Artificial Intelligence Tutorial

What is Artificial Intelligence?

In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day.

Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines. The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

AI is one of the fascinating and universal fields of Computer science which has a great scope in future. AI holds a tendency to cause a machine to work as a human.

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

So, we can define AI as:

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

Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems

With Artificial Intelligence you do not need to preprogram a machine to do some work, despite that you can create a machine with programmed algorithms which can work with own intelligence, and that is the awesomeness of AI.

It is believed that AI is not a new technology, and some people says that as per Greek myth, there were Mechanical men in early days which can work and behave like humans.

Why Artificial Intelligence?

Before Learning about Artificial Intelligence, we should know that what is the importance of AI and why should we learn it. Following are some main reasons to learn about AI:

- With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
- AI opens a path for other new technologies, new devices, and new Opportunities.

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.

What is Artificial Intelligence?

In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day.

Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines.The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

AI is one of the fascinating and universal fields of Computer science which has a great scope in future. AI holds a tendency to cause a machine to work as a human.

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

So, we can define AI as:

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

Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems

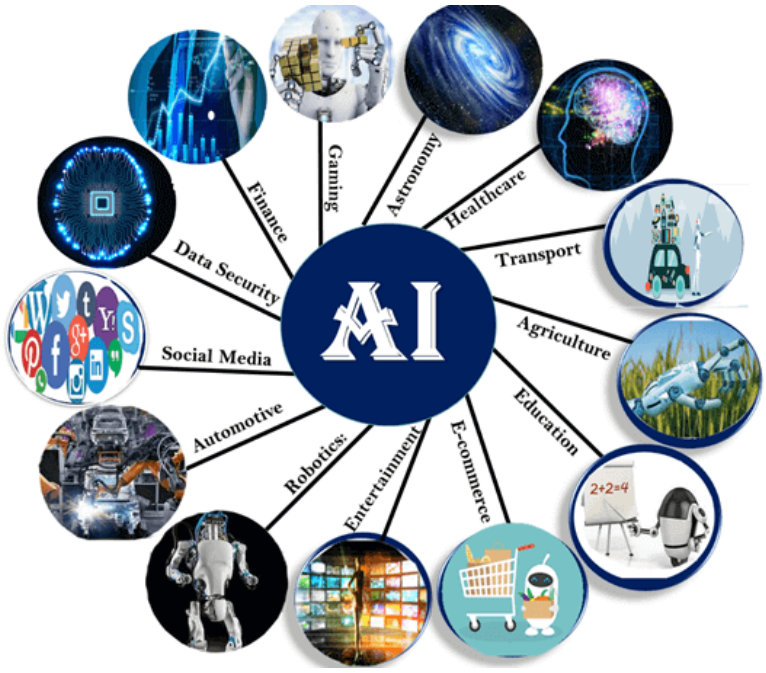
With Artificial Intelligence you do not need to preprogram a machine to do some work, despite that you can create a machine with programmed algorithms which can work with own intelligence, and that is the awesomeness of AI.

It is believed that AI is not a new technology, and some **people says that as per Greek myth, there were Mechanical men in early days which can work and behave like humans**.

Application of AI

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

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



1. AI in Astronomy

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

2. AI in Healthcare

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

3. AI in Gaming

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

4. AI in Finance

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

5. AI in Data Security

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

6. AI in Social Media

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

7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the

customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

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

11. AI in Agriculture

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

12. AI in E-commerce

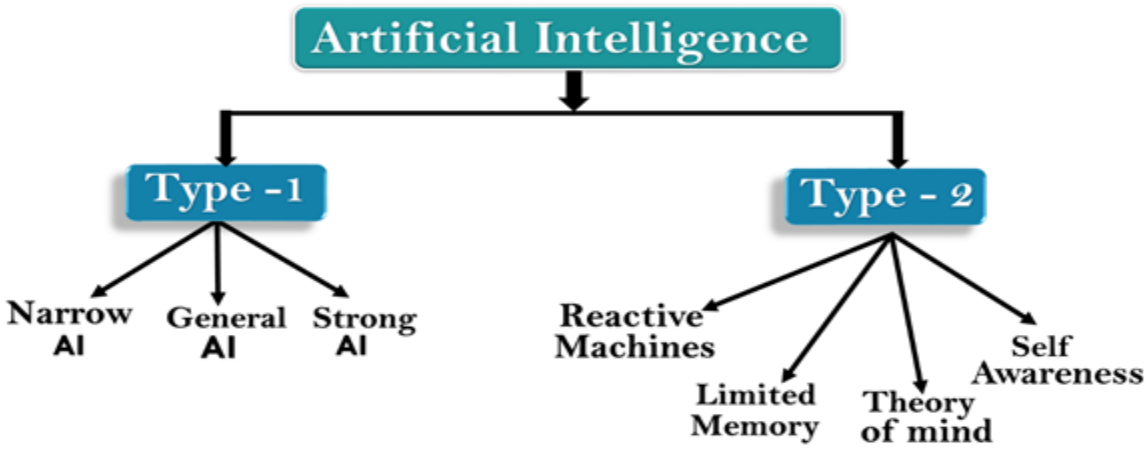
- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

Types of Artificial Intelligence:

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explain the types of AI.



AI type-1: Based on Capabilities

1. Weak AI or Narrow AI:

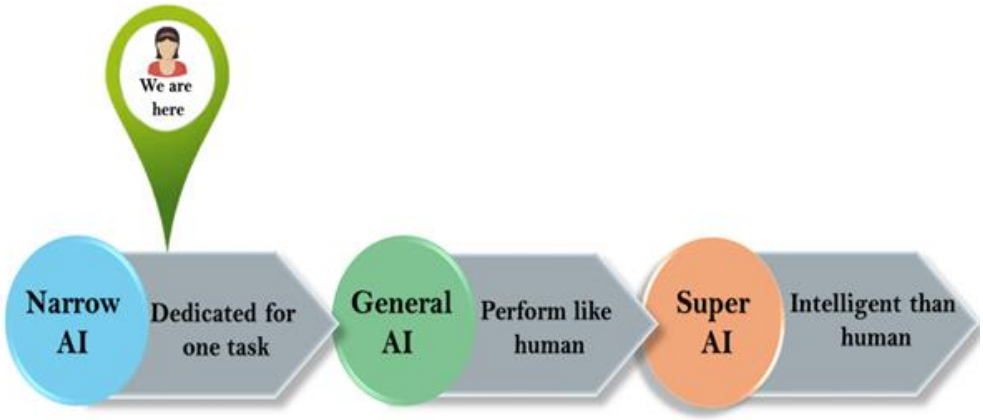
- Narrow AI is a type of AI which is able to perform a dedicated task with intelligence.The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
- Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
- Apple Siriis a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
- IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
- Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

2. General AI:

- General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research, and it will take lots of efforts and time to develop such systems.

3. Super AI:

- Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
- Some key characteristics of strong AI include capability include the ability to think, to reason,solve the puzzle, make judgments, plan, learn, and communicate by its own.
- Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.



Artificial Intelligence type-2: Based on functionality

1. Reactive Machines

- Purely reactive machines are the most basic types of Artificial Intelligence.
- Such AI systems do not store memories or past experiences for future actions.
- These machines only focus on current scenarios and react on it as per possible best action.
- IBM's Deep Blue system is an example of reactive machines.
- Google's AlphaGo is also an example of reactive machines.

2. Limited Memory

- Limited memory machines can store past experiences or some data for a short period of time.
- These machines can use stored data for a limited time period only.
- Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

3. Theory of Mind

- Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.
- This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

4. Self-Awareness

- Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
- These machines will be smarter than human mind.
- Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

Search Algorithms in Artificial Intelligence

Search algorithms are one of the most important areas of Artificial Intelligence. This topic will explain all about the search algorithms in AI.

Problem-solving agents:

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

Search Algorithm Terminologies:

- **Search:**Searchingis a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
    1. **Search Space:** Search space represents a set of possible solutions, which a system may have.
    2. **Start State:** It is a state from where agent begins **the search**.
    3. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.

Properties of Search Algorithms:

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

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

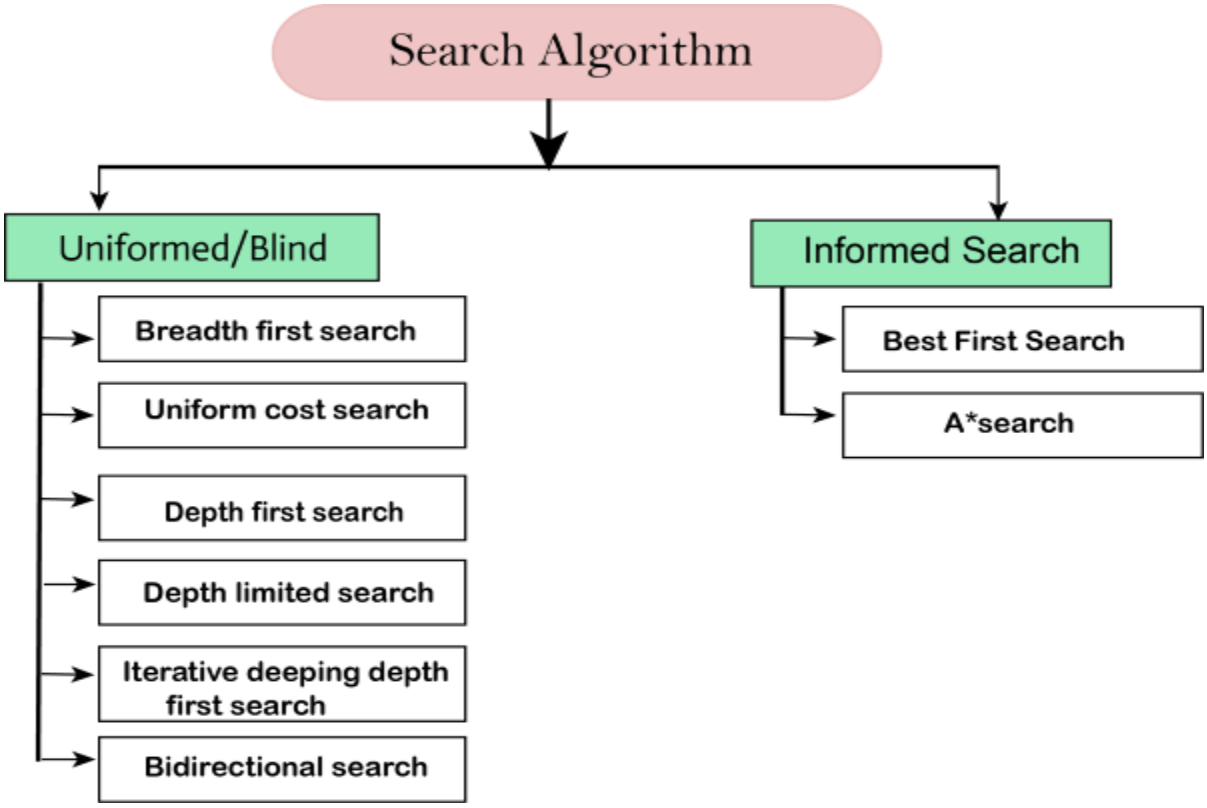
Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

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

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

Types of search algorithms

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



Un-informed/Blind Search:

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

It can be divided into five main types:

- Breadth-first search
- Uniform cost search
- Depth-first search
- Iterative deepening depth-first search
- Bidirectional Search

Informed Search

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

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

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

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

1. Greedy Search
2. A* Search

Un-informed Search Algorithms

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

Following are the various types of uninformed search algorithms:

1. **Breadth-first Search**
2. **Depth-first Search**
3. **Depth-limited Search**
4. **Iterative deepening depth-first search**
5. **Uniform cost search**
6. **Bidirectional Search**

1. Breadth-first Search:

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

Advantages:

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

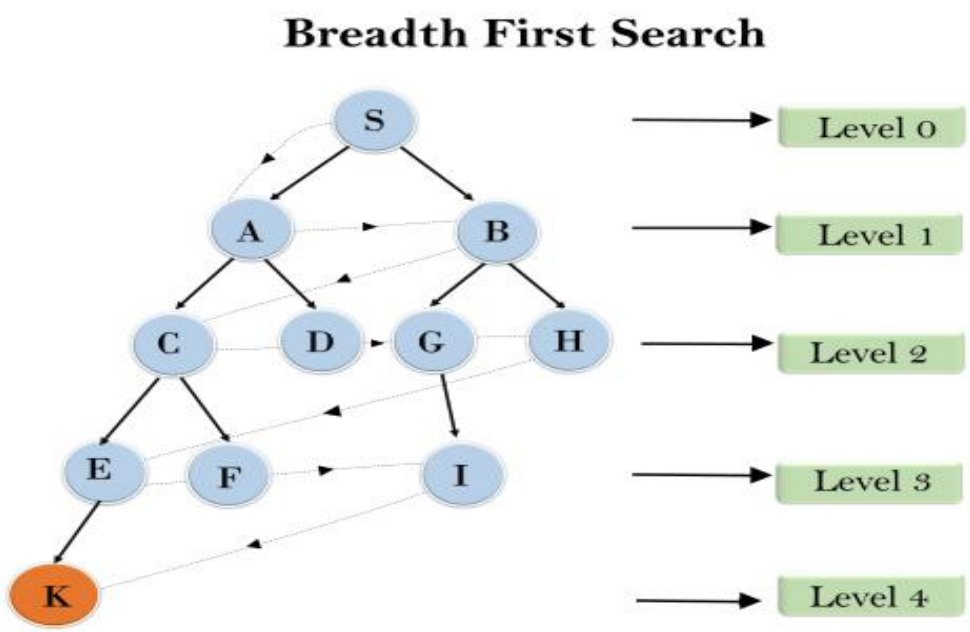
Disadvantages:

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

Example:

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

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



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

$$\mathbf{T(b) = b^0+b^1+b^2+b^3+.....+ b^d= O(b^d)}$$

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

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

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

2. Depth-first Search

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

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

Advantage:

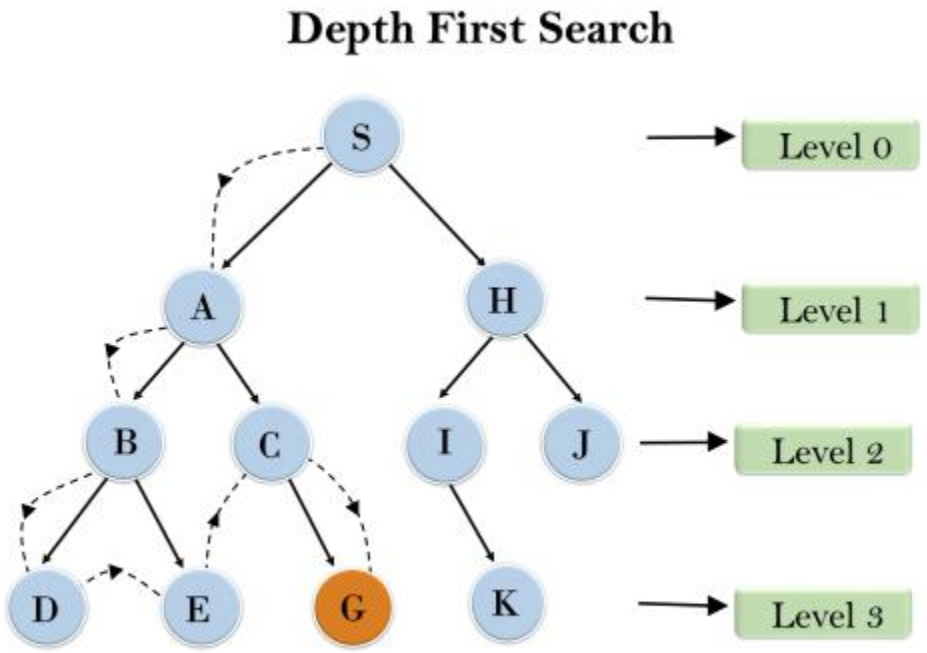
- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantage:

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

Example:

In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:
Root node--->Left node ----> right node.
It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal



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

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

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

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

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

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

3. Depth-Limited Search Algorithm:

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

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

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

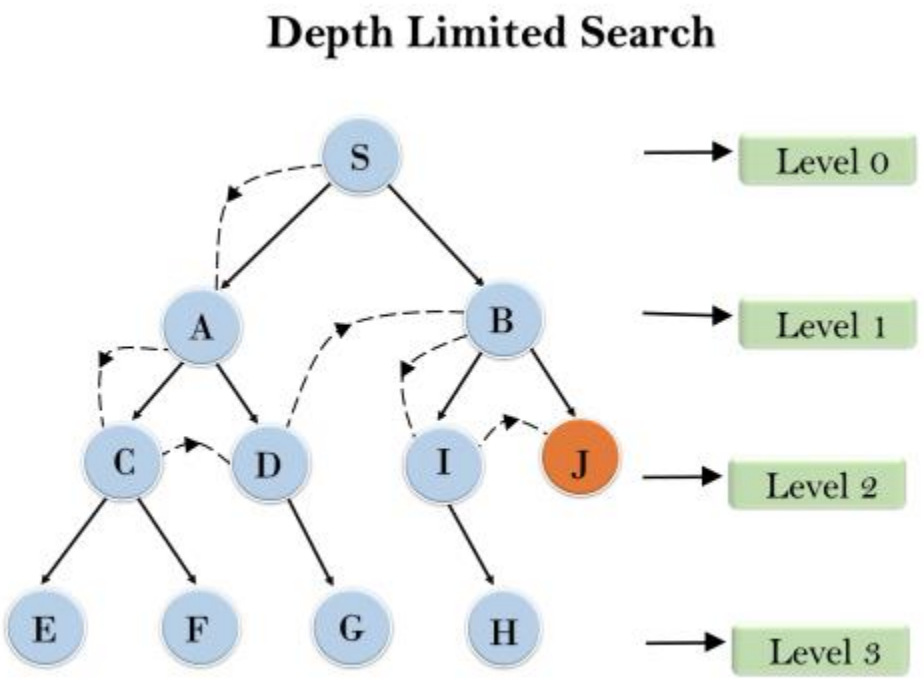
Advantages:

Depth-limited search is Memory efficient.

Disadvantages:

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

Example:



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

Time Complexity: Time complexity of DLS algorithm is **$O(b^{l})$, l=level**.

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

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

4. Uniform-cost Search Algorithm:

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

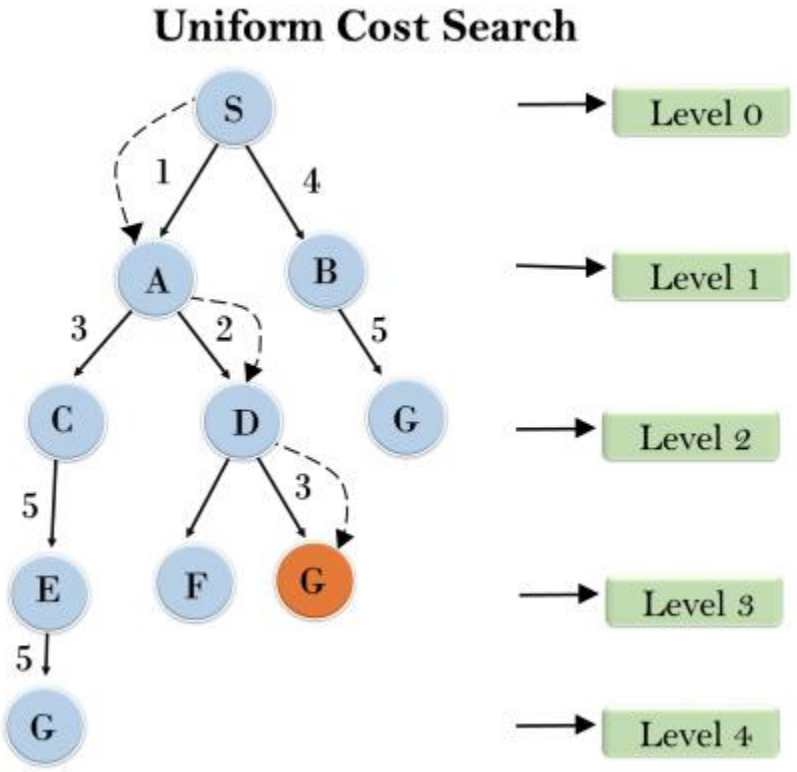
Advantages:

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

Disadvantages:

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

Example:



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

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

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

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

.

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

5. Iterative deepeningdepth-first Search:

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

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

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

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

Advantages:

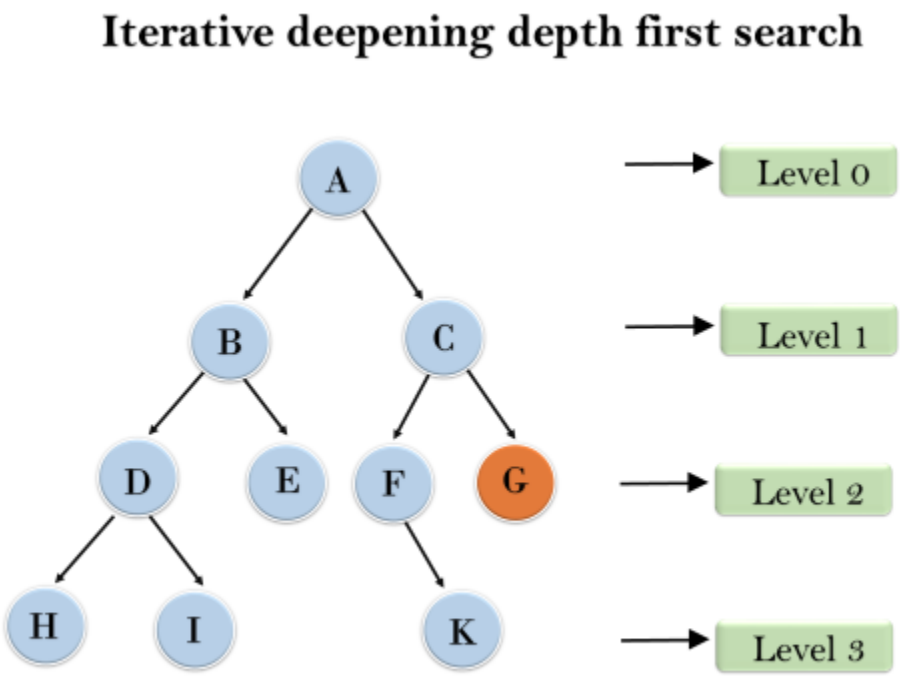
- Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

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

Example:

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



1'st Iteration-----> A
2'nd Iteration----> A, B, C

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

Completeness:

This algorithm is complete is ifthe branching factor is finite.

Time Complexity:

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

Space Complexity:

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

Optimal:

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

6. Bidirectional Search Algorithm:

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

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

Advantages:

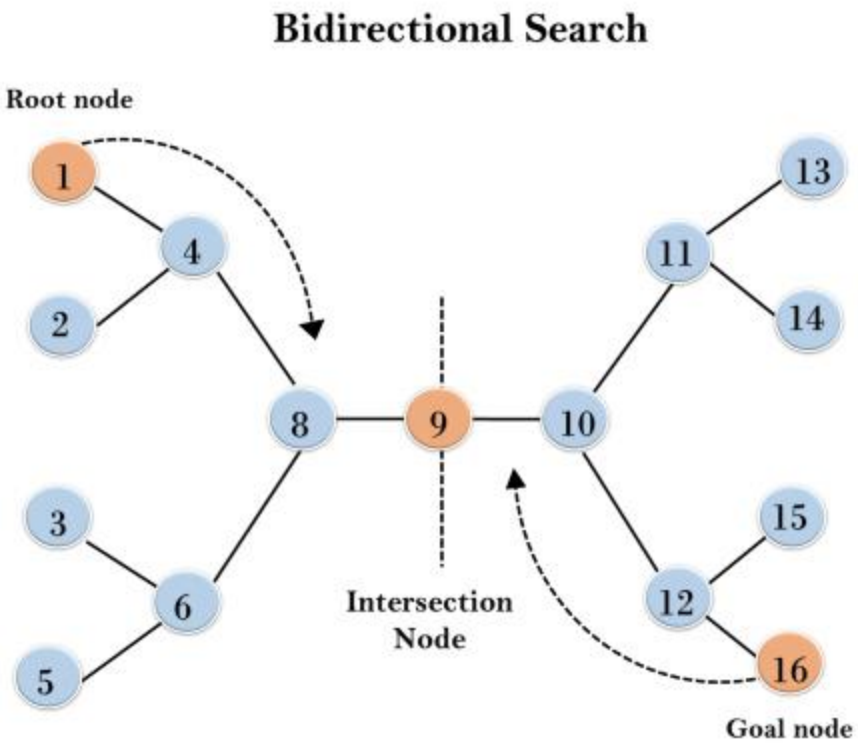
- Bidirectional search is fast.
- Bidirectional search requires less memory

Disadvantages:

- Implementation of the bidirectional search tree is difficult.
- **In bidirectional search, one should know the goal state in advance.**

Example:

In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
The algorithm terminates at node 9 where two searches meet.



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

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

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

Optimal: Bidirectional search is Optimal.

Parameters	Informed Search	Uninformed Search
Known as	It is also known as Heuristic Search.	It is also known as Blind Search.
Using Knowledge	It uses knowledge for the searching process.	It doesn't use knowledge for the searching process.
Performance	It finds a solution more quickly.	It finds solution slow as compared to an informed search.
Completion	It may or may not be complete.	It is always complete.
Cost Factor	Cost is low.	Cost is high.
Time	It consumes less time because of quick searching.	It consumes moderate time because of slow searching.
Direction	There is a direction given about the solution.	No suggestion is given regarding the solution in it.
Implementation	It is less lengthy while implemented.	It is more lengthy while implemented.
Efficiency	It is more efficient as efficiency takes into account cost and performance. The incurred cost is less and speed of finding solutions is	It is comparatively less efficient as incurred cost is more and the speed of finding the Breadth-

	quick.	Firstsolution is slow.
Computational requirements	Computational requirements are lessened.	Comparatively higher computational requirements.
Size of search problems	Having a wide scope in terms of handling large search problems.	Solving a massive search task is challenging.
Examples of Algorithms	• **Greedy Search** • **A* Search** • **AO* Search** • **Hill Climbing Algorithm**	• **Depth First Search (DFS)** • **Breadth First Search (BFS)** • **Branch and Bound**

Informed Search Algorithms

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

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

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

Admissibility of the heuristic function is given as:

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

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

Pure Heuristic Search:

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

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

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

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

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

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

1. **f(n)= h(n).**

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

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

Best first search algorithm:

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

Advantages:

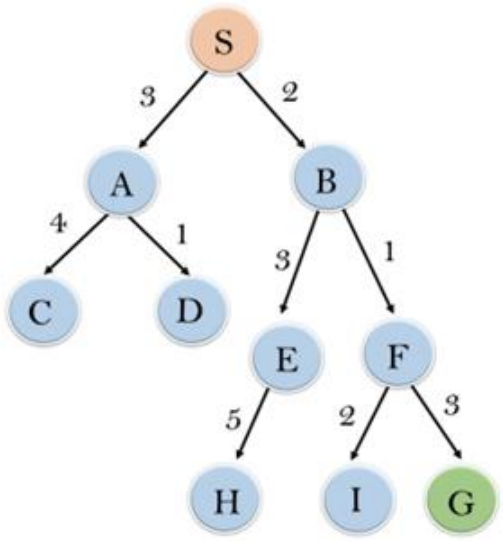
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

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

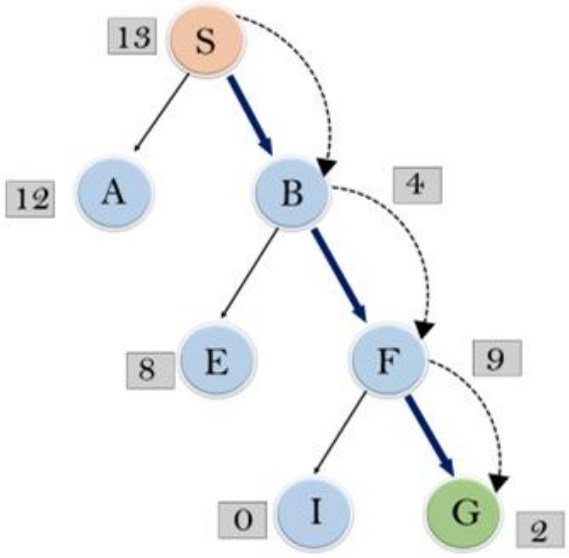
Example:

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



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

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



Expand the nodes of S and put in the CLOSED list

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

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

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

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

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

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

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

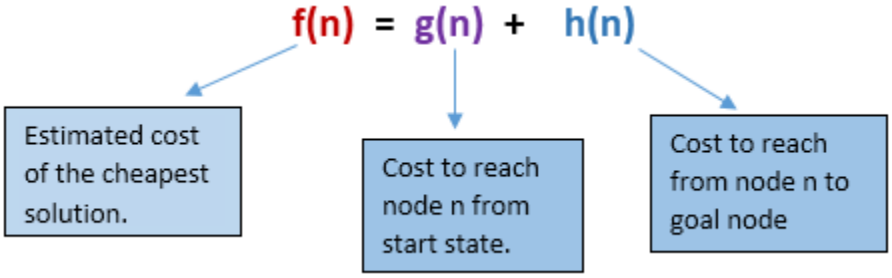
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A* Search Algorithm:

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

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



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

Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

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

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

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

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

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

Advantages:

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

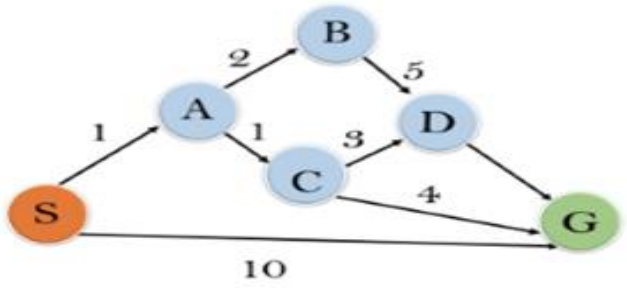
Disadvantages:

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

Example:

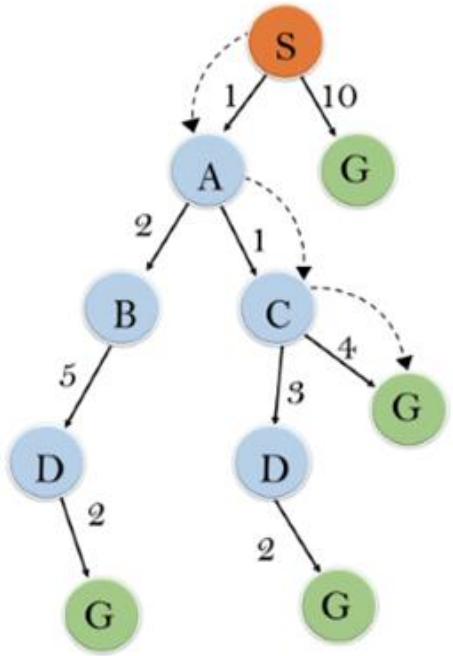
In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.

Here we will use OPEN and CLOSED list.



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

Solution:



Initialization: {(S, 5)}
Iteration1: {(S--> A, 4), (S-->G, 10)}
Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}
Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}
Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

Points to remember:

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

Complete: A* algorithm is complete as long as:

- Branching factor is finite.
- Cost at every action is fixed.

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

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

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

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

Hill Climbing Algorithm in Artificial Intelligence

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

Features of Hill Climbing:

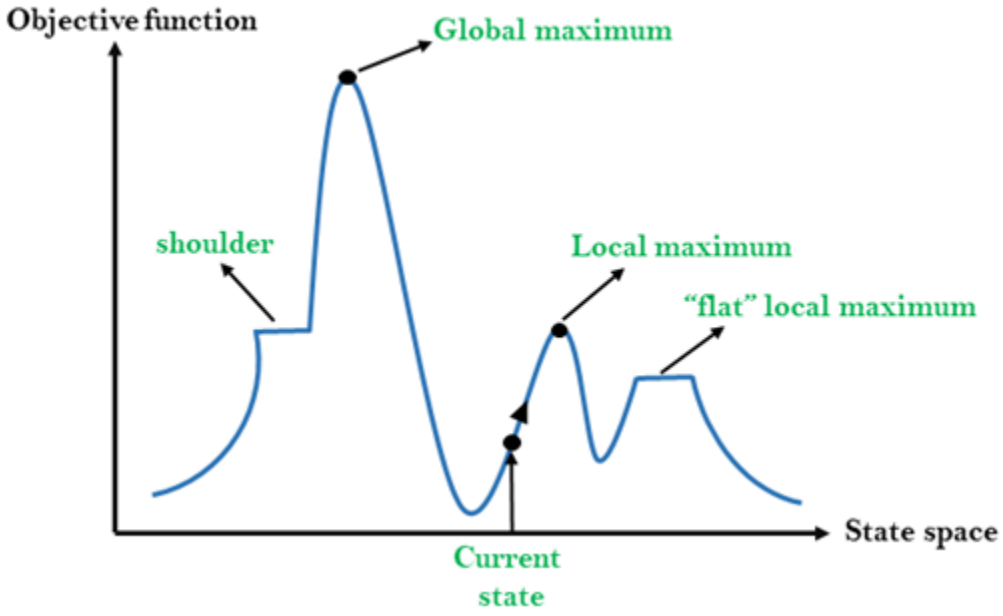
Following are some main features of Hill Climbing Algorithm:

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

State-space Diagram for Hill Climbing:

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

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



Different regions in the state space landscape:

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

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

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

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

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

Types of Hill Climbing Algorithm:

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

1. Simple Hill Climbing:

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

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

Algorithm for Simple Hill Climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
    1. If it is goal state, then return success and quit.
    2. Else if it is better than the current state then assign new state as a current state.
    3. Else if not better than the current state, then return to step2.
- **Step 5:** Exit.

2. Steepest-Ascent hill climbing:

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

Algorithm for Steepest-Ascent hill climbing:

- **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
- **Step 2:** Loop until a solution is found or the current state does not change.
    1. Let SUCC be a state such that any successor of the current state will be better than it.
    2. For each operator that applies to the current state:
        I. Apply the new operator and generate a new state.
        II. Evaluate the new state.
        III. If it is goal state, then return it and quit, else compare it to the SUCC.
        IV. If it is better than SUCC, then set new state as SUCC.
        V. If the SUCC is better than the current state, then set current state to SUCC.
- **Step 5:** Exit.

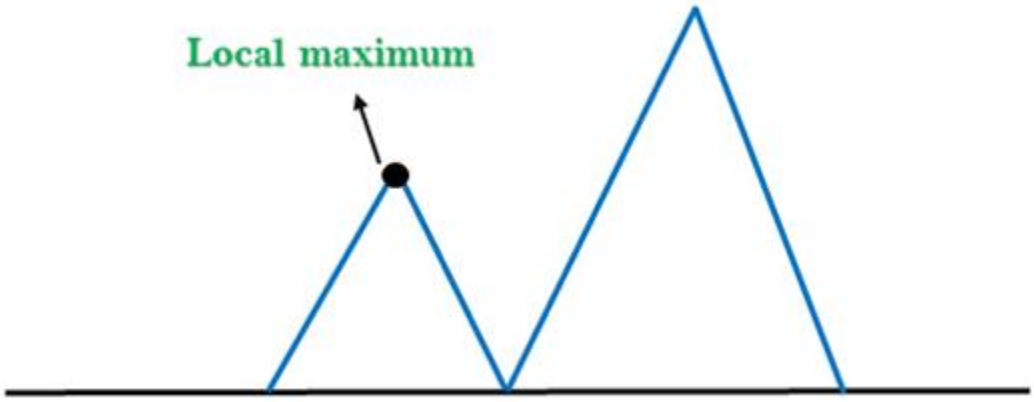
3. Stochastic hill climbing:

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

Problems in Hill Climbing Algorithm:

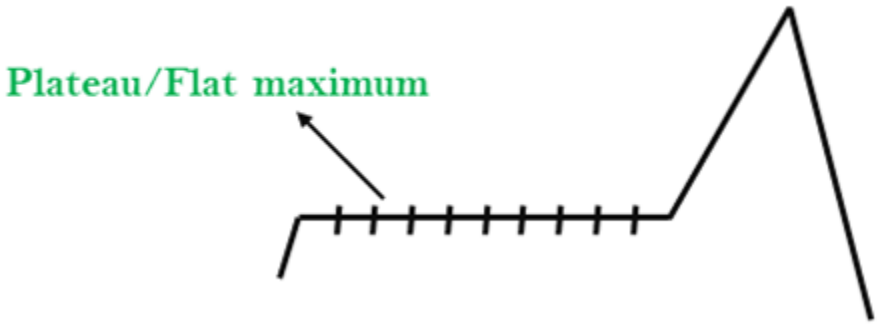
1. Local Maximum: A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

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



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

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



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

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

Simulated Annealing:

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

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

Next TopicMeans-Ends Analysis in AI

What is Prolog

- Prolog stands for programming in logic. In the logic programming paradigm, prolog language is most widely available. Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution. A logical relationship describes the relationships which hold for the given application.
- To obtain the solution, the user asks a question rather than running a program. When a user asks a question, then to determine the answer, the run time system searches through the database of facts and rules.
- The first Prolog was 'Marseille Prolog', which is based on work by Colmerauer. The major example of fourth-generation programming language was prolog. It supports the declarative programming paradigm.
- In 1981, a Japanese computer Project of 5^{th} generation was announced. After that, it was adopted Prolog as a development language. In this tutorial, the program was written in the 'Standard' Edinburgh Prolog. Prologs of PrologII family are the other kind of prologs which are descendants of Marseille Prolog.
- Prolog features are 'Logical variable', which means that they behave like uniform data structure, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variable, and input and out are interchangeable.
- To deduce the answer, there will be more than one way. In such case, the run time system will be asked to find another solution. To generate another solution, use the backtracking strategy. Prolog is a weakly typed language with static scope rules and dynamic type checking.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.
- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.
- In artificial intelligence applications, prolog is used. The artificial intelligence applications can be automated reasoning systems, natural language interfaces, and expert systems. The expert system consists of an interface engine and a database of facts. The prolog's run time system provides the service of an interface engine.
- A basic logic programming environment has no literal values. An identifier with upper case letters and other identifiers denote variables. Identifiers that start with lower-case letters denote data values. The basic Prolog elements are typeless. The most implementations of prolog have been enhanced to include integer value, characters, and operations. The Mechanism of prolog describes the tuples and lists.

- Functional programming language and prolog have some similarities like Hugs. A logic program is used to consist of relation definition. A functional programming language is used to consist of a sequence of function definitions. Both the logical programming and functional programming rely heavily on recursive definitions.

Applications of Prolog

The applications of prolog are as follows:

- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

Prerequisite

Before learning Prolog,

Audience

Our Prolog tutorial is designed to help beginners and professionals.

Problems

We assure that you will not find any problem in this Prolog Tutorial. But if there is any mistake, please post the problem in a contact form.

Starting Prolog

Prolog system is straightforward. From one person to other person, the precise details of Prolog will vary. Prolog will produce a number of lines of headings in the starting, which is followed by a line. It contains just

?-

The above symbol shows the system prompt. The prompt is used to show that the Prolog system is ready to specify one or more goals of sequence to the user. Using a full stop, we can terminate the sequence of goals.

For example:

?- write('Welcome to Javatpoint'),nl,write('Example of Prolog'),nl.

nl indicates 'start a new line'. When we press 'return' key, the above line will show the effect like this:

Welcome to Javatpoint

Example of Prolog

yes

?- prompt shows the sequence of goal which is entered by the user. The user will not type the prompt. Prolog system will automatically generate this prompt. It means that it is ready to receive a sequence of goals.

The above example shows a sequence of goals entered by the user like this:

write('Welcome to Javatpoint'), write('Example of Prolog'), nl(twice).

Consider the following sequence of goals:

write('Welcome to Javatpoint'),nl,write('Example of Prolog'),nl.

The above sequence of goals has to succeed in order to be succeeded.

- **write('Welcome to Javatpoint')**
  On the screen of the user, Welcome to Javatpoint has to be displayed
- **nl**
  On the screen of the user, a new line has to be output
- **write('Example of Prolog')**
  On the screen of the user, Example of Prolog has to be displayed
- **nl**
  On the screen of the user, a new line has to be output

All these goals will simply achieve by the Prolog system by outputting the line of text to the screen of the user. To show that the goals have succeeded, we will output **yes**.

The Prolog system predefined the meanings of **nl** and **write**. Write and nl are called as built-in predicates.

Halt and **statistics** are the two other built-in predicates. In almost all Prolog versions, these predicates are provided as standard.

- **?- halt.**
  The above command is used to terminate the Prolog system.
- **?- statistics.**
  This command will cause the Prolog system statistics. This statistics feature is mainly used to experienced user. In statistics, the following things will generate:

Memory Statistics	**Percent**	**Total Bytes**	**Free Bytes**
Input Space	(100%)	65536	65536
Reset Space	(100%)	65536	65536
Output Space	(100%)	65536	65536
Heap Space	(100%)	262130	261850
Total Elapsed Time	(100%)		626.871
Active Processing	(0%)		2.704
Waiting for Input	(100%)		623.977

Yes

The above output ends with Yes. Yes, specify that the goal has succeeded, as halt, statistics, and many other built-in predicates always do. When they evaluate, their values produce, which lies in the side-effects.

'**Query**' is a sequence of one or more goals. These goals are entered by the user at the prompt. In this tutorial, we are generally using the 'sequence of goals' term.

Prolog Programs

Using the built-in predicates, the sequence of goals, or specifying a goal at the system prompt would be of little value in itself. To write a Prolog program, firstly, the user has to write a program which is written in the Prolog language, load that program, and then specify a sequence of one or more goals at the prompt.

To create a program in Prolog, the simple way is to type it into the text editor and then save it as a text file like **prolog1.pl**.

The following example shows a simple program of Prolog. The program contains three components, which are known as clauses. Each clause is terminated using a full stop.

1. dog(rottweiler).
2. cat(munchkin).
3. animal(A) :- cat(A).

Using the built-in predicate '**consult**', the above program can be loaded in the Prolog system.

?-consult('prolog1.pl').

This shows that prolog1.pl file exists, and the prolog program is systemically correct, which means it has valid clauses, the goal will succeed, and to confirm that the program has been correctly read, it produces one or more lines of output. e.g.,

?-
 # 0.00 seconds to consult prolog1.pl
?-

The alternative of 'consult' is 'Load', which will exist on the menu option if the Prolog system has a graphical user interface.

When the program is loaded, the clause will be placed in a storage area, and that storage area is known as the Prolog database. In response to the system prompt, specify a sequence of goals, and it will cause Prolog to search for and use the clauses necessary to evaluate the goals.

Terminology

In the following program, three lines show the clauses.

1. dog(rottweiler).
2. cat(munchkin).
3. animal(A) :- cat(A).

Using the full stop, each clause will be terminated. Prolog programs have a sequence of clauses. Facts or rules are described by these clauses.

Example of **facts** is **dog(rottweiler)** and **cat(munchkin)**. They mean that '**rottweiler** is a dog' and '**munchkin** is a cat'.

Dog is called a predicate. Dog contains one argument. Word '**rottweiler'** enclosed in bracket( ). Rottweiler is called an atom.

The example of rule is the final line of the program.

1. animal(A) :- dog(A).

The colon(:-) character will be read as 'if'. Here A is a variable, and it represents any value. In a natural way, the rule can be read as "If A is an animal, then A is a dog".

The above clause shows that the **rottweiler** is an animal. Such deduction can also make by Prolog:

?- animal(rottweiler).
yes

To imply that **munchkin** is an animal, there is no evidence of this.

?- animal(munchkin).
no

More Terminology

Evaluating a goal term determines whether or not it is satisfied. It also means that goal evaluates to true or false.

Note that when a user enters a goal, then sometimes it can be interpreted as a command. For example,

?- halt. 'It is used to exit from the Prolog system.'

Sometimes it can be regarded as a question like,

?- animal(rottweiler). & 'Is rottweiler an animal?'

The following program shows another example about animals. It comprises eight clauses. The comment is shown by all the text between the /* and */.

```
1. /* Another Program of Animal */
2. Dog(rottweiler).
3. cat(sphynx).      dog(poodle).
4. dog(bulldog).     cat(bengal).
5. dog(dobermann).
6. cat(himalayan). cat(singapura).
7. /* This Prolog program consists of various clauses. It is always terminated using the full stop.*/
```

Predicate dog and predicate cat both have four clauses. Assume that in a text file 'animal.pl', the program has been saved, and output is generated by loading the program and at the system prompt, we are entering a sequence of goals as follows:

?- consult('animals1.pl'). **System prompt**
 # 0.01 seconds to consult animals.pl **animals.pl loaded using the consult**
?- dog(rottweiler).
yes
?- dog(boxer).
no
?- dog(A).
A = rottweiler **pauses- return key is pressed by the user**
?- dog(B).
B = rottweiler; **pauses ? user presses ;**
B = poodle; **pauses ? user presses ;**
B = bulldog; **pauses ? user presses ;**
B = dobermann **No pause ? It will go onto next line**
?- cat(A). A = sphinx; **pause ? user presses;**
A = Bengal **pauses ? user presses return**
?- listening(dog). **It will list all the clauses which define predicate dog**
/* dog/1 */
dog(rottweiler).
dog(poodle).
dog(bulldog).
dog(dobermann).
yes
?-

In this example, various new features of Prolog are introduced. The query is as follows:

?- dog(A).

It means that find the A's value, and it will be the name of the dog. The answer of Prolog is as follows:

A = rottweiler

Other possible answers of A are as follows, poodle, bulldog, dobermann. It will cause the Prolog pause, and because of this, we have to wait for the user to press the 'return' key before it outputs the system prompt ?-.

We can enter the next query as follows:

?- dog(B).

This query is the same as before. The above query means that 'find the B's value, and it will be the name of a dog'. The answer of Prolog is as follows:

B = rottweiler

Prolog will again pause. This time semicolon (;) key is pressed by the user. Now Prolog will find for an alternative value of B that satisfies the goal dog(B). It will reply as follows:

B = poodle

Prolog will again pause. The semicolon (;) key is again pressed by the user. Prolog produces a further solution as follows:

B = bulldog

Prolog will again pause. The semicolon (;) key is again pressed by the user. Prolog produces a further solution as follows:

B = dobermann

Prolog recognizes that there is no more available solution by not pausing, but the system prompt ?- by immediately going on to the output.

A new built-in predicate is introduced in this example. Specifying the goal

?- listing(dog)

In the above goal, Prolog will list all four clauses which define the predicate dog. They will define in the same order as they loaded into the database.

The use of variables in the query is shown by the following example. The sequence of goal is as follows:

?-cat(A),dog(B).

This will give us all possible combinations of a cat and a dog.

?-cat(A),dog(B).
A = sphinx,
B = rottweiler;
A = sphinx,
B = poodle;
A = sphinx,
B = bulldog;
A = sphinx,
B = dobermann;

etc.

In contrast, the sequence of goal is as follow:

?-cat(A), dog(A).

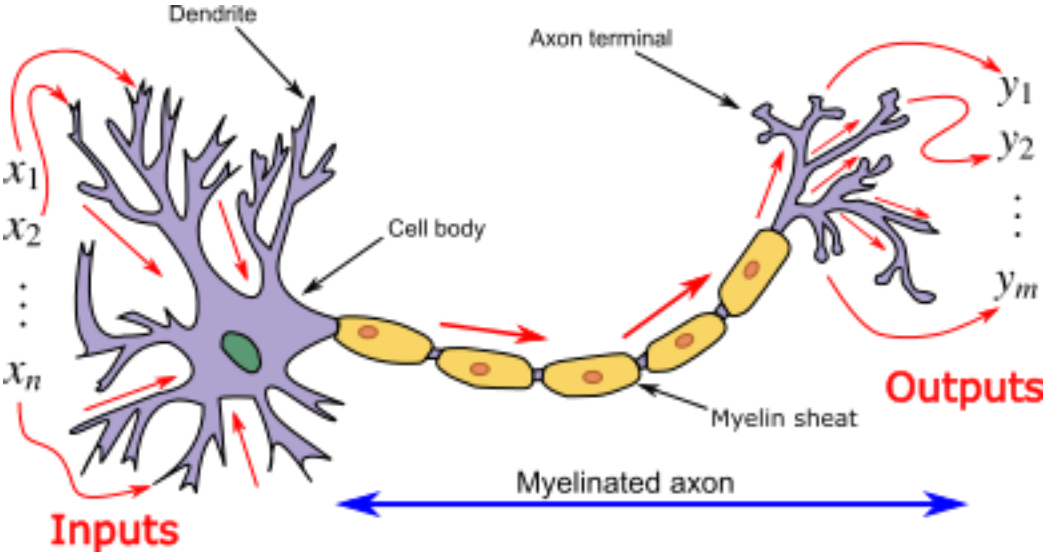
This will give all animals which are both a cat and a dog (in the database, there is no such animal). Here A is 'any value' in both cat(A) and dog(A), but both must have the same value.

?-cat(A),dog(A).
no

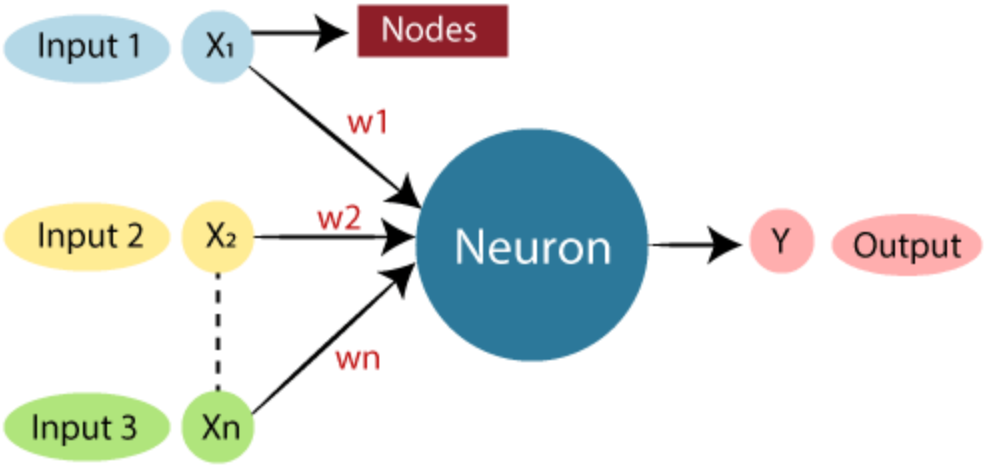
Artificial Neural Network Tutorial

What is Artificial Neural Network?

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.
The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse(Axon terminal) represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs

Cell nucleus	Nodes
Synapse(Axon terminal)	Weights
Axon	Output

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

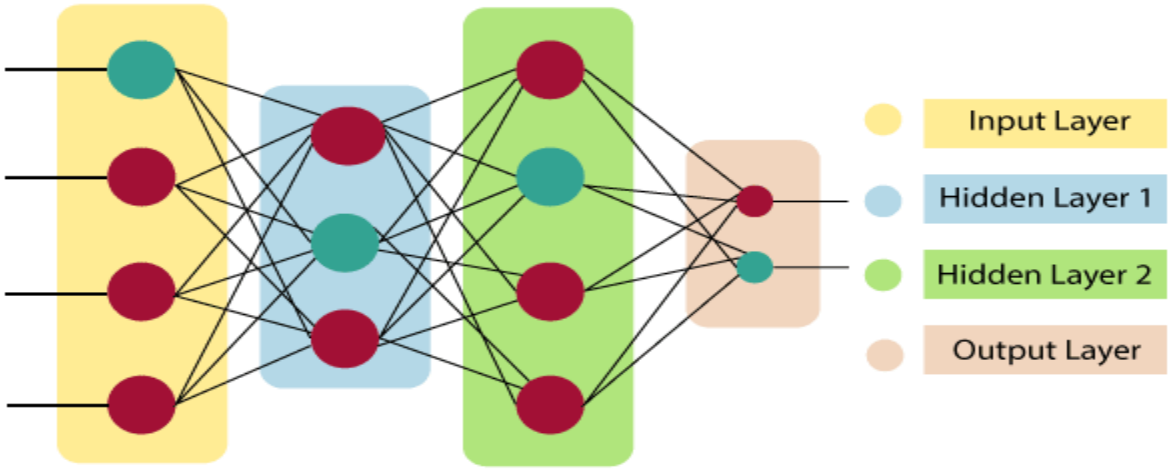
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



Input Layer:

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

The main differences between Artificial Neural Network and Biological Neural Network are as follows:

Features	Artificial Neural Network	Biological Neural Network
Definition	It is the mathematical model which is mainly inspired by the biological neuron system in the human brain.	It is also composed of several processing pieces known as neurons that are linked together via synapses.
Processing	Its processing was sequential and centralized.	It processes the information in a parallel and distributive manner.
Size	It is small in size.	It is large in size.
Control Mechanism	Its control unit keeps track of all computer-related operations.	All processing is managed centrally.
Rate	It processes the information at a faster speed.	It processes the information at a slow speed.
Complexity	It cannot perform complex pattern recognition.	The large quantity and complexity of the connections allow the brain to perform complicated tasks.
Feedback	It doesn't provide any feedback.	It provides feedback.
Fault tolerance	There is no fault tolerance.	It has fault tolerance.
Operating Environment	Its operating environment is well-defined and well-constrained	Its operating environment is poorly defined and unconstrained.
Memory	Its memory is separate from a processor, localized, and non-content addressable.	Its memory is integrated into the processor, distributed, and content-addressable.
Reliability	It is very vulnerable.	It is robust.
Learning	It has very accurate structures and formatted data.	They are tolerant to ambiguity.
Response time	Its response time is measured in milliseconds.	Its response time is measured in nanoseconds.

Advantages of Artificial Neural Network (ANN)

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network:

Assurance of proper network structure:

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

Hardware dependence:

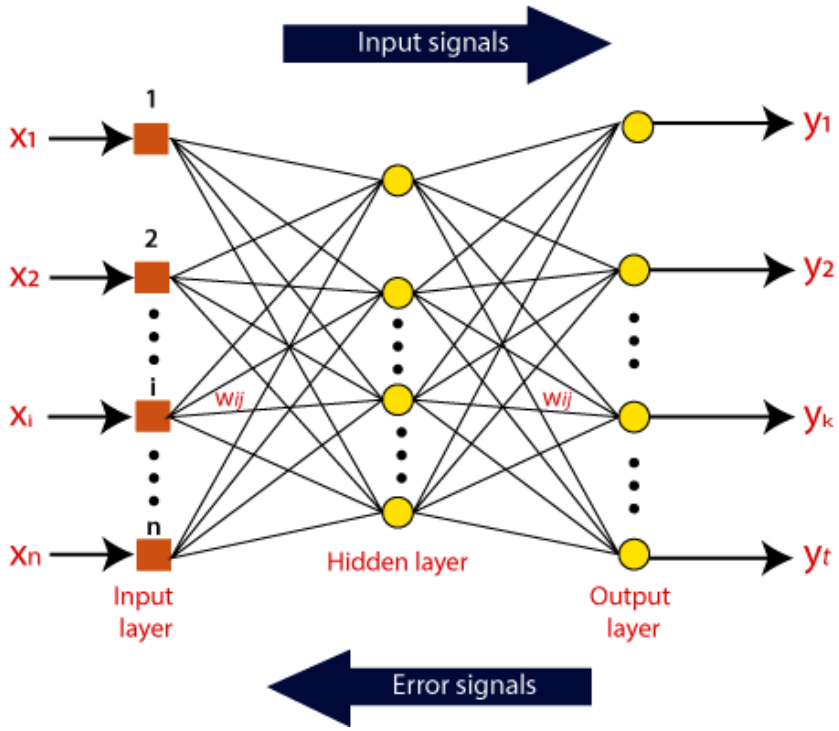
Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

Difficulty of showing the issue to the network:

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

How do artificial neural networks work?

1. Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights.
2. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



3. Afterward, each of the input is multiplied by its corresponding weights, these weights are the details utilized by the artificial neural networks to solve a specific problem. All the weighted inputs are summarized inside the computing unit.
4. If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.
5. The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and etc. activation functions.

$$\sum_{i=1}^{n} Wi * Xi + b$$

Let us take a look at each of them in details:

Binary:

In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

Sigmoidal Hyperbolic:

The Sigmoidal Hyperbola function is generally seen as an "**S**" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:

F(x) = (1/1 + exp(-????x))

Where ???? is considered the Steepness parameter.

Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

Feed-Forward ANN:

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

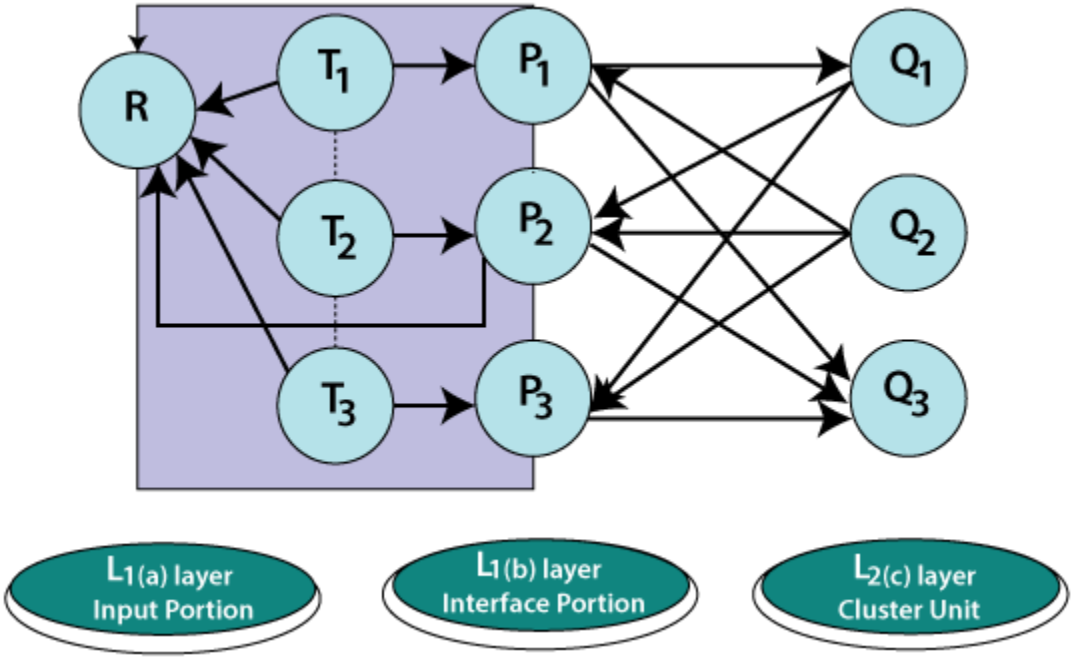
Adaptive Resonance Theory

The Adaptive Resonance Theory (ART) was incorporated as a hypothesis for human cognitive data handling. The hypothesis has prompted neural models for pattern recognition and unsupervised learning. ART system has been utilized to clarify different types of cognitive and brain data.

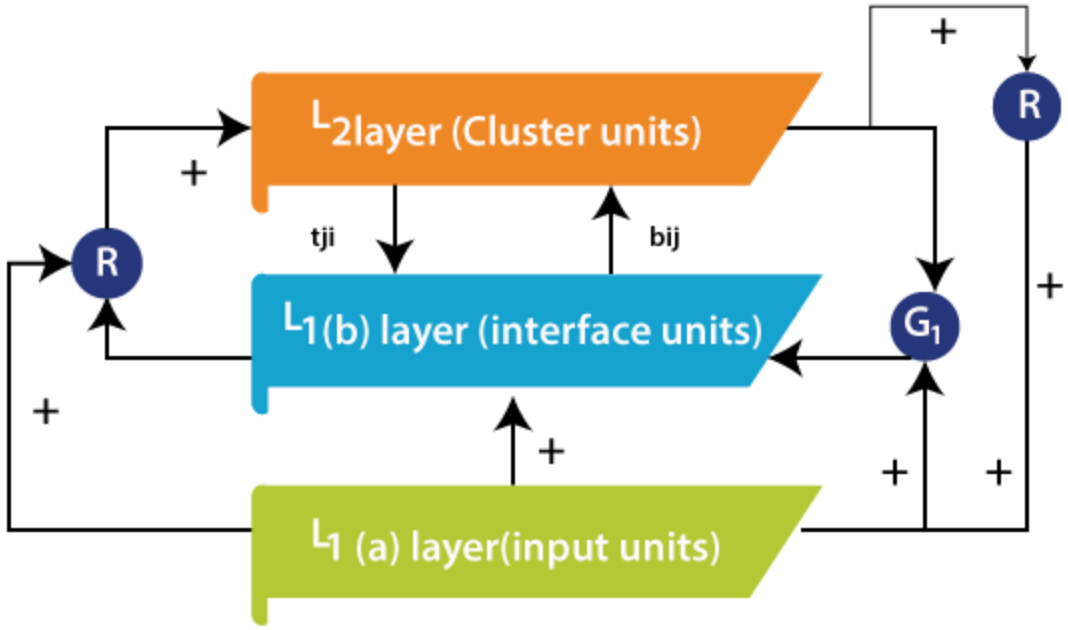
The Adaptive Resonance Theory addresses the **stability-plasticity**(stability can be defined as the nature of memorizing the learning and plasticity refers to the fact that they are flexible to gain new information) dilemma of a system that asks how learning can proceed in response to huge input patterns and simultaneously not to lose the stability for irrelevant patterns. Other than that, the stability-elasticity dilemma is concerned about how a system can adapt new data while keeping what was learned before. For such a task, a feedback mechanism is included among the ART neural network layers. In this neural network, the data in the form of processing elements output reflects back and ahead among layers. If an appropriate pattern is build-up, the resonance is reached, then adaption can occur during this period.

It can be defined as the formal analysis of how to overcome the learning instability accomplished by a competitive learning model, let to the presentation of an expended hypothesis, called **adaptive resonance theory** (ART). This formal investigation indicated that a specific type of top-down learned feedback and matching mechanism could significantly overcome the instability issue. It was understood that top-down attentional mechanisms, which had prior been found through an investigation of connections among cognitive and reinforcement mechanisms, had similar characteristics as these code-stabilizing mechanisms. In other words, once it was perceived how to solve the instability issue formally, it also turned out to be certain that one did not need to develop any quantitatively new mechanism to do so. One only needed to make sure to incorporate previously discovered attentional mechanisms. These additional mechanisms empower code learning to self- stabilize in response to an essentially arbitrary input system. **Grossberg** presented the basic principles of the adaptive resonance theory. A category of ART called ART1 has been described as an arrangement of ordinary differential equations by carpenter and Grossberg. These theorems can predict both the order of search as the function of the learning history of the system and the input patterns.

ART1 is an unsupervised learning model primarily designed for recognizing binary patterns. It comprises an attentional subsystem, an orienting subsystem, a vigilance parameter, and a reset module, as given in the figure given below. The vigilance parameter has a huge effect on the system. High vigilance produces higher detailed memories. The ART1 attentional comprises of two competitive networks, comparison field layer L1 and the recognition field layer L2, two control gains, Gain1 and Gain2, and two short-term memory (STM) stages S1 and S2. Long term memory (LTM) follows somewhere in the range of S1 and S2 multiply the signal in these pathways.



Gains control empowers L1 and L2 to recognize the current stages of the running cycle. STM reset wave prevents active L2 cells when mismatches between bottom-up and top-down signals happen at L1. The comparison layer gets the binary external input passing it to the recognition layer liable for coordinating it to a classification category. This outcome is given back to the comparison layer to find out when the category coordinates the input vector. If there is a match, then a new input vector is read, and the cycle begins once again. If there is a mismatch, then the orienting system is in charge of preventing the previous category from getting a new category match in the recognition layer. The given two gains control the activity of the recognition and the comparison layer, respectively. The reset wave specifically and enduringly prevents active L2 cell until the current is stopped. The offset of the input pattern ends its processing L1 and triggers the offset of Gain2. Gain2 offset causes consistent decay of STM at L2 and thereby prepares L2 to encode the next input pattern without bais.



ART1 Implementation process:

ART1 is a self-organizing neural network having input and output neurons mutually couple using bottom-up and top-down adaptive weights that perform recognition. To start our methodology, the system is first trained as per the adaptive resonance theory by inputting reference pattern data under the type of 5*5 matrix into the neurons for clustering within the output neurons. Next, the maximum number of nodes in L2 is defined following by the vigilance parameter. The inputted pattern enrolled itself as short term memory activity over a field of nodes L1. Combining and separating pathways from L1 to coding field L2, each weighted by an adaptive long-term memory track, transform into a net signal vector T. Internal competitive dynamics at L2 further transform T, creating a compressed code or content addressable memory. With strong competition, activation is concentrated at the L2 node that gets the maximal L1 → L2 signal. The primary objective of this work is divided into four phases as follows Comparision, recognition, search, and learning.

Advantage of adaptive learning theory(ART):

It can be coordinated and utilized with different techniques to give more precise outcomes.

It doesn't ensure stability in forming clusters.

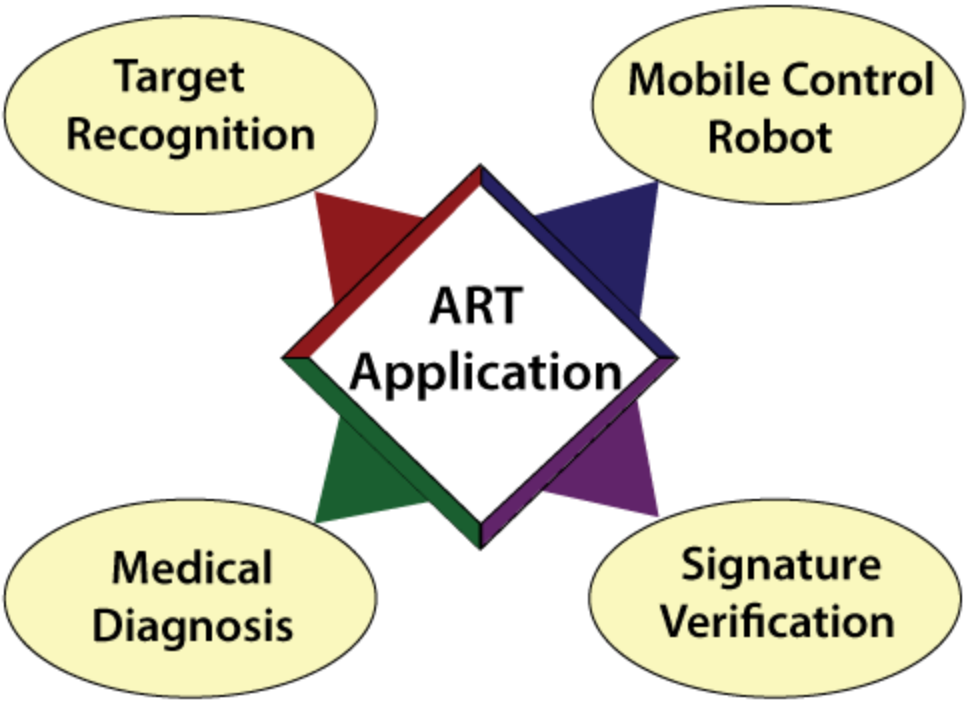
It can be used in different fields such as face recognition, embedded system, and robotics, target recognition, medical diagnosis, signature verification, etc.

It shows stability and is not disturbed by a wide range of inputs provided to inputs.

It has got benefits over competitive learning. The competitive learning cant include new clusters when considered necessary.

Application of ART:

ART stands for Adaptive Resonance Theory. ART neural networks used for fast, stable learning and prediction have been applied in different areas. The application incorporates target recognition, face recognition, medical diagnosis, signature verification, mobile control robot.



Target recognition:

Fuzzy ARTMAP neural network can be used for automatic classification of targets depend on their radar range profiles. Tests on synthetic data show the fuzzy ARTMAP can result in substantial savings in memory requirements when related to k nearest neighbor(kNN) classifiers. The utilization of multiwavelength profiles mainly improves the performance of both kinds of classifiers.

Medical diagnosis:

Medical databases present huge numbers of challenges found in general information management settings where speed, use, efficiency, and accuracy are the prime concerns. A direct objective of improved computer-assisted medicine is to help to deliver intensive care in situations that may be less than ideal. Working with these issues has stimulated several ART architecture developments, including ARTMAP-IC.

Signature verification:

Automatic signature verification is a well known and active area of research with various applications such as bank check confirmation, ATM access, etc. the training of the network is finished using ART1 that uses global features as input vector and the verification and recognition phase uses a two-step process. In the initial step, the input vector is coordinated with the stored reference vector, which was used as a training set, and in the second step, cluster formation takes place.

Mobile control robot:

Nowadays, we perceive a wide range of robotic devices. It is still a field of research in their program part, called artificial intelligence. The human brain is an interesting subject as a model for such an intelligent system. Inspired by the structure of the human brain, an artificial neural emerges. Similar to the brain, the artificial neural network contains numerous simple computational units, neurons that are interconnected mutually to allow the transfer of the signal from the neurons to neurons. Artificial neural networks are used to solve different issues with good outcomes compared to other decision algorithms.

Limitations of ART:

Some ART networks are contradictory as they rely on the order of the training data, or upon the learning rate.

Building Blocks

Neural networks are made of shorter modules or building blocks, same as atoms in matter and logic gates in electronic circuits. Once we know what the blocks are, we can combine them to solve a variety of problems.

Processing of Artificial neural network depends upon the given three building blocks:

- Network Topology
- Adjustments of weights or learning
- Activation functions

In this tutorial, we will discuss these three building blocks of ANN in detail.

Network Topology:

The topology of a neural network refers to the way how Neurons are associated, and it is a significant factor in network functioning and learning. A common topology in unsupervised learning is a direct mapping of inputs to a group of units that represents categories, for example, self-organizing maps. The most widely recognized topology in supervised learning is completely associated, three-layer, feedforward network (Backpropagation, Radial Basis Function Networks). All input values are associated with all neurons in the hidden layer (hidden because they are not noticeable in the input or the output), the output of the hidden neurons are associated to all neurons in the output layer, and the activation functions of the output neurons establish the output of the entire network. Such networks are well-known partly because hypothetically, they are known to be universal function approximators, for example, a sigmoid or Gaussian.

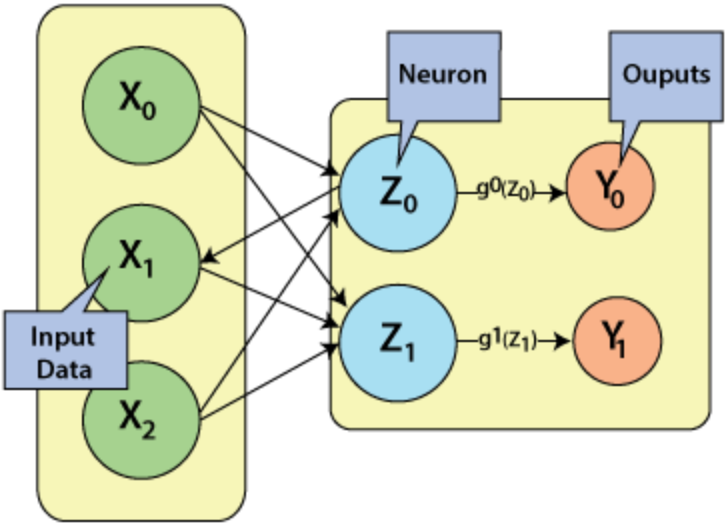
Feedforward network:

The advancement of layered feed-forward networks initiated in the late **1950s**, given by **Rosenblatt's** perceptron and **Widrow's** Adaptive linear Element (ADLINE).The perceptron and ADLINE can be defined as a single layer networks and are usually referred to as single-layer perceptron's. Single-layer perceptron's can only solve linearly separable problems. The limitations of the single-layer network have prompted the advancement of multi-layer feed-forward networks with at least one hidden layer, called multi-layer perceptron (MLP) networks. MLP networks overcome various limitations of single-layer perceptron's and can be prepared to utilize the backpropagation algorithm. The backpropagation method was invented autonomously several times.

In 1974, Werbos created a backpropagation training algorithm. However, Werbos work remained unknown in the scientific community, and in 1985, parker rediscovers the technique. Soon after Parker published his discoveries, Rumelhart, Hinton, and Williams also rediscovered the method. It is the endeavors of Rumelhart and the other individual if the Parallel Distributed Processing (PDP) group, that makes the backpropagation method a pillar of neurocomputing.

Single-layer feedforward network:

Rosenblatt first constructed the single-layer feedforward network in the late 1950s and early 1990s. The concept of feedforward artificial neural network having just one weighted layer. In other words, we can say that the input layer is completely associated with the outer layer.



Multilayer feedforward network:

A multilayer feedforward neural network is a linkage of perceptrons in which information and calculations flow are uni-directional, from the input data to the outputs. The total number of layers in a neural network is the same as the total number of layers of perceptrons. The easiest neural network is one with a single input layer and an output layer of perceptrons. The concept of feedforward artificial neural network having more than one weighted layer. As the system has at least one layer between the input and the output layer, it is called the hidden layer.

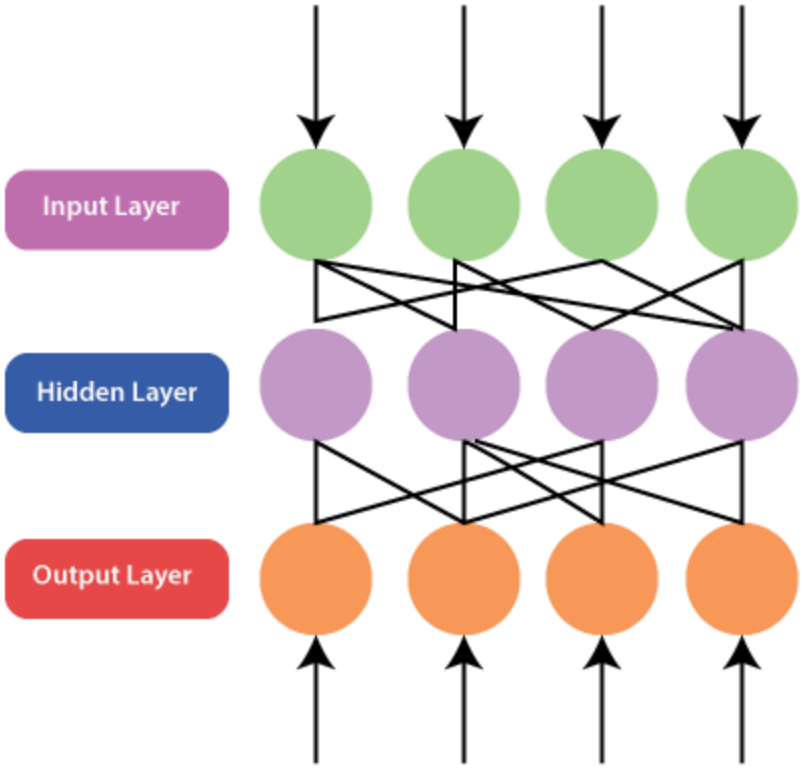
Feedback network:

A feedback based prediction refers to an approximation of an outcome in an iterative way where each iteration's operation depends on the present outcome. Feedback is a common way of making predictions in different fields, ranging from control hypothesis to psychology. Using feedback associations is also additionally exercised by biological organisms, and the brain is proposing a vital role for it in complex cognition.

In other words, we can say that a feedback network has feedback paths, which implies the signal can flow in both directions using loops. It makes a non-linear dynamic system, which changes continuously until it reaches the equilibrium state. It may be divided into the following types:

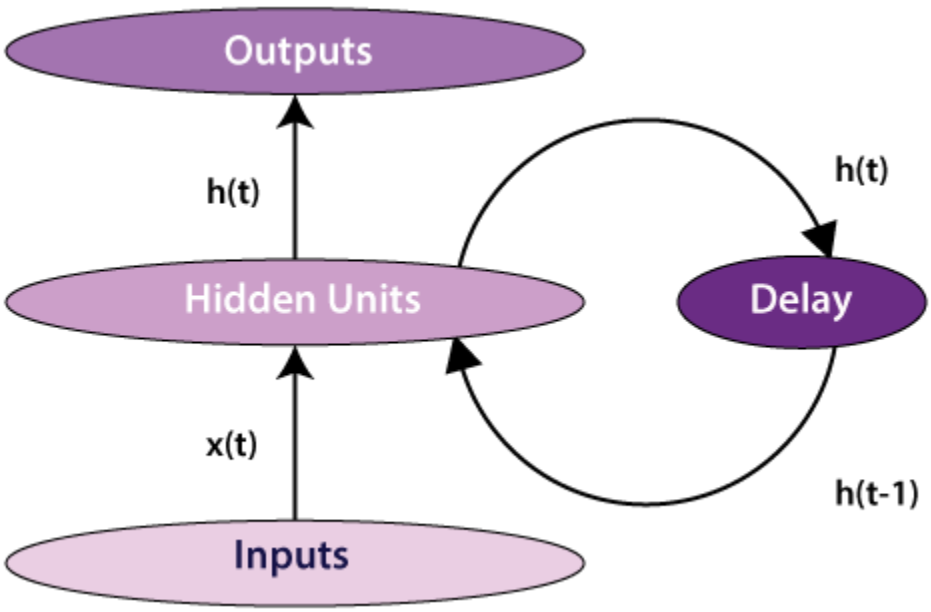
Recurrent network:

The human brain is a recurrent neural network that refers to a network of neurons with feedback connections. It can learn numerous behaviors, sequence, processing tasks algorithms, and programs that are not learnable by conventional learning techniques. It explains the rapidly growing interest in artificial recurrent networks for technical applications. For example, general computers that can learn algorithms to map input arrangements to output arrangements, with or without an instructor. They are computationally more dominant and biologically more conceivable than other adaptive methodologies. For example, Hidden Markov models (no continuous internal states), feedforward networks, and supportive vector machines (no internal states).



Fully recurrent network:

The most straightforward form of a fully recurrent neural network is a Multi-Layer Perceptron (MLP) with the previous set of hidden unit activations, feeding back along with the inputs. In other words, it is the easiest neural network design because all nodes are associated with all other nodes with every single node work as both input and output.

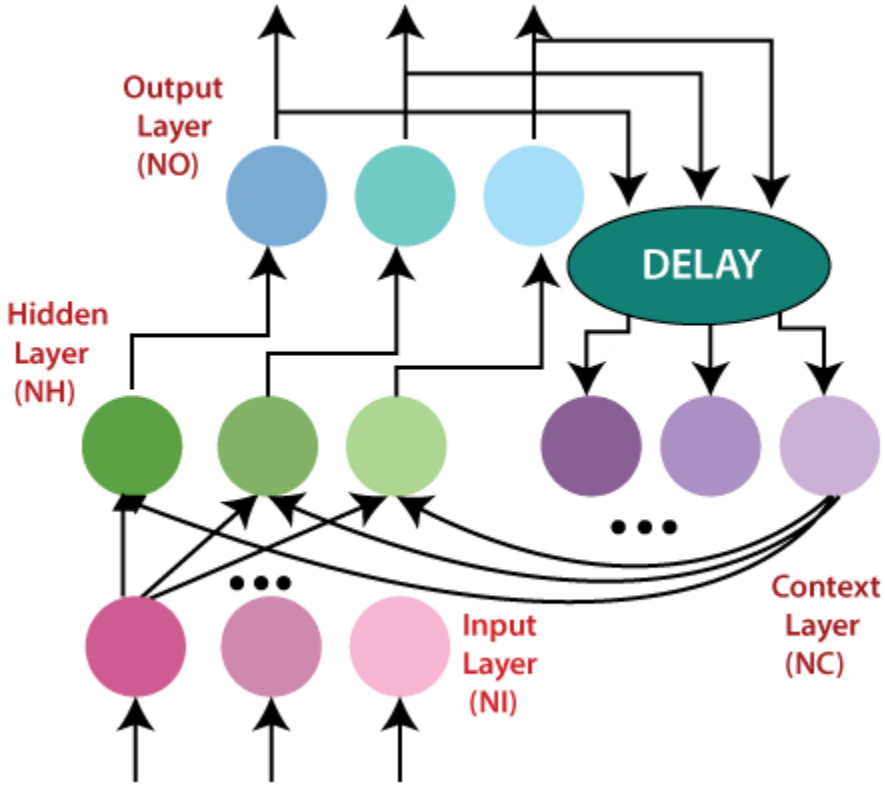


Note that the time '**t**' has to be discretized, with the activations updated at each time interval. The time scale may compare to the activity of real neurons, or for artificial systems whenever step size fitting for

the given problem can be used. A delay unit should be introduced to hold activations until they are prepared at the next time interval.

Jordan network:

The Jordan network refers to a simple neural structure in which only one value of the process input signal (from the previous sampling) and only one value of the delayed output signal of the model (from the previous sampling) are utilized as the inputs of the network. In order to get a computationally basic MPC (Model Predictive Control) algorithm, the nonlinear Jordan neural model is repeatedly linearized on line around an operating point, which prompts a quadratic optimization issue. Adequacy of the described MPC algorithm is compared with that of the nonlinear MPC scheme with on-line nonlinear optimization performed at each sampling instant.



Adjustments of Weights or Learning:

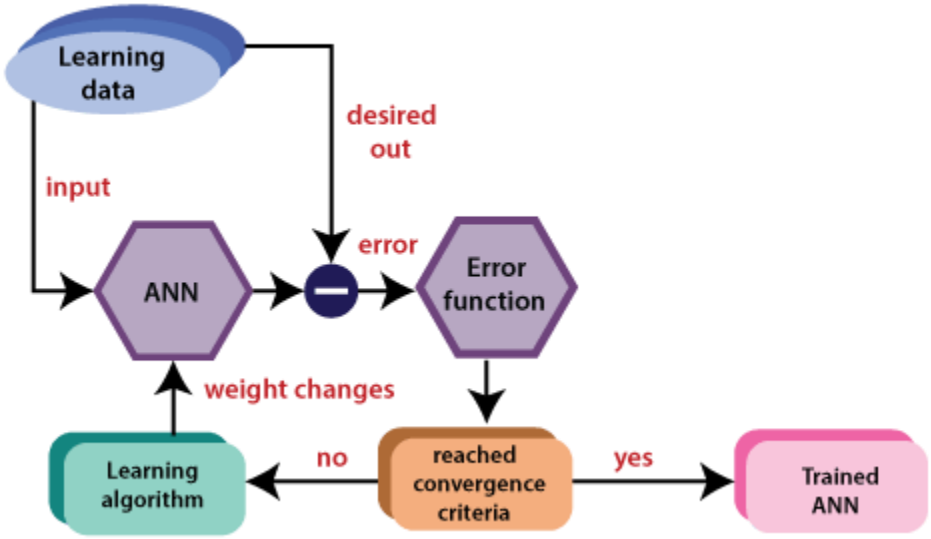
Learning in ANN is the technique for changing the weights of associations between the neurons of a specified network. Learning in artificial neural networks can be characterized into three different categories, namely supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning:

Supervised learning consists of two words supervised and learning. Supervise intends to guide. We have supervisors whose duty is to guide and show the way. We can see a similar case in the case of learning. Here the machine or program is learning with the help of the existing data set. We have a data set, and we assume the results of new data relying upon the behavior of the existing data sets. It implies the existing data sets acts as a supervisor or boss to find the new data. A basic example being electronic gadgets price prediction. The price of electronic gadgets is predicted depending on what is observed with the prices of other digital gadgets.

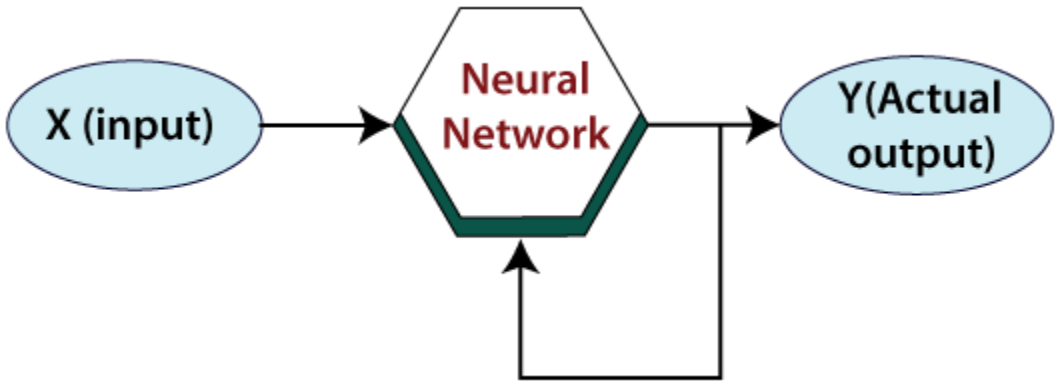
During the training of artificial neural networks under supervised learning, the input vector is given to the network, which offers an output vector. Afterward, the output vector is compared with the desired

output vector. An error signal is produced if there is a difference between the actual output and the desired output vector. Based on this error signal, the weight is adjusted until the actual output is matched with the desired output.



Unsupervised learning:

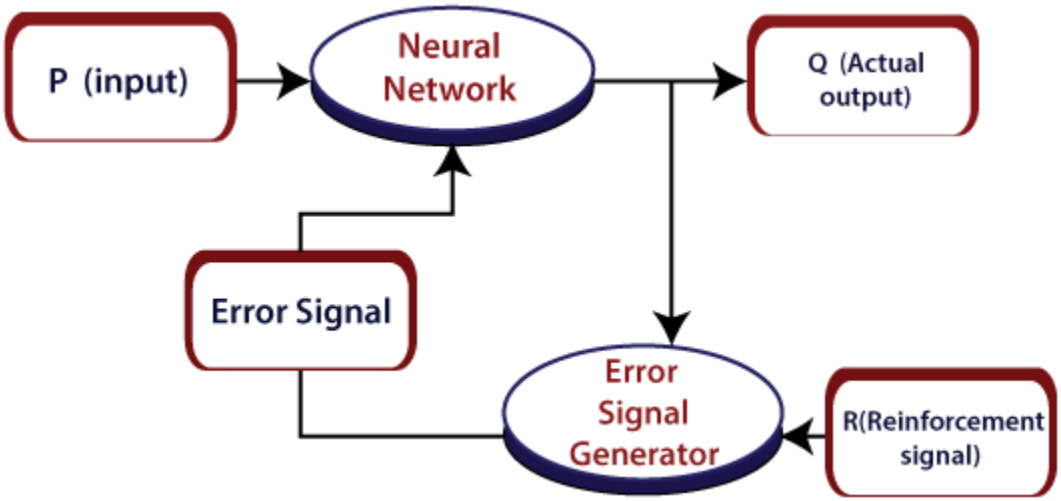
As the name suggests, unsupervised learning refers to predict something without any supervision or help from existing data. In this learning, the program learns by dividing the data with similar characteristics into similar groups. In supervised learning, the data are grouped, relying upon similar characteristics. In this situation, there are no existing data to look for direction. In other words, there is no supervisor. During the training of the artificial neural network under unsupervised learning, the input vectors of a comparative type are joined to form clusters. At the point when a new input pattern is implemented, then the neural network gives an output response showing the class to which the input pattern belongs. There is no feedback from the environment about what should be the ideal output and if it is either correct or incorrect. Consequently, in this type of learning, the network itself must find the patterns and features from the input data and the connection for the input data over the output.



Reinforcement learning:

Reinforcement Learning (RL) is a technique that helps to solve control optimization issues. By using control optimization, we can recognize the best action in each state visited by the system in order to optimize some objective function. Typically, reinforcement learning comes into existence when the system has a huge number of states and has a complex stochastic structure, which is not responsible to

closed-form analysis. If issues have a relatively small number of states, then the random structure is relatively simple, so that one can utilize dynamic programming.



As the name suggests, this kind of learning is used to strengthen the network over some analyst data. This learning procedure is like supervised learning. However, we may have very little information. In reinforcement learning, during the training of the network, the network gets some feedback from the system. This makes it fairly like supervised learning. The feedback acquired here is evaluative, not instructive, which implies there is no instructor as in supervised learning. After getting the feedback, the networks perform modifications of the weights to get better Analyst data in the future.

Activation Function:

Activation functions refer to the functions used in neural networks to compute the weighted sum of input and biases, which is used to choose the neuron that can be fire or not. It controls the presented information through some gradient processing, normally gradient descent. It produces an output for the neural network that includes the parameters in the data.

Activation function can either be linear or non-linear, relying on the function it shows. It is used to control the output of outer neural networks across various areas, such as speech recognition, segmentation, fingerprint detection, cancer detection system, etc.

In the artificial neural network, we can use activation functions over the input to get the precise output. These are some activation functions that are used in ANN.

Linear Activation Function:

The equation of the linear activation function is the same as the equation of a straight line i.e.

Y= MX+ C

If we have many layers and all the layers are linear in nature, then the final activation function of the last layer is the same as the linear function of the first layer. The range of a linear function is –infinitive to + infinitive.

Linear activation function can be used at only one place that is the output layer.

Sigmoid function:

Sigmoid function refers to a function that is projected as S - shaped graph.

$A = 1/(1+e^{-x})$

This function is non-linear, and the values of x lie between -2 to +2. So that the value of X is directly proportional to the values of Y. It means a slight change in the value of x would also bring about changes in the value of y.

Tanh Function:

The activation function, which is more efficient than the sigmoid function is Tanh function. Tanh function is also known as Tangent Hyperbolic Function. It is a mathematical updated version of the sigmoid function. Sigmoid and Tanh function are similar to each other and can be derived from each other.

$F(x)= \tanh(x) = 2/(1+e^{-2X}) - 1$
OR
Tanh (x) = 2 * sigmoid(2x) - 1

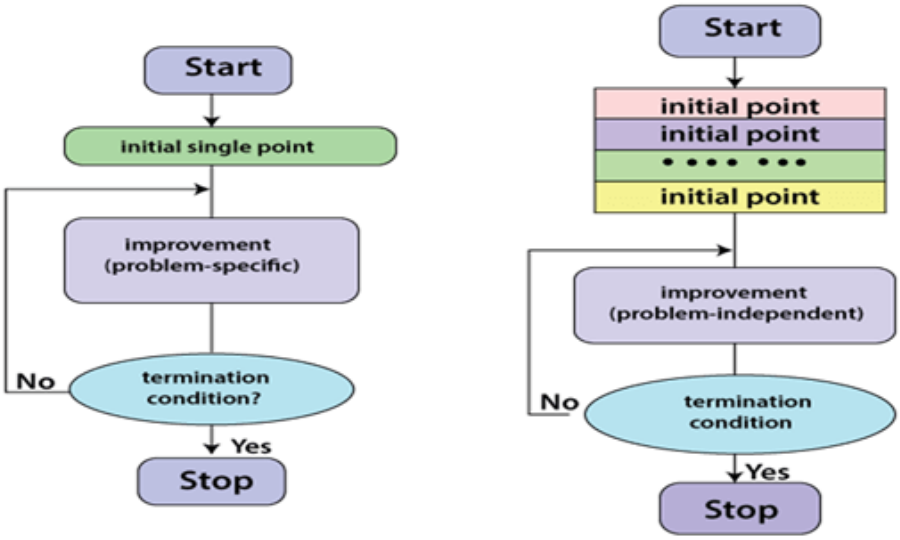
Genetic Algorithm

Genetic algorithm (GAs) are a class of search algorithms designed on the natural evolution process. Genetic Algorithms are based on the principles of **survival of the fittest**.

A Genetic Algorithm method inspired in the world of Biology, particularly, the Evolution Theory by **Charles Darwin,** is taken as the basis of its working. **John Holland** introduced the Genetic Algorithm in **1975**. Genetic Algorithms are utilized to tackle optimization problems by copying the evolutionary behavior of species. From an initial random population of solutions, this population is advanced through selection, mutation, and crossover operators, inspired in natural evolution. By implementing the given set of operations, the population goes through an iterative procedure in which it reaches various states, and each one is called **generation**. As a result of this procedure, the population is expected to reach a generation in which it contains a decent solution to the problem. In the Genetic Algorithm, the solution of the problem is coded as a **string of bits** or **real numbers**.

ANNs working principle:

The working principle of a standard Genetic Algorithm is illustrated in the given figure. The significant steps involved are the generation of a population of the solution, identifying the objective function and fitness function, and the application of genetic operators. These aspects are described with the assistance of a fundamental genetic algorithm as below.



Start:
It generates a random population of n chromosomes.
Fitness:
It calculates the fitness f(x) of each chromosome x in the population.
New Population:

It generates a new population by repeating the following steps until the New population is finished.

Selection:

It chooses two parent chromosomes from a population as per their fitness. The better fitness, the higher the probability of getting selected.

Crossover:

In crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, the offspring is the exact copy of the parents.

Mutation:

In mutation probability, mutate new offspring at each locus.

Accepting:

It places new offspring in the new population.

Replace:

It uses the newly generated population for a further run of the algorithm.

Test:

If the end condition is satisfied, then it stops and returns the best solution in the current population.

Loop:

In this step, you need to go to the second step for fitness evaluation.

The basic principle behind the genetic algorithms is that they generate and maintain a population of individuals represented by chromosomes. Chromosomes are a character string practically equivalent to the chromosomes appearing in DNA. These chromosomes are usually encoded solutions to a problem. It undergoes a process of evolution as per rules of selection, reproduction, and mutation. Each individual in the environment (represented by chromosome) gets a measure of its fitness in the environment. Reproduction chooses individuals with high fitness values in the population. Through crossover and mutation of such individuals, a new population is determined in which individuals might be an even better fit for their environment. The process of crossover includes two chromosomes swapping chunks of data and is analogous to the process of reproduction. Mutation introduces slight changes into a little extant of the population, and it is representative of an evolutionary step.

Difference between traditional and genetic approach:

An algorithm is a progression of steps for solving a problem. A genetic algorithm is a problem-solving technique that uses genetics as its model of problem-solving. It is a search method to find approximate solutions to optimization and search issues. One can easily distinguish between a traditional and a genetic algorithm.

Traditional Algorithm	Genetic Algorithm
It selects the next point in the series by a deterministic computation.	It selects the next population by computation, which utilizes random number generators.
It creates an individual point at each iteration. The sequence of points approaches an optimal solution.	It creates a population of points at every iteration. The best point in the population approaches an optimal solution.
Advancement in each iteration is problem specific.	Concurrence in each iteration is a problem independent.

Advantages of Genetic Algorithm:

The genetic algorithm concept is easy to understand.

The genetic algorithm supports multi-objective optimization.

Limitations of Genetic Algorithm:

Although Genetic algorithms have demonstrated to be a quick and powerful problem-solving approach, some limitations are found embedded in it. Some of these limitations are given below:

The first, and most significant, consideration in making a genetic algorithm is characterizing representation of the problem. The language used to determine candidate solutions must be robust. It must be able to endure random changes such that fatal errors don't mistake.

One significant obstacle of genetic algorithms is the coding of the fitness (evaluation) function so that a higher fitness can be achieved, and better solutions for the problem are produced.

Applications of Genetic Algorithm:

Genetic Algorithm in Robotics:

Robotics is one of the most discussed fields in the computer industry today. It is used in various industries in order to increase profitability efficiency and accuracy.

Genetic Algorithm in Financial Planning:

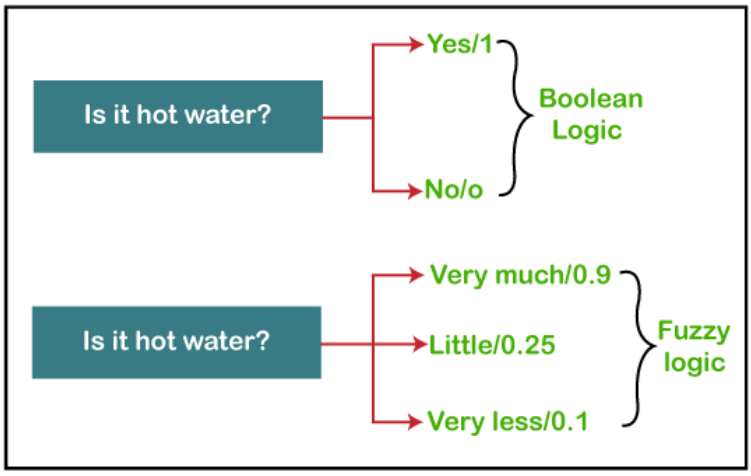
Models for **tactical asset distribution** and **international equity methodologies** have been enhanced with the use of Gas. Genetic algorithms are extremely efficient for financial modeling applications as they are driven by adjustments that can be used to improve the efficiency of predictions and return over the benchmark set. In addition, these methods are robust, permitting a greater range of extensions and constraints, which may not be accommodated in traditional techniques.

Fuzzy Logic

What is Fuzzy Logic?

The '**Fuzzy**' word means the things that are not clear or are vague. Sometimes, we cannot decide in real life that the given problem or statement is either true or false. At that time, this concept provides many values between the true and false and gives the flexibility to find the best solution to that problem.

Example of Fuzzy Logic as comparing to Boolean Logic



Fuzzy logic contains the multiple logical values and these values are the truth values of a variable or problem between 0 and 1. This concept provides the possibilities which are not given by computers, but similar to the range of possibilities generated by humans.

In the Boolean system, only two possibilities (0 and 1) exist, where 1 denotes the absolute truth value and 0 denotes the absolute false value.But in the fuzzy system, there are multiple possibilities present between the 0 and 1, which are partially false and partially true.

The Fuzzy logic can be **implemented in systems such as micro-controllers, workstation-based or large network-based systems for achieving the definite output.** It can also be implemented in both hardware or software.

Characteristics of Fuzzy Logic

Following are the characteristics of fuzzy logic:

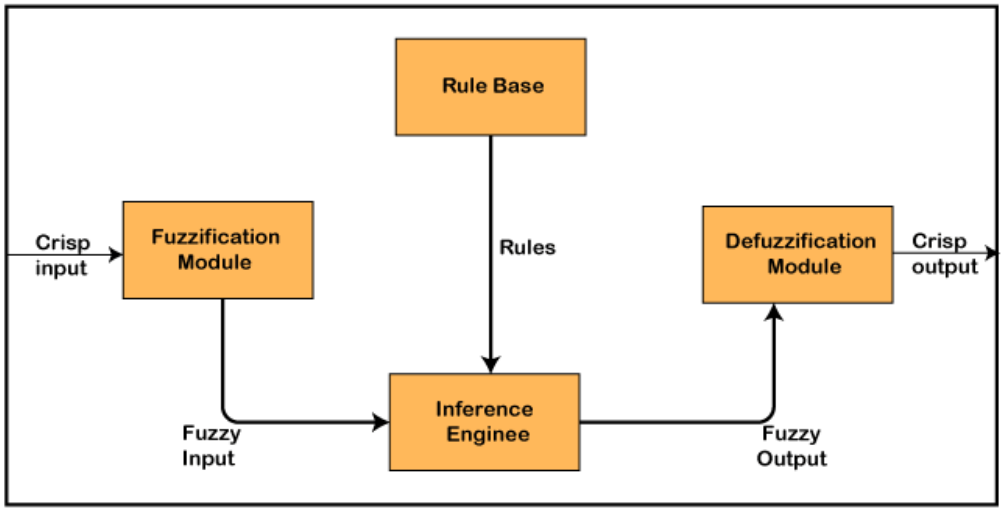
1. This concept is flexible and we can easily understand and implement it.
2. **It is used for helping the minimization of the logics created by the human.**
3. **It is the best method for finding the solution of those problems which are suitable for approximate or uncertain reasoning.**
4. **It allows users to build or create the functions which are non-linear of arbitrary complexity.**
5. In fuzzy logic, everything is a matter of degree.
6. In the Fuzzy logic, any system which is logical can be easily fuzzified.
7. **It is based on natural language processing.**
8. It is also used by the quantitative analysts for improving their algorithm's execution.
9. It also allows users to integrate with the programming.

Architecture of a Fuzzy Logic System

In the architecture of the **Fuzzy Logic** system, each component plays an important role. The architecture consists of the different four components which are given below.

1. Rule Base
2. Fuzzification
3. Inference Engine
4. Defuzzification

Following diagram shows the architecture or process of a Fuzzy Logic system:



1. Rule Base

Rule Base is a component used for storing the set of rules and the If-Then conditions given by the experts are used for controlling the decision-making systems. There are so many updates that come in the Fuzzy theory recently, which offers effective methods for designing and tuning of fuzzy controllers. These updates or developments decreases the number of fuzzy set of rules.

2. Fuzzification

Fuzzification **is a module or component for transforming the system inputs, i.e., it converts the crisp number into fuzzy steps. The crisp numbers are those inputs which are measured by the sensors and then fuzzification passed them into the control systems for further processing.** This component divides the input signals into following five states in any Fuzzy Logic system:

- Large Positive (LP)
- Medium Positive (MP)
- Small (S)
- Medium Negative (MN)
- Large negative (LN)

3. Inference Engine

This component is **a main component in any Fuzzy Logic system (FLS)**, because **all the information is processed in the Inference Engine. It allows users to find the matching degree between the current fuzzy input and the rules.** After the matching degree, this system determines which rule is to be added according to the given input field. When all rules are fired, then they are combined for developing the control actions.

4. Defuzzification

Defuzzification is a module or component, which takes the fuzzy set inputs generated by the **Inference Engine**, and then transforms them into a crisp value. It is the last step in the process of a fuzzy logic system. The crisp value is a type of value which is acceptable by the user. Various techniques are present to do this, but the user has to select the best one for reducing the errors.

Membership Function

The membership function is a function which represents the graph of fuzzy sets, and allows users to quantify the linguistic term. It is a graph which is used for mapping each element of x to the value between 0 and 1.

This function is also known as indicator or characteristics function.

For the Fuzzy set B, the membership function for X is defined as: $\mu B:X \rightarrow [0,1]$. In this function X, each element of set B is mapped to the value between 0 and 1. This is called a degree of membership or membership value.

Classical and Fuzzy Set Theory

To learn about classical and Fuzzy set theory, firstly you have to know about what is set.

Set:

A set is a term, which is a collection of unordered or ordered elements. Following are the various examples of a set:

1. A set of all-natural numbers
2. A set of students in a class.
3. A set of all cities in a state.
4. A set of upper-case letters of the alphabet.

Types of Set:

There are following various categories of set:

1. Finite
2. Empty
3. Infinite
4. Proper
5. Universal
6. Subset
7. Singleton
8. Equivalent Set
9. Disjoint Set

Classical Set

It is a type of set which collects the distinct objects in a group. The sets with the crisp boundaries are classical sets. In any set, each single entity is called an element or member of that set.

Mathematical Representation of Sets:

Any set can be easily denoted in the following two different ways:

1. Roaster Form: This is also called as a tabular form. In this form, the set is represented in the following way:

Set_name = { element1, element2, element3, ..., element N}

The elements in the set are enclosed within the brackets and separated by the commas. Following are the two examples which describes the set in Roaster or Tabular form:

Example 1:

Set of Natural Numbers: N={1, 2, 3, 4, 5, 6, 7, ...,n).

Example 2:

Set of Prime Numbers less than 50: X={2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47}.

2. Set Builder Form: Set Builder form defines a set with the common properties of an element in a set. In this form, the set is represented in the following way:

A = {x:p(x)}

The following example describes the set in the builder form:

The set {2, 4, 6, 8, 10, 12, 14, 16, 18} is written as:

B = {x:2 ≤ x < 20 and (x%2) = 0}

Operations on Classical Set

Following are the various operations which are performed on the classical sets:

1. Union Operation
2. Intersection Operation
3. Difference Operation
4. Complement Operation

1. Union: This operation is denoted by (A U B). A U B is the set of those elements which exist in two different sets A and B. This operation combines all the elements from both the sets and make a new set. It is also called a Logical OR operation.

It can be described as:

A ∪ B = { x | x ∈ A OR x ∈ B }.

Example:

Set A = {10, 11, 12, 13}, Set B = {11, 12, 13, 14, 15}, then A ∪ B = {10, 11, 12, 13, 14, 15}

2. Intersection: This operation is denoted by (A ∩ B). A ∩ B is the set of those elements which are common in both set A and B. It is also called a Logical OR operation.

It can be described as:

A ∩ B = { x | x ∈ A AND x ∈ B }.

Example:

Set A = {10, 11, 12, 13}, Set B = {11, 12, 14} then A ∩ B = {11, 12}

3. Difference Operation: This operation is denoted by (A - B). A-B is the set of only those elements which exist only in set A but not in set B.

It can be described as:

$A - B = \{ x \mid x \in A \text{ AND } x \notin B \}$.

4. Complement Operation: This operation is denoted by (A`). It is applied on a single set. A` is the set of elements which do not exist in set A.

It can be described as:

$A' = \{x \mid x \notin A\}$.

Properties of Classical Set

There are following various properties which play an essential role for finding the solution of a fuzzy logic problem.

1. Commutative Property: This property provides the following two states which are obtained by two finite sets A and B:

$A \cup B = B \cup A$

$A \cap B = B \cap A$

2. Associative Property: This property also provides the following two states but these are obtained by three different finite sets A, B, and C:

$A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$

3. Idempotency Property: This property also provides the following two states but for a single finite set A:

$A \cup A = A$

$A \cap A = A$

4. Absorption Property: This property also provides the following two states for any two finite sets A and B:

$A \cup (A \cap B) = A$

$A \cap (A \cup B) = A$

5. Distributive Property: This property also provides the following two states for any three finite sets A, B, and C:

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

6. Identity Property: This property provides the following four states for any finite set A and Universal set X:

$A \cup \phi = A$

$A \cap X = A$

$A \cap \phi = \phi$

$A \cup X = X$

7. Transitive property: This property provides the following state for the finite sets A, B, and C:

If $A \subseteq B \subseteq C$, then $A \subseteq C$

8. Ivolution property: This property provides following state for any finite set A:

$$\overline{\overline{A}} = A$$

9. De Morgan's Law: This law gives the following rules for providing the contradiction and tautologies:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

Fuzzy Set

The set theory of classical is the subset of Fuzzy set theory. Fuzzy logic is based on this theory, which is a generalization of the classical theory of set (i.e., crisp set).

A fuzzy set is a collection of values which exist between 0 and 1. Fuzzy sets are denoted or represented by the tilde (~) character. In the fuzzy set, the partial membership also exists. This theory released as an extension of classical set theory.

This theory is denoted mathematically asA fuzzy set (Ã) is a pair of U and M, where U is the Universe of discourse and M is the membership function which takes on values in the interval [0, 1]. The universe of discourse (U) is also denoted by Ω or X.

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$$

1. Fuzzy set is having degrees of membership between 1 and 0. Fuzzy sets are represented with tilde character(~). For example, Number of cars following traffic signals at a particular time out of all cars present will have membership value between [0,1].
2. Partial membership exists when member of one fuzzy set can also be a part of other fuzzy sets in the same universe.
3. The degree of membership or truth is not same as probability, fuzzy truth represents membership in vaguely defined sets.
4. If X is an universe of discourse and x is a particular element of X, then a fuzzy set A defined on X and can be written as a collection of ordered pairs Ã = {(x, μÃ (x)), x ϵ X }

Operations on Fuzzy Set

Given Ã and B are the two fuzzy sets, and X be the universe of discourse with the following respective member functions:

$\mu_{\tilde{A}}(x)$ and $\mu_{\tilde{B}}(x)$

The operations of Fuzzy set are as follows:

1. Union Operation: The union operation of a fuzzy set is defined by:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Example:

Let's suppose A is a set which contains following elements:

$A = \{(X_1, 0.6), (X_2, 0.2), (X_3, 1), (X_4, 0.4)\}$

And, B is a set which contains following elements:

$B = \{(X_1, 0.1), (X_2, 0.8), (X_3, 0), (X_4, 0.9)\}$

then,

$AUB = \{(X_1, 0.6), (X_2, 0.8), (X_3, 1), (X_4, 0.9)\}$

Because, according to this operation

For X_1

$\mu_{AUB}(X_1) = \max (\mu_A(X_1), \mu_B(X_1))$

$\mu_{AUB}(X_1) = \max (0.6, 0.1)$

$\mu_{AUB}(X_1) = 0.6$

For X_2

$\mu_{AUB}(X_2) = \max (\mu_A(X_2), \mu_B(X_2))$

$\mu_{AUB}(X_2) = \max (0.2, 0.8)$

$\mu_{AUB}(X_2) = 0.8$

For X_3

$\mu_{AUB}(X_3) = \max (\mu_A(X_3), \mu_B(X_3))$

$\mu_{AUB}(X_3) = \max (1, 0)$

$\mu_{AUB}(X_3) = 1$

For X_4

$\mu_{AUB}(X_4) = \max (\mu_A(X_4), \mu_B(X_4))$

$\mu_{AUB}(X_4) = \max (0.4, 0.9)$

$\mu_{AUB}(X_4) = 0.9$

2. Intersection Operation: The intersection operation of fuzzy set is defined by:

$$\mu_{A \cap B}(x) = \min (\mu_A(x), \mu_B(x))$$

Example:

Let's suppose A is a set which contains following elements:

$A = \{(X_1, 0.3), (X_2, 0.7), (X_3, 0.5), (X_4, 0.1)\}$

And, B is a set which contains following elements:

$B = \{(X_1, 0.8), (X_2, 0.2), (X_3, 0.4), (X_4, 0.9)\}$

then,

$A \cap B = \{(X_1, 0.3), (X_2, 0.2), (X_3, 0.4), (X_4, 0.1)\}$

Because, according to this operation

For X_1

$\mu_{A\cap B}(X_1) = \min(\mu_A(X_1), \mu_B(X_1))$

$\mu_{A\cap B}(X_1) = \min(0.3, 0.8)$

$\mu_{A\cap B}(X_1) = 0.3$

For X_2

$\mu_{A\cap B}(X_2) = \min(\mu_A(X_2), \mu_B(X_2))$

$\mu_{A\cap B}(X_2) = \min(0.7, 0.2)$

$\mu_{A\cap B}(X_2) = 0.2$

For X_3

$\mu_{A\cap B}(X_3) = \min(\mu_A(X_3), \mu_B(X_3))$

$\mu_{A\cap B}(X_3) = \min(0.5, 0.4)$

$\mu_{A\cap B}(X_3) = 0.4$

For X_4

$\mu_{A\cap B}(X_4) = \min(\mu_A(X_4), \mu_B(X_4))$

$\mu_{A\cap B}(X_4) = \min(0.1, 0.9)$

$\mu_{A\cap B}(X_4) = 0.1$

3. Complement Operation: The complement operation of fuzzy set is defined by:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

Example:

Let's suppose A is a set which contains following elements:

$A = \{(X_1, 0.3), (X_2, 0.8), (X_3, 0.5), (X_4, 0.1)\}$

then,

$\bar{A} = \{(X_1, 0.7), (X_2, 0.2), (X_3, 0.5), (X_4, 0.9)\}$

Because, according to this operation

For X_1

$\mu_{\bar{A}}(X_1) = 1 - \mu_A(X_1)$

$\mu_{\bar{A}}(X_1) = 1 - 0.3$

$\mu_{\bar{A}}(X_1) = 0.7$

For X_2

$\mu_{\bar{A}}(X_2) = 1 - \mu_A(X_2)$

$\mu_{\bar{A}}(X_2) = 1 - 0.8$

$\mu_{\bar{A}}(X_2) = 0.2$

For X_3

$\mu_{\bar{A}}(X_3) = 1 - \mu_A(X_3)$

$\mu_{\bar{A}}(X_3) = 1 - 0.5$

$\mu_{\bar{A}}(X_3) = 0.5$

For X_4

$\mu_{\bar{A}}(X_4) = 1-\mu_A(X_4)$

$\mu_{\bar{A}}(X_4) = 1 - 0.1$

$\mu_{\bar{A}}(X_4) = 0.9$

Classical Set Theory	Fuzzy Set Theory
1. This theory is a class of those sets having sharp boundaries.	1. This theory is a class of those sets having un-sharp boundaries.
2. This set theory is defined by exact boundaries only 0 and 1.	2. This set theory is defined by ambiguous boundaries.
3. In this theory, there is no uncertainty about the boundary's location of a set.	3. In this theory, there always exists uncertainty about the boundary's location of a set.
4. This theory is widely used in the design of digital systems.	4. It is mainly used for fuzzy controllers.

Difference Between Crisp Set and Fuzzy Set

S.No	Crisp Set	Fuzzy Set
1	Crisp set defines the value is either 0 or 1.	Fuzzy set defines the value between 0 and 1 including both 0 and 1.
2	It is also called a classical set.	It specifies the degree to which something is true.
3	It shows full membership	It shows partial membership.
4	Eg1. She is 18 years old. Eg2. Rahul is 1.6m tall	Eg1. She is about 18 years old. Eg2. Rahul is about 1.6m tall.
5	Crisp set application used for digital design.	Fuzzy set used in the fuzzy controller.
6	It is bi-valued function logic.	It is infinite valued function logic
7	Full membership means totally true/false, yes/no, 0/1.	Partial membership means true to false, yes to no, 0 to 1.

Applications of Fuzzy Logic

Following are the different application areas where the Fuzzy Logic concept is widely used:

1. It is used in **Businesses** for decision-making support system.
2. It is used in **Automative systems** for controlling the traffic and speed, and for improving the efficiency of automatic transmissions. **Automative systems** also use the shift scheduling method for automatic transmissions.
3. This concept is also used in the **Defence** in various areas. Defence mainly uses the Fuzzy logic systems for underwater target recognition and the automatic target recognition of thermal infrared images.

4. It is also widely used in the **Pattern Recognition and Classification** in the form of Fuzzy logic-based recognition and handwriting recognition. It is also used in the searching of fuzzy images.
5. Fuzzy logic systems also used in **Securities**.
6. It is also used in **microwave oven** for setting the lunes power and cooking strategy.
7. This technique is also used in the area of **modern control systems** such as expert systems.
8. **Finance** is also another application where this concept is used for predicting the stock market, and for managing the funds.
9. It is also used for controlling the brakes.
10. It is also used in the **industries of chemicals** for controlling the ph, and chemical distillation process.
11. It is also used in the **industries of manufacturing** for the optimization of milk and cheese production.
12. It is also used in the vacuum cleaners, and the timings of washing machines.
13. It is also used in heaters, air conditioners, and humidifiers.

Advantages of Fuzzy Logic

Fuzzy Logic has various advantages or benefits. Some of them are as follows:

1. The methodology of this concept works similarly as the human reasoning.
2. Any user can easily understand the structure of Fuzzy Logic.
3. It does not need a large memory, because the algorithms can be easily described with fewer data.
4. It is widely used in all fields of life and easily provides effective solutions to the problems which have high complexity.
5. This concept is based on the set theory of mathematics, so that's why it is simple.
6. It allows users for controlling the control machines and consumer products.
7. The development time of fuzzy logic is short as compared to conventional methods.
8. Due to its flexibility, any user can easily add and delete rules in the FLS system.

Disadvantages of Fuzzy Logic

Fuzzy Logic has various disadvantages or limitations. Some of them are as follows:

1. The run time of fuzzy logic systems is slow and takes a long time to produce outputs.
2. Users can understand it easily if they are simple.
3. The possibilities produced by the fuzzy logic system are not always accurate.
4. Many researchers give various ways for solving a given statement using this technique which leads to ambiguity.
5. Fuzzy logics are not suitable for those problems that require high accuracy.
6. The systems of a Fuzzy logic need a lot of testing for verification and validation.

Defuzzification:

Defuzzification is the process of representing a fuzzy set with a crisp number. Internal representations of data in a fuzzy system are usually fuzzy sets. But the output frequently needs to be a crisp number that can be used to perform a function such as commanding a valve to a desired position in a control application.

Defuzzification is the process of combining the successful fuzzy output sets produced by the inference mechanism. The purpose is to produce the most certain low-level controller action. Several methods exist in the literature to perform defuzzification, the most popular of which is the centre of gravity (CoG) method.

Defuzzification is the process of obtaining a single number from the output of the aggregated fuzzy set. It is used to transfer fuzzy inference results into a crisp output. In other words, defuzzification is realized by a decision-making algorithm that selects the best crisp value based on a fuzzy set. There are several forms of defuzzification including center of gravity (COG), mean of maximum (MOM), and center average methods.

Fuzzy composition can be defined just as it is for crisp (binary) relations. Suppose $\underline{R}$ is a fuzzy relation on X × Y, $\underline{S}$ is a fuzzy relation on Y × Z, and $\underline{T}$ is a fuzzy relation on X × Z; then,

Fuzzy Max–Min composition is defined as:

$$\underline{T} = \underline{R} \circ \underline{S} = \mu_{\bar{T}}(x,z) = \bigvee_{y \in Y} (\mu_{\bar{R}}(x,y) \wedge \mu_{\bar{S}}(y,z))$$

$$= max_{y \in Y}\{min(\mu_{\bar{R}}(x,y), \mu_{\bar{S}}(y,z))\}$$

Fuzzy Max–Product composition is defined as:

$$\underline{T} = \underline{R} \circ \underline{S} = \mu_{\bar{T}}(x,z) = \bigvee_{y \in Y} (\mu_{\bar{R}}(x,y).\mu_{\bar{S}}(y,z))$$

$$= max_{y \in Y}\{(\mu_{\bar{R}}(x,y) \times \mu_{\bar{S}}(y,z))\}$$

Min-Max (Game Playing) Algorithm in Artificial Intelligence

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

Pseudo-code for Min-Max Algorithm:

```
1. function minimax(node, depth, maximizing_Player) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if Maximizing_Player then      // for Maximizer Player
6. maxEva= -infinity
7.  for each child of node do
8.  eva= minimax(child, depth-1, false)
9. maxEva= max(maxEva,eva)        //gives Maximum of the values
10. return maxEva
11.
12. else                          // for Minimizer player
13.  minEva= +infinity
14.  for each child of node do
15.  eva= minimax(child, depth-1, true)
16.  minEva= min(minEva, eva)        //gives minimum of the values
17.  return minEva
```

Initial call:

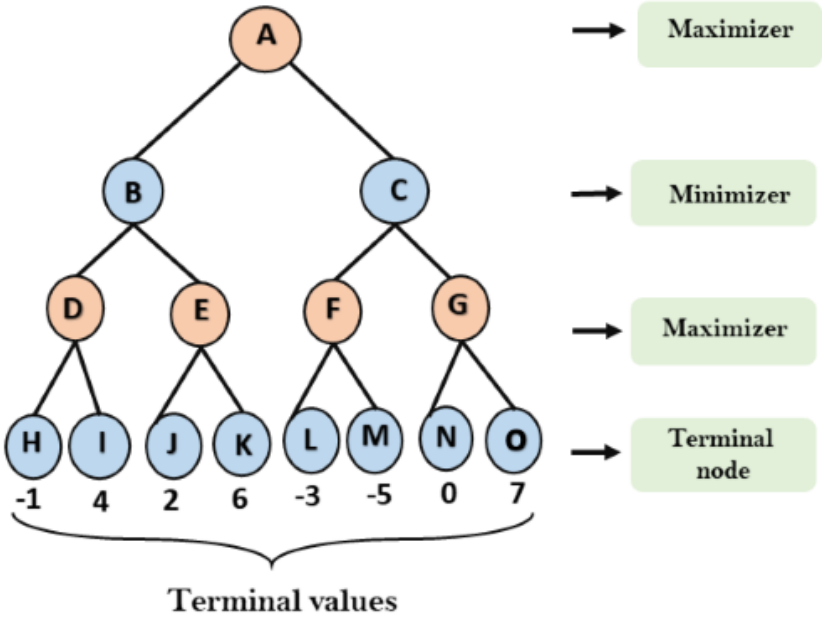
Minimax(node, 3, true)

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.

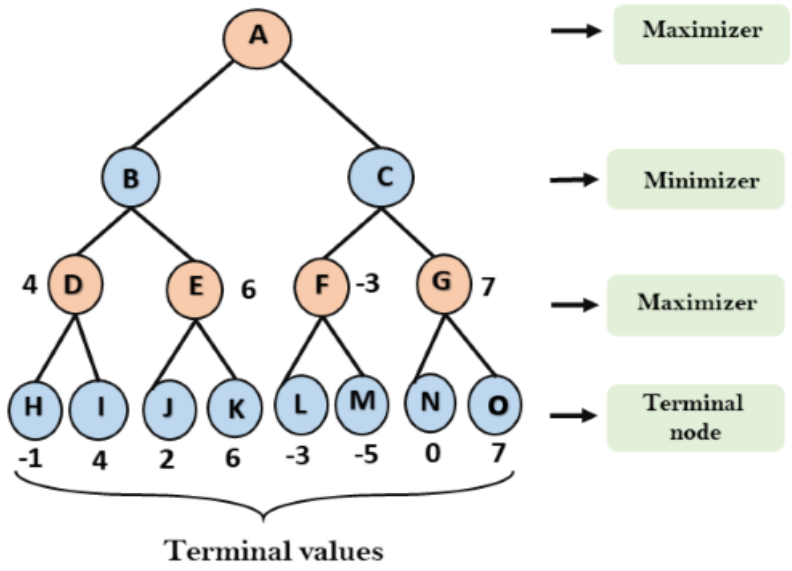
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves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

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



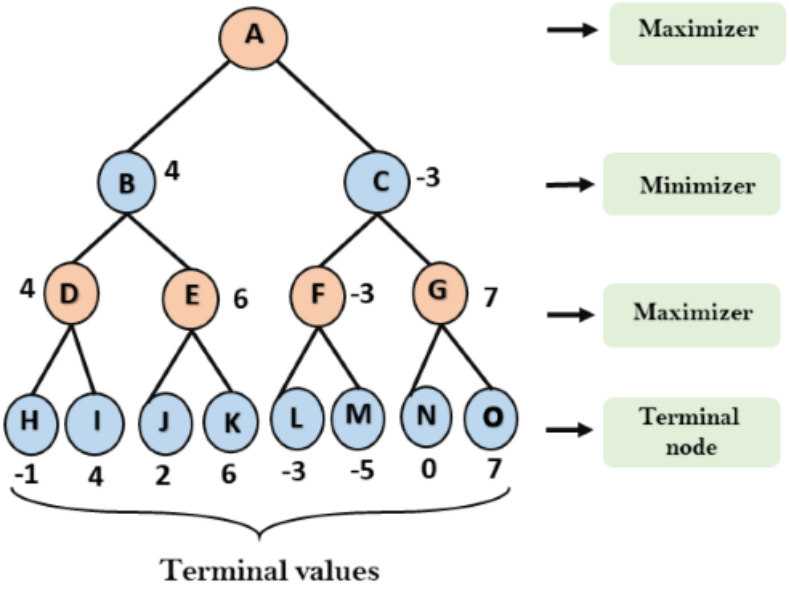
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

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



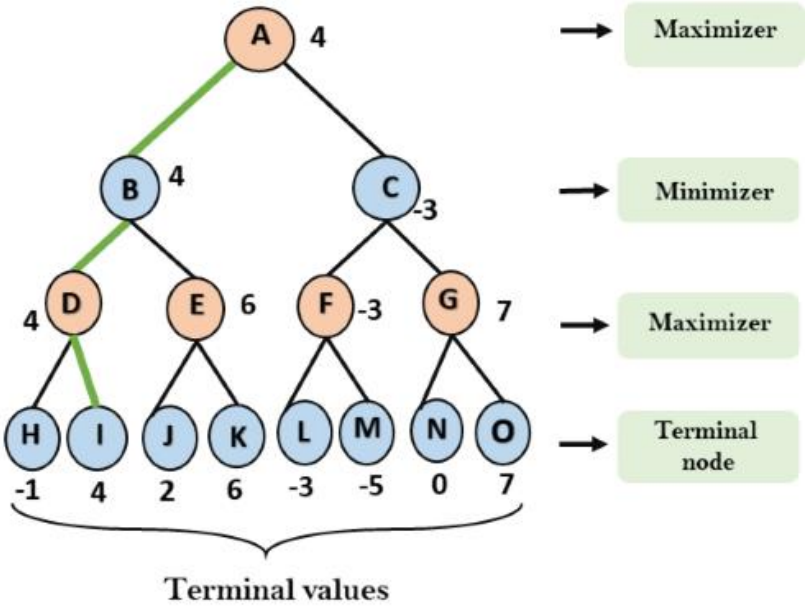
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

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



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

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



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

Properties of Mini-Max algorithm:

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

Limitation of the minimax Algorithm:

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

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

A. Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation.
B. Logical representation means drawing a conclusion based on various conditions. This representation lays down some important communication rules.
C. 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 logics

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.

2. Semantic Network Representation

A. Semantic networks are alternative of predicate logic for knowledge representation.
B. 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.
C. 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 consists of mainly two types of relations:

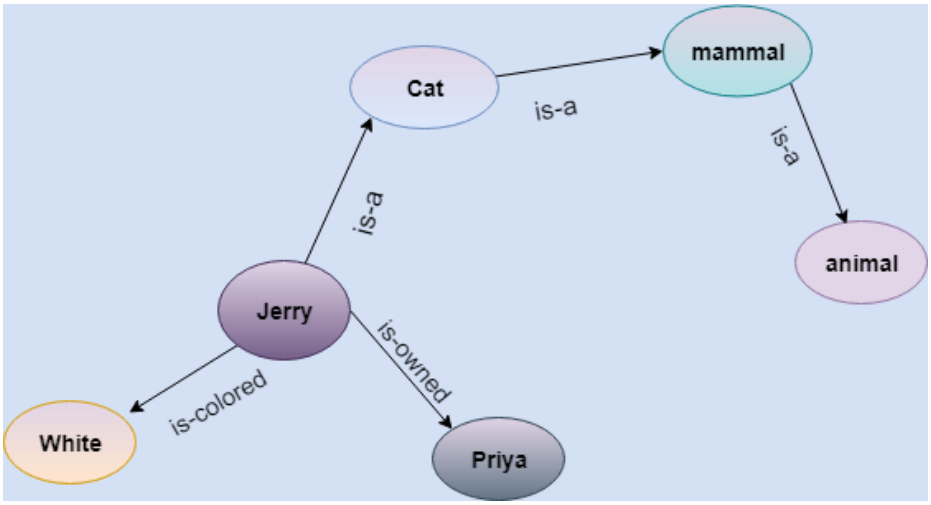
1. IS-A relation (Inheritance)
2. Kind-of-relation

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

Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.

4. Jerry is brown colored.
5. 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

1. 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.
2. **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.
3. 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:

1. The set of production rules
2. Working Memory
3. 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.

Semantic Networks:

A semantic network is a graphic notation for representing knowledge in patterns of interconnected nodes. Semantic networks became popular in artificial intelligence and natural language processing only because it represents knowledge or supports reasoning. These act as another alternative for predicate logic in a form of knowledge representation.

- Semantic nets consist of nodes, links and link labels. In these networks diagram, nodes appear in form of circles or ellipses or even rectangles which represents objects such as physical objects, concepts or situations.
- Links appear as arrows to express the relationships between objects, and link labels specify relations.
- Relationships provide the basic needed structure for organizing the knowledge, so therefore objects and relations involved are also not needed to be concrete.

- Semantic nets are also referred to as associative nets as the nodes are associated with other nodes.

Use of Semantic Networks:

- Representing data
- Revealing structure (relations, proximity, relative importance)
- Supporting conceptual edition
- Supporting navigation

Main Components Of Semantic Networks

- **Lexical component:** nodes denoting physical objects or links are relationships between objects; labels denote the specific objects and relationships
- **Structural component:** the links or nodes from a diagram which is directed.
- **Semantic component:** Here the definitions are related only to the links and label of nodes, whereas facts depend on the approval areas.
- **Procedural part:** constructors permit the creation of the new links and nodes. The removal of links and nodes are permitted by destructors.

Advantages Of Using Semantic Nets

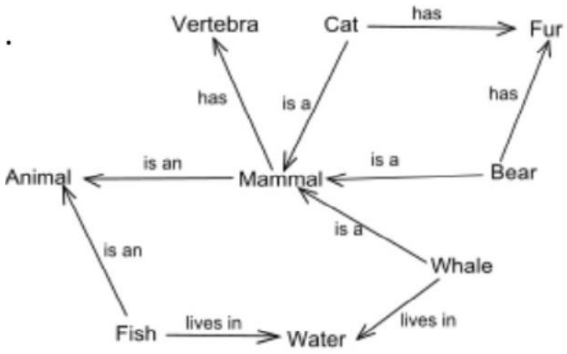
- The semantic network is more natural than the logical representation;
- The semantic network permits using of effective inference algorithm (graphical algorithm)
- They are simple and can be easily implemented and understood.
- The semantic network can be used as a typical connection application among various fields of knowledge, for instance, among computer science and anthropology.
- The semantic network permits a simple approach to investigate the problem space.
- The semantic network gives an approach to make the branches of related components
- The semantic network also reverberates with the methods of the people process data.
- The semantic network is characterized by greater cognitive adequacy compared to logic-based formalism.
- The semantic network has a greater expressiveness compared to logic.

Disadvantages Of Using Semantic Nets

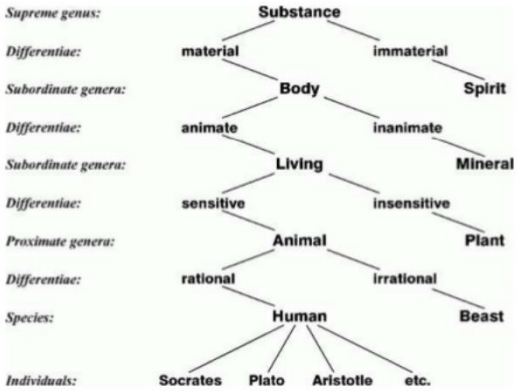
- There is no standard definition for link names
- Semantic Nets are not intelligent, dependent on the creator
- Links are not alike in function or form, confusion in links that asserts relationships and structural links
- Undistinguished nodes that represent classes and that represents individual objects
- Links on object represent only binary relations
- Negation and disjunction and general taxonomical knowledge are not easily expressed.

Six Types Of Semantic Networks:

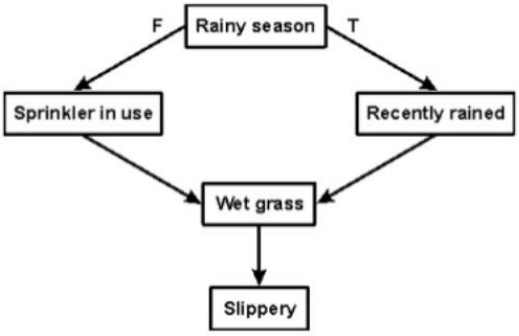
- **Definitional Networks-** These networks emphasises and deals with only the subtype or is a relation between a concept type and a newly defined subtype. A producing network is referred to as generalization hierarchy. It supports the inheritance rule for duplicating attributes.



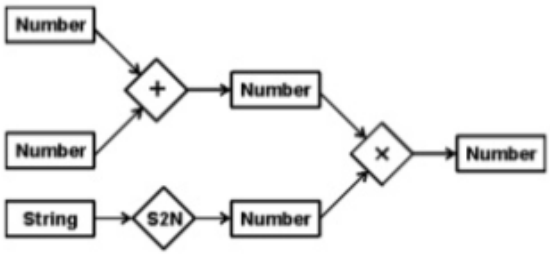
- **Assertion Networks –** Designed to assert propositions is intended to state recommendations. Mostly data in an assertion network is genuine unless it is marked with a modal administrator. Some assertion systems are even considered as the model of the reasonable structures underlying the characteristic semantic natural languages.



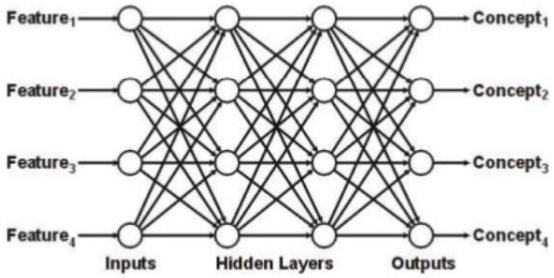
- **Implicational Networks –** Uses Implication as the primary connection for connecting nodes. These networks are also used to explain patterns of convictions, causality and even deductions.



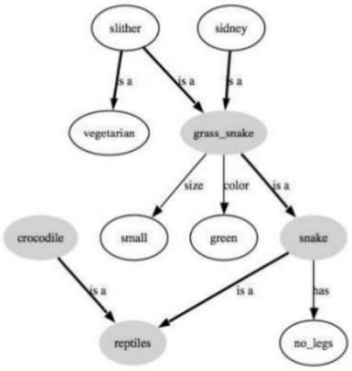
- **Executable Network-** Contains mechanisms that can cause some changes to the network itself by incorporating some techniques, for example, such as attached procedures or marker passing which can perform path messages, or associations and searches for patterns.



- **Learning Networks –** These are the networks that build and extend their representations by acquiring knowledge through examples. Contain mechanisms in such networks brings changes within the network itself through representation by securing information. A classic example could be like, the changing of new information from the old system by including and excluding nodes and arcs, or by changing numerical qualities called weights, and connected with the arcs and nodes.



- **Hybrid Networks –** Networks that combine two or more of previous techniques, either in a single network or in a separate, but closely interacting network Hybrid network has been clearly created to implement ideas regarding human cognitive mechanisms, while some are created generally for computer performance.



What is a frame in AI?

1. A frame is a data structure that represents a "snapshot" of the world at a particular moment in time. It contains all of the information that an AI system needs to know about the world in order to make decisions.

2. Frames are used extensively in AI systems, especially in those that use artificial neural networks. This is because they provide a way to store and manipulate information in a way that is similar to how the human brain does it.

3. Frames are also used in other AI paradigms, such as rule-based systems and decision trees. However, they are not as widely used in these paradigms as they are in neural networks.

Benefits of using frames:

There are many benefits to using frames in AI. Frames provide a structure for representing knowledge that can be used by AI systems to reason about the world. They can also be used to store and retrieve information from memory, and to make inferences about new situations. Frames can also be used to represent plans and goals, and to generate new actions.

Some common frame types in AI

1. **The rule-based system:** This type of AI uses a set of rules to determine how to act in a given situation.
2. **The decision tree:** This type of AI uses a tree-like structure to make decisions.
3. **The neural network:** This type of AI uses a network of interconnected nodes to make decisions.
4. **The genetic algorithm:** This type of AI uses a process of evolution to find solutions to problems.
5. **The fuzzy logic system:** This type of AI uses a set of rules that are not precise to make decisions.

Applications of Frames:

Frames are used in AI applications to represent knowledge in a way that is easy for computers to process. Frames are used to store information about objects, events, and relationships. This information can be used to reasoning and make decisions.

Scripts:

- A script is a structure that prescribes a set of circumstances that could be expected to follow from one another. It is similar to a chain of situations that could be anticipated. Like a script for a play, the script's structure is defined in terms of – (a) Actors (b) Roles (c) Props (d) Scenes
- A script is a structured representation describing a stereotyped sequence of events in a particular context.
- Scripts are used in natural language understanding systems to organize a knowledge base in terms of the situations that the system should understand. Scripts use a frame-like structure to represent the commonly occurring experience like going to the movies eating in a restaurant, shopping in a supermarket.
- Thus, a script is a structure that prescribes a set of circumstances that could be expected to follow on from one another.

Benefits of Scripts:

- Events tend to occur in known runs or patterns.
- A casual relationship between events exist.
- An entry condition exists which allows an event to take place.
- Prerequisites exist upon events taking place.

Components of a script

The components of a script include:

- **Entry condition:** These are basic condition which must be fulfilled before events in the script can occur.
- **Results:** Condition that will be true after events in script occurred.
- **Props:** Slots representing objects involved in events
- **Roles:** These are the actions that the individual participants perform.
- **Track:** Variations on the script. Different tracks may share components of the same scripts.
- **Scenes:** The sequence of events that occur.

Advantages of Scripts

- Ability to predict events.
- A single coherent interpretation maybe builds up from a collection of observations.

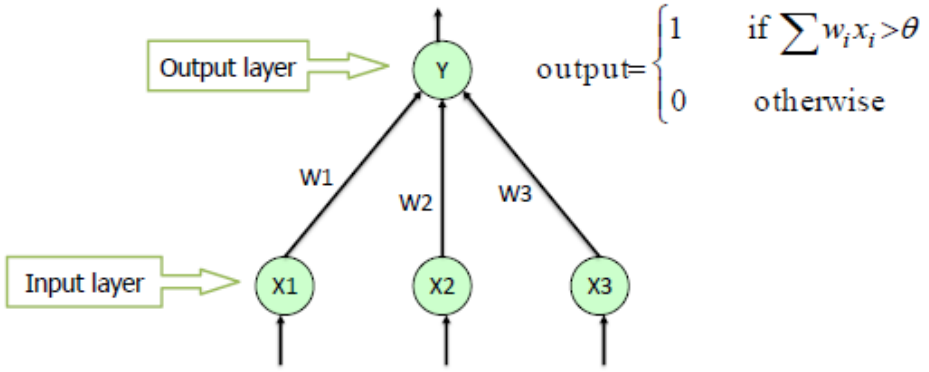
Disadvantages of Scripts

- Less general than frames.
- May not be suitable to represent all kinds of knowledge

Artificial Neural Network – Perceptron

A **single layer perceptron (SLP)** is a feed-forward network based on a threshold transfer function. SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1 , 0).

Single Layer Perceptron



Algorithm:

The single layer perceptron does not have a priori knowledge, so the initial weights are assigned randomly. SLP sums all the weighted inputs and if the sum is above the threshold (some predetermined value), SLP is said to be activated (output=1).

Output

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > \theta \Rightarrow 1$$

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \leq \theta \Rightarrow 0$$

The input values are presented to the perceptron, and if the predicted output is the same as the desired output, then the performance is considered satisfactory and no changes to the weights are made. However, if the output does not match the desired output, then the weights need to be changed to reduce the error.

Perceptron Weights Adjustment

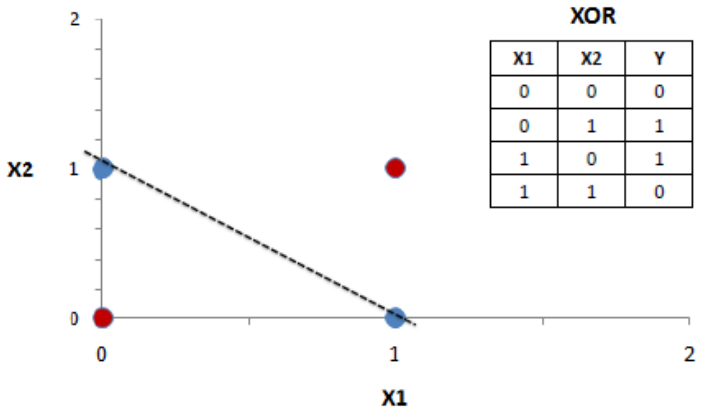
$$\Delta w = \eta \times d \times x$$

$d \rightarrow$ Predicted output - Desired output

$\eta \rightarrow$ Learning rate, usually less than 1

$x \rightarrow$ Input data

Because SLP is a linear classifier and if the cases are not linearly separable the learning process will never reach a point where all the cases are classified properly. The most famous example of the inability of perceptron to solve problems with linearly non-separable cases is the XOR problem.



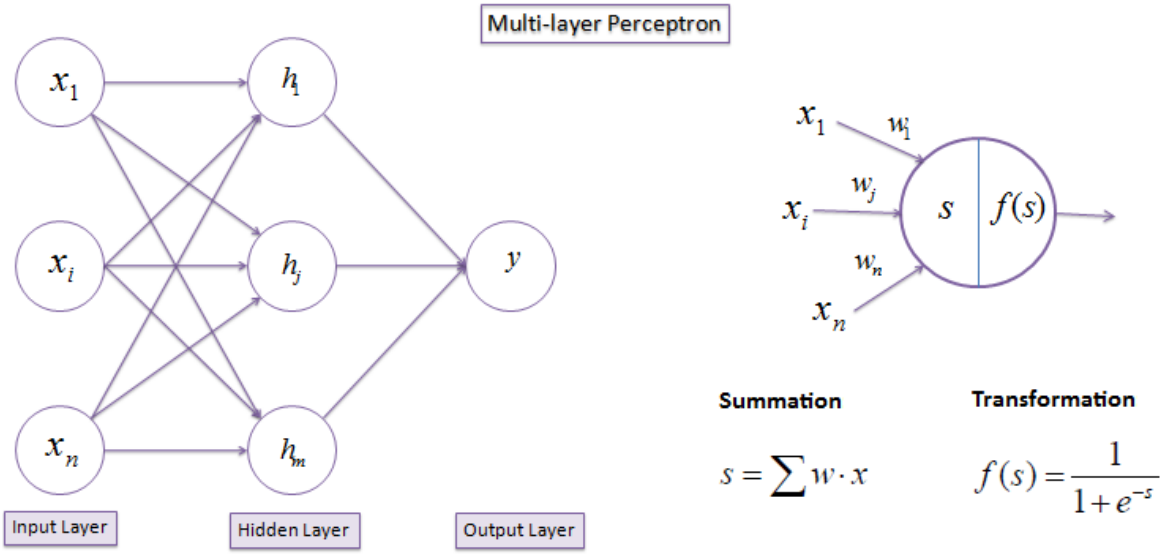
However, a multi-layer perceptron using the backpropagation algorithm can successfully classify the XOR data.

Multi-layer Perceptron - Backpropagation algorithm

- A multi-layer perceptron (MLP) has the same structure of a single layer perceptron with one or more hidden layers. The backpropagation algorithm consists of two phases: the forward phase where the activations are propagated from the input to the output layer, and the backward phase, where the error between the observed actual and the requested nominal value in the output layer is propagated backwards in order to modify the weights and bias values.
- Backpropagation algorithm calculates the gradient of the error function. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.
- The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation.

Forward propagation:

Propagate inputs by adding all the weighted inputs and then computing outputs using sigmoid threshold.



Backward propagation:

Propagates the errors backward by apportioning them to each unit according to the amount of this error the unit is responsible for.

1. Error in any ***output*** neuron

$$d_o = y \times (1-y) \times (t-y)$$

2. Error in any ***hidden*** neuron

$$d_i = y_i \times (1-y_i) \times (w_i \times d_o)$$

3. Change the ***weights***

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

Feed-forward Neural Networks

In a feed-forward network, signals can only move in one direction. These networks are considered non-recurrent network with inputs, outputs, and hidden layers. A layer of processing units receives input data and executes calculations there. Based on a weighted total of its inputs, each processing element performs its computation. The newly derived values are subsequently used as the new input values for the subsequent layer. This process continues until the output has been determined after going through all the layers.

Perceptron (linear and non-linear) and Radial Basis Function networks are examples of feed-forward networks. In fact, a single-layer perceptron network is the most basic type of neural network. It has a single layer of output nodes, and the inputs are fed directly into the outputs via a set of weights. Each node calculates the total of the products of the weights and the inputs. This neural network structure was one of the first and most basic architectures to be built.

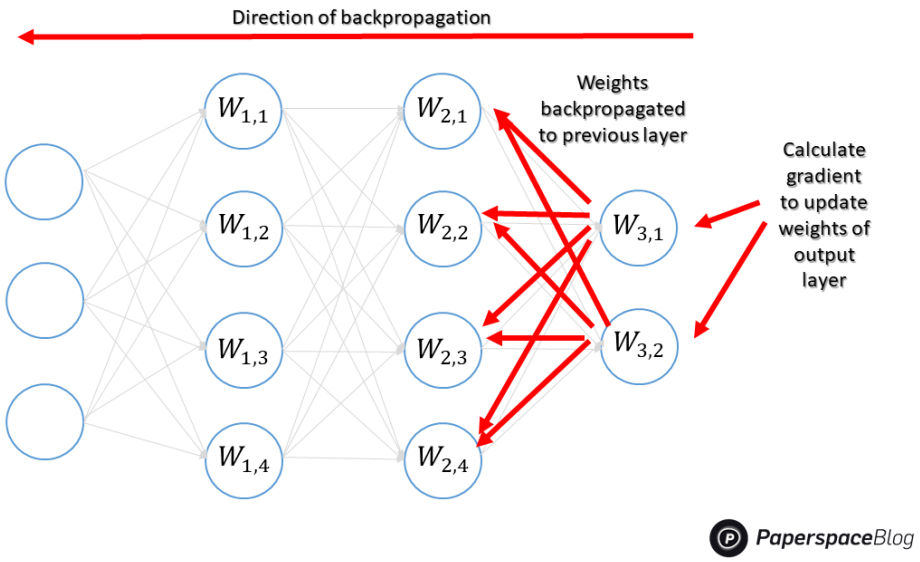
Learning is carried out on a multi-layer feed-forward neural network using the back-propagation technique. The properties generated for each training sample are stimulated by the inputs. The hidden layer is simultaneously fed the weighted outputs of the input layer. The weighted output of the hidden layer can be used as input for additional hidden layers, etc. The employment of many hidden layers is arbitrary; often, just one is employed for basic networks.

Convolution neural networks (CNNs) are one of the most well-known iterations of the feed-forward architecture. They offer a more scalable technique to image classification and object recognition tasks by using concepts from linear algebra, specifically matrix multiplication, to identify patterns within an image.

How a Feed-forward Neural Network is trained?

The typical algorithm for this type of network is **back-propagation**. It is a technique for adjusting a neural network's weights based on the error rate recorded in the previous epoch (i.e., iteration). By properly adjusting the weights, you may lower error rates and improve the model's reliability by broadening its applicability.

The gradient of the loss function for a single weight is calculated by the neural network's back propagation algorithm using the chain rule. In contrast to a native direct calculation, it efficiently computes one layer at a time. Although it computes the gradient, it does not specify how the gradient should be applied. It broadens the scope of the delta rule's computation.



Structure of Feedback Neural Networks:

A feed-back network, such as a **recurrent neural network (RNN)**, features feed-back paths, which allow signals to use loops to travel in both directions. Neuronal connections can be made in any way. Since this kind of network contains loops, it transforms into a non-linear dynamic system that evolves during training continually until it achieves an equilibrium state.

In research, RNN are the most prominent type of feed-back networks. They are an artificial neural network that forms connections between nodes into a directed or undirected graph along a temporal sequence. It can display temporal dynamic behavior as a result of this. RNNs may process input sequences of different lengths by using their internal state, which can represent a form of memory. They can therefore be used for applications like speech recognition or handwriting recognition.

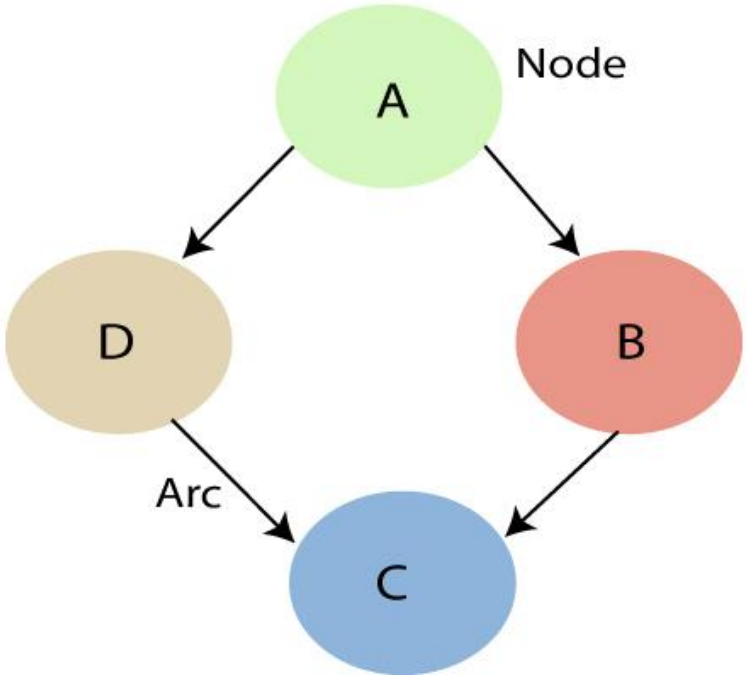
	Convolution Neural Networks (CNNs)	**Recurrent Neural Networks (RNNs)**
Architecture	Feed-forward neural network	Feed-back neural network
Layout	Multiple layers of nodes including convolutional layers	Information flows in different directions, simulating a memory effect
Data type	Image data	Sequence data
Input/Output	The size of the input and output are fixed (i.e input image with fixed size and outputs the classification)	The size of the input and output may vary (i.e receiving different texts and generating different translations for example)
Use cases	Image classification, recognition, medical imagery, image analysis, face detection	Text translation, natural language processing, language translation, sentiment analysis
Drawbacks	Large training data	Slow and complex training procedures
Description	CNN employs neuronal connection patterns. And, they are inspired by the arrangement of the individual neurons in the animal visual cortex, which allows them to respond to overlapping areas of the visual field.	Time-series information is used by recurrent neural networks. For instance, a user's previous words could influence the model prediction on what he can says next

Bayesian Belief Network

- Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

- It is also called a **Bayes network, belief network, decision network, or Bayesian model**.
- Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.
- Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.
- Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:
    - **Directed Acyclic Graph**
    - **Table of conditional probabilities.**
- The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.
- **A Bayesian network graph is made up of nodes and Arcs (directed links), where:**



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
  These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
    - **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**

    - **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
    - **Node C is independent of node A.**
- **Note: The Bayesian network graph does not contain any cyclic graph. Hence, it is known as a directed acyclic graph or DAG.**
- The Bayesian network has mainly two components:
    - Causal Component
    - Actual numbers
- Each node in the Bayesian network has condition probability distribution **P(Xi |Parent(Xi))**, which determines the effect of the parent on that node.
- Bayesian network is based on Joint probability distribution and conditional probability.