MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

MRCET CAMPUS

(AUTONOMOUS INSTITUTION – UGC, GOVT. OF INDIA)

Department of AERONAUTICAL ENGINEERING

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Prepared by:
M. THILAK RAJ
Assistant Professor
Department of ANE
thilakraj@mrcet.ac.in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

DIGITAL NOTES

B.TECH (R-22 Regulation)
(III YEAR – II SEM)
(2024-25)

DEPARTMENT AERONAUTCAL ENGINEERING

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of AERONAUTICAL ENGIERRING

Vision

- Department of Aeronautical Engineering aims to be indispensable source in Aeronautical Engineering which has a zeal to provide the value driven platform for the students to acquire knowledge and empower themselves to shoulder higher responsibility in building a strong nation..

Mission

- The primary mission of the department is to promote engineering education and research. To strive consistently to provide quality education, keeping in pace with time and technology. Department passions to integrate the intellectual, spiritual, ethical, and social development of the students for shaping them into dynamic engineers.

QUALITY POLICY

- Impart up-to date knowledge to the students in Aeronautical area to make them quality engineers.Make the students experience the applications on quality equipment and tools.Provide systems, resources, and training opportunities to achieve continuous improvement.Maintain global standards in education, training, and services.

PROGRAM OUTCOMES (PO's)

Engineering Graduates will be able to:

- Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- Design / development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
- Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
- Life- long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM EDUCATIONAL OBJECTIVES – Aeronautical Engineering

- PEO1 (PROFESSIONALISM & CITIZENSHIP): To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.

- PEO2 (TECHNICAL ACCOMPLISHMENTS): To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.

- PEO3 (INVENTION, INNOVATION AND CREATIVITY): To make the students to design, experiment, analyze, and interpret in the core field with the help of other multi disciplinary concepts wherever applicable.

- PEO4 (PROFESSIONAL DEVELOPMENT): To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.

- PEO5 (HUMAN RESOURCE DEVELOPMENT): To graduate the students in building national capabilities in technology, education and research

PROGRAM SPECIFIC OUTCOMES – Aeronautical Engineering

- To mould students to become a professional with all necessary skills, personality and sound knowledge in basic and advance technological areas.

- To promote understanding of concepts and develop ability in design manufacture and maintenance of aircraft, aerospace vehicles and associated equipment and develop application capability of the concepts sciences to engineering design and processes.

- Understanding the current scenario in the field of aeronautics and acquire ability to apply knowledge of engineering, science and mathematics to design and conduct experiments in the field of Aeronautical Engineering.

- 4. To develop leadership skills in our students necessary to shape the social, intellectual, business and technical worlds.

INDEX

R20

Code No: R20A0513

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

III B.Tech I Semester Regular Examinations, December 2022

Artificial Intelligence

(CSE, IT, CSE-CS, CSE-DS, CSE-IOT)

Roll No										

Time: 3 hours **Max. Marks: 70**

Note: This question paper Consists of 5 Sections. Answer **FIVE** Questions, Choosing ONE Question from each SECTION and each Question carries 14 marks.

***

SECTION-I

1	*A*	List and explain various AI Languages.	[7M]
	B	What are the basic components of AI problem solving methodology? Illustrate with an example.	[7M]
		OR	
2	*A*	Illustrate the heuristic Hill Climbing Algorithm with an example.	[7M]
	B	Explain A* Algorithm with example.	[7M]

SECTION-II

3	*A*	Discuss Alpha-Beta Pruning and its advantages over min-max method.	[10M]
	B	Explain the Syntax and Semantics of Propositional Logic.	[4M]
		OR	
4	*A*	Explain forward chaining and backward chaining	[7M]
	B	Compare and contrast the two variants of Logic-Predicate and Propositional.	[7M]

SECTION-III

5	*A*	Explain the issues in Knowledge Representation. Define Inheritance in Semantic Net.	[8M]
	B	Differentiate between monotonic and non monotonic reasoning.	[6M]
		OR	
6	*A*	Explain acting under uncertainity domain	[5M]
	B	Explain Bayesian Networks?	[9M]

SECTION-IV

7	*A*	Differentiate between Supervised Learning and Unsupervised Learning.	[4M]
	B	Discuss Winston's learning briefly with neat sketch.	[10M]
		OR	
8	*A*	Describe the role of information gain in Decision Tree Learning.	[7M]
	B	Explain decision tree algorithm.	[7M]

SECTION-V

9	*A*	Explain the Phases in Building Expert System.	[9M]
	B	Explain the Applications of the Expert Systems.	[5M]
		OR	
10	*A*	List the Characteristics of Expert Systems. Classify various Expert System shells and tools.	[8M]
	B	Explain about MYCIN Expert system in detail.	[6M]

****

R17

Code No: R17A1204

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech I Semester Supplementary Examinations, November 2022

Artificial Intelligence

(CSE)

Roll No										

Time: 3 hours **Max. Marks: 70**

Note: This question paper Consists of 5 Sections. Answer **FIVE** Questions, Choosing ONE Question from each SECTION and each Question carries 14 marks.

***

SECTION-I

1	***A***	Briefly explain how AI Technique can be represented and list out some of the task domain of AI.	**[7M]**
	B	How to define a problem as state space search? Discuss it with the help of an example.	**[7M]**
		OR	
2	***A***	Explain any one algorithm with the help of an example.	**[3M]**
		i. Hill Climbing: Steepest Ascent.	**[4M]**
		ii. Constraints Satisfaction	
	B	Identify the type of control strategy is used in the 8-puzzle problem. Explain	**[7M]**

SECTION-II

3	***A***	Justify the need for minimax algorithm. Explicate the steps of minimax algorithm	**[7M]**
	B	Explain Non – Monotonic reasoning and discuss the various logic associated with it.	**[7M]**
		OR	
4	***A***	Define the syntactic elements of first-Order logic	**[7M]**
	B	Explain in detail about forward chaining algorithm with example.	**[7M]**

SECTION-III

5		List out the steps involved in the knowledge Engineering process. Explain with an example.	**[14M]**
		OR	
6		Discuss about Bayesian Theory and Bayesian Network.	**[14M]**

SECTION-IV

7	***A***	Define learning. Summarize the learning from examples technique.	**[7M]**
	B	Explain the Winston's Learning Program.	**[7M]**
		OR	
8	***A***	What is a decision tree? Write the decision tree learning algorithm.	**[7M]**
	B	Explain the process of inducing decision trees from examples.	**[7M]**

SECTION-V

9	***A***	Outline stages in the development of an expert systems.	**[7M]**
	B	Summarize the expert system shells and tools.	**[7M]**
		OR	
10	***A***	Illustrate the Knowledge Acquisition system.	**[7M]**
	B	Explain the applications and domains in Expert systems.	**[7M]**

****

R18

Code No: R18A1205

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

III B.Tech I Semester Supplementary Examinations, June 2022

Artificial Intelligence

(EEE, CSE & IT)

Roll No										

Time: 3 hours **Max. Marks: 70**

Answer Any **Five** Questions
All Questions carries equal marks.
***

1	Define AI problems and its components. Explain how a problem solving agent works? Explain real-world AI problems with examples	**[14M]**
2	What is best first search? Explain in detail A* algorithm? Discuss BFS Algorithm	**[14M]**
3	Explain in detail about alpha-beta pruning with example.	**[14M]**
4	What is First Order Logic? State and Prove Baye's Theorem and mention its applications?	**[14M]**
5	Give a detail note on a generic knowledge-based agent. In the wumpus world, agent will have five sensors. Mention Various Other Knowledge Representation Schemes	**[14M]**
6	Prove the following assertion: for every game tree, the utility obtain by MAX using mini max decision against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN. Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?	**[14M]**
7	Discuss in detail about Winston's Learning Program with its implementation details.	**[14M]**
8	What is an Expert System? List various components of Knowledge Base? Differentiate Forward and Backward Chaining?	**[14M]**

****

R17

Code No: R17A1204

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech I Semester Supplementary Examinations, June 2022

Artificial Intelligence

(CSE)

Roll No										

Time: 3 hours **Max. Marks: 70**

Answer Any **Five** Questions
All Questions carries equal marks.
***

1 a. Explain depth first search strategy algorithm with suitable example. **[7M]**
b. Define Constraint Satisfaction Problem. Explain Constraint Satisfaction Problem for map colouring. **[7M]**

2 a. Define Agent Program. Explain the Following Agent Programs with Respect to Intelligent Systems **[5M]**
   i. Goal-Based Reflex Agent
   ii. Utility-Based Agent
b. Explain the following Heuristic Search Strategies with Suitable Examples: **[9M]**
   i. Generic Best-First Algorithm
   ii. A * Algorithm

3 a. Explain alpha –beta pruning search algorithm. **[7M]**
b. Explain the Symbols and Interpretation of First Order Logic **[7M]**

4 a. Explain MinMax Search Algorithm. **[5M]**
b. Explain the Forward Chaining and backward chaining Algorithms with suitable **[9M]**

5 a. Explain the Baye's Rule and Its Applications in Artificial Intelligence. **[7M]**
b. Differentiate between monotic and non-monotic reasoning. **[7M]**

6 a. With an Example, Discuss Conditional and Unconditional Probability **[7M]**
b. Define Bayesian Network. Explain the Semantics of Bayesian Network with suitable example. **[7M]**

7 a. Explain Winston's Learning Program with an Example **[7M]**
b. Explain the Following Forms of Learning **[7M]**
   i. Rote Learning
   ii. Reinforcement Learning

8 a. Explain the Knowledge Acquisition with a neat schematic diagram. **[7M]**
b. Explain the architecture of an Expert System and discuss the working of MYCIN Expert System **[7M]**

****

R18

Code No: R18A0526

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech I Semester Regular/Supplementary Examinations, November 2022

Machine Learning

(CSE & IT)

Roll No										

Time: 3 hours **Max. Marks: 70**

Note: This question paper Consists of 5 Sections. Answer **FIVE** Questions, Choosing ONE Question from each SECTION and each Question carries 14 marks.

***

		<u>SECTION-I</u>	
1	***A***	Explain components involved in the design of a learning system	**[7M]**
	B	Discuss different perspectives and issues in machine learning.	**[7M]**
		OR	
2		Explain different learning models of machine learning	**[14M]**
		<u>SECTION-II</u>	
3	***A***	Explain ID3 algorithm with example.	**[10M]**
	B	Explain different types of SVM algorithm:	**[4M]**
		OR	
4	***A***	Discuss the step wise analysis of k-means clustering algorithm	**[10M]**
	B	With a neat sketch explain the architecture of an artificial neural network.	**[4M]**
		<u>SECTION-III</u>	
5	***A***	What is the importance of ensemble learning? Explain the different methods involved in it .	**[4M]**
	B	Discuss Expectation-Maximization (EM) Algorithm	**[10M]**
		OR	
6	***A***	What are the advantages and disadvantages of random forest algorithm.	**[7M]**
	B	Explain the procedure of Multiexpert combination method	**[7M]**
		<u>SECTION-IV</u>	
7	***A***	Explain the an algorithm for Learning Q.	**[10M]**
	B	Define PAC-learnability with suitable example.	**[4M]**
		OR	
8		How to compute optimal policy? Explain with example.	**[14M]**
		<u>SECTION-V</u>	
9	***A***	Explain different Genetic Operators of Genetic Algorithm.	**[7M]**
	B	Define Hypothesis Space Search of Genetic Algorithm.	**[7M]**
		OR	
10		Discuss about Baldwin effect and Lamarckian evolution	**[14M]**

***

R20

Code No: R20A6601

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

III B.Tech I Semester Regular Examinations, December 2022

Machine Learning

(CSE-AIML)

Roll No										

Time: 3 hours **Max. Marks: 70**

Note: This question paper Consists of 5 Sections. Answer **FIVE** Questions, Choosing ONE Question from each SECTION and each Question carries 14 marks.

***

SECTION-I

1 **[14M]**

Colour	Hair	Height	Eyes	Class
White	Blond	2	brown	+
Black	Dark	4	brown	-
Brown	Red	2	Blue	+
Brown	Blond	4	Blue	-
Black	Blond	2	Blue	-
Black	Red	2	Brown	+
White	Blond	2	Black	-
Black	Red	4	Blue	-

Apply find 'S' algorithm and candidate elimination algorithm to find the version space

OR

2 The following table contains training examples for a classification problem. Use ID3 algorithm to construct a minimal decision tree that predicts the class label. Show each step of the computation. **[14M]**

ID	Temperature	rain	wind	Visibility	Class
1	Hot	No	Mid	NG	Yes
2	Cool	Yes	Mid	NG	Not
3	Hot	Yes	High	NG	Not
4	Hot	No	Low	Good	Yes
5	Comfort	Yes	Mid	Bad	Not
6	Hot	Yes	High	NG	Not
7	Cool	No	Low	Bad	Yes
8	Hot	No	Low	Good	Yes
9	Comfort	No	Low	Bad	Yes
10	Hot	Yes	Mid	Good	Not
11	Comfort	No	High	Good	Yes
12	Cool	No	Mid	NG	Yes
13	Cool	Yes	High	NG	Not
14	Comfort	Yes	Mid	NG	Not

SECTION-II

3 *A* .An aptitude test and statistics tests was conducted randomly for 5 students and the scores were depicted in the table. **[10M]**

Student	Aptitude score	Statistics score
1	60	70
2	70	85

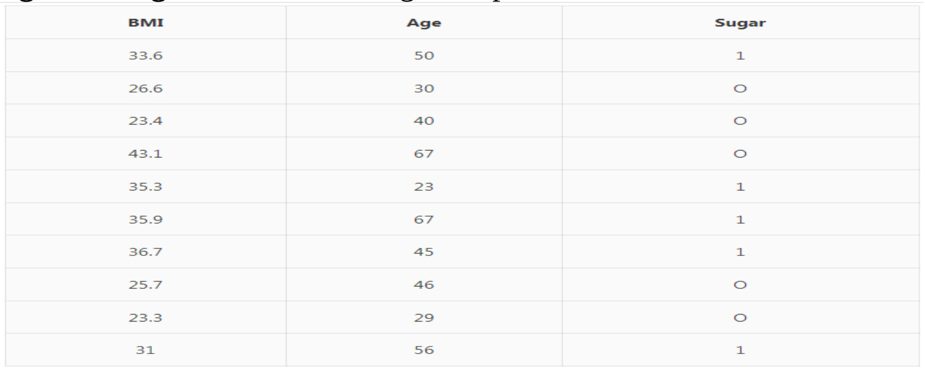
3	80	65
4	85	95
5	95	70

Based on math aptitude ratings, which linear regression equation best predicts statistics performance?
What grade would we anticipate a student to get in statistics if the student scored an 80 on the aptitude test?

B Compare the Multi linear regression, Polynomial regression, and Logistic regression. Provide the applications of each. **[4M]**

OR

4 ***A*** **Apply** K nearest neighbor classifier to predict the Sugar of diabetic patient with the given features BMI, Age. Assume K=3, **Test Example BMI=43.6, Age=40, Sugar=?** If the training examples are **[10M]**



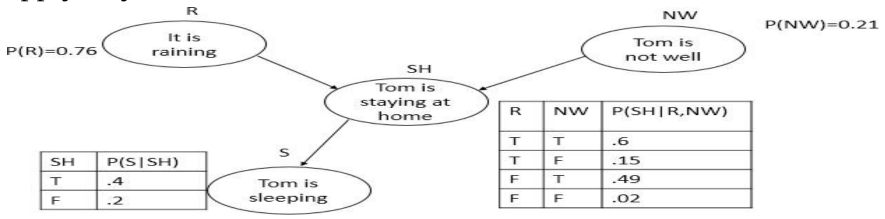
BMI	Age	Sugar
33.6	50	1
26.6	30	0
23.4	40	0
43.1	67	0
35.3	23	1
35.9	67	1
36.7	45	1
25.7	46	0
23.3	29	0
31	56	1

B What is case based reasoning explain with an example. **[4M]**

SECTION-III

5 ***A*** Write the steps in Naive Bayes algorithms for learning and classifying text. **[4M]**

B Apply Bayesian belief network, **[10M]**



SH	P(S|SH)
T	.4
F	.2

R	NW	P(SH|R,NW)
T	T	.6
T	F	.15
F	T	.49
F	F	.02

What is the probability that Tom is not sleeping although it is raining heavily and he is not well? He was upset that he could not join for the sleep over in his friend's home and his mom forced him to stay at home.

OR

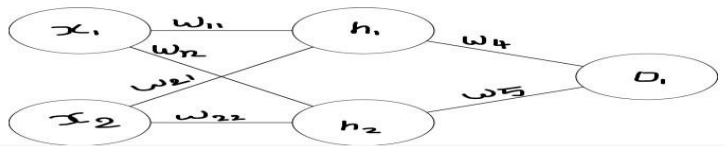
6 A Describe the naive Bayes theorem in text classification **[7M]**

B What are the limitations of Bayes optimal classifier? Explain how does the Gibbs algorithm tries to resolve the issues. **[7M]**

SECTION-IV

7 ***A*** Explain the steps in of the BACKPROPAGATION algorithm for feedforward networks that contains two layers of sigmoid units. **[7M]**

B Consider the Artificial Neural Network with the following values **[7M]**
X1=1,x2=0, w11=0.25,w12=0.10,w21=0.15,w22=0.1,w4=0.3,w5=0.4,b1 to h_1 and h_2=1 and bias b2 to O_1=1. Assume the actual output=0.95. Find the predicted output. Find the error and through backpropagation, find out the weights of w4 and w5 provided learning rate =0.3.



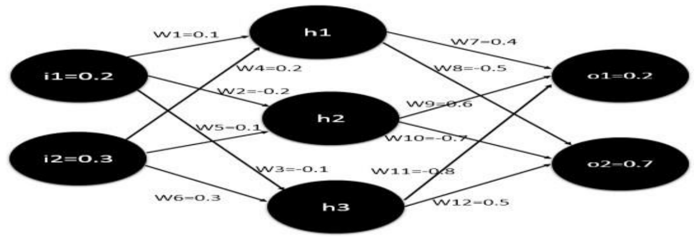
Input values x1, and x2, randomly assigned weights are w1, w2, w3, w4, w5, w6, w7 and w8. Target values o1 = 0.05 and o2 = 0.95. Bias values b1 and b2.

Use the sigmoid activation function. Learning rate **α** = 0.5.

OR

8 Consider the following neural network with the input, output and weight parameters values shown in the diagram. The activation values in each neuron is calculated using the sigmoid activation function. Now, answer the following: **[14M]**

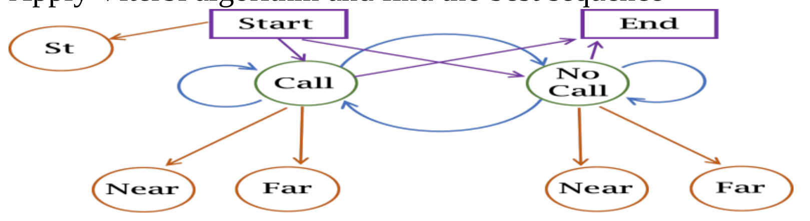
a. For the given input i1 and i2 as shown in the diagram, compute the output of the hidden layer and output layer neurons.
b. Compute the error in the network with the initialized weight parameters shown in the diagram.
c. Update the weight parameters for w7 and w8 using backpropagation algorithm in the first iteration. Consider learning rate as 0.01.



SECTION-V

9 ***A*** Illustrate how to find the state sequence with any example. **[4M]**

B Apply Viterbi algorithm and find the best sequence 10 **[10M]**



	Call	NoCall	End
Start	0.7	0.3	0
Call	0.2	0.7	0.1
NoCall	0.7	0.2	0.1

State transition probabilities

	St	Near	Far
Start	1	0	0
Call	0	0.7	0.3
NoCall	0	0.4	0.6

Emission probabilities

Find the hidden state sequence for Near Far Far.

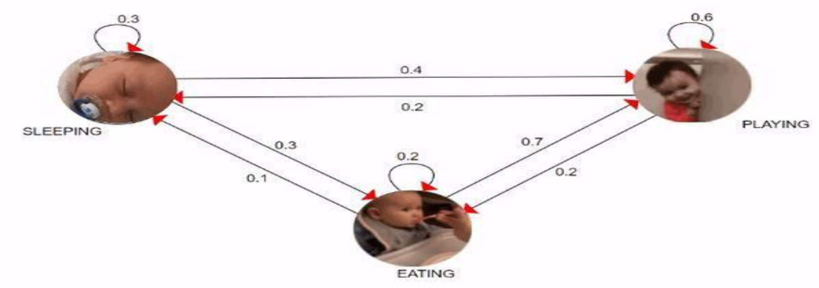
OR

10 ***A*** Describe the three states in Hidden Markov Models. **[4M]**

B Find out the transition matrix from the below diagram. Assume the initial probabilities for sleeping, eating and playing are denoted as
π={ 0.25,0.25,0.50} .

i. Find the probability of the series .Baby is sleeping, sleeping, eating, playing ,playing, sleeping. **[5M]**

ii. Find the probability of baby playing given baby is eating . **[5M]**



****

Code No: **R17A0534** **R17**

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech- II Semester Advance Supplementary Examinations, July 2022

Machine Learning

(CSE)

Roll No										

Time: 3 hours **Max. Marks: 70**

Answer Any **Five** Questions

All Questions carries equal marks.

***

1 Briefly explain about the various learning models in machine learning. **[14M]**

2 a) Define VC dimension. How VC dimension is related with no of training examples used for learning. **[8M]**

b) Briefly explain about PAC learning framework. **[6M]**

3 a) What are the benefits of pruning in decision tree induction? Explain different approaches to tree pruning? **[8M]**

b) Explain the concept of a Perceptron with a neat diagram. **[6M]**

4 a) Explain how Support Vector Machine can be used for classification of linearly separable data. **[8M]**

b) Give a detail note on kernel functions. **[6M]**

5 Describe boosting and ADA boosting algorithm with neat sketch **[14M]**

6 Explain the concept of EM Algorithm with Gaussian Mixtures model. **[14M]**

7 Summarize about the Q-learning model and explain with diagram **[14M]**

8 a) List out the Genetic algorithm steps with example. **[7M]**

b) Illustrate the prototypical genetic algorithm. **[7M]**

****

R18

Code No: R18A0526

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech I Semester Supplementary Examinations, June 2022

Machine Learning

(CSE & IT)

Roll No										

Time: 3 hours **Max. Marks: 70**

Answer Any **Five** Questions

All Questions carries equal marks.

***

1	Define Machine Learning? Explain different Types of Learning with example each?	**[14M]**
2	Explain about probabilistic and geometric models in machine learning?	**[14M]**
3	What is the difference between logistic regression and linear regression give an example?	**[14M]**
4	Write short notes on:	
	a) Multiple regression	**[7M]**
	b) Back propagation algorithm	**[7M]**
5	Analyze the Expectation-Maximisation (EM) Algorithm with an example?	**[14M]**
6	Explain K-Nearest Neighbor(KNN) Algorithm with an example and list the advantages and disadvantages of K-NN	**[14M]**
7	a) What is the goal of the support vector machine (SVM)? How to compute the margin?	**[7M]**
	b) What are the elements of reinforcement learning?	**[7M]**
8	Explain Genetic Programming with an example? And What are the operators of genetic algorithm?	**[14M]**

****

R17

Code No: R17A0534

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

IV B.Tech- II Semester Supplementary Examinations, May 2022

Machine Learning

(CSE)

Roll No										

Time: 3 hours **Max. Marks: 70**

Note: This question paper Consists of 5 Sections. Answer **FIVE** Questions, Choosing ONE Question from each SECTION and each Question carries 14 marks.

***

SECTION-I

1 a) Explain the useful perspectives of machine learning in different applications. **[7M]**

b) Describe in detail the rule for estimating training values. **[7M]**

OR

2 a) Differentiate between Supervised, Unsupervised and Reinforcement Learning. **[7M]**

b) Define the terms Hypothesis space and Version space. Illustrate with an example. **[7M]**

SECTION-II

3 a) Give the necessary steps and limitations of ID3 algorithm. **[7M]**

b) Explain about the linear regression. **[7M]**

OR

4 a) Explain how Support Vector Machine can be used for classification of linearly separable data. **[7M]**

b) Elucidate K-means algorithm with neat diagram. **[7M]**

SECTION-III

5 a) Describe the random forest algorithm to improve classifier accuracy **[7M]**

b) Explain the concept of Bagging with its uses? **[7M]**

OR

6 a) How can be the data classified using KNN algorithm with neat sketch? **[7M]**

b) Discuss the various distance measure algorithms. **[7M]**

SECTION-IV

7 a) Write about the learning Rule sets. **[7M]**

b) Write some common evaluation functions in the learning rule sets. **[7M]**

OR

8 Explain normal and Binomial Distributions with an example. **[14M]**

SECTION-V

9 a) Discuss about the mutation operator. **[7M]**

b) Examine how genetic algorithm searches large space of candidate objects with an example with fitness function **[7M]**

OR

10 Assess the parallelizing Genetic Algorithms with an example. **[14M]**

****

UNIT 1

Introduction

- AI problems
- Agents and Environments
- Structure of Agents
- Problem Solving Agents Basic Search Strategies:
  a. Problem Spaces
  b. Uninformed Search (Breadth-First, Depth-First Search, Depth-first with Iterative Deepening)
  c. Heuristic Search (Hill Climbing, Generic Best-First, A*)
  d. Constraint Satisfaction (Backtracking, Local Search)

Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines *"man-made,"* and intelligence defines *"thinking power"*, hence AI means *"a man-made thinking power."*

""It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions." "

A MACHINE HAS TO WORK AS A HUMAN

With the advancement of technology we are geared up with race in creating a new revolution in the world by making intelligent machines.The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

Intelligence is nothing but responding on its own, and can be listed as:

- Reasoning
- Learning
- Problem-Solving
- Perception
- Linguistic Intelligence

The main focus of artificial intelligence is towards understanding human behavior and performance. This can be done by creating computers with human-like intelligence and capabilities. This includes natural language processing, facial analysis and robotics. The main applications of AI are in military, healthcare, and computing; however, it's expected that these applications will start soon and become part of our everyday lives.

Uses of Artificial Intelligence:
Artificial Intelligence has many practical applications across various industries and domains, including:

1. **Healthcare:** AI is used for medical diagnosis, drug discovery, and predictive analysis of diseases.
2. **Finance:** AI helps in credit scoring, fraud detection, and financial forecasting.
3. **Retail:** AI is used for product recommendations, price optimization, and supply chain management.
4. **Manufacturing:** AI helps in quality control, predictive maintenance, and production optimization.
5. **Transportation:** AI is used for autonomous vehicles, traffic prediction, and route optimization.
6. **Customer service:** AI-powered chatbots are used for customer support, answering frequently asked questions, and handling simple requests.
7. **Security:** AI is used for facial recognition, intrusion detection, and cybersecurity threat analysis.

8. **Marketing:** AI is used for targeted advertising, customer segmentation, and sentiment analysis.
9. **Education:** AI is used for personalized learning, adaptive testing, and intelligent tutoring systems.

Forms of AI:

1) Weak AI:

- Weak AI is an AI that is created to solve a particular problem or perform a specific task.
- It is not a general AI and is only used for specific purpose.
- For example, the AI that was used to beat the chess grandmaster is a weak AI as that serves only 1 purpose but it can do it efficiently.

2) Strong AI:

- Strong AI is difficult to create than weak AI.
- It is a general purpose intelligence that can demonstrate human abilities.
- Human abilities such as learning from experience, reasoning, etc. can be demonstrated by this AI.

3) Super Intelligence

- As stated by a leading AI thinker Nick Bostrom, "Super Intelligence is an AI that is much smarter than the best human brains in practically every field".
- It ranges from a machine being just smarter than a human to a machine being trillion times smarter than a human.
- Super Intelligence is the ultimate power of AI.

The history of artificial intelligence:

- In ancient times, inventors made things called “automatons” which were mechanical and moved independently of human intervention. The word “automaton” comes from ancient Greek, and means “acting of one’s own will.” One of the earliest records of an automaton comes from 400 BCE and refers to a mechanical pigeon created by a friend of the philosopher Plato.
- Many years later, one of the most famous automatons was created by Leonardo da Vinci around the year 1495.

Groundwork for AI:

The ground work started in 1900-1950, there were many studies going on to create a machine with artificial humans. There were many questions rising in scientists whether is it possible to create an artificial brain?

- **1921:** Czech playwright Karel Capek released a science fiction play “Rossum’s Universal Robots” which introduced the idea of “artificial people” which he named robots. This was the first known use of the word.
- **1929:** Japanese professor Makoto Nishimura built the first Japanese robot, named Gakutensoku.
- **1949:** Computer scientist Edmund Callis Berkley published the book “Giant Brains, or Machines that Think” which compared the newer models of computers to human brains.

Birth of AI: 1950-1956

During this time the interest in AI really took peak achievement in research when Alan Turing published his work “Computer Machinery and Intelligence” which eventually became The Turing Test, which experts used to measure computer intelligence. The term “artificial intelligence” was coined and came into popular use.

- **1950:** Alan Turing published “Computer Machinery and Intelligence” which proposed a test of machine intelligence called The Imitation Game.
- **1952:** A computer scientist named Arthur Samuel developed a program to play checkers, which is the first to ever learn the game independently.
- **1955:** John McCarthy held a workshop at Dartmouth on “artificial intelligence” which is the first use of the word, and how it came into popular usage.

AI maturation: 1957-1979

This was the span of time where the phrase “artificial intelligence” was created, and the 1980s was a period of both rapid growth and struggle for AI research. The late 1950s through the 1960s was a time of creation. From programming languages that are still in use to this day to books and films that explored the idea of robots, AI became a mainstream idea quickly.

- **1958:** John McCarthy created LISP (acronym for List Processing), the first programming language for AI research, which is still in popular use to this day.

- **1959:** Arthur Samuel created the term “machine learning” when doing a speech about teaching machines to play chess better than the humans who programmed them.
- **1961:** The first industrial robot Unimate started working on an assembly line at General Motors in New Jersey, tasked with transporting die casings and welding parts on cars (which was deemed too dangerous for humans).
- **1965:** Edward Feigenbaum and Joshua Lederberg created the first “expert system” which was a form of AI programmed to replicate the thinking and decision-making abilities of human experts.
- **1966:** Joseph Weizenbaum created the first “chatterbot” (later shortened to chatbot), ELIZA, a mock psychotherapist, that used natural language processing (NLP) to converse with humans.1968: Soviet mathematician Alexey Ivakhnenko published “Group Method of Data Handling” in the journal “Avtomatika,” which proposed a new approach to AI that would later become what we now know as “Deep Learning.”
- **1973:** An applied mathematician named James Lighthill gave a report to the British Science Council, underlining that strides were not as impressive as those that had been promised by scientists, which led to much-reduced support and funding for AI research from the British government.
- **1979:** James L. Adams created The Standford Cart in 1961, which became one of the first examples of an autonomous vehicle. In ‘79, it successfully navigated a room full of chairs without human interference.
- **1979:** The American Association of Artificial Intelligence which is now known as the Association for the Advancement of Artificial Intelligence (AAAI) was founded.

AI boom: 1980-1987

Most of the 1980s showed a period of rapid growth and interest in AI, now labeled as the “AI boom.” This came from both breakthroughs in research, and additional government funding to support the researchers. Deep Learning techniques and the use of Expert System became more popular, both of which allowed computers to learn from their mistakes and make independent decisions.

Notable dates in this time period include:

- **1980:** First conference of the AAAI was held at Stanford.
- **1980:** The first expert system came into the commercial market, known as XCON (expert configurer). It was designed to assist in the ordering of computer systems by automatically picking components based on the customer’s needs.
- **1981:** The Japanese government allocated $850 million (over $2 billion dollars in today’s money) to the Fifth Generation Computer project. Their aim was to create computers that could translate, converse in human language, and express reasoning on a human level.
- **1984:** The AAAI warns of an incoming “AI Winter” where funding and interest would decrease, and make research significantly more difficult.

- **1985:** An autonomous drawing program known as AARON is demonstrated at the AAAI conference.
- **1986:** Ernst Dickmann and his team at Bundeswehr University of Munich created and demonstrated the first driverless car (or robot car). It could drive up to 55 mph on roads that didn't have other obstacles or human drivers.
- **1987:** Commercial launch of Alacrity by Alactrious Inc. Alacrity was the first strategy managerial advisory system, and used a complex expert system with 3,000+ rules.

AI winter: 1987-1993

As the AAAI warned, an AI Winter came. The term describes a period of low consumer, public, and private interest in AI which leads to decreased research funding, which, in turn, leads to few breakthroughs. Both private investors and the government lost interest in AI and halted their funding due to high cost versus seemingly low return. This AI Winter came about because of some setbacks in the machine market and expert systems, including the end of the Fifth Generation project, cutbacks in strategic computing initiatives, and a slowdown in the deployment of expert systems.

- **1987:** The market for specialized LISP-based hardware collapsed due to cheaper and more accessible competitors that could run LISP software, including those offered by IBM and Apple. This caused many specialized LISP companies to fail as the technology was now easily accessible.
- **1988:** A computer programmer named Rollo Carpenter invented the chatbotJabberwacky, which he programmed to provide interesting and entertaining conversation to humans.

AI agents: 1993-2011

Despite the lack of funding during the AI Winter, the early 90s showed some impressive strides forward in AI research, including the introduction of the first AI system that could beat a reigning world champion chess player. This era also introduced AI into everyday life via innovations such as the first Roomba and the first commercially-available speech recognition software on Windows computers.

The surge in interest was followed by a surge in funding for research, which allowed even more progress to be made.

- **1997:** Deep Blue (developed by IBM) beat the world chess champion, Gary Kasparov, in a highly-publicized match, becoming the first program to beat a human chess champion.
- **1997:** Windows released a speech recognition software (developed by Dragon Systems).
- **2000:** Professor Cynthia Breazeal developed the first robot that could simulate human emotions with its face,which included eyes, eyebrows, ears, and a mouth. It was called Kismet.
- **2002:** The first Roomba was released.
- **2003:** Nasa landed two rovers onto Mars (Spirit and Opportunity) and they navigated the surface of the planet without human intervention.

- **2006:** Companies such as Twitter, Facebook, and Netflix started utilizing AI as a part of their advertising and user experience (UX) algorithms.
- **2010:** Microsoft launched the Xbox 360 Kinect, the first gaming hardware designed to track body movement and translate it into gaming directions.
- **2011:** An NLP computer programmed to answer questions named Watson (created by IBM) won Jeopardy against two former champions in a televised game.
- **2011:** Apple released Siri, the first popular virtual assistant.

Artificial General Intelligence: 2012-present

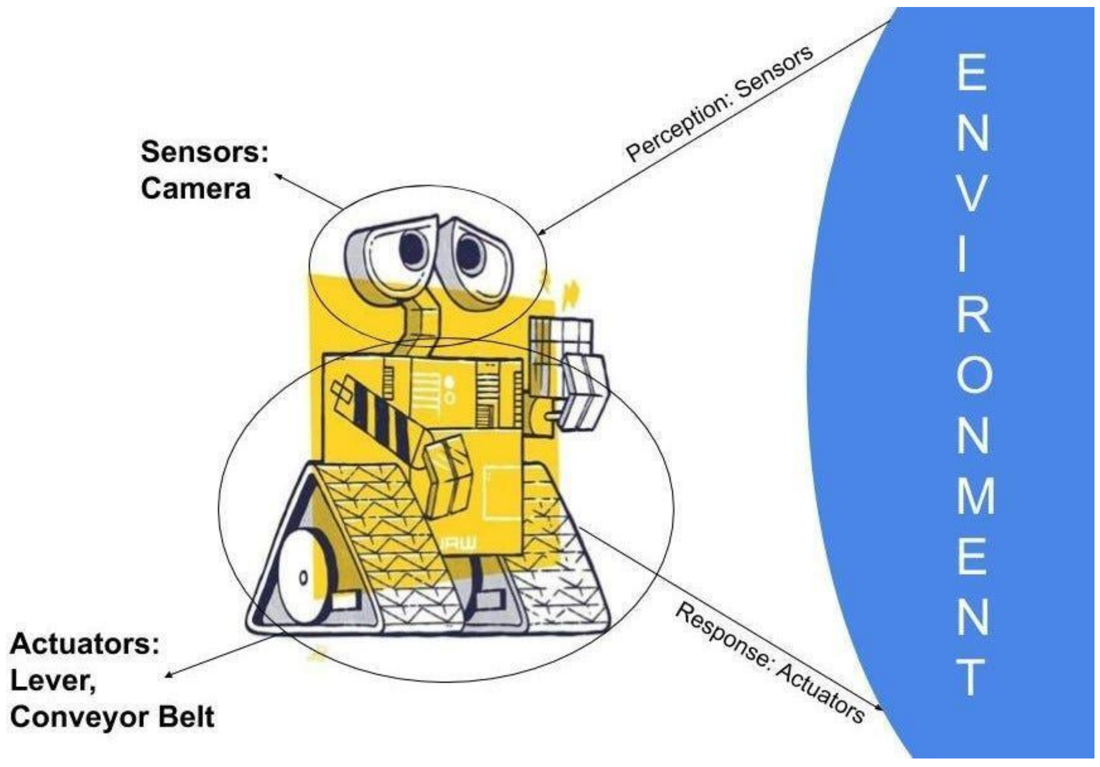
That brings us to the most recent developments in AI, up to the present day. We've seen a surge in common-use AI tools, such as virtual assistants, search engines, etc. This time period also popularized Deep Learning and Big Data..

- **2012:** Two researchers from Google (Jeff Dean and Andrew Ng) trained a neural network to recognize cats by showing it unlabeled images and no background information.
- **2015:** Elon Musk, Stephen Hawking, and Steve Wozniak (and over 3,000 others) signed an open letter to the worlds' government systems banning the development of (and later, use of) autonomous weapons for purposes of war.
- **2016:** Hanson Robotics created a humanoid robot named Sophia, who became known as the first "robot citizen" and was the first robot created with a realistic human appearance and the ability to see and replicate emotions, as well as to communicate.
- **2017:** Facebook programmed two AI chatbots to converse and learn how to negotiate, but as they went back and forth they ended up forgoing English and developing their own language, completely autonomously.
- **2018:** A Chinese tech group called Alibaba's language-processing AI beat human intellect on a Stanford reading and comprehension test.
- **2019:** Google's AlphaStar reached Grandmaster on the video game StarCraft 2, outperforming all but .2% of human players.
- **2020:** OpenAI started beta testing GPT-3, a model that uses Deep Learning to create code, poetry, and other such language and writing tasks. While not the first of its kind, it is the first that creates content almost indistinguishable from those created by humans.
- **2021:** OpenAI developed DALL-E, which can process and understand images enough to produce accurate captions, moving AI one step closer to understanding the visual world.
- **2022:**
- **2023:**

An AI system is composed of an agent and its environment.

An agent(e.g., human or robot) is anything that can perceive its environment through sensors and acts upon that environment through effectors. Intelligent agents must be able to set goals and achieve them. In classical planning problems, the agent can assume that it is the only system acting in the world, allowing the agent to be certain of the consequences of its actions. However, if the agent is not the only actor, then it requires that the agent can reason under uncertainty.

This calls for an agent that cannot only assess its environment and make predictions but also evaluate its predictions and adapt based on its assessment. Natural language processing gives machines the ability to read and understand human language. Some straightforward applications of natural language processing include information retrieval, text mining, question answering, and machine translation. Machine perception is the ability to use input from sensors (such as cameras, microphones, sensors, etc.) to deduce aspects of the world. e.g., Computer Vision. Concepts such as game theory, and decision theory, necessitate that an agent can detect and model human emotions.



PEAS stands for performance measure, environment, actuators, and sensors. PEAS defines AI models and helps determine the task environment for an intelligent agent.

1. **Performance measure:** It defines the success of an agent. It evaluates the criteria that determines whether the system performs well.
2. **Environment:** It refers to the external context in which an AI system operates. It encapsulates the physical and virtual surroundings, including other agents, objects, and conditions.
3. **Actuators:** They are responsible for executing actions based on the decisions made. They interact with the environment to bring about desired changes.
4. **Sensors:** An agent observes and perceives its environment through sensors. Sensors provide input data to the system, enabling it to make informed decisions.

P: Performance Measure

E: Environment

A: Actuators

S: Sensors

Example:

- **Performance Measure:** **The performance is measured as How safe is the taxi, Fast as per the traffic, Comfortable ride, Maximize Customers**
- **Environment: The taxi must check and adapt to the Roads its travelling, Traffic , Pedestrians, Customers**
- **Actuators: According to the steering and controls the response resulting like accelerator, brake, signal, horn etc.**
- **Sensors: Displays or readable data like Cameras, LIDAR, Speedometers, GPS, Odometer, Engine Sensors, Keyboards**

Examples

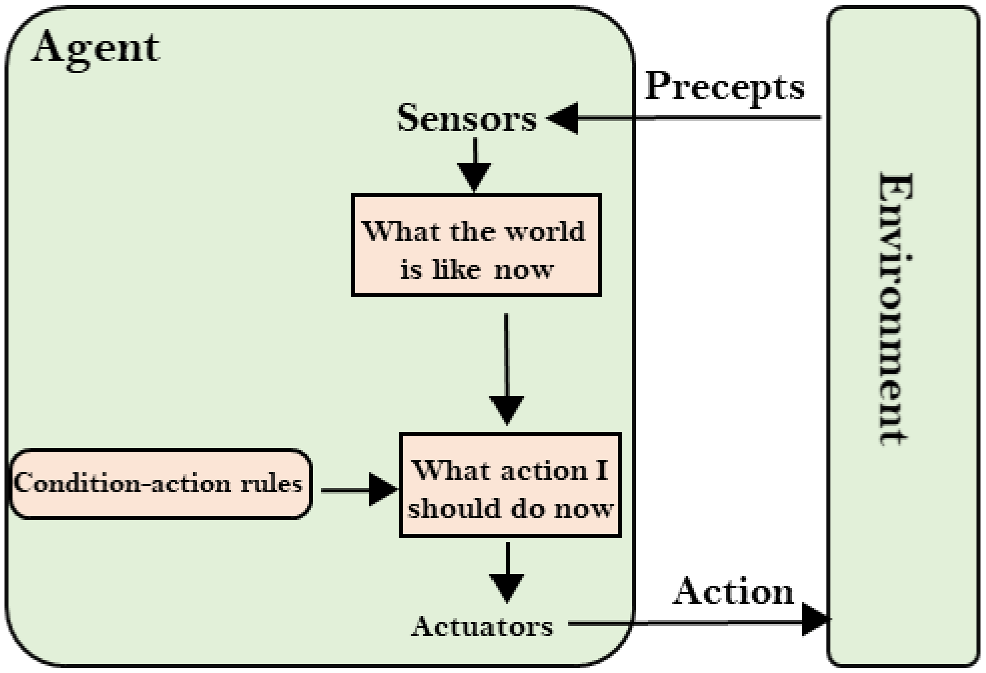
Agent	Performance measure	Environment	Actuators	Sensors
Vacuum cleaner	Cleanliness, security, battery	Room, table, carpet, floors	Wheels, brushes	Camera, sensors
Chatbot system	Helpful responses, accurate responses	Messaging platform, internet, website	Sender mechanism, typer	NLP algorithms
Autonomous vehicle	Efficient navigation, safety, time, comfort	Roads, traffic, pedestrians, road signs	Brake, accelerator, steer, horn	Cameras, GPS, speedometer
Hospital	Patient's health, cost	Doctors, patients, nurses, staff	Prescription, diagnosis, tests, treatments	Symptoms

Types of AI Agents

Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below:

- Simple Reflex Agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

1. Simple Reflex agent:

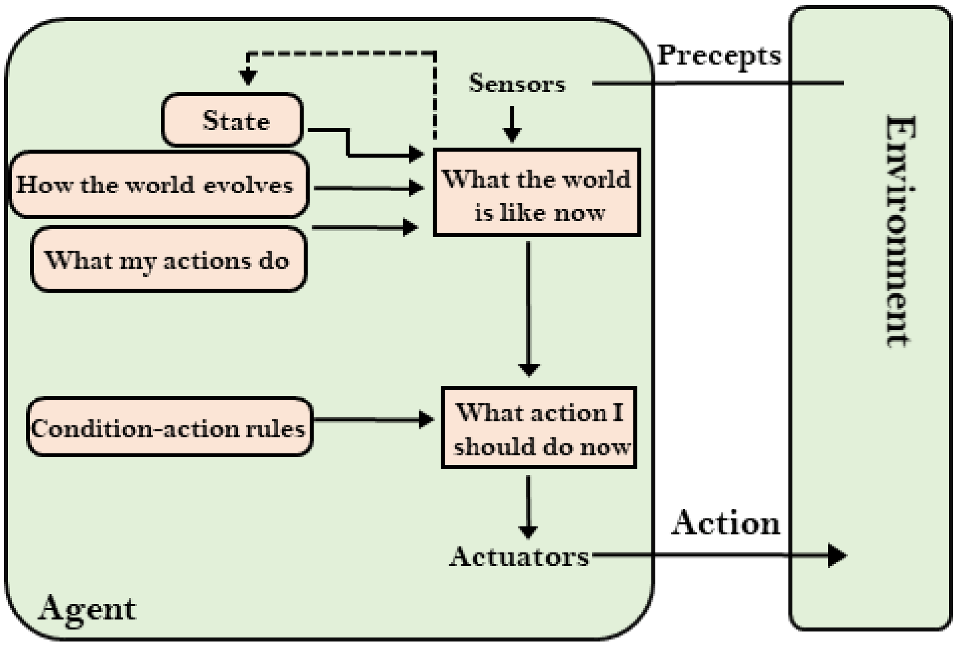


- The Simple reflex agents are the simplest agents. These agents take decisions on the basis of the current percepts and ignore the rest of the percept history.
- These agents only succeed in the fully observable environment.
- The Simple reflex agent does not consider any part of percepts history during their decision and action process.
- The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.
- Problems for the simple reflex agent design approach:

- They have very limited intelligence
- They do not have knowledge of non-perceptual parts of the current state
- Mostly too big to generate and to store.
- Not adaptive to changes in the environment.

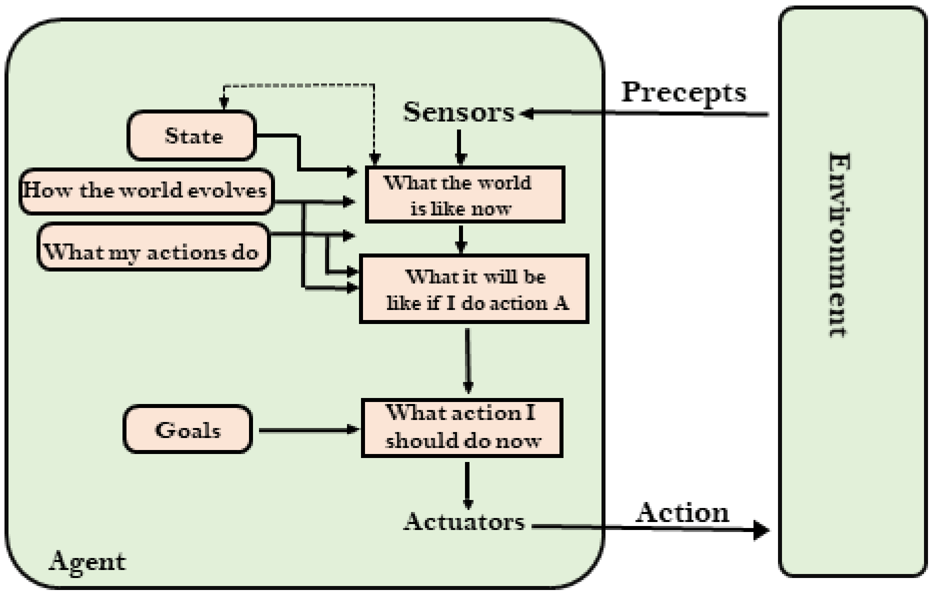
2. Model-based reflex agent

- The Model-based agent can work in a partially observable environment, and track the situation.
- A model-based agent has two important factors:
  - **Model:** It is knowledge about "how things happen in the world," so it is called a Model-based agent.
  - **Internal State:** It is a representation of the current state based on percept history.
- These agents have the model, "which is knowledge of the world" and based on the model they perform actions.
- Updating the agent state requires information about:
  a. How the world evolves
  b. How the agent's action affects the world.



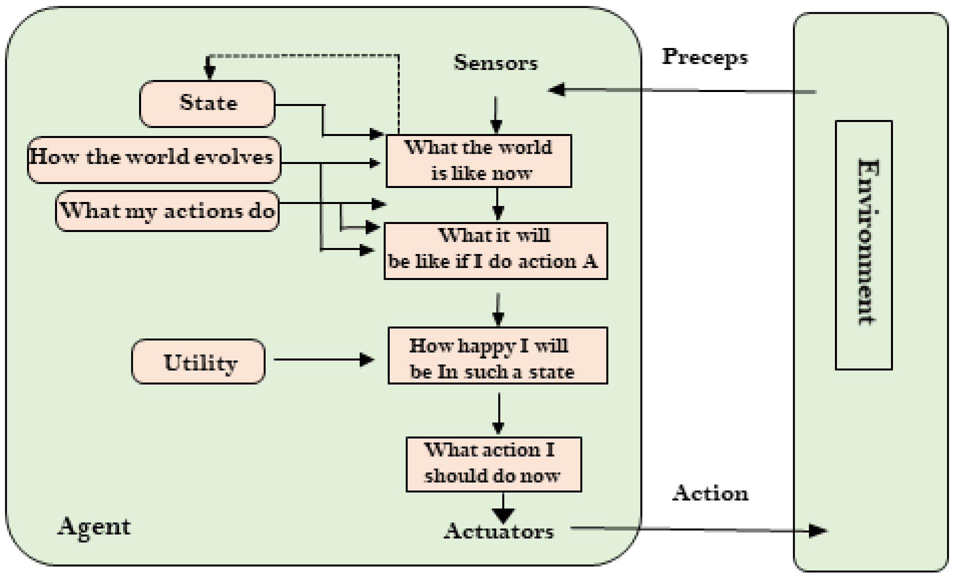
3. Goal-based agents

- The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.
- The agent needs to know its goal which describes desirable situations.
- Goal-based agents expand the capabilities of the model-based agent by having the "goal" information.
- They choose an action, so that they can achieve the goal.
- These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.



4. Utility-based agents

- These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.
- Utility-based agent act based not only goals but also the best way to achieve the goal.
- The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goals.


Sensors
Preceps
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be In such a state
What action I should do now
Environment
Action
Agent
Actuators

5. Learning Agents

- A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities.
- It starts to act with basic knowledge and then able to act and adapt automatically through learning.
- A learning agent has mainly four conceptual components, which are:
  a. **Learning element:** It is responsible for making improvements by learning from environment
  b. **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
  c. **Performance element:** It is responsible for selecting external action
  d. **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

Hence, learning agents are able to learn, analyze performance, and look for new ways to improve the performance.

Agent Environment in AI

An environment is everything in the world which surrounds the agent, but it is not a part of an agent itself. An environment can be described as a situation in which an agent is present.

The environment is where agent lives, operate and provide the agent with something to sense and act upon it. An environment is mostly said to be non-feministic.

Features of Environment

As per Russell and Norvig, an environment can have various features from the point of view of an agent:

1. Fully observable vs Partially Observable
2. Static vs Dynamic
3. Discrete vs Continuous
4. Deterministic vs Stochastic
5. Single-agent vs Multi-agent
6. Episodic vs sequential
7. Known vs Unknown
8. Accessible vs Inaccessible

1. Fully observable vs Partially Observable:

- If an agent sensor can sense or access the complete state of an environment at each point of time then it is **a fully observable** environment, else it is **partially observable**.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.
- An agent with no sensors in all environments then such an environment is called as **unobservable**.

2. Deterministic vs Stochastic:

- If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

3. Episodic vs Sequential:

- In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action.
- However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

4. Single-agent vs Multi-agent

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.
- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.
- The agent design problems in the multi-agent environment are different from single agent environment.

5. Static vs Dynamic:

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment.
- Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action.
- However for dynamic environment, agents need to keep looking at the world at each action.
- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

6. Discrete vs Continuous:

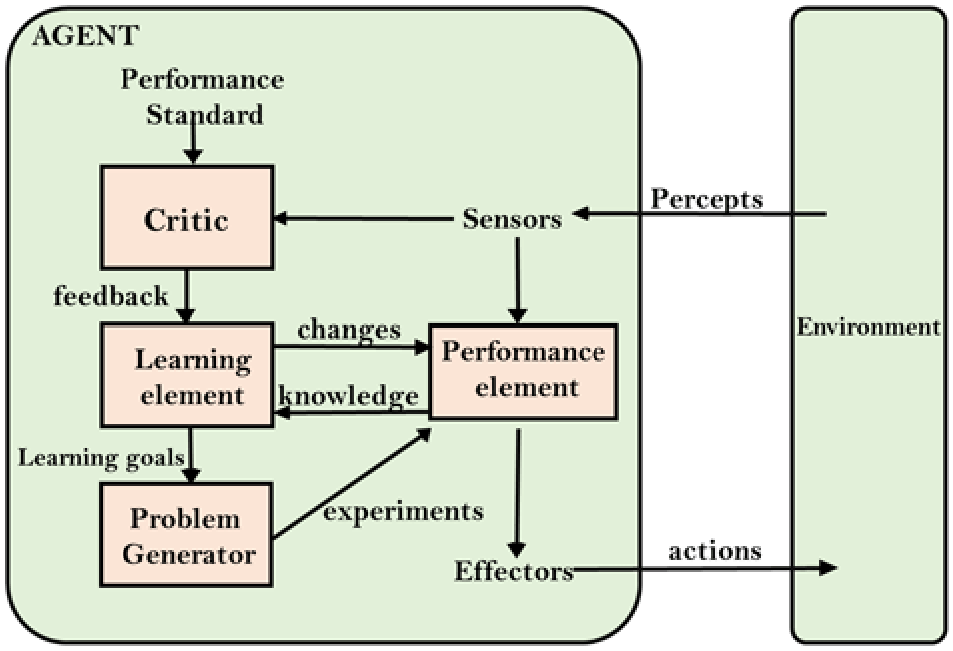
- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess gamecomes under discrete environment as there is a finite number of moves that can be performed.
- A self-driving car is an example of a continuous environment.

7. Known vs Unknown

- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.

8. Accessible vs Inaccessible

- If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.
- Information about an event on earth is an example of Inaccessible environment.



Problem-solving agents:

In Artificial Intelligence, Search techniques are universal problem-solving methods.

Rational agents or **Problem-solving agents** in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result. Problem-solving agents are the goal-based agents and use atomic representation. In this topic, we will learn various problem-solving search algorithms.

Search Algorithm Terminologies:

- **Search:** Searchingis a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
    a. **Search Space:** Search space represents a set of possible solutions, which a system may have.
    b. **Start State:** It is a state from where agent begins **the search**.
    c. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.

Search tree: A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.

Actions: It gives the description of all the available actions to the agent.

Transition model: A description of what each action do, can be represented as a transition model.

Path Cost: It is a function which assigns a numeric cost to each path.

Solution: It is an action sequence which leads from the start node to the goal node.

Optimal Solution: If a solution has the lowest cost among all solutions.

Properties of Search Algorithms:

Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

Completeness: A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

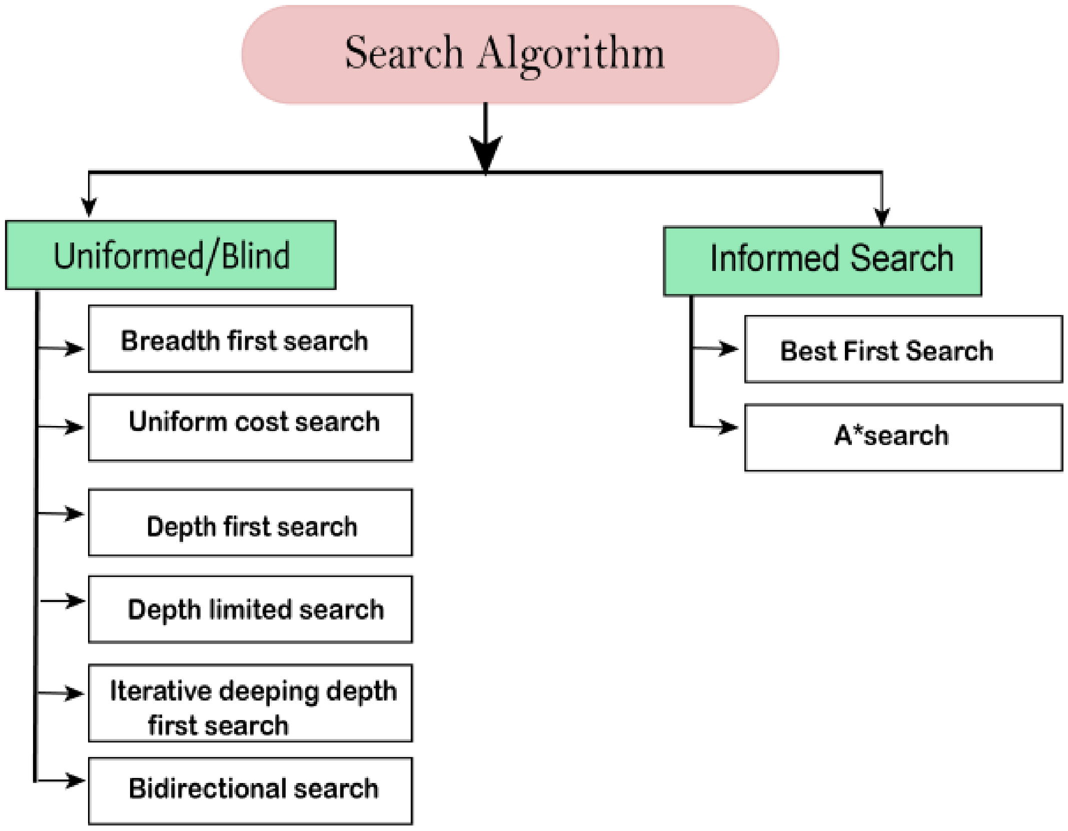
Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

Time Complexity: Time complexity is a measure of time for an algorithm to complete its task.

Space Complexity: It is the maximum storage space required at any point during the search, as the complexity of the problem.

Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.



Uninformed/Blind Search:

The uninformed search does not contain any domain knowledge such as closeness, the location of the goal. It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search.It examines each node of the tree until it achieves the goal node.

It can be divided into five main types:

- Breadth-first search
- Uniform cost search
- Depth-first search

- Iterative deepening depth-first search
- Bidirectional Search

Informed Search

Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy. Informed search is also called a Heuristic search.

A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Informed search can solve much complex problem which could not be solved in another way.

An example of informed search algorithms is a traveling salesman problem.

1. Greedy Search
2. A* Search

1. Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

Advantages:

- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

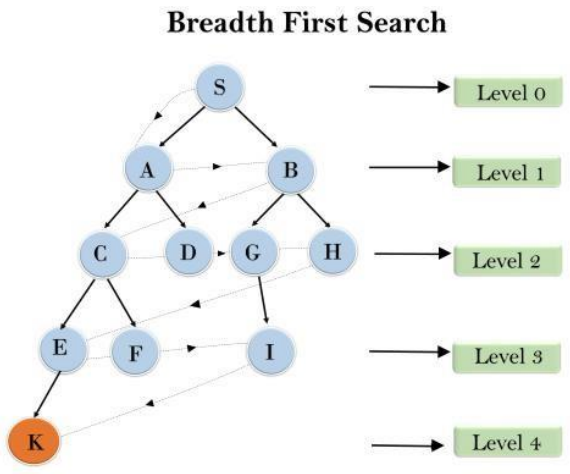
Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

Example:

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

1. S---> A--->B---->C--->D---->G--->H--->E---->F---->I --- >K



Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$T (b) = 1+b^2+b^3+ \ldots\ldots + b^d= O (b^d)$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Python Code:

```
graph = {

  'HOD - Dr. Srikar' : ['ANE','MECH'],

'ANE' : ['II YR', 'III YR'],

'MECH' : ['2 YR'],

  'II YR' : ['Dr. MOHAMMED'],

  'III YR' : ['Mrs. SUSHMA'],

  '2 YR' : ['Mr. GOPAL'],

  'Dr. MOHAMMED':[],

  'Mr. GOPAL':["AIML"],

  'Mrs. SUSHMA':["AIML"],

  'AIML':[]




}
```

```
visited = [] # List to keep track of visited nodes.

queue = []     #Initialize a queue


defbfs(visited, graph, node):

visited.append(node)

queue.append(node)


while queue:

    s = queue.pop(0)

print (s, end = " ")


forneighbour in graph[s]:

ifneighbour not in visited:

visited.append(neighbour)

queue.append(neighbour)


# Driver Code

bfs(visited, graph, 'HOD - Dr. Srikar')
```

```
from collections import deque


defbfs(graph, start):

visited = set()

queue = deque([start])

while queue:

vertex = queue.popleft()
```

```
if vertex not in visited:

visited.add(vertex)

print(vertex)

queue.extend(graph[vertex] - visited)


# Example usage

graph = {

    'A': set(['B', 'C']),

    'B': set(['A', 'D', 'E']),

    'C': set(['A', 'F']),

    'D': set(['B']),

    'E': set(['B', 'F']),

    'F': set(['C', 'E'])

}


bfs(graph, 'A')
```

2. Depth-first Search

- Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

Note: Backtracking is an algorithm technique for finding all possible solutions using recursion.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantage:

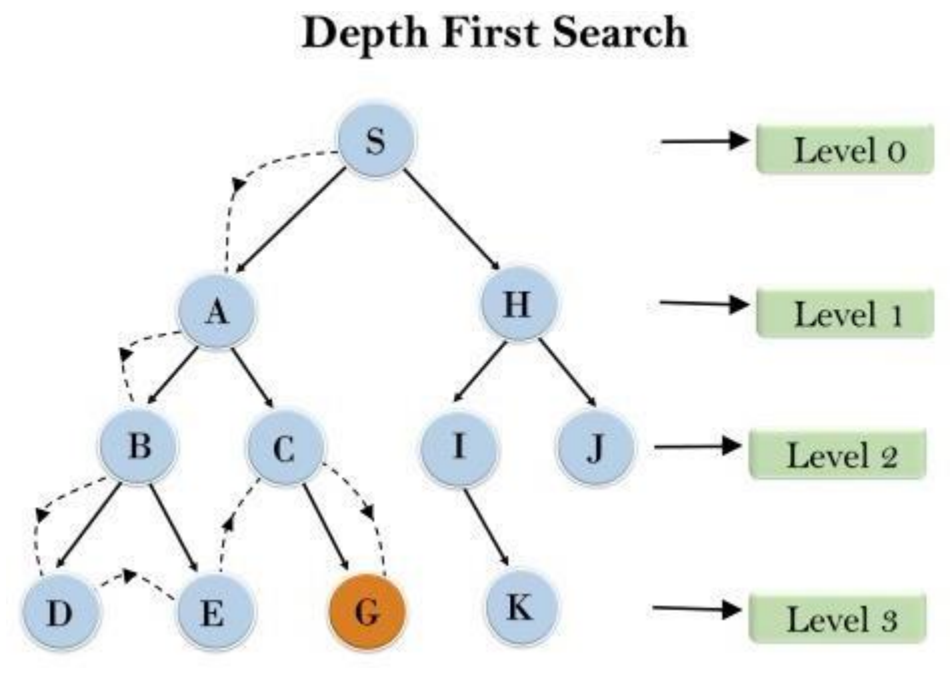
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Example:

In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:

Root node--->Left node---- > right node.

It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.



Completeness: DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

$T(n)= 1+ n^2+ n^3 + ... + n^m=O(n^m)$

Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

Depth-Limited Search Algorithm:

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

Depth-limited search can be terminated with two Conditions of failure:

- Standard failure value: It indicates that problem does not have any solution.
- Cutoff failure value: It defines no solution for the problem within a given depth limit.

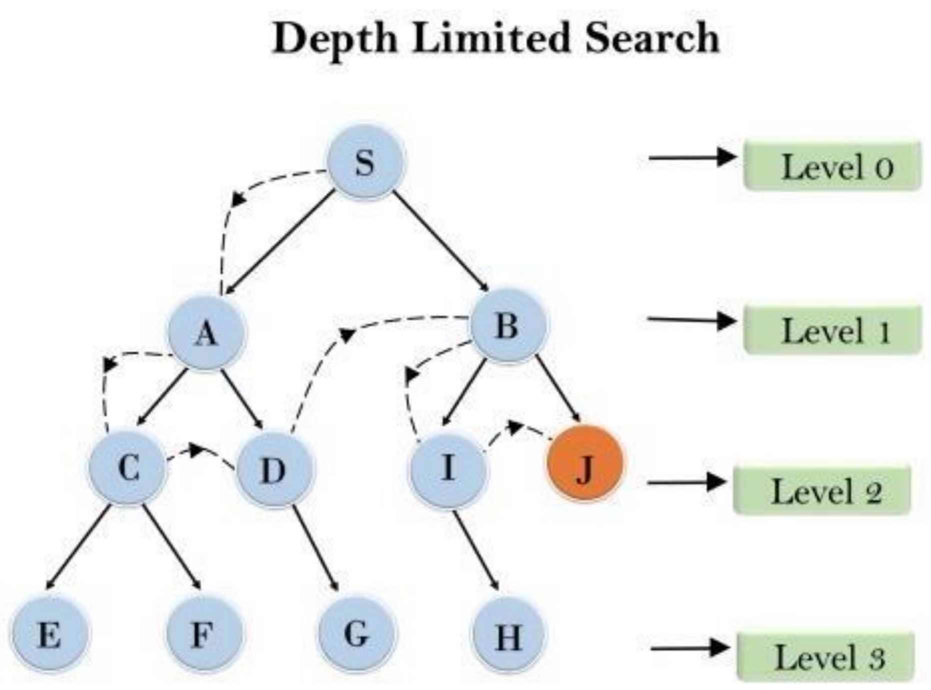
Advantages:

Depth-limited search is Memory efficient.

Disadvantages:

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

Example:



Completeness: DLS search algorithm is complete if the solution is above the depth-limit.

Time Complexity: Time complexity of DLS algorithm is **$O(b^{\ell})$**.

Space Complexity: Space complexity of DLS algorithm is O**($b \times \ell$)**.

Optimal: Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if $\ell > d$.

3. Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

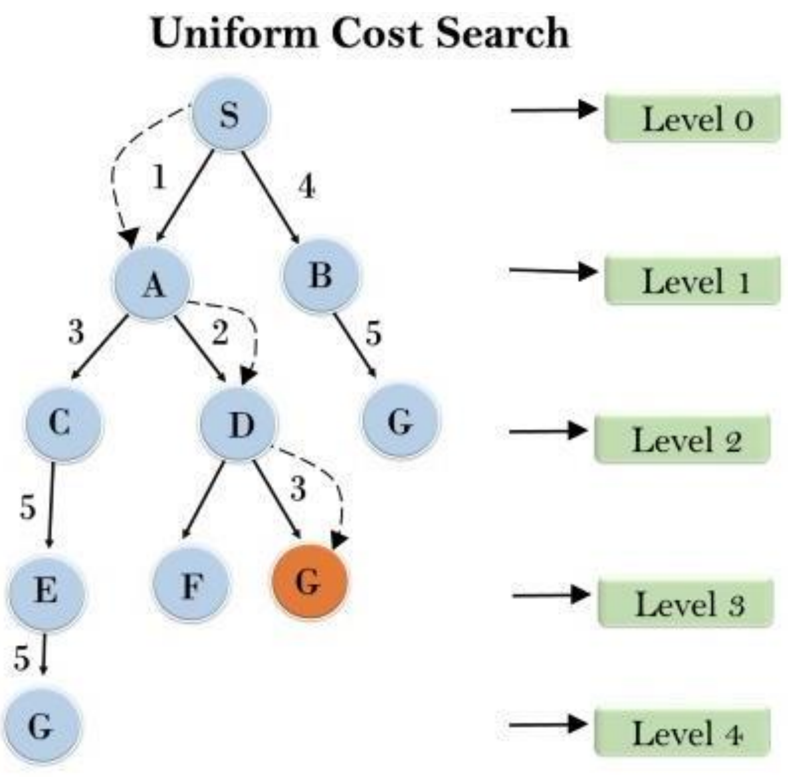
Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

Example:



Completeness:

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

Time Complexity:

Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

Hence, the worst-case time complexity of Uniform-cost search is**$O(b^{1 + [C^*/\varepsilon]})$/**.

Space Complexity:

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is **$O(b^{1 + [C^*/\varepsilon]})$**.

Optimal:

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

4. Iterative deepening depth-first Search:

The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.

This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.

This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.

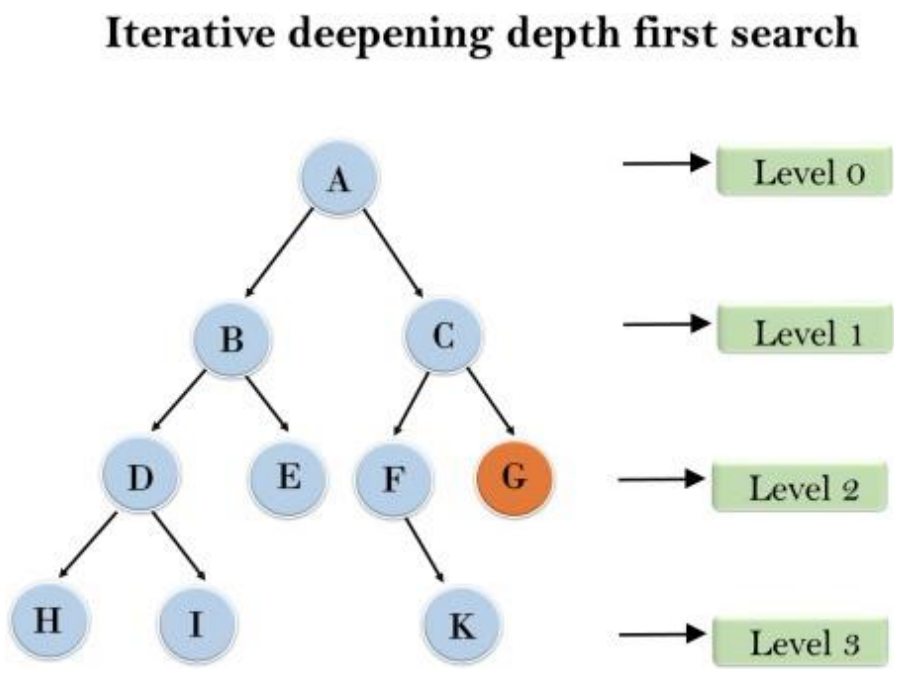
The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

Advantages:

- Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

- The main drawback of IDDFS is that it repeats all the work of the previous phase.



1'stIteration ---- > A
2'nd Iteration----> A, B, C
3'rd Iteration ----- >A, B, D, E, C, F, G
4'th Iteration ----- >A, B, D, H, I, E, C, F, K, G
In the fourth iteration, the algorithm will find the goal node.

Completeness:

This algorithm is complete is ifthe branching factor is finite.

Time Complexity:

Let's suppose b is the branching factor and depth is d then the worst-case time complexity is **O(b^d)**.

Space Complexity:

The space complexity of IDDFS will be **O(bd)**.

Optimal:

IDDFS algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

Hill Climbing Algorithm in Artificial Intelligence

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features of Hill Climbing:

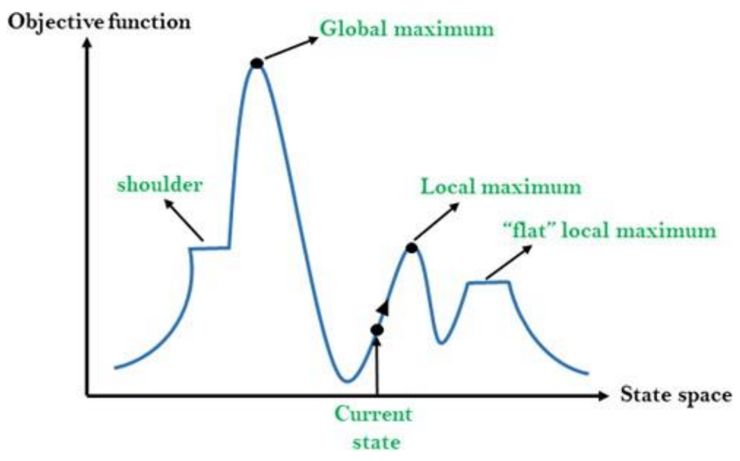
Following are some main features of Hill Climbing Algorithm:

- **Generate and Test variant:** Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
- **Greedy approach:** Hill-climbing algorithm search moves in the direction which optimizes the cost.
- **No backtracking:** It does not backtrack the search space, as it does not remember the previous states.

State-space Diagram for Hill Climbing:

The state-space landscape is a graphical representation of the hill-climbing algorithm which is showing a graph between various states of algorithm and Objective function/Cost.

On Y-axis we have taken the function which can be an objective function or cost function, and state-space on the x-axis. If the function on Y-axis is cost then, the goal of search is to find the global minimum and local minimum. If the function of Y-axis is Objective function, then the goal of the search is to find the global maximum and local maximum.



Different regions in the state space landscape:

Local Maximum: Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.

Global Maximum: Global maximum is the best possible state of state space landscape. It has the highest value of objective function.

Current state: It is a state in a landscape diagram where an agent is currently present.

Flat local maximum: It is a flat space in the landscape where all the neighbor states of current states have the same value.

Shoulder: It is a plateau region which has an uphill edge.

Types of Hill Climbing Algorithm:

- Simple hill Climbing:
- Steepest-Ascent hill-climbing:

- Stochastic hill Climbing:

1. Simple Hill Climbing:

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state. This algorithm has the following features:

- Less time consuming
- Less optimal solution and the solution is not guaranteed

Algorithm for Simple Hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
    a. If it is goal state, then return success and quit.
    b. Else if it is better than the current state then assign new state as a current state.
    c. Else if not better than the current state, then return to step2.

Step 5: Exit.

2. Steepest-Ascent hill climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

Algorithm for Steepest-Ascent hill climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
- **Step 2:** Loop until a solution is found or the current state does not change.

   a. Let SUCC be a state such that any successor of the current state will be better than it.
   b. For each operator that applies to the current state:
      a. Apply the new operator and generate a new state.
      b. Evaluate the new state.
      c. If it is goal state, then return it and quit, else compare it to the SUCC.
      d. If it is better than SUCC, then set new state as SUCC.
      e. If the SUCC is better than the current state, then set current state to SUCC.

Step 5: Exit.

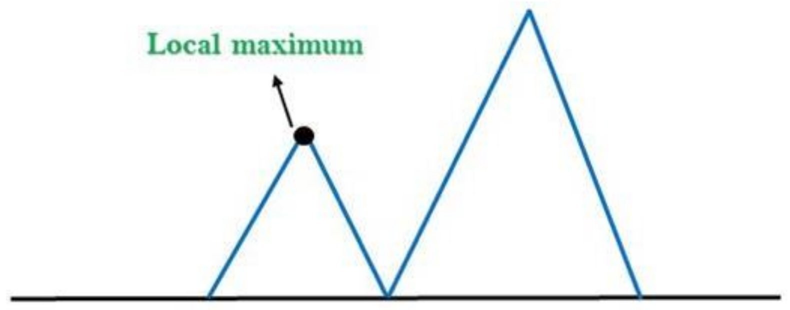
3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

Problems in Hill Climbing Algorithm:

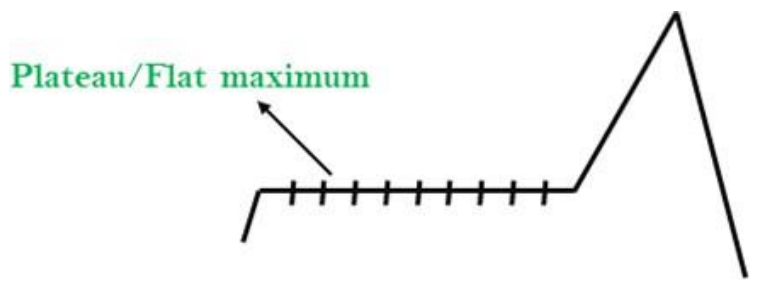
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



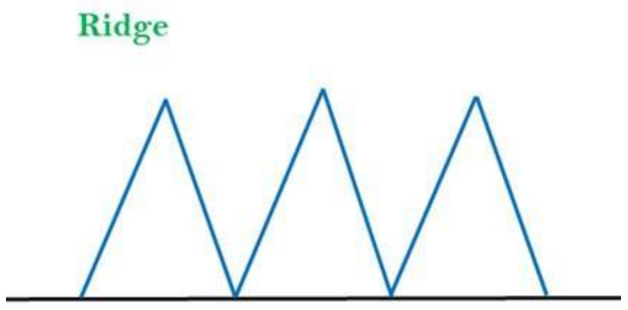
2. Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



3. Ridges: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Solution: With the use of bidirectional search, or by moving in different directions, we can improve this problem.



Simulated Annealing:

A hill-climbing algorithm which never makes a move towards a lower value guaranteed to be incomplete because it can get stuck on a local maximum. And if algorithm applies a random walk, by moving a successor, then it may complete but not efficient. **Simulated Annealing** is an algorithm which yields both efficiency and completeness.

In mechanical term **Annealing** is a process of hardening a metal or glass to a high temperature then cooling gradually, so this allows the metal to reach a low-energy crystalline state. The same process is used in simulated annealing in which the algorithm picks a random move, instead of picking the best move. If the random move improves the state, then it follows the same path. Otherwise, the algorithm follows the path which has a probability of less than 1 or it moves downhill and chooses another path.

Informed Search Algorithms

So far we have talked about the uninformed search algorithms which looked through search space for all possible solutions of the problem without having any additional knowledge about search space. But informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the search space and find more efficiently the goal node.

The informed search algorithm is more useful for large search space. Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.

Heuristics function: Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Admissibility of the heuristic function is given as:

1. $h(n) <= h^*(n)$

Here h(n) is heuristic cost, and h*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

Pure Heuristic Search:

Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

In the informed search we will discuss two main algorithms which are given below:

- **Best First Search Algorithm(Greedy search)**
- **A* Search Algorithm**

1.) Best-first Search Algorithm (Greedy Search):

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e. f(n)= g(n).

Were, h(n)= estimated cost from node n to the goal.

The greedy best first algorithm is implemented by the priority queue.

Best first search algorithm:

- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Advantages:

- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.

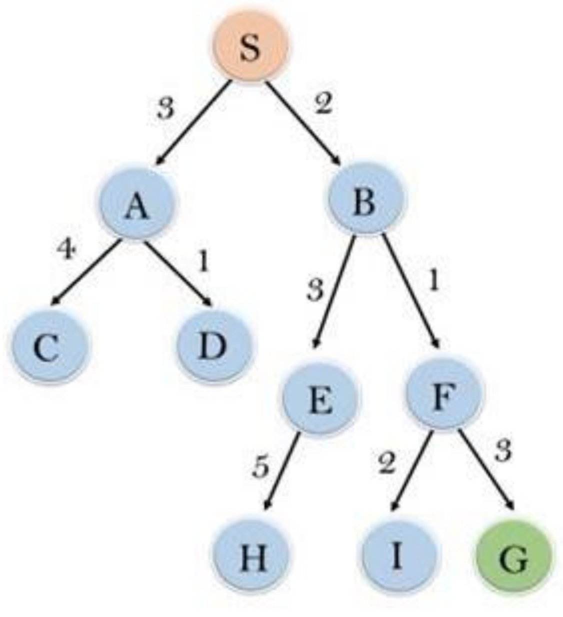
- This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

- It can behave as an unguided depth-first search in the worst case scenario.
- It can get stuck in a loop as DFS.
- This algorithm is not optimal.

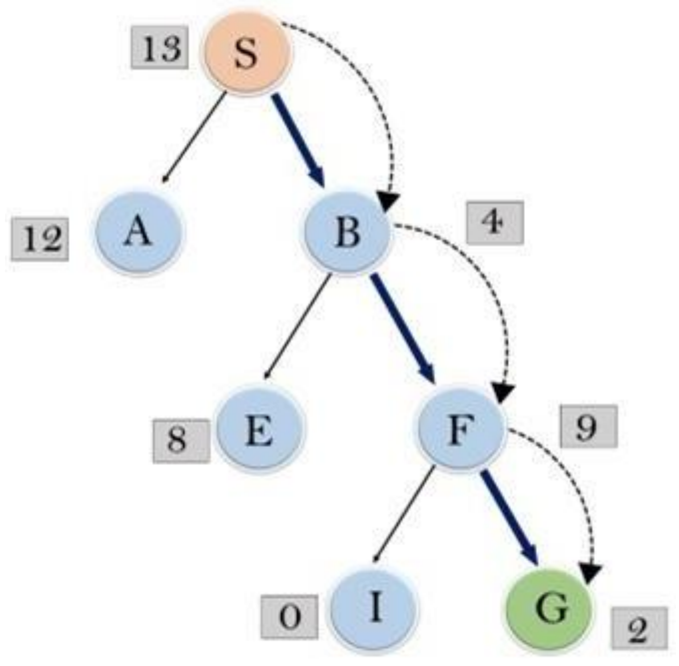
Example:

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F --- > G**

Time Complexity: The worst case time complexity of Greedy best first search is O(b^m).

Space Complexity: The worst case space complexity of Greedy best first search is O(b^m). Where, m is the maximum depth of the search space.

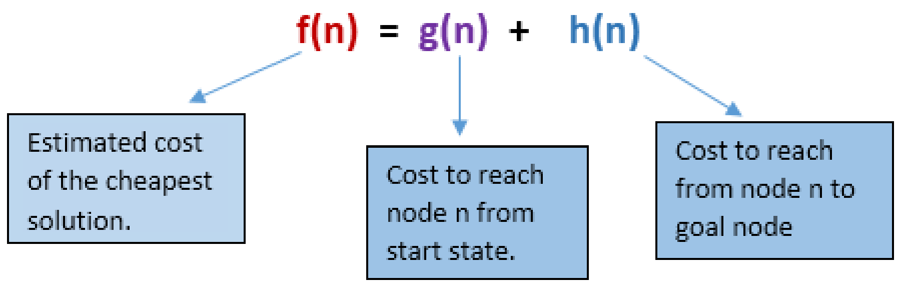
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A* Search Algorithm:

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.


f(n) = g(n) + h(n)
Estimated cost of the cheapest solution.
Cost to reach node n from start state.
Cost to reach from node n to goal node

At each point in the search space, only those node is expanded which have the lowest value of f(n), and the algorithm terminates when the goal node is found.

Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to **Step 2**.

Advantages:

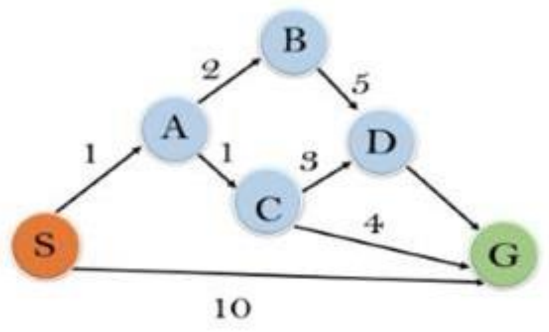
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

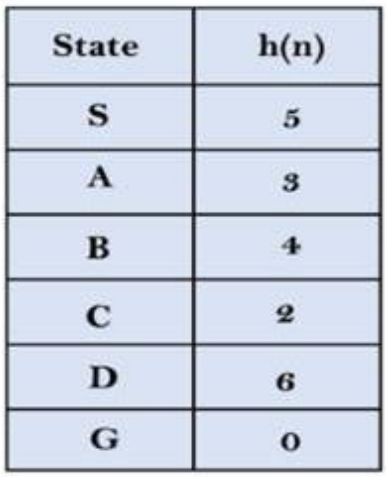
Disadvantages:

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Example:
In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.

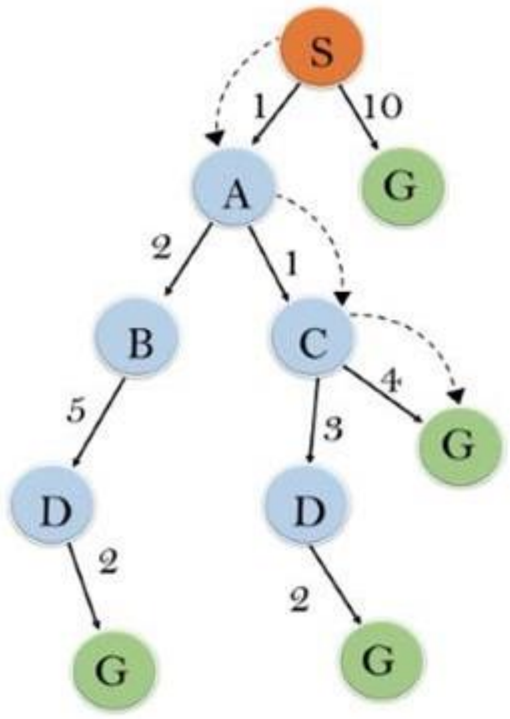
Here we will use OPEN and CLOSED list.





State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution:



Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

Points to remember:

- A* algorithm returns the path which occurred first, and it does not search for all remaining paths.
- The efficiency of A* algorithm depends on the quality of heuristic.
- A* algorithm expands all nodes which satisfy the condition f(n)<="" li="">

Complete: A* algorithm is complete as long as:

- Branching factor is finite.

- Cost at every action is fixed.

Optimal: A* search algorithm is optimal if it follows below two conditions:

- **Admissible:** the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- **Consistency:** Second required condition is consistency for only A* graph-search.

If the heuristic function is admissible, then A* tree search will always find the least cost path.

Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **O(b^d)**

Constraint Satisfaction Problems in Artificial Intelligence

It is noticed that during search we encountered a wide variety of methods, including adversarial search and instant search, to address various issues. Every method for issue has a single purpose in mind: to locate a remedy that will enable that achievement of the objective. However there were no restrictions just on bots' capability to resolve issues as well as arrive at responses in adversarial search and local search, respectively.

This examines the constraint optimization methodology, another form or real concern method. By its name, constraints fulfillment implies that such an issue must be solved while adhering to a set of restrictions or guidelines.

Whenever a problem is actually variables comply with stringent conditions of principles, it is said to have been addressed using the solving multi - objective method. Wow what a method results in a study sought to achieve of the intricacy and organization of both the issue.

Three factors affect restriction compliance, particularly regarding:

- It refers to a group of parameters, or X.
- D: The variables are contained within a collection several domain. Every variables has a distinct scope.
- C: It is a set of restrictions that the collection of parameters must abide by.

In constraint satisfaction, domains are the areas wherein parameters were located after the restrictions that are particular to the task. Those three components make up a constraint satisfaction technique in its entirety. The pair "scope, rel" makes up the number of something like the requirement. The scope is a tuple of variables that contribute to the restriction, as well as rel is indeed a relationship that contains a list of possible solutions for the parameters should assume in order to meet the restrictions of something like the issue.

Issues with Contains A certain amount Solved

For a constraint satisfaction problem (CSP), the following conditions must be met:

- States area
- fundamental idea while behind remedy.

The definition of a state in phase space involves giving values to any or all of the parameters, like as

X1 = v1, X2 = v2, etc.

There are 3 methods to economically beneficial to something like a parameter:

1. Consistent or Legal Assignment: A task is referred to as consistent or legal if it complies with all laws and regulations.
2. Complete Assignment: An assignment in which each variable has a number associated to it and that the CSP solution is continuous. One such task is referred to as a completed task.
3. A partial assignment is one that just gives some of the variables values. Projects of this nature are referred to as incomplete assignment.

Domain Categories within CSP

The parameters utilize one of the two types of domains listed below:

- Discrete Domain: This limitless area allows for the existence of a single state with numerous variables. For instance, every parameter may receive a endless number of beginning states.
- It is a finite domain with continous phases that really can describe just one area for just one particular variable. Another name for it is constant area.

Types of Constraints in CSP

Basically, there are three different categories of limitations in regard towards the parameters:

- Unary restrictions are the easiest kind of restrictions because they only limit the value of one variable.
- Binary resource limits: These restrictions connect two parameters. A value between x1 and x3 can be found in a variable named x2.
- Global Resource limits: This kind of restriction includes a unrestricted amount of variables.

The main kinds of restrictions are resolved using certain kinds of resolution methodologies:

- In linear programming, when every parameter carrying an integer value only occurs in linear equation, linear constraints are frequently utilised.
- Non-linear Constraints: With non-linear programming, when each variable (an integer value) exists in a non-linear form, several types of restrictions were utilised.

Think of a Sudoku puzzle where some of the squares have initial fills of certain integers.

You must complete the empty squares with numbers between 1 and 9, making sure that no rows, columns, or blocks contains a recurring integer of any kind. This solving multi - objective issue is pretty elementary. A problem must be solved while taking certain limitations into consideration.

The integer range (1-9) that really can occupy the other spaces is referred to as a domain, while the empty spaces themselves were referred as variables. The values of the variables are drawn first from realm. Constraints are the rules that determine how a variable will select the scope.

UNIT - II: Advanced Search:

- Constructing Search Trees
- Stochastic Search
- AO* Search Implementation
- Minimax Search
- Alpha-Beta Pruning Basic Knowledge Representation and Reasoning: Propositional Logic
- First-Order Logic
- Forward Chaining and Backward Chaining
- Introduction to Probabilistic Reasoning
- Bayes Theorem

Constructing trees

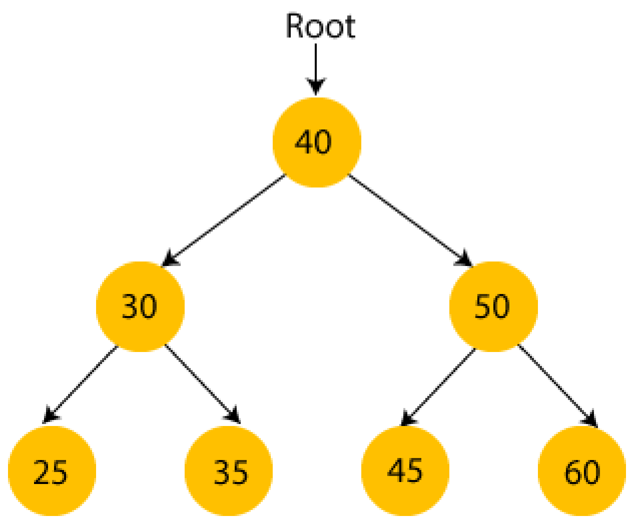
A tree is a kind of data structure that is used to represent the data in hierarchical form. It can be defined as a collection of objects or entities called as nodes that are linked together to simulate a hierarchy. Tree is a non-linear data structure as the data in a tree is not stored linearly or sequentially.

Now, let's start the topic, the Binary Search tree.

What is a Binary Search tree?

A binary search tree follows some order to arrange the elements. In a Binary search tree, the value of left node must be smaller than the parent node, and the value of right node must be greater than the parent node. This rule is applied recursively to the left and right subtrees of the root.

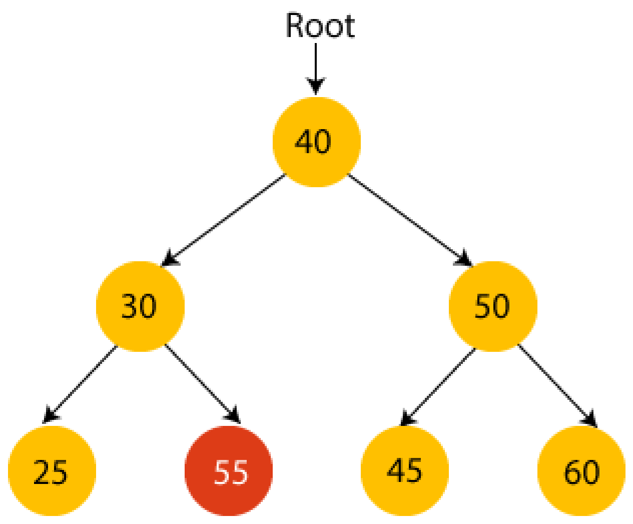
Let's understand the concept of Binary search tree with an example.



In the above figure, we can observe that the root node is 40, and all the nodes of the left subtree are smaller than the root node, and all the nodes of the right subtree are greater than the root node.

Similarly, we can see the left child of root node is greater than its left child and smaller than its right child. So, it also satisfies the property of binary search tree. Therefore, we can say that the tree in the above image is a binary search tree.

Suppose if we change the value of node 35 to 55 in the above tree, check whether the tree will be binary search tree or not.



In the above tree, the value of root node is 40, which is greater than its left child 30 but smaller than right child of 30, i.e., 55. So, the above tree does not satisfy the property of Binary search tree. Therefore, the above tree is not a binary search tree.

Advantages of Binary search tree

- Searching an element in the Binary search tree is easy as we always have a hint that which subtree has the desired element.
- As compared to array and linked lists, insertion and deletion operations are faster in BST.

Example of creating a binary search tree

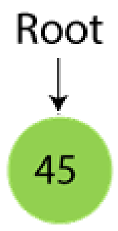
Now, let's see the creation of binary search tree using an example.

Suppose the data elements are **- 45, 15, 79, 90, 10, 55, 12, 20, 50**

- First, we have to insert **45** into the tree as the root of the tree.
- Then, read the next element; if it is smaller than the root node, insert it as the root of the left subtree, and move to the next element.
- Otherwise, if the element is larger than the root node, then insert it as the root of the right subtree.

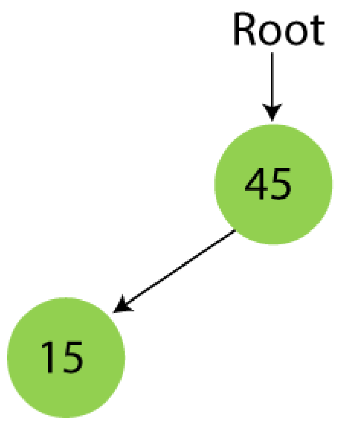
Now, let's see the process of creating the Binary search tree using the given data element. The process of creating the BST is shown below -

Step 1 - Insert 45.



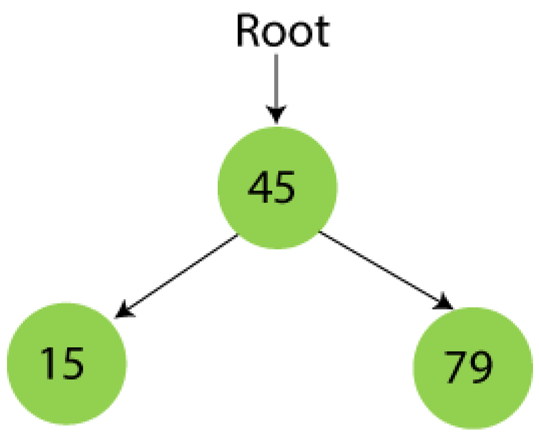
Step 2 - Insert 15.

As 15 is smaller than 45, so insert it as the root node of the left subtree.



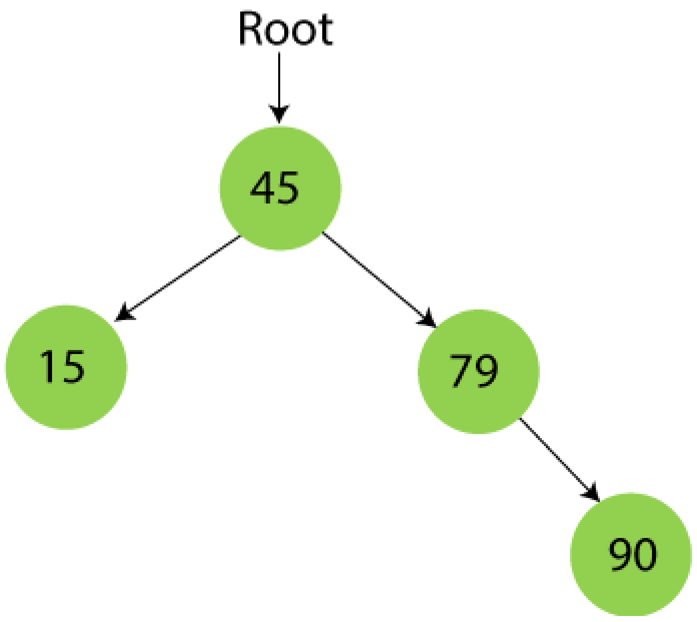
Step 3 - Insert 79.

As 79 is greater than 45, so insert it as the root node of the right subtree.



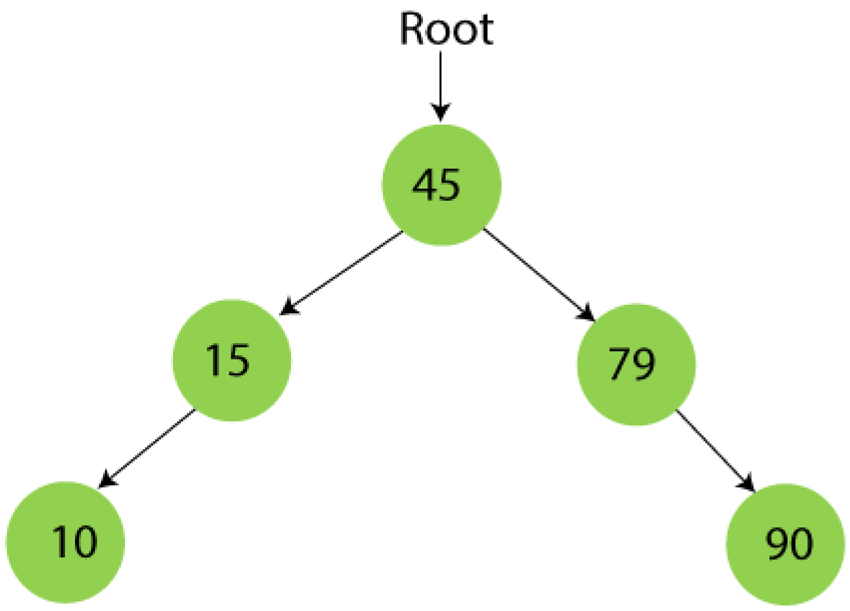
Step 4 - Insert 90.

90 is greater than 45 and 79, so it will be inserted as the right subtree of 79.



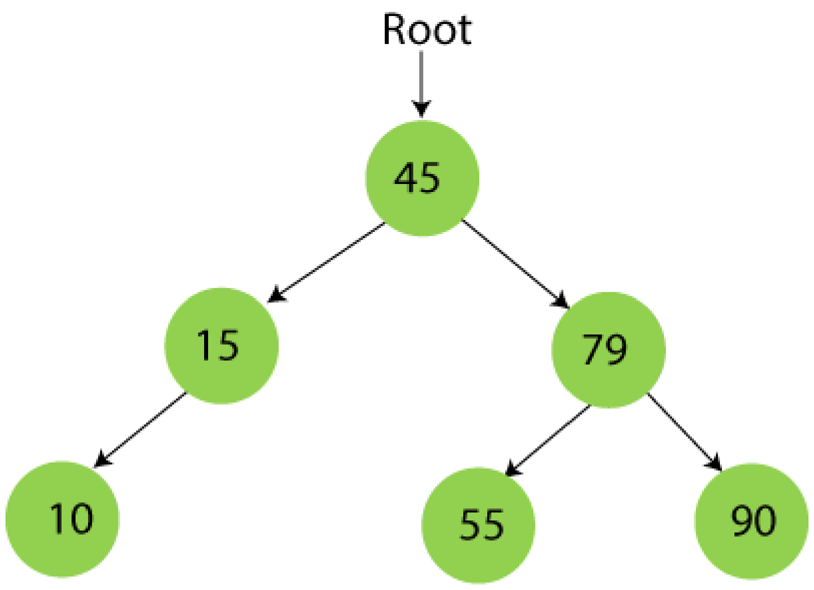
Step 5 - Insert 10.

10 is smaller than 45 and 15, so it will be inserted as a left subtree of 15.



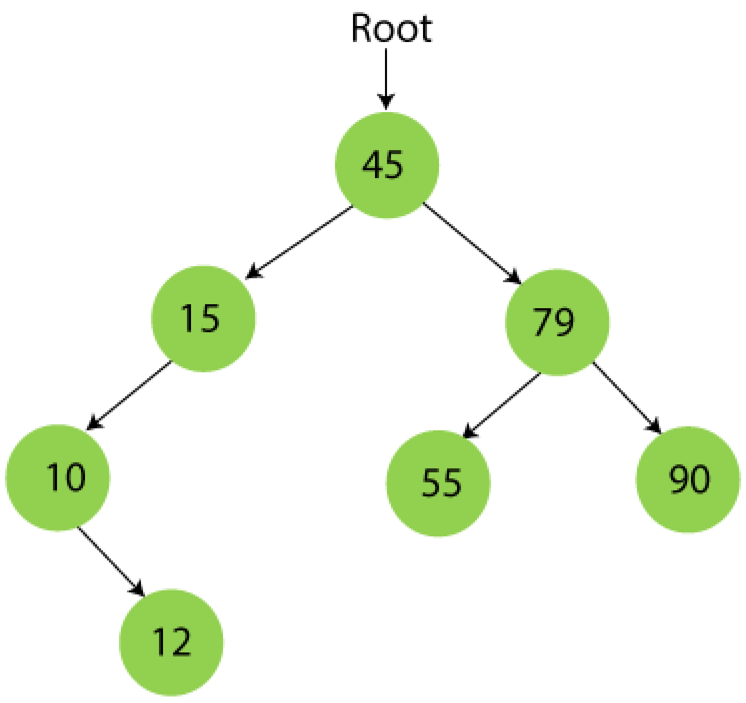
Step 6 - Insert 55.

55 is larger than 45 and smaller than 79, so it will be inserted as the left subtree of 79.



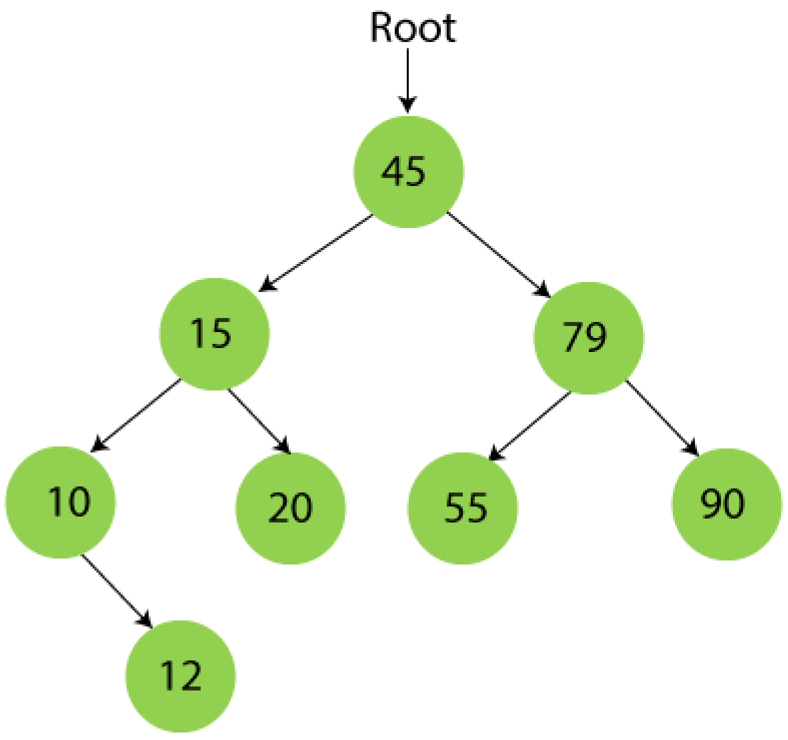
Step 7 - Insert 12.

12 is smaller than 45 and 15 but greater than 10, so it will be inserted as the right subtree of 10.



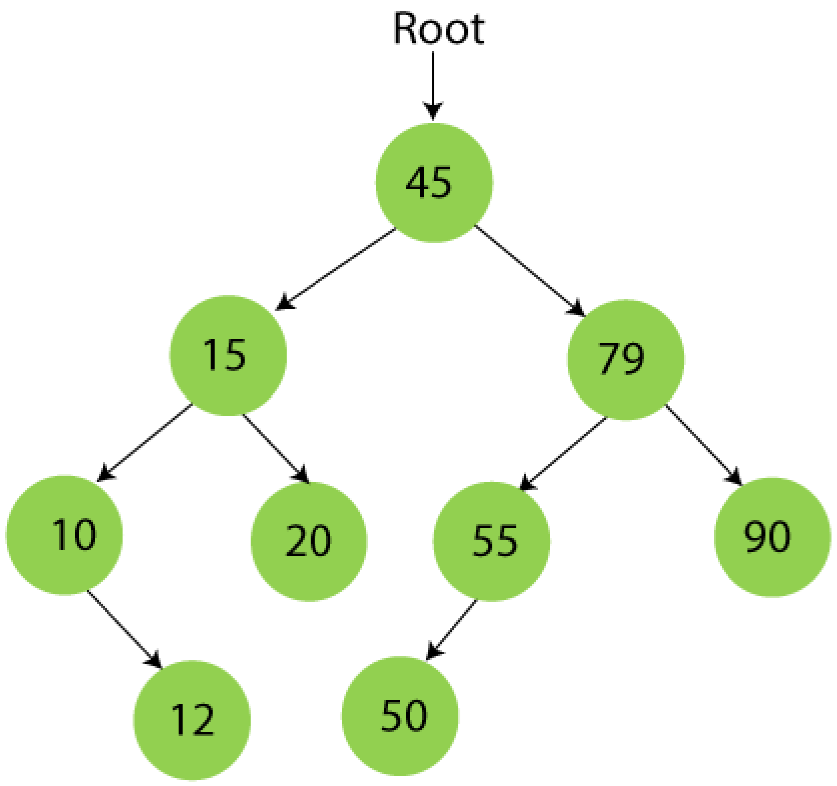
Step 8 - Insert 20.

20 is smaller than 45 but greater than 15, so it will be inserted as the right subtree of 15.



Step 9 - Insert 50.

50 is greater than 45 but smaller than 79 and 55. So, it will be inserted as a left subtree of 55.



Now, the creation of binary search tree is completed. After that, let's move towards the operations that can be performed on Binary search tree.

We can perform insert, delete and search operations on the binary search tree.

Let's understand how a search is performed on a binary search tree.

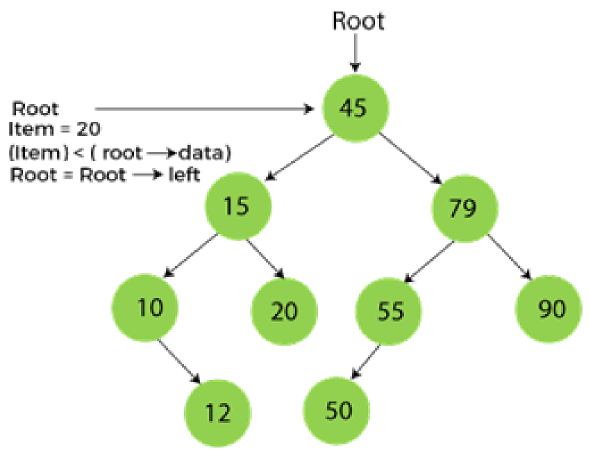
Searching in Binary search tree

Searching means to find or locate a specific element or node in a data structure. In Binary search tree, searching a node is easy because elements in BST are stored in a specific order. The steps of searching a node in Binary Search tree are listed as follows -

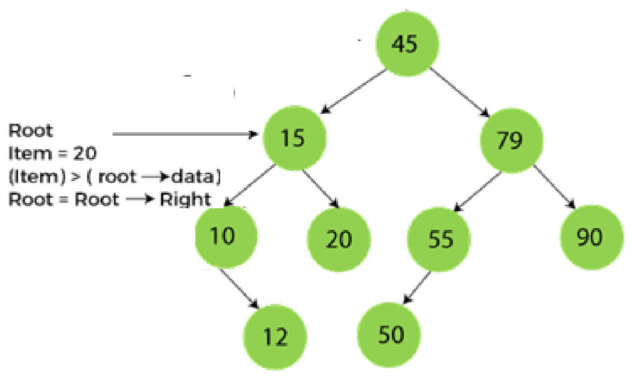
1. First, compare the element to be searched with the root element of the tree.
2. If root is matched with the target element, then return the node's location.
3. If it is not matched, then check whether the item is less than the root element, if it is smaller than the root element, then move to the left subtree.
4. If it is larger than the root element, then move to the right subtree.
5. Repeat the above procedure recursively until the match is found.
6. If the element is not found or not present in the tree, then return NULL.

Now, let's understand the searching in binary tree using an example. We are taking the binary search tree formed above. Suppose we have to find node 20 from the below tree.

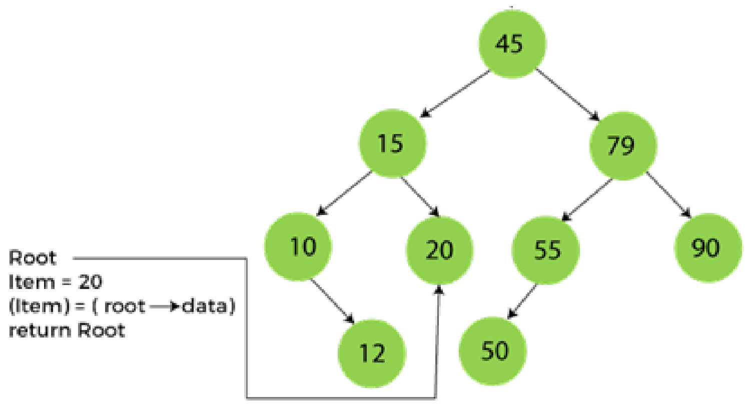
Step1:



Step2:



Step3:



Now, let's see the algorithm to search an element in the Binary search tree.

Algorithm to search an element in Binary search tree

```
1. Search (root, item)
2. Step 1 - if (item = root → data) or (root = NULL)
3. return root
4. else if (item < root → data)
5. return Search(root → left, item)
6. else
7. return Search(root → right, item)
8. END if
9. Step 2 - END
```

Now let's understand how the deletion is performed on a binary search tree. We will also see an example to delete an element from the given tree.

Deletion in Binary Search tree

In a binary search tree, we must delete a node from the tree by keeping in mind that the property of BST is not violated. To delete a node from BST, there are three possible situations occur -

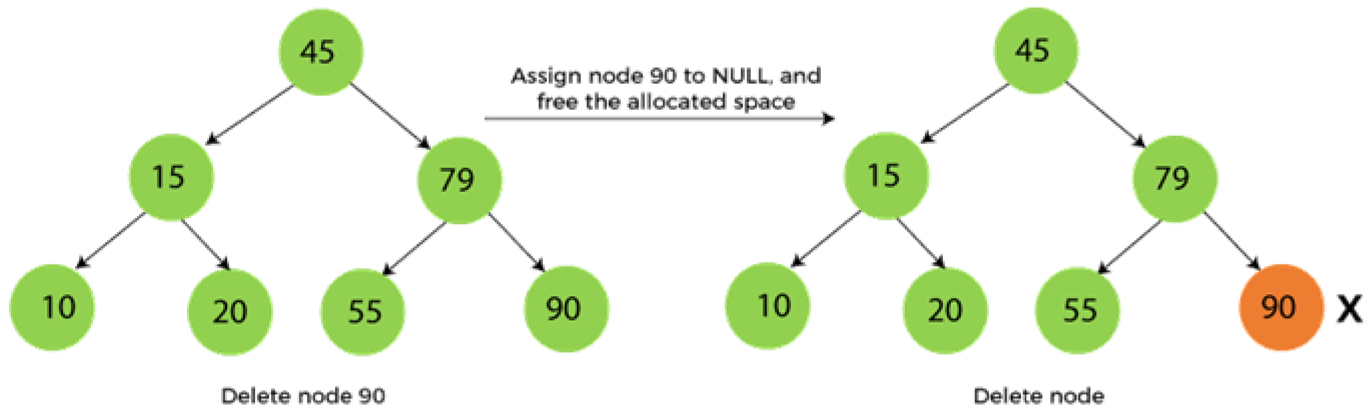
- The node to be deleted is the leaf node, or,
- The node to be deleted has only one child, and,
- The node to be deleted has two children

We will understand the situations listed above in detail.

When the node to be deleted is the leaf node

It is the simplest case to delete a node in BST. Here, we have to replace the leaf node with NULL and simply free the allocated space.

We can see the process to delete a leaf node from BST in the below image. In below image, suppose we have to delete node 90, as the node to be deleted is a leaf node, so it will be replaced with NULL, and the allocated space will free.

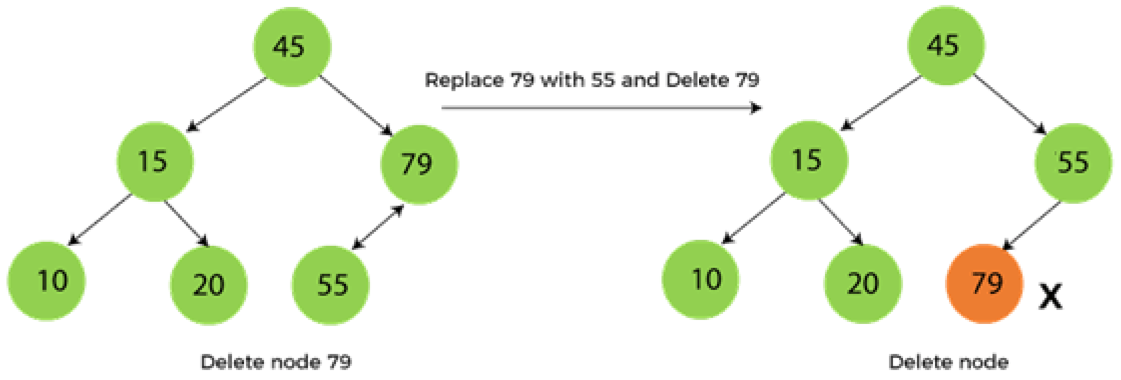


When the node to be deleted has only one child

In this case, we have to replace the target node with its child, and then delete the child node. It means that after replacing the target node with its child node, the child node will now contain the value to be deleted. So, we simply have to replace the child node with NULL and free up the allocated space.

We can see the process of deleting a node with one child from BST in the below image. In the below image, suppose we have to delete the node 79, as the node to be deleted has only one child, so it will be replaced with its child 55.

So, the replaced node 79 will now be a leaf node that can be easily deleted.



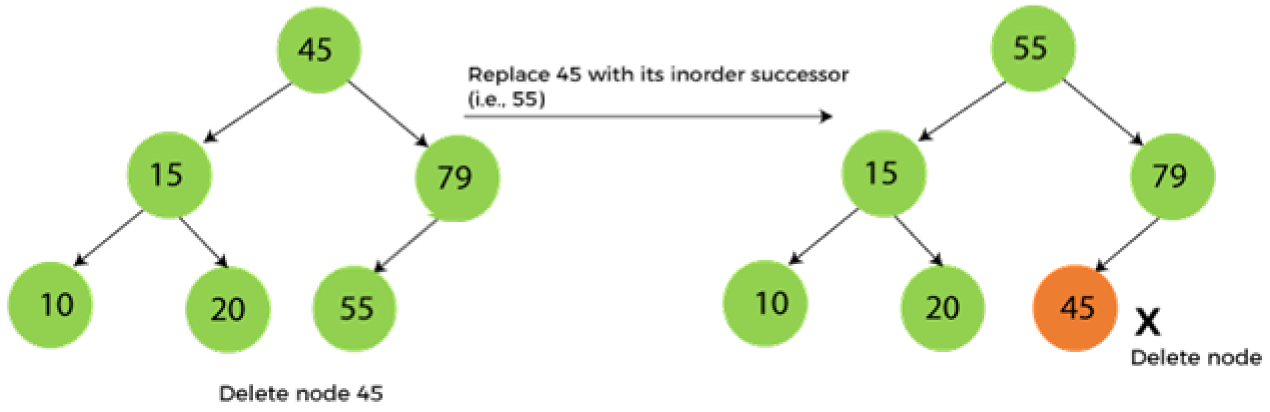
When the node to be deleted has two children

This case of deleting a node in BST is a bit complex among other two cases. In such a case, the steps to be followed are listed as follows -

- First, find the inorder successor of the node to be deleted.
- After that, replace that node with the inorder successor until the target node is placed at the leaf of tree.
- And at last, replace the node with NULL and free up the allocated space.

The inorder successor is required when the right child of the node is not empty. We can obtain the inorder successor by finding the minimum element in the right child of the node.

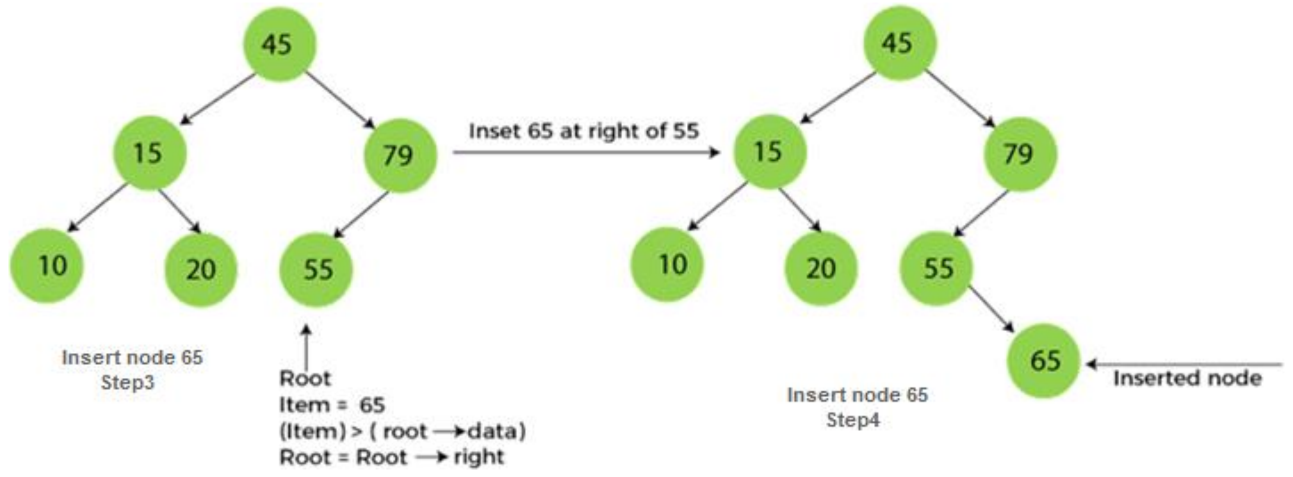
We can see the process of deleting a node with two children from BST in the below image. In the below image, suppose we have to delete node 45 that is the root node, as the node to be deleted has two children, so it will be replaced with its inorder successor. Now, node 45 will be at the leaf of the tree so that it can be deleted easily.



Now let's understand how insertion is performed on a binary search tree.

Insertion in Binary Search tree

A new key in BST is always inserted at the leaf. To insert an element in BST, we have to start searching from the root node; if the node to be inserted is less than the root node, then search for an empty location in the left subtree. Else, search for the empty location in the right subtree and insert the data. Insert in BST is similar to searching, as we always have to maintain the rule that the left subtree is smaller than the root, and right subtree is larger than the root.



The complexity of the Binary Search tree

Let's see the time and space complexity of the Binary search tree. We will see the time complexity for insertion, deletion, and searching operations in best case, average case, and worst case.

1. Time Complexity

Operations	Best case time complexity	Average case time complexity	Worst case time complexity
Insertion	O(log n)	O(log n)	O(n)
Deletion	O(log n)	O(log n)	O(n)
Search	O(log n)	O(log n)	O(n)

Where 'n' is the number of nodes in the given tree.

2. Space Complexity

Operations	Space complexity
Insertion	O(n)
Deletion	O(n)
Search	O(n)

- The space complexity of all operations of Binary search tree is O(n).

Mini-Max Algorithm in Artificial Intelligence

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm:

```
1.  function minimax(node, depth, maximizingPlayer) is
2.  if depth ==0 or node is a terminal node then
3.  return static evaluation of node
4.
5.  if MaximizingPlayer then      // for Maximizer Player
6.  maxEva= -infinity
7.   for each child of node do
```

```
8.     eva= minimax(child, depth-1, false)
9.    maxEva= max(maxEva,eva)        //gives Maximum of the values
10. return maxEva
11.
12. else                      // for Minimizer player
13.   minEva= +infinity
14.   for each child of node do
15.   eva= minimax(child, depth-1, true)
16.   minEva= min(minEva, eva)        //gives minimum of the values
17.   return minEva
```

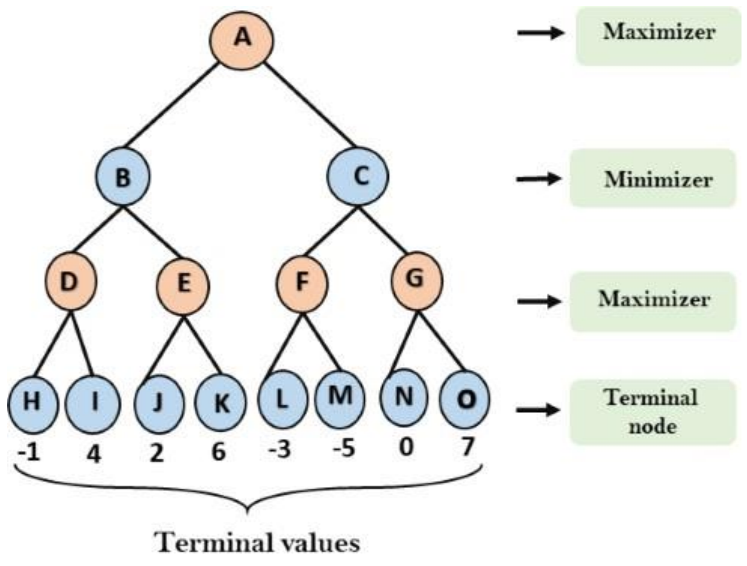
Initial call:

Minimax(node, 3, true)

Working of Min-Max Algorithm:

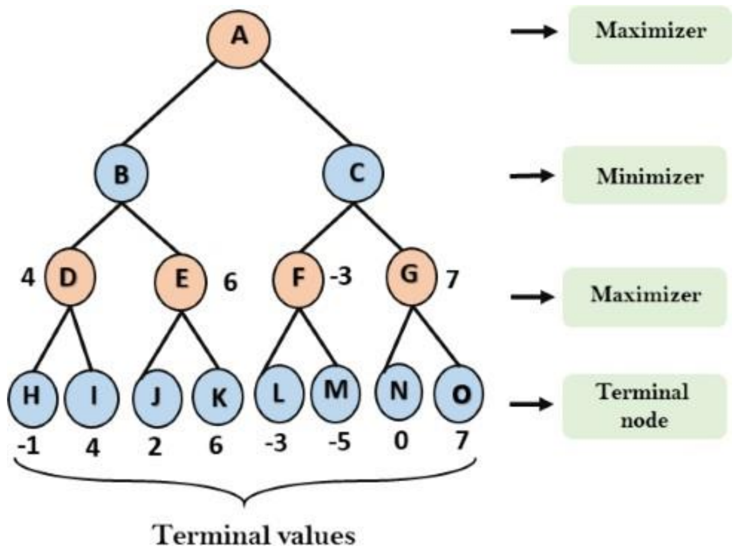
- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



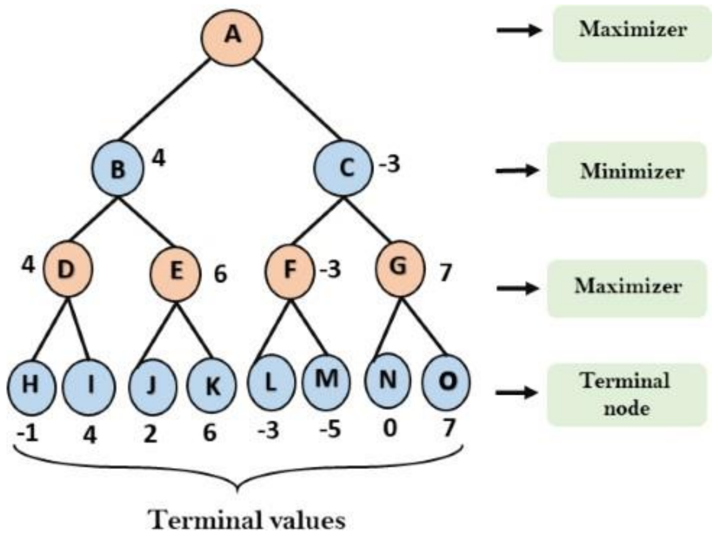
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- For node D max(-1,- -∞) => max(-1,4)= 4
- For Node E max(2, -∞) => max(2, 6)= 6
- For Node F max(-3, -∞) => max(-3,-5) = -3
- For node G max(0, -∞) = max(0, 7) = 7



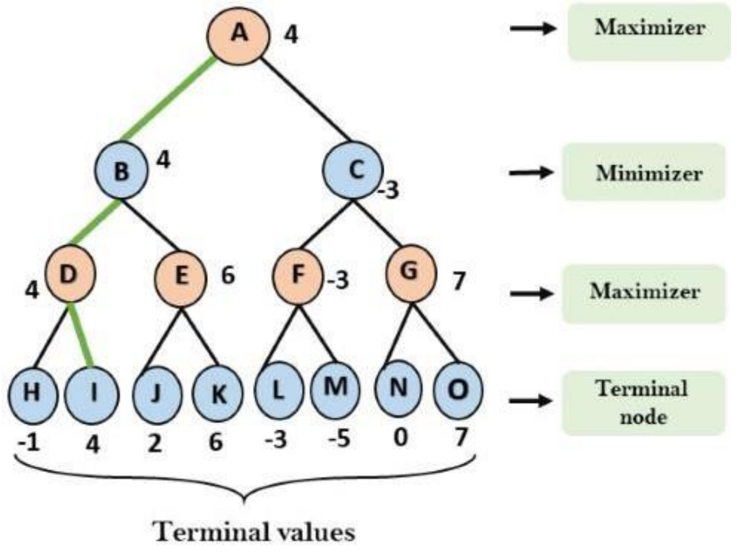
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- For node A max(4, -3)= 4



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning**

Alpha-Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- The two-parameter can be defined as:
    a. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
    b. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Note: To better understand this topic, kindly study the minimax algorithm.

Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1. α>=β

Key points about alpha-beta pruning:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

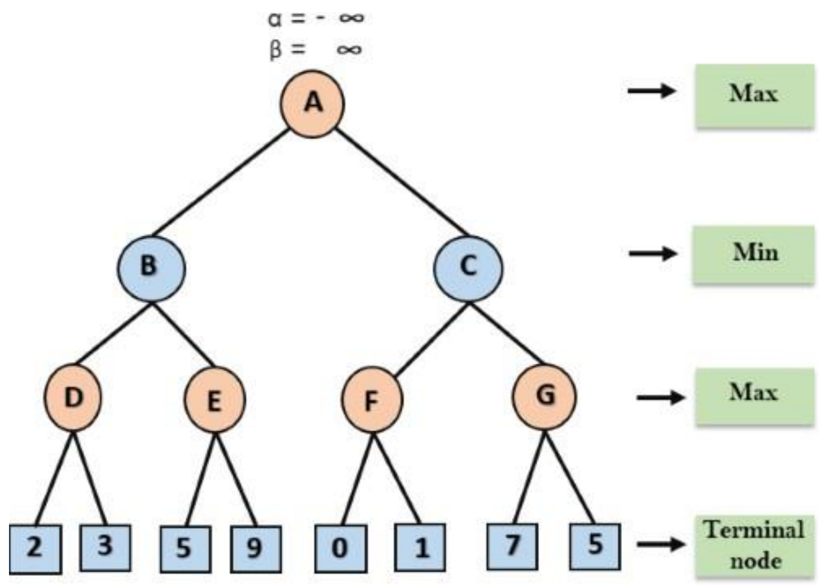
Pseudo-code for Alpha-beta Pruning:

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
if depth ==0 or node is a terminal node then
return static evaluation of node

if MaximizingPlayer then      // for Maximizer Player
   maxEva= -infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, False)
  maxEva= max(maxEva, eva)
  alpha= max(alpha, maxEva)
   if beta<=alpha
 break
 return maxEva

else                        // for Minimizer player
   minEva= +infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, true)
   minEva= min(minEva, eva)
   beta= min(beta, eva)
    if beta<=alpha
  break
 return minEva
```

Working of Alpha-Beta Pruning:

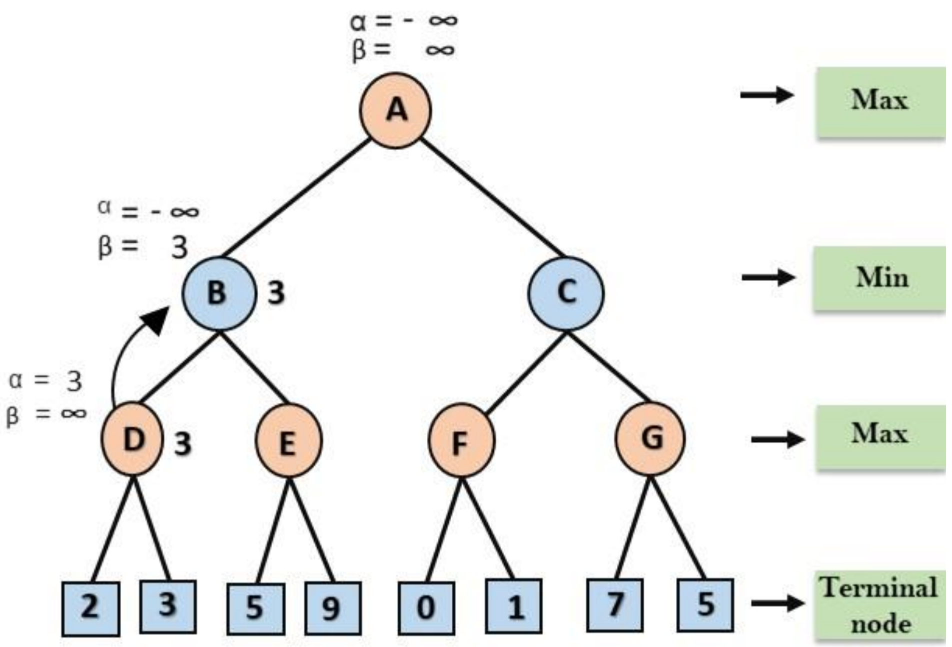
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

Step 1: At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



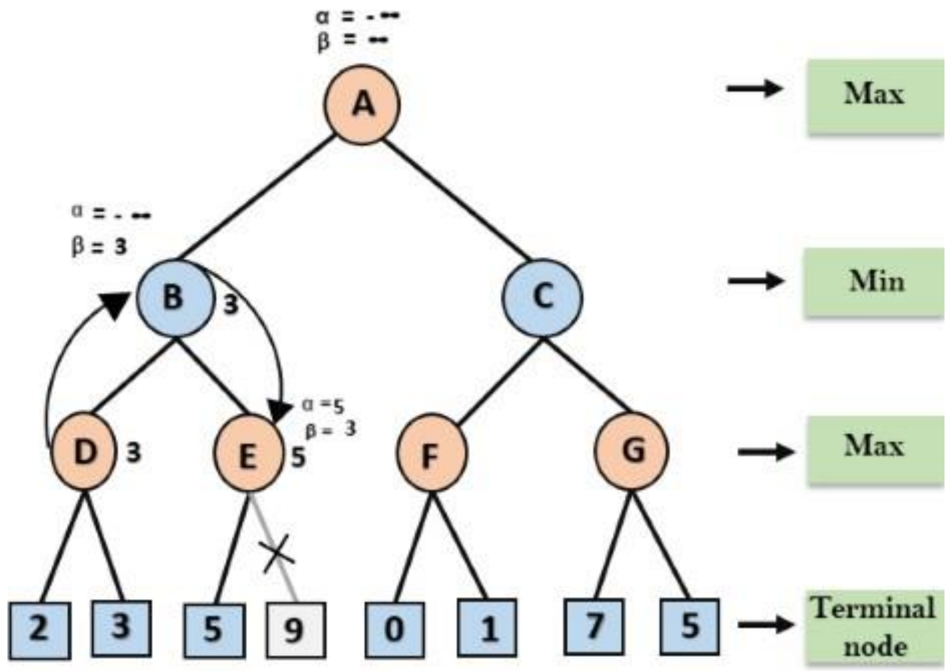
Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.

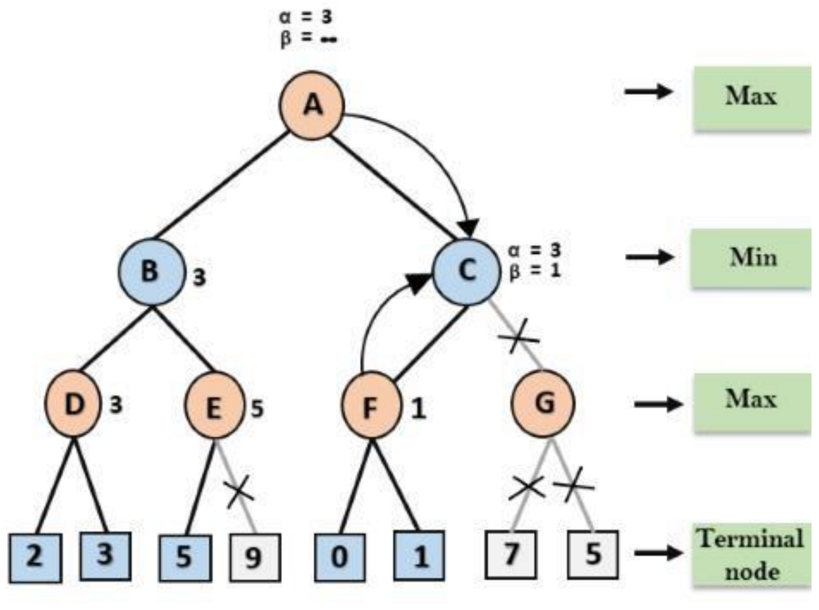


Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

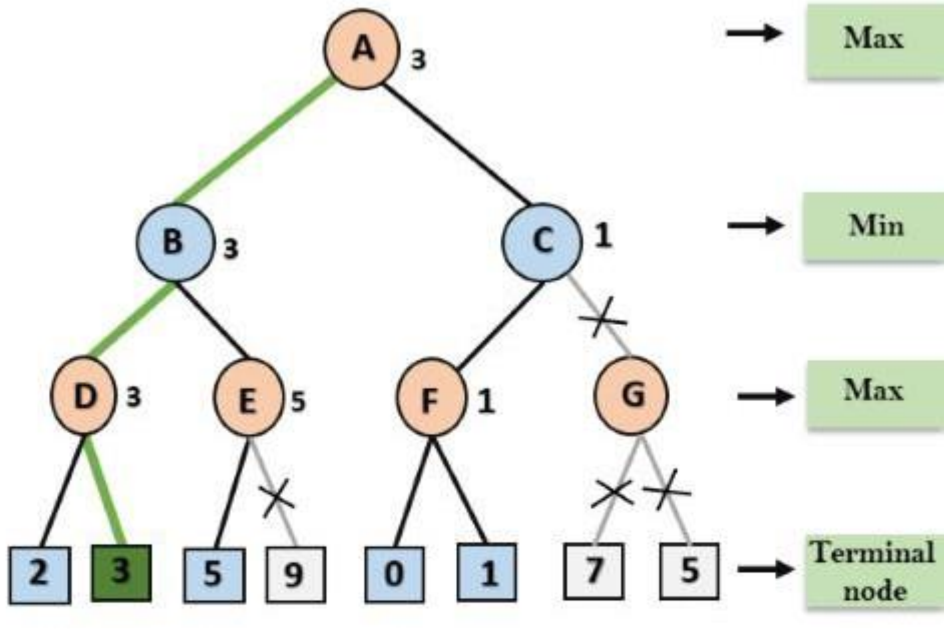
At node C, α=3 and β= +∞, and the same values will be passed on to node F.

Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.

Step 7: Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Move Ordering in Alpha-Beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is $O(b^m)$.
- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is $O(b^{m/2})$.

Rules to find good ordering:

Following are some rules to find good ordering in alpha-beta pruning:

- Occur the best move from the shallowest node.
- Order the nodes in the tree such that the best nodes are checked first.

- Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
- We can bookkeep the states, as there is a possibility that states may repeat.

Forward Chaining and backward chaining in AI

In artificial intelligence, forward and backward chaining is one of the important topics, but before understanding forward and backward chaining lets first understand that from where these two terms came.

Inference engine:

The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:

a. **Forward chaining**

b. **Backward chaining**

Horn Clause and Definite clause:

Horn clause and definite clause are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the **first-order definite clause**.

Definite clause: A clause which is a disjunction of literals with **exactly one positive literal** is known as a definite clause or strict horn clause.

Horn clause: A clause which is a disjunction of literals with **at most one positive literal** is known as horn clause. Hence all the definite clauses are horn clauses.

Example: $(\neg p \lor \neg q \lor k)$. It has only one positive literal k.

It is equivalent to $p \land q \rightarrow k$.

A. Forward Chaining

Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Consider the following famous example which we will use in both approaches:

Example:

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

Prove that **"Robert is criminal."**

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts Conversion into FOL:

- It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

  American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

- Country A has some missiles. **?p Owns(A, p) ∧ Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

  Owns(A, T1)................ (2)

  Missile(T1)................. (3)

- All of the missiles were sold to country A by Robert.

  ?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) .... (4)

- Missiles are weapons.

  Missile(p) → Weapons (p)................... (5)

- Enemy of America is known as hostile.

  Enemy(p, America) →Hostile(p) ....(6)

- Country A is an enemy of America.

  Enemy (A, America) .... (7)

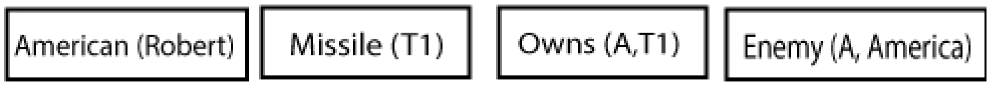
- Robert is American

  American(Robert). .... (8)

Forward chaining proof:

Step-1:

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1)**. All these facts will be represented as below.



Step-2:

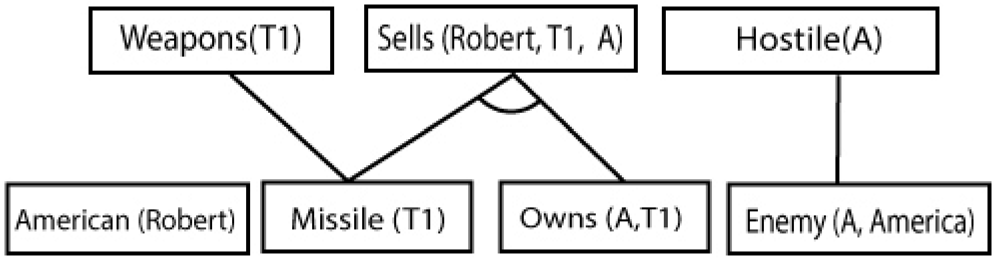
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

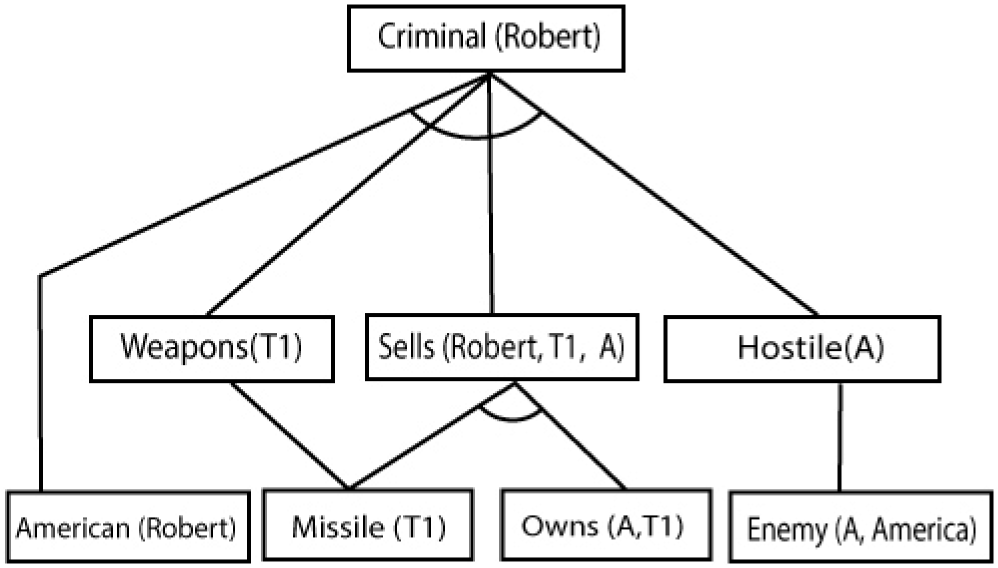
Rule-(4) satisfy with the substitution {p/T1}, **so Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



Step-3:

At step-3, as we can check Rule-(1) is satisfied with the substitution **{p/Robert, q/T1, r/A}, so we can add Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is Criminal using forward chaining approach.

B. Backward Chaining:

Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties of backward chaining:

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- The backward-chaining method mostly used a **depth-first search** strategy for proof.

Example:

In backward-chaining, we will use the same above example, and will rewrite all the rules.

- **American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)**
  Owns(A, T1)......................(2)
- **Missile(T1)**

- **?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)** ....**(4)**
- **Missile(p) → Weapons (p)**.......................**(5)**
- **Enemy(p, America) →Hostile(p)** .... **(6)**
- **Enemy (A, America) .... (7)**
- **American(Robert). .... (8)**

Backward-Chaining proof:

In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.

Step-1:

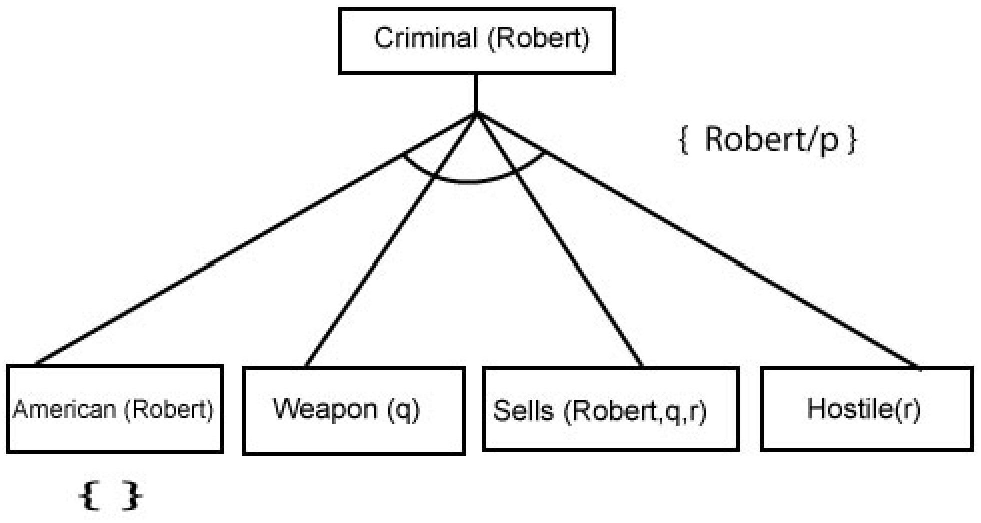
At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

Criminal (Robert)

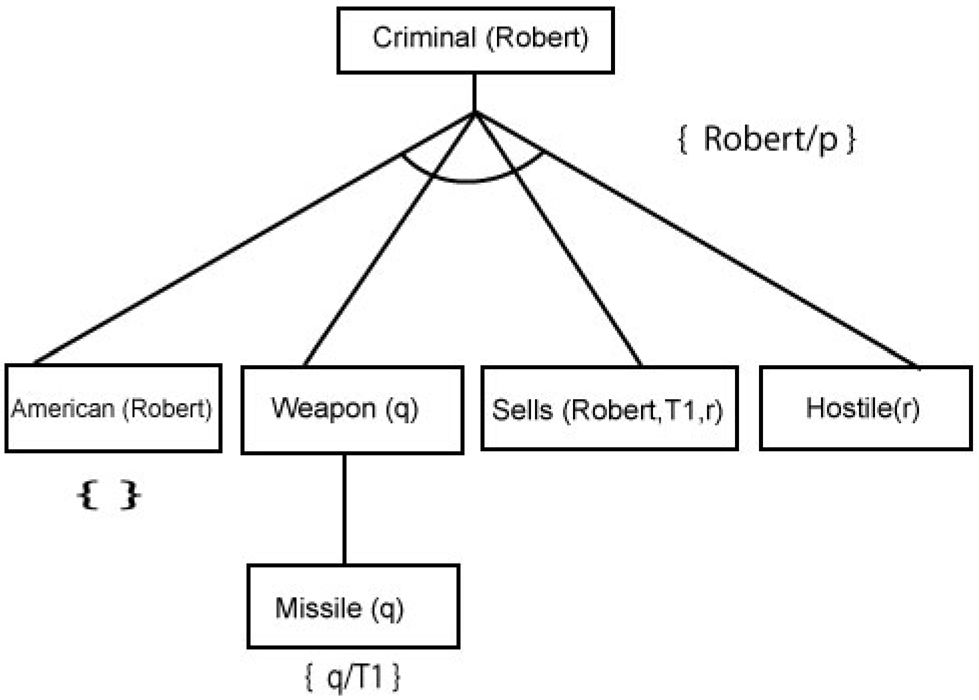
Step-2:

At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.

Here we can see American (Robert) is a fact, so it is proved here.

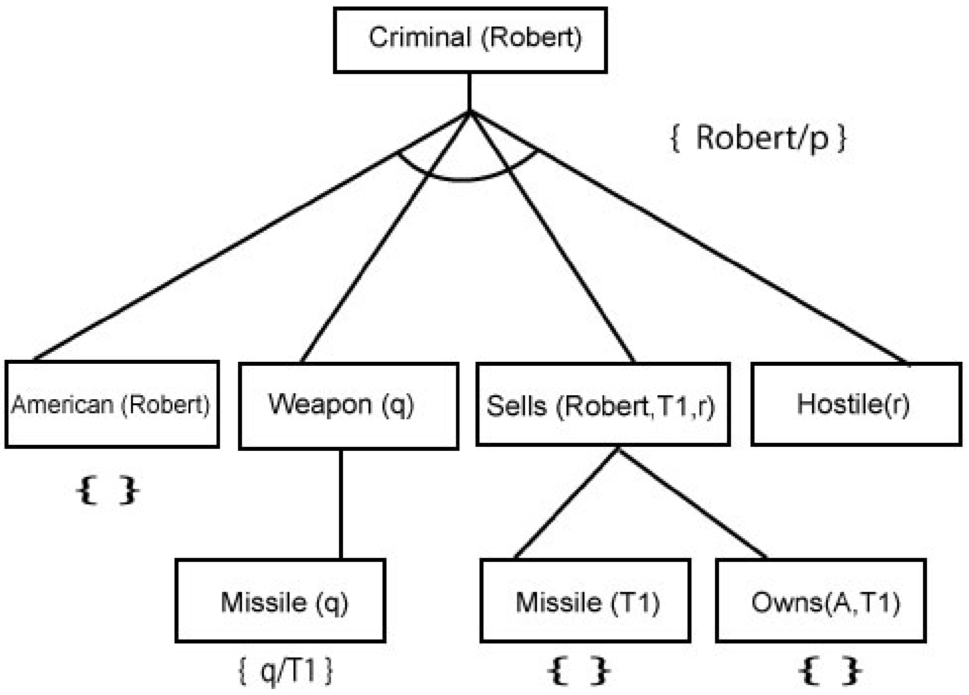


Step-3:t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



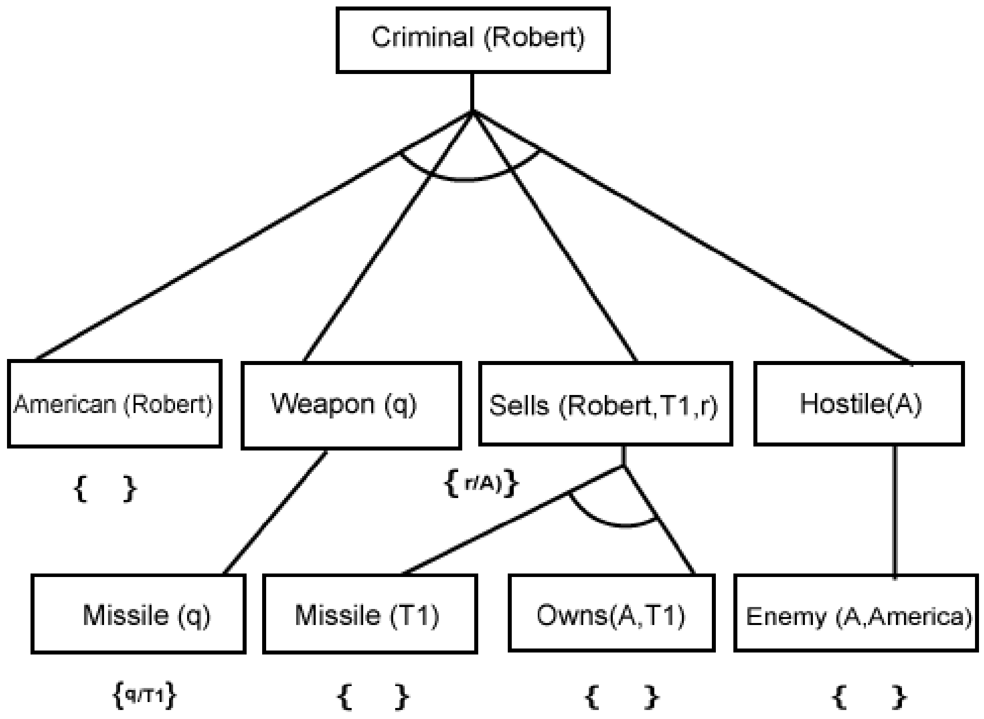
Step-4:

At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the **Rule-4**, with the substitution of A in place of r. So these two statements are proved here.



Step-5:

At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.



Difference between Forwarding Chaining and Backward Chaining:

	Forward Chaining	Backward Chaining
1.	When based on available data a decision is taken then the process is called as Forward chaining.	Backward chaining starts from the goal and works backward to determine what facts must be asserted so that the goal can be achieved.
2.	Forward chaining is known as data-driven technique because we reaches to the goal using the available data.	Backward chaining is known as goal-driven technique because we start from the goal and reaches the initial state in order to extract the facts.
3.	It is a bottom-up approach.	It is a top-down approach.
4.	It applies the Breadth-First Strategy.	It applies the Depth-First Strategy.
5.	Its goal is to get the conclusion.	Its goal is to get the possible facts or the required data.
6.	Slow as it has to use all the rules.	Fast as it has to use only a few rules.
7.	It operates in forward direction i.e it works from initial state to final decision.	It operates in backward direction i.e it works from goal to reach initial state.
8.	Forward chaining is used for the planning, monitoring, control, and interpretation application.	It is used in automated inference engines, theorem proofs, proof assistants and other artificial intelligence applications.

Probabilistic reasoning in Artificial intelligence

Uncertainty:

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors
3. Equipment fault
4. Temperature variation
5. Climate change.

Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.

- When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- **Bayes' rule**
- **Bayesian Statistics**

Note: We will learn the above two rules in later chapters.

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

1. $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.

1. P(A) = 0, indicates total uncertainty in an event A.

1. P(A) =1, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- P(¬A) = probability of a not happening event.
- P(¬A) + P(A) = 1.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional probability:

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

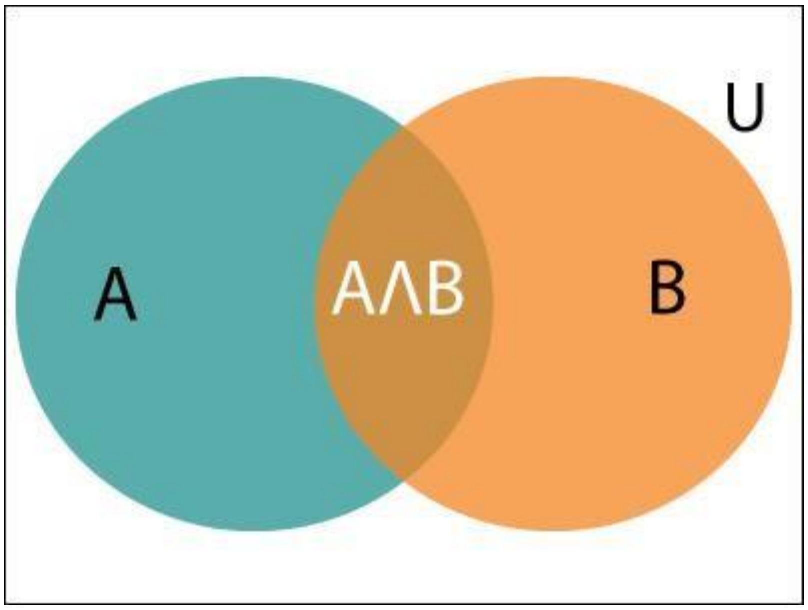
Where P($A \wedge B$)= Joint probability of a and B

P(B)= Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A∧*B*) by P(B)**.



Example:

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

Solution:

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

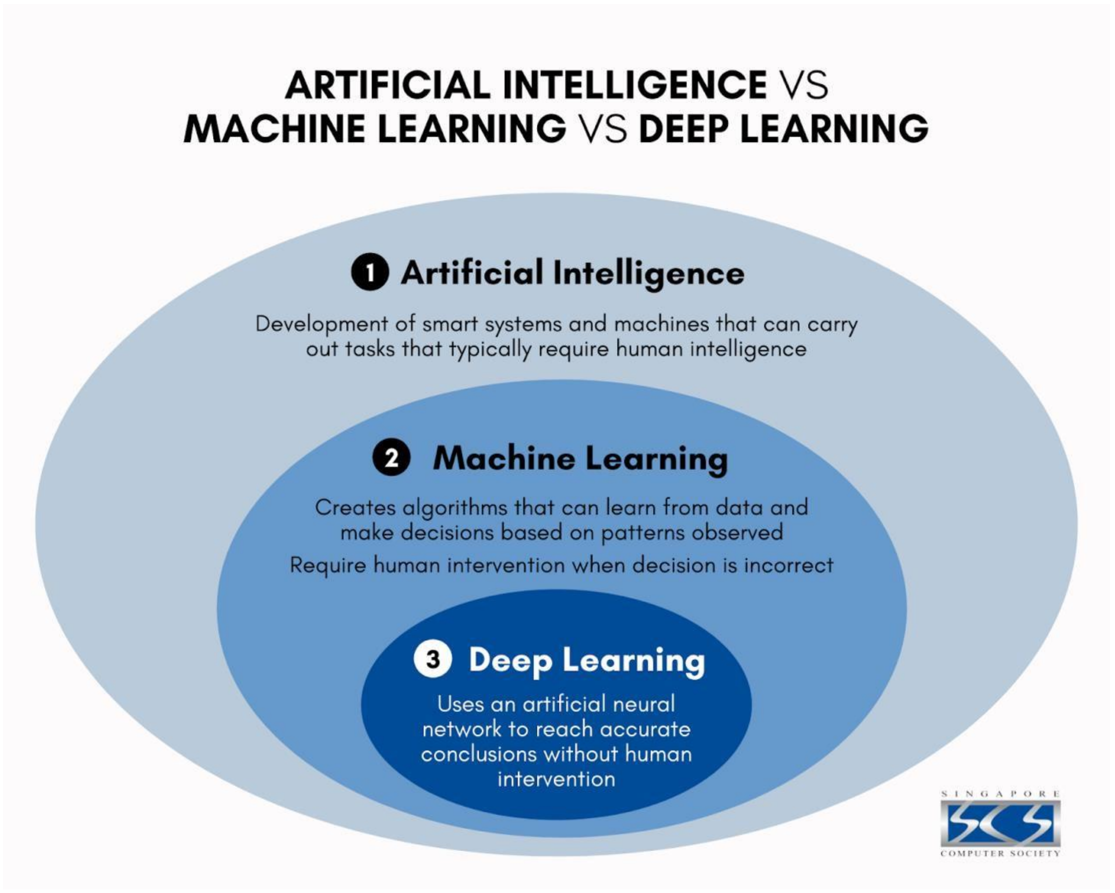
$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

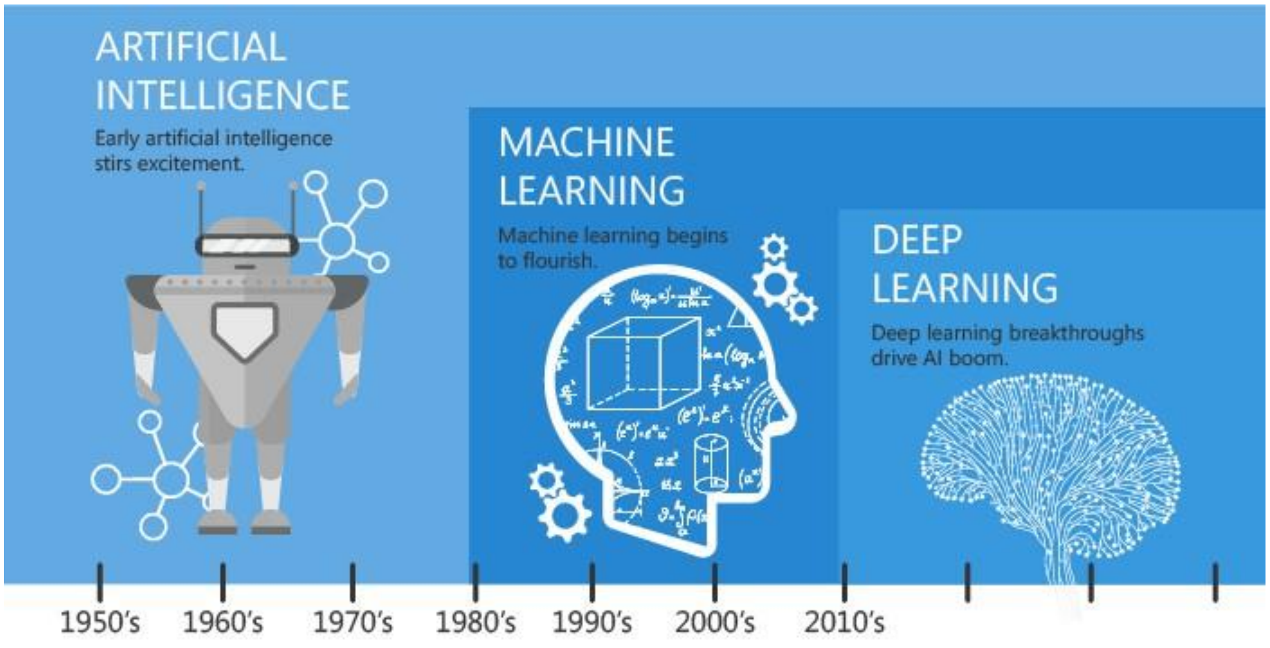
UNIT - III: Machine-Learning :

- Introduction. Machine Learning Systems
- Forms of Learning: Supervised and Unsupervised Learning, reinforcement
- theory of learning
- feasibility of learning
- Data Preparation
- Training versus testing and split.

Machine learning: Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.



ARTIFICIAL INTELLIGENCE VS MACHINE LEARNING VS DEEP LEARNING
1 Artificial Intelligence
Development of smart systems and machines that can carry out tasks that typically require human intelligence
2 Machine Learning
Creates algorithms that can learn from data and make decisions based on patterns observed
Require human intervention when decision is incorrect
3 Deep Learning
Uses an artificial neural network to reach accurate conclusions without human intervention
SINGAPORE SCS COMPUTER SOCIETY

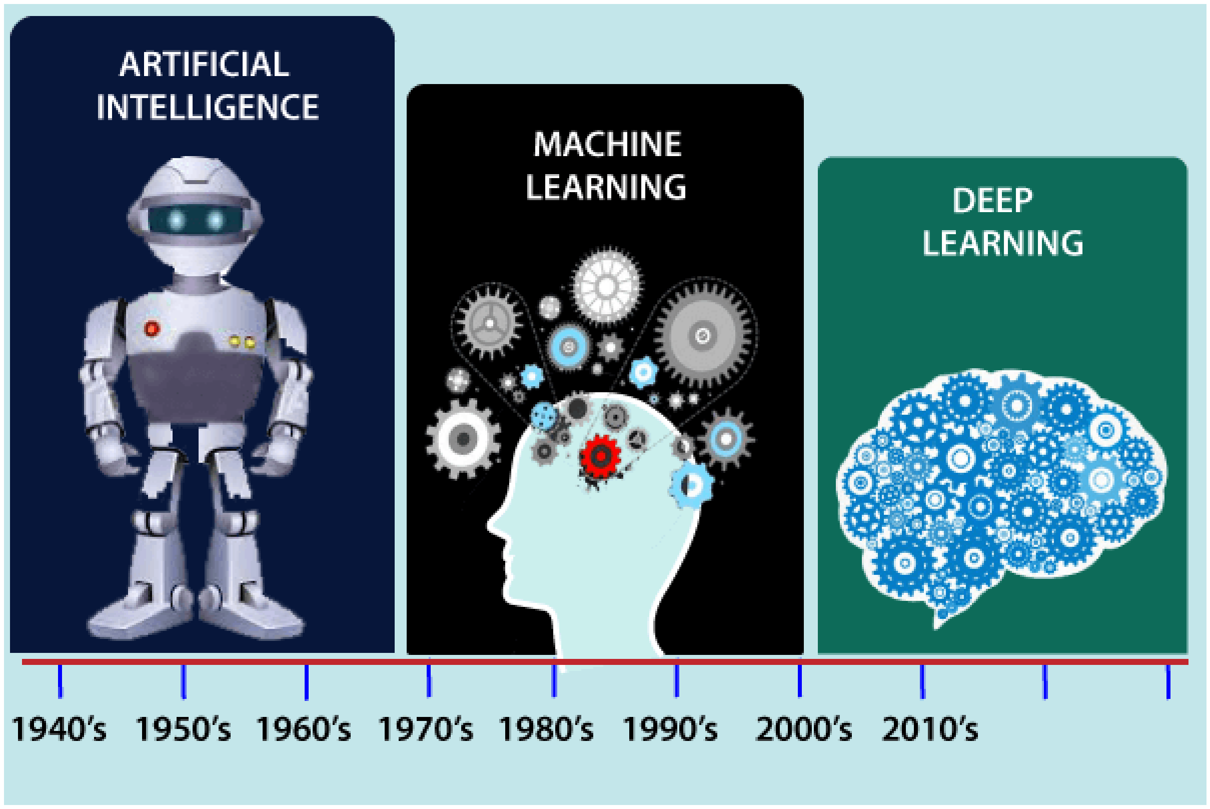


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

History of Machine Learning

Before some years (about 40-50 years), machine learning was science fiction, but today it is the part of our daily life. Machine learning is making our day to day life easy from **self-driving cars** to **Amazon virtual assistant "Alexa"**. However, the idea behind machine learning is so old and has a long history. Below some milestones are given which have occurred in the history of machine learning:



The early history of Machine Learning (Pre-1940):

- **1834:** In 1834, Charles Babbage, the father of the computer, conceived a device that could be programmed with punch cards. However, the machine was never built, but all modern computers rely on its logical structure.
- **1936:** In 1936, Alan Turing gave a theory that how a machine can determine and execute a set of instructions.

The era of stored program computers:

- **1940:** In 1940, the first manually operated computer, "ENIAC" was invented, which was the first electronic general-purpose computer. After that stored program computer such as EDSAC in 1949 and EDVAC in 1951 were invented.
- **1943:** In 1943, a human neural network was modeled with an electrical circuit. In 1950, the scientists started applying their idea to work and analyzed how human neurons might work.

Computer machinery and intelligence:

- **1950:** In 1950, Alan Turing published a seminal paper, "**Computer Machinery and Intelligence**," on the topic of artificial intelligence. **In his paper, he asked, "Can machines think?"**

Machine intelligence in Games:

- **1952:** Arthur Samuel, who was the pioneer of machine learning, created a program that helped an IBM computer to play a checkers game. It performed better more it played.
- **1959:** In 1959, the term "Machine Learning" was first coined by **Arthur Samuel**.

The first "AI" winter:

- The duration of 1974 to 1980 was the tough time for AI and ML researchers, and this duration was called as **AI winter**.
- In this duration, failure of machine translation occurred, and people had reduced their interest from AI, which led to reduced funding by the government to the researches.

Machine Learning from theory to reality

- **1959:** In 1959, the first neural network was applied to a real-world problem to remove echoes over phone lines using an adaptive filter.
- **1985:** In 1985, Terry Sejnowski and Charles Rosenberg invented a neural network **NETtalk**, which was able to teach itself how to correctly pronounce 20,000 words in one week.
- **1997:** The IBM's **Deep blue** intelligent computer won the chess game against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

Machine Learning at 21st century

2006:

- Geoffrey Hinton and his group presented the idea of profound getting the hang of utilizing profound conviction organizations.
- The Elastic Compute Cloud (EC2) was launched by Amazon to provide scalable computing resources that made it easier to create and implement machine learning models.

2007:

- Participants were tasked with increasing the accuracy of Netflix's recommendation algorithm when the Netflix Prize competition began.
- Support learning made critical progress when a group of specialists utilized it to prepare a PC to play backgammon at a top-notch level.

2008:

- Google delivered the Google Forecast Programming interface, a cloud-based help that permitted designers to integrate AI into their applications.
- Confined Boltzmann Machines (RBMs), a kind of generative brain organization, acquired consideration for their capacity to demonstrate complex information conveyances.

2009:

- Profound learning gained ground as analysts showed its viability in different errands, including discourse acknowledgment and picture grouping.
- The expression "Large Information" acquired ubiquity, featuring the difficulties and open doors related with taking care of huge datasets.

2010:

- The ImageNet Huge Scope Visual Acknowledgment Challenge (ILSVRC) was presented, driving progressions in PC vision, and prompting the advancement of profound convolutional brain organizations (CNNs).

2011:

- On Jeopardy! IBM's Watson defeated human champions., demonstrating the potential of question-answering systems and natural language processing.

2012:

- AlexNet, a profound CNN created by Alex Krizhevsky, won the ILSVRC, fundamentally further developing picture order precision and laying out profound advancing as a predominant methodology in PC vision.
- Google's Cerebrum project, drove by Andrew Ng and Jeff Dignitary, utilized profound figuring out how to prepare a brain organization to perceive felines from unlabeled YouTube recordings.

2013:

- Ian Goodfellow introduced generative adversarial networks (GANs), which made it possible to create realistic synthetic data.
- Google later acquired the startup DeepMind Technologies, which focused on deep learning and artificial intelligence.

2014:

- Facebook presented the DeepFace framework, which accomplished close human precision in facial acknowledgment.
- AlphaGo, a program created by DeepMind at Google, defeated a world champion Go player and demonstrated the potential of reinforcement learning in challenging games.

2015:

- Microsoft delivered the Mental Toolbox (previously known as CNTK), an open-source profound learning library.
- The performance of sequence-to-sequence models in tasks like machine translation was enhanced by the introduction of the idea of attention mechanisms.

2016:

- The goal of explainable AI, which focuses on making machine learning models easier to understand, received some attention.
- Google's DeepMind created AlphaGo Zero, which accomplished godlike Go abilities to play without human information, utilizing just support learning.

2017:

- Move learning acquired noticeable quality, permitting pretrained models to be utilized for different errands with restricted information.
- Better synthesis and generation of complex data were made possible by the introduction of generative models like variational autoencoders (VAEs) and Wasserstein GANs.
- These are only a portion of the eminent headways and achievements in AI during the predefined period. The field kept on advancing quickly past 2017, with new leap forwards, strategies, and applications arising.

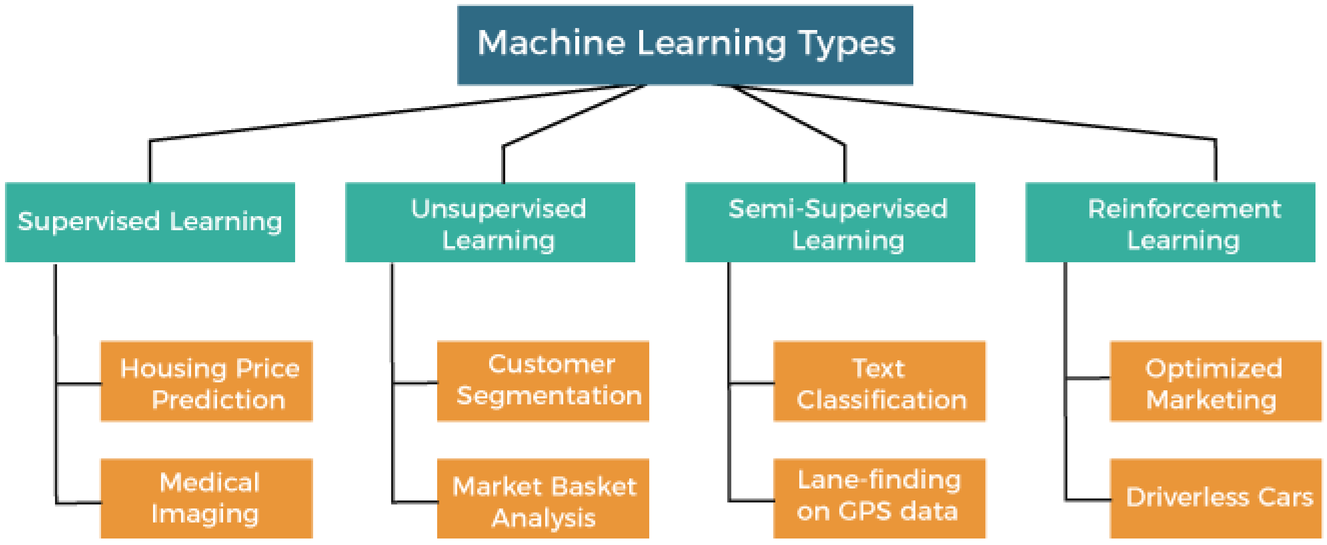
Types of Machine Learning

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



1. Supervised Machine Learning

Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc.** After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are **Risk Assessment, Fraud Detection, Spam filtering,** etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as **"Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

Advantages and Disadvantages of Supervised Learning

Advantages:

- Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.
- These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages:

- These algorithms are not able to solve complex tasks.
- It may predict the wrong output if the test data is different from the training data.
- It requires lots of computational time to train the algorithm.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

- **Image Segmentation:**
  Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
- **Medical Diagnosis:**
  Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
- **Fraud Detection -** Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
- **Spam detection -** In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
- **Speech Recognition -** Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

- **Clustering**
- **Association**

1) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

- **K-Means Clustering algorithm**
- **Mean-shift algorithm**
- **DBSCAN Algorithm**
- **Principal Component Analysis**
- **Independent Component Analysis**

2) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**

Advantages and Disadvantages of Unsupervised Learning Algorithm

Advantages:

- These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
- Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages:

- The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
- Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

Applications of Unsupervised Learning

- **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
- **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
- **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
- **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self-analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

Advantages and disadvantages of Semi-supervised Learning

Advantages:

- It is simple and easy to understand the algorithm.
- It is highly efficient.
- It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

Disadvantages:

- Iterations results may not be stable.
- We cannot apply these algorithms to network-level data.
- Accuracy is low.

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

Due to its way of working, reinforcement learning is employed in different fields such as **Game theory, Operation Research, Information theory, multi-agent systems.**

A reinforcement learning problem can be formalized using **Markov Decision Process(MDP).** In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.

Categories of Reinforcement Learning **Reinforcement learning is categorized mainly into two types of methods/algorithms:**

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.

- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition.

Real-world Use cases of Reinforcement Learning

- **Video Games:**
  RL algorithms are much popular in gaming applications. It is used to gain super-human performance. Some popular games that use RL algorithms are **AlphaGO** and **AlphaGO Zero**.
- **Resource Management:**
  The "Resource Management with Deep Reinforcement Learning" paper showed that how to use RL in computer to automatically learn and schedule resources to wait for different jobs in order to minimize average job slowdown.
- **Robotics:**
  RL is widely being used in Robotics applications. Robots are used in the industrial and manufacturing area, and these robots are made more powerful with reinforcement learning. There are different industries that have their vision of building intelligent robots using AI and Machine learning technology.
- **Text Mining**
  Text-mining, one of the great applications of NLP, is now being implemented with the help of Reinforcement Learning by Salesforce company.

Advantages and Disadvantages of Reinforcement Learning

Advantages

- It helps in solving complex real-world problems which are difficult to be solved by general techniques.
- The learning model of RL is similar to the learning of human beings; hence most accurate results can be found.
- Helps in achieving long term results.

Disadvantage

- RL algorithms are not preferred for simple problems.
- RL algorithms require huge data and computations.
- Too much reinforcement learning can lead to an overload of states which can weaken the results.

Train and Test datasets in Machine Learning

Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. ***Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model***.

What is Training Dataset?

The ***training data is the biggest (in -size) subset of the original dataset, which is used to train or fit the machine learning model***. Firstly, the training data is fed to the ML algorithms, which lets them learn how to make predictions for the given task.

For example, for training a sentiment analysis model, the training data could be as below:

Input	**Output (Labels)**
The New UI is Great	Positive
Update is really Slow	Negative

The training data varies depending on whether we are using Supervised Learning or Unsupervised Learning Algorithms.

For **Unsupervised learning**, the training data contains unlabeled data points, i.e., inputs are not tagged with the corresponding outputs. Models are required to find the patterns from the given training datasets in order to make predictions.

On the other hand, for supervised learning, the training data contains labels in order to train the model and make predictions.

The type of training data that we provide to the model is highly responsible for the model's accuracy and prediction ability. It means that the better the quality of the training data, the better will be the performance of the model. Training data is approximately more than or equal to 60% of the total data for an ML project.

What is Test Dataset?

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. ***The test dataset is another subset of original data, which is independent of the training dataset***. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project.

At this stage, we can also check and compare the testing accuracy with the training accuracy, which means how accurate our model is with the test dataset against the training dataset. If the accuracy of the model on training data is greater than that on testing data, then the model is said to have overfitting.

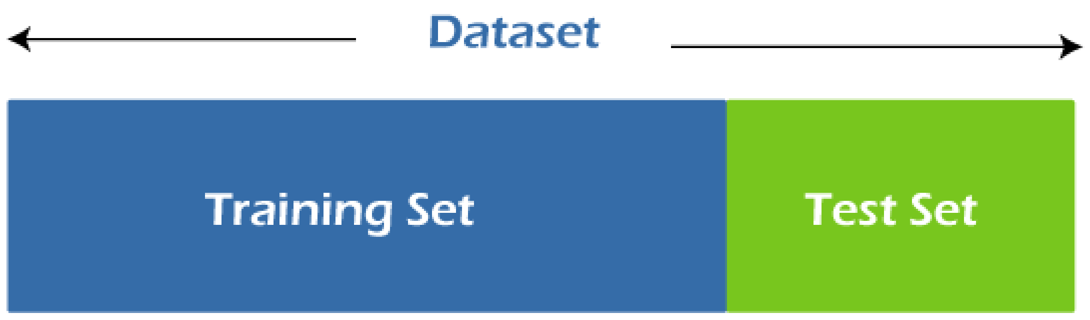
The testing data should:

- Represent or part of the original dataset.
- It should be large enough to give meaningful predictions.

Need of Splitting dataset into Train and Test set

Splitting the dataset into train and test sets is one of the important parts of data pre-processing, as by doing so, we can improve the performance of our model and hence give better predictability.

We can understand it as if we train our model with a training set and then test it with a completely different test dataset, and then our model will not be able to understand the correlations between the features.



Therefore, if we train and test the model with two different datasets, then it will decrease the performance of the model. Hence it is important to split a dataset into two parts, i.e., train and test set.

In this way, we can easily evaluate the performance of our model. Such as, if it performs well with the training data, but does not perform well with the test dataset, then it is estimated that the model may be overfitted.

For splitting the dataset, we can use the **train_test_split** function of **scikit-learn.**

The bellow line of code can be used to split dataset:

1. from sklearn.model_selection import train_test_split
2. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)

Explanation:

In the first line of the above code, we have imported the train_test_split function from the **sklearn** library.

In the second line, we have used four variables, which are

- x_train: It is used to represent features for the training data
- x_test: It is used to represent features for testing data
- y_train: It is used to represent dependent variables for training data
- y_test: It is used to represent independent variable for testing data
- In the train_test_split() function, we have passed four parameters. Which first two are for arrays of data, and test_size is for specifying the size of the test set. The test_size may be .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter, random_state, is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

Overfitting and Underfitting issues

Overfitting and underfitting are the most common problems that occur in the Machine Learning model.

*A model can be said as **overfitted** when it performs quite well with the training dataset but does not generalize well with the new or unseen dataset*. The issue of overfitting occurs when the model tries to cover all the data points and hence starts caching noises present in the data. Due to this, it can't generalize well to the new dataset. Because of these issues, the accuracy and efficiency of the model degrade. Generally, the complex model has a high chance of overfitting. There are various ways by which we can avoid overfitting in the model, such as Using the **Cross-Validation method, early stopping the training, or by regularization**, etc.

On the other hand, the ***model is said to be under-fitted when it is not able to capture the underlying trend of the data***. It means the model shows poor performance even with the training dataset. In most cases, underfitting issues occur when the model is not perfectly suitable for the problem that we are trying to solve. To avoid the overfitting issue, we can either increase the training time of the model or increase the number of features in the dataset.

Training data vs. Testing Data

- The main difference between training data and testing data is that training data is the subset of original data that is used to train the machine learning model, whereas testing data is used to check the accuracy of the model.
- The training dataset is generally larger in size compared to the testing dataset. The general ratios of splitting train and test datasets are **80:20, 70:30, or 90:10.**
- Training data is well known to the model as it is used to train the model, whereas testing data is like unseen/new data to the model.

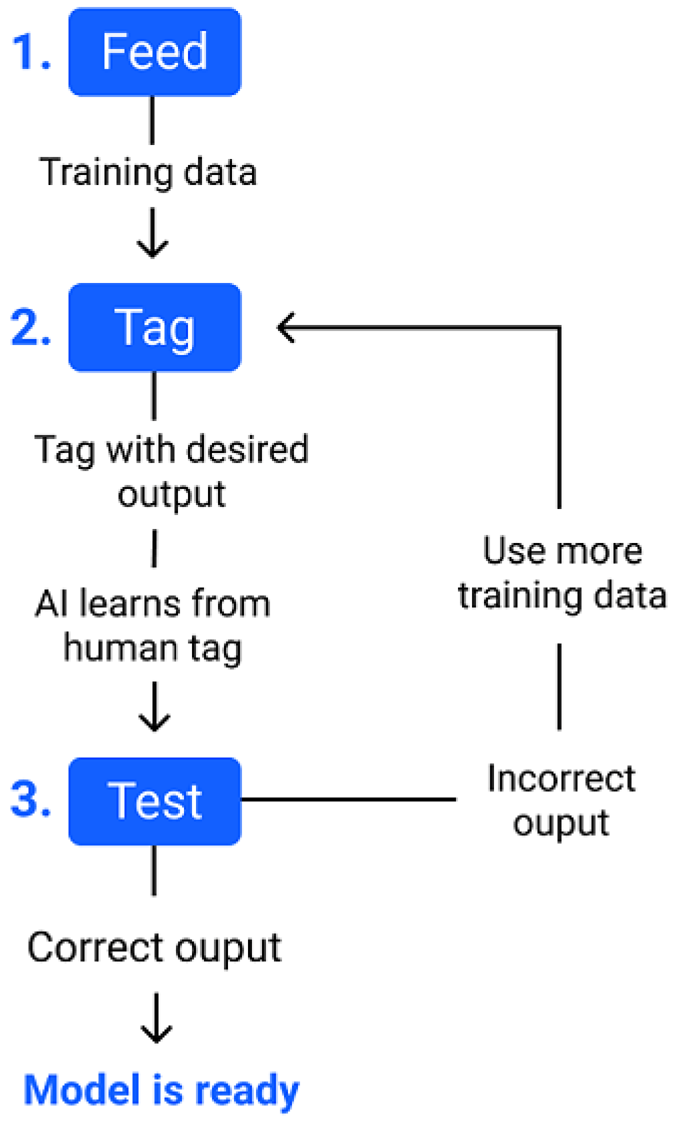
How do training and testing data work in Machine Learning?

Machine Learning algorithms enable the machines to make predictions and solve problems on the basis of past observations or experiences. These experiences or observations an algorithm can take from the training data, which is fed to it. Further, one of the great things about ML algorithms is that they can learn and improve over time on their own, as they are trained with the relevant training data.

Once the model is trained enough with the relevant training data, it is tested with the test data. We can understand the whole process of training and testing in three steps, which are as follows:

1. **Feed:** Firstly, we need to train the model by feeding it with training input data.
2. **Define:** Now, training data is tagged with the corresponding outputs (in Supervised Learning), and the model transforms the training data into text vectors or a number of data features.
3. **Test:** In the last step, we test the model by feeding it with the test data/unseen dataset. This step ensures that the model is trained efficiently and can generalize well.

The above process is explained using a flowchart given below:



Traits of Quality training data

As the ability to the prediction of an ML model highly depends on how it has been trained, therefore it is important to train the model with quality data. Further, ML works on the concept of "Garbage In, Garbage Out." It means that whatever type of data we will input into our model, it will make the predictions accordingly. For a quality training data, the below points should be considered:

1. Relevant

The very first quality of training data should be relevant to the problem that you are going to solve. It means that whatever data you are using should be relevant to the current problem. For example, if you are building a model to analyze social media data, then data should be taken from different social sites such as Twitter, Facebook, Instagram, etc.

2. Uniform:

There should always be uniformity among the features of a dataset. It means all data for a particular problem should be taken from the same source with the same attributes.

3. Consistency: In the dataset, the similar attributes must always correspond to the similar label in order to ensure uniformity in the dataset.

4. Comprehensive: The training data must be large enough to represent sufficient features that you need to train the model in a better way. With a comprehensive dataset, the model will be able to learn all the edge cases.

In machine learning (ML), a fundamental task is the development of algorithm models that analyze scenarios and make predictions. During this work, analysts fold various examples into training, validation, and test datasets. Below, we review the differences between each function.

Train vs. Validate vs. Test

Training datasets comprise samples used to fit machine learning models under construction, i.e., carry out the actual AI development. Constructing these robust pillars of AI involves following best practices.

Training data are collections of examples or samples that are used to 'teach' or 'train the machine learning model. The model uses a training data set to understand the patterns and relationships within the data, thereby learning to make predictions or decisions without being explicitly programmed to perform a specific task.

In contrast, validation datasets contain different samples to evaluate trained ML models. It is still possible to tune and control the model at this stage. Working on validation data is used to assess the model performance and fine-tune the parameters of the model. This becomes an iterative process wherein the model learns from the training data and is then validated and fine-tuned on the validation set. A validation dataset tells us how well the model is learning and adapting, allowing for adjustments and optimizations to be made to the model's parameters or hyperparameters before it's finally put to the test.

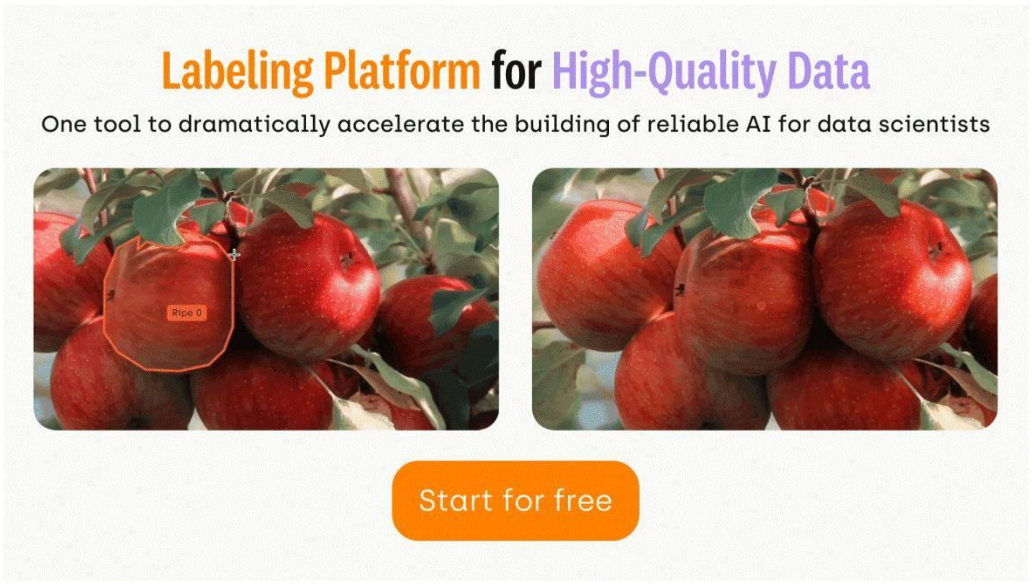
Finally, a test data set is a separate sample, an unseen data set, to provide an unbiased final evaluation of a model fit. The inputs in the test data are similar to the previous stages but not the same data. The test data set mirrors real-world data the machine learning model has never seen before. Its primary purpose is to offer a fair and final assessment of how the model would perform when it encounters new data in a live, operational environment.

The training data and validation data can play additional roles in model preparation. They can aid in feature selection, a process in which the most relevant or significant variables in the data are selected to improve the model performance. They can also contribute to tuning the model's complexity, balancing fitting the data well, and maintaining a good level of generalization. The fit of the final model is a combined result from the aggregate of these inputs.

Training Data Sets

Initially, machine learning development involves starting inputs within specified project parameters. These parameters could include the specific problem the model is designed to solve, the type of data it will process, the performance metrics it aims to optimize, etc.

The process of training machine learning models involves setting up a complex network of connections between the individual elements within the model. These individual elements are often referred to as 'neurons' in the context of neural networks.


Labeling Platform for High-Quality Data
One tool to dramatically accelerate the building of reliable AI for data scientists
Ripe 0
Start for free

One of the critical aspects requires the expert setting of weightings between the various connections of so-called neurons within the ML model or estimator*. However, these initial settings are not static; they are modified during the training process based on the feedback received from the model performance.

After introducing this first set of training data, developers compare the resulting output to target answers. This comparison forms the basis for an error or loss function that quantifies the model performance - the discrepancy between the predictions and the actual values. Next, they adjust the model's parameters, weighting, and functionality. This adjustment is done using various optimization algorithms, the most common of which is Gradient Descent.

More than one 'epoch' or iteration of this adjustment loop is often necessary. The ultimate goal is to create models that can make accurate predictions when exposed to new, unknown data that they weren't trained on. To ensure that the machine learning model can generalize well, one must strike a balance during the training process to avoid underfitting (where the model is too simple to capture the underlying pattern) or overfitting (where the model is overly complex and captures the noise in the training data).

Validation Data Sets

The next stage involves using a validation set to estimate the accuracy of the ML model concerned. During this phase, developers ensure that new data classification is precise and results are predictable.

Validation sets comprise unbiased inputs and expected results designed to check the function and performance of the model. Cross-validation (CV) techniques come into play during this phase, and there are different methods of CV that exist, but all aim to ensure stability by estimating how a predictive model will perform. An example is the usage of rotation estimation or out-of-sample testing to assure reasonable precision.

Resampling and fine-tuning involve various iterations. Whatever the methodology, these verification techniques aim to assess the results and check them against independent inputs. It is also possible to adjust the hyperparameters, i.e., the values used to control the overall process.

That said, not all models require validation sets. Some experts consider that ML models with no hyperparameters or those that do not have tuning options do not need a validation set. Still, in most practical applications, validation sets play a crucial role in ensuring the model's robustness and performance.

Test Data Sets

The final step is to use a test set to verify the model's functionality. Some publications refer to the validation dataset as a test set, especially if there are only two subsets instead of three. Similarly, if records in this final test set have not formed part of a previous evaluation or cross-validation, they might also constitute a holdout set.

Test samples provide a simulated real-world check using unseen inputs and expected results. In practice, there could be some overlap between validation and testing. Each procedure shows that the ML model will function in a live environment once out of testing.

The difference is that while validating, the results provide metrics as feedback to train the model better. In contrast, the performance of a test procedure merely confirms that the model works overall, i.e. as a black box with inputs passed through it. During this final evaluation, there is no adjustment of hyperparameters.

How to Split Your Machine Learning Data

Above, we have seen the distinction between the different types of sets. The next decision is splitting enough data between each of them.

The optimum ratio when dividing records with enough data between each function – train, validate, and test – depends on the application usage, model type, and data dimensions. Most ML models benefit from having a substantial number of scenarios from which to train.

At the validation stage, models with few or no hyperparameters are straightforward to validate and tune. Thus, a relatively small dataset should suffice.

In contrast, models with multiple hyperparameters require enough data to validate likely inputs. Cross-validation might be helpful in these cases, too. Generally, apportioning 80 percent of the data to train, 10 percent to validate, and 10 percent to test scenarios ought to be a reasonable initial split.

Common Pitfalls in The Training Data Split

However, a validation dataset must not be too small. Otherwise, the ML model will be untuned, imprecise, or even inaccurate. In particular, the F1 score – a statistical measure of precision and recall – will vary too widely.

One cycle through a complete dataset in artificial neural networks is called an epoch. And as mentioned earlier, training a model usually takes more than one epoch.

The train-test-validation ratio depends on the usage case. Getting the procedure right comes with experience.

Cross-Validation

An alternative approach involves splitting an initial dataset into two halves, training and testing. Firstly, with the test data set kept to one side, a proportion of randomly chosen training data becomes the actual training set. The remaining values in the array are for later iterations to validate inputs. For example, the split might vary from two halves to a ratio of 80:20 percent.

This cross-validation involves one or more splits of the training data set and validation data set. In particular, K-fold cross-validation aims to maximize accuracy in testing by dividing the source data into several bins or groups. All except one of these are for training and validation purposes. The last is for testing.

In this method, each data set runs as a separate experiment. Analysts then calculate the average of all the runs to obtain the mean accuracy. Once the result falls within specified limits, the final step before signoff is using the remaining fold of test data to double-check the findings.

Other methods of cross-validation are Stratified K-Fold cross-validation to guarantee a suitable representation of each class and avoid bias, Leave-P-Out cross-validation, which is ideally used for smaller sample sizes because of its high computational demands, and Rolling cross-validation for time series-based data.

Low-Quality Training Data

Like other areas of IT, machine learning algorithms follow the time-tested principle of GIGO: garbage in, garbage out. So, to ensure reliable and robust algorithms, the following three components are necessary:

- Quantity. Sufficient data is important for the model to learn how to interact with users. As an analogy, humans need a considerable amount of information before becoming an expert.
- Quality. In themselves, data will not guarantee reliable results. Real-world scenarios and test cases that represent likely conditions are vital. Data should mimic the user input that the new algorithm will receive. It is essential to fold in data on which the application will rely, such as a combination of images, videos, sounds, and voices.
- Diversity. ML requires algorithms trained on more than one input fold to simulate most, if not all, likely and possible cases.

Designers should seek to prevent bias in models. Applications must comply with legislation and should conform to inclusivity guidelines. They should not display prejudice based on age, race, gender, language, marital status, or other identifying factors.

Overfitting

When developing an ML model, an essential principle is that the validation set and test set must remain separate. Otherwise, an overfit might occur. Overfitting is when exceptional or unreal conditions lead to incorrect outputs. In other words, the statistical model fits precisely with the inputs used to train it and will probably not be accurate.

Instead, the aim should be to generalize. The proper application of cross-validation ought to minimize overfitting and ensure that the algorithm's prediction and classification functionality are correct.

Overemphasis on Validation and Test Set Metrics

Overfitting can also arise if the development methodology overuses search techniques to look for empirical relationships in the input samples. This approach tends to identify relationships that might not apply in the real world.

As an analogy, it is akin to looking for connections that do not exist between random events. Nonetheless, discerning between occasional coincidences and the emergence of new patterns does involve a delicate balancing act, with careful evaluation of the probabilities involved.

Although a validation set contains different data, it is essential to remember that evaluations should not be too lengthy. Otherwise, the model tends to become more biased as validation data perfuses into the configuration.

Training, Validation & Test Sets: Key Takeaways

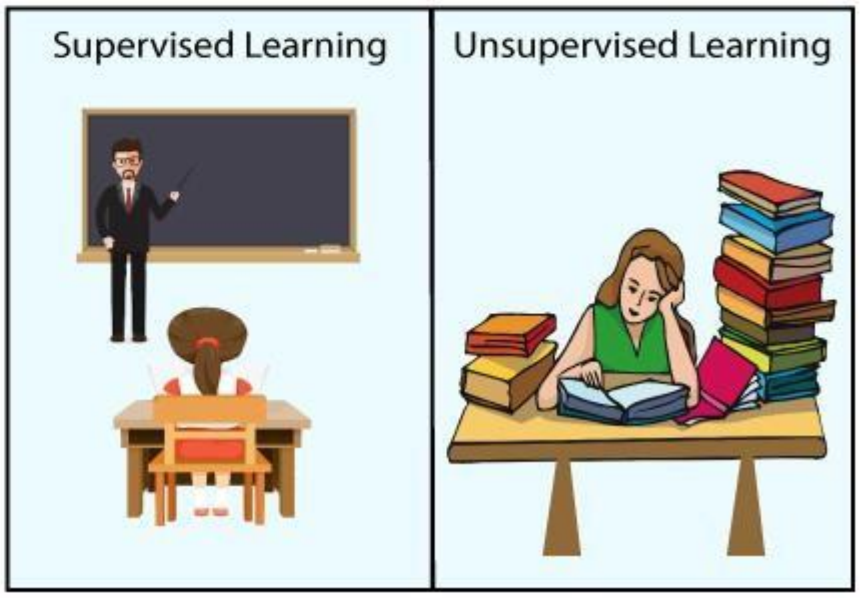
Quality is paramount for AI to deliver accurate results in the ingenious and expanding field of ML. Sound predictions and stable system behavior require the correct application of various key principles.

Essential considerations when organizing data for test sets are that:

- Training data builds the ML algorithm. Analysts feed in information and look for corresponding expected outputs. The import should be within specification and sufficient in quantity.
- When validating, it is necessary to use fold-in unseen values. Here, the new inputs and outputs enable ML designers to evaluate how accurate the new model's predictions are.
- Overfitting can result from too much searching or excessive amounts of validation data.
- While supervised ML approaches use a tag or classifier to identify training records, test records must remain untagged. The same data labels could enable the ML model to single out a shared reference, leading to anomalies in the results.

Difference between Supervised and Unsupervised Learning

Supervised and Unsupervised learning are the two techniques of machine learning. But both the techniques are used in different scenarios and with different datasets. Below the explanation of both learning methods along with their difference table is given.



Supervised Machine Learning:

Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: **Classification** and **Regression**.

Learn more Supervised Machine Learning

Example: Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.

Unsupervised Machine Learning:

Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

Learn more Unsupervised Machine Learning

Unsupervised learning can be used for two types of problems: **Clustering** and **Association**.

Example: To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

The main differences between Supervised and Unsupervised learning are given below:

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train	Unsupervised learning does not need any

the model.	supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

UNIT - IV

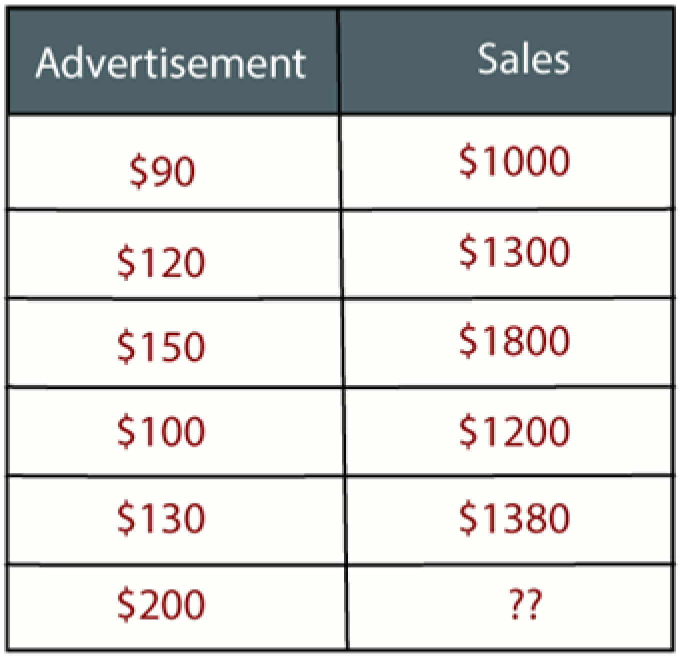
Supervised Learning:
Regression: Linear Regression, multi linear regression, Polynomial Regression, logistic regression, Non-linear Regression, Model evaluation methods. **Classification:** – support vector machines (SVM) , Naïve Bayes classification

Regression Analysis in Machine learning

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.

We can understand the concept of regression analysis using the below example:

Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:



Advertisement	Sales
\$90	\$1000
\$120	\$1300
\$150	\$1800
\$100	\$1200
\$130	\$1380
\$200	??

Now, the company wants to do the advertisement of \$200 in the year 2019 **and wants to know the prediction about the sales for this year**. So to solve such type of prediction problems in machine learning, we need regression analysis.

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.

In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data. In simple words, ***"Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum."*** The distance between datapoints and line tells whether a model has captured a strong relationship or not.

Some examples of regression can be as:

- Prediction of rain using temperature and other factors
- Determining Market trends
- Prediction of road accidents due to rash driving.

Terminologies Related to the Regression Analysis:

- **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
- **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

Why do we use Regression Analysis?

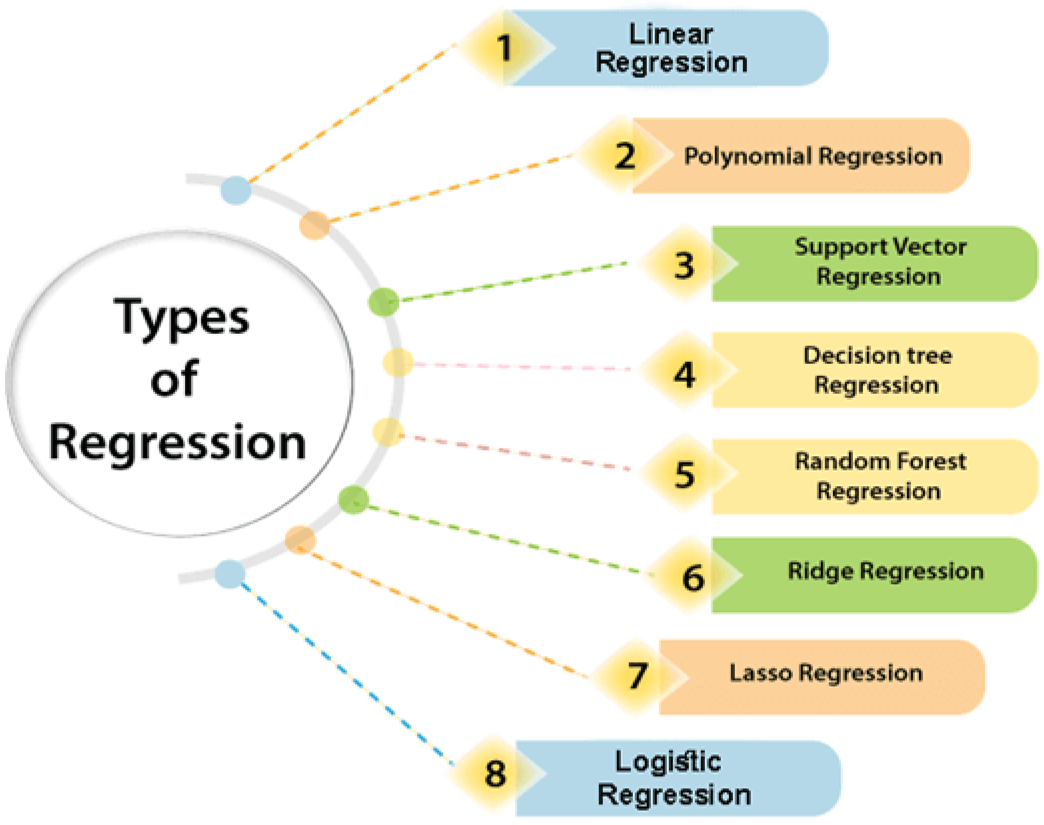
As mentioned above, Regression analysis helps in the prediction of a continuous variable. There are various scenarios in the real world where we need some future predictions such as weather condition, sales prediction, marketing trends, etc., for such case we need some technology which can make predictions more accurately. So for such case we need Regression analysis which is a statistical method and used in machine learning and data science. Below are some other reasons for using Regression analysis:

- Regression estimates the relationship between the target and the independent variable.
- It is used to find the trends in data.
- It helps to predict real/continuous values.
- By performing the regression, we can confidently determine the **most important factor, the least important factor, and how each factor is affecting the other factors**.

Types of Regression

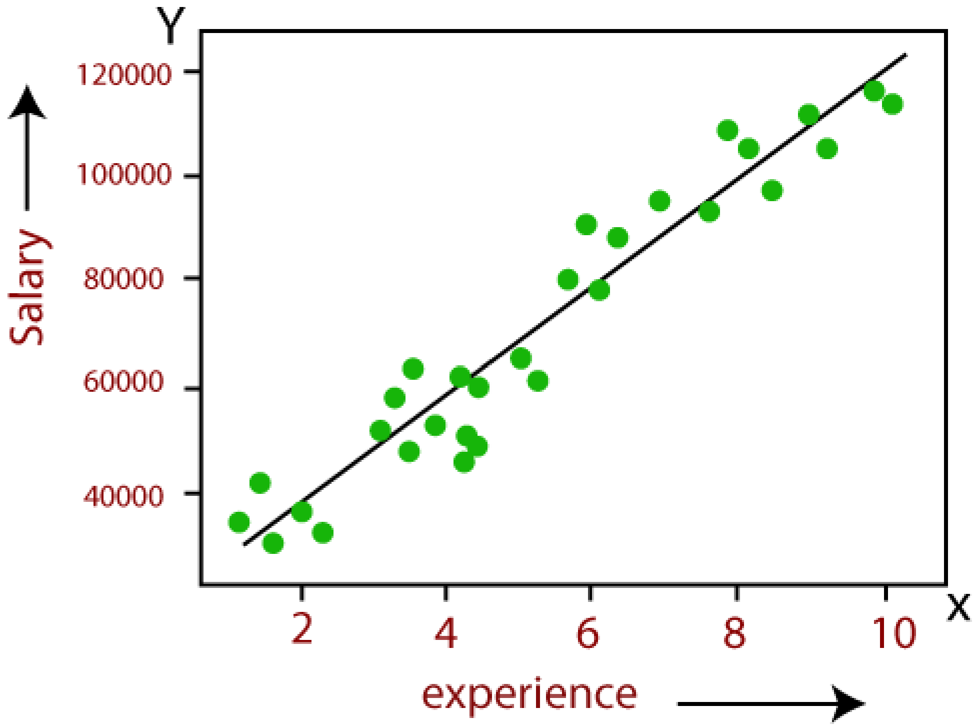
There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

- **Linear Regression**
- **Logistic Regression**
- **Polynomial Regression**
- **Support Vector Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **Ridge Regression**
- **Lasso Regression:**



Linear Regression:

- Linear regression is a statistical regression method which is used for predictive analysis.
- It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
- It is used for solving the regression problem in machine learning.
- Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
- If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**.
- The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of **the year of experience**.



- Below is the mathematical equation for Linear regression:

1. Y= aX+b

Here, Y = dependent variables (target variables),
X= Independent variables (predictor variables),
a and b are the linear coefficients

Some popular applications of linear regression are:

- **Analyzing trends and sales estimates**
- **Salary forecasting**
- **Real estate prediction**
- **Arriving at ETAs in traffic.**

Logistic Regression:

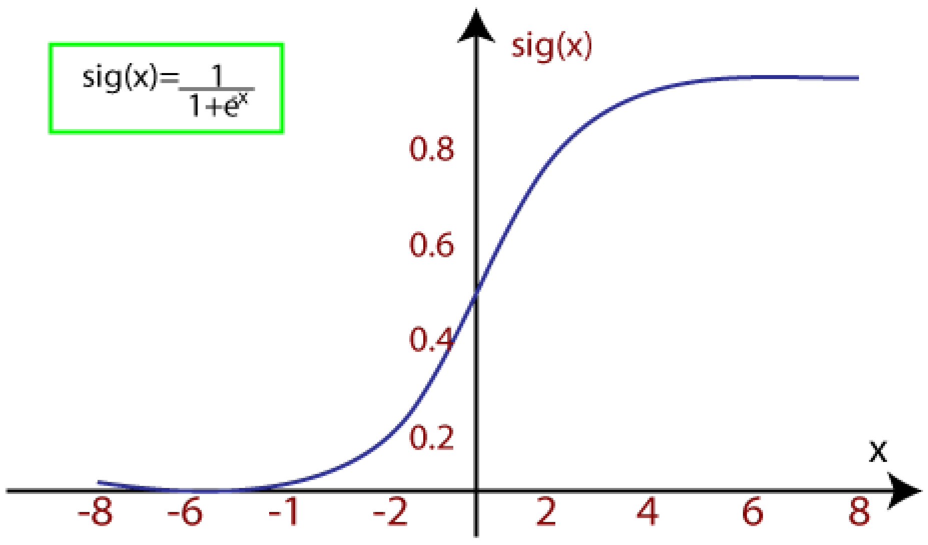
- Logistic regression is another supervised learning algorithm which is used to solve the classification problems. In **classification problems**, we have dependent variables in a binary or discrete format such as 0 or 1.
- Logistic regression algorithm works with the categorical variable such as 0 or 1, Yes or No, True or False, Spam or not spam, etc.
- It is a predictive analysis algorithm which works on the concept of probability.

- Logistic regression is a type of regression, but it is different from the linear regression algorithm in the term how they are used.
- Logistic regression uses **sigmoid function** or logistic function which is a complex cost function. This sigmoid function is used to model the data in logistic regression. The function can be represented as:

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

- f(x)= Output between the 0 and 1 value.
- x= input to the function
- e= base of natural logarithm.

When we provide the input values (data) to the function, it gives the S-curve as follows:



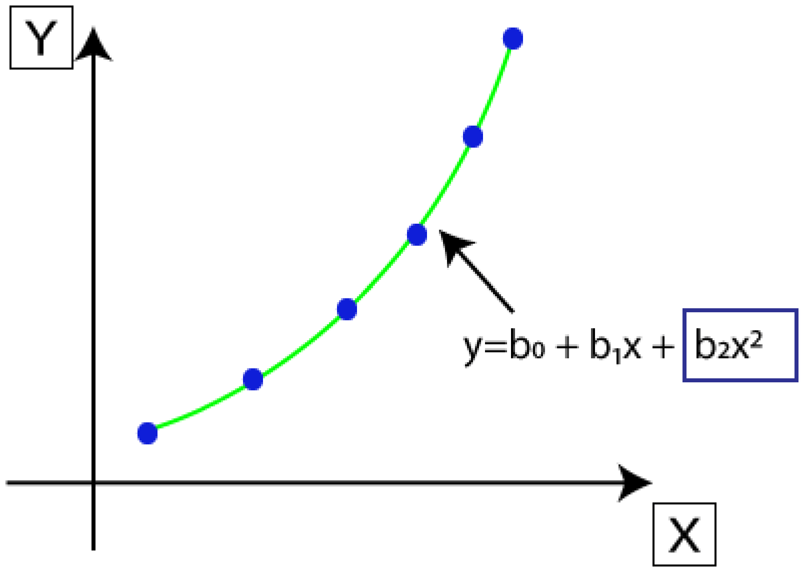
- It uses the concept of threshold levels, values above the threshold level are rounded up to 1, and values below the threshold level are rounded up to 0.

There are three types of logistic regression:

- **Binary(0/1, pass/fail)**
- **Multi(cats, dogs, lions)**
- **Ordinal(low, medium, high)**

Polynomial Regression:

- Polynomial Regression is a type of regression which models the **non-linear dataset** using a linear model.
- It is similar to multiple linear regression, but it fits a non-linear curve between the value of x and corresponding conditional values of y.
- Suppose there is a dataset which consists of datapoints which are present in a non-linear fashion, so for such case, linear regression will not best fit to those datapoints. To cover such datapoints, we need Polynomial regression.
- **In Polynomial regression, the original features are transformed into polynomial features of given degree and then modeled using a linear model.** Which means the datapoints are best fitted using a polynomial line.



- The equation for polynomial regression also derived from linear regression equation that means Linear regression equation Y= b_0+ b_1x, is transformed into Polynomial regression equation Y= b_0+b_1x+ b_2x^2+ b_3x^3+.........+ b_nx^n.
- Here Y is the **predicted/target output, b_0, b_1,. ...b_n are the regression coefficients**. x is our **independent/input variable**.
- The model is still linear as the coefficients are still linear with quadratic

Note: This is different from Multiple Linear regression in such a way that in Polynomial regression, a single element has different degrees instead of multiple variables with the same degree.

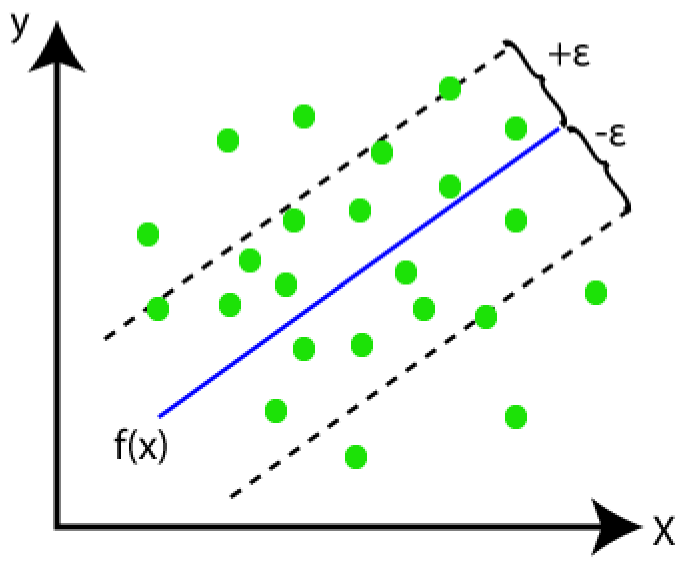
Support Vector Regression:

Support Vector Machine is a supervised learning algorithm which can be used for regression as well as classification problems. So if we use it for regression problems, then it is termed as Support Vector Regression.

Support Vector Regression is a regression algorithm which works for continuous variables. Below are some keywords which are used in **Support Vector Regression**:

- **Kernel:** It is a function used to map a lower-dimensional data into higher dimensional data.
- **Hyperplane:** In general SVM, it is a separation line between two classes, but in SVR, it is a line which helps to predict the continuous variables and cover most of the datapoints.
- **Boundary line:** Boundary lines are the two lines apart from hyperplane, which creates a margin for datapoints.
- **Support vectors:** Support vectors are the datapoints which are nearest to the hyperplane and opposite class.

In SVR, we always try to determine a hyperplane with a maximum margin, so that maximum number of datapoints are covered in that margin. ***The main goal of SVR is to consider the maximum datapoints within the boundary lines and the hyperplane (best-fit line) must contain a maximum number of datapoints***. Consider the below image:

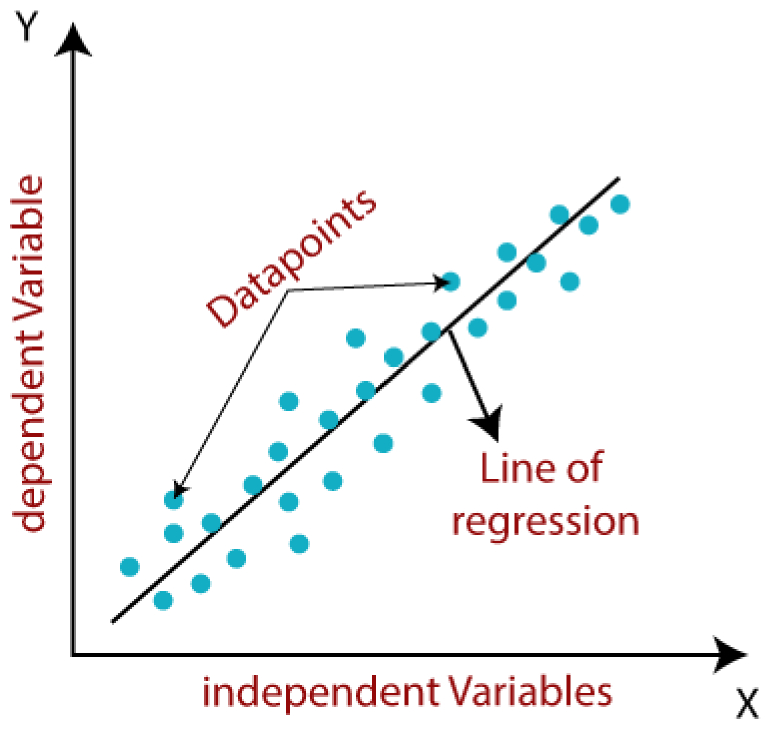


Here, the blue line is called hyperplane, and the other two lines are known as boundary lines.

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



$$y = a_0 + a_1 x + \varepsilon$$

Here,

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- **Simple Linear Regression:**

  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- **Multiple Linear regression:**

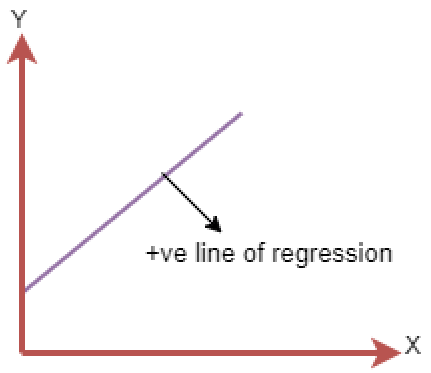
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:**

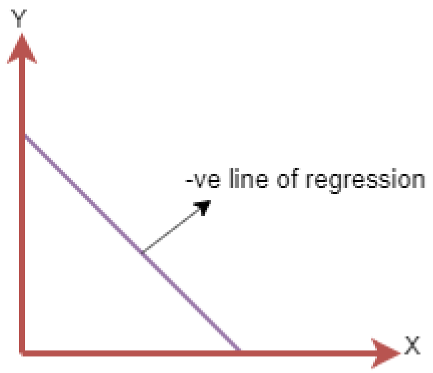
  If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: **$Y= a_0+a_1x$**

- **Negative Linear Relationship:**

- If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: **$Y= -a_0+a_1x$**

Finding the best fit line:

When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Cost function-

- The different values for weights or coefficient of lines (a_0, a_1) gives the different line of regression, and the cost function is used to estimate the values of the coefficient for the best fit line.
- Cost function optimizes the regression coefficients or weights. It measures how a linear regression model is performing.
- We can use the cost function to find the accuracy of the **mapping function**, which maps the input variable to the output variable. This mapping function is also known as **Hypothesis function**.

For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values. It can be written as:

$$\text{MSE}= 1\frac{1}{N}\sum_{i=1}^{n}(y_i \ - (a_1x_i + a_0))^2$$

Where,

N=Total number of observation

Yi = Actual value

$(a1x_i+a_0)$= Predicted value.

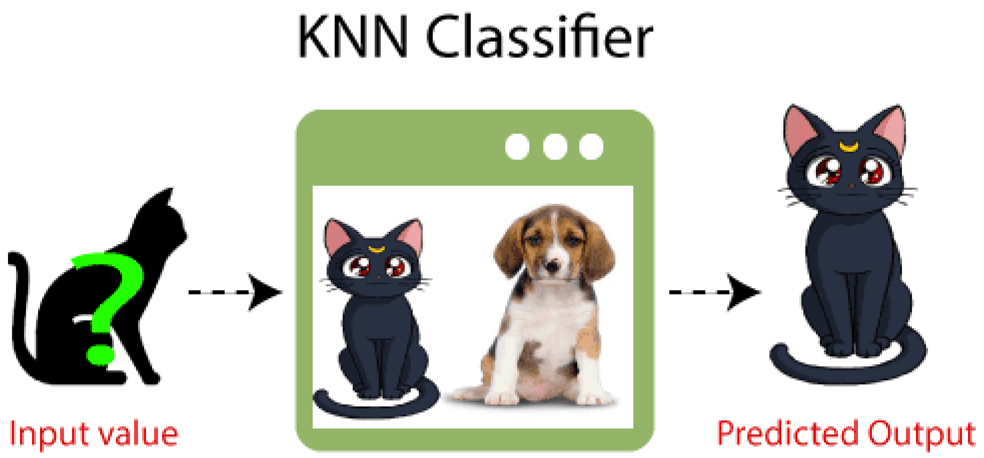
Residuals: The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function

UNIT- V

Unsupervised learning
Nearest neighbor models – K-means – clustering around medoids– silhouettes – hierarchical clustering – k-d trees ,Clustering trees – learning ordered rule lists – learning unordered rule .
Reinforcement learning- Example: Getting Lost -State and Action Spaces

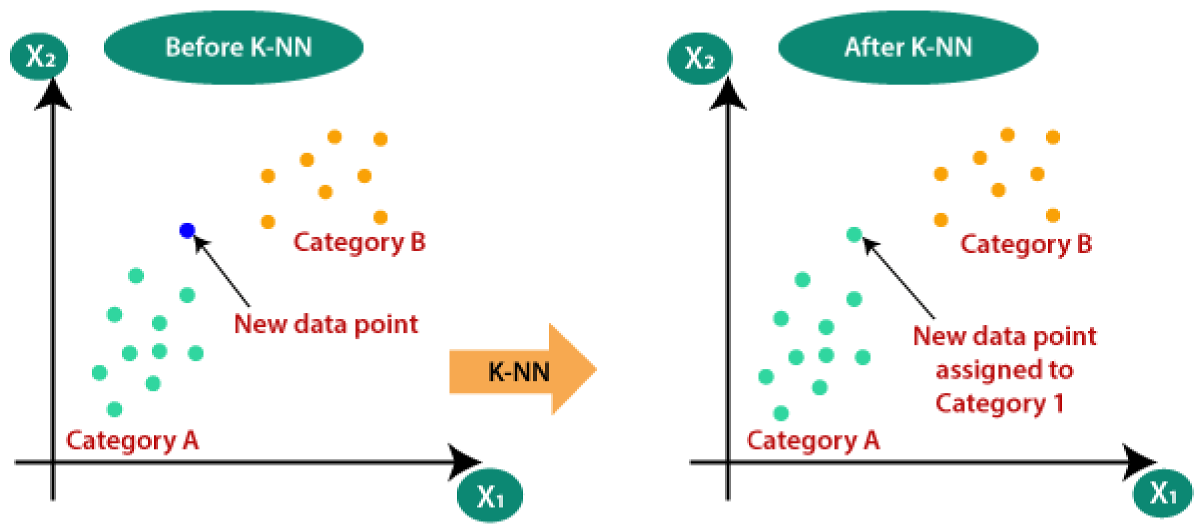
K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



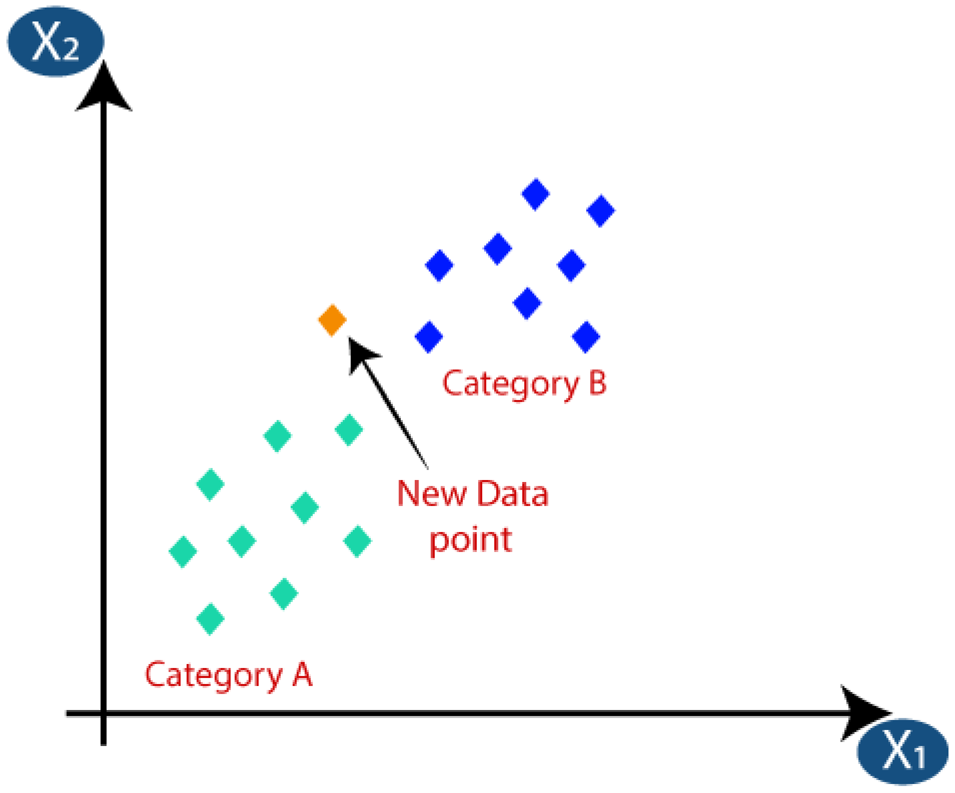
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

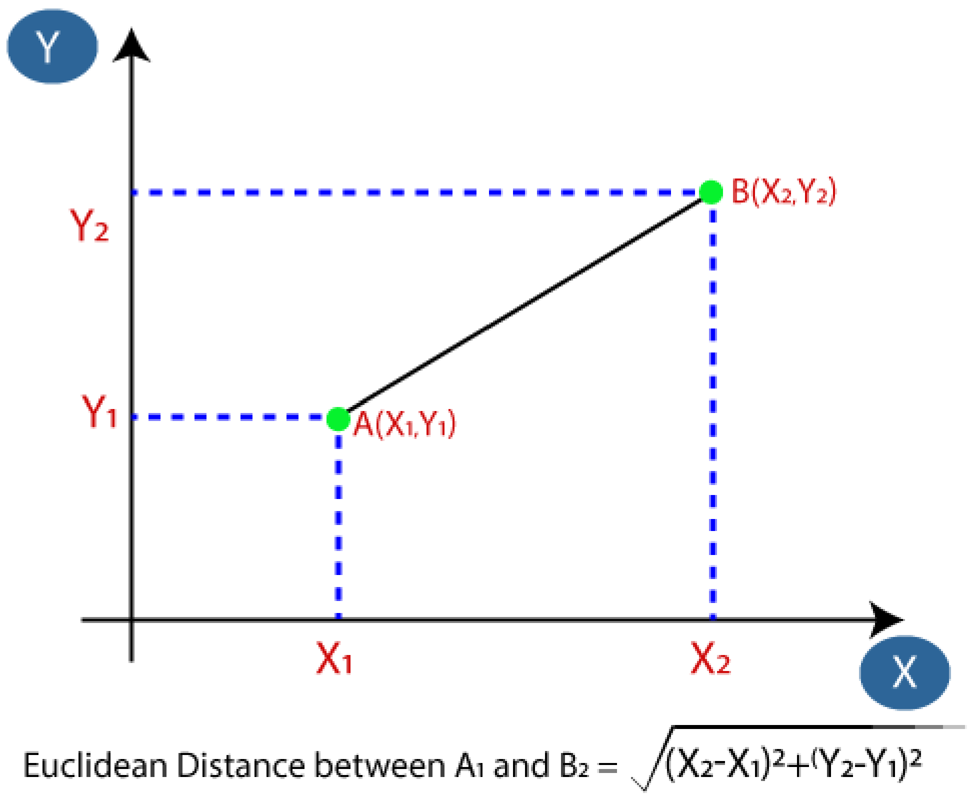
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

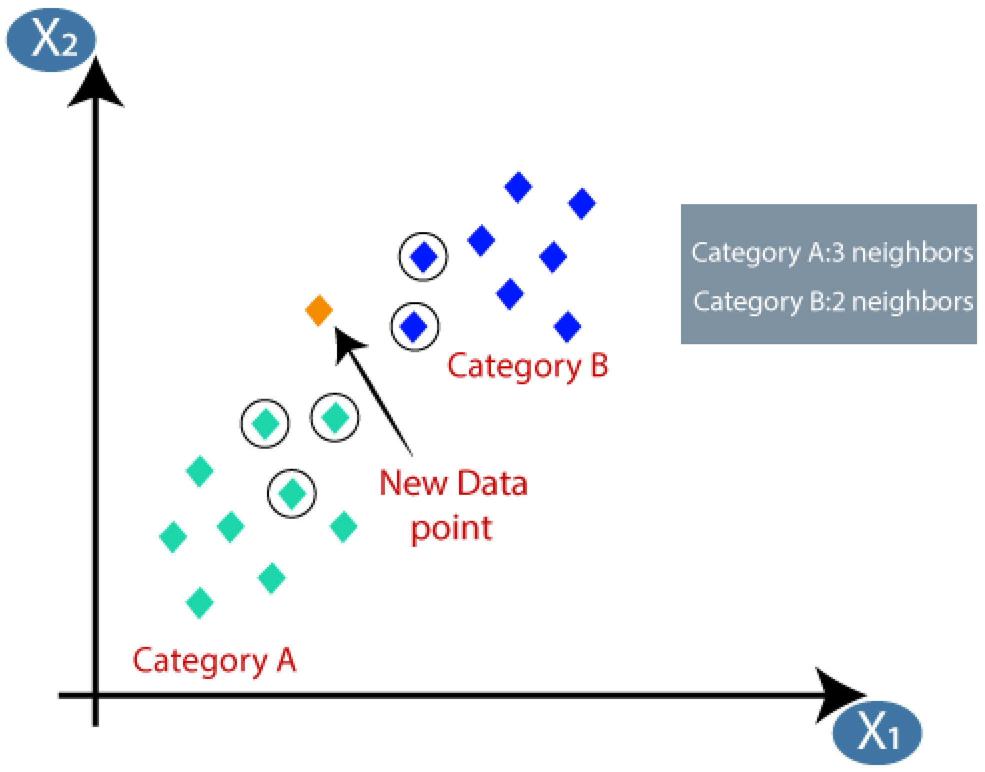
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:


Y
X
Y_2
Y_1
X_1
X_2
$B(X_2, Y_2)$
$A(X_1, Y_1)$
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.
- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters

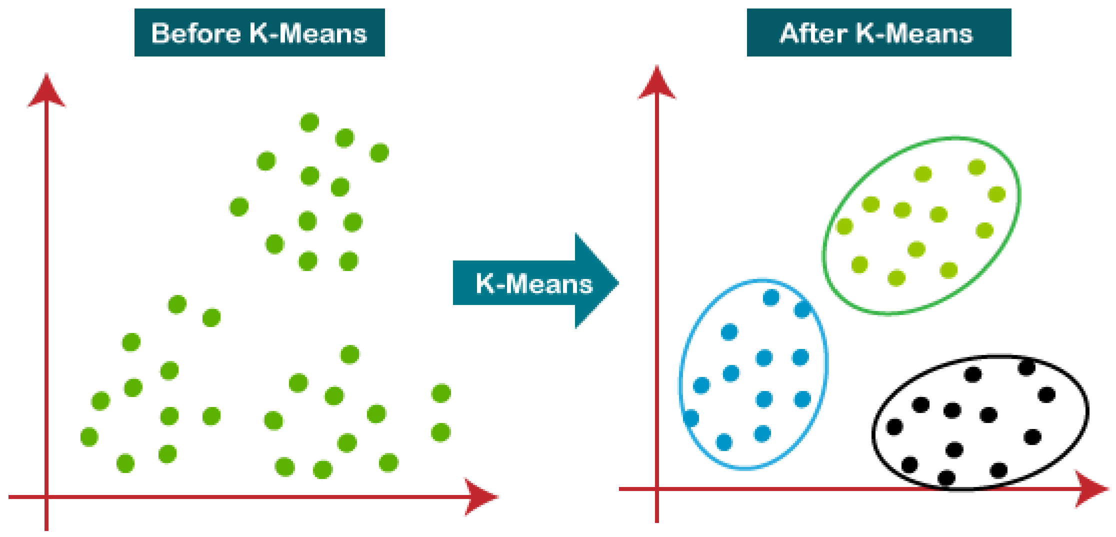
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

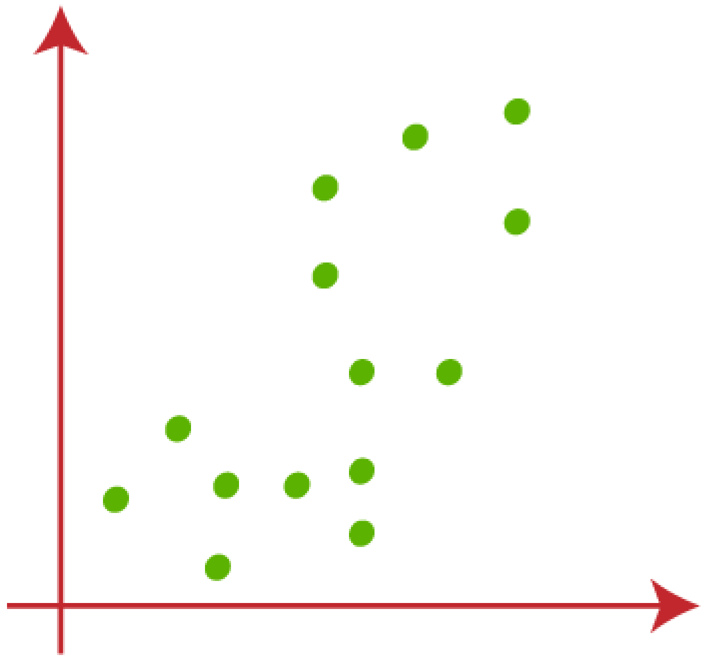
Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

- **Step-4:** Calculate the variance and place a new centroid of each cluster.

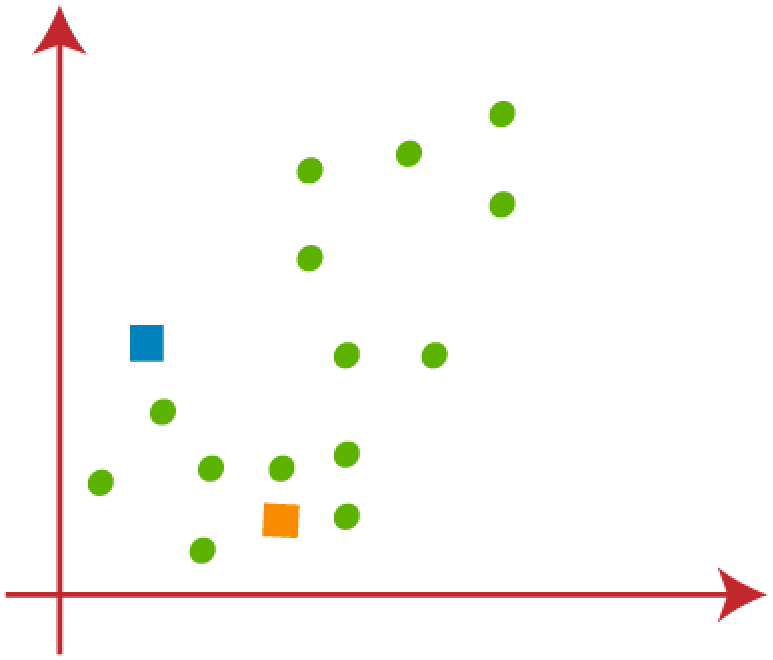
- **Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
- **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
- **Step-7**: The model is ready.
- Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



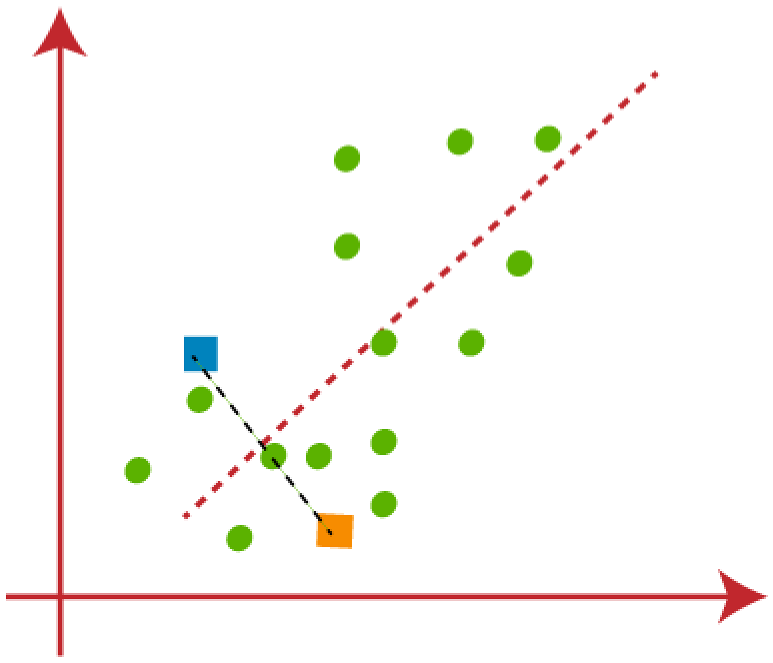
-

- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k

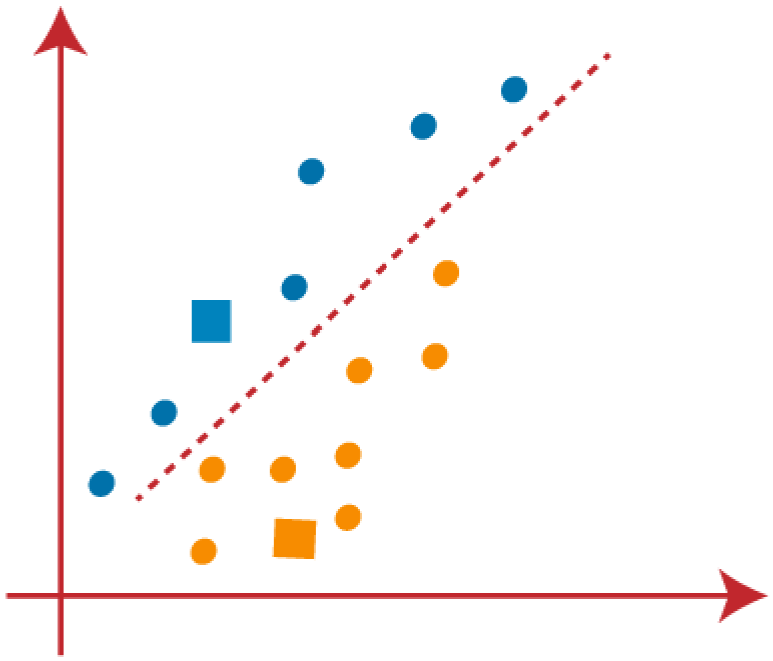
points, which are not the part of our dataset. Consider the below image:



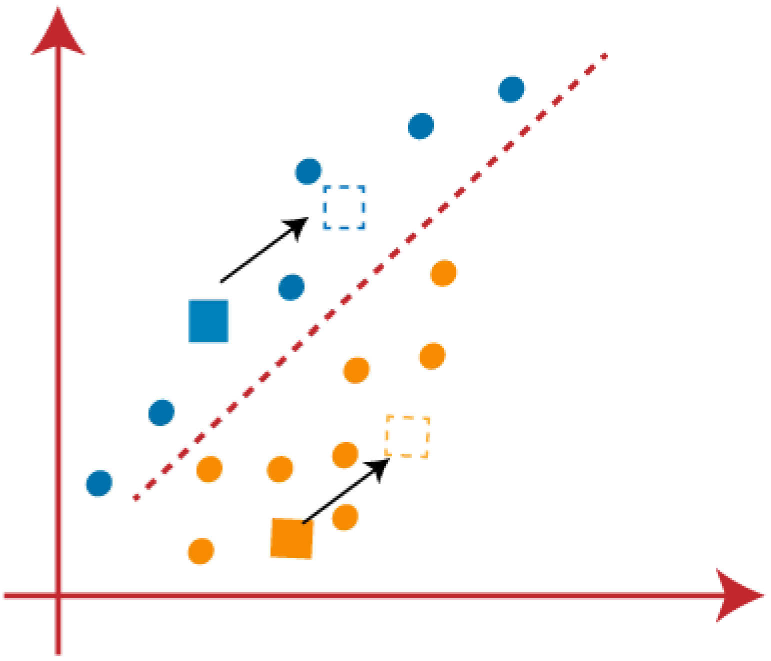
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



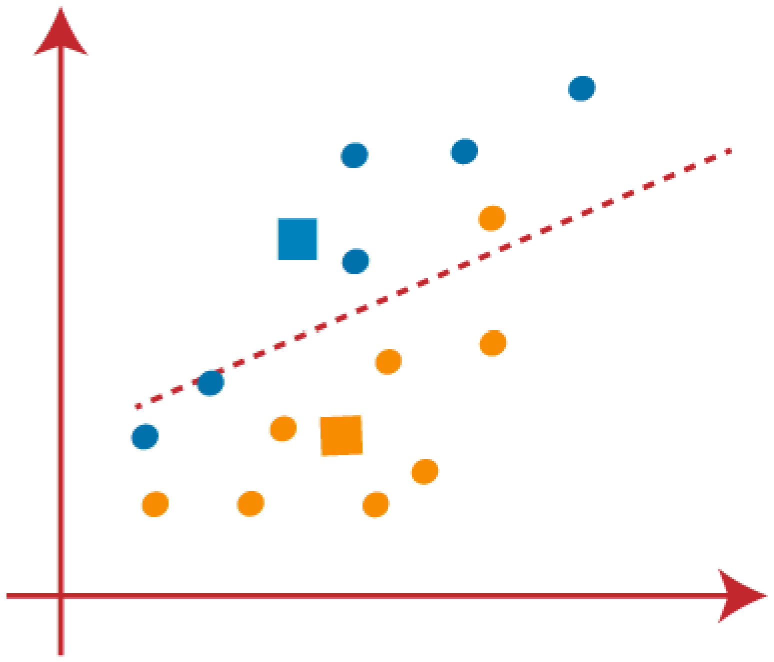
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



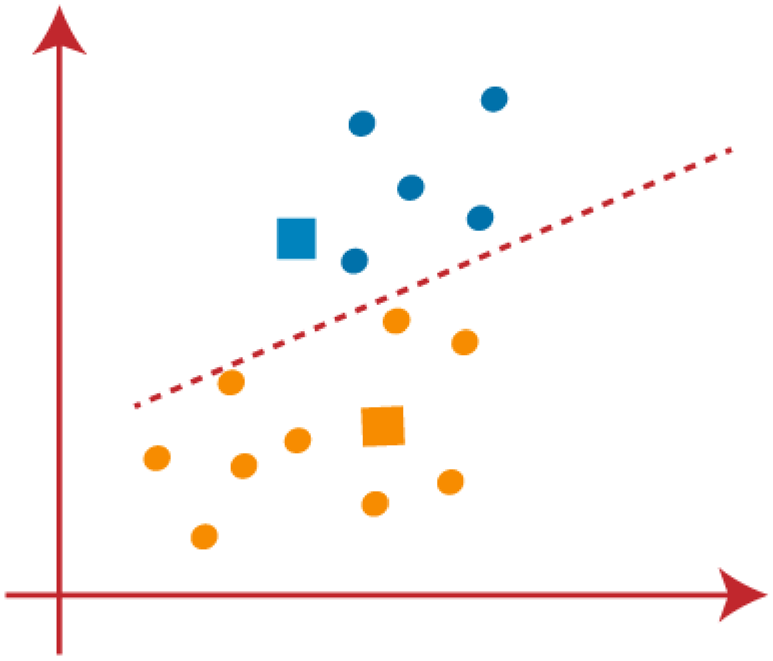
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



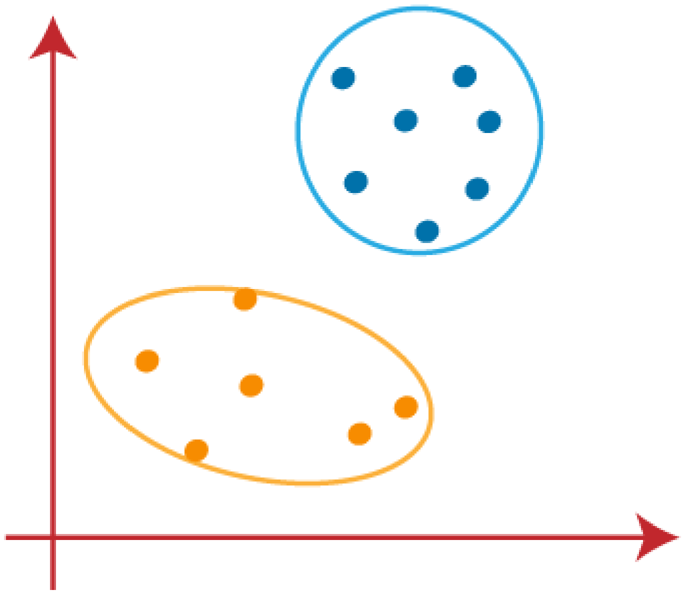
- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



- From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.



-



-

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

Why hierarchical clustering?

As we already have other clustering algorithms such as **K-Means Clustering**, then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

Agglomerative Hierarchical clustering

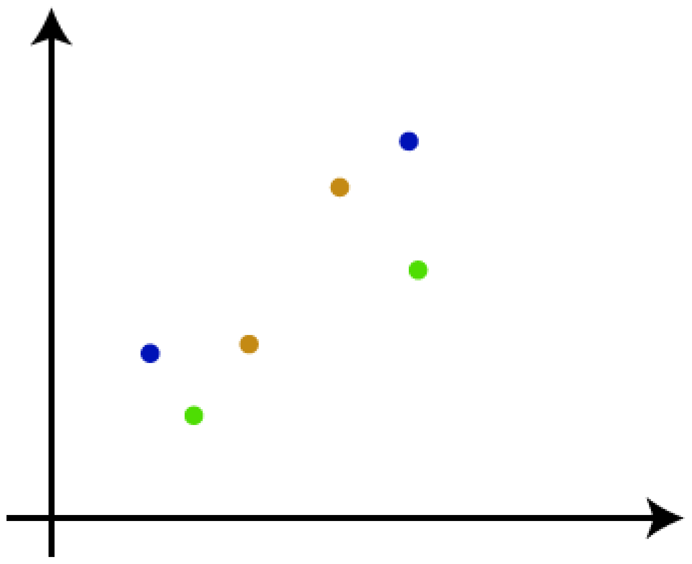
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

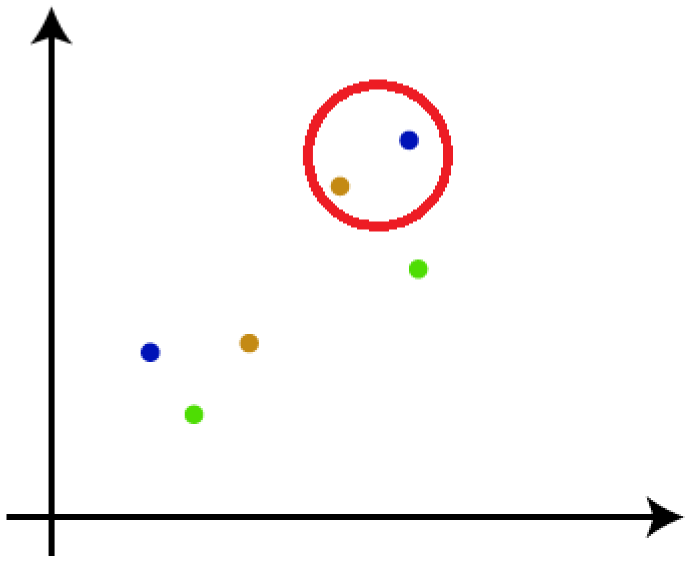
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

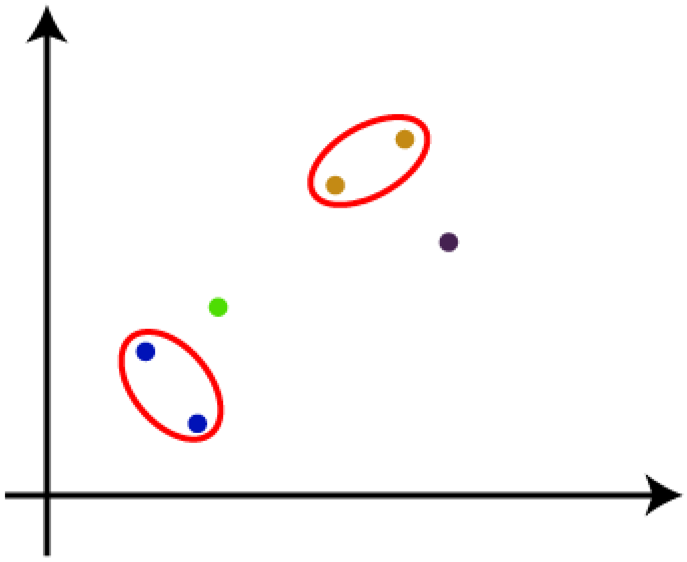
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



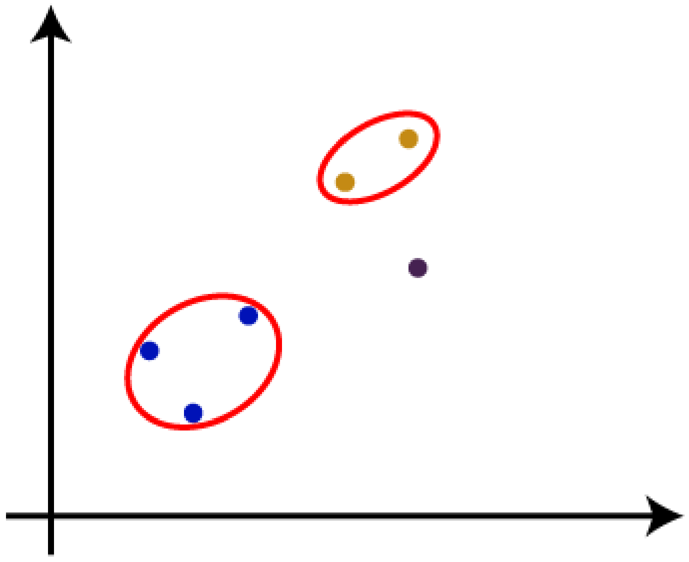
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

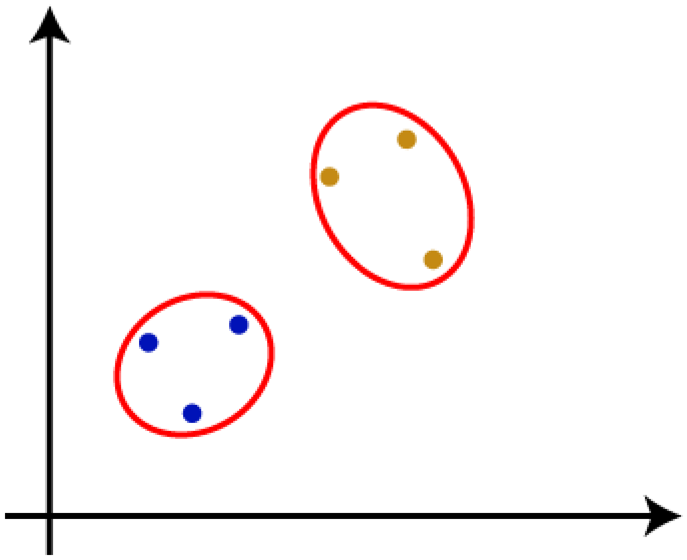


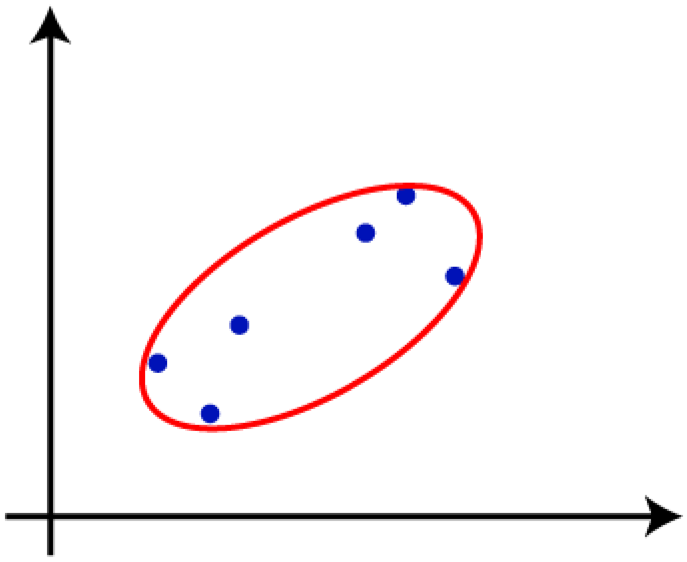
- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:





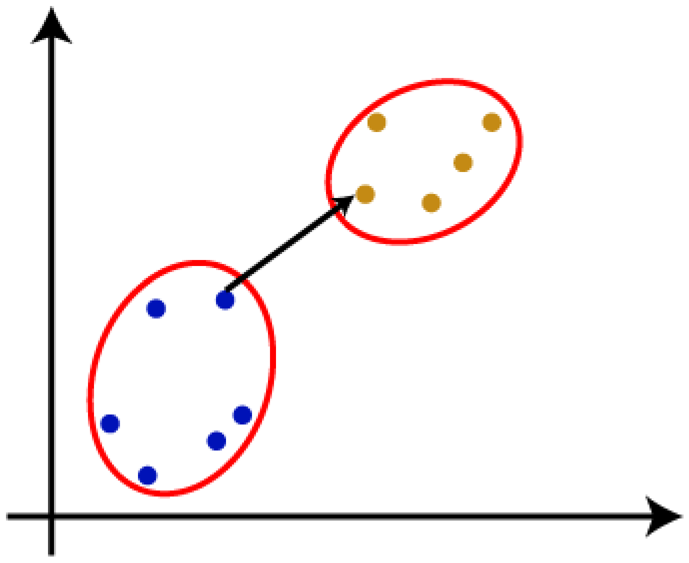


- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

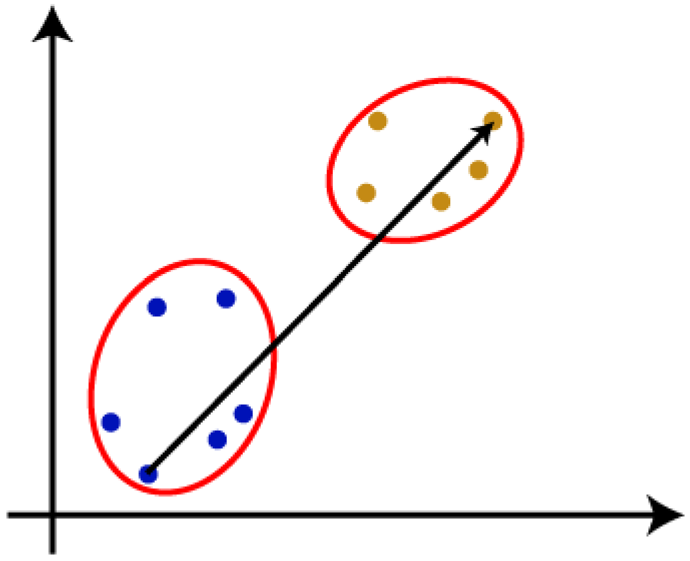
Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:

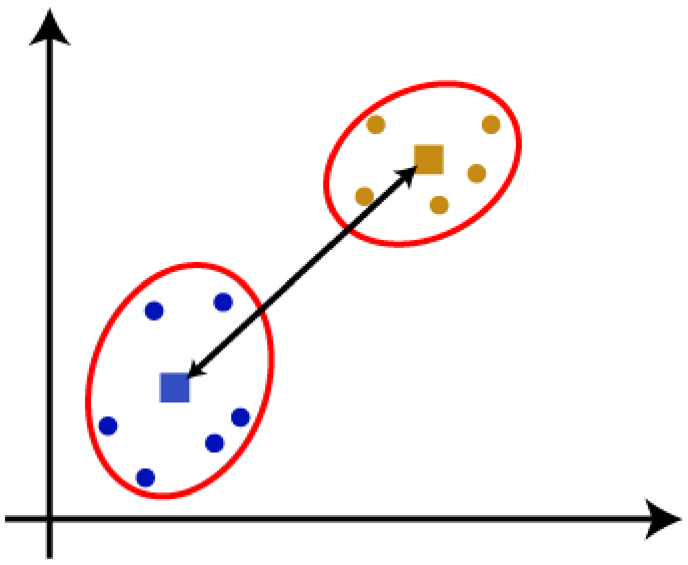


2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:

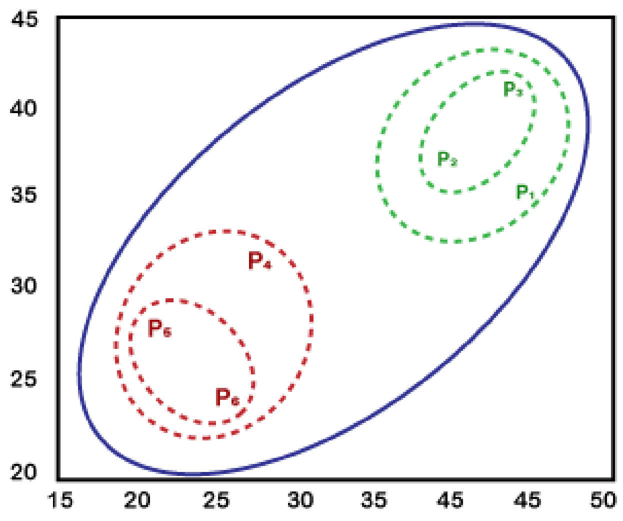


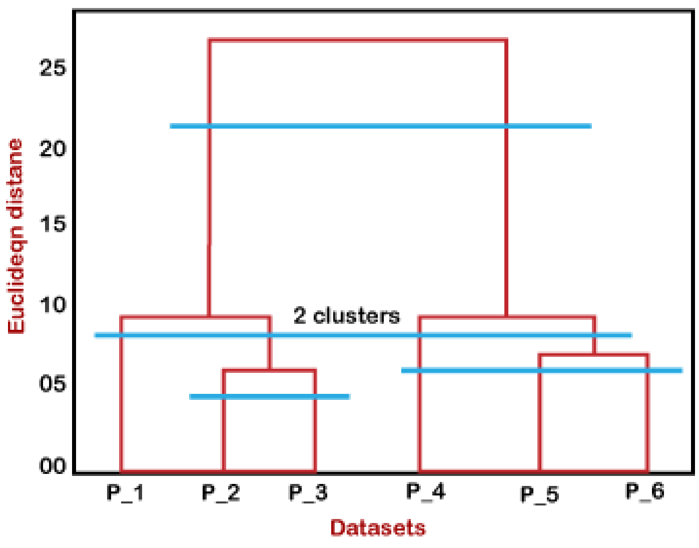
From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

Woking of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:





In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as ***"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."***

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

The clustering technique is commonly used for **statistical data analysis.**

> Note: Clustering is somewhere similar to the classification algorithm, but the difference is the type of dataset that we are using. In classification, we work with the labeled data set, whereas in clustering, we work with the unlabelled dataset.

Example: Let's understand the clustering technique with the real-world example of Mall: When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

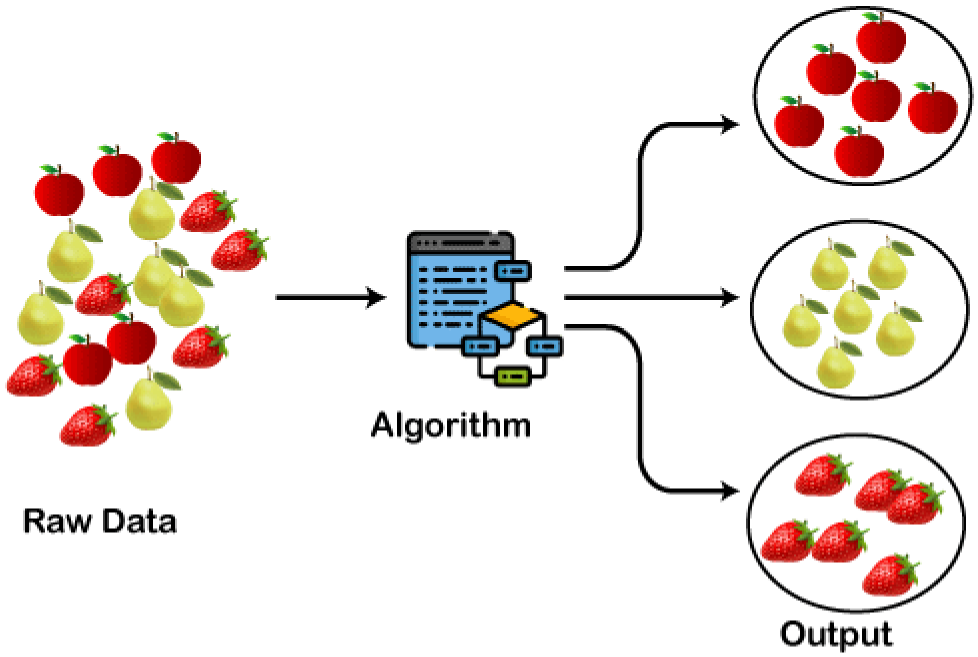
The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis
- Social network analysis

- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation system to provide the recommendations as per the past search of products. **Netflix** also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods

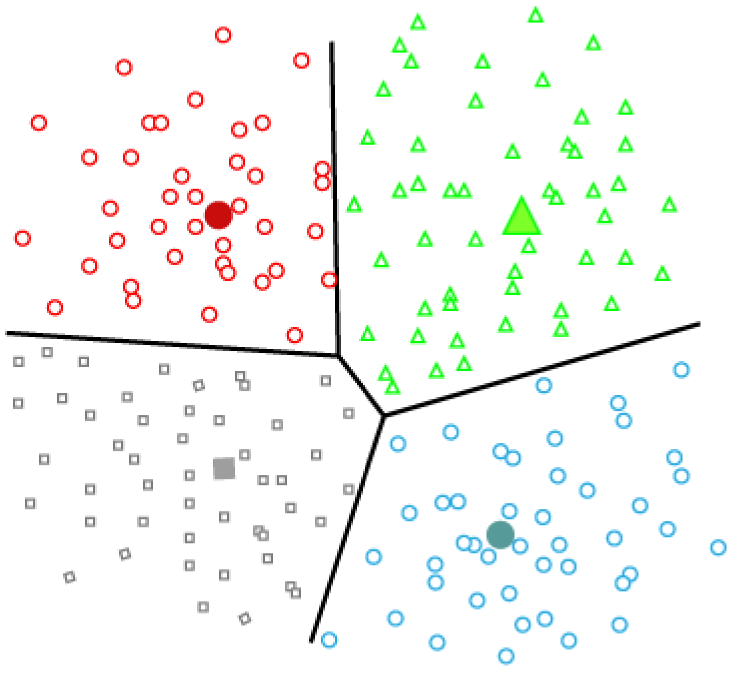
The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. **Partitioning Clustering**
2. **Density-Based Clustering**
3. **Distribution Model-Based Clustering**
4. **Hierarchical Clustering**
5. **Fuzzy Clustering**

Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

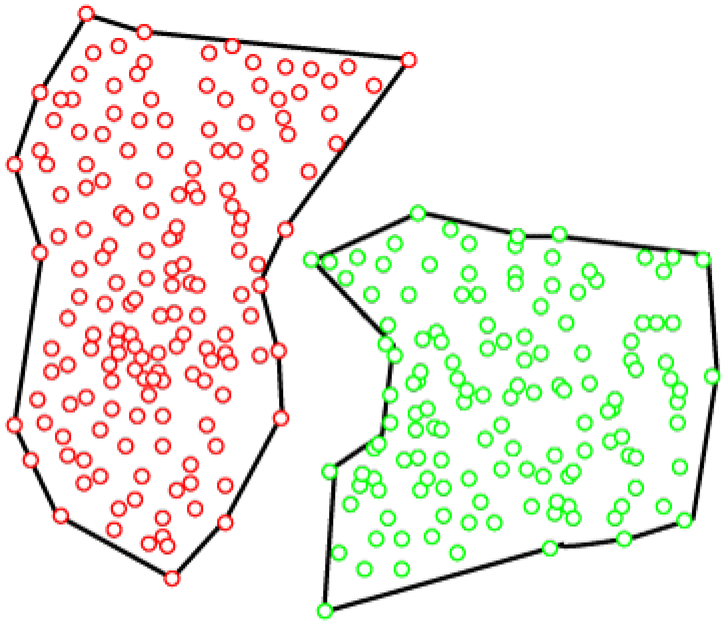
In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

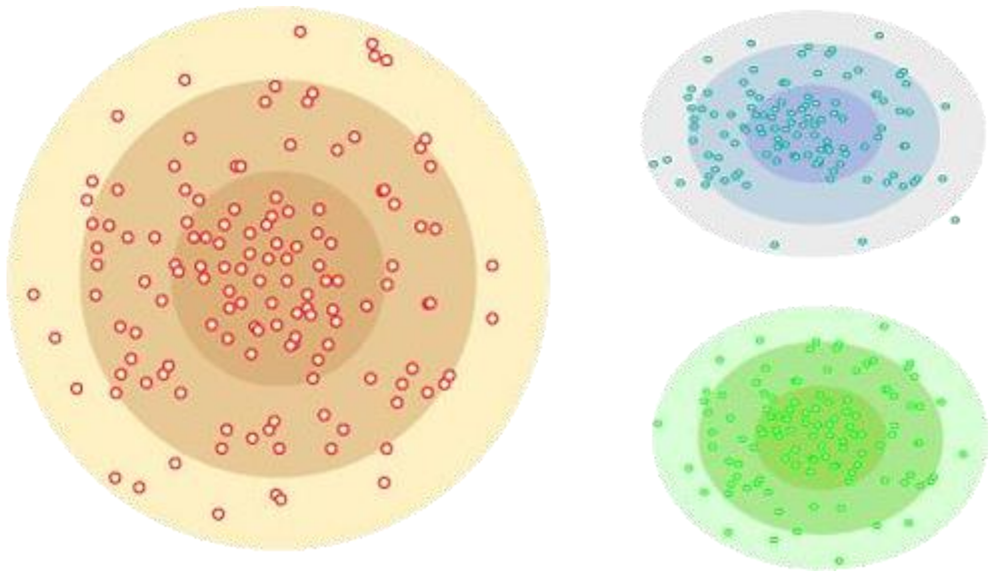
These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Distribution Model-Based Clustering

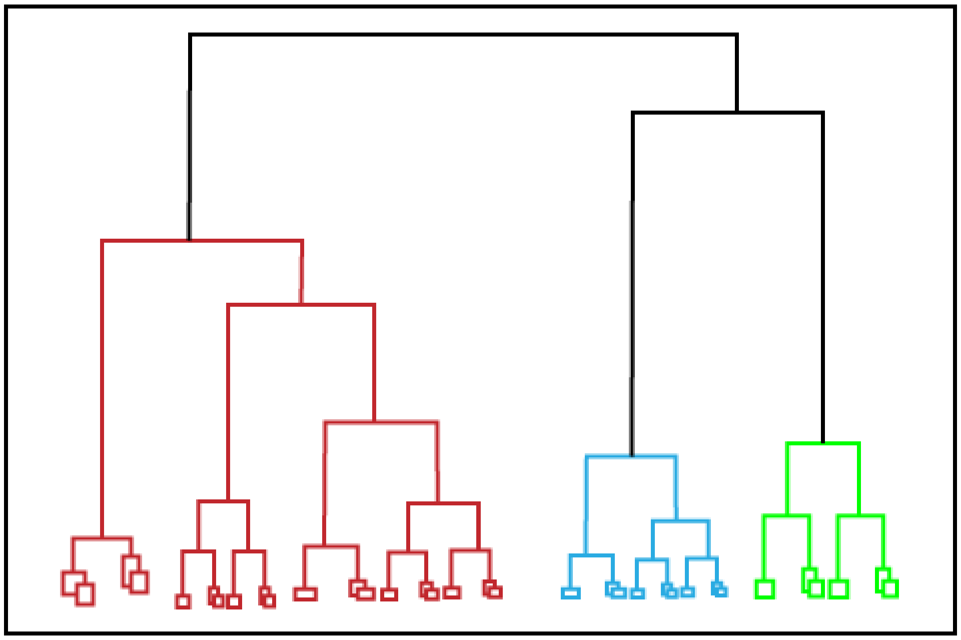
In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly **Gaussian Distribution**.

The example of this type is the **Expectation-Maximization Clustering algorithm** that uses Gaussian Mixture Models (GMM).



Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the **Agglomerative Hierarchical algorithm**.



Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. **Fuzzy C-means algorithm** is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Clustering Algorithms

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. **K-Means algorithm:** The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of **O(n).**
2. **Mean-shift algorithm:** Mean-shift algorithm tries to find the dense areas in the smooth density of data

points. It is an example of a centroid-based model, that works on updating the candidates for centroid to be the center of the points within a given region.

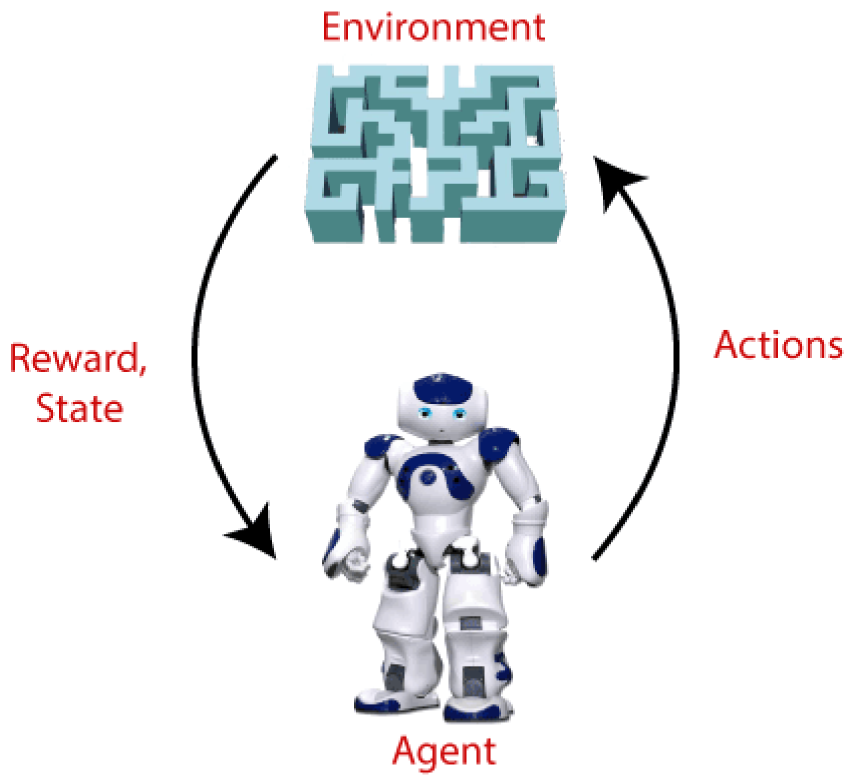
3. **DBSCAN Algorithm:** It stands **for Density-Based Spatial Clustering of Applications with Noise**. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.
4. **Expectation-Maximization Clustering using GMM:** This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.
5. **Agglomerative Hierarchical algorithm:** The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.
6. **Affinity Propagation:** It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has $O(N^2T)$ time complexity, which is the main drawback of this algorithm.

Applications of Clustering

Below are some commonly known applications of clustering technique in Machine Learning:

- **In Identification of Cancer Cells:** The clustering algorithms are widely used for the identification of cancerous cells. It divides the cancerous and non-cancerous data sets into different groups.
- **In Search Engines:** Search engines also work on the clustering technique. The search result appears based on the closest object to the search query. It does it by grouping similar data objects in one group that is far from the other dissimilar objects. The accurate result of a query depends on the quality of the clustering algorithm used.
- **Customer Segmentation:** It is used in market research to segment the customers based on their choice and preferences.
- **In Biology:** It is used in the biology stream to classify different species of plants and animals using the image recognition technique.
- **In Land Use:** The clustering technique is used in identifying the area of similar lands use in the GIS database. This can be very useful to find that for what purpose the particular land should be used, that means for which purpose it is more suitable.

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."*** How a Robotic dog learns the movement of his arms is an example of Reinforcement learning.
- It is a core part of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human intervention.
- **Example:** Suppose there is an AI agent present within a maze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.
- The agent continues doing these three things (**take action, change state/remain in the same state, and get feedback**), and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Key Features of Reinforcement Learning

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the hit and trial process.
- The agent takes the next action and changes states according to the feedback of the previous action.
- The agent may get a delayed reward.

- The environment is stochastic, and the agent needs to explore it to reach to get the maximum positive rewards.

Approaches to implement Reinforcement Learning

There are mainly three ways to implement reinforcement-learning in ML, which are:

1. **Value-based:**
   The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
2. **Policy-based:**
   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.
   The policy-based approach has mainly two types of policy:
   - **Deterministic:** The same action is produced by the policy (π) at any state.
   - **Stochastic:** In this policy, probability determines the produced action.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

1) Policy: A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

For deterministic policy: a = π(s)
For stochastic policy: π(a | s) = P*At =a | St = s]

2) Reward Signal: The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

3) Value Function: The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.

4) Model: The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

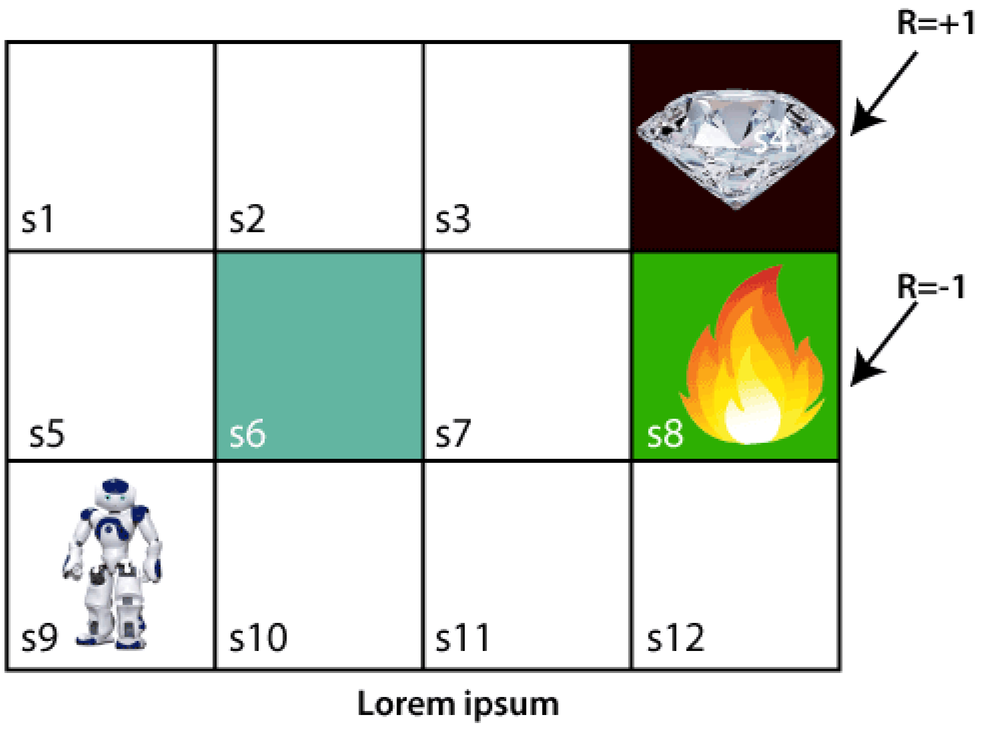
The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

How does Reinforcement Learning Work?

To understand the working process of the RL, we need to consider two main things:

- **Environment:** It can be anything such as a room, maze, football ground, etc.
- **Agent:** An intelligent agent such as AI robot.

Let's take an example of a maze environment that the agent needs to explore. Consider the below image:

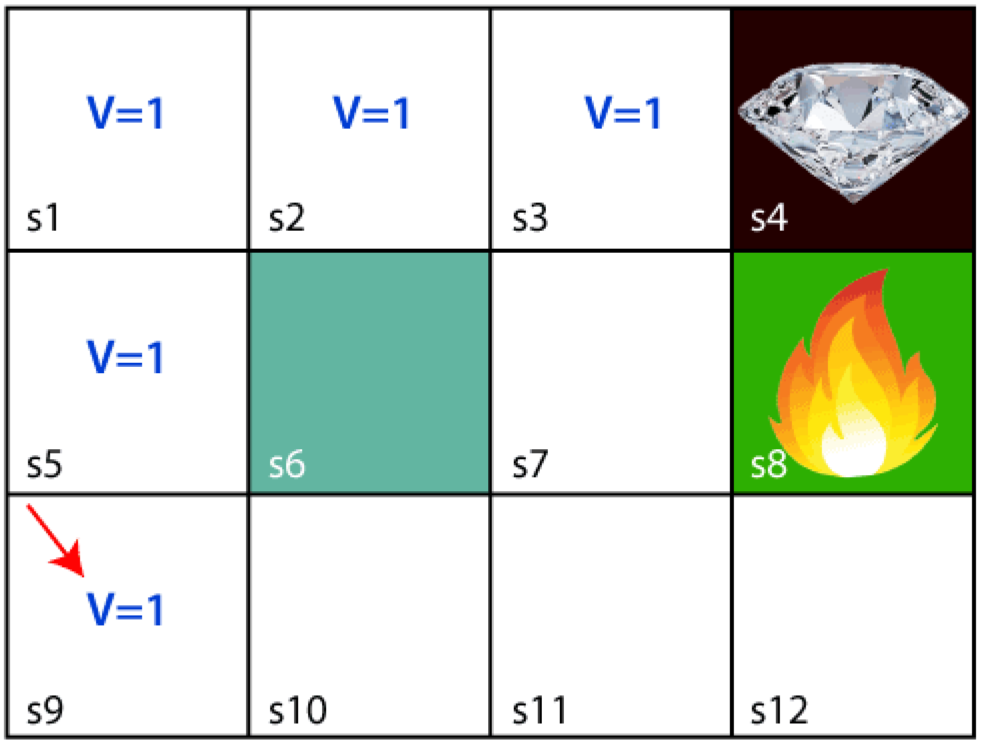


In the above image, the agent is at the very first block of the maze. The maze is consisting of an S_6 block, which is a **wall**, S_8 a **fire pit**, and S_4 a **diamond block.**

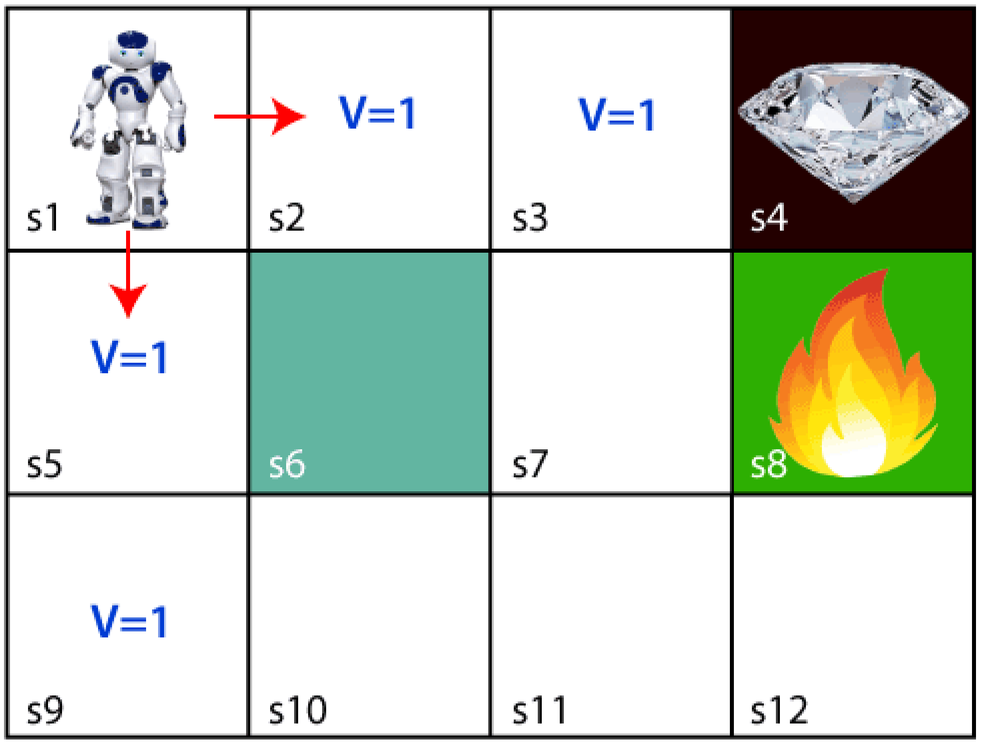
The agent cannot cross the S_6 block, as it is a solid wall. If the agent reaches the S_4 block, then get the **+1 reward;** if it reaches the fire pit, then gets **-1 reward point**. It can take four actions**: move up, move down, move left, and move right.**

The agent can take any path to reach to the final point, but he needs to make it in possible fewer steps. Suppose the agent considers the path **S9-S5-S1-S2-S3**, so he will get the +1-reward point.

The agent will try to remember the preceding steps that it has taken to reach the final step. To memorize the steps, it assigns 1 value to each previous step. Consider the below step:



Now, the agent has successfully stored the previous steps assigning the 1 value to each previous block. But what will the agent do if he starts moving from the block, which has 1 value block on both sides? Consider the below diagram:



It will be a difficult condition for the agent whether he should go up or down as each block has the same value. So, the above approach is not suitable for the agent to reach the destination. Hence to solve the problem, we will use the **Bellman equation**, which is the main concept behind reinforcement learning.

The Bellman Equation

The Bellman equation was introduced by the Mathematician **Richard Ernest Bellman in the year 1953**, and hence it is called as a Bellman equation. It is associated with dynamic programming and used to calculate the values of a decision problem at a certain point by including the values of previous states.

It is a way of calculating the value functions in dynamic programming or environment that leads to modern reinforcement learning.

The key-elements used in Bellman equations are:

- ction performed by the agent is referred to as "a"
- State occurred by performing the action is "s."
- The reward/feedback obtained for each good and bad action is "R."
- A discount factor is Gamma "γ."

The Bellman equation can be written as:

1. V(s) = max [R(s,a) + γV(s`)+

Where,

V(s)= value calculated at a particular point.

R(s,a) = Reward at a particular state s by performing an action.

γ = Discount factor

V(s`) = The value at the previous state.

In the above equation, we are taking the max of the complete values because the agent tries to find the optimal solution always.

So now, using the Bellman equation, we will find value at each state of the given environment. We will start from the block, which is next to the target block.

For 1st block:

- ction performed by the agent is referred to as "a"

- State occurred by performing the action is "s."
- The reward/feedback obtained for each good and bad action is "R."
- A discount factor is Gamma "γ."

The Bellman equation can be written as:

1. V(s) = max [R(s,a) + γV(s`)+

Where,

V(s)= value calculated at a particular point.

R(s,a) = Reward at a particular state s by performing an action.

γ = Discount factor

V(s`) = The value at the previous state.

In the above equation, we are taking the max of the complete values because the agent tries to find the optimal solution always.

So now, using the Bellman equation, we will find value at each state of the given environment. We will start from the block, which is next to the target block.

For 1st block:

V(s3) = max *R(s,a) + γV(s`)+, here V(s')= 0 because there is no further state to move.

V(s3)= max[R(s,a)]=> V(s3)= max[1]=> **V(s3)= 1.**

For 2nd block:

V(s2) = max *R(s,a) + γV(s`)+, here γ= 0.9(lets), V(s')= 1, and R(s, a)= 0, because there is no reward at this state.

V(s2)= max[0.9(1)]=> V(s)= max[0.9]=> **V(s2) =0.9**

For 3rd block:

V(s1) = max *R(s,a) + γV(s`)+, here γ= 0.9(lets), V(s')= 0.9, and R(s, a)= 0, because there is no reward at this state also.

V(s1)= max[0.9(0.9)]=> V(s3)= max[0.81]=> **V(s1) =0.81**

For 4th block:

V(s5) = max *R(s,a) + γV(s`)+, here γ= 0.9(lets), V(s')= 0.81, and R(s, a)= 0, because there is no reward at this state also.

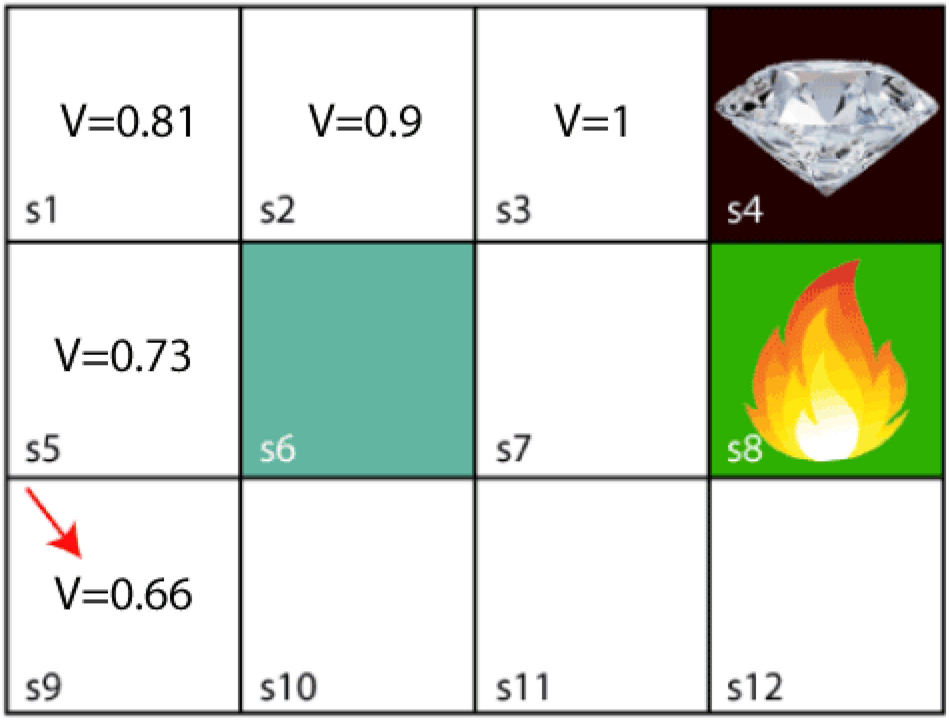
V(s5)= max[0.9(0.81)]=> V(s5)= max[0.81]=> **V(s5) =0.73**

For 5th block:

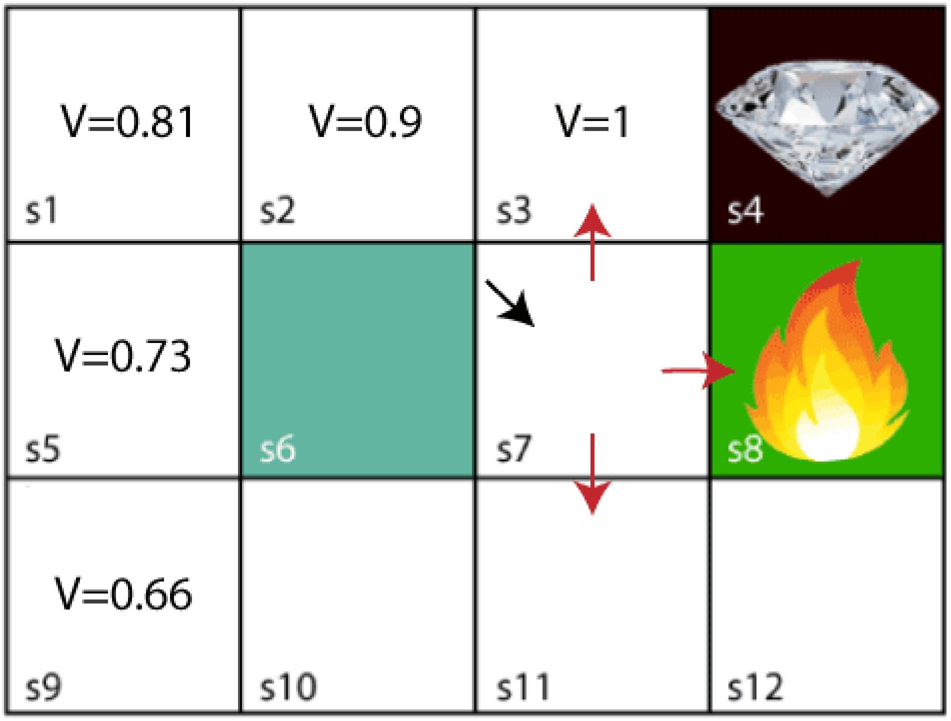
V(s9) = max *R(s,a) + γV(s`)+, here γ= 0.9(lets), V(s')= 0.73, and R(s, a)= 0, because there is no reward at this state also.

V(s9)= max[0.9(0.73)]=> V(s4)= max[0.81]=> **V(s4) =0.66**

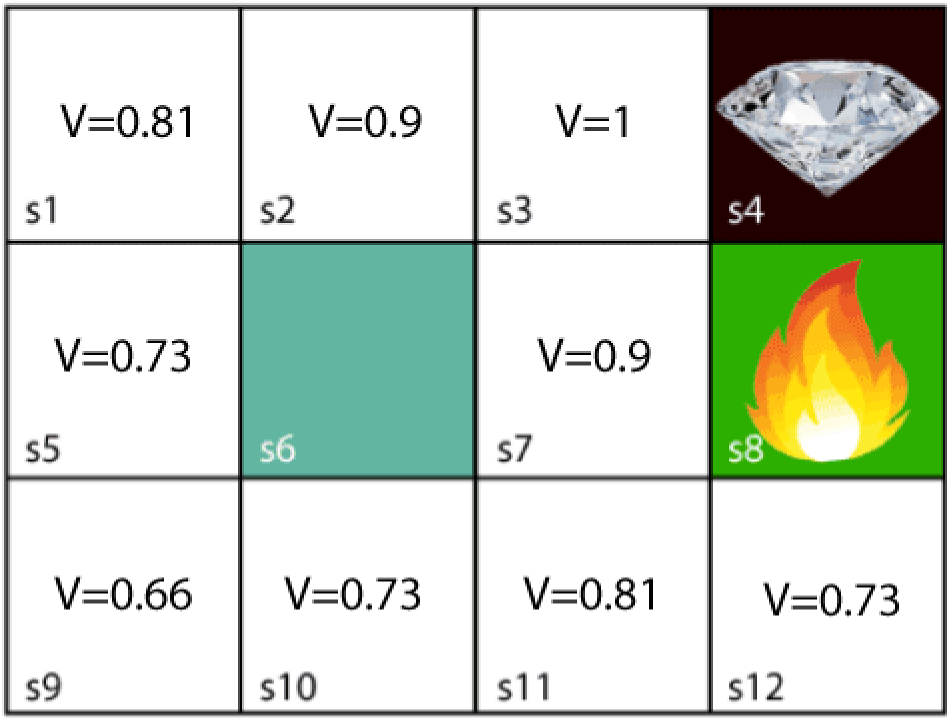
Consider the below image:



Now, we will move further to the 6th block, and here agent may change the route because it always tries to find the optimal path. So now, let's consider from the block next to the fire pit.



Now, the agent has three options to move; if he moves to the blue box, then he will feel a bump if he moves to the fire pit, then he will get the -1 reward. But here we are taking only positive rewards, so for this, he will move to upwards only. The complete block values will be calculated using this formula. Consider the below image:



Types of Reinforcement learning

There are mainly two types of reinforcement learning, which are:

- **Positive Reinforcement**
- **Negative Reinforcement**

Positive Reinforcement:

The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.

This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

Negative Reinforcement:

The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.

It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.

How to represent the agent state?

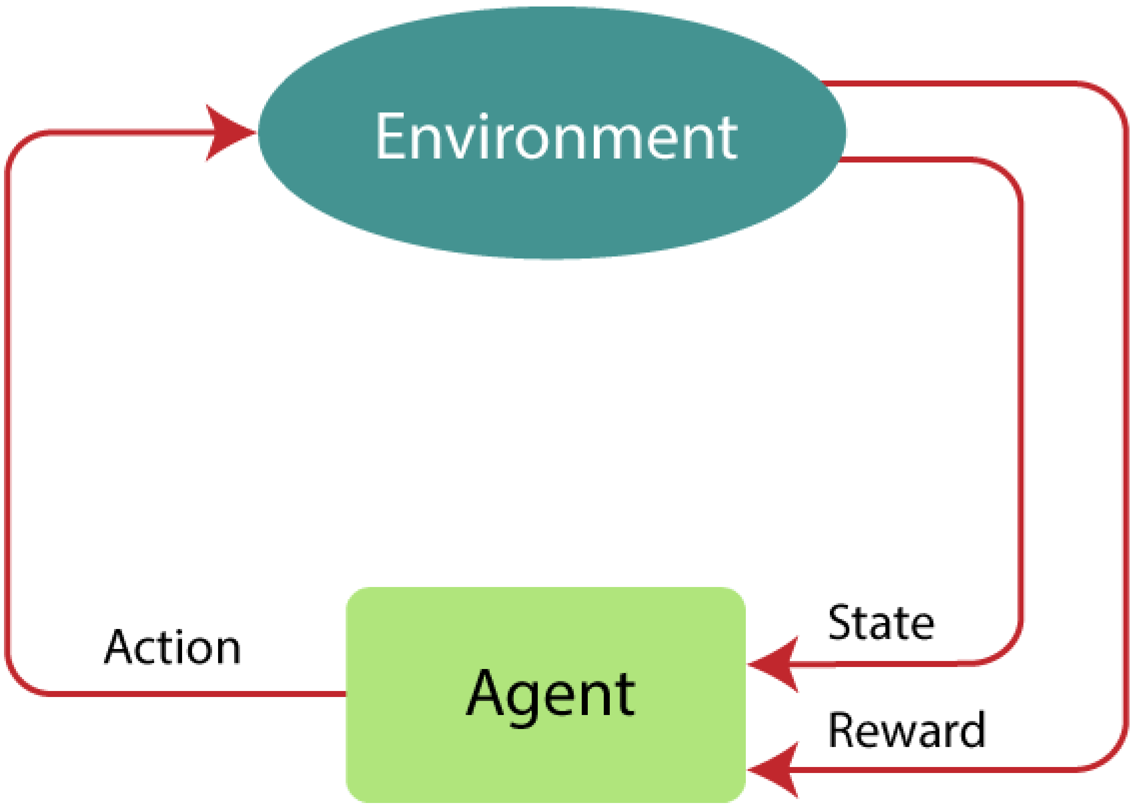
We can represent the agent state using the **Markov State** that contains all the required information from the history. The State St is Markov state if it follows the given condition:

$P[S_t+1 \mid S_t] = P[S_t+1 \mid S_1, \ldots\ldots, S_t]$

The Markov state follows the **Markov property**, which says that the future is independent of the past and can only be defined with the present. The RL works on fully observable environments, where the agent can observe the environment and act for the new state. The complete process is known as Markov Decision process, which is explained below:

Markov Decision Process

Markov Decision Process or MDP, is used to **formalize the reinforcement learning problems**. If the environment is completely observable, then its dynamic can be modeled as a **Markov Process**. In MDP, the agent constantly interacts with the environment and performs actions; at each action, the environment responds and generates a new state.



MDP is used to describe the environment for the RL, and almost all the RL problem can be formalized using MDP.

MDP contains a tuple of four elements (S, A, P_a, R_a):

- A set of finite States S
- A set of finite Actions A
- Rewards received after transitioning from state S to state S', due to action a.
- Probability P_a.

MDP uses **Markov property**, and to better understand the MDP, we need to learn about it.

Markov Property:

It says that ***"If the agent is present in the current state S1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states."***

Or, in other words, as per Markov Property, the current state transition does not depend on any past action or state. Hence, MDP is an RL problem that satisfies the Markov property. Such as in a **Chess game, the players only focus on the current state and do not need to remember past actions or states**.

Finite MDP:

A finite MDP is when there are finite states, finite rewards, and finite actions. In RL, we consider only the finite MDP.

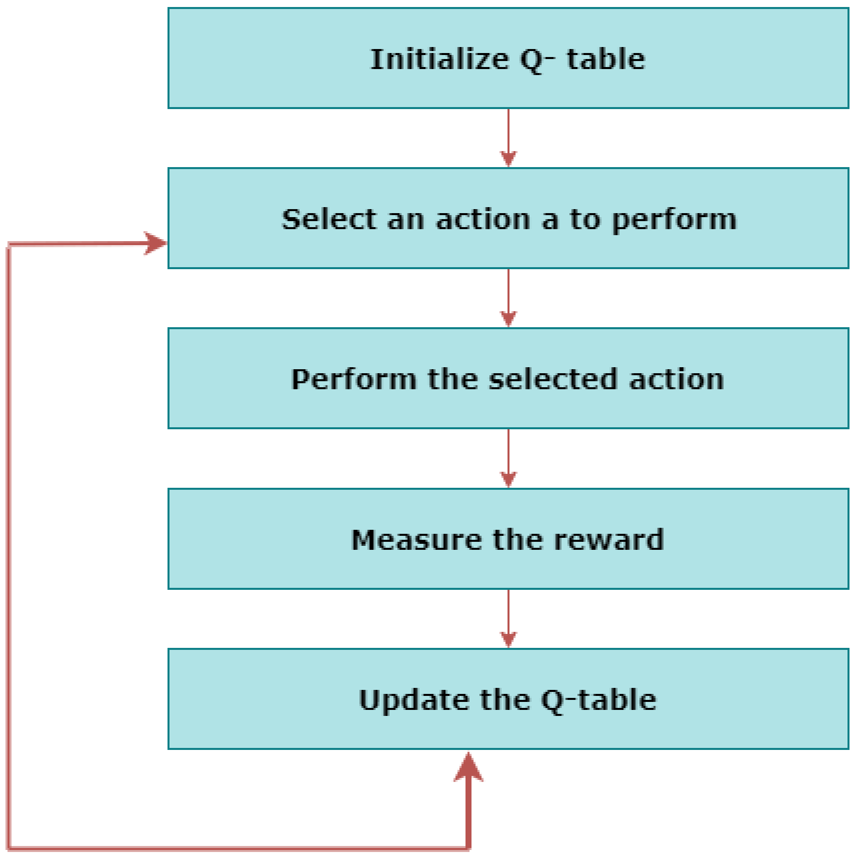
Markov Process:

Markov Process is a memoryless process with a sequence of random states S_1, S_2, , S_t that uses the Markov Property. Markov process is also known as Markov chain, which is a tuple (S, P) on state S and transition function P. These two components (S and P) can define the dynamics of the system.

Reinforcement Learning Algorithms

Reinforcement learning algorithms are mainly used in AI applications and gaming applications. The main used algorithms are:

- **Q-Learning:**
    - Q-learning is an **Off policy RL algorithm**, which is used for the temporal difference Learning. The temporal difference learning methods are the way of comparing temporally successive predictions.
    - It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."
    - The below flowchart explains the working of Q- learning:



- **State Action Reward State action (SARSA):**
  - SARSA stands for **State Action Reward State action**, which is an **on-policy** temporal difference learning method. The on-policy control method selects the action for each state while learning using a specific policy.
  - The goal of SARSA is to calculate the **Q π (s, a) for the selected current policy π and all pairs of (s-a).**
  - The main difference between Q-learning and SARSA algorithms is that **unlike Q-learning, the maximum reward for the next state is not required for updating the Q-value in the table.**
  - In SARSA, new action and reward are selected using the same policy, which has determined the original action.
  - The SARSA is named because it uses the quintuple **Q(s, a, r, s', a').**
  - Where, **s: original state**
  - **a: Original action**
- **r: reward observed while following the states s' and a': New state, action pair.**

- **Deep Q Neural Network (DQN):**
    - As the name suggests, DQN is a **Q-learning using Neural networks**.
    - For a big state space environment, it will be a challenging and complex task to define and update a Q-table.
    - To solve such an issue, we can use a DQN algorithm. Where, instead of defining a Q-table, neural network approximates the Q-values for each action and state.

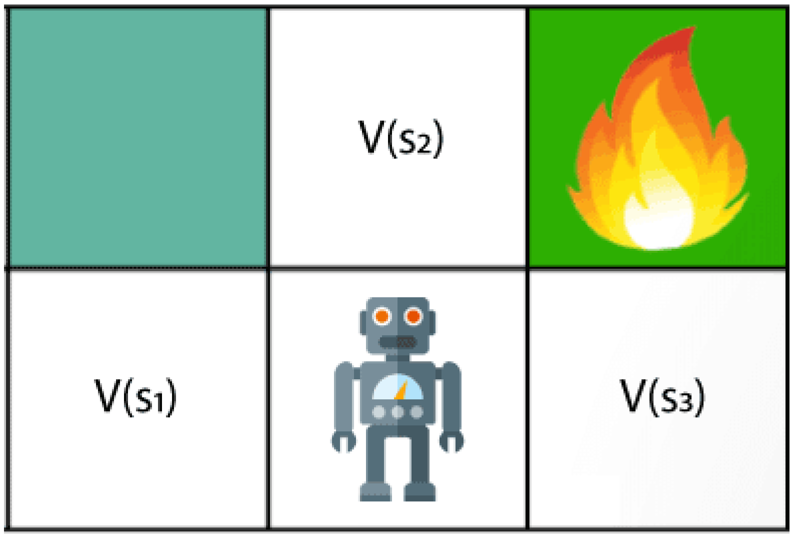
Now, we will expand the Q-learning.

Q-Learning Explanation:

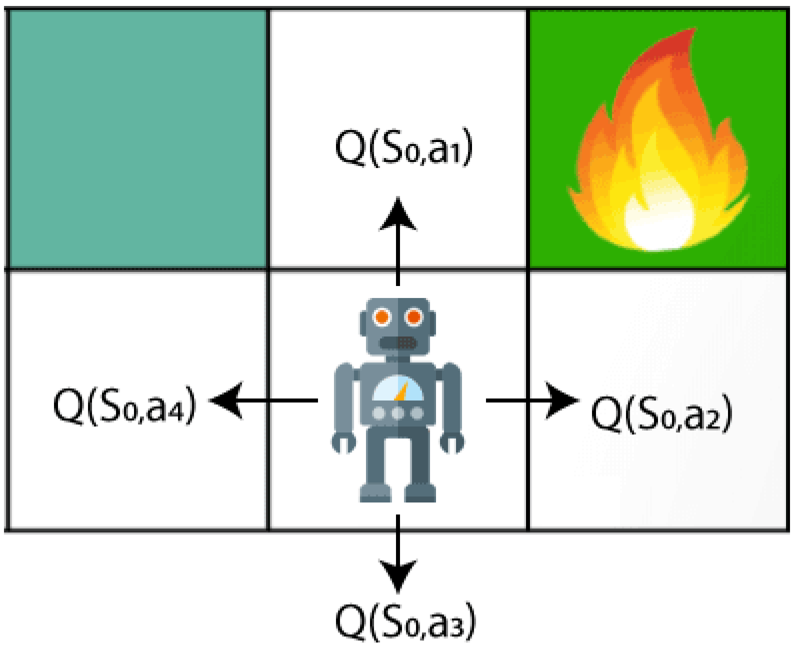
- Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation.
- **The main objective of Q-learning is to learn the policy which can inform the agent that what actions should be taken for maximizing the reward under what circumstances.**
- It is an **off-policy RL** that attempts to find the best action to take at a current state.
- The goal of the agent in Q-learning is to maximize the value of Q.
- The value of Q-learning can be derived from the Bellman equation. Consider the Bellman equation given below:

$$V(s) = \max [R(s,a) + \gamma \textstyle\sum_{s'} P(s,a,s')V(s`)]$$

In the equation, we have various components, including reward, discount factor (γ), probability, and end states s'. But there is no any Q-value is given so first consider the below image:



In the above image, we can see there is an agent who has three values options, V(s_1), V(s_2), V(s_3). As this is MDP, so agent only cares for the current state and the future state. The agent can go to any direction (Up, Left, or Right), so he needs to decide where to go for the optimal path. Here agent will take a move as per probability bases and changes the state. But if we want some exact moves, so for this, we need to make some changes in terms of Q-value. Consider the below image:



Q- represents the quality of the actions at each state. So instead of using a value at each state, we will use a pair of state and action, i.e., Q(s, a). Q-value specifies that which action is more lubricative than others, and according to the best Q-value, the agent takes his next move. The Bellman equation can be used for deriving the Q-value.

To perform any action, the agent will get a reward R(s, a), and also he will end up on a certain state, so the Q -value equation will be:

$$Q(S, a)= R(s, a)+ \gamma\sum_{s'} P(s, a, s')V(s`)$$

Hence, we can say that, ***V(s) = max [Q(s, a)]***

$$Q(S, a)= R(s, a)+ \gamma\sum_{s'}(P(s, a, s')\max Q(s',a`))$$

The above formula is used to estimate the Q-values in Q-Learning.

What is 'Q' in Q-learning?

The Q stands for **quality** in **Q-learning**, which means it specifies the quality of an action taken by the agent.

Q-table:

A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, i.e., [s, a], and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table.

The RL agent uses this Q-table as a reference table to select the best action based on the q-values.

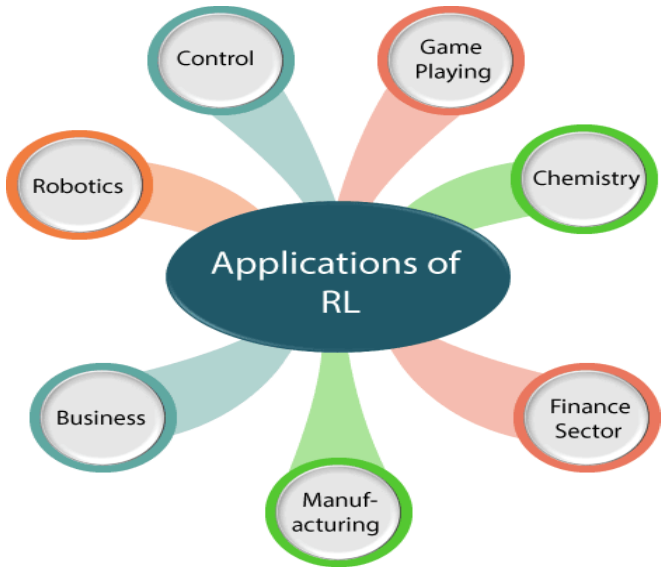
Difference between Reinforcement Learning and Supervised Learning

The Reinforcement Learning and Supervised Learning both are the part of machine learning, but both types of learnings are far opposite to each other. The RL agents interact with the environment, explore it, take action, and get rewarded. Whereas supervised learning algorithms learn from the labeled dataset and, on the basis of the training, predict the output.

The difference table between RL and Supervised learning is given below:

Reinforcement Learning	**Supervised Learning**
RL works by interacting with the environment.	Supervised learning works on the existing dataset.
The RL algorithm works like the human brain works when making some decisions.	Supervised Learning works as when a human learns things in the supervision of a guide.
There is no labeled dataset is present	The labeled dataset is present.
No previous training is provided to the learning agent.	Training is provided to the algorithm so that it can predict the output.
RL helps to take decisions sequentially.	In Supervised learning, decisions are made when input is given.

Reinforcement Learning Applications



1. **Robotics:**
   a. RL is used in **Robot navigation, Robo-soccer, walking, juggling**, etc.
2. **Control:**

. RL can be used for **adaptive control** such as Factory processes, admission control in telecommunication, and Helicopter pilot is an example of reinforcement learning.

3. **Game Playing:**

. RL can be used in **Game playing** such as tic-tac-toe, chess, etc.

4. **Chemistry:**

. RL can be used for optimizing the chemical reactions.

5. **Business:**

. RL is now used for business strategy planning.

6. **Manufacturing:**

In various automobile manufacturing companies, the robots use deep reinforcement learning to pick goods and put them in some containers.

7. **Finance Sector:**

The RL is currently used in the finance sector for evaluating trading strategies.

Conclusion:

From the above discussion, we can say that Reinforcement Learning is one of the most interesting and useful parts of Machine learning. In RL, the agent explores the environment by exploring it without any human intervention. It is the main learning algorithm that is used in Artificial Intelligence. But there are some cases where it should not be used, such as if you have enough data to solve the problem, then other ML algorithms can be used more efficiently. The main issue with the RL algorithm is that some of the parameters may affect the speed of the learning, such as delayed feedback.