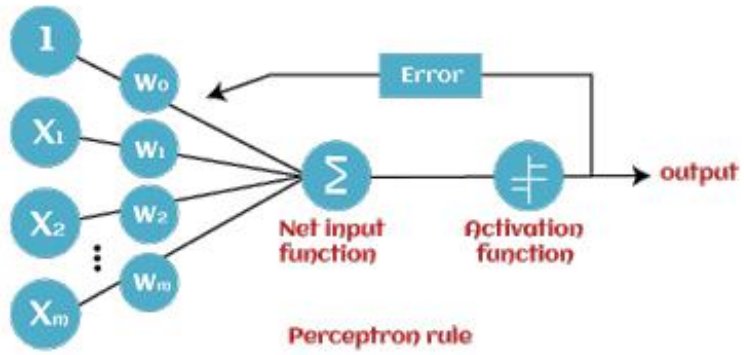
Components of a Perceptron : A perceptron consists of following

1. **Inputs (x1,x2,…,xn)**:Features of the data (numerical values).
2. **Weights (w1,w2,…,wn)**:Each input has a corresponding weight that determines its influence on the output.
3. **Bias (b)**:A constant term that shifts the decision boundary. The bias term helps the perceptron make adjustments independent of the input, improving its flexibility in learning. bias is a numerical value that is added to the weighted sum of inputs. It's an adjustable parameter that helps the model learn and perform better.
4. **Weighted Sum (z):** perceptron computes a weighted sum of the inputs. The perceptron can be represented as:

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

5. **Activation Function (f(z)):** The perceptron model begins with multiplying all input values and their weights, then adds these values to create the weighted sum. Further, this weighted sum is applied to the activation function to obtain the desired output. This activation function is also known as the step function and is represented by g(h).

$$g(h) = \begin{cases} 1 & \text{if } h > \theta \\ 0 & \text{if } h \leq \theta. \end{cases}$$



Working of the Perceptron

1. **Input Features**: Take a vector of input features (x1,x2,…,…,xn) from the dataset.
2. **Compute Weighted Sum**: Calculate

$$z = \sum w_i x_i + b.$$

3. **Apply Activation Function**: Use the step function to decide the output (1 or 0).
4. **Update Weights (During Training)**:

    - If the predicted output is incorrect, adjust the weights and bias using the **Perceptron Learning Algorithm**.

Mathematical Model: The perceptron can be represented as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Where:

- y: The predicted output (1 or -1).
- w_i: Weights.
- x_i: Inputs.
- b: Bias.
- $f(z)$: Activation function.

Types of Perceptron

1. **Single-Layer Perceptron** is a type of perceptron is limited to learning linearly separable patterns. It is effective for tasks where the data can be divided into distinct categories through a straight line. While powerful in its simplicity, it struggles with more complex problems where the relationship between inputs and outputs is non-linear.
2. **Multi-Layer Perceptron** possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

Perceptron Learning Algorithm:

A perceptron is like a very basic "brain" for a machine. It looks at input data (numbers) and makes a decision: Class A or Class B (e.g., "yes" or "no").

Step 1: Start with inputs and weights

- Imagine you have some input features (e.g., x1, x2) like:
- Each input has a weight (w1, w2) that tells the perceptron how important that input is.

Step 2: Compute the "weighted sum"

- Multiply each input by its weight, and add them all together. Then, add a bias (b), which is like a nudge to adjust the sum.

$$\text{Weighted sum } (z) = w1 \cdot x1 + w2 \cdot x2 + b$$

Step 3: Decide the output

- Use a simple rule: If the weighted sum (z) is positive, output 1 (e.g., "yes").
- If it's negative or zero, output -1 (e.g., "no").

This rule is called the **activation function**.

Step 4: Check if the output is correct

- Compare the perceptron's guess (y^) with the actual answer (y).
- If it's correct, you're good! If it's wrong, adjust the weights and bias.

Step 5: Update the weights and bias

- Adjust the weights and bias to make the perceptron "learn"

$$w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$$

$$b \leftarrow b + \eta \cdot (y - \hat{y})$$

- $y - \hat{y}$: How wrong the guess was.
- η: Learning rate (how big the adjustments are).

Step 6: Repeat

- Go through the dataset multiple times, adjusting weights and bias each time the perceptron makes a mistake.

Key Features

- **Linear Model**: The perceptron can only separate data that is linearly separable.
- **Supervised Learning**: It requires labeled data for training.
- **Binary Classification**: It predicts one of two possible classes (1 or −1).

Strengths

- Simple and easy to understand.
- Efficient for linearly separable data.
- Forms the foundation for more complex neural networks.

Limitations

- **Cannot Handle Non-linear Data**: It fails when data is not linearly separable.
- **Binary Outputs**: Limited to binary classification tasks.
- **Sensitive to Feature Scaling**: Requires normalization or scaling for effective learning.

Applications

- Image recognition (basic tasks).
- Spam filtering (binary classification).
- Sentiment analysis (positive vs. negative sentiment).

EXAMPLE: Here's an example of a perceptron for the logical AND function with the given parameters:

- w1=1.2, w2=0.6 (weights)
- θ=1= 1 (threshold)
- η=0.5 = 0.5 (learning rate)

Step 1: AND Function Truth Table

x1	x2	AND Output (Target t)
0	0	0
0	1	0
1	0	0
1	1	1

Step 2: Activation Function

The perceptron output is calculated as:

$$y = \begin{cases} 1, & \text{if } w_1 x_1 + w_2 x_2 \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

Step 3: Training with Initial Weights

Iteration 1

For each input, compute the weighted sum and update weights if necessary, using:

$$w_j^{new} = w_j^{old} + \eta(t - y)x_j$$

Testing Initial Weights

1. **For (0,0):** (w1x1*w2x2)

$$(1.2 \times 0) + (0.6 \times 0) = 0 < 1 \Rightarrow y = 0$$

☑ No update needed.

2. **For (0,1):**

$$(1.2 \times 0) + (0.6 \times 1) = 0.6 < 1 \Rightarrow y = 0$$

☑ No update needed.

3. **For (1,0):**

$$(1.2 \times 1) + (0.6 \times 0) = 1.2 \geq 1 \Rightarrow y = 1$$

❌ Incorrect (target = 0), so update weights:

$$w_1 = 1.2 + 0.5(0 - 1)(1) = 0.7$$

$$w_2 = 0.6 + 0.5(0 - 1)(0) = 0.6$$

4. **For (1,1):**

$$(0.7 \times 1) + (0.6 \times 1) = 1.3 \geq 1 \Rightarrow y = 1$$

☑ Correct.

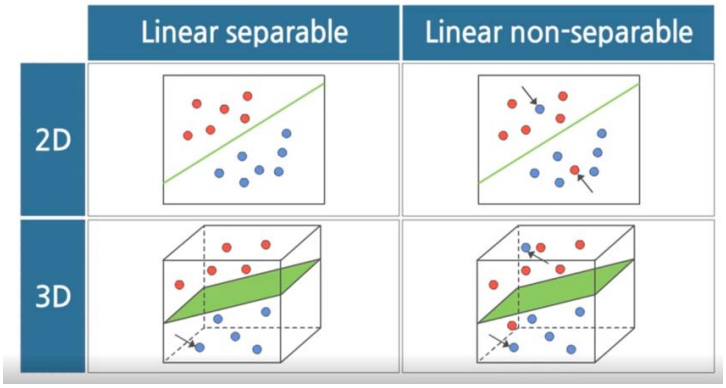
Final Weights

After training, the perceptron correctly classifies AND function with:

$$w_1 = 0.7, \quad w_2 = 0.6$$

LINEAR SEPARABILITY

Linear separability is an important concept in machine learning, particularly in the field of supervised learning. It refers to the ability of a set of data points to be separated into distinct categories using a linear decision boundary. In other words, if there exists a straight line that can cleanly divide the data into two classes, then the data is said to be linearly separable.



A dataset is **linearly separable** if there exists a hyperplane that can separate the data points into distinct classes without any misclassification.

- **In 2D:** A straight line separates two classes.
- **In 3D:** A plane separates two classes.

Linear separability means that you can draw a straight line (or a flat surface, or a hyperplane) that separates two groups of data points perfectly without any overlap.

In other words, there is a clear boundary where:

- One group of data points lies on one side of the boundary.
- The other group of data points lies on the other side.

Imagine you have two types of points:

- **Class 1 (Positive)**: Points like (1, 2), (2, 3)
- **Class 2 (Negative)**: Points like (-1, -2), (-2, -3)

You can draw a straight line between the two classes, and the points on one side belong to Class 1, while the points on the other side belong to Class 2.

This line is the **decision boundary**, and the data is **linearly separable** because the line separates the two classes without overlap.

Real-World Applications

1. **Image Classification**:
    - Linear separability is rare; deep learning handles non-linear boundaries.
2. **Medical Diagnosis**:
    - Linearly separable cases may involve straightforward conditions; complex diseases often require advanced methods.
3. **Spam Detection**:
    - Simple keyword-based filters assume linear separability, while modern techniques use non-linear models.

Why is Linear Separability Important?

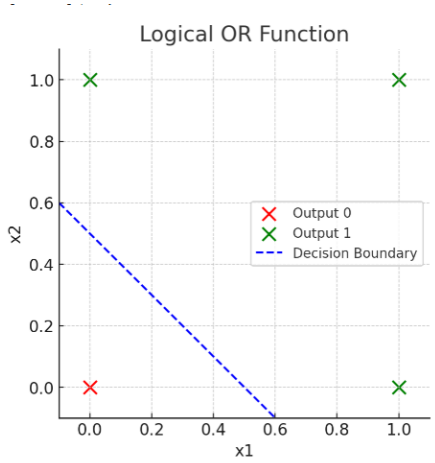
The concept of linear separability helps us decide which machine learning algorithms to use. Some algorithms work well when the data is linearly separable, while others are better for more complex, non-linearly separable data.

- **Linear Models (e.g., Perceptron, SVM)**: These work best when the data is linearly separable. They try to find the straightest line or plane to divide the data.
- **Non-Linear Models (e.g., Neural Networks, Decision Trees)**: These are more flexible and can handle non-linearly separable data. They can create complex decision boundaries.

Example: Logical OR Function is linearly seperable

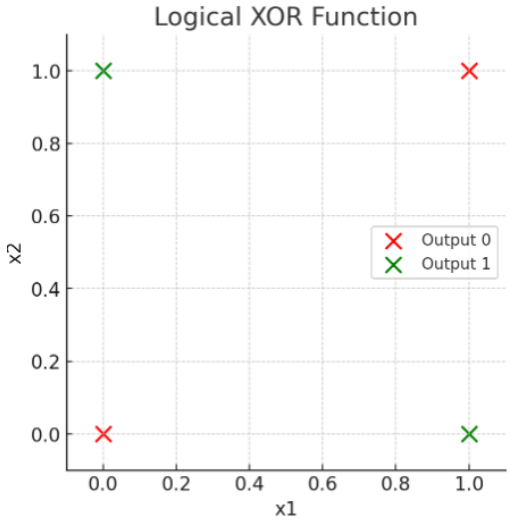
The logical OR function follows this truth table:

x1	x2	OR Output y
0	0	0
0	1	1
1	0	1
1	1	1



Example: Logical XOR Function is not linearly separable

x1	x2	XOR Output y
0	0	0
0	1	1
1	0	1
1	1	0



1. **Linearly Separable Data**:
    - Blue points (+1) are separated from red points (−1) by the dashed green line ($x2=x1+1$).
    - The data is perfectly separable by this straight line.
2. **Non-Linearly Separable Data (XOR)**:
    - Blue points (+1) and red points (−1) are arranged in such a way that no single straight line can separate the classes.
    - This is a classic XOR problem where the decision boundary requires a more complex, non-linear solution.

Advantages of Linear Separability

1. **Simplicity**:
    - Linear separability allows using simple models with fewer parameters.
2. **Faster Training**:
    - Models converge quickly during training due to straightforward optimization.
3. **Interpretability**:
    - Easy to visualize and understand the decision boundary.
4. **Optimal Solution**:
    - Algorithms like SVM find the maximum margin boundary, ensuring optimal performance for separable data.
5. **Good Generalization**:
    - Models are less likely to overfit due to their simplicity.

Disadvantages of Linear Separability

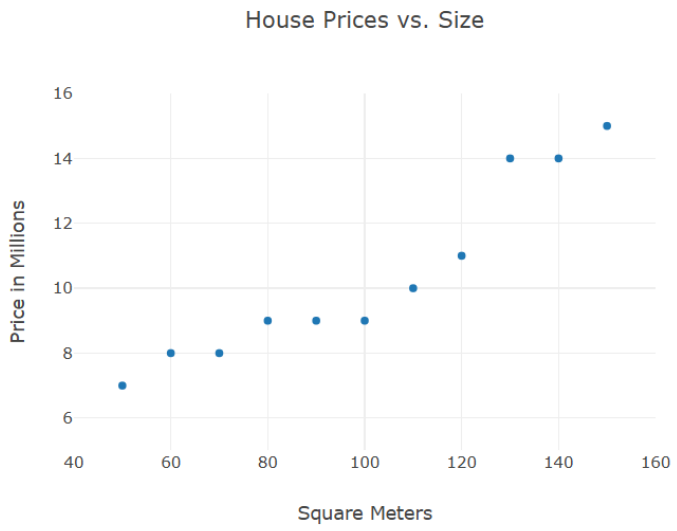
1. **Limited Applicability**:
    - Many real-world datasets are not linearly separable.
2. **Lack of Flexibility**:
    - Cannot capture complex patterns in the data.
3. **Over-Simplification**:
    - May miss subtle relationships or nuances.
4. **Sensitive to Noise**:
    - Outliers or noisy data near the boundary can disrupt the model.
5. **Feature Dependence**:
    - Requires feature transformations for non-linearly separable data.
6. **Failure for Non-Linearly Separable Data**:
    - Cannot separate inherently non-linear datasets without additional techniques.

LINEAR REGRESSION

Linear regression is a fundamental supervised learning algorithm used in machine learning for modeling the relationship between one or more independent variables (features) and a dependent variable (target). The goal is to find the best-fit line (or hyperplane in higher dimensions) that minimizes the error in predicting the dependent variable.

In machine learning, labeled datasets contain input data (features) and output labels (target values). For linear regression in machine learning, we represent features as independent variables and target values as the dependent variable. It predicts the continuous output variables based on the independent input variable. like the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.

Square Meters (m^2)	Price (Millions)
50	7
60	8
70	8.5
80	9
90	9.5
100	10
110	11
120	12
130	13
140	14
150	15.5

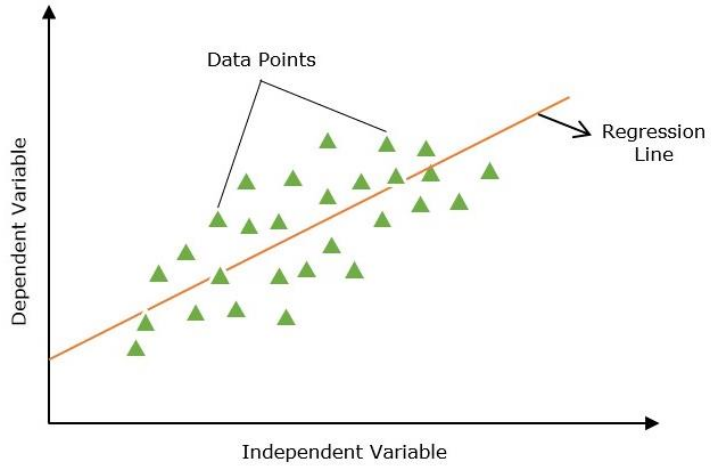


In the above data, the target House Price is the dependent variable represented by Y, and the feature, Square Feet, is the independent variable represented by X. The input features (X) are used to predict the target label (Y). So, the independent variables are also known as predictor variables, and the dependent variable is known as the response variable.

The main goal of the linear regression model is to find the best-fitting straight line (often called a regression line) through a set of data points.

Line of Regression

A straight line that shows a relation between the dependent variable and independent variables is known as the line of regression or regression line.



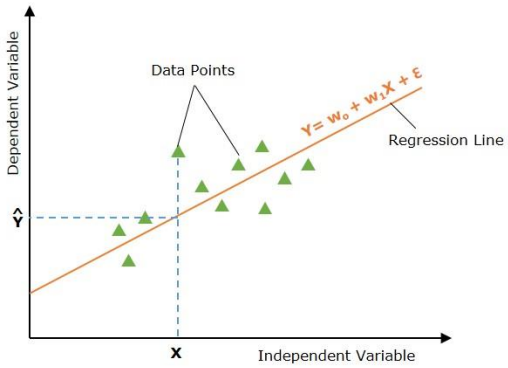
Types of Linear Regression

Linear regression is of the following two types −

- Simple Linear Regression
- Multiple Linear Regression

1. Simple Linear Regression

Simple linear regression is a type of regression analysis in which a single independent variable (also known as a predictor variable) is used to predict the dependent variable. In other words, it models the linear relationship between the dependent variable and a single independent variable.



In the above image, the straight line represents the simple linear regression line where $\hat{Y}$ is the predicted value, and X is the input value.

Mathematically, the relationship can be modelled as a linear equation −

$$Y=w0+w1X+\epsilon$$

Where,

- Y is the dependent variable (target).
- X is the independent variable (feature).
- w_0 is the y-intercept of the line.
- w_1 is the slope of the line, representing the effect of X on Y.
- ε is the error term, capturing the variability in Y not explained by X.

2. Multiple Linear Regression

Multiple linear regression is basically the extension of simple linear regression that predicts a response using two or more features.

When dealing with more than one independent variable, we extend simple linear regression to multiple linear regression. The model is expressed as:

Multiple linear regression extends the concept of simple linear regression to multiple independent variables. The model is expressed as:

$$Y=w0+w1X1+w2X2+\cdots+wpXp+\epsilon$$

Where,

- X_1, X_2, ..., X_p are the independent variables (features).
- w_0, w_1, ..., w_p are the coefficients for these variables.
- ε is the error term.

How Does Linear Regression Work?

The main goal of linear regression is to find the best-fit line through a set of data points that minimizes the difference between the actual values and predicted values. So it is done? This is done by estimating the parameters w_0, w_1 etc.

The working of linear regression in machine learning can be broken down into many steps as follows −

- **Hypothesis**− We assume that there is a linear relation between input and output.
- **Cost Function** − Define a loss or cost function. The cost function quantifies the model's prediction error. The cost function takes the model's predicted values and actual values and returns a single scaler value that represents the cost of the model's prediction.
- **Optimization** − Optimize (minimize) the model's cost function by updating the model's parameters.

It continues updating the model's parameters until the cost or error of the model's prediction is optimized (minimized).

Hypothesis Function for Linear Regression

In linear regression problems, we assume that there is a linear relationship between input features (X) and predicted value ($\hat{Y}$). The hypothesis function returns the predicted value for a given input value. Generally we represent a hypothesis by $h_w(X)$ and it is equal to $\hat{Y}$.

Hypothesis function for simple linear regression −

$$\hat{Y} = w_0 + w_1 X$$

Hypothesis function for multiple linear regression –

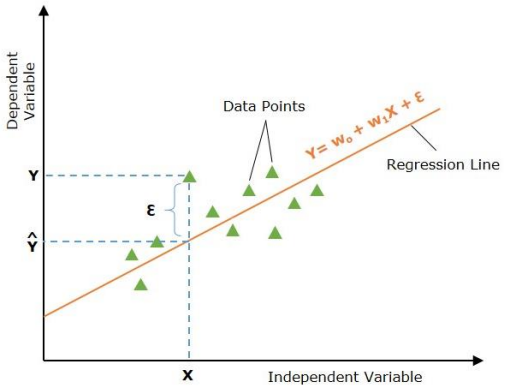
$$\hat{Y} = w_0 + w_1 X_1 + w_2 X_2 + \cdots + w_p X_p$$

For different values of parameters (weights), we can find many regression lines. The main goal is to find the best-fit lines.

Finding the Best Fit Line

A regression line is said to be the best fit if the error between actual and predicted values is minimal.

Below image shows a regression line with error (ε) at input data point X. The error is calculated for all data points and our goal is to minimize the average error/ loss. We can use different types of loss functions such as mean square error (MSE), mean average error (MAE), L_1 loss, L_2 Loss, etc.



Loss Function for Linear Regression

The error between actual and predicted values can be quantified using a loss function of the cost function. The cost function takes the model's predicted values and actual values and returns a single scaler value that represents the cost of the model's prediction. Our main goal is to minimize the cost function.

The most commonly used cost function is the mean squared error function.

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i \right)^2$$

Where,

- n is the number of data points.
- Y_i is the observed value for the i-th data point.
- $\hat{Y}_i = w_0 + w_1 X_i$ is the predicted value for the i-th data point.

Applications of Linear Regression

1. Predictive Modeling: Linear regression is widely used for predictive modeling. For instance, in real estate, predicting house prices based on features such as size, location, and number of bedrooms can help buyers, sellers, and real estate agents make informed decisions.
2. Feature Selection: In multiple linear regression, analyzing the coefficients can help in feature selection. Features with small or zero coefficients might be considered less important and can be dropped to simplify the model.
3. Financial Forecasting: In finance, linear regression models predict stock prices, economic indicators, and market trends. Accurate forecasts can guide investment strategies and financial planning.
4. Risk Management: Linear regression helps in risk assessment by modeling the relationship between risk factors and financial metrics. For example, in insurance, it can model the relationship between policyholder characteristics and claim amounts.

Advantages of Linear Regression

- **Interpretability** − Linear regression is easy to understand, which is useful when explaining how a model makes decisions.
- **Speed** − Linear regression is faster to train than many other machine learning algorithms.
- **Predictive analytics** − Linear regression is a fundamental building block for predictive analytics.
- **Linear relationships** − Linear regression is a powerful statistical method for finding linear relationships between variables.

- **Simplicity** − Linear regression is simple to implement and interpret.
- **Efficiency** − Linear regression is efficient to compute.

Common Challenges with Linear Regression

1. Overfitting: Overfitting occurs when the regression model performs well on training data but lacks generalization on test data. Overfitting leads to poor prediction on new, unseen data.
2. Multicollinearity: When the dependent variables (predictor or feature variables) correlate, the situation is known as multicollinearity. In this, the estimates of the parameters (coefficients) can be unstable.
3. Outliers and Their Impact: Outliers can cause the regression line to be a poor fit for the majority of data points.