STUDY MATERIAL

On

Artificial intelligence and Machine Learning

(For 6th Semester CSE)

Prepared by :

SMARANIKA MAHARANA(CSE DEPT),
GIET (POLY),JAGATPUR,CUTTACK

1. Introduction to AI

1.1Definition of AI, History of AI

Definition of AI:-

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.

History of AI:-

Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths. Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.

Maturation of Artificial Intelligence (1943-1952)

Year 1943: The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.

Year 1949: Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.

Year 1950: The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

The birth of Artificial Intelligence (1952-1956)

Year 1955: An Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.

Year 1956: The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden years-Early enthusiasm (1956-1974)

Year 1966: The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizenbaum created the first chatbot in 1966, which was named as ELIZA.

Year 1972: The first intelligent humanoid robot was built in Japan which was named as WABOT-1.

The first AI winter (1974-1980)

The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.

During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

Year 1980: After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.

In the Year 1980, the first national conference of the American Association of Artificial Intelligence **was held at Stanford University**.

A boom of AI (1980-1987)

Year 1980: After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.

In the Year 1980, the first national conference of the American Association of Artificial Intelligence **was held at Stanford University**.

The second AI winter (1987-1993)

The duration between the years 1987 to 1993 was the second AI Winter duration.

Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

The emergence of intelligent agents (1993-2011)

Year 1997: In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.

Year 2002: for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.

Year 2006: AI came in the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.

Deep learning, big data and artificial general intelligence (2011-present)

Year 2011: In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.

Year 2012: Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.

Year 2014: In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test."

Year 2018: The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.

Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

1.2 Goals and Applications of AI:-

Goals of Artificial Intelligence:-

AI can be achieved by reading the behavior of humans and using the results to develop intelligent systems. For example, they learn, make decisions and act in certain situations. Observing humans while problem-solving in simple tasks and using its results to develop intelligent systems.

The overall research goal of artificial intelligence is to create technology that allows computers and machines to work intelligently. The general problem of simulating (or creating) intelligence is broken down into sub-problems.

The symptoms described below receive the most attention. These include special traits or abilities that researchers expect an intelligent system to exhibit. Eric Sandwell emphasizes planning and learning that is relevant and applicable to the given situation.

Logic, problem-solving: Early researchers developed algorithms that simulate humans' step-by-step reasoning when solving puzzles or making logical deductions. By the late 1980s and 1990s, AI research had developed methods for dealing with uncertain or incomplete information, employing concepts from probability and economics. For difficult problems, algorithms can require enormous computational resources-most experience a "**combinatorial explosion**": the amount of memory or computer time needed for problems of a certain size becomes astronomical. The search for more efficient problem-solving algorithms is a high priority.

Knowledge representation: Knowledge representation and knowledge engineering are central to AI research. Many of the problems that machines are expected to solve will require extensive world knowledge. The things AI needs to represent are objects, properties, categories, and relationships between objects; situations, events, states, and times; Cause and Effect; Knowledge about knowledge (what other people know about what we know); and many other, less well-researched domains. A representation of "what exists" is an ontology: the set of **objects, relations**, concepts, and so on about which the machine knows. The most general is upper ontology, which attempts to provide a foundation for all other knowledge.

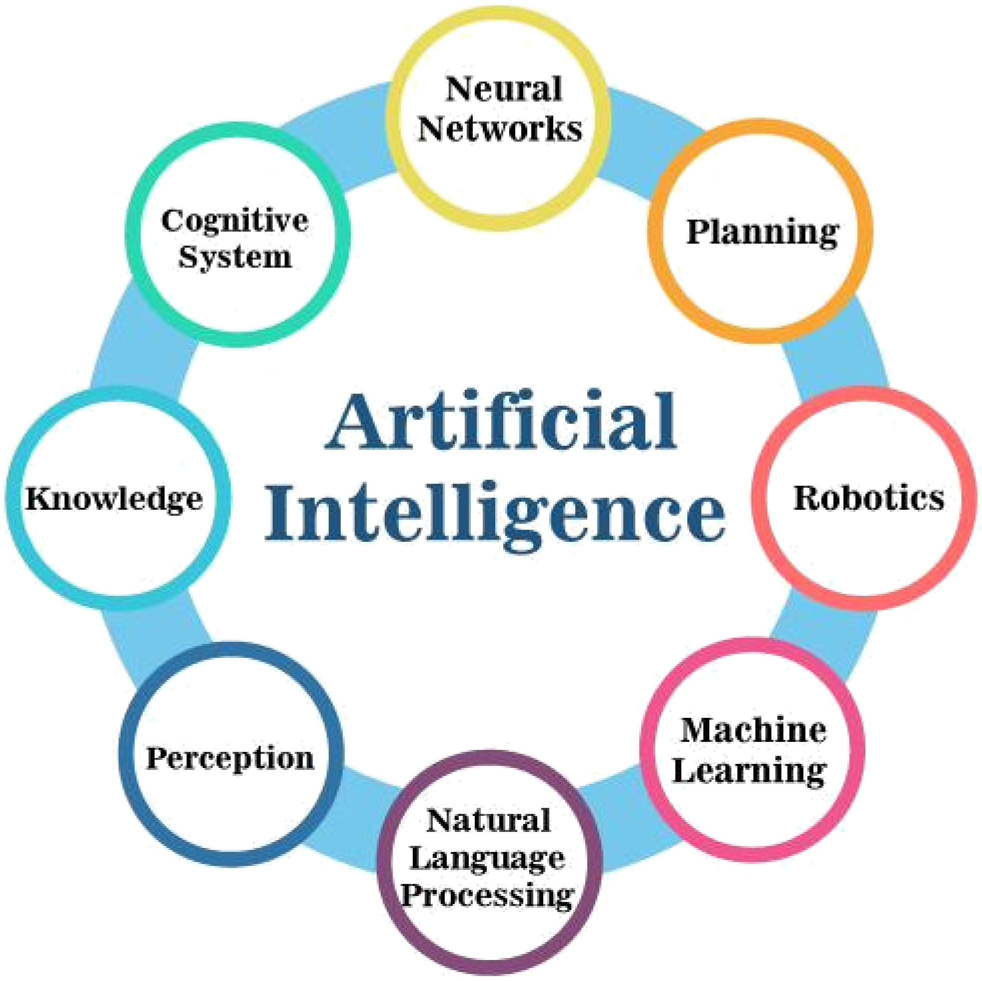
Planning: Intelligent agents must be able to set goals and achieve them. They need a way to envision the future - a representation of the state of the world and make predictions about how their actions will change it - and be able to make choices that maximize the utility (or "**value**") of the options available. 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, it requires that the agent reason under uncertainty. It calls for an agent to assess its environment, make predictions, evaluate its predictions, and adapt based on its assessment.

Learning: Machine learning, a fundamental concept of AI research since the field's inception, is the study of computer algorithms that automatically improve through experience. Unsupervised learning is the ability to find patterns in a stream of input. Supervised learning includes both classification and numerical regression. After seeing several examples of things from several categories, classification is used to determine which category something falls into. Regression attempts to construct a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change.

Social Intelligence: Effective computing is the study and development of systems that can detect, interpret, process, and simulate human It is an interdisciplinary field spanning computer science, psychology, and cognitive science. While the origins of the field can be traced to early philosophical inquiries into emotion, the more modern branch of computer science originated from Rosalind Picard's **1995** paper on "**effective computing**".

Creativity: A sub-field of AI addresses creativity theoretically (philosophical, psychological perspective) and practically (the specific implementation of systems that produce novel and useful outputs). Some related areas of computational research include artificial intuition and artificial thinking.

General Intelligence: Many researchers think that their work will eventually result in a machine with artificial general intelligence, combining all the skills described above and exceeding human capacity in most or all of these areas. Some believe that such a project may require anthropomorphic features such as artificial consciousness or an artificial brain.



Applications of AI

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

1. AI in Astronomy

Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.

Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

AI in Social Media

- Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.

Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

AI in Robotics:

Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.

Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

AI in Entertainment

We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

AI in Agriculture

Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

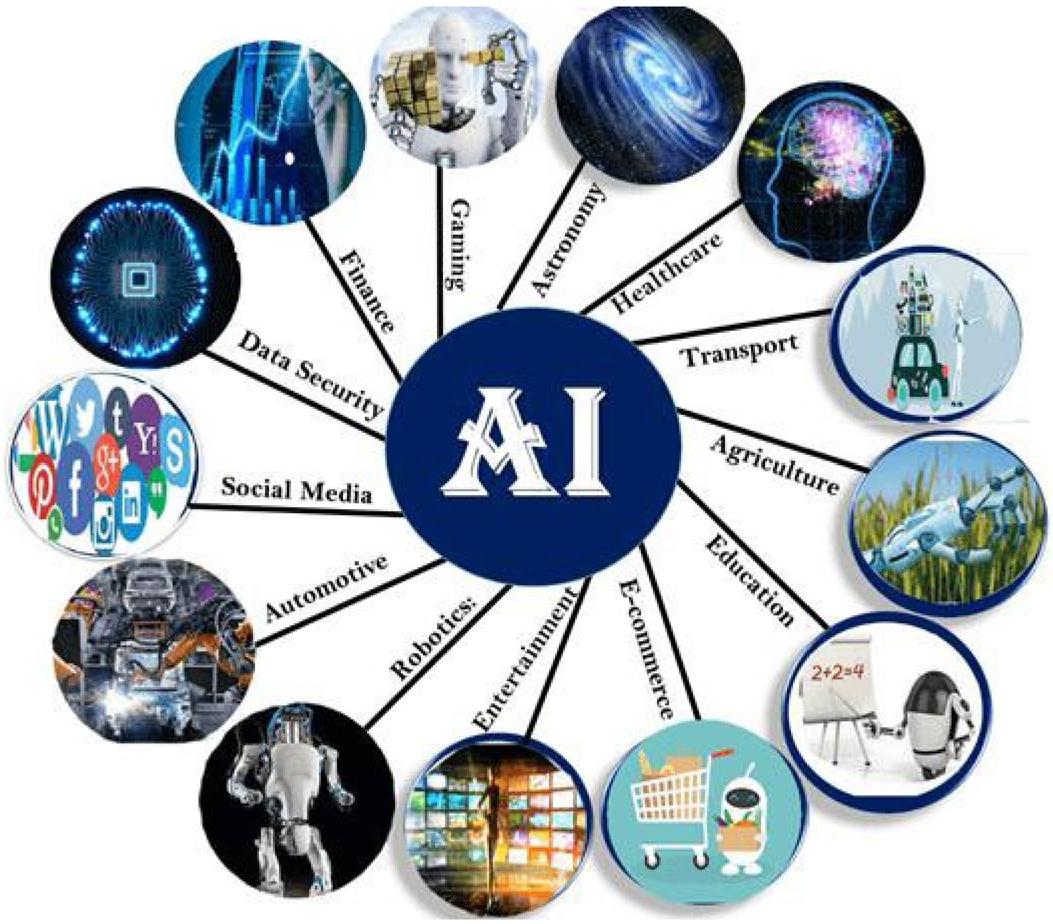
AI in E-commerce

AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

AI in education:

AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.

AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.



1.3Intelligent agent

Agents in Artificial Intelligence

An AI system can be defined as the study of the rational agent and its environment. The agents sense the environment through sensors and act on their environment through actuators. An AI agent can have mental properties such as knowledge, belief, intention, etc.

What is an Agent?

An agent can be anything that perceiveits environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**. An agent can be:

> **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
>
> **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
>
> **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

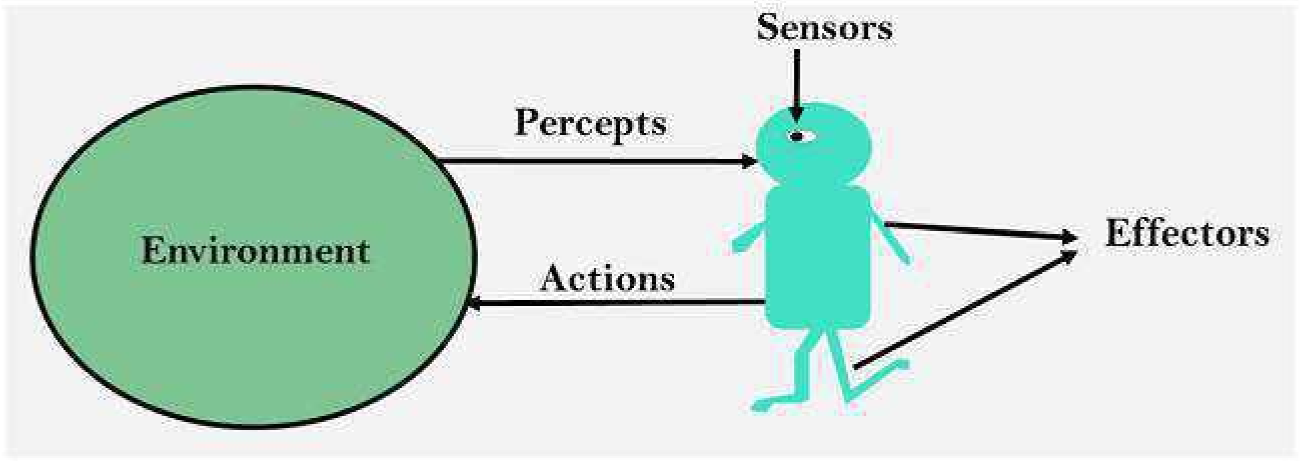
Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

Before moving forward, we should first know about sensors, effectors, and actuators.

Sensor: Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

Actuators: Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.

Effectors: Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.



Intelligent Agents:

An intelligent agent is an autonomous entity which acts upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

Rule 1: An AI agent must have the ability to perceive the environment.

Rule 2: The observation must be used to make decisions.

Rule 3: Decision should result in an action.

Rule 4: The action taken by an AI agent must be a rational action.

1.4 Computer vision

Computer vision is **a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs** — and take actions or make recommendations based on that information.

Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

Here are a few common tasks that computer vision systems can be used for:

Object classification. The system parses visual content and classifies the object on a photo/video to the defined category. For example, the system can find a dog among all objects in the image.

Object identification. The system parses visual content and identifies a particular object on a photo/video. For example, the system can find a specific dog among the dogs in the image.

Object tracking. The system processes video finds the object (or objects) that match search criteria and track its movement.

How does computer vision work?

Computer vision technology tends to mimic the way the human brain works. But how does our brain solve visual object recognition? One of the popular hypothesis states that our brains rely on patterns to decode individual objects. This concept is used to create computer vision systems.

Computer vision algorithms that we use today are based on pattern recognition. We train computers on a massive amount of visual data—computers process images, label objects on them, and find patterns in those objects. For example, if we send a million images of flowers, the computer will analyze them, identify patterns that are similar to all flowers and, at the end of this process, will create a model "flower." As a result, the computer will be able to accurately detect whether a particular image is a flower every time we send them pictures.

Golan Levin, in his article Image Processing and Computer Vision, provides technical details about the process that machines follow in interpreting images. In short, machines interpret images as a series of pixels, each with their own set of color values. For example, below is a picture of Abraham Lincoln. Each pixel's brightness in this image is represented by a single 8-bit number, ranging from 0 (black) to 255 (white). These numbers are what software sees when you input an image. This data is provided as an input to the computer vision algorithm that will be responsible for further analysis and decision making.

1.5Natural language processing

What is natural language processing?

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There's a good chance you've interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

NLP tasks

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in sentence structure— these just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start, if those applications are going to be useful.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.
Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'
Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.
Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).
Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

1.6Turing Test in AI

In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed that the computer can be said to be an intelligent if it can mimic human response under specific conditions.

Turing Test was introduced by Turing in his 1950 paper, "Computing Machinery and Intelligence," which considered the question, "Can Machine think?"

The Turing test is based on a party game "Imitation game," with some modifications. This game involves three players in which one player is Computer, another player is human responder, and the third player is a human Interrogator, who is isolated from other two players and his job is to find that which player is machine among two of them.

Consider, Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

The conversation between all players is via keyboard and screen so the result would not depend on the machine's ability to convert words as speech.

The test result does not depend on each correct answer, but only how closely its responses like a human answer. The computer is permitted to do everything possible to force a wrong identification by the interrogator.

The questions and answers can be like:

Interrogator: Are you a computer?

PlayerA (Computer): No

Interrogator: Multiply two large numbers such as (256896489*456725896)

Player A: Long pause and give the wrong answer.

In this game, if an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully, and the machine is said to be intelligent and can think like a human.

1.7 Problem solving in games

Problem solving games are activities that require players to use critical thinking skills to solve puzzles. Example activities include escape rooms, Sudoku, and murder mysteries. The purpose of these exercises is to sharpen reasoning and decision-making skills in group settings and to do team building with employees.

2. Introduction to search algorithm

2.1 Search, Search space and search tree

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:

Search Space: Search space represents a set of possible solutions, which a system may have.

Start State: It is a state from where agent begins **the search**.

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.

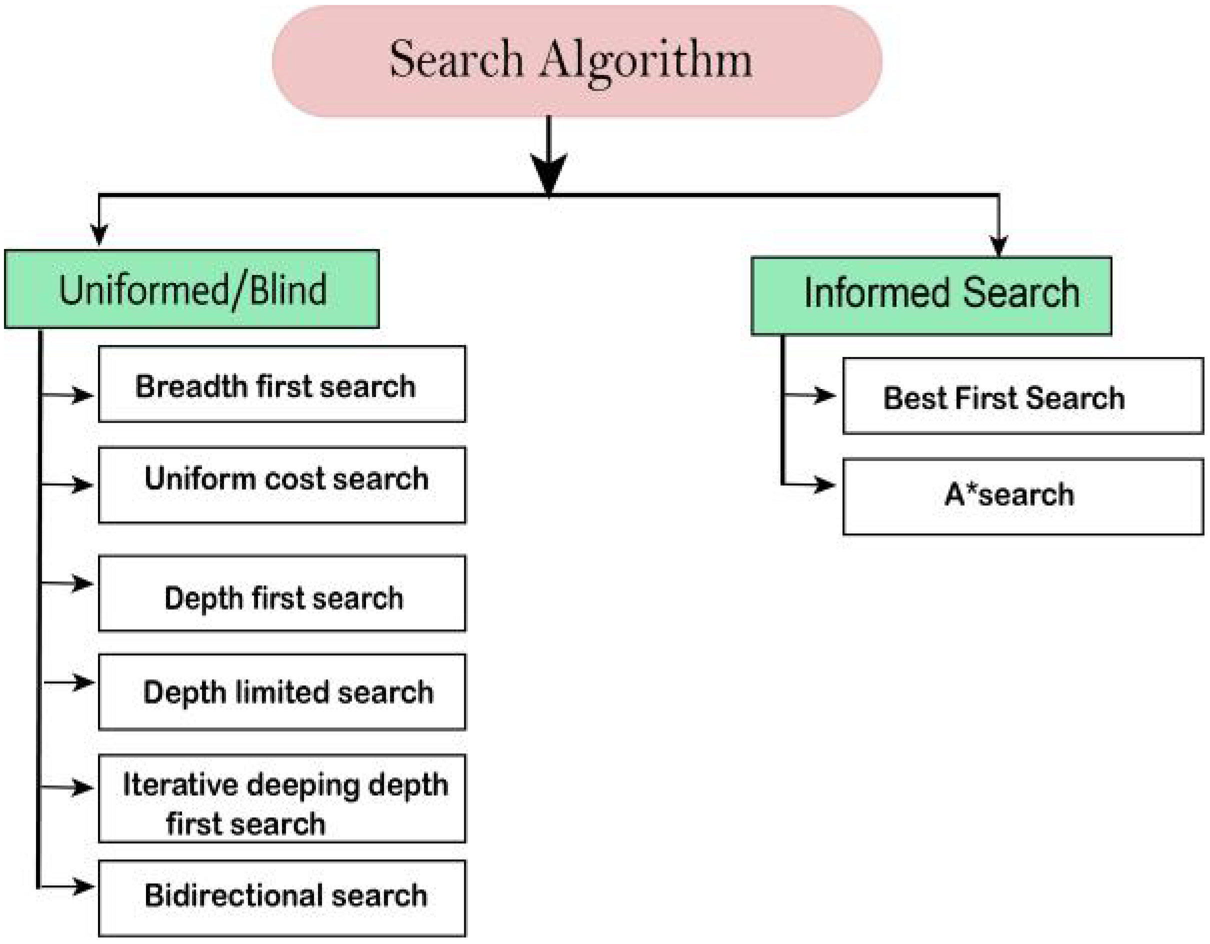
Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

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.

2.2Types 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.

- Greedy Search
- A* Search

Uninformed Search Algorithms

Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search.

Following are the various types of uninformed search algorithms:

- **Breadth-first Search**

Depth-first Search

Depth-limited Search

Iterative deepening depth-first search

Uniform cost search

Bidirectional Search

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:

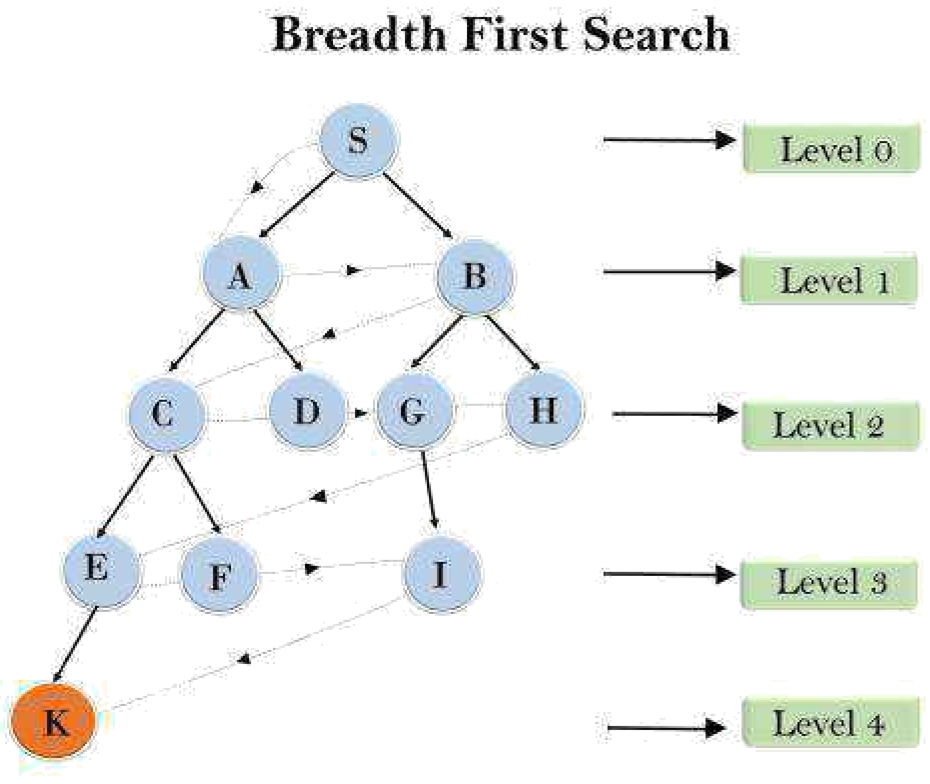
It requires lots of memory since each level of the tree must be saved into memory to expand the next level.

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:

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.

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

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.

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.

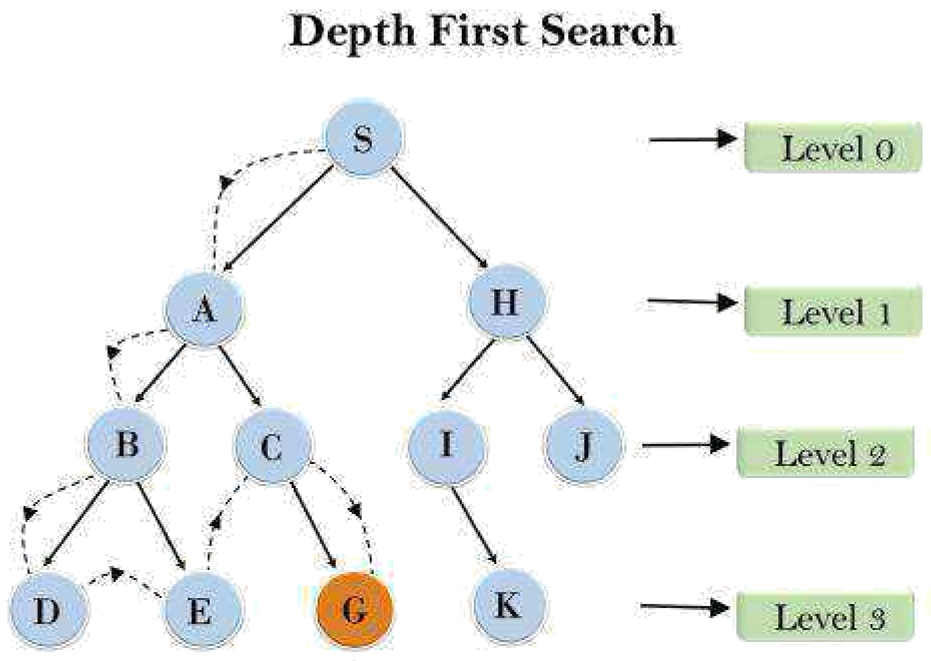
DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

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.

3. 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:

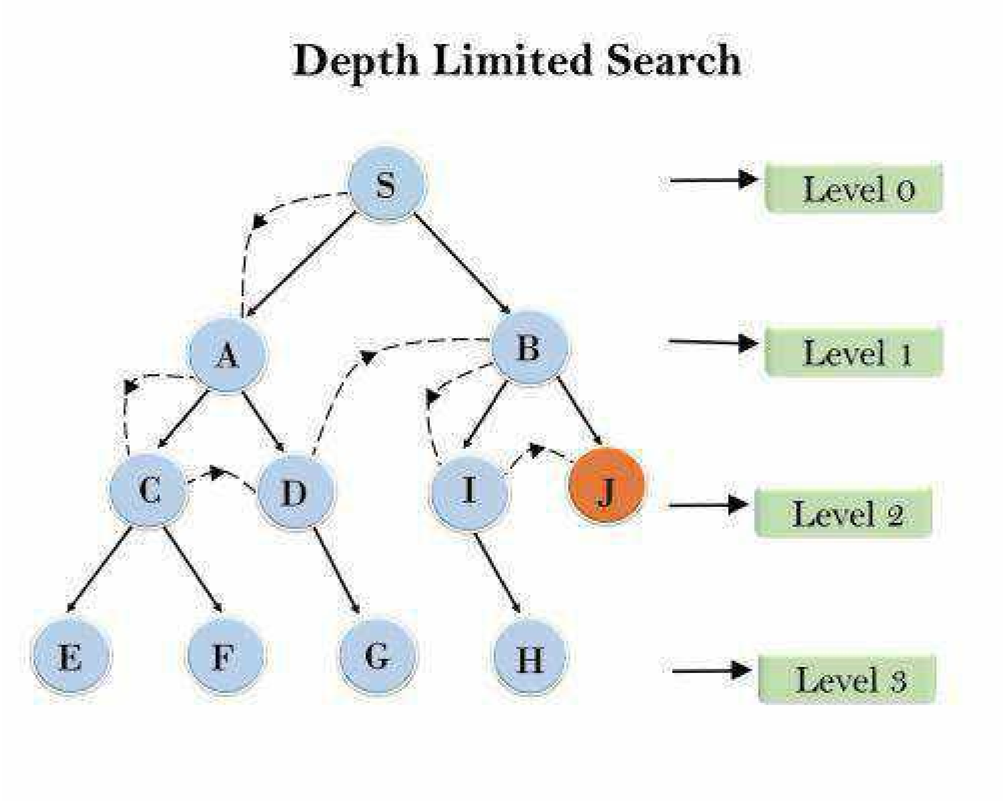
Depth-limited search is Memory efficient.

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 $\mathbf{O(b^{\ell})}$.

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

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

4. 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.

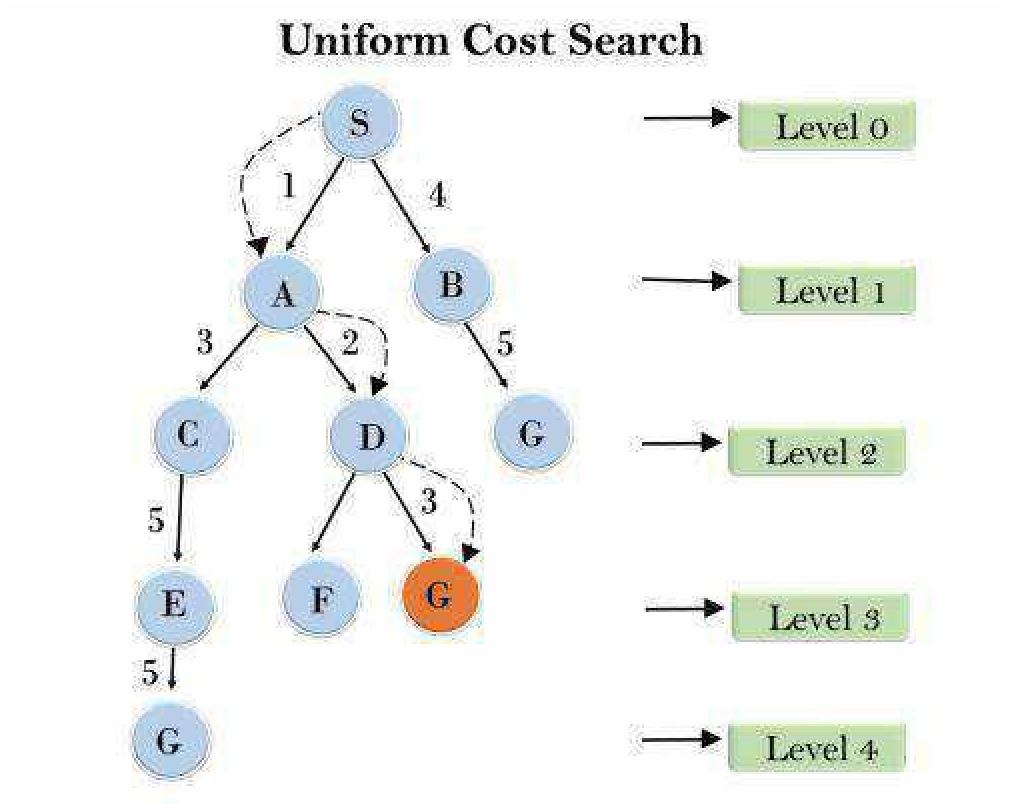
Advantages:

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.

5. Iterative deepeningdepth-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.

Example:

Following tree structure is showing the iterative deepening depth-first search. IDDFS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:

1'st Iteration----- > 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:

he 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

6. Bidirectional Search Algorithm:

Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.

Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

Advantages:

Bidirectional search is fast.

Bidirectional search requires less memory

Disadvantages:

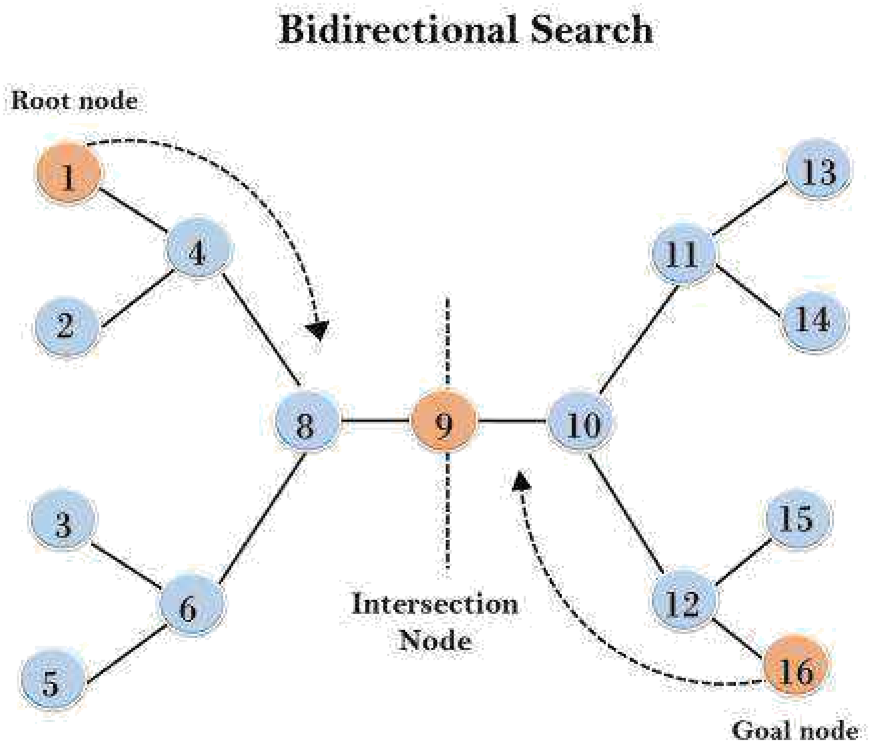
Implementation of the bidirectional search tree is difficult.

In bidirectional search, one should know the goal state in advance.

Example:

In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.

The algorithm terminates at node 9 where two searches meet.



Completeness: Bidirectional Search is complete if we use BFS in both searches.

Time Complexity: Time complexity of bidirectional search using BFS is **O(b^d)**.

Space Complexity: Space complexity of bidirectional search is **O(b^d)**.

Optimal: Bidirectional search is Optimal.

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:

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

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.

Best First Search Algorithm(Greedy search)

A* Search Algorithm

) 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:

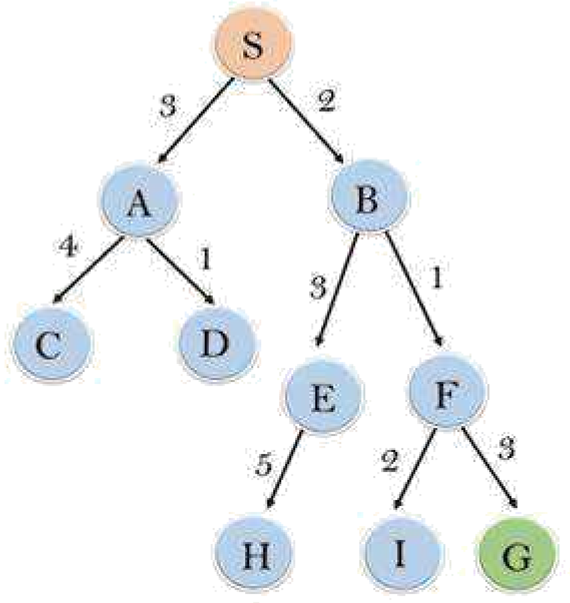
It can behave as an unguided depth-first search in the worst case scenario.

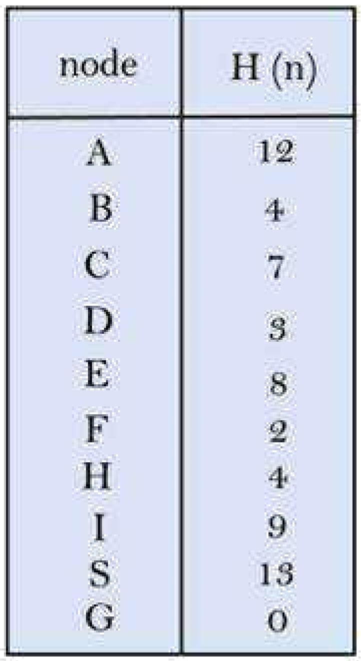
It can get stuck in a loop as DFS.

This algorithm is not optimal.

Example:

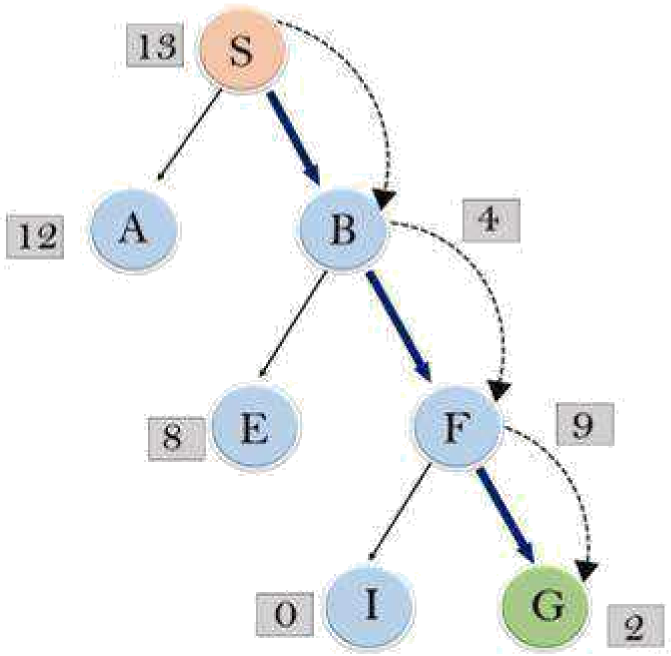
Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.





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.

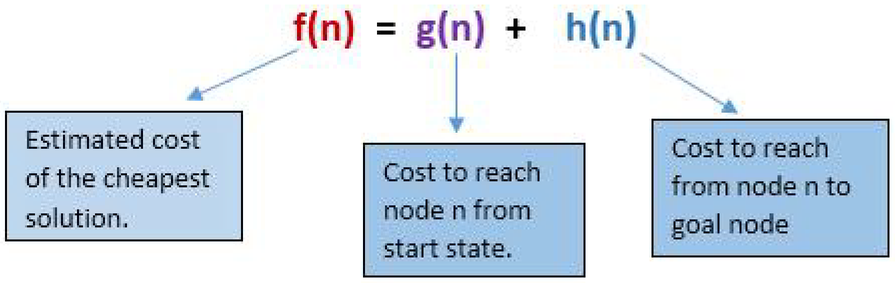
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

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

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.

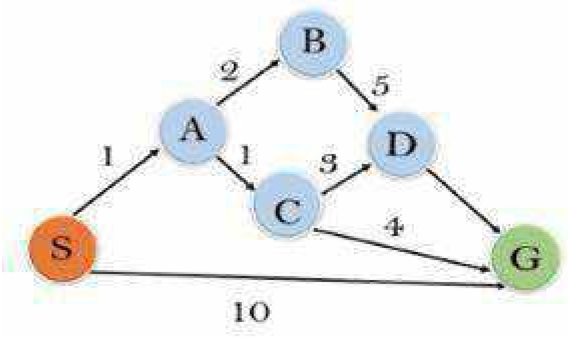
A* search algorithm has some complexity issues.

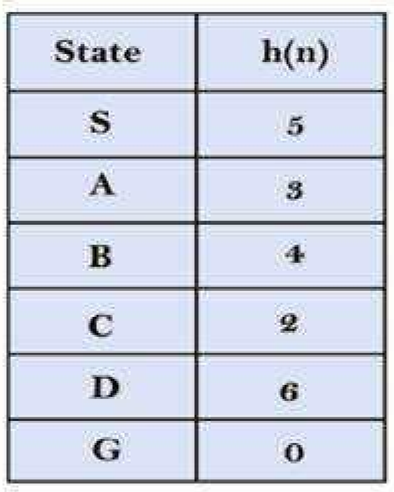
The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

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.

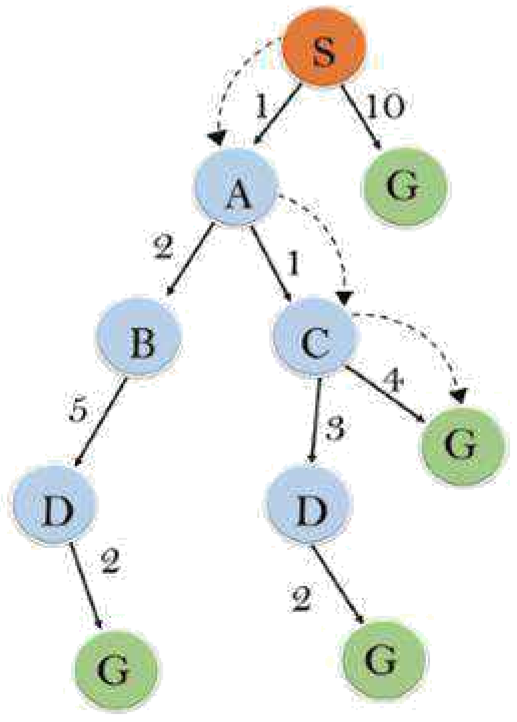
Here we will use OPEN and CLOSED list.





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)**

AO* Algorithm:

Our real-life situations can't be exactly decomposed into either AND tree or OR tree but is always a combination of both. So, we need an AO* algorithm where O stands for 'ordered'. AO* algorithm represents a part of the search graph that has been explicitly generated so far. AO* algorithm is given as follows:

Step-1: Create an initial graph with a single node (start node).
Step-2: Transverse the graph following the current path, accumulating node that has not yet been expanded or solved.
Step-3: Select any of these nodes and explore it. If it has no successors then call this value-FUTILITY else calculate f'(n) for each of the successors.
Step-4: If **f'(n)=0**, then mark the node as **SOLVED**.
Step-5: Change the value of f'(n) for the newly created node to reflect its successors by backpropagation.
Step-6: Whenever possible use the most promising routes, If a node is marked as SOLVED then mark the parent node as SOLVED.
Step-7: If the starting node is SOLVED or value is greater than **FUTILITY** then stop else repeat from Step-2.

Adversarial Search

Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.

But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.

The environment with more than one agent is termed as **multi-agent environment**, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.

So, **Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games**.

Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

Types of Games in AI:

Perfect information: A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.

Imperfect information: If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.

Deterministic games: Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.

Non-deterministic games: Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

Zero-Sum Game

Zero-sum games are adversarial search which involves pure competition.

In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.

One player of the game try to maximize one single value, while other player tries to minimize it.

Each move by one player in the game is called as ply.

Chess and tic-tac-toe are examples of a Zero-sum game.

Zero-sum game: Embedded thinking

The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:

What to do.

How to decide the move

Needs to think about his opponent as well

The opponent also thinks what to do

Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

Formalization of the problem:

A game can be defined as a type of search in AI which can be formalized of the following elements:

Initial state: It specifies how the game is set up at the start.

Player(s): It specifies which player has moved in the state space.

Action(s): It returns the set of legal moves in state space.

Result(s, a): It is the transition model, which specifies the result of moves in the state space.

Terminal-Test(s): Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.

Utility(s, p): A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

Game tree:

A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

Example: Tic-Tac-Toe game tree:

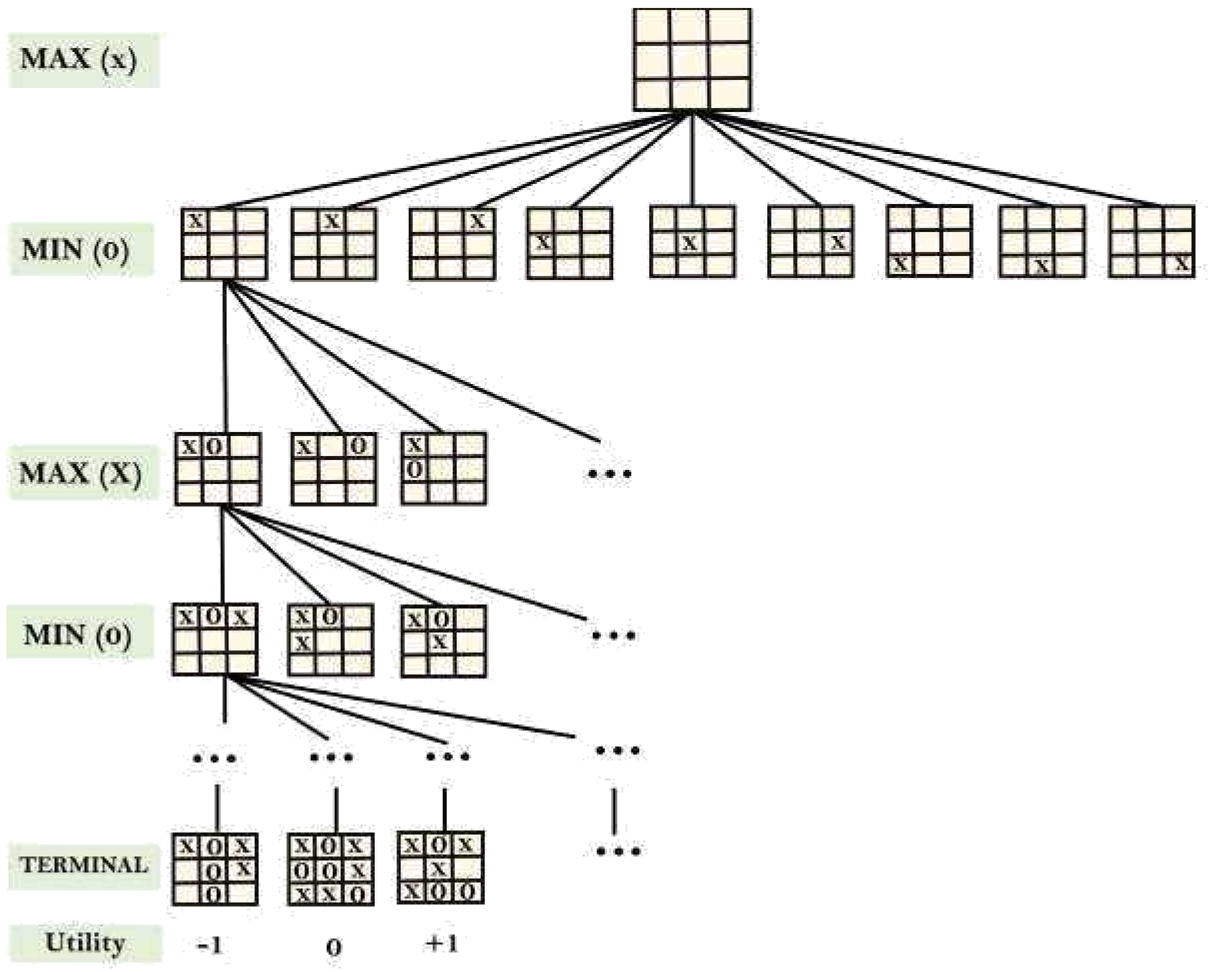
The following figure is showing part of the game-tree for tic-tac-toe game. Following are some key points of the game:

There are two players MAX and MIN.

Players have an alternate turn and start with MAX.

MAX maximizes the result of the game tree

MIN minimizes the result.



Example Explanation:

From the initial state, MAX has 9 possible moves as he starts first. MAX place x and MIN place o, and both player plays alternatively until we reach a leaf node where one player has three in a row or all squares are filled.

Both players will compute each node, minimax, the minimax value which is the best achievable utility against an optimal adversary.

Suppose both the players are well aware of the tic-tac-toe and playing the best play. Each player is doing his best to prevent another one from winning. MIN is acting against Max in the game.

So in the game tree, we have a layer of Max, a layer of MIN, and each layer is called as **Ply**. Max place x, then MIN puts o to prevent Max from winning, and this game continues until the terminal node.

In this either MIN wins, MAX wins, or it's a draw. This game-tree is the whole search space of possibilities that MIN and MAX are playing tic-tac-toe and taking turns alternately.

Hence adversarial Search for the minimax procedure works as follows:

It aims to find the optimal strategy for MAX to win the game.

It follows the approach of Depth-first search.

In the game tree, optimal leaf node could appear at any depth of the tree.

Propagate the minimax values up to the tree until the terminal node discovered.

In a given game tree, the optimal strategy can be determined from the minimax value of each node, which can be written as MINIMAX(n). MAX prefer to move to a state of maximum value and MIN prefer to move to a state of minimum value then:

For a state S MINIMAX(s) =

$$\begin{cases} \text{UTILITY}(s) & \text{If TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{If PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{If PLAYER}(s) = \text{MIN.} \end{cases}$$

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.

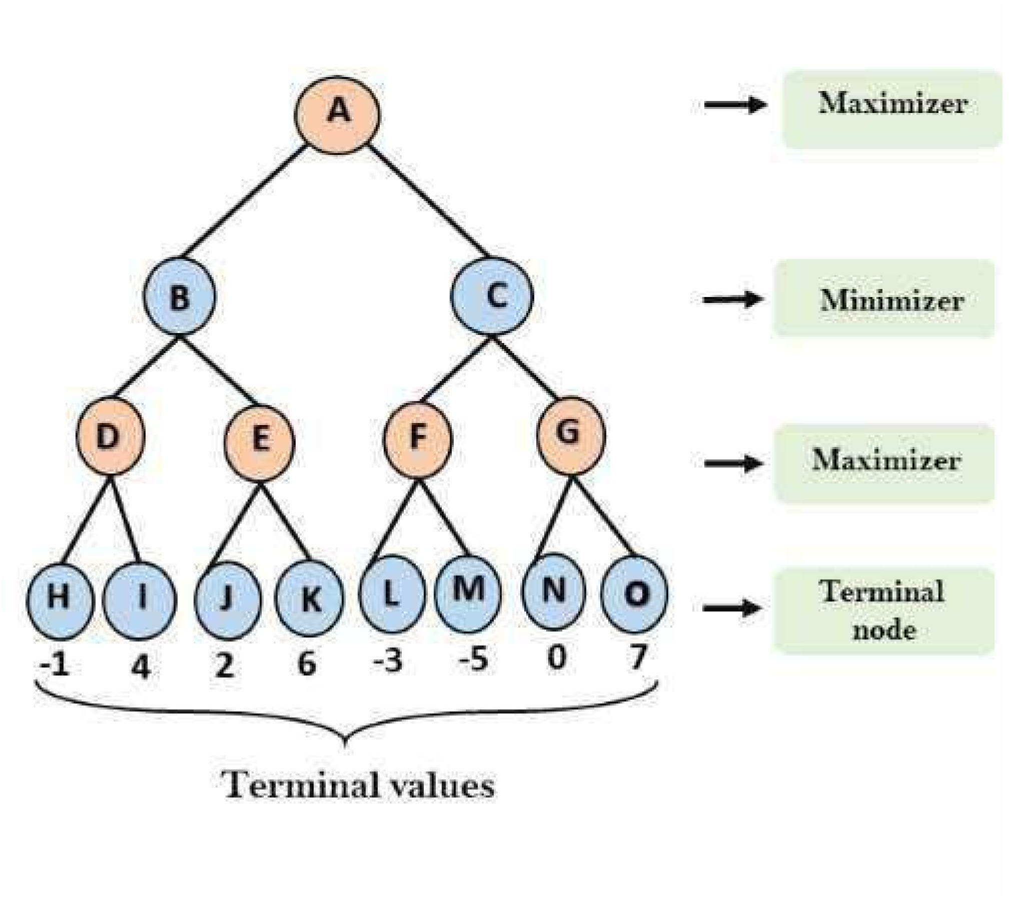
Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.

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.

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 Dmax(-1,- -∞) => max(-1,4)= 4

For Node Emax(2, -∞) => max(2, 6)= 6

For Node Fmax(-3, -∞) => max(-3,-5) = -3

For node Gmax(0, -∞) = max(0, 7) = 7

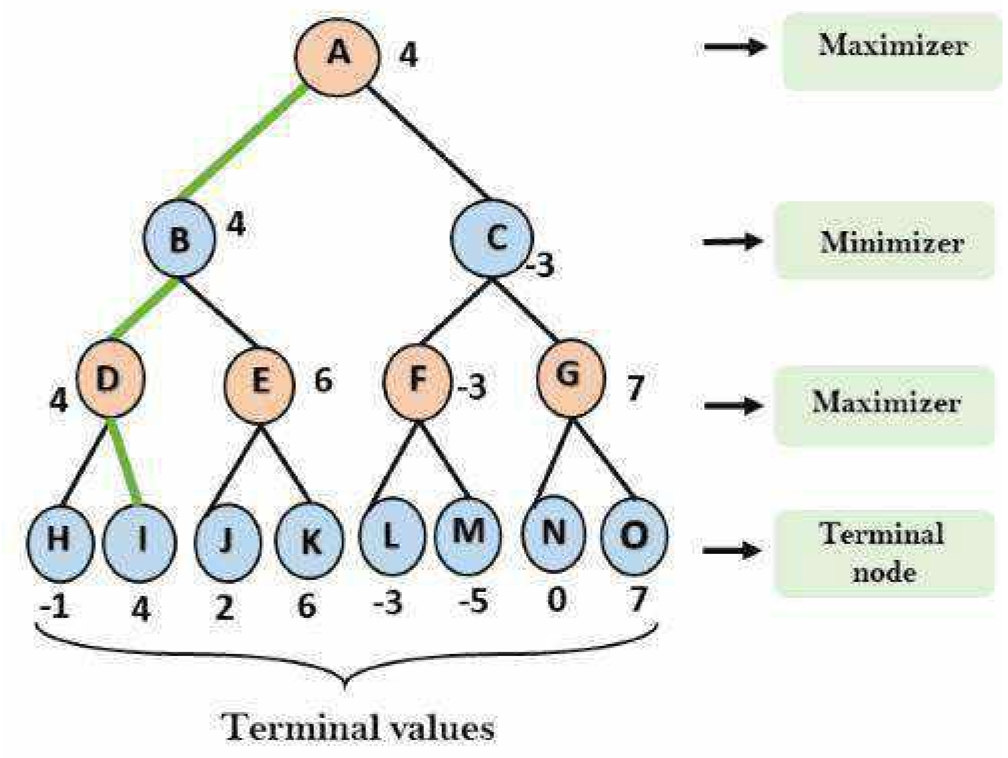
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

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** which we have discussed in the next topic.

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:

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 **-∞**.

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.

Condition for Alpha-beta pruning:

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

$\alpha >= \beta$

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.

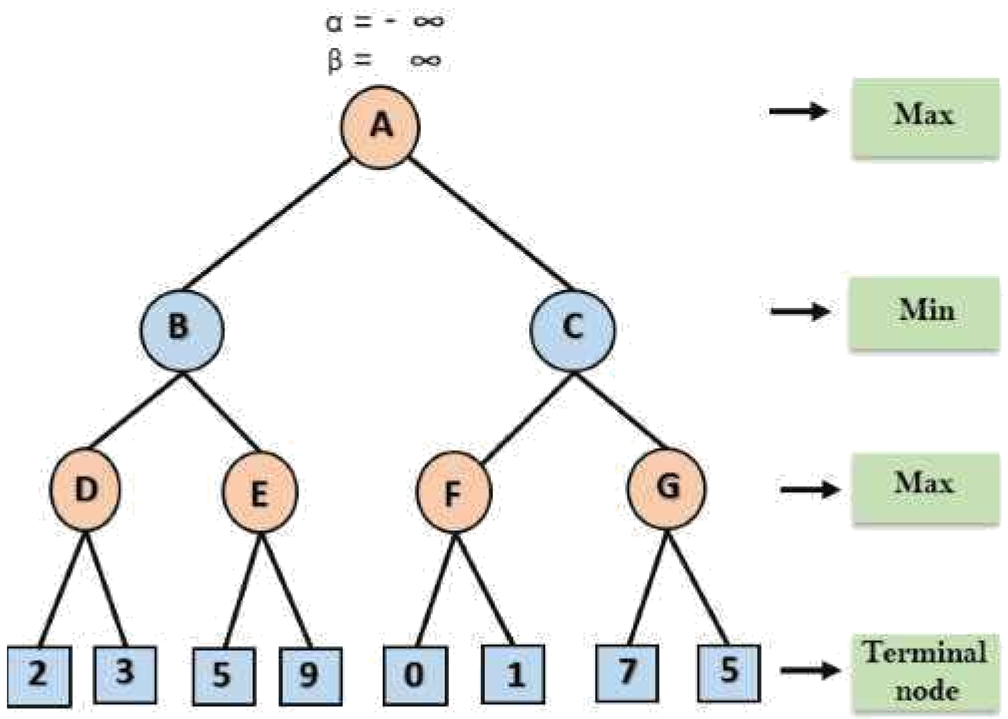
While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.

We will only pass the alpha, beta values to the child

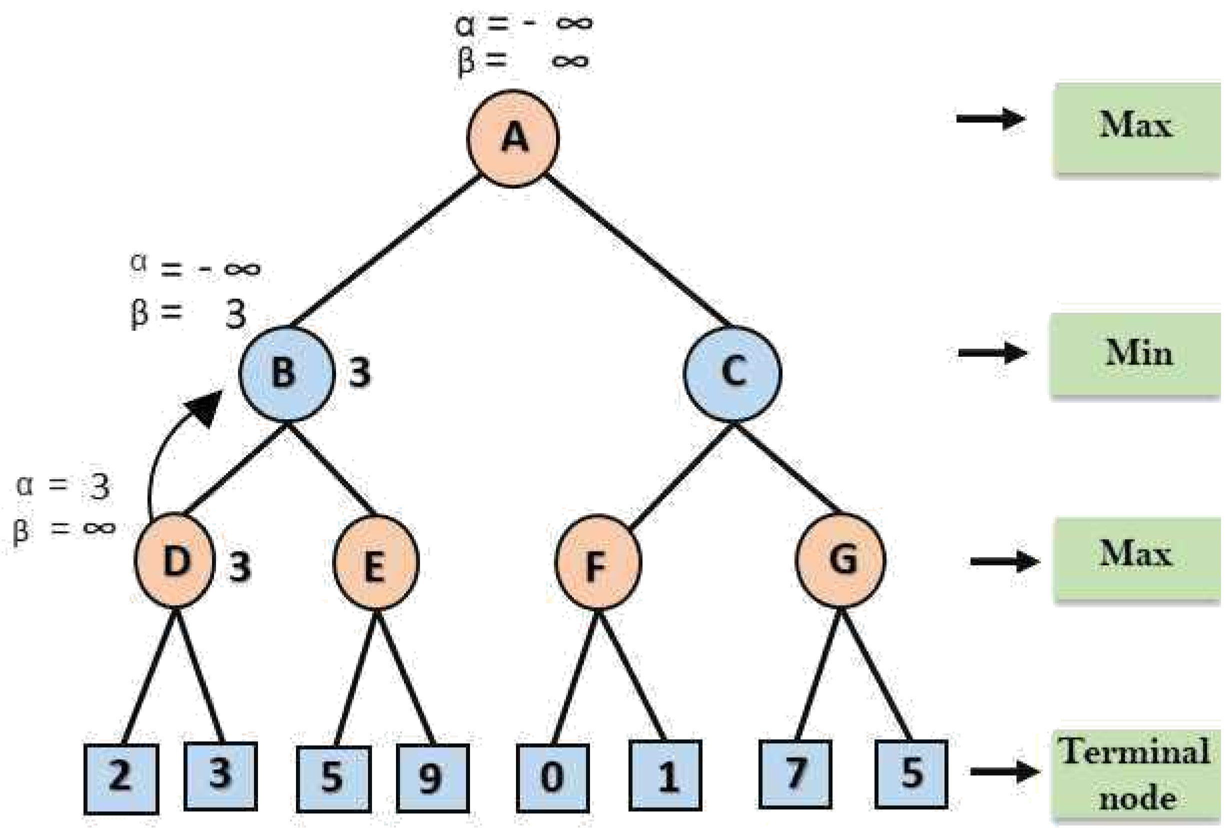
nodes. Working of Alpha-Beta Pruning:

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 $\alpha= -\infty$ and $\beta= +\infty$, these value of alpha and beta passed down to node B where again $\alpha= -\infty$ and $\beta= +\infty$, and Node B passes the same value to its child D.

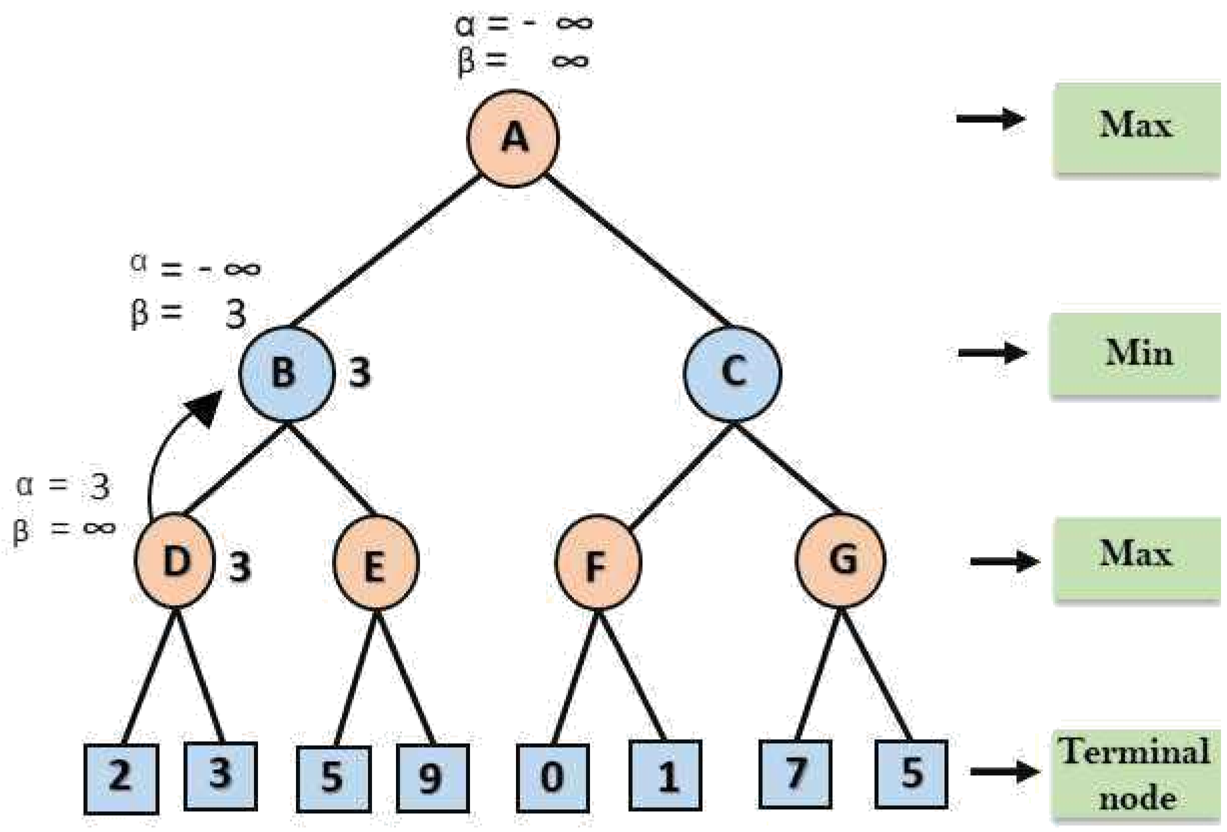


Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and



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.

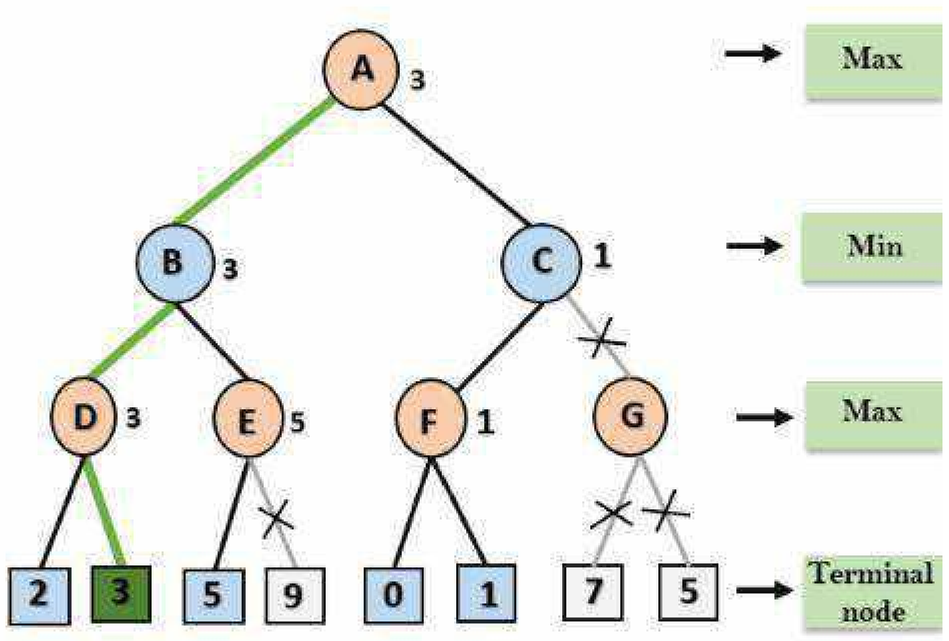
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

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 o f 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.



Knowledge representation and reasoning

1What to represent, Knowledge:-

What is knowledge representation?

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**. Hence we can describe Knowledge representation as following:

Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.

It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.

It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

Object: All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.

Events: Events are the actions which occur in our world.

Performance: It describe behavior which involves knowledge about how to do things.

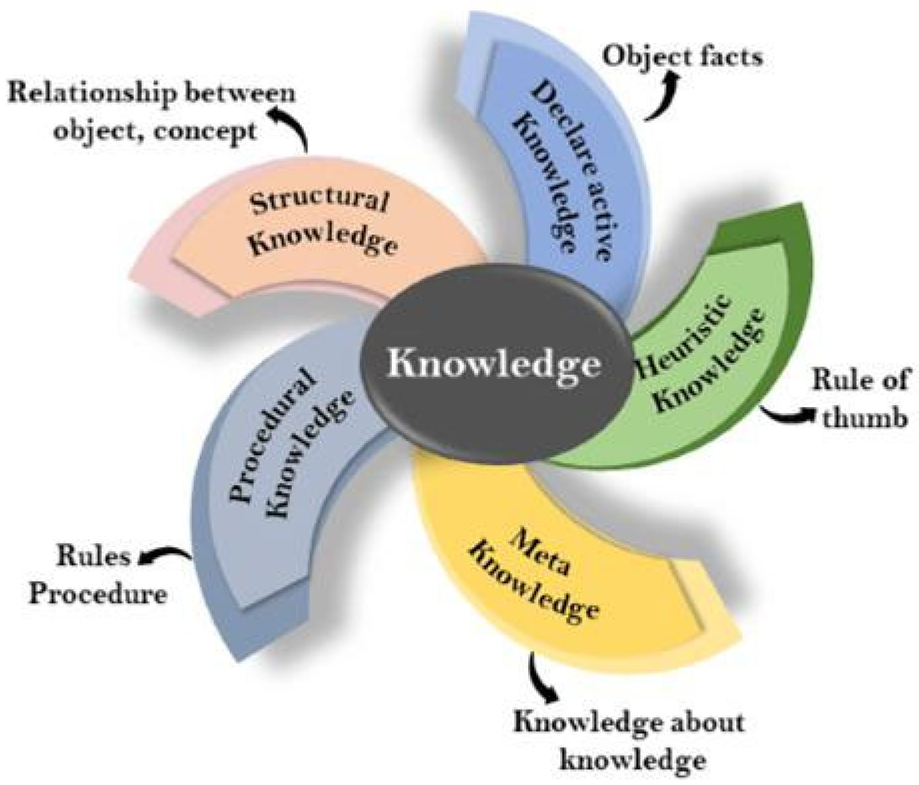
Meta-knowledge: It is knowledge about what we know.

Facts: Facts are the truths about the real world and what we represent.

Knowledge-Base: The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge: Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Types of knowledge



1.Declarative Knowledge:

Declarative knowledge is to know about something.

It includes concepts, facts, and objects.

It is also called descriptive knowledge and expressed in declarativesentences.

It is simpler than procedural language.

Procedural Knowledge

It is also known as imperative knowledge.

Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.

It can be directly applied to any task.

It includes rules, strategies, procedures, agendas, etc.

Procedural knowledge depends on the task on which it can be applied.

Meta-knowledge:

Knowledge about the other types of knowledge is called Meta-knowledge.

Heuristic knowledge:

Heuristic knowledge is representing knowledge of some experts in a filed or subject.

Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

Structural knowledge:

Structural knowledge is basic knowledge to problem-solving.

It describes relationships between various concepts such as kind of, part of, and grouping of something.

It describes the relationship that exists between concepts or objects.

The relation between knowledge and intelligence:

Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input.

Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.

As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.

3.2Properties of knowledge representation system, Approaches

The following properties should be possessed by a knowledge representation system.

Representational Adequacy
the ability to represent the required knowledge;
Inferential Adequacy
- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency

the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

To date no single system optimizes all of the above.

Approaches to knowledge representation:

There are mainly four approaches to knowledge representation, which are given below:

1. Simple relational knowledge:

It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.

This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.

This approach has little opportunity for inference.

Example: The following is the simple relational knowledge representation.

Player	**Weight**	**Age**
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable knowledge:

In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.

All classes should be arranged in a generalized form or a hierarchal manner.

In this approach, we apply inheritance property.

- Elements inherit values from other members of a class.

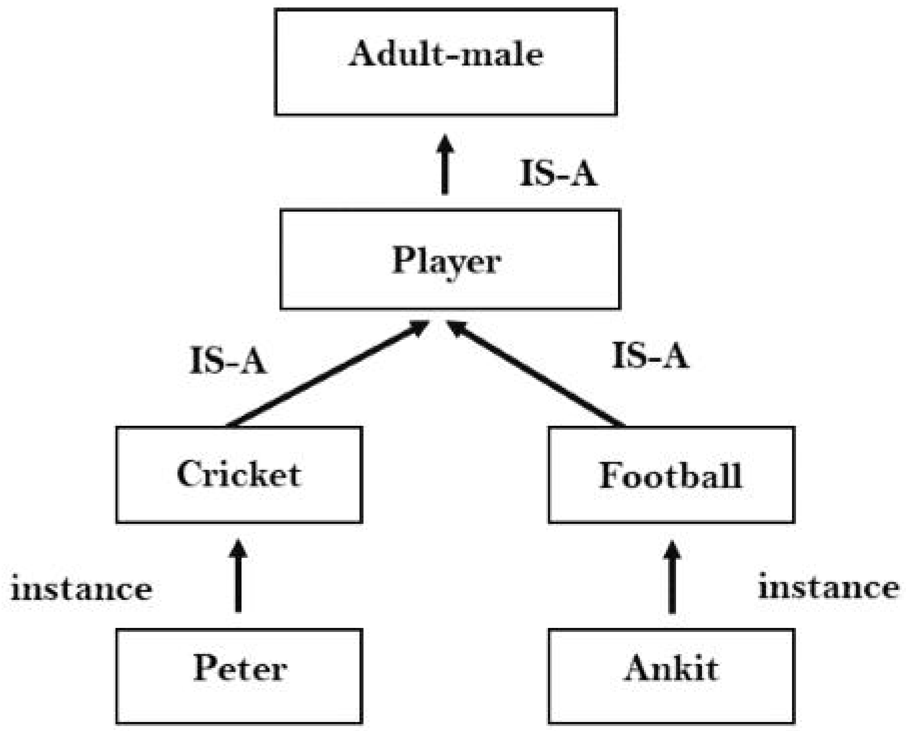
This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.

Every individual frame can represent the collection of attributes and its value.

In this approach, objects and values are represented in Boxed nodes.

We use Arrows which point from objects to their values.

Example:



3. Inferential knowledge:

Inferential knowledge approach represents knowledge in the form of formal logics.

This approach can be used to derive more facts.

It guaranteed correctness.

Example: Let's suppose there are two statements: a. Marcus is a man

b. All men are mortal Then it can represent as;

man(Marcus)
∀x = man (x) ----------> mortal (x)s

4. Procedural knowledge:

Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.

In this approach, one important rule is used which is **If-Then rule**.

- In this knowledge, we can use various coding languages such as **LISP language** and **Prolog language**.

We can easily represent heuristic or domain-specific knowledge using this approach.

But it is not necessary that we can represent all cases in this approach.

3.3 Knowledge representation

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the **beliefs, intentions**, and **judgments** of an **intelligent agent** can be expressed suitably for automated reasoning. One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

Knowledge Representation and Reasoning (**KR, KRR**) represents information from the real world for a computer to understand and then utilize this knowledge to solve **complex real-life problems** like communicating with human beings in natural language. Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

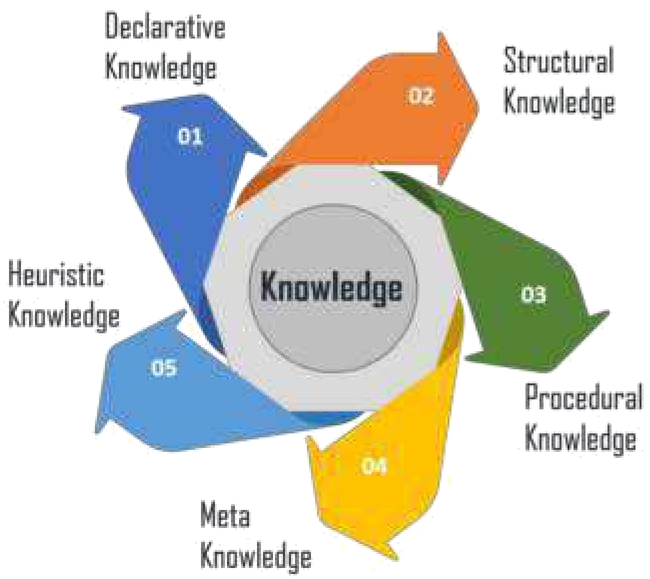
The different kinds of knowledge that need to be represented in AI include:

Objects
Events
Performance
Facts
Meta-Knowledge
Knowledge-base

Now that you know about Knowledge representation in AI, let's move on and know about the different types of Knowledge.

Different Types of Knowledge

There are 5 types of Knowledge such as:



Declarative Knowledge – It includes concepts, facts, and objects and expressed in a declarative sentence.
Structural Knowledge – It is a basic problem-solving knowledge that describes the relationship between concepts and objects.
Procedural Knowledge – This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.
Meta Knowledge – Meta Knowledge defines knowledge about other types of Knowledge.
Heuristic Knowledge – This represents some expert knowledge in the field or subject.

3.4 Reasoning and types of reasoning

Reasoning:

The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or we can say, "**Reasoning is a way to infer facts from existing data**." It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

Types of Reasoning

In artificial intelligence, reasoning can be divided into the following categories:

Deductive reasoning

Inductive reasoning

Abdicative reasoning

Common Sense Reasoning

Monotonic Reasoning

Non-monotonic Reasoning

1.Deductive reasoning:

Deductive reasoning is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.

Deductive reasoning is a type of propositional logic in AI, and it requires various rules and facts. It is sometimes referred to as top-down reasoning, and contradictory to inductive reasoning.

In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.

Deductive reasoning mostly starts from the general premises to the specific conclusion, which can be explained as below example.

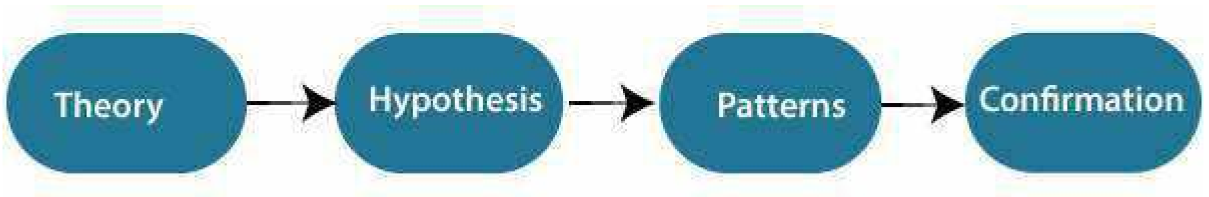
Example:

Premise-1: All the human eats veggies

Premise-2: Suresh is human.

Conclusion: Suresh eats veggies.

The general process of deductive reasoning is given below:



2. Inductive Reasoning:

Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with the series of specific facts or data and reaches to a general statement or conclusion.

Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.

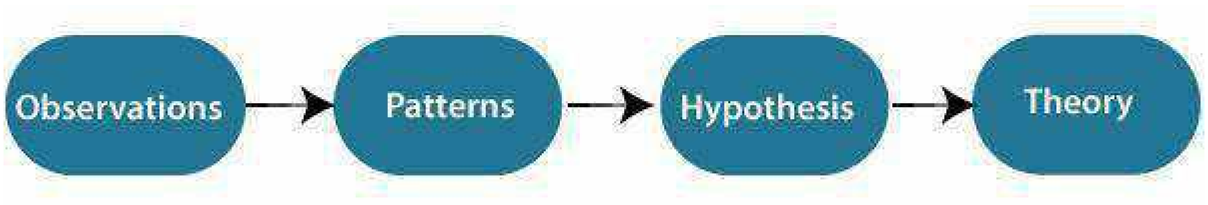
In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.

In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.

Example:

Premise: All of the pigeons we have seen in the zoo are white.

Conclusion: Therefore, we can expect all the pigeons to be white.



3.Abductive reasoning:

Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.

Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, the premises do not guarantee the conclusion.

Example:

Implication: Cricket ground is wet if it is raining

Axiom: Cricket ground is wet.

Conclusion It is raining.

4. Common Sense Reasoning

Common sense reasoning is an informal form of reasoning, which can be gained through experiences.

Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day.

It relies on good judgment rather than exact logic and operates on **heuristic knowledge** and **heuristic rules**.

Example:

One person can be at one place at a time.

If I put my hand in a fire, then it will burn.

The above two statements are the examples of common sense reasoning which a human mind can easily understand and assume.

5. Monotonic Reasoning:

In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.

To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.

Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Any theorem proving is an example of monotonic reasoning.

Example:

Earth revolves around the Sun.

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

Advantages of Monotonic Reasoning:

In monotonic reasoning, each old proof will always remain valid.

If we deduce some facts from available facts, then it will remain valid for always.

Disadvantages of Monotonic Reasoning:

- We cannot represent the real world scenarios using Monotonic reasoning.

Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.

Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

Non-monotonic Reasoning

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example: Let suppose the knowledge base contains the following knowledge:

Birds can fly

Penguins cannot fly

Pitty is a bird

So from the above sentences, we can conclude that **Pitty can fly**.

However, if we add one another sentence into knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

Advantages of Non-monotonic reasoning:

For real-world systems such as Robot navigation, we can use non-monotonic reasoning.

In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

Disadvantages of Non-monotonic Reasoning:

In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.

It cannot be used for theorem **proving**.

4.Machine learning

4.1 Machine learning

What is machine learning?

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

What are the different types of machine learning?

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches:supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists choose to use depends on what type of data they want to predict.

Supervised learning: In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

Unsupervised learning: This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

Semi-supervised learning: This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

Reinforcement learning: Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

4.2Statistical or Un-supervised learning

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

Unsupervised learning is helpful for finding useful insights from the data.

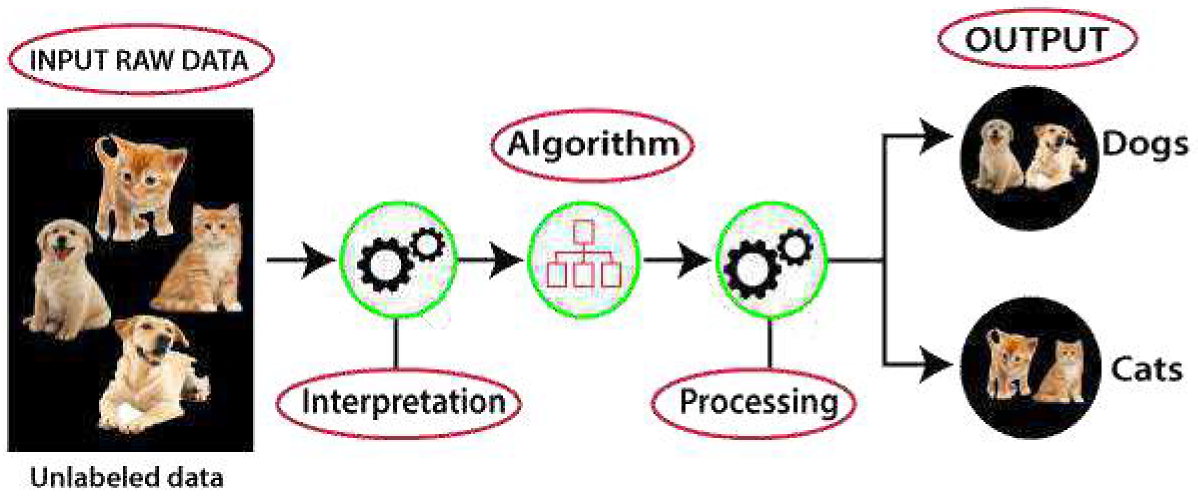
Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.

Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.

In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

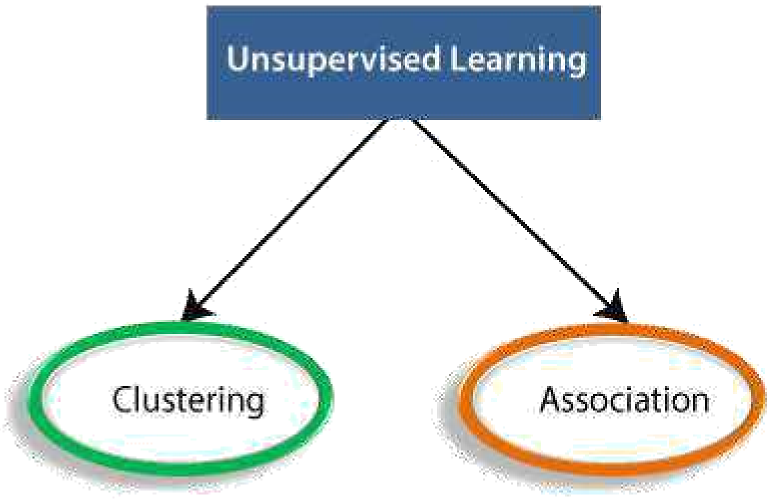


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



Clustering: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

Association: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

K-means clustering

KNN (k-nearest neighbors)

Hierarchal clustering

Anomaly detection

Neural Networks

Principle Component Analysis

Independent Component Analysis

Apriori algorithm

Singular value decomposition

Advantages of Unsupervised Learning

Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.

Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.

The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

4.3ML properties

1- THE ABILITY TO PERFORM AUTOMATED DATA VISUALIZATION

A massive amount of data is being generated by businesses and common people on a regular basis. By visualizing notable relationships in data, businesses can not only make better decisions but build confidence as well. Machine learning offers a number of tools that provide rich snippets of data which can be applied to both unstructured and structured data. With the help of user-friendly automated data visualization platforms in machine learning, businesses can obtain a wealth of new insights in an effort to increase productivity in their processes.

2- AUTOMATION AT ITS BEST

One of the biggest characteristics of machine learning is its ability to automate repetitive tasks and thus, increasing productivity. A huge number of organizations are already using machine learning-powered paperwork and email automation. In the financial sector, for example, a huge number of repetitive, data-heavy and predictable tasks are needed to be performed. Because of this, this sector uses different types of machine learning solutions to a great extent. The make accounting tasks faster, more insightful, and more accurate. Some aspects that have been already addressed by machine learning include addressing financial queries with the help of chatbots, making predictions, managing expenses, simplifying invoicing, and automating bank reconciliations.

3- CUSTOMER ENGAGEMENT LIKE NEVER BEFORE

For any business, one of the most crucial ways to drive engagement, promote brand loyalty and establish long-lasting customer relationships is by triggering meaningful conversations with its target customer base. Machine learning plays a critical role in enabling businesses and brands to spark more valuable conversations in terms of customer engagement. The technology analyzes particular phrases, words, sentences, idioms, and content formats which resonate with certain audience members. You can think of Pinterest which is successfully using machine learning to personalize suggestions to its users. It uses the technology to source content in which users will be interested, based on objects which they have pinned already.

4- THE ABILITY TO TAKE EFFICIENCY TO THE NEXT LEVEL WHEN MERGED WITH IOT

Thanks to the huge hype surrounding the IoT, machine learning has experienced a great rise in popularity. IoT is being designated as a strategically significant area by many companies. And many others have launched pilot projects to gauge the potential of IoT in the context of business operations. But attaining financial benefits through IoT isn't easy. In order to achieve success, companies, which are offering IoT consulting services and platforms, need to clearly determine the areas that will change with the implementation of IoT strategies. Many of these businesses have failed to address it. In this scenario, machine learning is probably the best technology that can be used to attain higher levels of efficiency. By merging machine learning with IoT, businesses can boost the efficiency of their entire production processes.

5- THE ABILITY TO CHANGE THE MORTGAGE MARKET

It's a fact that fostering a positive credit score usually takes discipline, time, and lots of financial planning for a lot of consumers. When it comes to the lenders, the consumer credit score is one of the biggest measures of creditworthiness that involve a number of factors including payment history, total debt, length of credit history etc. But wouldn't it be great if there is a simplified and better measure? With the help of machine learning, lenders can now obtain a more comprehensive consumer picture. They can now predict whether the customer is a low spender or a high spender and understand his/her tipping point of spending. Apart from mortgage lending, financial institutions are using the same techniques for other types of consumer loans.

6- ACCURATE DATA ANALYSIS

Traditionally, data analysis has always been encompassing trial and error method, an approach which becomes impossible when we are working with large and heterogeneous datasets. Machine learning comes as the best solution to all these issues by offering effective alternatives to analyzing massive volumes of data. By developing efficient and fast algorithms, as well as, data-driven models for processing of data in real-time, machine learning is able to generate accurate analysis and results.

7- BUSINESS INTELLIGENCE AT ITS BEST

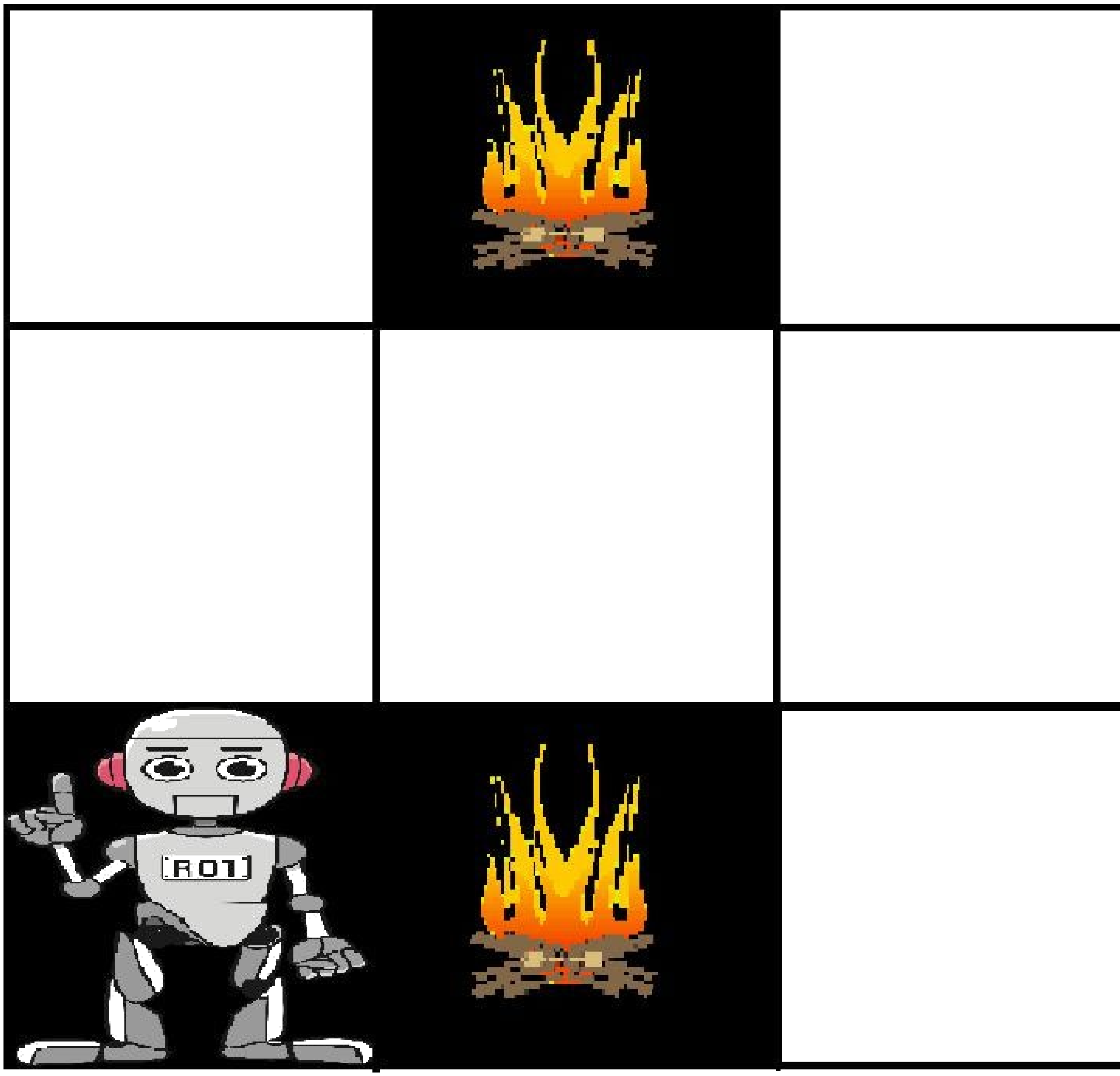
Machine learning characteristics, when merged with big data analytical work, can generate extreme levels of business intelligence with the help of which several different industries are making strategic initiatives. From retail to financial services to healthcare, and many more – machine learning has already become one of the most effective technologies to boost business operations.

4.4Reinforcement learning

Reinforcement learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Example: The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.



The above image shows the robot, diamond, and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that are fired. The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond

Main points in Reinforcement learning –

Input: The input should be an initial state from which the model will start

Output: There are many possible outputs as there are a variety of solutions to a particular problem
Training: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
The model keeps continues to learn.
The best solution is decided based on the maximum reward.
Types of Reinforcement: There are two types of Reinforcement:

Positive –
Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.
Advantages of reinforcement learning are:

Maximizes Performance
Sustain Change for a long period of time
Too much Reinforcement can lead to an overload of states which can diminish the results

Negative –
Negative Reinforcement is defined as strengthening of behavior because a negative condition is stopped or avoided.
Advantages of reinforcement learning:

Increases Behavior
Provide defiance to a minimum standard of performance
It Only provides enough to meet up the minimum behavior

Various Practical applications of Reinforcement Learning –

RL can be used in robotics for industrial automation.
RL can be used in machine learning and data processing
RL can be used to create training systems that provide custom instruction and materials according to the requirement of students.
RL can be used in large environments in the following situations:

A model of the environment is known, but an analytic solution is not available;
Only a simulation model of the environment is given (the subject of simulation-based optimization)
The only way to collect information about the environment is to interact with it.

4.5Decision tree

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent**

the features of a dataset, branches represent the decision rules and **each leaf node represents the outcome.**

In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***

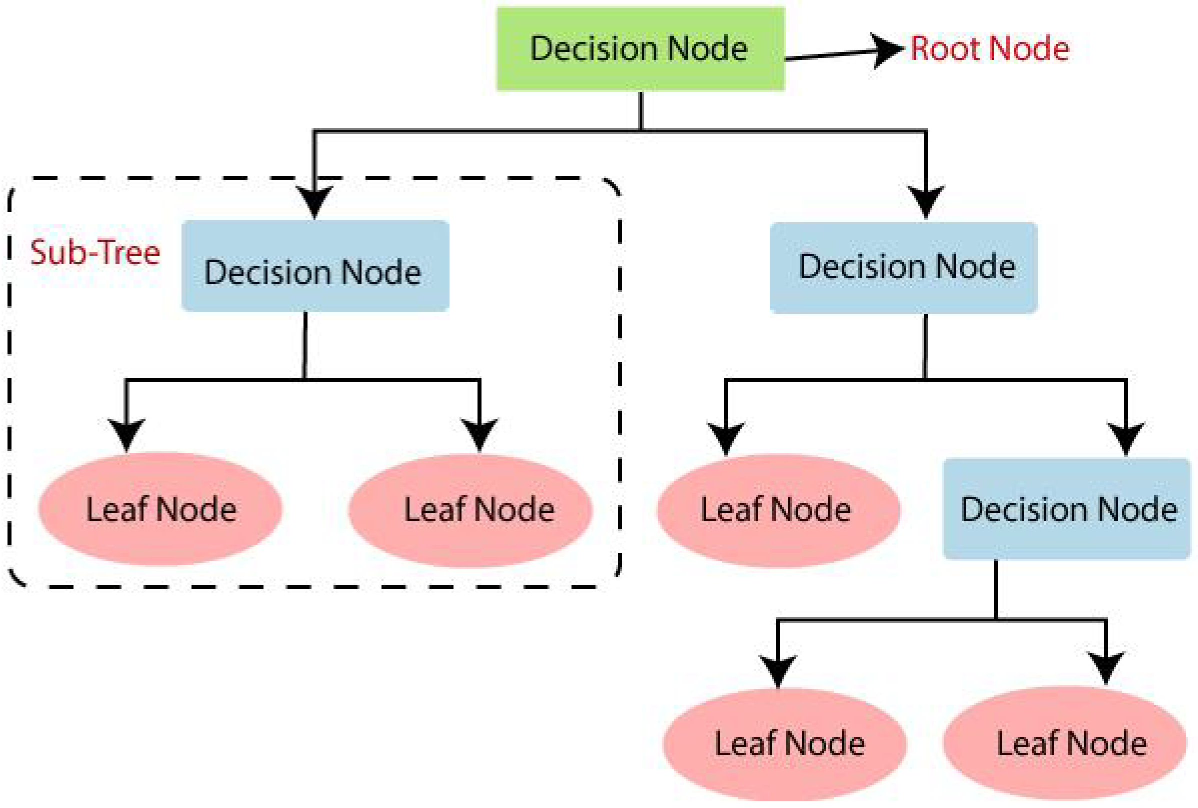
It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Below diagram explains the general structure of a decision tree:

Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.

The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

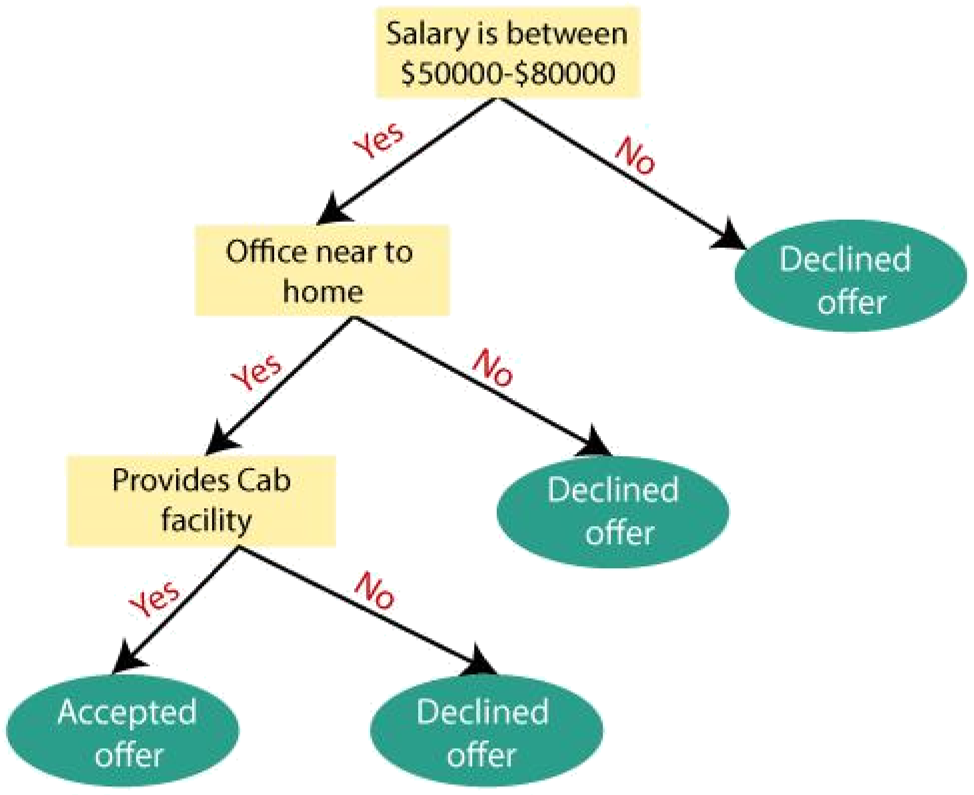
Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



5.Pattern Recognition

5.1Introduction to pattern recognition

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms.
Example: The colors on the clothes, speech pattern, etc. In computer science, a pattern is represented using vector feature values.

Pattern recognition is **the process of recognizing patterns by using a machine learning algorithm**. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation.

One of the important aspects of pattern recognition is its application potential.

Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis.
In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves the classification and cluster of patterns.

In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.

Clustering generated a partition of the data which helps decision making, the specific decision-making activity of interest to us. Clustering is used in unsupervised learning.

Features may be represented as continuous, discrete, or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

Example: consider our face then eyes, ears, nose, etc are features of the face.

A set of features that are taken together, forms the **features vector**.

Example: In the above example of a face, if all the features (eyes, ears, nose, etc) are taken together then the sequence is a feature vector([eyes, ears, nose]). The feature vector is the sequence of a feature represented as a d-dimensional column vector. In the case of speech, MFCC (Mel-frequency Cepstral Coefficient) is the spectral feature of the speech. The sequence of the first 13 features forms a feature vector.

Pattern recognition possesses the following features:

Pattern recognition system should recognize familiar patterns quickly and accurate

Recognize and classify unfamiliar objects

Accurately recognize shapes and objects from different angles

Identify patterns and objects even when partly hidden

Recognize patterns quickly with ease, and with automaticity.

Training and Learning in Pattern Recognition

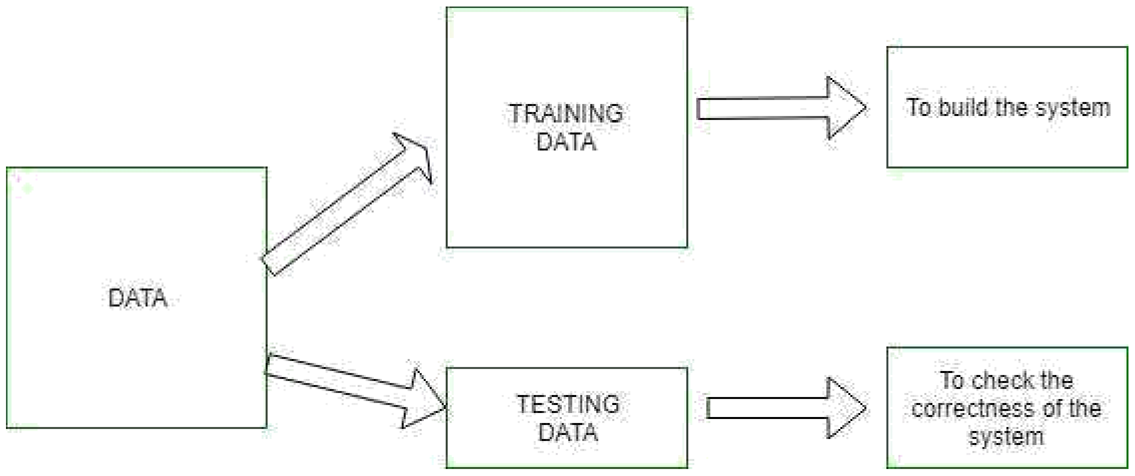
Learning is a phenomenon through which a system gets trained and becomes adaptable to give results in an accurate manner. Learning is the most important phase as to how well the system performs on the data provided to the system depends on which algorithms are used on the data. The entire dataset is divided into two categories, one which is used in training the model i.e. Training set, and the other that is used in testing the model after training, i.e. Testing set.

Training set:

The training set is used to build a model. It consists of the set of images that are used to train the system. Training rules and algorithms are used to give relevant information on how to associate input data with output decisions. The system is trained by applying these algorithms to the dataset, all the relevant information is extracted from the data, and results are obtained. Generally, 80% of the data of the dataset is taken for training data.

Testing set:

Testing data is used to test the system. It is the set of data that is used to verify whether the system is producing the correct output after being trained or not. Generally, 20% of the data of the dataset is used for testing. Testing data is used to measure the accuracy of the system. For example, a system that identifies which category a particular flower belongs to is able to identify seven categories of flowers correctly out of ten and the rest of others wrong, then the accuracy is 70 %



Real-time Examples and Explanations:
A pattern is a physical object or an abstract notion. While talking about the classes of animals, a description of an animal would be a pattern. While talking about various types of balls, then a description of a ball is a pattern. In the case balls considered as pattern, the classes could be football, cricket ball, table tennis ball, etc. Given a new pattern, the class of the pattern is to be determined. The choice of attributes and representation of patterns is a very important step in pattern classification. A good representation is one that makes use of discriminating attributes and also reduces the computational burden in pattern classification.
An obvious representation of a pattern will be a **vector**. Each element of the vector can represent one attribute of the pattern. The first element of the vector will contain the value of the first attribute for the pattern being considered.
Example: While representing spherical objects, (25, 1) may be represented as a spherical object with 25 units of weight and 1 unit diameter. The class label can form a part of the vector. If spherical objects belong to class 1, the vector would be (25, 1, 1), where the first element represents the weight of the object, the second element, the diameter of the object and the third element represents the class of the object.
Advantages:
Pattern recognition solves classification problems
Pattern recognition solves the problem of fake biometric detection.
It is useful for cloth pattern recognition for visually impaired blind people.
It helps in speaker diarization.
We can recognize particular objects from different angles.
Disadvantages:
The syntactic pattern recognition approach is complex to implement and it is a very slow process.
Sometimes to get better accuracy, a larger dataset is required.
It cannot explain why a particular object is recognized.

Image processing, segmentation, and analysis
Pattern recognition is used to give human recognition intelligence to machines that are required in image processing.

Computer vision

Pattern recognition is used to extract meaningful features from given image/video samples and is used in computer vision for various applications like biological and biomedical imaging.

Seismic analysis

The pattern recognition approach is used for the discovery, imaging, and interpretation of temporal patterns in seismic array recordings. Statistical pattern recognition is implemented and used in different types of seismic analysis models.

Radar signal classification/analysis

Pattern recognition and signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.

Speech recognition

The greatest success in speech recognition has been obtained using pattern recognition paradigms. It is used in various algorithms of speech recognition which tries to avoid the problems of using a phoneme level of description and treats larger units such as words as pattern

Fingerprint identification

Fingerprint recognition technology is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches are widely used.

5.2Design Principles of pattern recognition system

In pattern recognition system, for recognizing the pattern or structure two basic approaches are used which can be implemented in diferrent techniques. These are –

Statistical Approach and
Structural Approach

Statistical Approach:

Statistical methods are mathematical formulas, models, and techniques that are used in the statistical analysis of raw research data. The application of statistical methods extracts information from research data and provides different ways to assess the robustness of research outputs.

Two main statistical methods are used :

1.Descriptive Statistics: It summarizes data from a sample using indexes such as the mean or standard deviation.

2.Inferential Statistics: It draw conclusions from data that are subject to random variation.

Structural Approach:

The Structural Approach is a technique wherein the learner masters the pattern of sentence. Structures are the different arrangements of words in one accepted style or the other.

Types of structures:

Sentence Patterns
Phrase Patterns
Formulas
Idioms

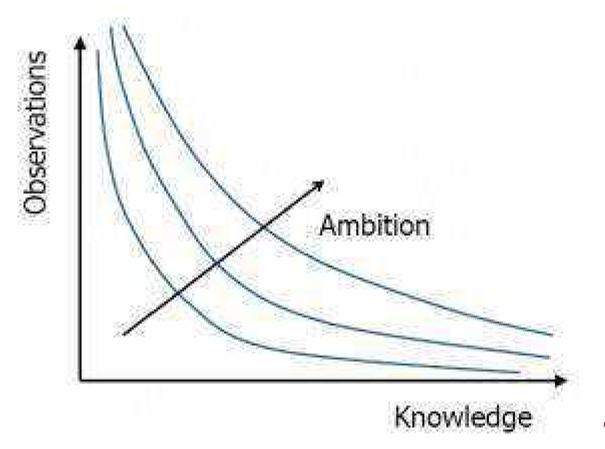
Difference Between Statistical Approach and Structural Approach:

Sr. No.	Statistical Approach	Structural Approach
1	Statistical decision theory.	Human perception and cognition.
2	Quantitative features.	Morphological primitives
3	Fixed number of features.	Variable number of primitives.
4	Ignores feature relationships.	Captures primitives relationships.
5	Semantics from feature position.	Semantics from primitives encoding.
6	Statistical classifiers.	Syntactic grammars.

5.3 Statistical Pattern recognition system

Statistical pattern recognition refers to the use of statistics to learn from examples. It means to collect observations, study and digest them in order to infer general rules or concepts that can be applied to new, unseen observations. How should this be done in an automatic way? What tools are needed?

Previous discussions on prior knowledge and Plato and Aristotle make clear that learning from observations is impossible when context is missing. Context information should be available for learning to occur! Why? Because, otherwise, we don't know where to look for or how define meaningful patterns.Without the reference to the context, there is no way to derive a single general statement on the observations, because many of them are equally possible. **Context is necessary to make a choice.**

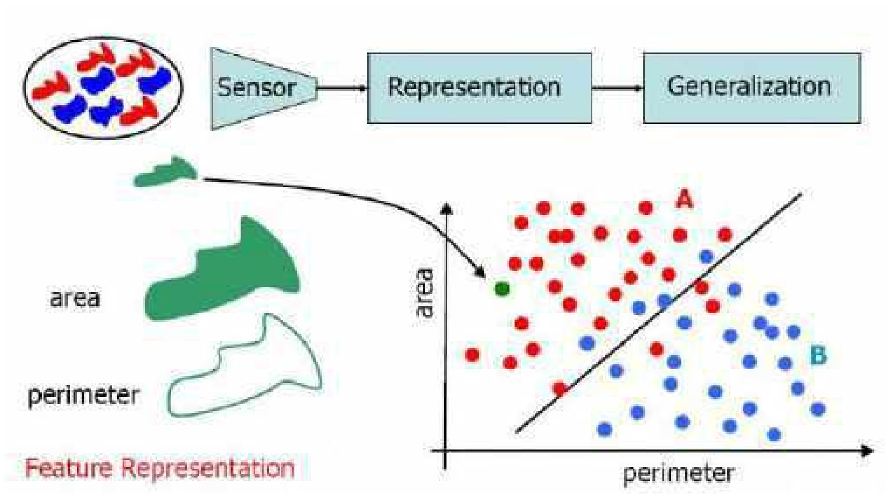


There is a trade-off between how much knowledge that we already have and the number of observations that is needed to gain some specific additional insight. It is shown in the figure on the right. If we know everything no new observations are needed. The less we know in advance the more examples are needed to reach a specific target. This all depends on how ambitious we are in reaching a specific goal.

Discussions like this are very symbolic. Knowledge does not have a size that can be measured. It can at most be ranked partially: from a specific starting point it can grow. Information theory might be helpful when we accept that knowledge can be expressed in bits of information. This implies always an uncertainty. Prior knowledge however usually comes as certain. The expert cannot estimate how convinced he is that he is right. The medical doctor who tells us what to measure where, or what are clear examples of a disease cannot tell what the probability is that this is correct. Moreover, in daily life, we may be absolutely certain after a finite number of observations: we meet somebody and within a second we are sure it is our neighbor. This does not fit in an information theoretic approach.

This is again an aspect of the struggle to learn from examples. Prior knowledge might be wrong, but is is the foundation for new knowledge. For the time being we solve this dilemma by converting existing knowledge into facts of where we are sure of and unknowns between them that have to be uncovered. In the so-called Bayesian approaches it is assumed that some prior probability for the unknowns are known, for sure (!?). The next step in any learning approach, Bayesian or not, is now to bring the known facts, the observations and the unknowns in the same framework, in some mathematical description, by which they can be related. This is called representation.

On the basis of the representation the observations can be related. The pattern recognition task is to generalize these relations to rules that should hold for new observations outside the set of the given ones. The entire process of representation and generalization is illustrated by the below figure. There are some real world objects. The given knowledge is that they come in two different, distinguishable classes, the red ones and the blue ones. It is also given that their size (area) and perimeter length are important for distinguishing them. Examples are given The rule to distinguish them is unknown. It should be found such that it can be applied to new observation for which the class is unknown and has to be estimated.



The objects are represented in a 2-dimensional vector space. Every object is represented as a point (a vector) in this space. It appears that the training examples of the two classes are represented in almost separable regions in this space. A straight line may serve as a decision boundary. It may be applied to new objects with unknown class membership to estimate the class they belong to.

In this example still some training objects go wrong. So it is to be expected that the classification rule is not perfect and that it will make errors for future examples. The main target in the development of a pattern recognition system is to make this error, as small as possible. Additional targets may be the minimization of the cost of learning and classification. To achieve these targets we need to study:

representation: the way the objects are related, in this case by a 2-dimensional vector space.
generalization: the rules and concepts that can be derived from the given representation of the set of examples.
evaluation: accurate and trustworthy estimates of the performance of the system.

5.4 Machine perception

Machine Perception refers to the added functionality in computer systems that enables reaction based on senses, similar to human perception. Computers now have the added capacity to see, hear, touch, and in some cases even smell. The goal of this functionality is to learn and react as a human would, so the computer can alert human operators to any impending issues and help troubleshoot.

Computer vision, sometimes called machine vision, refers to the way in which computers analyze and interpret images or videos. Obtaining and understanding images is a functionality used quite often in this digital revolution for facial recognition software and image classification through convolutional neural networks (CNN). Machine hearing is the computer's ability to decipher sounds, such as speech and music, and process the sound data. This is used for recording music and in voice recognition software in cars and on smartphones. Machine touch generally attempts to gain information based on tactile interaction with physical surroundings. This functionality is less widely used, as recreating a real-world physical reaction in an artificial intelligence (AI) capacity has not yet been fully realized. Similarly, machine smell, or olfaction, is still in its early stages. The intended use of machine olfaction is for chemical analysis and necessary alerts.

Machine learning refers to the overall data analysis that improves over time as it "learns", but machine perception specifically involves the human senses and their capacity to receive and process information. Whether the incoming data is a face or an image or a string of music notes, object recognition and analysis are improving daily. As each new set of data adds on, the system as a whole becomes more appropriately reactive and even predictive. Fully realizing the benefits of machine learning requires analysis through all human senses and how they continuously learn, grow, and react to incoming information.

What are the business advantages of using Machine Perception?

Predictive functionality: Accessing data that is processed through human-like senses is the closest alternative to consumer testing. Machine perception can help a business predict how a consumer or user will see, hear, and experience a new product, site, or service.

Robotics: Machines with robotic capabilities are advancing the manufacturing and production sectors, but organizations can significantly reduce the number of malfunctions with the added capabilities of machine vision or tactile responsiveness. Smarter robots that can detect visible errors and respond to equipment failure can save the organization costly repairs and replacements.

Accuracy: Collecting and analyzing data with computational methods is an exact science. Even analyzing through models based on human senses will be more accurate than human analysis alone.

Efficiency and productivity: Computer analysis and computer processing are much faster than human employees can physically function. Reducing the number of error-prone tasks that are carried out by humans will reduce both errors and time spent.

Recommendations: Machine perception empowers the use of predictive analytics, but aside from predicting customer reactions, businesses can also forecast what consumers will like and buy. This provides an additional opportunity for revenue by recommending additional products and services based on data-backed customer preferences.

5.5 Line finding and interception

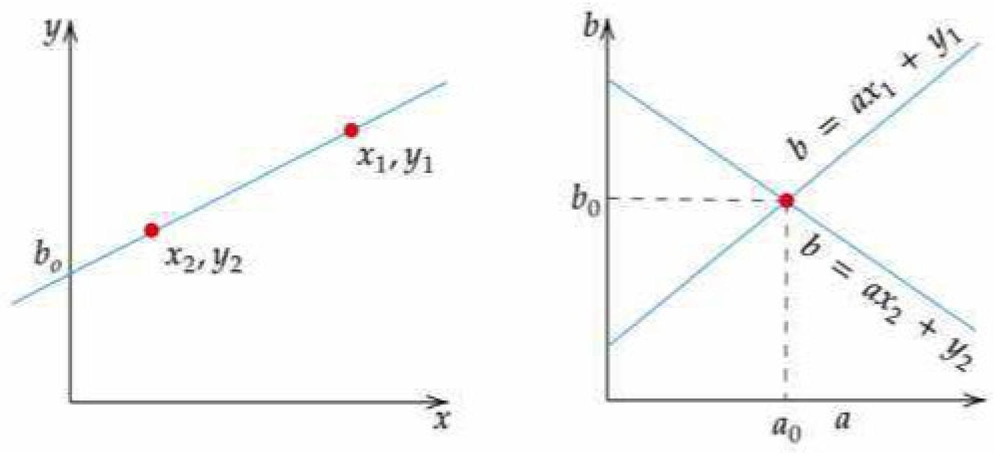
The Hough Transform

The Hough Transform is an algorithm patented by Paul V. C. Hough and was originally invented to recognize complex lines in photographs (Hough, 1962). Since its inception, the algorithm has been modified and enhanced to be able to recognize other shapes such as circles and quadrilaterals of specific types. In order to understand how the Hough Transform algorithm works, it is important to understand four concepts: edge image, the Hough Space, and the mapping of edge points onto the Hough Space, an alternate way to represent a line, and how lines are detected.

Edge Image

An edge image is the output of an edge detection algorithm. An edge detection algorithm detects edges in an image by determining where the brightness/intensity of an image changes drastically ("Edge Detection — Image Processing with Python", 2020). Examples of edge detection algorithms include: Canny, Sobel, Laplacian, etc. It is common for an edge image to be binarized meaning all of its pixel values are either a 1 or a 0. Depending on your situation, either a 1 or a 0 can signify an edge pixel. For the Hough Transform algorithm, it is crucial to perform edge detection first to produce an edge image which will then be used as input into the algorithm.

The Hough Space and the Mapping of Edge Points onto the Hough Space



The Hough Space is a 2D plane that has a horizontal axis representing the slope and the vertical axis representing the intercept of a line on the edge image. A line on an edge image is represented in the form of $y = ax + b$ (Hough, 1962). One line on the edge image produces a point on the Hough Space since a line is characterized by its slope a and intercept b. On the other hand, an edge point (x_i, y_i) on the edge image can have an infinite number of lines pass through it. Therefore, an edge point produces a line in the Hough Space in the form of $b = ax_i + y_i$ (Leavers, 1992). In the Hough Transform algorithm, the Hough Space is used to determine whether a line exists in the edge image.

An Alternate Way to Represent a Line

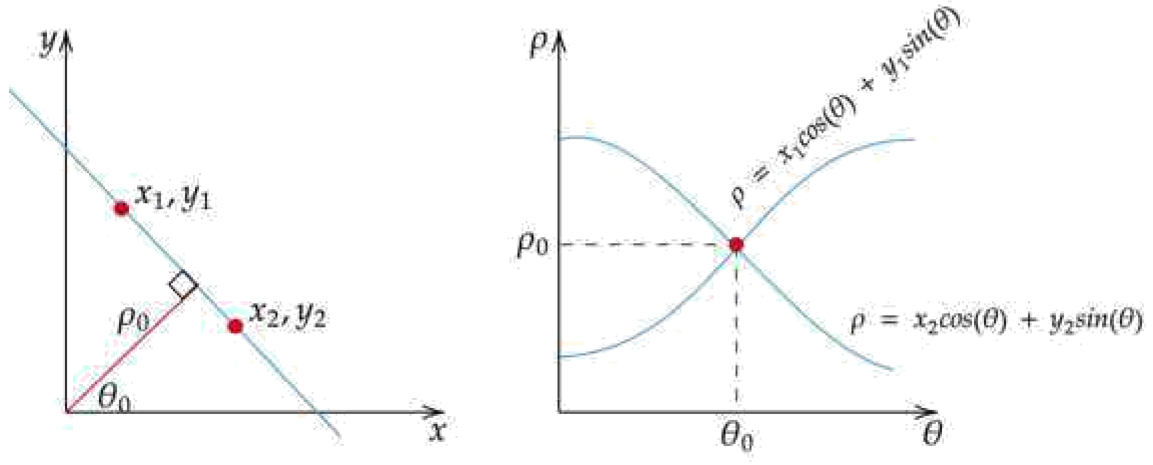
$$m = \frac{rise}{run} = \frac{y_2 - y_1}{x_2 - x_1}$$

m = slope

(x_1, y_1) = first point

(x_2, y_2) = second point

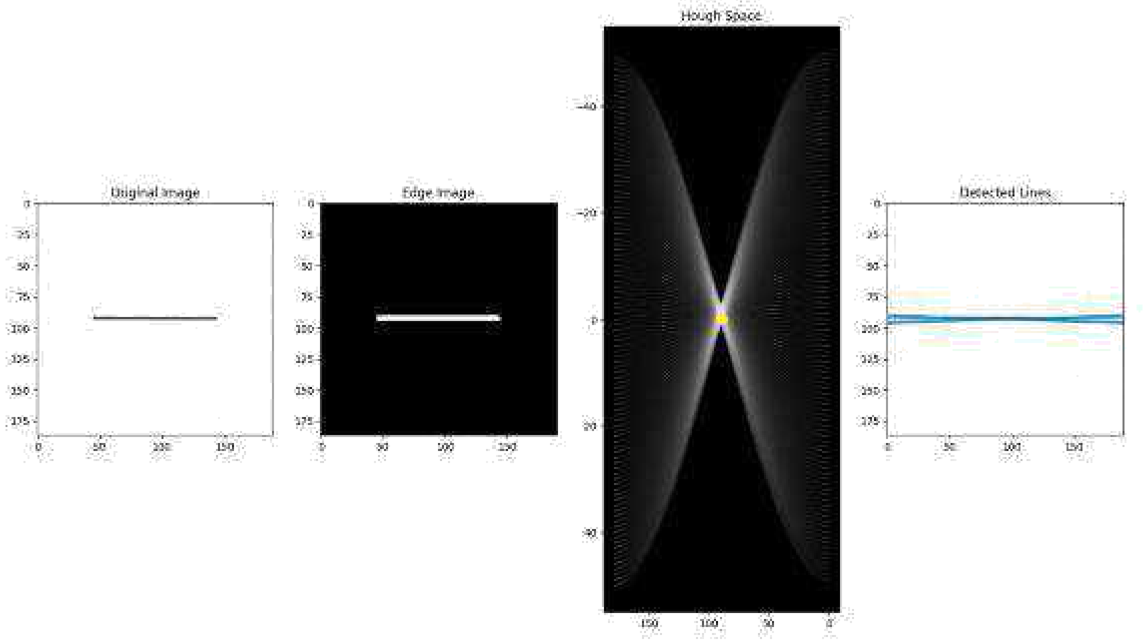
There is one flaw with representing lines in the form of $y = ax + b$ and the Hough Space with the slope and intercept. In this form, the algorithm won't be able to detect vertical lines because the slope a is undefined/infinity for vertical lines (Leavers, 1992). Programmatically, this means that a computer would need an infinite amount of memory to represent all possible values of a. To avoid this issue, a straight line is instead represented by a line called the normal line that passes through the origin and perpendicular to that straight line. The form of the normal line is $\rho = x\cos(\theta) + y\sin(\theta)$ where ρ is the length of the normal line and θ is the angle between the normal line and the x axis.



y
x_1, y_1
x_2, y_2
ρ_0
θ_0
x
ρ
$\rho = x_1\cos(\theta) + y_1\sin(\theta)$
ρ_0
$\rho = x_2\cos(\theta) + y_2\sin(\theta)$
θ_0
θ

Using this, instead of representing the Hough Space with the slope *a* and intercept *b*, it is now represented with ρ and θ where the horizontal axis is for the θ values and the vertical axis are for the ρ values. The mapping of edge points onto the Hough Space works in a similar manner except that an edge point (x_i, y_i) now generates a cosine curve in the Hough Space instead of a straight line (Leavers, 1992). This normal representation of a line eliminates the issue of the unbounded value of *a* that arises when dealing with vertical lines.

Line Detection



As mentioned, an edge point produces a cosine curve in the Hough Space. From this, if we were to map all the edge points from an edge image onto the Hough Space, it will generate a lot of cosine curves. If two edge points lay on the same line, their corresponding cosine curves will intersect each other on a specific (ρ, θ) pair. Thus, the Hough Transform algorithm detects lines by finding the (ρ, θ) pairs that have a number of intersections larger than a certain threshold. It is worth noting that this method of thresholding might not always yield the best result without doing some preprocessing like neighborhood suppression on the Hough Space to remove similar lines in the edge image.

6.Expert system

6.1Introduction to expert system

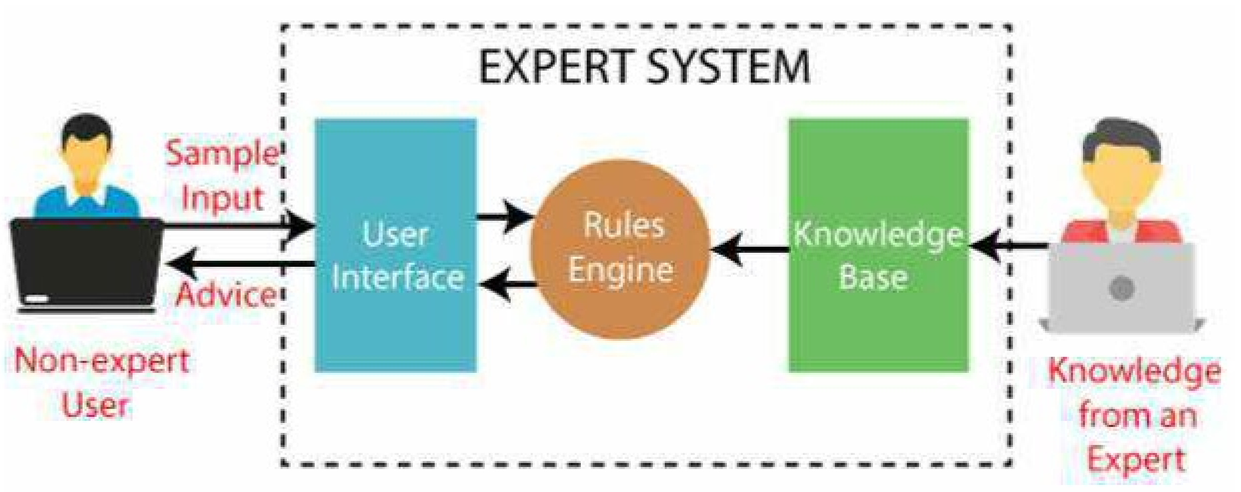
What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for compsex problems using **both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science,** etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

Below is the block diagram that represents the working of an expert system:



Note: It is important to remember that an expert system is not used to replace the human experts; instead, it is used to assist the human in making a complex decision. These systems do not have human capabilities of thinking and work on the basis of the knowledge base of the particular domain.

Below are some popular examples of the Expert System:

DENDRAL: It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.

MYCIN: It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.

PXDES: It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.

CaDeT: The CaDet expert system is a diagnostic support system that can detect cancer at early stages.

Characteristics of Expert System

High Performance: The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.

Understandable: It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.

Reliable: It is much reliable for generating an efficient and accurate output.

Highly responsive: ES provides the result for any complex query within a very short period of time.

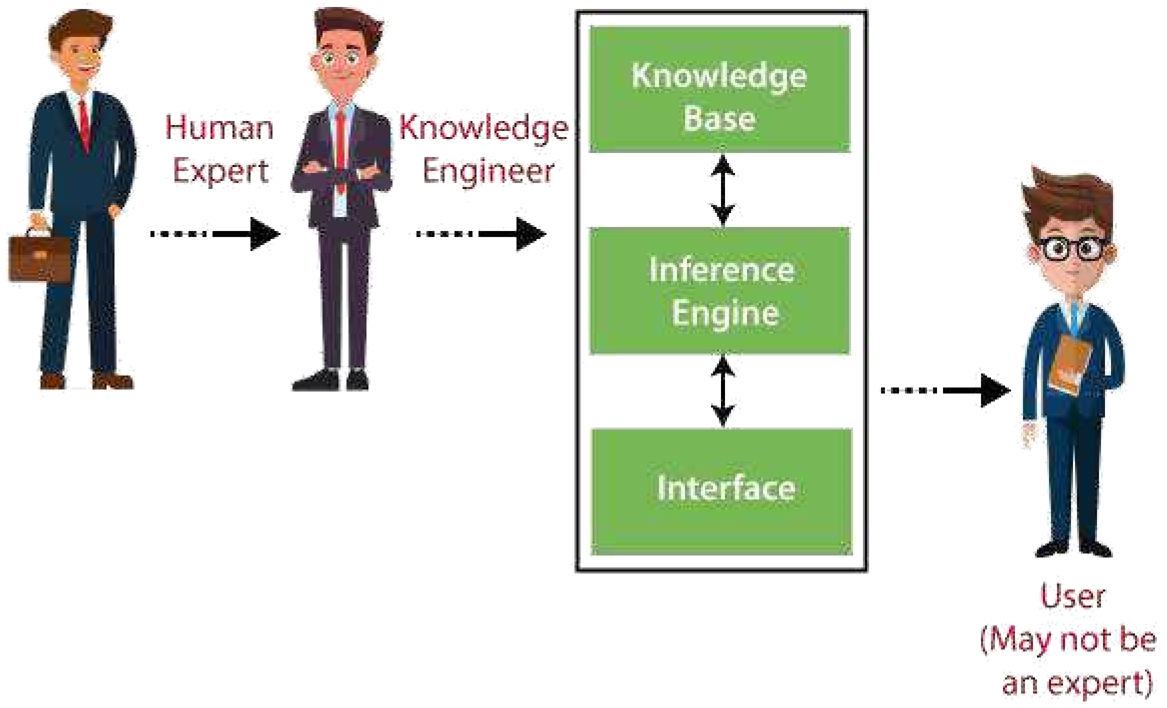
Components of Expert System

An expert system mainly consists of three components:

User Interface

Inference Engine

Knowledge Base



1. User Interface

With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, **it is an interface that helps a non-expert user to communicate with the expert system to find a solution**.

2. Inference Engine(Rules of Engine)

The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.

With the help of an inference engine, the system extracts the knowledge from the knowledge base.

There are two types of inference engine:

Deterministic Inference engine: The conclusions drawn from this type of inference engine are assumed to be true. It is based on **facts** and **rules**.

Probabilistic Inference engine: This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

Forward Chaining: It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.

Backward Chaining: It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

Knowledge Base

The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.

It is similar to a database that contains information and rules of a particular domain or subject.

One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

Factual Knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

Heuristic Knowledge: This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

6.2Basic Architecture of expert system

Typical expert system architecture is shown in Figure 11.1.

The knowledge base contains the specific domain knowledge that is used by an expert to derive conclusions from facts.

In the case of a rule-based expert system, this domain knowledge is expressed in the form of a series of rules.

The explanation system provides information to the user about how the inference engine arrived at its conclusions. This can often be essential, particularly if the advice being given is of a critical nature, such as with a medical diagnosis system.

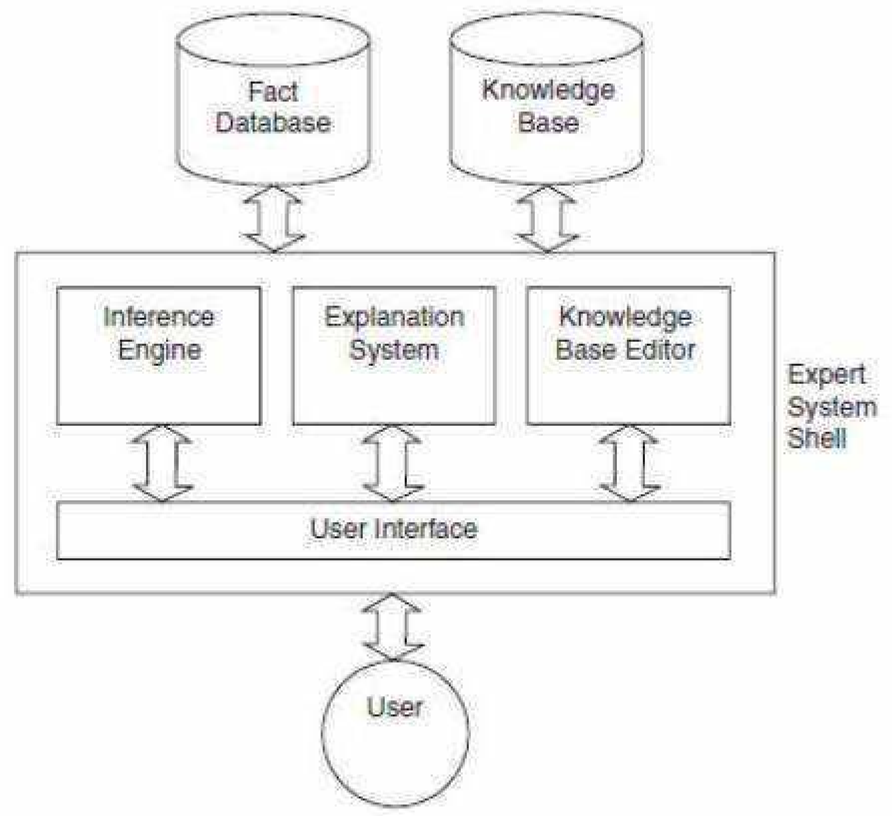


Fig Expert System Architecture

If the system has used faulty reasoning to arrive at its conclusions, then the user may be able to see this by examining the data given by the explanation system.

□

The fact database contains the case-specific data that are to be used in a particular case to derive a conclusion.

□

In the case of a medical expert system, this would contain information that had been obtained about the patient's condition.

The user of the expert system interfaces with it through a user interface, which provides access to the inference engine, the explanation system, and the knowledge-base editor.

□

The inference engine is the part of the system that uses the rules and facts to derive conclusions. The inference engine will use forward chaining, backward chaining, or a combination of the two to make inferences from the data that are available to it.

The knowledge-base editor allows the user to edit the information that is contained in the knowledge base.

The knowledge-base editor is not usually made available to the end user of the system but is used by the knowledge engineer or the expert to provide and update the knowledge that is contained within the system.

6.3Types of problem solved by expert system

The expert systems are capable of −

Advising

Instructing and assisting a human in decision making

Demonstrating

Deriving a solution

Diagnosing

Explaining

Interpreting input

Predicting results

Justifying the conclusion

6.4Features of an expert system

Human experts are perishable, but an expert system is permanent.
It helps to distribute the expertise of a human.

One expert system may contain knowledge from more than one human experts thus making the solutions more efficient.
It decreases the cost of consulting an expert for various domains such as medical diagnosis.
They use a knowledge base and inference engine.
Expert systems can solve complex problems by deducing new facts through existing facts of knowledge, represented mostly as if-then rules rather than through conventional procedural code.
Expert systems were among the first truly successful forms of artificial intelligence (AI) software.

6.5 Expert system architecture

Expert architecture is internally structure that represents to the knowledge base has the certain domain knowledge that is implemented by an expert to server conclusion from facts.

The architecture of an expert system can be better understood by comparing it with a conventional programming.

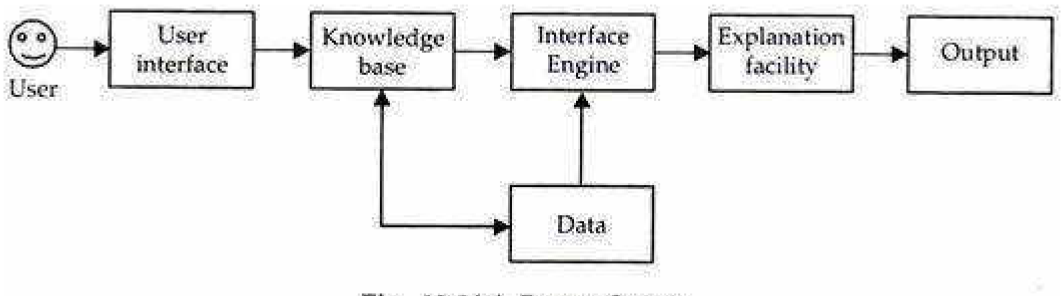


Fig. 12.3(*a*) *Expert System.*

The main components in an Expert System Software are knowledge base, inference engine including explanation facility, user interface mechanism and interfacing, whereas the main components of a conventional software are data (database), program code, interpreter/compiler though not obvious to the user and sparse user-interface.

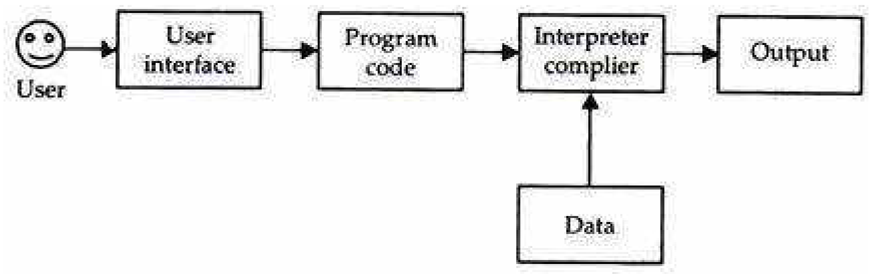


Fig. 12.3(*b*) *Conventional Software program.*

6.6 Expert system tools

The expert system tools are programming systems which simplify the job of constructing an expert system. They range from very high-level programming language to low-level support facilities. We divide expert system tools into four major categories as shown in Fig. 12.12. We will describe these tools and explain briefly how they are used.

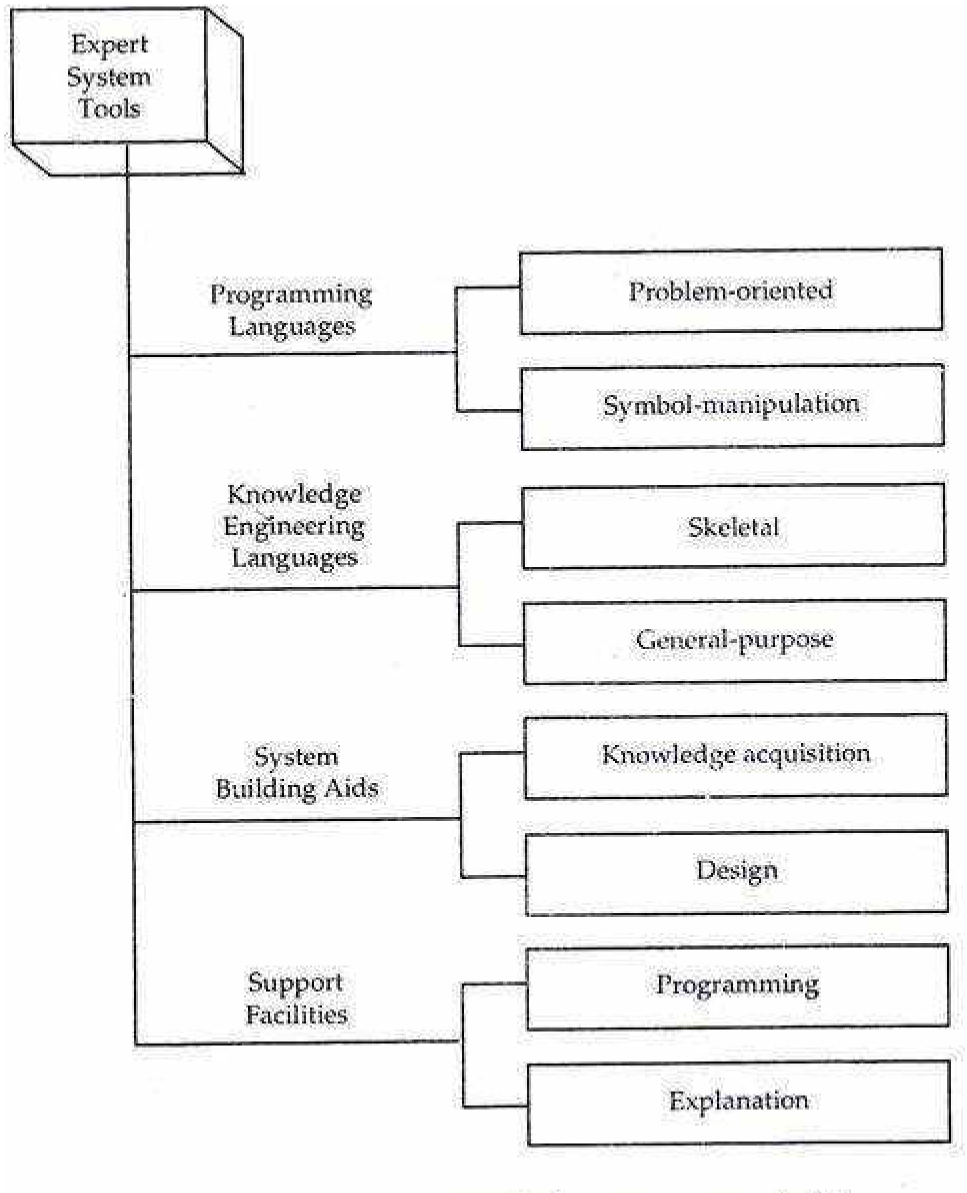


Fig. 12.12. *Types of tools available for expert system building*

6.7 Existing expert systems

There are mainly five types of expert systems. They are **rule based expert system, frame based expert system, fuzzy expert system, neural expert system and neuro-fuzzy expert system.**

6.8 Applications of expert system technology

Some popular Application of Expert System:

Information management
Hospitals and medical facilities
Help desks management
Employee performance evaluation
Loan analysis
Virus detection
Useful for repair and maintenance projects
Warehouse optimization
Planning and scheduling
The configuration of manufactured objects
Financial decision making Knowledge publishing
Process monitoring and control
Supervise the operation of the plant and controller
Stock market trading
Airline scheduling & cargo schedules