KGRL PG COURSES

BHIMAVARAM

ARTIFICIAL INTELLIGENCE

BY

K.ISSACK BABU

Assistant Professor

UNIT –I

1.Artificial Intelligence

- ❑ Modeling human cognition or mental faculties using computers
- ❑ Study of making computers do things which at the moment people are better
- ❑ Making computers do things which require intelligence

1.1More Formal Definition of AI

- AI is a branch of computer science which is concerned with the study and creation of computer systems that exhibit
  - ❑ some form of intelligence

  OR

  - ❑ those characteristics which we associate with intelligence in human behavior
  - ❑ AI is a broad area consisting of different fields, from machine vision, expert systems to the creation of machines that can "think".
  - ❑ In order to classify machines as "thinking", it is necessary to define frontier

3. What is Intelligence?

- Intelligence is a property of mind that encompasses many related mental abilities, such as the capabilities to
  - ❑ reason
  - ❑ plan
  - ❑ solve problems
  - ❑ think abstractly
  - ❑ comprehend ideas and language and
  - ❑ learn

3.1. Characteristics of AI systems

- ■ learn new concepts and tasks
- ■ reason and draw useful conclusions about the world around us

    - remember complicated interrelated facts and draw conclusions from them (inference)
- understand a natural language or perceive and comprehend a visual scene
    - look through cameras and see what's there (vision), to move themselves and objects around in the real world (robotics)
- plan sequences of actions to complete a goal
- offer advice based on rules and situations
- may not necessarily imitate human senses and thought processes
    - but indeed, in performing some tasks differently, they may actually exceed human abilities
- capable of performing intelligent tasks effectively and efficiently
- perform tasks that require high levels of intelligence

3.2. Understanding of AI

- AI techniques and ideas seem to be harder to understand than most things in computer science
- AI shows best on complex problems for which general principles don't help much, though there are a few useful general principles
- Artificial intelligence is also difficult to understand by its content.
- Boundaries of AI are not well defined.
- Often it means the advanced software engineering, sophisticated software techniques for hard problems that can't be solved in any easy way.
- AI programs - like people - are usually not perfect, and even make mistakes.
- It often means, nonnumeric ways of solving problems, since people can't handle numbers well.
- Nonnumeric ways are generally "common sense" ways, not necessarily the best ones.
- Understanding of AI also requires an understanding of related terms such as intelligence, knowledge, reasoning, thought, cognition, learning, and a number of other computer related terms.

3.3. Categories of AI System

- Systems that think like humans
- Systems that act like humans

- Systems that think rationally
- Systems that act rationally

3.3.1.Systems that think like humans

- Most of the time it is a black box where we are not clear about our thought process.
- One has to know functioning of brain and its mechanism for prossessing information.
- It is an area of cognitive science.
  - The stimuli are converted into mental representation.
  - Cognitive processes manipulate representation to build new representations that are used to generate actions.
- Neural network is a computing model for processing information similar to brain.

3.3.2.Systems that act like humans

- The overall behaviour of the system should be human like.
- It could be achieved by observation

3.3.3.Systems that think rationally

- Such systems rely on logic rather than human to measure correctness.
- For thinking rationally or logically, logic formulas and theories are used for synthesizing outcomes.
- For example,
  - given John is a human and all humans are mortal then one can conclude logically that John is mortal
- Not all intelligent behavior are mediated by logical deliberation.

3.3.4.Systems that act rationally

- Rational behavior means doing right thing.
- Even if method is illogical, the observed behavior must be rational.

Foundations of Artificial Intelligence:

- Philosophy

  e.g., foundational issues (can a machine think?), issues of knowledge and believe, mutualknowledge

Psychology and Cognitive Science

e.g., problem solving skills

- Neuro-Science

   e.g., brain architecture

- Computer Science And Engineering

   e.g., complexity theory, algorithms, logic and inference, programming languages, and systembuilding.

- Mathematics and Physics

   e.g., statistical modeling, continuous mathematics,

- Statistical physics and complex

- Sub Areas of AI:

1) **Game Playing**

   Deep Blue Chess program beat world champion Gary Kasparov

2) Speech Recognition

   PEGASUS spoken language interface to American Airlines' EAASY SABRE reseration system, which allows users to obtain flight information and make reservations over the telephone. The 1990s hasseen significant advances in speech recognition so that limited systems are now successful.

3) Computer Vision

   Face recognition programs in use by banks, government, etc. The ALVINN system from CMU autonomously drove a van from Washington, D.C. to San Diego (all but 52 of 2,849 miles), averaging63 mph day and night, and in all weather conditions. Handwriting recognition, electronics and manufacturing inspection, photo interpretation, baggage inspection, reverse engineering to automatically construct a 3D geometric model.

4) Expert Systems

   Application-specific systems that rely on obtaining the knowledge of human experts in an area andprogramming that knowledge into a system.

   a. **Diagnostic Systems** : MYCIN system for diagnosing bacterial infections of the blood and suggesting treatments. Intellipath pathology diagnosis system

(AMA approved). Pathfinder medical diagnosis system, which suggests tests and makes diagnoses. Whirlpool customer assistance center.

b. System Configuration

DEC's XCON system for custom hardware configuration. Radiotherapy treatment planning.

c. Financial Decision Making

Credit card companies, mortgage companies, banks, and the U.S. government employ AI systems to detect fraud and expedite financial transactions. For example, AMEX credit check.

d. Classification Systems

Put information into one of a fixed set of categories using several sources of information. E.g., financial decision making systems. NASA developed a system for classifying very faint areas in astronomical images into either stars or galaxies with very high accuracy by learning from human experts' classifications.

5) Mathematical Theorem Proving

Use inference methods to prove new theorems.

6) Natural Language Understanding

AltaVista's translation of web pages. Translation of Catepillar Truck manuals into 20 languages.

7) Scheduling and Planning

Automatic scheduling for manufacturing. DARPA's DART system used in Desert Storm and Desert Shield operations to plan logistics of people and supplies. American Airlines rerouting contingency planner. European space agency planning and scheduling of spacecraft assembly, integration and verification.

8) Artificial Neural Networks:

9)Machine learning

Application of AI::

AI algorithms have attracted close attention of researchers and have also been applied successfully to solve problems in engineering. Nevertheless, for large and complex problems, AI algorithms consume considerable computation time due to stochastic feature of the search approaches

1) Business; financial strategies
2) Engineering: check design, offer suggestions to create new product, expert systems for all engineering problems
3) Manufacturing: assembly, inspection and maintenance
4) Medicine: monitoring, diagnosing
5) Education: in teaching
6) Fraud detection
7) Object identification
8) Information retrieval
9) Space shuttle scheduling

Building AI Systems:

1) Perception

   Intelligent biological systems are physically embodied in the world and experience the world through their sensors (senses). For an autonomous vehicle, input might be images from a camera and range information from a rangefinder. For a medical diagnosis system, perception is the set of symptoms and test results that have been obtained and input to the system manually.

2) Reasoning

   Inference, decision-making, classification from what is sensed and what the internal "model" is of the world. Might be a neural network, logical deduction system, Hidden Markov Model induction, heuristic searching a problem space, Bayes Network inference, genetic algorithms, etc.

   Includes areas of knowledge representation, problem solving, decision theory, planning, game theory, machine learning, uncertainty reasoning, etc.

3) Action

   Biological systems interact within their environment by actuation, speech, etc. All behavior is centered around actions in the world. Examples include controlling the steering of a Mars rover or autonomous vehicle, or suggesting tests and making diagnoses for a medical diagnosis system. Includes areas of robot actuation, natural language generation, and speech synthesis.

Goals of Artificial Intelligence

Following are the main goals of Artificial Intelligence:

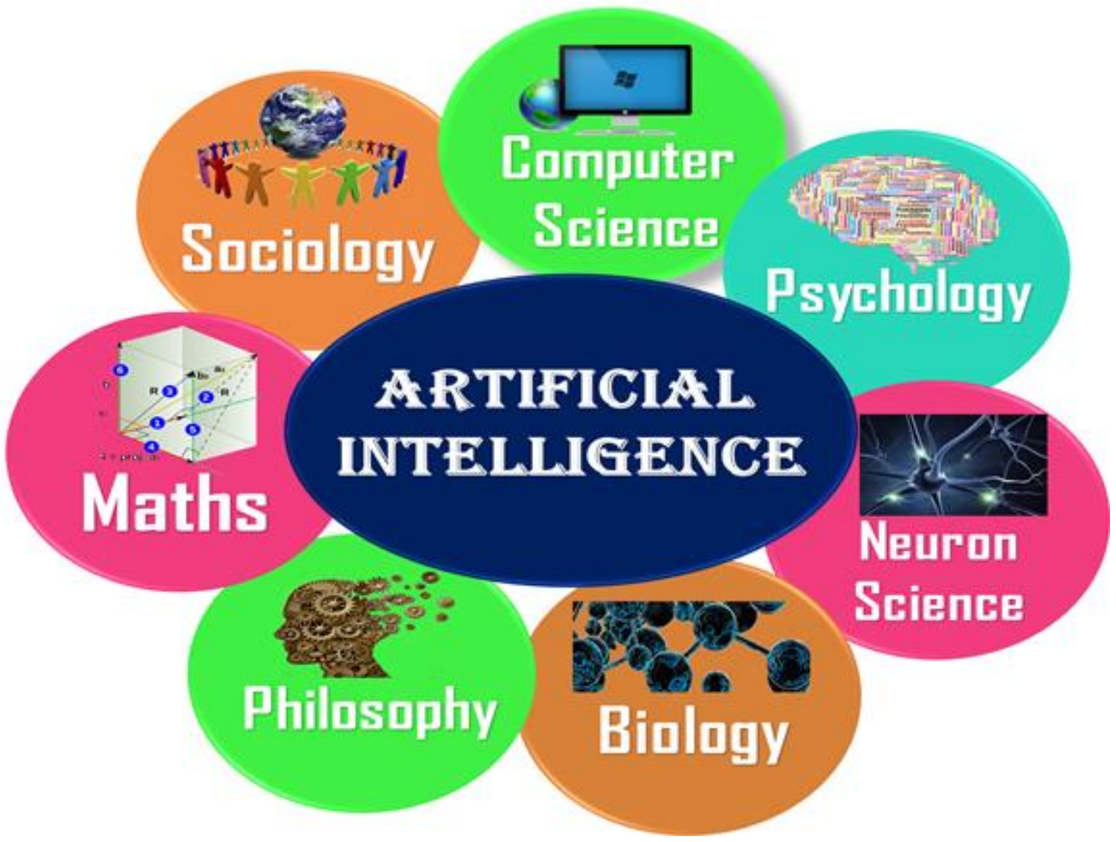
1. Replicate human intelligence
2. Solve Knowledge-intensive tasks
3. An intelligent connection of perception and action
4. Building a machine which can perform tasks that requires human intelligence such as:
    - Proving a theorem
    - Playing chess
    - Plan some surgical operation
    - Driving a car in traffic
5. Creating some system which can exhibit intelligent behavior, learn new things by itself, demonstrate, explain, and can advise to its user.

What Comprises to Artificial Intelligence?

Artificial Intelligence is not just a part of computer science even it's so vast and requires lots of other factors which can contribute to it. To create the AI first we should know that how intelligence is composed, so the Intelligence is an intangible part of our brain which is a combination of **Reasoning, learning, problem-solving perception, language understanding, etc**.

To achieve the above factors for a machine or software Artificial Intelligence requires the following discipline:

- Mathematics
- Biology
- Psychology
- Sociology
- Computer Science
- Neurons Study
- Statistics



Advantages of Artificial Intelligence

Following are some main advantages of Artificial Intelligence:

- **High Accuracy with less errors:** AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
- **High-Speed:** AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.
- **High reliability:** AI machines are highly reliable and can perform the same action multiple times with high accuracy.
- **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
- **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.

- **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security purpose, Natural language processing to communicate with the human in human-language, etc.

Disadvantages of Artificial Intelligence

Every technology has some disadvantages, and thesame goes for Artificial intelligence. Being so advantageous technology still, it has some disadvantages which we need to keep in our mind while creating an AI system. Following are the disadvantages of AI:

- **High Cost:** The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
- **Can't think out of the box:** Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
- **No feelings and emotions:** AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
- **Increase dependency on machines:** With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
- **No Original Creativity:** As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

Top 4 Techniques of Artificial Intelligence

Artificial Intelligence can be divided into different categories based on the machine’s capacity to use past experiences to predict future decisions, memory, and self-awareness. IBM came up with Deep Blue, a chess program that can identify the pieces in the chessboard. But it does not have the memory to predict future actions. This system though useful, but it cannot be adapted to another situation. Another type of AI system that uses past experiences and has the bonus of a limited memory to predict the decisions. An example of this kind of AI system can be found in the functions of decision making in case of the self-driving cars. Here the observations help in the actions to be taken shortly, which does not get stored permanently as the observations change frequently. At the same time with the advancement in technology, it might be possible to have machines with a sense or consciousness where the machines understand the current state of things, which can be used to infer what is to be done. But such systems do not exist.

Below are the various categories of Artificial I

1. Machine Learning :: It is one of the applications of AI where machines are not explicitly programmed to perform certain tasks; rather, they learn and improve from experience automatically. Deep Learning is a subset of machine learning based on artificial neural networks for predictive analysis. There are various machine learning algorithms, such as Unsupervised Learning, Supervised Learning, and Reinforcement Learning. In Unsupervised Learning, the algorithm does not use classified information to act on it without any guidance. In Supervised Learning, it deduces a function from the training data, which consists of a set of an input object and the desired output. Reinforcement learning is used by machines to take suitable actions to increase the reward to find the best possibility which should be taken in to account.

2. NLP (Natural Language Processing)

It is the interactions between computers and human language where the computers are programmed to process natural languages. Machine Learning is a reliable technology for Natural Language Processing to obtain meaning from human languages. In NLP, the audio of a human talk is captured by the machine. Then the audio to text conversation occurs, and then the text is processed where the data is converted into audio. Then the machine uses the audio to respond to humans. Applications of Natural Language Processing can be found in IVR (Interactive Voice Response) applications used in call centres, language translation applications like Google Translate and word processors such as Microsoft Word to check the accuracy of grammar in text. However, the nature of human languages makes the Natural Language Processing difficult because of the rules which are involved in the passing of information using natural language, and they are not easy for the computers to understand. So NLP uses algorithms to recognize and abstract the rules of the natural languages where the unstructured data from the human languages can be converted to a format that is understood by the computer.

3. Automation and Robotics

The purpose of Automation is to get the monotonous and repetitive tasks done by machines which also improve productivity and in receiving cost-effective and more efficient results. Many organizations use machine learning, neural networks, and graphs in automation. Such automation can prevent fraud issues while financial transactions online by using CAPTCHA technology. Robotic process automation is programmed to perform high volume repetitive tasks which can adapt to the change in different circumstances.

4. Machine Vision

Machines can capture visual information and then analyze it. Here cameras are used to capture the visual information, the analogue to digital conversion is used to convert the image to digital data, and digital signal processing is employed to process the data .Then the resulting data is fed to a computer.In machine vision, two vital aspects are sensivity, which is the ability of the machine to perceive impulses that are weak and resolution,the range to which the machine can distinguish the objects. The usage of machine vision can be found in signature identification ,pattern recognition, and medical image analysis , etc.

ASSUMPTIONS OF AI ::

1. An AI strategy is not one size fits all

I can’t tell you how many times I’ve met a CIO or CTO who wanted to know what our AI strategy was. I usually try to qualify their interest with a question of my own, “What does AI mean to you?”

That question is typically followed by dead silence. The issue is that CTOs expect the same AI strategy to fit every use case. The real question that should be asked is, “What are customers actually trying to do?”

Usually, the use cases come down to basic machine learning. Customers want a system to watch and learn what their knowledge workers are doing as part of their daily routines – especially routines they are considering automating with robotic process automation (RPA). They want a system that can then start recommending various courses of action based on learned behavior. Finally, they want the knowledge worker to be able to direct the system to automate the learned behavior after getting comfortable with past recommendations.

2. AI and machine learning are not interchangeable

Machine learning is often misinterpreted for AI, as are cognitive processing, natural language processing, and deep learning. Machine learning is an application of AI whereas it gives machines access to data and lets them learn for themselves, thus enabling AI.

It's the combination of RPA and machine learning that organizations interpret as AI. Once organizations have automated various tasks by adding a level of learned intelligence, they'll be able to monitor and understand the impact those efforts are having on their organization via real-time process discovery and process intelligence.

3. Algorithms are not more important than data

This is akin to the chicken-or-the-egg conundrum. Algorithms need data to work, yet algorithms enable you to do whatever you want with data. However, the design of an algorithm is guided by the data structure they are supposed to work with, so in reality, data is more important than the algorithm.

4. AI lacks an important human trait: Empathy

In reality, AI replaces mundane, repetitive and error-prone tasks so humans can focus on value-added processes that require creativity, problem solving and flexibility. However, workers carry unique characteristics like empathy and judgment that robots lack.

5. Expect a cultural and skills shift

Organizations can count on seeing dramatic change within the company culture and employee skillset within the next three to five years. However, embracing AI needs to lead from the CEO and spread throughout the workforce in order to be successful due to the human nature's reluctance to accept change. Having management spearhead the cultural shift will foster acceptance quicker.

From a skills perspective, the introduction of AI into the general workplace will result in more tasks being addressed by system of record applications. For example, in the mortgage lending market, the dependency on a loan origination officer to drive the loan process will lessen due to the loan origination system being able to make intelligent decisions based on past funding behavior. This will leave only rules-based exceptions to require a loan processor's attention. As a result, this will lighten the overall workload for loan officers, allowing them to be more responsive when an exception rises and should allow mortgage lenders to increase the productivity of their operations.

Major Problems Associated with Artificial Intelligence

The following are a few of the major problems associated with Artificial Intelligence and its possible solutions.

1. Job Loss Problem

Job loss concerns related to Artificial Intelligence has been a subject of numerous business cases and academic studies. As per an Oxford Study, more than 47% of American jobs will be under threat due to automation by the mid-2030s. As per the World Economic Forum, Artificial Intelligence automation will replace more than 75 million jobs by 2022. Some of the figures are even more daunting. As per another Mckinsey report, AI-bases robots could replace 30% of the current global workforce. As per the AI expert and Venture Capitalist Kai-Fu Lee, 40% of the world jobs will be replaced by AI-based bots in the next 10-15 years. Low income and low skilled workers will be the worst hit by this change. As the AI becomes smarter by the day even the High paid, High skill workers, become more vulnerable to job losses as, given the high cost of skilled workers, the companies get better margins by automating their work. However, these issues related to Job loss and wages can be addressed by focussing on the following measures.

- Overhauling the education system and giving more focus on skills like Critical Thinking, Creativity, and Innovation as these skills are hard to replicate.
- Increasing both public and private investment in developing human capital so that they are better aligned with industry demand.
- Improving the condition of the labor market by bridging the demand-supply gap and giving impetus to the gig economy.

2. Safety Problem

There has always been much furor about safety issues associated with Artificial Intelligence. When experts like Elon Musk, Stephen Hawking, Bill Gates among various others express concern related to AI safety we should pay heed to its safety issues. There have been various instances where Artificial Intelligence has gone wrong when Twitter Chabot started spewing abusive and Pro-Nazi sentiments and in other instance when Facebook AI bots started interacting with each other in a language no one else would understand, ultimately leading to the project being shut down.

There are grave concerns about Artificial Intelligence doing something harmful to humankind. The case in point is autonomous weapons which can be programmed to kill other humans. There are also imminent concerns with AI forming "Mind of their Own" and doesn't value human life. If such weapons are deployed, it will be very difficult to undo its repercussions. The following are the measures that can be taken to mitigate these concerns.

- We need to have strong regulations especially when it comes to creation or experimentation of Autonomous weapons
- Global Co-operation on issues concerning such kind of weapons is needed so as to ensure no one gets involved in the rat race
- Complete transparency in the system where such technologies have experimented is essential to ensure its safe usage

3. Trust Related Problem

As Artificial Intelligence algorithms become more powerful by the day, it also brings several trust-related issues on its ability to make decisions that are fair and for the betterment of humankind. With AI slowly reaching human-level cognitive abilities the trust issue becomes all the more significant. There are several applications where AI operates as a black box. Example- in High-Frequency trading even the Program developers don’t have a good understanding of the basis on which AI executed the trade. Some more striking examples include Amazon AI-based algorithm for same-day delivery which was inadvertently biased against black neighbourhood, another example was Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) where the Artificial Intelligence algorithm while profiling suspects was biased against the black community.

Following are few of the measures that can be taken to bridge trust-related issues in Artificial Intelligence

- All the major Artificial Intelligence providers need to set up guiding rules and principles related to trust and transparency in AI implementation. These principles need to be religiously followed by all the stakeholders involved in Artificial Intelligence development and usage
- All the stakeholders should be aware of the bias which inherently comes with AI algorithm and should have a robust bias detection mechanism and ways to handle it
- Awareness is another key factor that plays a major role in bridging the trust gap. The users should be sensitized about the AI operations, its capabilities and even the shortfall that is associated with Artificial Intelligence

4. Computation Problem

Artificial Intelligence algorithm involves analyzing the humongous amount of data that require an immense amount of computational power. So far the problem was dealt with with the help of Cloud Computing and Parallel Processing. However, as the amount of data increases and more complex deep learning algorithm comes in the mainstream, the present-day computational power will not be enough to cater to the complex requirement. We will need more storage and computational power which can handle crunching exabytes and Zettabytes of data.

Quantum Computing can address the processing speed problem in the medium to long terms

Quantum computing which is based on concepts of Quantum theory might be the answer to solving computation power challenges. Quantum computing is 100 Million times faster than a normal computer we use at home. Although currently, it is in the research and experimental stage. As per an estimate by different experts, we can see its mainstream implementation in the next 10-15 years.

The aforementioned problems are certainly not impossible to solve, however, it does require rapid evolution in technology as well as human co-operation. While we are well on course in terms of rate of technological advancement but we still have a long way to go to develop principles, methodology, and frameworks to ensure that powerful technology like AI is not misused or misapplied which may result in unintended consequences.

UNIT 2

Generate and Test Search

Introduction:

Generate and Test Search is a heuristic search technique based on Depth First Search with Backtracking which guarantees to find a solution if done systematically and there exists a solution. In this technique, all the solutions are generated and tested for the best solution. It ensures that the best solution is checked against all possible generated solutions.

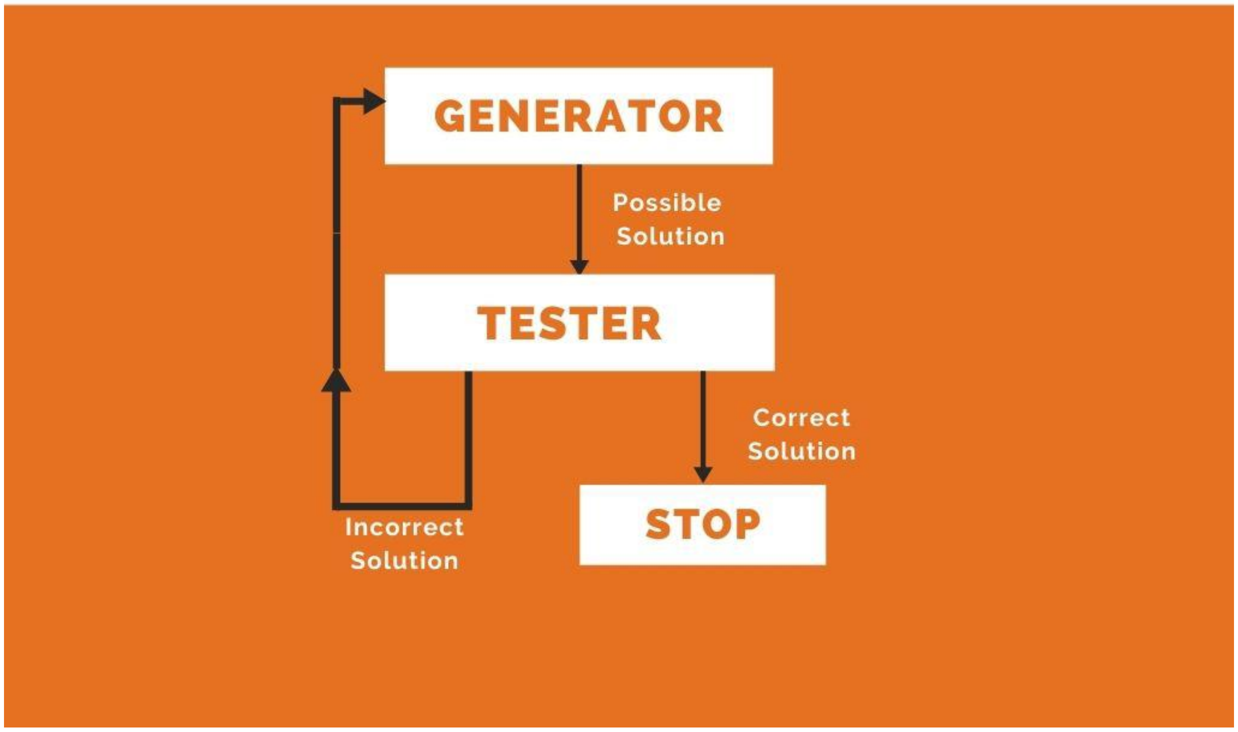
It is also known as British Museum Search Algorithm as it's like looking for an exhibit at random or finding an object in the British Museum by wandering randomly.

The evaluation is carried out by the heuristic function as all the solutions are generated systematically in generate and test algorithm but if there are some paths which are most unlikely to lead us to result then they are not considered. The heuristic does this by ranking all the alternatives and is often effective in doing so. Systematic Generate and Test may prove to be ineffective while solving complex problems. But there is a technique to improve in complex cases as well by combining generate and test search with other techniques so as to reduce the search space. For example in Artificial Intelligence Program DENDRAL we make use of two techniques, the first one is Constraint Satisfaction Techniques followed by Generate and Test Procedure to work on reduced search space i.e. yield an effective result by working on a lesser number of lists generated in the very first step.

Algorithm

1. *Generate a possible solution. For example, generating a particular point in the problem space or generating a path for a start state.*
2. *Test to see if this is a actual solution by comparing the chosen point or the endpoint of the chosen path to the set of acceptable goal states*
3. *If a solution is found, quit. Otherwise go to Step 1*

DIAGRAMMATIC REPRESENTATION

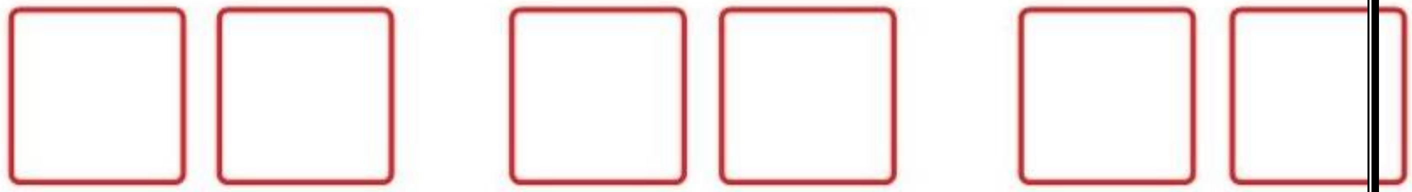


Properties of Good Generators:

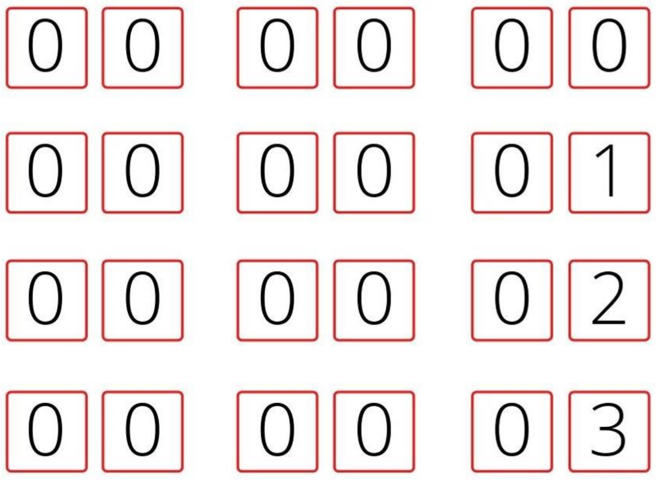
The good generators need to have the following properties:

- ***Complete:*** *Good Generators need to be complete i.e. they should generate all the possible solutions and cover all the possible states. In this way, we can guaranty our algorithm to converge to the correct solution at some point in time.*
- ***Non Redundant:*** *Good Generators should not yield a duplicate solution at any point of time as it reduces the efficiency of algorithm thereby increasing the time of search and making the time complexity exponential. In fact, it is often said that if solutions appear several times in the depth-first search then it is better to modify the procedure to traverse a graph rather than a tree.*
- ***Informed:*** *Good Generators have the knowledge about the search space which they maintain in the form of an array of knowledge. This can be used to search how far the agent is from the goal, calculate the path cost and even find a way to reach the goal.*

Let us take a simple example to understand the importance of a good generator. Consider a pin made up of three 2 digit numbers i.e. the numbers are of the form,



In this case, one way to find the required pin is to generate all the solutions in a brute force manner for example,



The total number of solutions in this case is $(100)^3$ which is approximately 1M. So if we do not make use of any informed search technique then it results in exponential time complexity. Now let's say if we generate 5 solutions every minute. Then the total numbers generated in 1 hour are 5*60=300 and the total number of solutions to be generated are 1M. Let us consider the brute force search technique for example linear search whose average time complexity is N/2. Then on an average, the total number of the solutions to be generated are approximately 5 lakhs. Using this technique even if you work for about 24 hrs a day then also you will need 10 weeks to complete the task.

Now consider using heuristic function where we have domain knowledge that every number is a prime number between 0-99 then the possible number of solutions are $(25)^3$ which is approximately 15,000. Now consider the same case that you are generating 5 solutions every minute and working for 24 hrs then you can find the solution in less than 2 days which was being done in 10 weeks in the case of uninformed search.

We can conclude for here that if we can find a good heuristic then time complexity can be reduced gradually. But in the worst-case time and space complexity will be exponential. It all depends on the generator i.e. better the generator lesser is the time complexity.

Interesting Real Life Examples:

- *If a sufficient number of monkeys were placed in front of a set of typewriters, and left alone long enough, then they would eventually produce all the works of Shakespeare.*
- *Dendral which infers the structure of organic compounds using NMR spectrogram also uses plan-generate-test.*
-

WHAT IS HILL CLIMBING IN ARTIFICIAL INTELLIGENCE?

Another important research area in the field of artificial intelligence, **Hill Climbing algorithm in artificial intelligence** is an important optimization and heuristic search technique, used to generate the most accurate and **optimal solution** for a given problem, leveraging the concept of iteration.

Also considered to be an iterative algorithm, the hill **climbing algorithm** is similar to the greedy local search algorithm, as it explores the **search space** until all possible solutions are generated and it reaches the expected **goal state.** Furthermore, it only considers the current states and not the future states, while searching for the optimal solution and hence requires fewer conditions than other search techniques.

DEFINING HILL CLIMBING ALGORITHM IN ARTIFICIAL INTELLIGENCE WITH EXAMPLE:

The **travelling salesman problem** is the most common example used by people to define the concepts of the Hill Climbing Algorithm, wherein the target is to minimize the distance he travels. However, another example used to define the concepts of this algorithm is **n-queens problems.**

Wherein, different queens are positioned on an n × n board, with none of them sharing the same row, column, or diagonal. In short, the focus here is to place a number of queens on the board in such a manner that none of them are attacking each other. This can be accomplished by reducing the heuristic cost to zero and by calculating the cost of the board after each move.

FEATURES OF HILL CLIMBING ALGORITHM:

Often used in cases when a good heuristic function is available for evaluating different states, as well as when no other useful information or data is provided, the hill-climbing algorithm is further characterized by the following features:

It is a heuristic approach for optimizing problem

- It only looks at the current state and the immediate future state.
- A variant of the generate and test algorithm.
- Uses a greedy approach to keep generating possible solutions, until it finds the expected output.
- Finds a better solution by making an incremental change to the solution.

Now that we understand the concepts of Hill Climbing Algorithm, let us move on to the workings of the hill-climbing algorithm.

WORKINGS OF HILL CLIMBING ALGORITHM:

The workings of the hill algorithm can be grouped into the following steps:

1. INITIAL STATE EVALUATION:

The process is initiated by evaluating the initial state. If this state is the goal state, the process is ended here itself. However, if it isn't the goal state, the process moves on to the next stage.

2. SELECT NEW STATE FOR COMPARISON:

As the current state was not the goal state, the process is kept in the loop, until a solution is found or no further options are left for comparison. However, if this state is near the goal state it becomes the current state.

2. EVALUATE THE NEW STATE: THIS EVALUATION CAN RESULT IN THREE CONDITIONS:

- If this is the goal state, the process is finally ended.
- If it is better than the current state then it becomes the current state.
- If it isn't better than the current state, then the second stage is repeated.

In short, the focus of Hill Climbing is to reach the goal state, by directly evaluating the immediate/current state, until there are no more options left.

VARIANTS OF HILL CLIMBING ALGORITHM:

Here are the various **types of Hill Climbing in AI:**

1. SIMPLE HILL CLIMBING:

The first type of hill climbing algorithm used in AI, simple hill climbing examines the neighboring nodes one by one and chooses the first neighboring node which optimizes the current cost as the next node.

2. STEEPEST ASCENT HILL CLIMBING:

Unlike the former, the Steepest Ascent hill-climbing first examines all the neighboring nodes and then selects a neighbor or the node closest to the solution state as of the next node. In short, this type of hill-climbing algorithm compares all successors and selects the one closest to the solution.

3. STOCHASTIC HILL CLIMBING:

The third type of hill-climbing algorithm, stochastic hill-climbing randomly selects a neighboring node and based on the amount of the improvement decides whether or not to move to the next node.

4. RANDOM-RESTART HILL CLIMBING:

Considered to be a meta-algorithm which is built on **top of the hill** climbing algorithm, random-restart hill climbing or shotgun hill climbing performs the process iteratively with a random initial condition, in each phase.

ADVANTAGES OF HILL CLIMBING ALGORITHM:

From being memory efficient to helping agents get accurate results, the advantages offered by the hill climbing-algorithm are various, like:

- It can be used in conversions and discrete domains.
- It requires fewer conditions than other search techniques.

- Helpful in solving pure **optimization problems,** where the focus is on finding the best state.

PROBLEMS OF HILL CLIMBING ALGORITHM:

Though the benefits of the hill climbing algorithm are numerous, there are some issues or problems associated with it, which can impact the processing of the search. Therefore, here are the various problems of the **hill climbing technique,** which prevent the algorithm to reach the best state or the **global maximum:**

1. LOCAL MAXIMA:

Local maxima or **local maximum** is where all neighboring states have values worse than the current state. This leads to the termination of the process, even if a better solution is available. To reach the solution, backtracking is used.

2. RIDGES & ALLEYS:

In ridges and alleys, the process moves downwards in all possible directions, which makes it look like a peak and leads to the termination of the process. To avoid this, bidirectional search is used.

3. PLATEAU:

Here all neighbors share the same value, which makes it impossible to choose a direction as well as reach the goal state. To avoid this, a random state is selected which is far away from the current state.
Apart from these, other disadvantages offered by the hill-climbing algorithm are:

- It is not an efficient searching method.
- Not suitable for problems where the value of heuristic function drops suddenly.
- It is a local method, where the focus is on getting the immediate solution after considering the current state, rather than exploring all possible outcomes.

CONCLUSION:

From the above discussion, we can conclude the search algorithms like Alpha Beta Pruning are creating a revolution in the field of Artificial Intelligence and helping it pay a new and more efficient way of creating advanced approaches to solving such problems. Moreover, the alpha-beta pruning algorithm is a quality optimization over the min-max algorithm, and together with the minimax algorithm is becoming the foundation of the searching techniques.

BEST FIRST SEARCH::

Best First Search is another method to solve the problem by searching in an Intelligent Agent. In this search method, the agent simply chooses the best node based on heuristic value irrespective of where the node is. The heuristic value is attained by using the heuristic function which accepts the current state as an input and returns a heuristic value which is an integer value between -10 and +10. The state which has the maximum heuristic value is chosen to be the best state. The goal state always has a heuristic value of +10.

After getting the heuristic value of all the neighboring states, the agent gets the best value by simply sorting all the values in descending order and then simply choosing the first node in the order.

The **best first search** is efficient to use in many manners such as there is no need for the agent to go to each and every state and check for it to be the goal state. The agent can simply jump on to the best node which it gets from the sorted array.

The only drawback of this searching method is that if there are many numbers of neighboring nodes present, then finding all their heuristic values and sorting them and then finding the best node is a lengthy process.

Algorithm for Best First Search

Step 1: Evaluate the initial state. If it is the goal state, then quit and return the node.

Step 2: Evaluate the neighboring nodes.

Step 3: Loop until all the neighboring nodes or child nodes are explored and store them in an array.

Step 4: Sort the array in descending order according to their heuristic value.

Step 5: Choose the best node, which is the first node in the array.

Step 6: This is our goal state, so quit and return the node. **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:

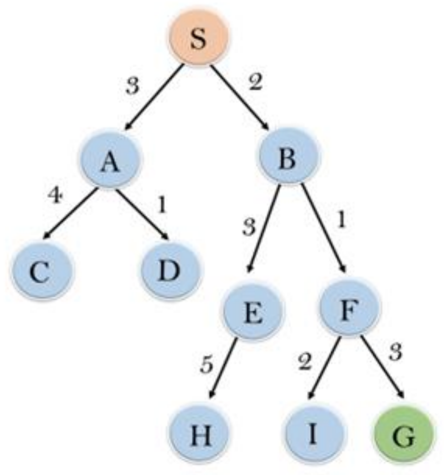
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

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.

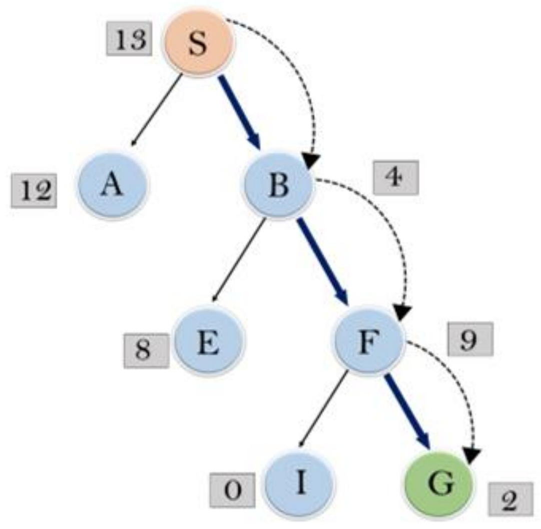
Example:

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..7.1.Problem Reduction

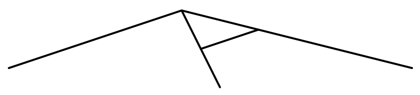
 So far search strategies discussed were for OR graphs.

 Here several arcs indicate a different ways of solving problem.

- Another kind of structure is AND-OR graph (tree).
- Useful for representing the solution of problem by decomposing it into smaller sub-problems.
- Each sub-problem is solved and final solution is obtained by combining solutions of each sub-problem.
- Decomposition generates arcs that we will call AND arc.
- One AND arc may point to any number of successors, all of which must be solved.
- Such structure is called AND–OR graph rather than simply AND graph.

Example of AND-OR Tree

Acquire TV



Steal TV Earn Money Buy TV

AND–OR Graph

- To find a solution in AND–OR graph, we need an algorithm similar to A*

     with the ability to handle AND arc appropriately.

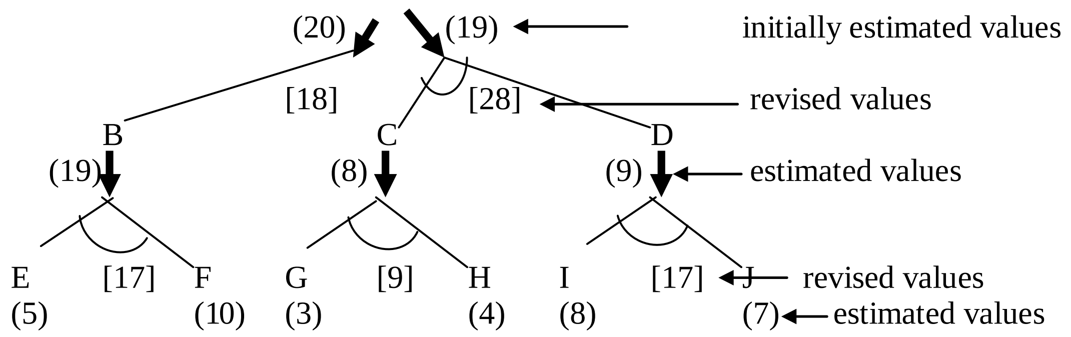
- In search for AND-OR graph, we will also use the value of heuristic function f for each node.

AND–OR Graph Search

- Traverse AND-OR graph, starting from the initial node and follow the current best path.

  Accumulate the set of nodes that are on the best path which have not yet been expanded.

- Pick up one of these unexpanded nodes and expand it.
- Add its successors to the graph and compute f (using only h) for each of them.
- Change the f estimate of newly expanded node to reflect the new information provided by its successors.
- Propagate this change backward through the graph to the start.
- Mark the best path which could be different from the current best path.
- Propagation of revised cost in AND-OR graph was not there in A*.
- Consider AND-OR graph given on next slide.
- Let us assume that each arc with single successor will have a cost of 1 and each AND arc with multiple successor will have a cost of 1 for each of its components for the sake of simplicity.
- Here the numbers listed in the circular brackets ( ) are estimated cost and the revised costs are enclosed in square brackets [ ].
- Thick lines indicate paths from a given node.



- Initially we start from start node A and compute heuristic values for each of its successors, say {B, (C and D)} as {19, (8, 9)}.
- The estimated cost of paths from A to B is 20 (19 + cost of one arc from A to B) and from A to (C and D) path is 19 (8+9 + cost of two arcs A to C and A to D).

  The path from A to (C and D) seems to be better. So expend this AND path by expending C to {(G and H)} and D to {(I and J)}.
- Now heuristic values of G, H, I and J are 3, 4, 8 and 7 respectively.
- This leads to revised cost of C and D as 9 and 17 respectively.
- These values are propagated up and the revised costs of path from A through (C and D) is calculated as 28 (9 + 17 + cost of arcs A to C and A to D).
- Now the revised cost of this path is 28 instead of earlier estimation of 19 and this path is no longer a best path.
- Then choose path from A to B for expansion.
- After expansion we see that heuristic value of node B is 17 thus making cost of path from A to B to be 18.
- This path is still best path so far, so further explore path from A to B.
- The process continues until either a solution is found or all paths have lead to dead ends, indicating that there is no solution.

start to current node) as it is not possible to compute a single such value since there may be many paths to the same state.

Introduction **OF GENETIC ::**

A genetic algorithm is used to solve complicated problems with a greater number of variables & possible outcomes/solutions. The combinations of different solutions are passed through the Darwinian based algorithm to find the best solutions. The poorer solutions are then replaced with the offspring of good solutions.

It all works on the Darwinian theory, where only the fittest individuals are chosen for reproduction. The various solutions are considered the elements of the population, and only the fittest solutions are allowed to reproduce (to create better solutions). Genetic algorithms help in optimizing the solutions to any particular problem.

The whole process of genetic algorithms is a computer program simulation in which the attributes of the problem & solution are treated as the attributes of the Darwinian theory. The basic processes which are involved in genetic algorithms are as follows:

- A population of solutions is built to any particular problem. The elements of the population compete with each other to find out the fittest one.
- The elements of the population that are fit are only allowed to create offspring (better solutions).
- The genes from the fittest parents (solutions) create a better offspring. Thus, future solutions will be better and sustainable.

Working of Genetic Algorithms in AI

The working of a **genetic algorithm in AI** is as follows:

- The components of the population, i.e., elements, are termed as genes in **genetic algorithms in AI**. These genes form an individual in the population (also termed as a chromosome).
- A search space is created in which all the individuals are accumulated. All the individuals are coded within a finite length in the search space.
- Each individual in the search space (population) is given a fitness score, which tells its ability to compete with other individuals.
- All the individuals with their respective fitness scores are sought & maintained by the genetic algorithm & the individuals with high fitness scores are given a chance to reproduce.
- The new offspring are having better 'partial solutions' as compared to their parents. Genetic algorithms also keep the space of the search space dynamic for accumulating the new solutions (offspring).
- This process is repeated until the offsprings do not have any new attributes/features than their parents (convergence). The population converges at the end, and only the fittest solutions remain along with their offspring (better solutions). The fitness score of new individuals in the population (offspring) are also calculated.

Key Terminologies in Genetic Algorithms

- **Selection Operator** – This operator in **genetic algorithms in AI** is responsible for selecting the individuals with better fitness scores for reproduction.
- **Crossover Operator** – The crossover operator chooses a crossover site from where the merge will happen. The crossover sites in both the individuals available for mating are chosen randomly and form new individuals.
- **Mutation Operator** – This operator in the genetic algorithm is responsible for embedding random genes in the offspring to maintain diversity and avoid premature convergence.
- **Premature Convergence** – If a problem is optimized quickly, it means that the offspring were not produced at many levels. The solutions will also not be of optimal quality. To avoid premature convergence, new genes are added by the mutation operator.
- **Allele** – The value of a particular gene in a chromosome is termed as an allele. The specified set of alleles for each gene defines the possible chromosomes of that particular gene.

Benefits and Uses of Genetic Algorithms

- The solutions created through genetic algorithms are strong & reliable as compared to other solutions.
- They increase the size of solutions as solutions can be optimized over a large search scale. This algorithm also can manage a large population.

- The solutions produced by genetic algorithms do not deviate much on slightly changing the input. They can handle a little bit of noise.
- Genetic algorithms have a stochastic distribution that follows probabilistic transition rules, making them hard to predict but easy to analyze.
- Genetic algorithms can also perform in noisy environments. It can also work in case of complex & discrete problems.
- Due to their effectiveness, genetic algorithms have many applications like neural networks, fuzzy logic, code-breaking, filtering & signal processing. You can learn more about the **genetic algorithms in AI** via the top courses offered by upGrad.
- Conclusion
  Genetic algorithms are an important concept in AI and are one of the best performing AI algorithms when it comes to finding optimal solutions.

Means-Ends Analysis in Artificial Intelligence

- We have studied the strategies which can reason either in forward or backward, but a mixture of the two directions is appropriate for solving a complex and large problem. Such a mixed strategy, make it possible that first to solve the major part of a problem and then go back and solve the small problems arise during combining the big parts of the problem. Such a technique is called **Means-Ends Analysis**.
- Means-Ends Analysis is problem-solving techniques used in Artificial intelligence for limiting search in AI programs.
- It is a mixture of Backward and forward search technique.
- The MEA technique was first introduced in 1961 by Allen Newell, and Herbert A. Simon in their problem-solving computer program, which was named as General Problem Solver (GPS).
- The MEA analysis process centered on the evaluation of the difference between the current state and goal state.

How means-ends analysis Works:

The means-ends analysis process can be applied recursively for a problem. It is a strategy to control search in problem-solving. Following are the main Steps which describes the working of MEA technique for solving a problem.

a. First, evaluate the difference between Initial State and final State.
b. Select the various operators which can be applied for each difference.
c. Apply the operator at each difference, which reduces the difference between the current state and goal state.

Operator Subgoaling

In the MEA process, we detect the differences between the current state and goal state. Once these differences occur, then we can apply an operator to reduce the differences. But sometimes it is possible that an operator cannot be applied to the current state. So we create the subproblem of the current state, in which operator can be applied, such type of backward chaining in which operators are selected, and then sub goals are set up to establish the preconditions of the operator is called **Operator Subgoaling**.

Algorithm for Means-Ends Analysis:

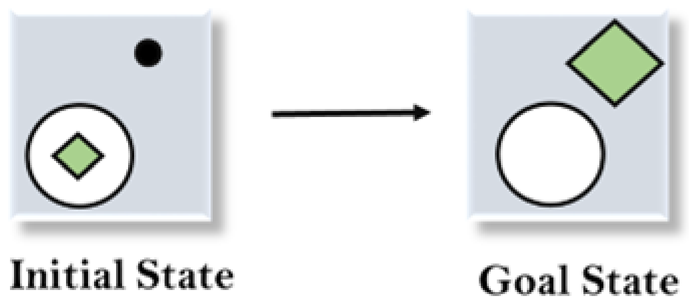
Let's we take Current state as CURRENT and Goal State as GOAL, then following are the steps for the MEA algorithm.

- **Step 1:** Compare CURRENT to GOAL, if there are no differences between both then return Success and Exit.
- **Step 2:** Else, select the most significant difference and reduce it by doing the following steps until the success or failure occurs.
    a. Select a new operator O which is applicable for the current difference, and if there is no such operator, then signal failure.
    b. Attempt to apply operator O to CURRENT. Make a description of two states.
    i) O-Start, a state in which O?s preconditions are satisfied.
    ii) O-Result, the state that would result if O were applied In O-start.
    c. If
    (First-Part <------ MEA (CURRENT, O-START)
    And
    (LAST-Part <----- MEA (O-Result, GOAL), are successful, then signal Success and return the result of combining FIRST-PART, O, and LAST-PART.

The above-discussed algorithm is more suitable for a simple problem and not adequate for solving complex problems.

Example of Mean-Ends Analysis:

Let's take an example where we know the initial state and goal state as given below. In this problem, we need to get the goal state by finding differences between the initial state and goal state and applying operators.



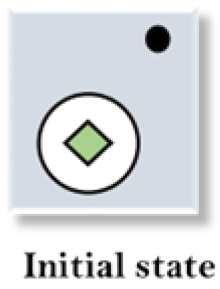
-

Solution:

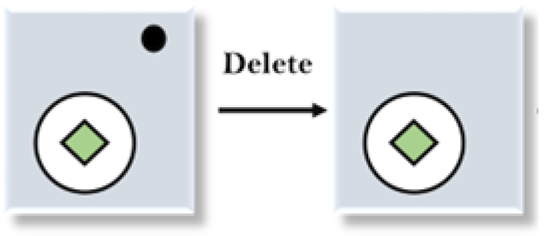
To solve the above problem, we will first find the differences between initial states and goal states, and for each difference, we will generate a new state and will apply the operators. The operators we have for this problem are:

- **Move**
- **Delete**
- **Expand**

1. Evaluating the initial state: In the first step, we will evaluate the initial state and will compare the initial and Goal state to find the differences between both states.

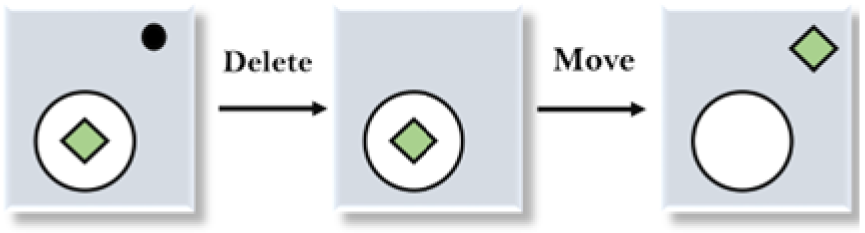


2. Applying Delete operator: As we can check the first difference is that in goal state there is no dot symbol which is present in the initial state, so, first we will apply the **Delete operator** to remove this dot.



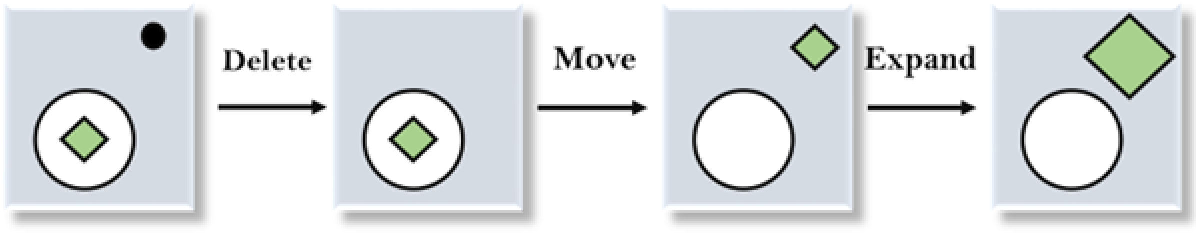
Initial state

3. Applying Move Operator: After applying the Delete operator, the new state occurs which we will again compare with goal state. After comparing these states, there is another difference that is the square is outside the circle, so, we will apply the **Move Operator**.



Initial state

4. Applying Expand Operator: Now a new state is generated in the third step, and we will compare this state with the goal state. After comparing the states there is still one difference which is the size of the square, so, we will apply **Expand operator**, and finally, it will generate the goal state.



Initial state Goal state

Introduction of constraint satisfaction problems::

Just like AI Planning as Satisfiability, we can use an existing technique — Constraint Satisfaction Problems to help us solve AI Planning Problems. This way we can use the existing well-developed algorithms for solving CSPs to solve our AI Planning Problems.

We will first go through the general introduction of CSPs. We then continue to see how to encode AI Planning Problems into CSPs. Finally, we use CSP Backtracking Algorithm to solve our problems. We will prove all these theories by implementing them in Python.

Constraint Satisfaction Problems

From Wikipedia,

CSPs are mathematical questions defined as a set of objects whose state must satisfy a number of constraints.

$$P = \{X, D, C\}$$
$$X = \{x_1, ..., x_n\}$$
$$D = \{D_1, ..., D_n\}$$
$$C = \{c_1, ..., c_m\}$$

$$P = \{X, D, C\}$$
$$X = \{x_1, ..., x_n\}$$
$$D = \{D_1, ..., D_n\}$$
$$C = \{c_1, ..., c_m\}$$

Constraint Satisfaction Problems (Image by Author)

The picture depicts A CSP over finite domains. We have a set of variables — X, a set of list of domains — D, and a set of constraints — C.

Values assigned to variables in X must be one of their domain in D.

$$x_i \in D_i$$

$$x_i \in D_i$$

Variable and Domain (Image by Author)

A constraint restricts the possible values of a subset of variables from X.

Example

A good example is Map Coloring Problem. You can click on the link to see the map. In that problem we have the **variables** which are the regions, the **domains** are the colors that we can assign to the variables (red, green, yellow, blue); in this example all variables have the same domains.

[illegible]

$$X = \{AB, MB, NL, NT, ON, QC, YT\}$$
$$D = \{\{Red, Green, Yellow, Blue\}, ...\}$$

Map Coloring Variables and Domains (Image by Author)

The constraint is that a color that is assigned to a region cannot be assigned to the adjacent regions.

CSPs on finite domains are typically resolved using some kind of search algorithms. Now that we understand what CSP on finite domains is, we can start looking into how we can encode Planning Problems into CSP.

Encoding Planning Problems into CSPs

Just like with the “Planning as Satisfiability” in which we encode the Planning Problem into the Propositional Satisfiability Problem, we can do the same with this approach.

We encode the Bounded Planning Problem — restricted Planning Problem with a fixed length of plan into a Constraint Satisfaction Problem. There are four steps to fully encode it. We will see them one-by-one below.

CSP Variables and Domains

In this first step, we want to create CSP Variables and their corresponding CSP Domains. There are two components that we need to convert into CSP Variables, they are:

- Predicates
- Actions

For predicates, we instantiate all predicates for each step. Remember that we have a fixed length of plan (*k*).

For example, let's assume that we have a simple planning domain in which we only have one robot and two locations, represented in the following pddl file. And, our Bounded Planning Problem has a length of 1.
Simple Domain (Code by Author)

We can encode the planning domain into the following CSP Variables and Domains:

[illegible]
[illegible]

$$atl(0, rob) = \{loc1, loc2\}$$
$$atl(1, rob) = \{loc1, loc2\}$$

CSP Variables and Domains — Predicates (Image by Author)

The number denotes the step, so for every step from 0 to the length of our Bounded Planning Problem, we will have all possible predicates enumerated. In this example, we have two variables in which each of them has the same Domain *{loc1, loc2}*.

Similarly for actions, we instantiate from step 0 to the step length-1.

$$act(0) = \{move(rob, loc1, loc2), move(rob, loc2, loc1), no - op\}$$

CSP Variables and Domain — Actions (Image by Author)

For our simple domain, we only have one variable for action, which is the *act(0)* with two possible values in its domain. But additionally, we will need to add a no-op action (No Operation Action) to the domain. It is an action that has no preconditions nor effects.

Now, we have our CSP variables and the corresponding domains, in the next steps, we'll have our CSP constraints.

CSP Constraints on Initial State and Goal State

In this step, we encode our initial state and goal state into CSP constraints. For our example, we'll use a Planning Problem of the simple Planning Domain represented in the pddl file below.
Simple Planning Problem (Code by Author)

In the initial state, our robot is at *loc1* and in the goal state, it is at *loc2*. It is really simple.

Now, this step is very straightforward. We just need to convert the initial state and the goal state into unary constraints, with each affected variable assigned to a value from its domain.

$atl(0, rob) = loc1$

$atl(1, rob) = loc2$

$$atl(0, rob) = loc1$$
$$atl(1, rob) = loc2$$

CSP Constraints on Initial and Goal States (Image by Author)

This means that at step 0 the location of the robot is restricted to *loc1* and at step 1 it is restricted at *loc2*.

CSP Constraints on Actions

In this step, we convert the actions into binary constraints. We write sets of constraints as follows.

Preconditions:

$$\{(act(j) = move(r, l, m), atl(j, r, l)) \mid adjacent(l, m) \wedge j = 0\}$$

Preconditions Constraints Set Builder (Image by Author)

$$\{(act(0) = move(rob, loc1, loc2), atl(0, rob) = loc1),\\ (act(0) = move(rob, loc2, loc1), atl(0, rob) = loc2)\}$$

Preconditions Constraints Example (Image by Author)

Effects:

$$\{(act(j) = move(r, l, m), atl(j + 1, r, m)) \mid adjacent(l, m) \wedge j = 0\}$$

Effects Constraints Set Builder (Image by Author)

$$\{(act(0) = move(rob, loc1, loc2), atl(1, rob) = loc2),\\ (act(0) = move(rob, loc2, loc1), atl(1, rob) = loc1)\}$$

Effects Constraints Example (Image by Author)

These constraints are to make sure that the variables are assigned correctly. If *act(0)* has been assigned to *move(rob, loc1, loc2), atl(0, rob)* can only be *loc1*.

CSP Constraints of Frame Axioms

A Frame Axiom Constraint is to say that variables that are not explicitly modified (or also called invariants) in effects of action remain unchanged. To give an example we need to add another predicate to our

Planning Domain. Say we have a new predicate *loaded(r)={cont, nil}* that represents whether the robot is loaded with a container or not. This predicate isn't affected by *move(r, l, m)* action and so it will remain the same before and after *move(r, l, m)* action is executed. We represent this fact as follows.

[illegible]

$$\{(act(0) = move(rob, loc1, loc2), loaded(0, rob) = cont, loaded(1, rob) = cont),$$
$$(act(0) = move(rob, loc1, loc2), loaded(0, rob) = nil, loaded(1, rob) = nil),$$
$$(act(0) = move(rob, loc2, loc1), loaded(0, rob) = cont, loaded(1, rob) = cont),$$
$$(act(0) = move(rob, loc2, loc1), loaded(0, rob) = nil, loaded(1, rob) = nil)\}$$

Frame Axiom Constraints Example (Image by Author)

From the example above, we state that after executing the move action the robot remains unloaded/loaded depending on its initial state.

Algorithms to solve CSPs

We now have our Bounded Planning Problem — AI Planning Problem with a fixed length *k*, encoded into a Constraint Satisfaction Problem. We can use the existing resolvers to solve the CSP.

If a solution exists, the solvers return assigned variables (see our CSP variables in the previous section). We are particularly interested in the *Action CSP Variables* for us to extract the plan.

We can then order the extracted *Action CSP Variables* based on the step and have our plan.

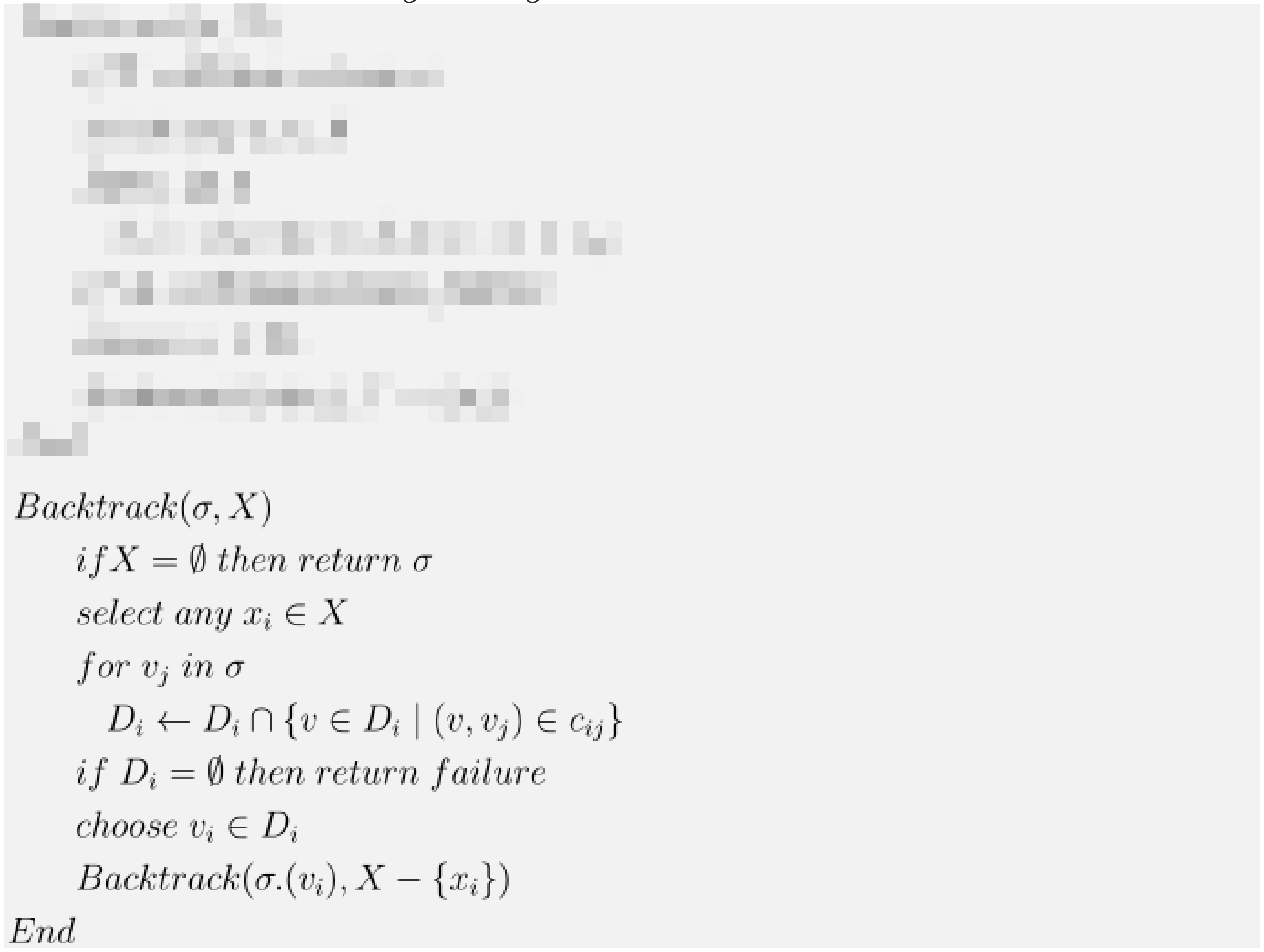
[illegible]

$$\pi = \langle act(0), \ldots, act(k-1) \rangle$$

Ordered Plan (Image by Author)

Search Algorithms to solve CSPs

A common solver is a backtracking search algorithm such as the one below.



[illegible]

```
$Backtrack(\sigma, X)$
    $if X = \emptyset\ then\ return\ \sigma$
    $select\ any\ x_i \in X$
    $for\ v_j\ in\ \sigma$
      $D_i \leftarrow D_i \cap \{v \in D_i \mid (v, v_j) \in c_{ij}\}$
    $if\ D_i = \emptyset\ then\ return\ failure$
    $choose\ v_i \in D_i$
    $Backtrack(\sigma.(v_i), X - \{x_i\})$
$End$
```

Backtrack Search Algorithm (Image by Author)

This algorithm is quite simple. It receives the partial solution in σ and the unassigned variables in X.

If all variables have been assigned, that means we have our solution and the algorithm returns the σ.

Otherwise, it will select one variable to assign a value to it. Then it proceeds to remove values that are not consistent from the Domain, D.

It then assigns a value to it, it then calls itself recursively. If it fails at some point, it will backtrack and choose another value at “choose vi ∈ Di”.

It will be easier to read when we implement this algorithm in Python.

Conclusion and Code in Python

In this post, we learned how we can use the existing technique — CSPs, to find a solution plan to a Planning Problem by encoding our Classical Planning Problem representation into a Bounded Planning Problem and into a Constraint Satisfaction Problem.

We then use the classic CSP Backtracking Algorithm to solve our CSP.

One limitation of this approach is that we need to specify the length of the plan, which means we may need to try several times before knowing whether or not a solution plan to our problem exists.

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:

	Deterministic	Chance Moves
Perfect information	Chess, Checkers, go, Othello	Backgammon, monopoly
Imperfect information	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble, nuclear war

- **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.

Note: In this topic, we will discuss deterministic games, fully observable environment, zero-sum, and where each agent acts alternatively.

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:

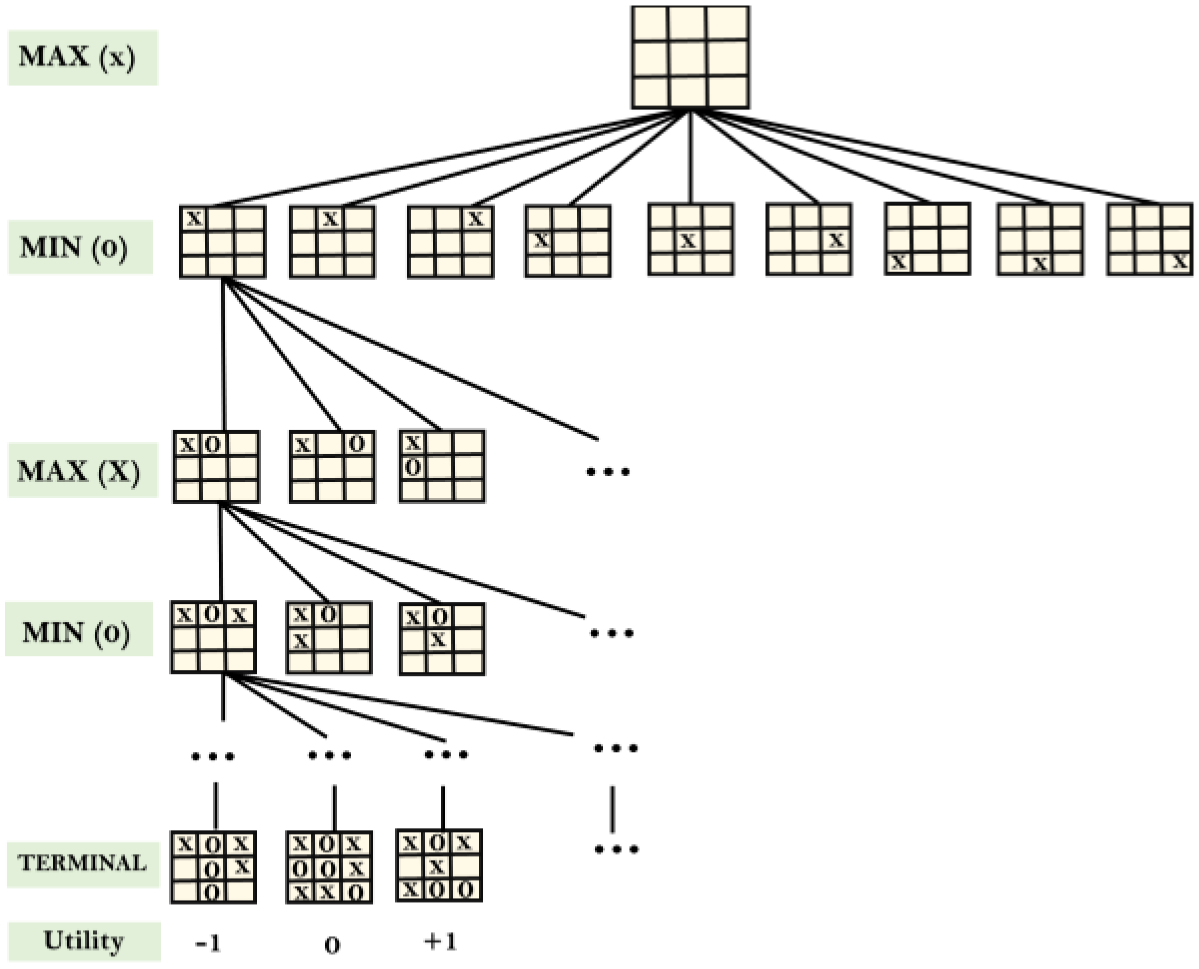
A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

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) =

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

Optimal Strategy for a Game | DP-31
Consider a row of n coins of values v1 . . . vn, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.
Note: The opponent is as clever as the user.

Let us understand the problem with few examples:

1. 5, 3, 7, 10 : The user collects maximum value as 15(10 + 5)
2. 8, 15, 3, 7 : The user collects maximum value as 22(7 + 15)

Does choosing the best at each move gives an optimal solution? No.
In the second example, this is how the game can be finished:

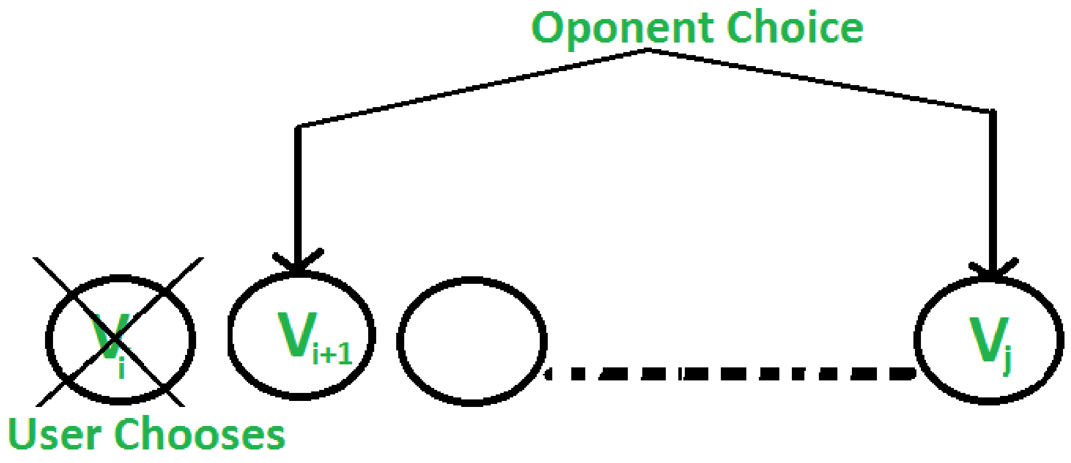
1. …….User chooses 8.
   …….Opponent chooses 15.
   …….User chooses 7.
   …….Opponent chooses 3.
   Total value collected by user is 15(8 + 7)
2. …….User chooses 7.
   …….Opponent chooses 8.
   …….User chooses 15.

…….Opponent chooses 3.
Total value collected by user is 22(7 + 15)

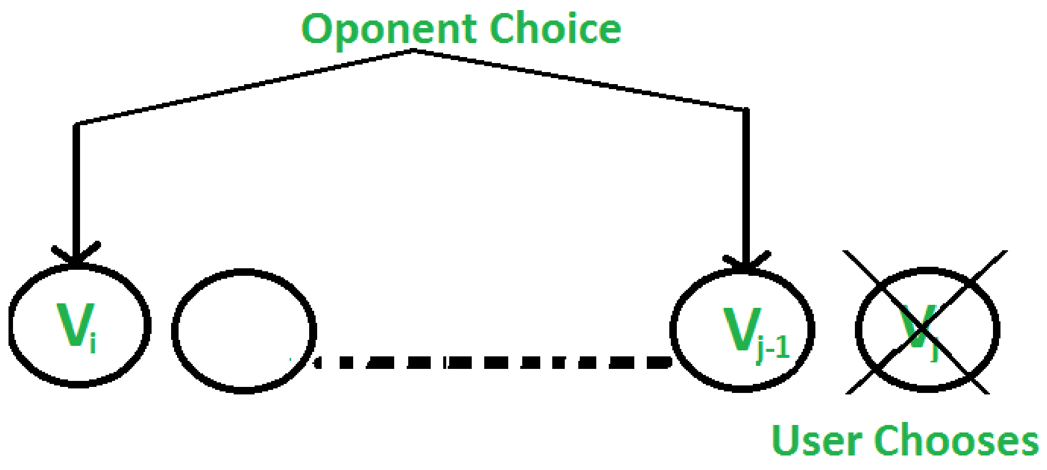
So if the user follows the second game state, the maximum value can be collected although the first move is not the best.

There are two choices:

- The user chooses the 'ith' coin with value 'Vi': The opponent either chooses (i+1)th coin or jth coin. The opponent intends to choose the coin which leaves the user with **minimum value**.
  i.e. The user can collect the value **Vi + min(F(i+2, j), F(i+1, j-1))**.



- The user chooses the 'jth' coin with value 'Vj': The opponent either chooses 'ith' coin or '(j-1)th' coin. The opponent intends to choose the coin which leaves the user with minimum value, i.e. the user can collect the value **Vj + min(F(i+1, j-1), F(i, j-2))**.



Following is the recursive solution that is based on the above two choices. We take a maximum of two choices.

F(i, j) represents the maximum value the user
can collect from i'th coin to j'th coin.

F(i, j) = Max(Vi + min(F(i+2, j), F(i+1, j-1)),
              Vj + min(F(i+1, j-1), F(i, j-2)))

As user wants to maximise the number of coins.

Base Cases

F(i, j) = Vi   If j == i

F(i, j) = max(Vi, Vj) If j == i + 1

- C++
- Java
- Python3
- C#
- PHP
- Javascript

```
// C++ program to find out

// maximum value from a given

// sequence of coins

#include <bits/stdc++.h>

using namespace std;

 // Returns optimal value possible

// that a player can collect from

// an array of coins of size n.

// Note than n must be even
```

```
int optimalStrategyOfGame(
    int* arr, int n)
{
    // Create a table to store
    // solutions of subproblems
    int table[n][n];
     // Fill table using above
    // recursive formula. Note
    // that the table is filled
    // in diagonal fashion (similar
    // to http:// goo.gl/PQqoS),
    // from diagonal elements to
    // table[0][n-1] which is the result.
    for (int gap = 0; gap < n; ++gap) {
        for (int i = 0, j = gap; j < n; ++i, ++j) {
            // Here x is value of F(i+2, j),
            // y is F(i+1, j-1) and
            // z is F(i, j-2) in above recursive
            // formula
            int x = ((i + 2) <= j)
                    ? table[i + 2][j]
                    : 0;
            int y = ((i + 1) <= (j - 1))
                    ? table[i + 1][j - 1]
                    : 0;
            int z = (i <= (j - 2))
                    ? table[i][j - 2]
                    : 0;
             table[i][j] = max(
               arr[i] + min(x, y),
               arr[j] + min(y, z));
          }
       }


    return table[0][n - 1];
}
```

```
// Driver program to test above function

int main()

{

    int arr1[] = { 8, 15, 3, 7 };

    int n = sizeof(arr1) / sizeof(arr1[0]);

    printf("%d\n",

           optimalStrategyOfGame(arr1, n));

     int arr2[] = { 2, 2, 2, 2 };

    n = sizeof(arr2) / sizeof(arr2[0]);

    printf("%d\n",

           optimalStrategyOfGame(arr2, n));

     int arr3[] = { 20, 30, 2, 2, 2, 10 };

    n = sizeof(arr3) / sizeof(arr3[0]);

    printf("%d\n",

           optimalStrategyOfGame(arr3, n));

     return 0;

}
```

Output:

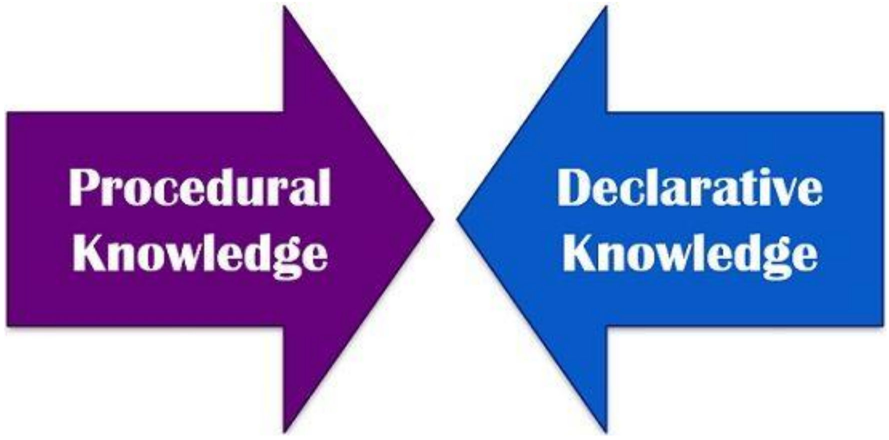
```
22

4

42
```

Complexity Analysis:

- **Time Complexity:** O(n^2).
  Use of a nested for loop brings the time complexity to n^2.
- **Auxiliary Space:** O(n^2).
  As a 2-D table is used for storing states.

UNIT 3

Difference Between Procedural and Declarative Knowledge



The **knowledge** can be expressed in the various forms to the inference engine in the computers to solve the problems in context of the rule. So, in this article, we are going to discuss the two representation of procedural knowledge and declarative knowledge. The prior difference between them is that the **declarative** representation just specifies the knowledge but not the mechanism to implement the knowledge. On the contrary, the **procedural** representation provides the control information along with the knowledge.

Computers are not able to obtain and represent knowledge by their own like humans, as it is hard for the devices to interpret and represent the knowledge in the form of natural language. So, in devices like a computer, the knowledge is managed by involving given steps such as knowledge acquisition, storing, retrieval and reasoning. The steps are described below:

- **Knowledge acquisition**: It gathers, structures and organizes the knowledge.
- **Knowledge storing**: In this step, the knowledge is reposited over the computer.
- **Knowledge retrieva**l: This step helps in fetching the knowledge when required.
- **Reasoning**: It provides a conclusion, inferences or explanation.

Content: Procedural Knowledge and Declarative Knowledge

Comparison Chart

BASIS FOR COMPARISON	PROCEDURAL KNOWLEDGE	DECLARATIVE KNOWLEDGE
Basic	Includes the knowledge of how a particular thing can be accomplished.	Includes the basic knowledge about something.
Alternate name	Interpretive knowledge	Descriptive knowledge
Stated by	Direct application to the task and difficult to articulate formally.	Declarative sentences and easily articulated.
Popularity	Less common	Generally used
Ease of sharing the knowledge	Hard to communicate	Can be easily shared, copied, processed and stored.
Taken from	Experience, action, and subjective insight.	Artifact of some type as a principle, procedure, process and concepts.
Nature	Process oriented	Data-oriented

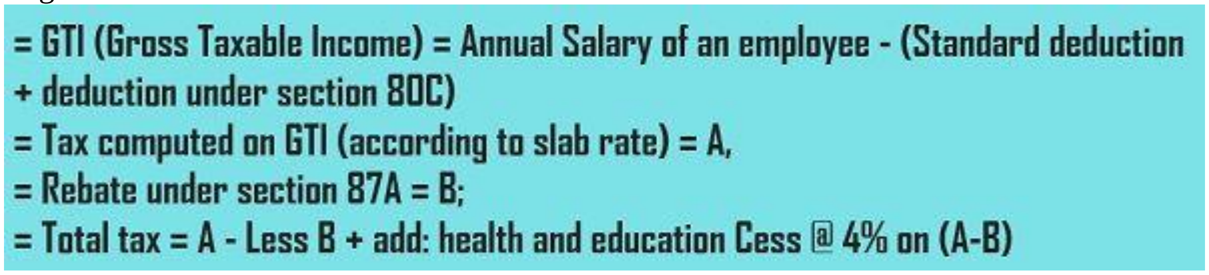
BASIS FOR COMPARISON	PROCEDURAL KNOWLEDGE	DECLARATIVE KNOWLEDGE
Represented by	Set of rules	Production systems
Feature	Debugging is difficult	Validation is quite simple

Definition of Procedural Knowledge

The **Procedural knowledge** is a type of knowledge where the essential control information that is required to use the information is integrated in the knowledge itself. It also used with an interpreter to employ the knowledge which follows the instructions given in the knowledge.

Example

Let's understand this by an example, it can include a group of logical assertions merged with a resolution theorem prover to provide an absolute program for solving problems. Here, the implied income tax of an employee salary can be thought of as a procedural knowledge as it would require a process to calculate it as given below.



= GTI (Gross Taxable Income) = Annual Salary of an employee - (Standard deduction + deduction under section 80C)
= Tax computed on GTI (according to slab rate) = A,
= Rebate under section 87A = B;
= Total tax = A - Less B + add: health and education Cess @ 4% on (A-B)

So, this is how the tax of an employee is calculated by following a lengthy process instead of just collecting facts.

Definition of Declarative Knowledge

A **Declarative knowledge** is where only knowledge is described but not the use to which the knowledge is employed is not provided. So, in order to use this declarative knowledge, we need to add it with a program that indicates what is to be done to the knowledge and how it is to be done.

Example

Let us understand this by the example of an employee whose ID, name, address, salary have to stored in a database where this information is fact-based does not require much effort to acquire it.

Key Differences Between Procedural and Declarative Knowledge

1. When the conscious perception and conscious planning is involved in the knowledge it is known as procedural knowledge. On the contrary, in declarative knowledge is not conscious.
2. Declarative knowledge is verbalized, shared, copied, processed and stored in an easy way while procedural knowledge is hard to express.
3. Among procedural and declarative knowledge the declarative knowledge is more commonly used.
4. Procedural knowledge is obtained from experience, action and subjective insight. As against, declarative knowledge is obtained from artifact, procedure, process and concepts.
5. Procedural knowledge is process-oriented in nature whereas declarative knowledge is data-oriented.

Conclusion

The procedural and declarative knowledge can easily be distinguished by identifying where control information is residing and whether it is used consciously and unconsciously.

Difference between Procedural and Declarative Knowledge

Procedural Knowledge:

Procedural Knowledge also known as Interpretive knowledge, is the type of knowledge in which it clarifies how a particular thing can be accomplished. It is not so popular because it is generally not used. It emphasize **how to do** something to solve a given problem.

Let's see it with an example:

```
var a=[1, 2, 3, 4, 5];

var b=[];

for(var i=0;i<a.length;i++)

{

  b.push(a[i]);

}

console.log(b);
```

Output is:

```
[1, 2, 3, 4, 5]
```

Declarative Knowledge:

Declarative Knowledge also known as Descriptive knowledge, is the type of knowledge which tells the basic knowledge about something and it is more popular than Procedural Knowledge.

It emphasize **what to do** something to solve a given problem.
Let's see it with an example:

```
var a=[1, 2, 3, 4, 5];

var b=a.map(function(number)

{

 return number*1});

console.log(b);
```

Output is:

```
[1, 2, 3, 4, 5]
```

In both example we can see that the output of a given problem is same because the only difference in that two methods to achieve the output or solution of problem.

Difference the Procedural and Declarative Knowledge:

S.NO	Procedural Knowledge	Declarative Knowledge
1.	It is also known as Interpretive knowledge.	It is also known as Descriptive knowledge.
2.	Procedural Knowledge means how a particular thing can be accomplished.	While Declarative Knowledge means basic knowledge about something.
3.	Procedural Knowledge is generally not used means it is not more popular.	Declarative Knowledge is more popular.

4.	Procedural Knowledge can't be easily communicate.	Declarative Knowledge can be easily communicated.
5.	Procedural Knowledge is generally process oriented in nature.	Declarative Knowledge is data oriented in nature.
6.	In Procedural Knowledge debugging and validation is not easy.	In Declarative Knowledge debugging and validation is easy.
7.	Procedural Knowledge is less effective in competitive programming.	Declarative Knowledge is more effective in competitive programming.

Difference Between Forward and Backward Reasoning in AI

In Artificial intelligence, the purpose of the search is to find the path through a problem space. There are two ways to pursue such a search that are forward and backward reasoning. The significant difference between both of them is that forward reasoning starts with the initial data towards the goal. Conversely, backward reasoning works in opposite fashion where the purpose is to determine the initial facts and information with the help of the given results.

Content: Forward Vs Backward Reasoning

Comparison Chart

BASIS FOR COMPARISON	FORWARD REASONING	BACKWARD REASONING
Basic	Data-driven	Goal driven
Begins with	New Data	Uncertain conclusion
Objective is to find the	Conclusion that must follow	Facts to support the conclusions
Type of approach	Opportunistic	Conservative
Flow	Incipient to consequence	Consequence to incipient

Definition of Forward Reasoning

The solution of a problem generally includes the initial data and facts in order to arrive at the solution. These unknown facts and information is used to deduce the result. For example, while diagnosing a patient the doctor first check the symptoms and medical condition of the body such as temperature, blood pressure, pulse, eye colour, blood, etcetera. After that, the patient symptoms are analysed and compared against the predetermined symptoms. Then the doctor is able to provide the medicines according to the symptoms of the patient. So, when a solution employs this manner of reasoning, it is known as **forward reasoning**.

Steps that are followed in the forward reasoning

The inference engine explores the knowledge base with the provided information for constraints whose precedence matches the given current state.

- In the first step, the system is given one or more than one constraints.
- Then the rules are searched in the knowledge base for each constraint. The rules that fulfil the condition are selected(i.e., IF part).
- Now each rule is able to produce new conditions from the conclusion of the invoked one. As a result, THEN part is again included in the existing one.
- The added conditions are processed again by repeating step 2. The process will end if there is no new conditions exist.

Definition of Backward Reasoning

The **backward reasoning** is inverse of forward reasoning in which goal is analysed in order to deduce the rules, initial facts and data. We can understand the concept by the similar example given in the above definition, where the doctor is trying to diagnose the patient with the help of the inceptive data such as symptoms. However, in this case, the patient is experiencing a problem in his body, on the basis of which the doctor is going to prove the symptoms. This kind of reasoning comes under backward reasoning.

Steps that are followed in the backward reasoning

In this type of reasoning, the system chooses a goal state and reasons in the backward direction. Now, let's understand how does it happens and what steps are followed.

- Firstly, the goal state and the rules are selected where the goal state reside in the THEN part as the conclusion.
- From the IF part of the selected rule the subgoals are made to be satisfied for the goal state to be true.
- Set initial conditions important to satisfy all the subgoals.
- Verify whether the provided initial state matches with the established states. If it fulfils the condition then the goal is the solution otherwise other goal state is selected.

Key Differences Between Forward and Backward Reasoning in AI

1. The forward reasoning is data-driven approach while backward reasoning is a goal driven.
2. The process starts with new data and facts in the forward reasoning. Conversely, backward reasoning begins with the results.

3. Forward reasoning aims to determine the result followed by some sequences. On the other hand, backward reasoning emphasis on the acts that support the conclusion.
4. The forward reasoning is an opportunistic approach because it could produce different results. As against, in backward reasoning, a specific goal can only have certain predetermined initial data which makes it restricted.
5. The flow of the forward reasoning is from the antecedent to consequent while backward reasoning works in reverse order in which it starts from conclusion to incipient.

Conclusion

The production system structure of the search process facilitates in the interpretation of the forward and backward reasoning. The forward and backward reasoning are differentiated on the basis of their purpose and process, in which forward reasoning is directed by the initial data and intended to find the goal while the backward reasoning is governed by goal instead of the data and aims to discover the basic data and facts.

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:

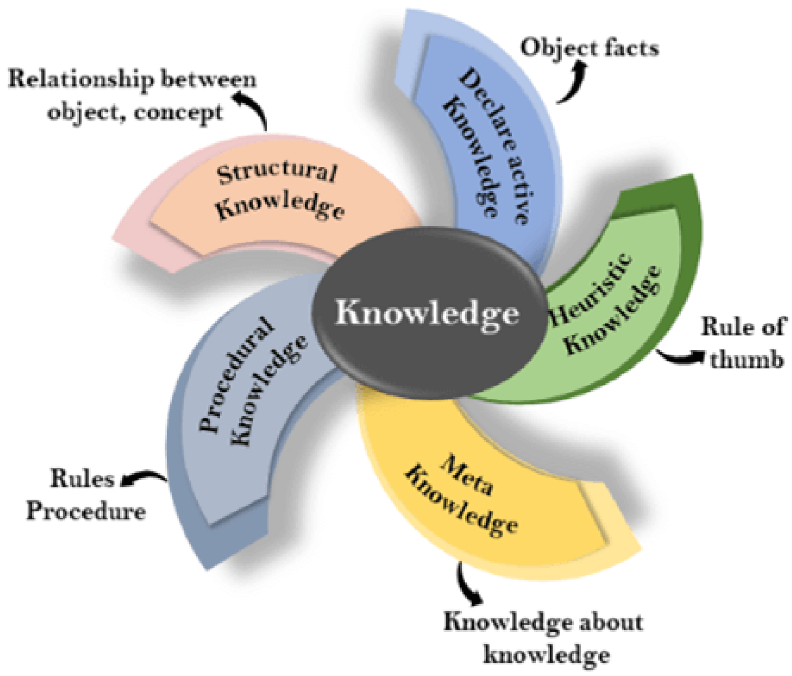
- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.

- **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

Following are the various 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 declarative sentences.
- It is simpler than procedural language.

2. 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.

3. Meta-knowledge:

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

4. 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.

5. Structural knowledge:

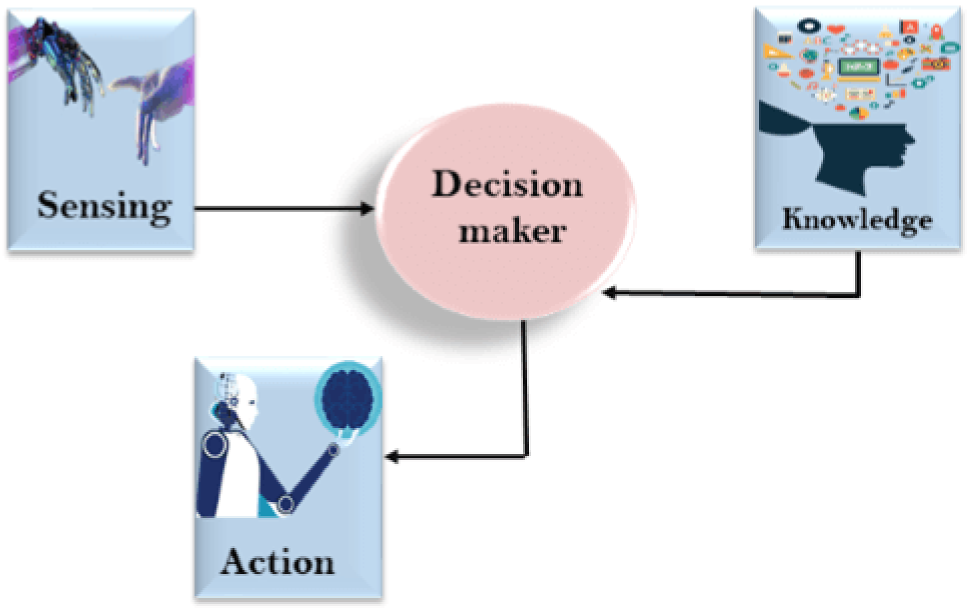
- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

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.

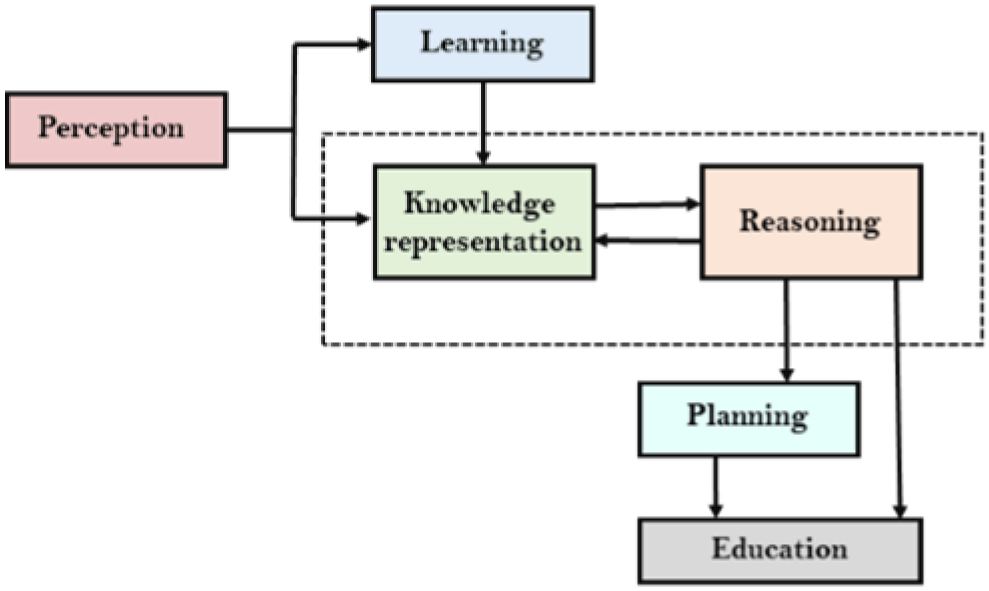
As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.



AI knowledge cycle:

An Artificial intelligence system has the following components for displaying intelligent behavior:

- Perception
- Learning
- Knowledge Representation and Reasoning
- Planning
- Execution



The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

Approaches to knowledge representation:

There are mainly four approaches to knowledge representation, which are givenbelow:

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.

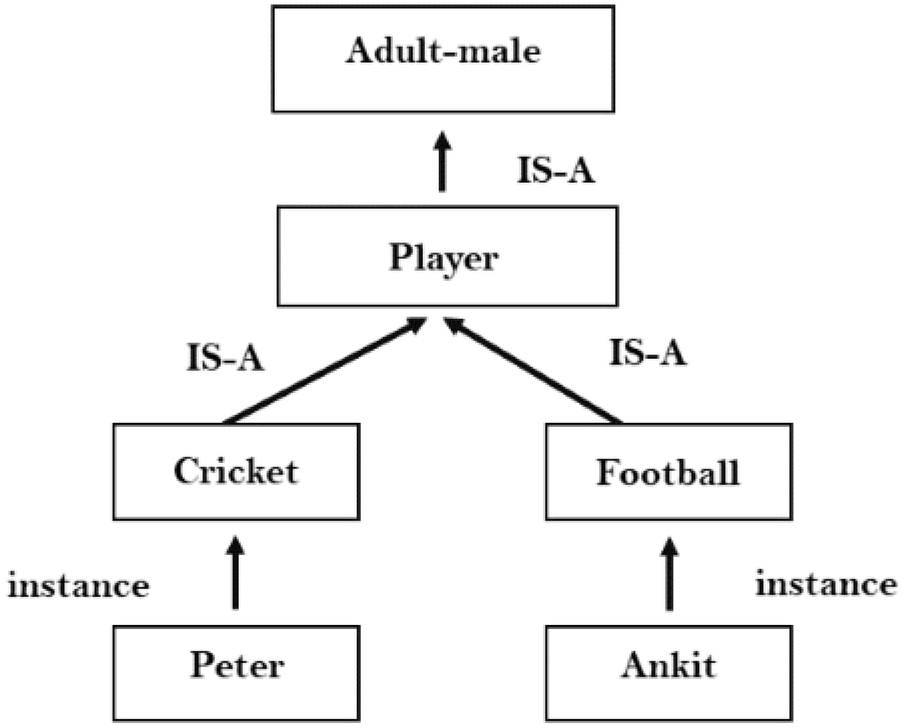
Example: The following is the simple relational knowledge representation.

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

     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.

Requirements for knowledge Representation system:

A good knowledge representation system must possess the following properties.

1. **1.RepresentationalAccuracy:**
   KR system should have the ability to represent all kind of required knowledge.
2. **2.InferentialAdequacy:**
   KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.
3. **3.InferentialEfficiency:**
   The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
4. **4. Acquisitional efficiency-** The ability to acquire the new knowledge easily using automatic methods.

The Most Common Problems of Artificial Intelligence Implementation

AI Algorithm Bias

Bias is one of the biggest challenges AI has to overcome in the future. Bias can find a way to lurk into algorithms in some ways. AI technology utilizes training datasets to make predictions. And these datasets happen to include human-based decisions laced with gender, sex, race or any other personal information. Back in October 2018, Amazon uncovered that their AI recruiting engine was biased against women. Later on, it was revealed that Amazon's training models were trained to range candidates by analyzing CVs for a certain period of time. As it turns out those came from men, as they used to dominate the industry. The AI tool devised the CVs containing the words 'woman' and 'women's'. And there are plenty of stories to back up the theory that for now, AI technology is biased.

Not Everyone Understands What AI Is

To implement AI into business, one should be well aware of its possibilities and limitations, pros and cons. Truth to be told, people barely know what the technology is and how it can address various business challenges. The most popular thing that comes to mind, when one hears the word 'artificial intelligence' is robots taking over humankind. The thing is that the lack of understanding of AI technology slows down the adoption of it in many industries. To solve this problem, people should educate themselves on the problem of AI and its current use cases. And maybe slowly but surely, the technology will start paving the way into our lives.

Artificial Intelligence Can Make Mistakes

The idea that AI can be inaccurate or go out of control is quite plausible. There is a chance that AI-driven systems can make mistakes and cause harm if anomalies go undetected. You've probably heard about the scenario where AI technology is aimed to go after some virus. In this case, AI is used to study the virus-related genes to create a vaccine for it. Instead of developing a vaccine, AI weaponizes the virus putting humanity at risk.

Take facial recognition technology. The sophisticated technology might oftentimes make mistakes or be inaccurate, if poorly trained. More importantly, when an AI system makes a mistake, it may be really difficult to identify the exact place where something went wrong. Possibly, the best solution might be quality of input, efficient AI software development and powerful bug tracking systems.

AI is Vulnerable to Cyber Attacks

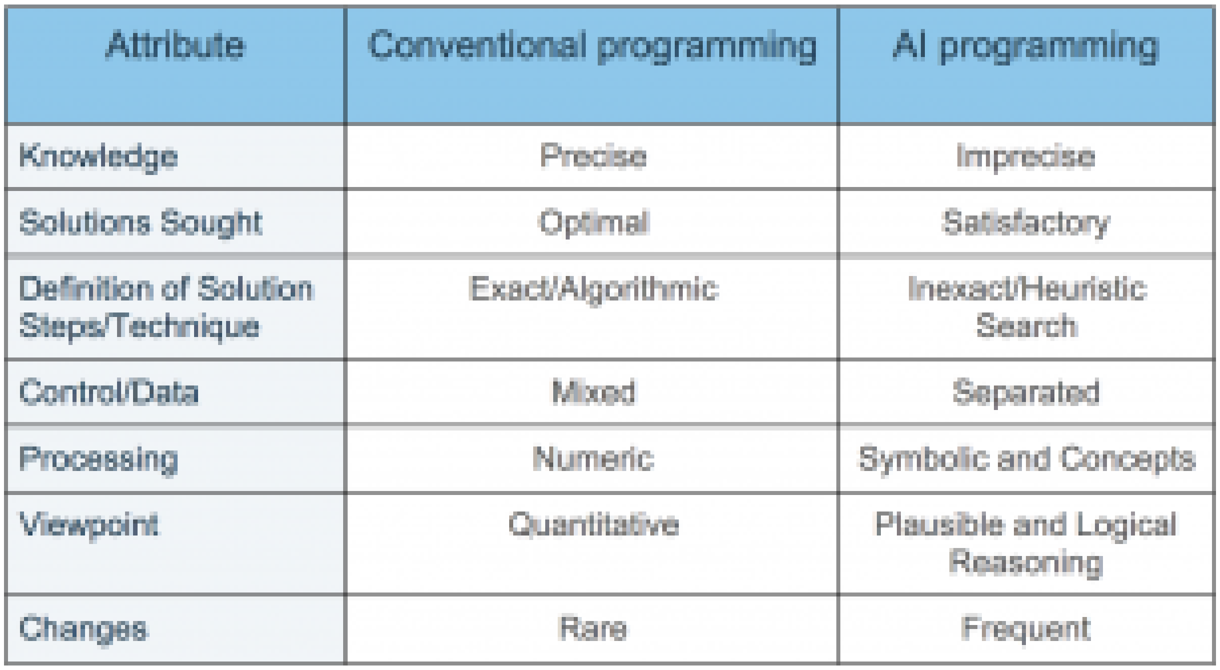
Even the most sophisticated systems have flaws. And AI is no exception to it. Security is paramount in the AI process. It has to be integrated starting from strategic planning throughout the entire cycle. Security is aimed to shield the user's privacy and their business from data leaks.

Many AI apps are based on the wealth of data. And this data is oftentimes sensitive and personal by nature. The thing is, the AI systems rely on data and cannot go without it. And that is the root of the problem. The more AI trains neural networks, the more vulnerabilities come in the code. The systems get prone to data leaks and identity theft. Even with the GDPR having come into existence, the problem of sensitive data is still unsolved. To avoid data leaks, errors and reputational risks, companies need to put AI security first. The possible way out is to test AI products thoroughly and fix flaws before the products come into the practice or out on the market.

Different Development Approach

AI implementation is quite complicated. Even now, in the era of digitization, it still requires time and effort. Most development in a traditional system environment follows the usual phases such as plan, analyze, design, build, test, and deploy. The AI environment is a bit different. Most of the time, development is about identifying data sources and then gathering content, cleansing it and curating it. Such an approach requires different skills and mindsets, as well as different methodologies. In addition, AI-powered intellectual systems have to be trained in a particular domain.

In case we compare conventional (regular) and AI programming, the differences will look the following way:



Attribute	Conventional programming	AI programming
Knowledge	Precise	Imprecise
Solutions Sought	Optimal	Satisfactory
Definition of Solution Steps/Technique	Exact/Algorithmic	Inexact/Heuristic Search
Control/Data	Mixed	Separated
Processing	Numeric	Symbolic and Concepts
Viewpoint	Quantitative	Plausible and Logical Reasoning
Changes	Rare	Frequent

credit: Artificial Intelligence Methods and Applications

Generally speaking, with AI we are not developing a system but training, giving feedback and supervising an AI-powered solution.

A System is Only as Good as the Data it Learns from

Everyone already knows that AI needs data to learn about things. Specifically, data plays a significant role in the implementation of artificial intelligence. AI and machine learning rely on enormous amounts of high-quality data from which to observe trends and behavior patterns, as well as being able to quickly adapt to improve the accuracy of the conclusions derived from the analysis of that data. Basically, first, you get the data, then you get the AI. The worse data, the more problems you have with artificial intelligence. Such systems don't just require more information than humans to understand concepts or recognize features, they require hundreds of thousands of times more. Another important thing is the quality of data used to train predictive models. The data sets need to be extremely representative and balanced, otherwise, the system will eventually adopt bias that those

No Clear View on how Insight is Generated

Artificial intelligence advancements are flourishing, as well as its problems. For example, artificial intelligence development is hiding in its experimental nature. It is difficult to say how much of an improvement it may bring to a project. Therefore it is almost impossible to predict ROI. This makes it really hard to get everyone to understand the whole concept. One thing that is necessary to optimize the result is a skilled team that can write or adapt publicly available algorithms, select the right algorithm for the desired result and combine algorithms as needed to optimize the result.

AI is a robust tool poised to revolutionize the world we live in. There's plenty of innovative solutions AI has brought so far, some are yet to come. Today, AI technology helps a great number of companies meet their needs and stand out from the crowd on the competitive market. But it's no news that there are benefits and dark sides to every disruptive technology, and AI is no exception to this rule. The important thing for every company is to address challenges and make sure that they can take full advantage of the benefits while minimizing the tradeoffs that may impose artificial intelligence problems.

Problem Solving in Artificial Intelligence

The reflex agent of AI directly maps states into action. Whenever these agents fail to operate in an environment where the state of mapping is too large and not easily performed by the agent, then the stated problem dissolves and sent to a problem-solving domain which breaks the large stored problem into the smaller storage area and resolves one by one. The final integrated action will be the desired outcomes.

On the basis of the problem and their working domain, different types of problem-solving agent defined and use at an atomic level without any internal state visible with a problem-solving algorithm. The problem-solving agent performs precisely by defining problems and several solutions. So we can say that problem solving is a part of artificial intelligence that encompasses a number of techniques such as a tree, B-tree, heuristic algorithms to solve a problem.

We can also say that a problem-solving agent is a result-driven agent and always focuses on satisfying the goals.

Steps problem-solving in AI: The problem of AI is directly associated with the nature of humans and their activities. So we need a number of finite steps to solve a problem which makes human easy works. These are the following steps which require to solve a problem :

- **Goal Formulation:** This one is the first and simple step in problem-solving. It organizes finite steps to formulate a target/goals which require some action to achieve the goal. Today the formulation of the goal is based on AI agents.
- **Problem formulation:** It is one of the core steps of problem-solving which decides what action should be taken to achieve the formulated goal. In AI this core part is dependent upon software agent which consisted of the following components to formulate the associated problem.

Components to formulate the associated problem:

- **Initial State:** This state requires an initial state for the problem which starts the AI agent towards a specified goal. In this state new methods also initialize problem domain solving by a specific class.
- **Action:** This stage of problem formulation works with function with a specific class taken from the initial state and all possible actions done in this stage.
- **Transition:** This stage of problem formulation integrates the actual action done by the previous action stage and collects the final stage to forward it to their next stage.
- **Goal test:** This stage determines that the specified goal achieved by the integrated transition model or not, whenever the goal achieves stop the action and forward into the next stage to determines the cost to achieve the goal.
- **Path costing:** This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware software and human working cost.
-

Depth First Search (DFS)

DFS is also an important type of uniform search. DFS visits all the vertices in the graph. This type of algorithm always chooses to go deeper into the graph. After DFS visited all the reachable vertices from a particular sources vertices it chooses one of the remaining undiscovered vertices and continues the search. DFS reminds the space limitation of breath first search by always generating next a child of the deepest unexpanded nodded. The data structure stack or last in first out (LIFO) is used for DFS. One interesting property of DFS is that, the discover and finish time of each vertex from a parenthesis structure. If we use one open parenthesis when a vertex is finished then the result is properly nested set of parenthesis.

Concept:

Step 1: Traverse the root node.

Step 2: Traverse any neighbour of the root node.

Step 3: Traverse any neighbour of neighbour of the root node.

Step 4: This process will continue until we are getting the goal node.

Algorithm:

Step 1: PUSH the starting node into the stack.

Step 2: If the stack is empty then stop and return failure.

Step 3: If the top node of the stack is the goal node, then stop and return success.

Step 4: Else POP the top node from the stack and process it. Find all its neighbours that are in ready state and PUSH them into the stack in any order.

Step 5: Go to step 3.

Step 6: Exit.

Implementation:

Let us take an example for implementing the above DFS algorithm.

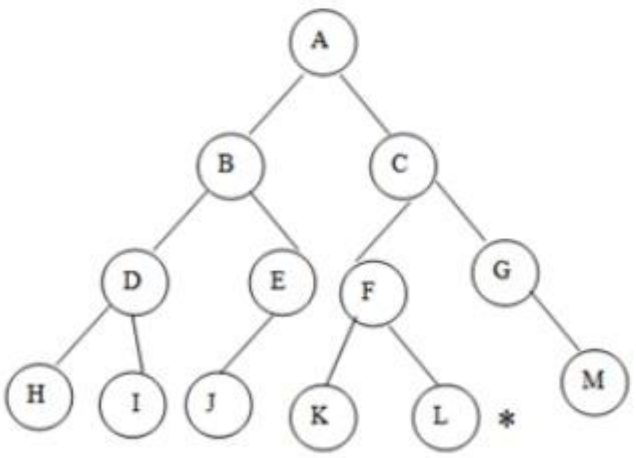


Figure Examples of DFS

Consider A as the root node and L as the goal node in the graph figure

Step 1: PUSH the starting node into the stack i.e.

Step 1: PUSH the starting node into the stack i.e.

A	

Step 2: Now the stack is not empty and A is not our goal node. Hence move to next step.

Step 3: POP the top node from the stack i.e. A and find the neighbours of A i.e. B and C.

B	C	

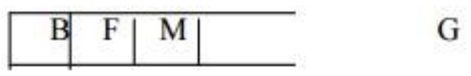
A

Step 4: Now C is top node of the stack. Find its neighbours i.e. F and G.

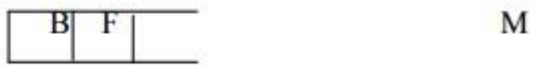
B	F	G	

C

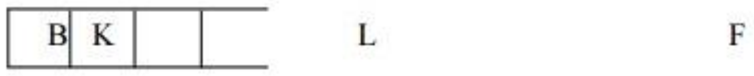
Step 5: Now G is the top node of the stack. Find its neighbour i.e. M



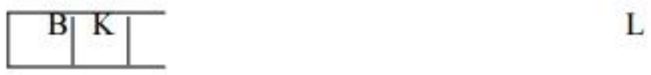
Step 6: Now M is the top node and find its neighbour, but there is no neighbours of M in the graph s
POP it from the stack.



Step 7: Now F is the top node and its neighbours are K and L. so PUSH them on to the stack.



Step 8: Now L is the top node of the stack, which is our goal node.



Also you can traverse the graph starting from the root A and then insert in the order C and B into the stack. Check your answer.

Advantages:

DFS consumes very less memory space.

It will reach at the goal node in a less time period than BFS if it traverses in a right path.

It may find a solution without examining much of search because we may get the desired solution in the very first go.

Disadvantages:

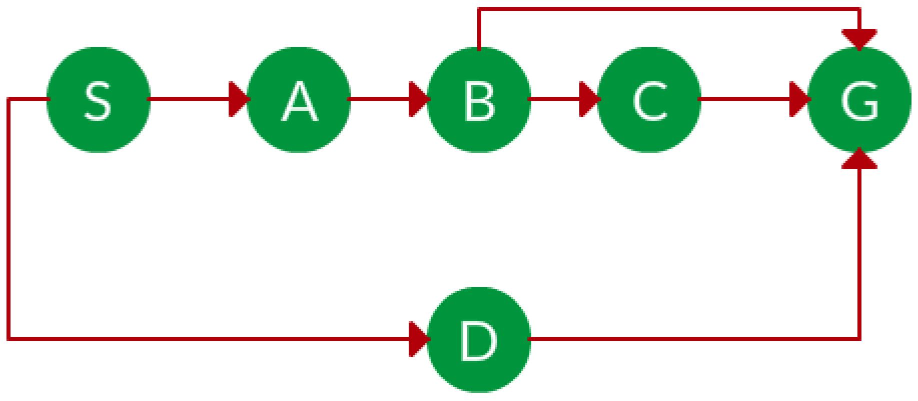
It is possible that may states keep reoccurring. There is no guarantee of finding the goal node.

Sometimes the states may also enter into infinite loops.

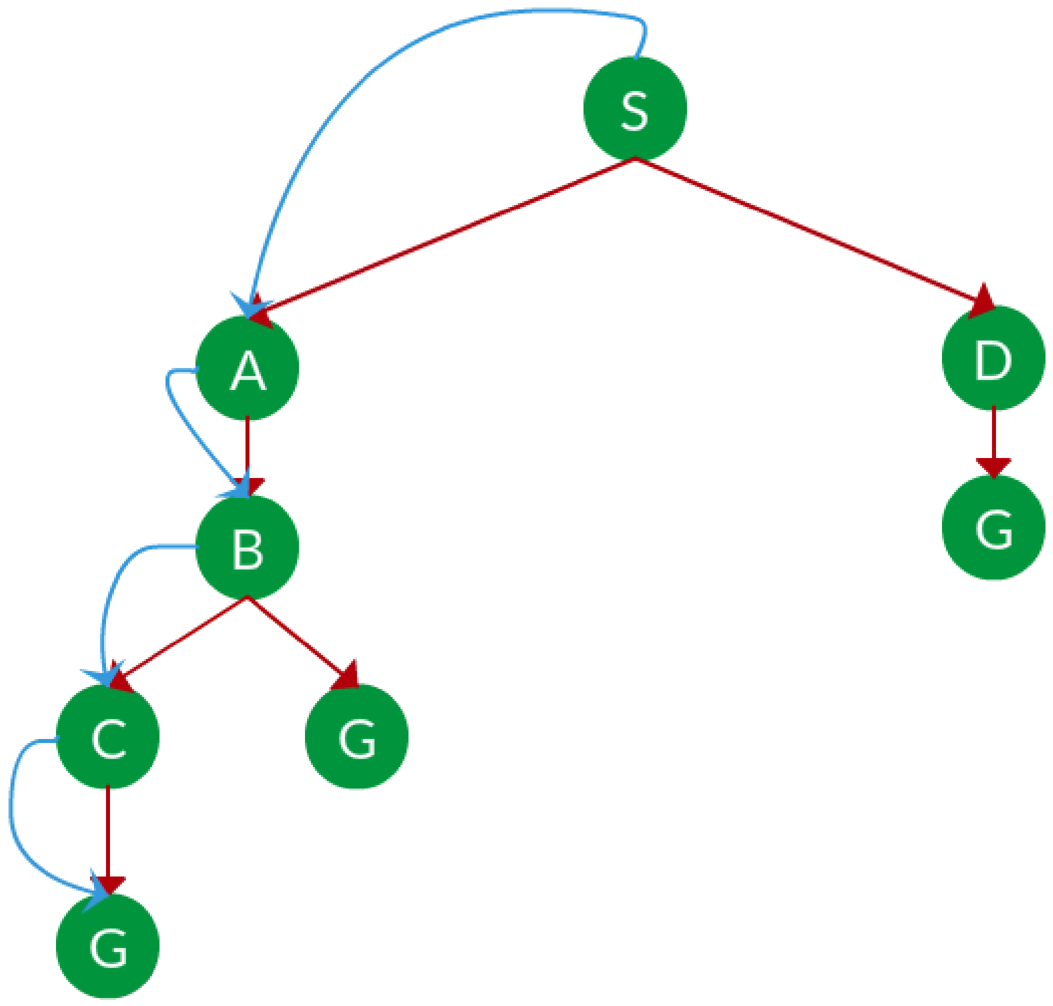
Depth First Search:

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

Example:
Question. Which solution would DFS find to move from node S to node G if run on the graph below?



Solution. The equivalent search tree for the above graph is as follows. As DFS traverses the tree "deepest node first", it would always pick the deeper branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.



Path: S -> A -> B -> C -> G

= the depth of the search tree = the number of levels of the search tree.

= number of nodes in level .

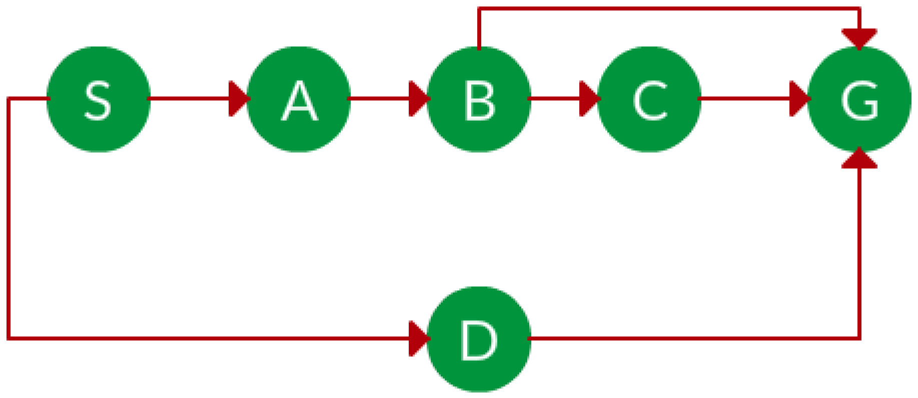
Time complexity: *Equivalent to the number of nodes traversed in*

DFS.

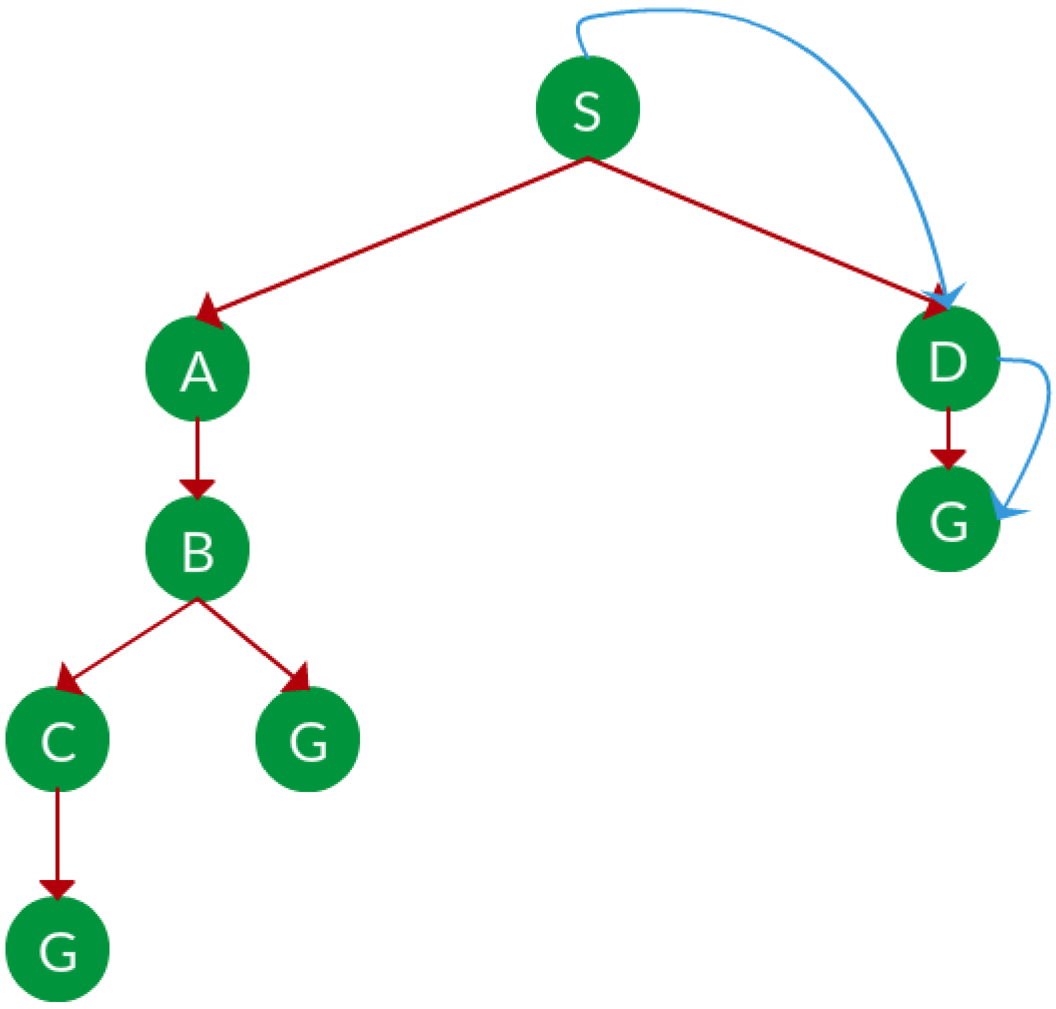
Space complexity: *Equivalent to how large can the fringe get.*
Completeness: *DFS is complete if the search tree is finite, meaning for a given finite search tree, DFS will come up with a solution if it exists.*
Optimality: *DFS is not optimal, meaning the number of steps in reaching the solution, or the cost spent in reaching it is high.*

Breadth First Search:
Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbour nodes at the present depth prior to moving on to the nodes at the next depth level.

Example:
Question. Which solution would BFS find to move from node S to node G if run on the graph below?



Solution. The equivalent search tree for the above graph is as follows. As BFS traverses the tree "shallowest node first", it would always pick the shallower branch until it reaches the solution (or it runs out of nodes, and goes to the next branch). The traversal is shown in blue arrows.



Path: S -> D -> G
 = *the depth of the shallowest solution.*

 = *number of nodes in level .*

Time complexity: *Equivalent to the number of nodes traversed in BFS until the shallowest*

solution.

Space complexity: *Equivalent to how large can the fringe get.*
Completeness: *BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.*
Optimality: *BFS is optimal as long as the costs of all edges are equal.*

Breadth First Search (BFS)

Breadth first search is a general technique of traversing a graph. Breadth first search may use more memory but will always find the shortest path first. In this type of search the state space is represented in form of a tree. The solution is obtained by traversing through the tree. The nodes of the tree represent the start value or starting state, various intermediate states and the final state. In this search a queue data structure is used and it is level by level traversal. Breadth first search expands nodes in order of their distance from the root. It is a path finding algorithm that is capable of always finding the solution if one exists. The solution which is found is always the optional solution. This task is completed in a very memory intensive manner. Each node in the search tree is expanded in a breadth wise at each level.

Concept:

Step 1: Traverse the root node

Step 2: Traverse all neighbours of root node.

Step 3: Traverse all neighbours of neighbours of the root node.

Step 4: This process will continue until we are getting the goal node.

Algorithm:

Step 1: Place the root node inside the queue.

Step 2: If the queue is empty then stops and return failure.

Step 3: If the FRONT node of the queue is a goal node then stop and return success.

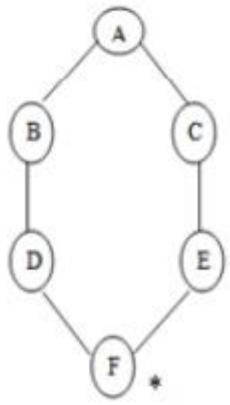
Step 4: Remove the FRONT node from the queue. Process it and find all its neighbours that are in ready state then place them inside the queue in any order.

Step 5: Go to Step 3.

Step 6: Exit.

Implementation:

Let us implement the above algorithm of BFS by taking the following suitable example.

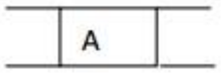


Figure

Consider the graph in which let us take A as the starting node and F as the goal node (*)

Step 1:

Place the root node inside the queue i.e. A

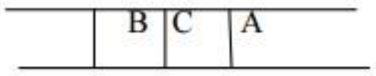


Step 2:

Now the queue is not empty and also the FRONT node i.e. A is not our goal node. So move to step 3.

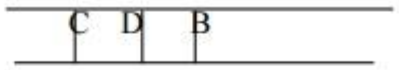
Step 3:

So remove the FRONT node from the queue i.e. A and find the neighbour of A i.e. B and C



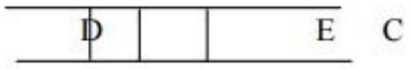
Step 4:

Now b is the FRONT node of the queue .So process B and finds the neighbours of B i.e. D.



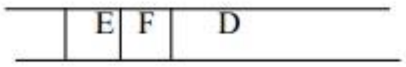
Step 5:

Now find out the neighbours of C i.e. E



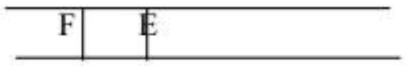
Step 6:

Next find out the neighbours of D as D is the FRONT node of the queue



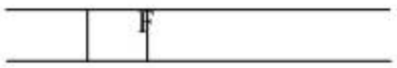
Step 7:

Now E is the front node of the queue. So the neighbour of E is F which is our goal node.



Step 8:

Finally F is our goal node which is the FRONT of the queue. So exit.



Advantages:

In this procedure at any way it will find the goal.

It does not follow a single unfruitful path for a long time. It finds the minimal solution in case of multiple paths.

Disadvantages:

BFS consumes large memory space. Its time complexity is more.

It has long pathways, when all paths to a destination are on approximately the same search depth.

<u>UNIT 4</u>

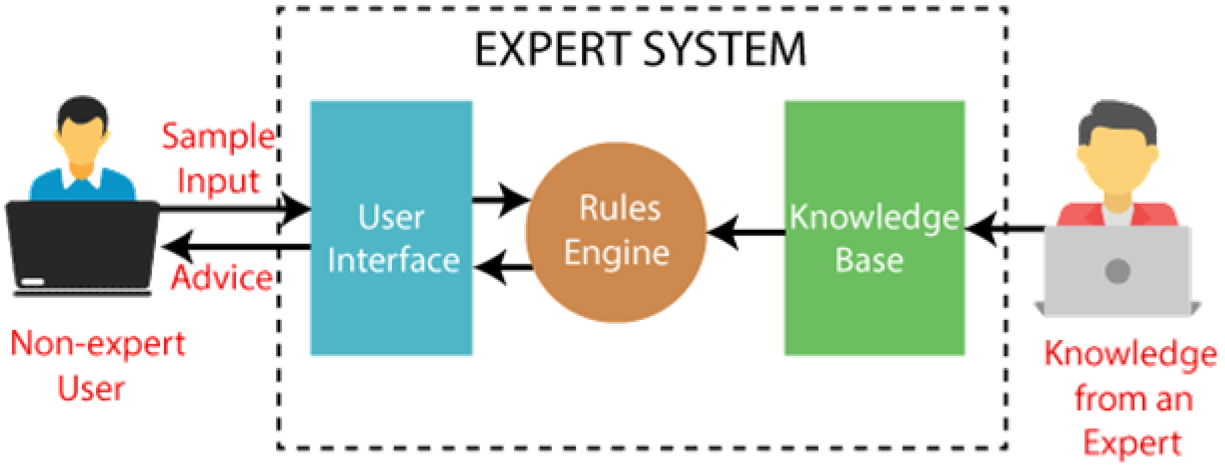
What is an Expert System?

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.

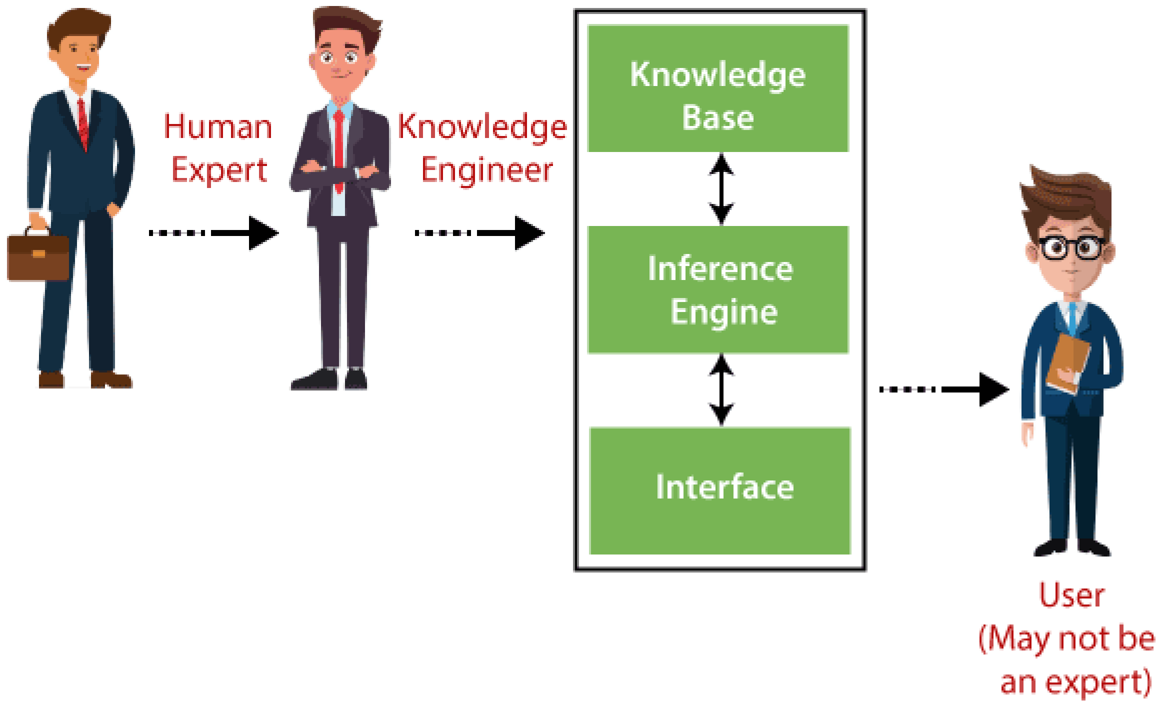
Characteristics of Expert System

- **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.

3. 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.

Development of Expert System

Here, we will explain the working of an expert system by taking an example of MYCIN ES. Below are some steps to build an MYCIN:

- Firstly, ES should be fed with expert knowledge. In the case of MYCIN, human experts specialized in the medical field of bacterial infection, provide information about the causes, symptoms, and other knowledge in that domain.
- The KB of the MYCIN is updated successfully. In order to test it, the doctor provides a new problem to it. The problem is to identify the presence of the bacteria by inputting the details of a patient, including the symptoms, current condition, and medical history.

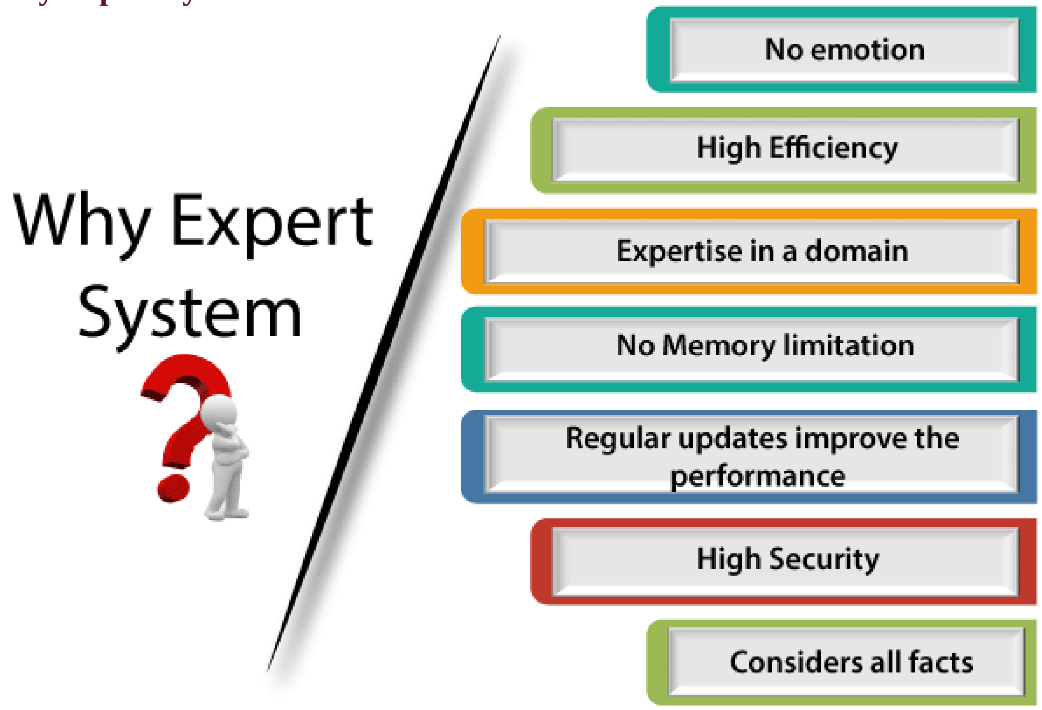
- The ES will need a questionnaire to be filled by the patient to know the general information about the patient, such as gender, age, etc.
- Now the system has collected all the information, so it will find the solution for the problem by applying if-then rules using the inference engine and using the facts stored within the KB.
- In the end, it will provide a response to the patient by using the user interface.

Participants in the development of Expert System

There are three primary participants in the building of Expert System:

1. **Expert:** The success of an ES much depends on the knowledge provided by human experts. These experts are those persons who are specialized in that specific domain.
2. **Knowledge Engineer:** Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.
3. **End-User:** This is a particular person or a group of people who may not be experts, and working on the expert system needs the solution or advice for his queries, which are complex.

Why Expert System?



Before using any technology, we must have an idea about why to use that technology and hence the same for the ES. Although we have human experts in every field, then what is the need to develop a computer-based system. So below are the points that are describing the need of the ES:

1. **No memory Limitations:** It can store as much data as required and can memorize it at the time of its application. But for human experts, there are some limitations to memorize all things at every time.
2. **High Efficiency:** If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.
3. **Expertise in a domain:** There are lots of human experts in each domain, and they all have different skills, different experiences, and different skills, so it is not easy to get a final output for the query. But if we put the knowledge gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge
4. **Not affected by emotions:** These systems are not affected by human emotions such as fatigue, anger, depression, anxiety, etc.. Hence the performance remains constant.
5. **High security:** These systems provide high security to resolve any query.
6. **Considers all the facts:** To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.
7. **Regular updates improve the performance:** If there is an issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

Capabilities of the Expert System

Below are some capabilities of an Expert System:

- **Advising:** It is capable of advising the human being for the query of any domain from the particular ES.
- **Provide decision-making capabilities:** It provides the capability of decision making in any domain, such as for making any financial decision, decisions in medical science, etc.
- **Demonstrate a device:** It is capable of demonstrating any new products such as its features, specifications, how to use that product, etc.
- **Problem-solving:** It has problem-solving capabilities.
- **Explaining a problem:** It is also capable of providing a detailed description of an input problem.
- **Interpreting the input:** It is capable of interpreting the input given by the user.
- **Predicting results:** It can be used for the prediction of a result.
- **Diagnosis:** An ES designed for the medical field is capable of diagnosing a disease without using multiple components as it already contains various inbuilt medical tools.

Advantages of Expert System

- These systems are highly reproducible.
- They can be used for risky places where the human presence is not safe.
- Error possibilities are less if the KB contains correct knowledge.
- The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- They provide a very high speed to respond to a particular query.

Limitations of Expert System

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.
- Its maintenance and development costs are very high.
- Knowledge acquisition for designing is much difficult.
- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

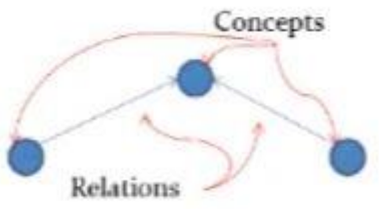
Applications of Expert System

- **Indesigningandmanufacturingdomain**
  It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.
- **Intheknowledgedomain**
  These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.
- **Inthefinancedomain**
  In the finance industries, it is used to detect any type of possible fraud, suspicious activity, and advise bankers that if they should provide loans for business or not.
- **Inthediagnosisandtroubleshootingofdevices**
  In medical diagnosis, the ES system is used, and it was the first area where these systems were used.
- **PlanningandScheduling**
  The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

- ○
- ○

SemanticNetwork

- A semantic network or net is a graph structure for representing knowledge in patterns of interconnected nodes and arcs.
- Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics.
- The Giant Global of the Semantic Web is a large semantic network.
- What is common to all semantic networks is a declarative graphic representation that can be used to represent knowledge and support automated systems for reasoning about the knowledge.
- Some versions are highly informal, but others are formally defined systems of logic.
- It is define objects in terms of their association with other objects ex: snow, white, snowman, ice, slippery.
- Represent knowledge as a graph:



- Concepts at lower levels inherit characteristics from their parent concepts.

Six most common kinds of semantic networks:

1: Definitionalnetworks:

- Emphasize the subtype or is-a relation between a concept type and a newly defined subtype.
- The resulting network, also called a generalization or subsumption hierarchy, supports the rule of inheritance for copying properties defined for a supertype to all of its subtypes.
- Since definition are true by definition, the information in these networks is often assumed to be necessarily true.

2: **Assertionalnetworks:**

- Are designed to assert propositions.Unlike definitional network, the information in an assertional network is assumed to be contingently true, unless it is explicitly marked with a modal operator.
- Some assertional networks have been proposed as models of the conceptual structures underlying natural language semantics.

3: **Implicationalnetworks:**

- Use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.
- Implicational networks emphasize implication, they are capable of expressing all the Boolean connectives by allowing a conjunction of inputs to a propositional node and a disjunction of outputs.

4: **Executablenetworks:**

- Include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations.
- Executable semantic networks contain mechanisms that can cause some change to the network itself.

5: **Learningnetworks:**

- Build or extend their representations by acquiring knowledge from examples.The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called weights, associated with the nodes and arcs.
- The purpose of learning, both from a natural and AI standpoint, is to create modifications that enable the system to respond more effectively within its environment.

6: **Hybridnetworks:**

- Combine two or more of the previous techniques, either in a single network or in separate, but closely interacting networks..
- System are usually called hybrids if their component languages have different syntax... The most widely used hybrid of multiple network notations is the Unified Modeling Language (UML), which was by designed by three authors....who merged their competing notations.

Semantic Networks Example

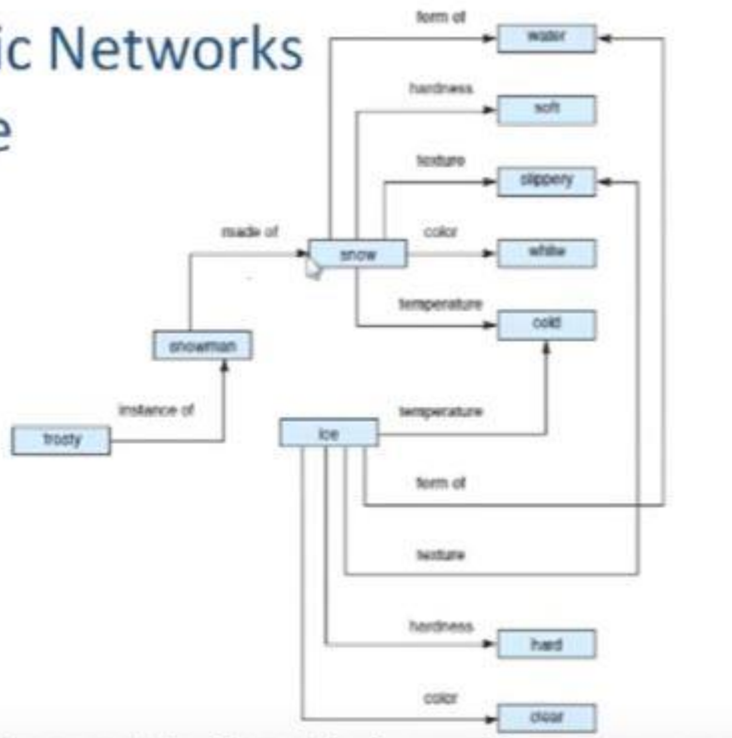
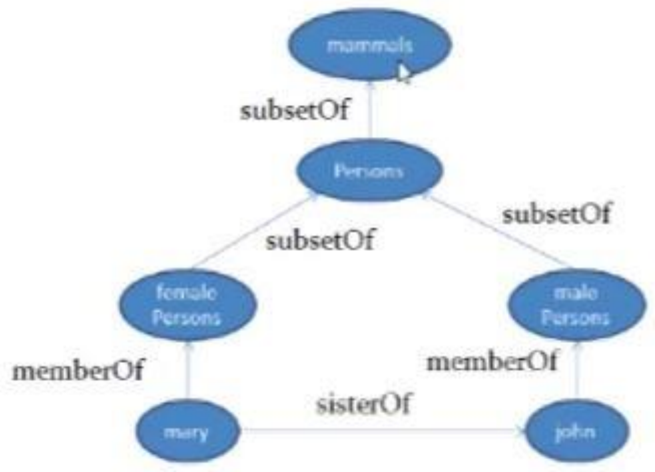


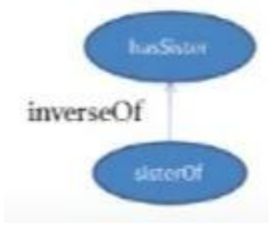
Fig 7.2 Network representation of properties of snow and ice (From: Luger: Artificial Intelligence, 6th edition. © Pearson Education Limited, 2009)



Inference Mechanism

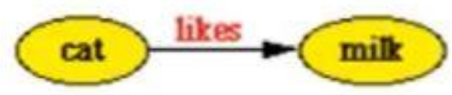
- Inheritance
  - e.g. Person by default have 2 legs. How many legs does Mary have ? john?
- Use of inverse Links (through reification)

- e.g. hasSister (p,s) and sisterOf (s,p)

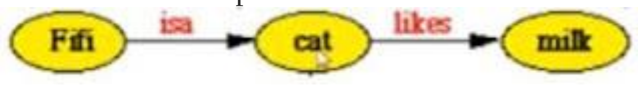


Simple semantics nets

- Nodes are labeled with names (nouns).
- Arcs labeled with relationships.



- Special link label "isa" means "is a".
- Show membership or subset relationships



Advantages

- Simple and transparent inference processes.
- Ability to assign default values for categories.
- Ability to include procedural attachment.

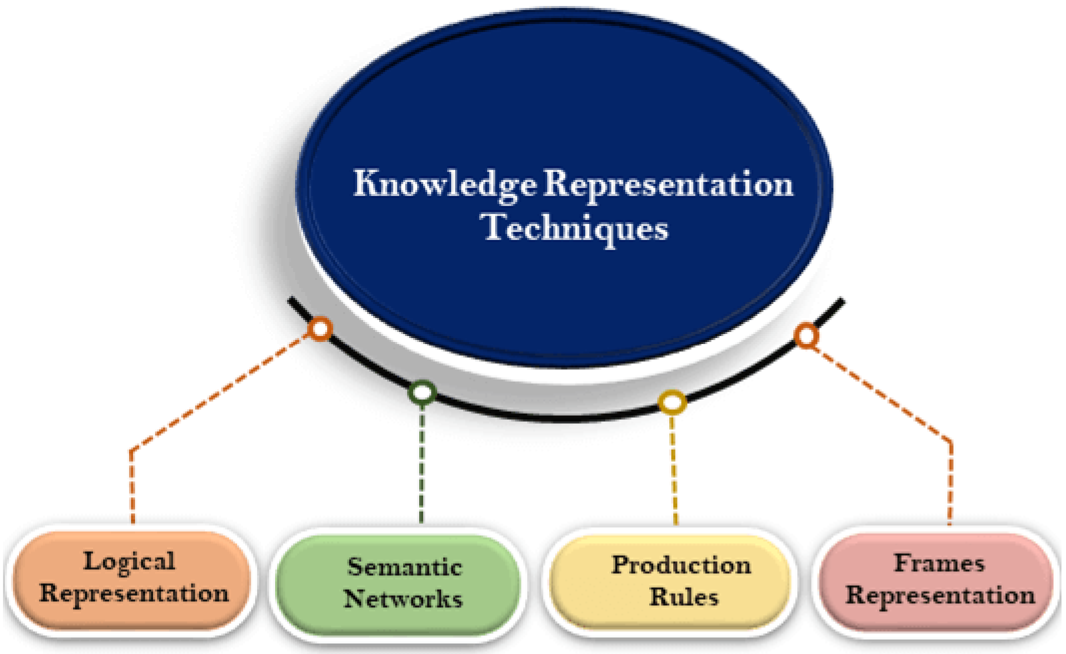
Disadvantage

- Simple query language may be too limiting to express complex queries.
- Does not represent full FOL since it does not provide means to use negation, disjuction, and existential qualification.
- n-ary functions must be mapped onto binary functions.

Techniques of knowledge representation

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules



1. Logical Representation

Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules. It consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

Syntax:

- Syntaxes are the rules which decide how we can construct legal sentences in the logic.
- It determines which symbol we can use in knowledge representation.
- How to write those symbols.

Semantics:

- Semantics are the rules by which we can interpret the sentence in the logic.
- Semantic also involves assigning a meaning to each sentence.

Logical representation can be categorised into mainly two logics:

a. Propositional Logics

b. Predicate logic

Note: We will discuss Prepositional Logics and Predicate logics in later chapters.

Advantages of logical representation:

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

Disadvantages of logical Representation:

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.

Note: Do not be confused with logical representation and logical reasoning as logical representation is a representation language and reasoning is a process of thinking logically.

2. Semantic Network Representation

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

This representation consist of mainly two types of relations:

a. IS-A relation (Inheritance)

b. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

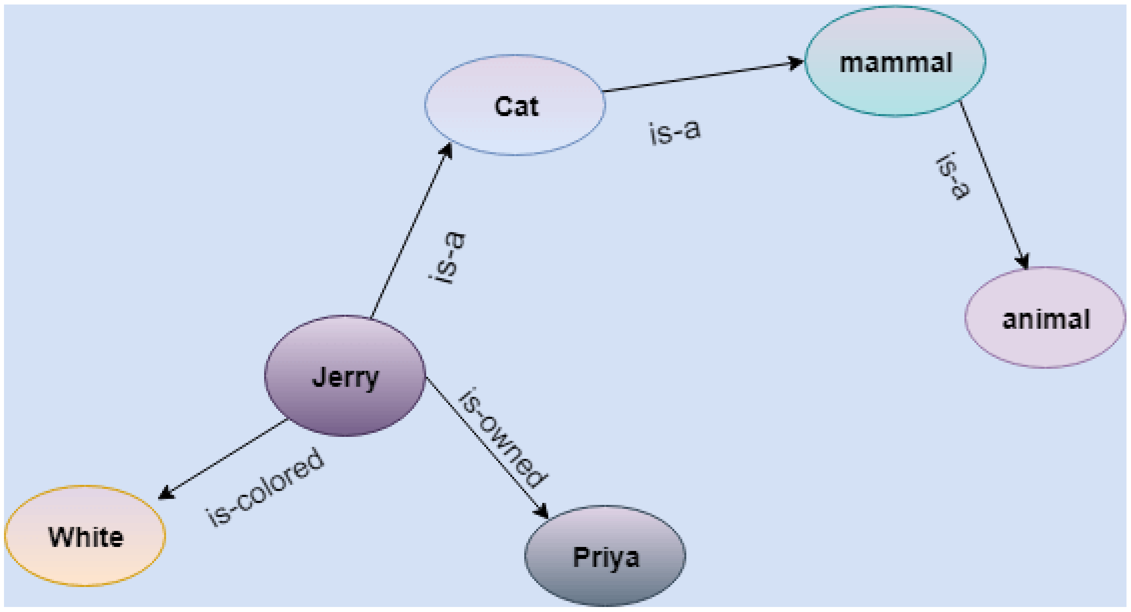
a. Jerry is a cat.

b. Jerry is a mammal

c. Jerry is owned by Priya.

d. Jerry is brown colored.

e. All Mammals are animal.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

3. Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

Facets: The various aspects of a slot is known as **Facets**. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example: 1

Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Example 2:

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

4. Production Rules

Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- Working Memory
- The recognize-act-cycle

In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.

If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

Example:

- **IF (at bus stop AND bus arrives) THEN action (get into the bus)**
- **IF (on the bus AND paid AND empty seat) THEN action (sit down).**
- **IF (on bus AND unpaid) THEN action (pay charges).**
- **IF (bus arrives at destination) THEN action (get down from the bus).**

Advantages of Production rule:

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

Disadvantages of Production rule:

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. During the execution of the program, many rules may be active hence rule-based production systems are inefficient.

Speech Recognition in AI: What you Need to Know?

Speech recognition refers to a computer interpreting the words spoken by a person and converting them to a format that is understandable by a machine. Depending on the end-goal, it is then converted to text or voice or another required format.
For instance, Apple's Siri and Google's Alexa use AI-powered speech recognition to provide voice or text support whereas voice-to-text applications like Google Dictate transcribe your dictated words to text. Voice recognition is another form of speech recognition where a source sound is recognized and matched to a person's voice.

Speech recognition AI applications have seen significant growth in numbers in recent times as businesses are increasingly adopting digital assistants and automated support to streamline their services. Voice assistants, smart home devices, search engines, etc are a few examples where speech recognition has seen prominence. As per Research and Markets, the global market for speech recognition is estimated to grow at a CAGR of 17.2% and reach $26.8 billion by 2025.

Speech Recognition and Artificial Intelligence

Speech recognition is fast overcoming the challenges of poor recording equipment and noise cancellation, variations in people's voices, accents, dialects, semantics, contexts, etc using artificial intelligence and machine learning. This also includes challenges of understanding human disposition, and the varying human language elements like colloquialisms, acronyms, etc. The technology can provide a 95% accuracy now as compared to traditional models of speech recognition, which is at par with regular human communication.

Furthermore, it is now an acceptable format of communication given the large companies that endorse it and regularly employ speech recognition in their operations. It is estimated that a majority of search engines will adopt voice technology as an integral aspect of their search mechanism.

This has been made possible because of improved AI and machine learning (ML) algorithms which can process significantly large datasets and provide greater accuracy by self-learning and adapting to evolving changes. Machines are programmed to "listen" to accents, dialects, contexts, emotions and process sophisticated and arbitrary data that is readily accessible for mining and machine learning purposes.

Speech Recognition and Natural Language Processing

Natural language processing (NLP) is a division of artificial intelligence that involves analyzing natural language data and converting it into a machine-readable format. Speech recognition and AI play an integral role in NLP models in improving the accuracy and efficiency of human language recognition. From smart home devices and appliances that take instructions, and can be switched on and off remotely, digital assistants that can set reminders, schedule meetings, recognize a song playing in a pub, to search engines that respond with relevant search results to user queries, speech recognition has become an indispensable part of our lives.

Plenty of businesses now include speech-to-text software to enhance their business applications and streamline the customer experience. Using speech recognition and natural language processing, companies can transcribe calls, meetings, and even translate them. Apple, Google, Facebook, Microsoft, and Amazon are among the tech giants who continue to leverage AI-backed speech recognition applications to provide an exemplary user experience.

Use Cases of Speech Recognition

Let's explore the uses of speech recognition applications in different fields:

1. Voice-based speech recognition software is now used to initiate purchases, send emails, transcribe meetings, doctor appointments, and court proceedings, etc.
2. Virtual assistants or digital assistants and smart home devices use voice recognition software to answer questions, provide weather news, play music, check traffic, place an order, and so on.
3. Companies like Venmo and PayPal allow customers to make transactions using voice assistants. Several banks in North America and Canada also provide online banking using voice-based software.
4. Ecommerce is significantly powered by voice-based assistants and allows users to make purchases quickly and seamlessly.
5. Speech recognition is poised to impact transportation services and streamline scheduling, routing, and navigating across cities.

6. Podcasts, meetings, and journalist interviews can be transcribed using voice recognition. It is also used to provide accurate subtitles to a video.
7. There has been a huge impact on security through voice biometry where the technology analyses the varying frequencies, tone and pitch of an individual's voice to create a voice profile. An example of this is Switzerland's telecom company Swisscom which has enabled voice authentication technology in its call centres to prevent security breaches.
8. Customer care services are being traced by AI-based voice assistants, and chatbots to automate repeatable tasks.

Other industries that are actively investing in voice-based speech recognition technologies are law enforcement, marketing, tourism, content creation, and translation.

Global Impact of Speech Recognition in Artificial Intelligence

Speech recognition has by far been one of the most powerful products of technological advancement. As the likes of Siri, Alexa, Echo Dot, Google Assistant, and Google Dictate continue to make our daily lives easier, the demand for such automated technologies is only bound to increase.

Businesses worldwide are investing in automating their services to improve operational efficiency, increase productivity and accuracy, and make data-driven decisions by studying customer behaviours and purchasing habits.

AI has facilitated an exponential growth in a wide range of sectors of the global economy. It is estimated that AI's contribution to the global economy will hit $15.7 trillion in 2030, which is significantly higher than China and India's combined output.

The future of speech recognition is tremendously noteworthy. As per reports, Apple has plans to launch the Siri-controlled Apple TV, there will be a rise in smart wearable devices like watches, earbuds, jewellery, and voice-based software that are being programmed to identify the context of user requests to provide enhanced support.

As speech recognition and AI impact both professional and personal lives at workplaces and homes respectively, the demand for skilled AI engineers and developers, Data Scientists, and Machine Learning Engineers, is expected to be at an all-time high.

There will be a requirement for skilled AI professionals to enhance the relationship between humans and digital devices. As job opportunities are created, they will result in increased perks and benefits for those in this field.

As per PayScale, the average salary for an Artificial Intelligence professional in India today is ₹15 lakh. Furthermore, the field offers lucrative career advancement opportunities, both financially and profile-wise. However, this requires investing in an Artificial Intelligence course to master Data Science and learn to create intuitive, human-like software solutions using real-time data.

Inductive Learning Algorithm

Inductive Learning Algorithm (ILA) is an iterative and inductive machine learning algorithm which is used for generating a set of a classification rule, which produces rules of the form "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

Basic Idea:

There are basically two methods for knowledge extraction firstly from domain experts and then with machine learning.

For a very large amount of data, the domain experts are not very useful and reliable. So we move towards the machine learning approach for this work.

To use machine learning One method is to replicate the experts logic in the form of algorithms but this work is very tedious, time taking and expensive.

So we move towards the inductive algorithms which itself generate the strategy for performing a task and need not instruct separately at each step.

Need of ILA in presence of other machine learning algorithms:
The ILA is a new algorithm which was needed even when other reinforcement learnings like ID3 and AQ were available.

- The need was due to the pitfalls which were present in the previous algorithms, one of the major pitfalls was lack of generalisation of rules.
- The ID3 and AQ used the decision tree production method which was too specific which were difficult to analyse and was very slow to perform for basic short classification problems.
- The decision tree-based algorithm was unable to work for a new problem if some attributes are missing.
- The ILA uses the method of production of a general set of rules instead of decision trees, which overcome the above problems

THE ILA ALGORITHM:
General requirements at start of the algorithm:-

1. list the examples in the form of a table 'T' where each row corresponds to an example and each column contains an attribute value.
2. create a set of m training examples, each example composed of k attributes and a class attribute with n possible decisions.
3. create a rule set, R, having the initial value false.
4. initially all rows in the table are unmarked.

Steps in the algorithm:-
Step 1:
divide the table 'T' containing m examples into n sub-tables (t1, t2,…..tn). One table for each possible value of the class attribute. (repeat steps 2-8 for each sub-table)
Step 2:
Initialize the attribute combination count ' j ' = 1.
Step 3:
For the sub-table on which work is going on, divide the attribute list into distinct combinations, each combination with 'j ' distinct attributes.
Step 4:
For each combination of attributes, count the number of occurrences of attribute values that appear under the same combination of attributes in unmarked rows of the sub-table under consideration, and at the same time, not appears under the same combination of attributes of other sub-tables. Call the first combination with the maximum number of occurrences the max-combination ' MAX'.
Step 5:
If 'MAX' = = null , increase ' j ' by 1 and go to Step 3.
Step 6:
Mark all rows of the sub-table where working, in which the values of 'MAX' appear, as classi?ed.
Step 7:
Add a rule (IF attribute = "XYZ" –> THEN decision is YES/ NO) to R whose left-hand side will have attribute names of the 'MAX' with their values separated by AND, and its right-hand side contains the decision attribute value associated with the sub-table.
Step 8:
If all rows are marked as classi?ed, then move on to process another sub-table and go to Step 2. else, go to Step 4. If no sub-tables are available, exit with the set of rules obtained till then.

An example showing the use of ILA
suppose an example set having attributes Place type, weather, location, decision and seven examples, our task is to generate a set of rules that under what condition what is the decision.

Example no.	Place type	weather	location	decision
I)	hilly	winter	kullu	Yes
II)	mountain	windy	Mumbai	No
III)	mountain	windy	Shimla	Yes
IV)	beach	windy	Mumbai	No
V)	beach	warm	goa	Yes
VI)	beach	windy	goa	No
VII)	beach	warm	Shimla	Yes

step 1
subset 1

s.no	place type	weather	location	decision
1	hilly	winter	kullu	Yes
2	mountain	windy	Shimla	Yes
3	beach	warm	goa	Yes

s.no	place type	weather	location	decision
4	beach	warm	Shimla	Yes

subset 2

s.no	place type	weather	location	decision
5	mountain	windy	Mumbai	No
6	beach	windy	Mumbai	No
7	beach	windy	goa	No

step (2-8)
at iteration 1

row 3 & 4 column weather is selected and row 3 & 4 are marked. the rule is added to R IF weather is warm then a decision is yes.

at iteration 2
row 1 column place type is selected and row 1 is marked.
the rule is added to R IF place type is hilly then the decision is yes.

at iteration 3
row 2 column location is selected and row 2 is marked.
the rule is added to R IF location is Shimla then the decision is yes.

at iteration 4
row 5&6 column location is selected and row 5&6 are marked.
the rule is added to R IF location is Mumbai then a decision is no.

at iteration 5
row 7 column place type & the weather is selected and row 7 is marked.
rule is added to R IF place type is beach AND weather is windy then the decision is no.

finally we get the rule set :-
Rule Set

- **Rule 1:** IF the weather is warm THEN the decision is yes.
- **Rule 2:** IF place type is hilly THEN the decision is yes.
- **Rule 3:** IF location is Shimla THEN the decision is yes.
- **Rule 4:** IF location is Mumbai THEN the decision is no.

- **Rule 5:** IF place type is beach AND the weather is windy THEN the decision is no.

Decision Tree in AI: Introduction, Types & Creation

A Decision tree is the denotative representation of a decision-making process. Decision trees in artificial intelligence are used to arrive at conclusions based on the data available from decisions made in the past. Further, these conclusions are assigned values, deployed to predict the course of action likely to be taken in the future.

Decision trees are statistical, algorithmic models of machine learning that interpret and learn responses from various problems and their possible consequences. As a result, decision trees know the rules of decision-making in specific contexts based on the available data. The learning process is continuous and based on feedback. This improves the outcome of learning over time. This kind of learning is called supervised learning. Therefore, decision tree models are support tools for supervised learning.

Thus, decision trees provide a scientific decision-making process based on facts and values rather than intuition. In business, organizations use this process to make significant business decisions.

Type of Decision Tree Models

These models can be used to solve problems depending upon the kind of data that requires prediction. They fall into the following categories:

1. **Prediction of continuous variables**
2. **Prediction of categorical variables**

1. Prediction of Continuous Variables

The prediction of continuous variables depends on one or more predictors. For instance, the prices of houses in an area may depend on many variables such as an address, availability of amenities like a swimming pool, number of rooms, etc. In this case, the decision tree will predict a house's price based on various variable values. The predicted value will also be a variable value.

The decision tree model used to indicate such values is called a continuous variable decision tree. Continuous various decision trees solve regression-type problems. In such cases, labeled datasets are used to predict a continuous, variable, and numbered output.

2. Prediction of Categorical Variables

The prediction of categorical variables is also based on other categorical or continuous variables. However, instead of predicting a value, this problem is about classifying a new dataset into the available classes of datasets. For example, analyzing a comment on Facebook to classify text as negative or supportive. Performing diagnosis for illness based on a patient's symptoms is also an example of a categorical variable decision tree model. Categorical variable decision trees solve classification-type problems where the output is a class instead of a value.

How Decision Trees in Artificial Intelligence Are Created

As the name suggests, the decision tree algorithm is in the form of a tree-like structure. Yet, it is inverted. A decision tree starts from the root or the top decision node that classifies data sets based on the values of carefully selected attributes.

The root node represents the entire dataset. This is where the first step in the algorithm selects the best predictor variable. It makes it a decision node. It also classifies the whole dataset into various classes or smaller datasets.
The set of criteria for selecting attributes is called Attribute Selection Measures (ASM). ASM is based on selection measures, including information gain, entropy, Gini index, Gain ratio, and so on. These attributes, also called features, create decision rules that help in branching. The branching process splits the root node into sub-nodes, splitting further into more sub-nodes until leaf nodes are formed. Leaf nodes cannot be divided further.

Determining whether a given picture is that of a cat or a dog is a typical example of classification. Here, the features or attributes could be the presence of claws or paws, length of ears, type of tongue, etc. The dataset will be split further into smaller classes based on these input variables until the result is obtained.

Conclusion

Decision trees are classic and natural learning models. They are based on the fundamental concept of divide and conquer. In the world of artificial intelligence, decision trees are used to develop learning machines by teaching them how to determine success and failure. These learning machines then analyze incoming data and store it.
Then, they make innumerable decisions based on past learning experiences. These decisions form the basis for predictive modeling that helps to predict outcomes for problems. In business, organizations use these techniques to make innumerable small and big business decisions leading to giant gains or losses.

Introduction to Single Layer Perceptron

In this article we will go through a single-layer perceptron this is the first and basic model of the artificial neural networks. It is also called the feed-forward neural network. The working of the single-layer perceptron (SLP) is based on the threshold transfer between the nodes. This is the simplest form of ANN and it is generally used in the linearly based cases for the machine learning problems.

Working of Single Layer Perceptron

The SLP looks like the below:

Let’s understand the algorithms behind the working of Single Layer Perceptron:

- In a single layer perceptron, the weights to each input node are assigned randomly since there is no a priori knowledge associated with the nodes.
- Also, a threshold value is assigned randomly

- Now SLP sums all the weights which are inputted and if the sums are is above the threshold then the network is activated.
- If the calculated value is matched with the desired value, then the model is successful
- If it is not, then since there is no back-propagation technique involved in this the error needs to be calculated using the below formula and the weights need to be adjusted again.

Perceptron Weight Adjustment

Below is the equation in Perceptron weight adjustment:

$\Delta w = \eta * d * x$

Where,

- **d:** Predicted Output – Desired Output
- **η:** Learning Rate, Usually Less than 1.
- **x:** Input Data.

Since this network model works with the linear classification and if the data is not linearly separable, then this model will not show the proper results.

Example to Implement Single Layer Perceptron

Let's understand the working of SLP with a coding example:

We will solve the problem of the XOR logic gate using the Single Layer Perceptron. In the below code we are not using any machine learning or deep learning libraries we are simply using python code to create the neural network for the prediction.

Let's first see the logic of the XOR logic gate:

- **1 1 —> 0**
- **1 0 —> 1**
- **0 1 —> 1**
- **0 0 —> 0**

XOR is the Reverse of OR Gate so:

- If Both the inputs are True then output is false
- If Both the inputs are false then output is True.
- If Any One of the inputs is true, then output is true.

So, let's get started with the coding

- We are using the two libraries for the import that is the NumPy module for the linear algebra calculation and matplotlib library for the plotting the graph.

```
import numpy as np

import matplotlib.pyplot as plt
```

- Defining the inputs that are the input variables to the neural network

```
X = np.array([[1,1,0],[1,0,1],[1,0,0],[1,1,1]])
```

- Similarly, we will create the output layer of the neural network with the below code

```
y = np.array([[1],[1],[0],[0]])
```

- Now we will right the activation function which is the sigmoid function for the network

```
def sigmoid(x):

return 1/(1 + np.exp(-x))
```

- The function basically returns the exponential of the negative of the inputted value
- Now we will write the function to calculate the derivative of the sigmoid function for the backpropagation of the network

```
def sigmoid_deriv(x):

return sigmoid(x)*(1-sigmoid(x))
```

- This function will return the derivative of sigmoid which was calculated by the previous function
- Function for the feed-forward network which will also handle the biases

```
def forward(x,w1,w2,predict=False):

a1 = np.matmul(x,w1)

z1 = sigmoid(a1)

bias = np.ones((len(z1),1))

z1 = np.concatenate((bias,z1),axis=1)

a2 = np.matmul(z1,w2)

z2 = sigmoid(a2)

if predict:

return z2

return a1,z1,a2,z2
```

- Now we will write the function for the backpropagation where the sigmoid derivative is also multiplied so that if the expected output is not matched with the desired output then the network can learn in the techniques of backpropagation

```
def backprop(a2,z0,z1,z2,y):

delta2 = z2 - y

Delta2 = np.matmul(z1.T,delta2)

delta1 = (delta2.dot(w2[1:,:].T))*sigmoid_deriv(a1)

Delta1 = np.matmul(z0.T,delta1)

return delta2,Delta1,Delta2
```

- Now we will initialize the weights in LSP the weights are randomly assigned so we will do the same by using the random function

```
w1 = np.random.randn(3,5)

w2 = np.random.randn(6,1)
```

- Now we will initialize the learning rate for our algorithm this is also just an arbitrary number between 0 and 1

```
lr = 0.89

costs = []
```

- Once the learning rate is finalized then we will train our model using the below code.

```
epochs = 15000

m = len(X)

for i in range(epochs):

a1,z1,a2,z2 = forward(X,w1,w2)

delta2,Delta1,Delta2 = backprop(a2,X,z1,z2,y)

w1 -= lr*(1/m)*Delta1

w2 -= lr*(1/m)*Delta2

c = np.mean(np.abs(delta2))

costs.append(c)

if i % 1000 == 0:

print(f"iteration: {i}. Error: {c}")

print("Training complete")
```

- Once the model is trained then we will plot the graph to see the error rate and the loss in the learning rate of the algorithm

```
z3 = forward(X,w1,w2,True)

print("Precentages: ")

print(z3)

print("Predictions: ")

print(np.round(z3))
```

```
plt.plot(costs)

plt.show()
```

Example

The above lines of code depicted are shown below in the form of a single program:

```
import numpy as np

import matplotlib.pyplot as plt

#nneural network for solving xor problem

#the xor logic gate is

# 1 1 ---> 0

# 1 0 ---> 1

# 0 1 ---> 1

# 0 0 ---> 0

#Activation funtion

def sigmoid(x):

return 1/(1 + np.exp(-x))

#sigmoid derivative for backpropogation

def sigmoid_deriv(x):

return sigmoid(x)*(1-sigmoid(x))

#the forward funtion
```

```
def forward(x,w1,w2,predict=False):

a1 = np.matmul(x,w1)

z1 = sigmoid(a1)

#create and add bais

bias = np.ones((len(z1),1))

z1 = np.concatenate((bias,z1),axis=1)

a2 = np.matmul(z1,w2)

z2 = sigmoid(a2)

if predict:

return z2

return a1,z1,a2,z2

def backprop(a2,z0,z1,z2,y):

delta2 = z2 - y

Delta2 = np.matmul(z1.T,delta2)

delta1 = (delta2.dot(w2[1:,:].T))*sigmoid_deriv(a1)

Delta1 = np.matmul(z0.T,delta1)

return delta2,Delta1,Delta2

#first column = bais

X = np.array([[1,1,0],

[1,0,1],
[1,0,0],

[1,1,1]])

#Output

y = np.array([[1],[1],[0],[0]])

#initialize weights

w1 = np.random.randn(3,5)

w2 = np.random.randn(6,1)

#initialize learning rate

lr = 0.89

costs = [] #initiate epochs

epochs = 15000

m = len(X)

#start training

for i in range(epochs):

#forward

a1,z1,a2,z2 = forward(X,w1,w2)

#backprop

delta2,Delta1,Delta2 = backprop(a2,X,z1,z2,y)

w1 -= lr*(1/m)*Delta1

w2 -= lr*(1/m)*Delta2
```

```
# add costs to list for plotting

c = np.mean(np.abs(delta2))

costs.append(c)

if i % 1000 == 0:

print(f"iteration: {i}. Error: {c}")

#training complete

print("Training complete")

#Make prediction

z3 = forward(X,w1,w2,True)

print("Precentages: ")

print(z3)

print("Predictions: ")

print(np.round(z3))

plt.plot(costs)

plt.show()
```

Output:

Explanation to the above code: We can see here the error rate is decreasing gradually it started with 0.5 in the 1st iteration and it gradually reduced to 0.00 till it came to the 15000 iterations. Since we have already defined the number of iterations to 15000 it went up to that.

Graph Explanation

We can see the below graph depicting the fall in the error rate

Advantages and Disadvantages

Below we discuss the advantages and disadvantages for the same:

Advantages

- Single Layer Perceptron is quite easy to set up and train.
- The neural network model can be explicitly linked to statistical models which means the model can be used to share covariance Gaussian density function.
- The SLP outputs a function which is a sigmoid and that sigmoid function can easily be linked to posterior probabilities.
- We can interpret and input the output as well since the outputs are the weighted sum of inputs.

Disadvantages

- This neural network can represent only a limited set of functions.
- The decision boundaries that are the threshold boundaries are only allowed to be hyperplanes.
- This model only works for the linearly separable data.

Conclusion – Single Layer Perceptron

In this article, we have seen what exactly the Single Layer Perceptron is and the working of it. Through the graphical format as well as through an image classification code. We have also checked out the advantages and disadvantages of this perception.

Introduction to Inductive Learning in Artificial Intelligence

Machine learning is one of the most important subfields of artificial intelligence. It has been viewed as a viable way of avoiding the knowledge bottleneck problem in developing knowledge-based systems.

Inductive Learning, also known as Concept Learning, is how AI systems attempt to use a generalized rule to carry out observations.

To generate a set of classification rules, Inductive Learning Algorithms (APIs) are used. These generated rules are in the “If this then that” format.

These rules determine the state of an entity at each iteration step in Learning and how the Learning can be effectively changed by adding more rules to the existing ruleset.

When the output and examples of the function are fed into the AI system, inductive Learning attempts to learn the function for new data.

The Fundamental Concept of Inductive Learning

There are two methods for obtaining knowledge in the real world: first, from domain experts, and second, from machine learning.

Domain experts are not very useful or reliable for large amounts of data. As a result, for this project, we are adopting a machine learning approach.

The other method, using machine learning, replicates the logic of ‘experts’ in algorithms, but this work may be very complex, time-consuming, and expensive.

As a result, an option is the inductive algorithms, which generate a strategy for performing a task without requiring instruction at each step.

According to Jason Brownlee in his article “Basic Concepts in Machine Learning,” an excellent method to understand how Inductive Learning works is, for example, if we are given input samples (x) and output samples (f(x)) from the perspective of inductive Learning, and the problem is to estimate the function (f).

It is necessary then to generalize from the samples and the mapping so that it can be used to estimate the output for new samples in the future.

In practice, estimating the function is almost always too complicated, so we seek excellent approximations.

Some practical examples of induction are:

Credit risk assessment.

- The x is the property of the customer.
- The f(x) is credit approved or not.

Disease diagnosis.

- The x is the characteristics of a given patient.
- The f(x) is the patient’s disease.

Face recognition.

- The x are bitmaps of the faces we want to recognize.
- The f(x) is a name assigned to that face.

Automatic steering (autonomous driving).

- The x is bitmap images from a camera in front of the car.
- The f(x) is the degree to which the steering wheel should be turned.

Application

There are some situations in which inductive Learning is not a good idea. It is critical to understand when and when not to use supervised machine learning.

Inductive Learning may be helpful in the following four situations:

- **Problems in which no human expertise is available**. People cannot write a program to solve a problem if they do not know the answer. These are areas ripe for exploration.
- **Humans can complete the task, but no one knows how to do it.** There are situations in which humans can do things that computers cannot or do not do well. Riding a bike or driving a car are two examples.
- **Problems where the desired function is frequently changing**. Humans could describe it and write a program to solve it, but the problem changes too frequently. It is not economical. The stock market is one example.
- **Problems where each user requires a unique function.** Writing a custom program for each user is not cost-effective. Consider Netflix or Amazon recommendations for movies or books.

During the last years, there has been an increase in the amount of research on inductive Learning and its applications to various domains.

Concept learning is the most common application of inductive Learning.

The goal of concept learning is to find symbolic descriptions expressed in high-level terms that people can understand.

The following is a definition of concept learning:

- A set of (positive and negative) examples of a concept is provided and possibly some background knowledge.
- A general description (hypothesis) of the concept describes all of the positive examples but none of the negative ones.

MARBLE, a knowledge-based artificial intelligence (AI) system, has been developed to assess the riskiness of business loan applicants.

The paper "**Applying inductive learning to enhance knowledge-based expert systems**" describes the use of inductive Learning in MARBLE, a knowledge-based expert system developed to aid in business loan evaluation. MARBLE is unique in that it can learn.

Thus, the AI system employs inference rules to simulate a lending officer's thought process when evaluating a loan request, generating loan-granting decision rules based on historical and proforma financial data.

This paper presents a learning method for inducing decision rules from training examples.

In the paper “**Inductive learning for risk classification**,” the authors discuss the application of inductive Learning to credit risk analysis, a similar domain application.

The authors address three risk classification issues:

- Business loan evaluation
- Bond-rating
- Prediction of Bankruptcy

It is discussed how to use the previously mentioned Marble system, a knowledge-based decision-support system that employs approximately 80 decision rules to evaluate commercial loans.

The paper describes an aspect of Marble that uses inductive Learning to classify financial risks, and it discusses the technique’s effectiveness.

Another example we can find in the paper “**On the Application of Inductive Machine Learning Tools to Geographical Analysis**” discusses the role of inductive machine learning in geographical analysis.

The presented discussion is not based on comparative results or mathematical descriptions, but rather is based on how the various inductive learning approaches differ operationally, describing:

- How the feature space is partitioned or clustered.
- To find suitable solutions, search mechanisms are used.
- The various biases imposed by each technique.

When considering complex geographic feature spaces, the consequences of these issues are then detailed.

The overall goal is to provide the basis for developing reliable inductive analysis methods rather than relying on piecemeal or haphazard experimentation with the various operational criteria that inductive learning tools require.

Inductive Learning has also been used in education. For example, because global education allows students to obtain education from multiple education providers through various study exchange programs, it necessitates comparing available study courses in foreign institutions to courses on the institution's curriculum that issue the degree.

This issue necessitates a manual course comparison, which can be time-consuming and necessitates highly skilled experts to avoid the comparison, resulting in a superficial intuitive judgment and, as a result, course incompatibility issues.

The study, "**Interactive Inductive Learning: Application in Domain of Education.**" presents a technique for using Interactive inductive Learning supported by enterprise modeling as a way to support mechanisms that can help save time and effort in study course comparison.

Because inductive learning algorithms are domain-independent, they can be used in any classification or pattern recognition task.

Several applications of the rules family of induction algorithms to visual inspection are presented in the paper entitled "**Applications of Inductive Learning to Automated Visual Inspection.**"

The primary value of using Inductive Learning for visual inspection, according to authors Mehmet Sabih Aksoy, Orhan Torkul, Abdullah Almudimigh, and Ismail H. Cedimoglu:

- The systems have no orientation issues, which are critical in digital image processing.
- Because rules represent the pattern, it is not necessary to store it in graphics form in memory. Memory space is saved as a result of this.
- Because the number of conditions in each rule and the total number of rules is insignificant, the decision can be reached quickly.
- Because these systems are not very complex, it is simpler to develop software and design hardware for them.

Conclusion

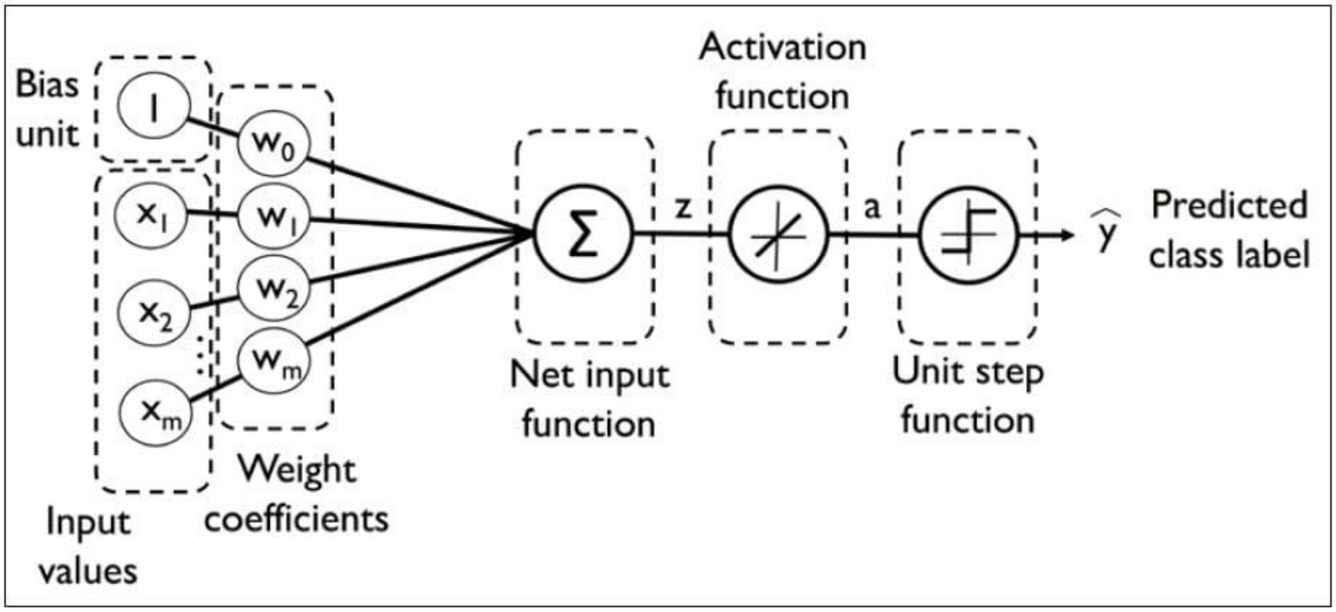
In cases where traditional statistical approaches fail due to scalability and flexibility issues, Inductive Machine Learning tools, like neural networks and decision trees, provide alternative methods for classification, clustering, and pattern recognition that, in theory, can extend to the complex or "deep" data sets that pervade geography.

Inductive learning algorithms are domain-agnostic, and they can be applied in any task that requires classification or pattern recognition.

There has been an increase in the amount of research on inductive Learning and its applications to various domains in recent years. Several good algorithms have emerged as a result of this research. It is worth keeping an eye on.

Understanding Single-layer ANN

Perceptron rule and Adaline rule were used to train a single-layer neural network.



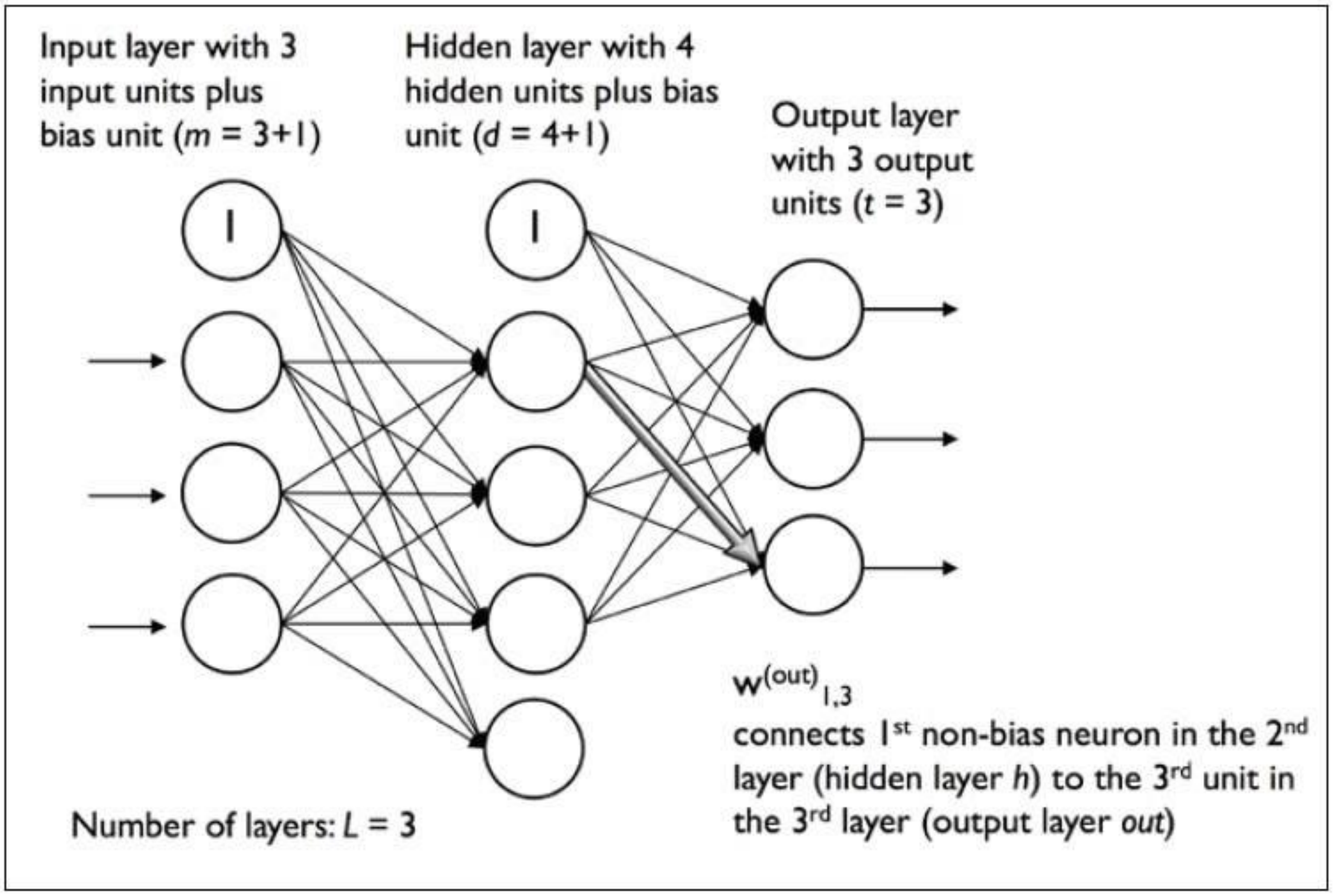
Weights are updated based on a unit function in perceptron rule or on a linear function in Adaline Rule.

History of Multi-layer ANN

Deep Learning deals with training multi-layer artificial neural networks, also called Deep Neural Networks. After Rosenblatt perceptron was developed in the 1950s, there was a lack of interest in neural networks until 1986, when Dr.Hinton and his colleagues developed the backpropagation algorithm to train a multilayer neural network. Today it is a hot topic with many leading firms like Google, Facebook, and Microsoft which invest heavily in applications using deep neural networks.

Multi-layer ANN

A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).


Input layer with 3 input units plus bias unit (m = 3+1)
Hidden layer with 4 hidden units plus bias unit (d = 4+1)
Output layer with 3 output units (t = 3)
1
1
$w^{(out)}_{1,3}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number of layers: L = 3

It has 3 layers including one hidden layer. If it has more than 1 hidden layer, it is called a deep ANN. An MLP is a typical example of a feedforward artificial neural network. In this figure, the ith activation unit in the lth layer is denoted as ai(l).

The number of layers and the number of neurons are referred to as hyperparameters of a neural network, and these need tuning. Cross-validation techniques must be used to find ideal values for these.

The weight adjustment training is done via backpropagation. Deeper neural networks are better at processing data. However, deeper layers can lead to vanishing gradient problems. Special algorithms are required to solve this issue.

Notations

In the representation below:

$$a^{(in)} = \begin{bmatrix} a_0^{(in)} \\ a_1^{(in)} \\ \vdots \\ a_m^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(in)} \\ \vdots \\ x_m^{(in)} \end{bmatrix}$$

- ai(in) refers to the ith value in the input layer
- ai(h) refers to the ith unit in the hidden layer
- ai(out) refers to the ith unit in the output layer
- ao(in) is simply the bias unit and is equal to 1; it will have the corresponding weight w0
- The weight coefficient from layer l to layer l+1 is represented by wk,j(l)

A simplified view of the multilayer is presented here. This image shows a fully connected three-layer neural network with 3 input neurons and 3 output neurons. A bias term is added to the input vector.

Forward Propagation

In the following topics, let us look at the forward propagation in detail.

MLP Learning Procedure

The MLP learning procedure is as follows:

- Starting with the input layer, propagate data forward to the output layer. This step is the forward propagation.
- Based on the output, calculate the error (the difference between the predicted and known outcome). The error needs to be minimized.
- Backpropagate the error. Find its derivative with respect to each weight in the network, and update the model.

Repeat the three steps given above over multiple epochs to learn ideal weights.

Finally, the output is taken via a threshold function to obtain the predicted class labels.

Forward Propagation in MLP

In the first step, calculate the activation unit al(h) of the hidden layer.

$$z_1^{(h)} = a_0^{(in)} w_{0,1}^{(h)} + a_1^{(in)} w_{1.1}^{(h)} + \cdots + a_m^{(in)} w_{m,1}^{(h)}$$

$$a_1^{(h)} = \phi\left(z_1^{(h)}\right)$$

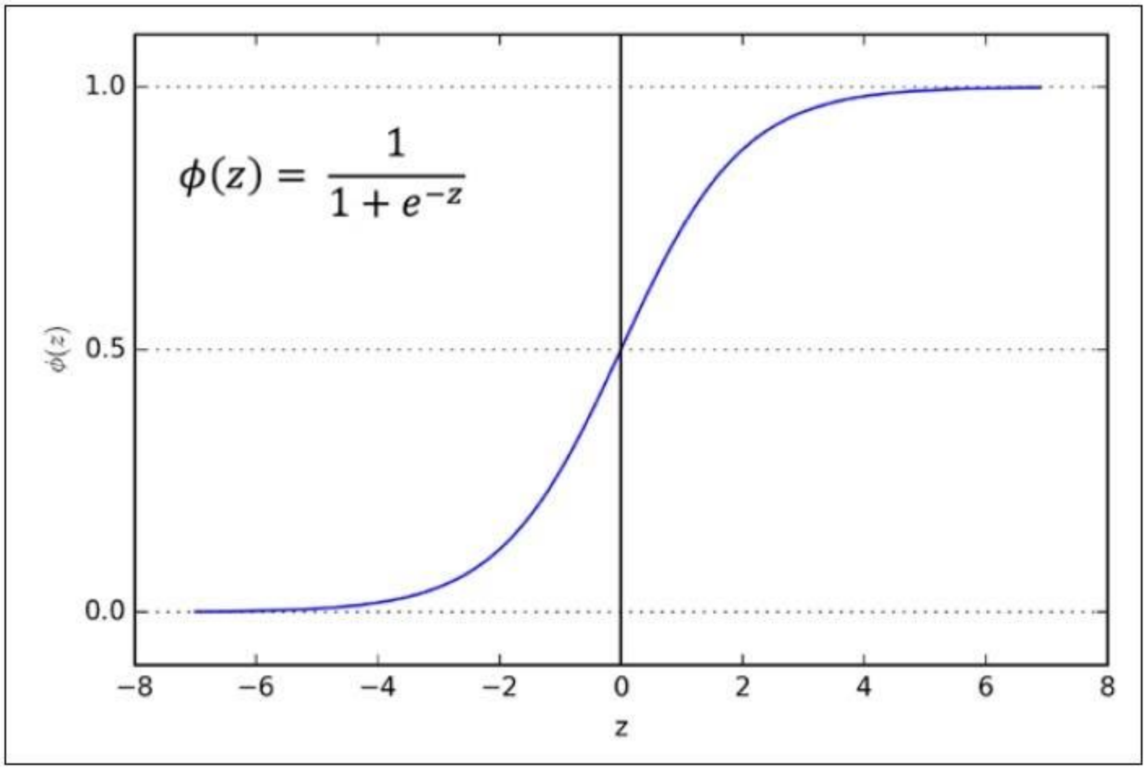
Activation unit is the result of applying an activation function φ to the z value. It must be differentiable to be able to learn weights using gradient descent. The activation function φ is often the sigmoid (logistic) function.

$$\phi(z) = \frac{1}{1+e^{-z}}$$

It allows nonlinearity needed to solve complex problems like image processing.

Sigmoid Curve

The sigmoid curve is an S-shaped curve.



Activation of Hidden Layer

The activation of the hidden layer is represented as:

z(h) = a(in) W(h)

a(h) =a(in)

For the output layer:

Z(out) = A(h) W(out)

A(out) =A(h)

****************THE END**********************