VISHNU INSTITUTE OF TECHNOLOGY :: BHIMAVARAM

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

UNIT 1

Topics :

1. **Introduction to Artificial Intelligence**
2. **Foundations of Artificial Intelligence**
3. **History of Artificial Intelligence**
4. **The state of the Art**
5. **Agents and Environments**
6. **Good Behavior : The concept of Rationality**
7. **The Nature of Environments**
8. **Structure of Agents**

Introduction to Artificial Intelligence

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. It is also 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 sub fields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.

It is a branch of Computer Science that pursues creating the computers or machines as intelligent as human beings. It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human

intelligence, but AI does not have to confine itself to methods that are biologically observable.

Definition: Artificial Intelligence is the study of how to make computers do things, which, at the moment, people do better.

According to the father of Artificial Intelligence, John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs". **Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think.**

AI is accomplished by studying how human brain thinks and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems. It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting.

AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.

From a business perspective AI is a set of very powerful tools, and methodologies for using those tools to solve business problems. From a programming perspective, AI includes the study of symbolic programming, problem solving, and search.

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

In Figure 1.1 we see eight definitions of AI, laid out along two dimensions. The definitions on top are concerned with thought processes and reasoning, whereas the ones on the bottom address behavior. The definitions on the left measure success in terms of fidelity to human performance, whereas RATIONALITY the ones on the right, measure against an ideal performance measure, called rationality. A system is rational if it does the "right thing," given what it knows.

Historically, all four approaches to AI have been followed, each by different people with different methods. A human-centered approach must be in part an empirical science, involving observations and hypotheses about human behavior. A rationalist approach involves a combination of mathematics and engineering.

1. **Acting Humanly : The Turing Test Approach**

The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence. A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.we note that programming a computer to pass a rigorously applied test provides

plenty to work on. The computer would need to possess the following capabilities:

- Natural Language Processing
- Knowledge Representation
- Automated Reasoning
- Machine Learning

2. **Thinking humanly: The cognitive modelling approach**

If we are going to say that a given program thinks like a human, we must have some way of determining how humans think. We need to get *inside* the actual workings of human minds. There are two ways to do this: through introspection--trying to catch our own thoughts as they go by--or through psychological experiments. Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input/output and timing behavior matches human behavior, that is evidence that some of the program's mechanisms may also be operating in humans.

3. Thinking Rationally : The laws of Thought Approach

The Greek philosopher Aristotle was one of the first to attempt to codify ``right thinking,'' that is, irrefutable reasoning processes. His famous **syllogisms** provided patterns for argument structures that always gave correct conclusions given correct premises. For example, ``Socrates is a man; all men are mortal; therefore Socrates is mortal.'' These laws of thought were supposed to govern the operation of the mind, and initiated the field of **logic**.

4. **Acting Rationally : The rational agent approach**

Acting rationally means acting so as to achieve one's goals, given one's beliefs. An **agent** is just something that perceives and acts. (This may be an unusual use of the word, but you will get used

to it.) In this approach, AI is viewed as the study and construction of rational agents.

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:

1. 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.
2. With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
3. With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
4. 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.

Foundations of Artificial Intelligence

1. Philosophy : AI in philosophy is mainly concerned with the following :

- Can formal rules be used to draw valid conclusions?
- How does the mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

Aristotle was the first to formulate a precise set of laws governing the rational part of the mind. He developed an informal system of syllogisms for proper reasoning, which in principle allowed one to generate conclusions mechanically, given initial premises.

2. Mathematics : Mathematical modeling is the activity devoted to the study of the simulation of physical phenomena by computational processes. The goal of the simulation is to predict the behavior of some artifact within its environment. The AI in Mathematics is mainly concerned with :

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

Philosophers staked out most of the important ideas of AI, but the leap to a formal science required a level of mathematical formalization in three fundamental areas: logic, computation, and probability.The first nontrivial algorithm is thought to be Euclid's algorithm for computing greatest common denominators. The study of algorithms as objects in themselves goes back to al-Khowarazmi, a Persian mathematician of the 9th century, whose writings also introduced Arabic numerals and algebra to Europe.

3. Economics :

Most people think of economics as being about money, but economists will say that they are really studying how people make choices that lead to preferred outcomes or utility. **Decision theory**, which combines probability theory with utility theory, provides a formal and complete

framework for decisions made under uncertainty that is, in cases where probabilistic descriptions appropriately capture the decision-maker's environment. This is suitable for "large" economies. For "small" economies, the situation is much more like a game: the actions of one player can significantly affect the utility of another. Von Neumann and Morgenstern's development of **Game Theory** included the surprising result that, for some games, a rational agent should act in a random fashion, or at least in a way that appears random to the adversaries.

In the field of operations research, which emerged in World War II from efforts in Britain to optimize radar installations, and later found civilian applications in complex management decisions. The work of Richard Bellman (1957) formalized a class of sequential decision problems called Markov decision processes

4. Neuroscience :

Neuroscience is the study of the nervous system, particularly the brain. The exact way in which the brain enables thought is one of the great mysteries of science.Brain is recognized as the seat of consciousness.The truly amazing conclusion is that a collection of simple cells can lead to thought, action, and consciousness or, in other words, that brains cause minds.Brains and computers perform quite different tasks and have different properties. Moore's Law predicts that the CPU's gate count will equal the brain's neuron count around 2020.

Human brain capacity doubles roughly every 2 to 4 million years. Even though a computer is a million times faster in raw switching speed, the brain ends up being 100,000 times faster at what it does.

5. Psychology :

The view of the brain as an information-processing device, which is a principal characteristic of cognitive psychology, can be traced back at least to the works of William James. The development of computer modeling led to

the creation of the field of cognitive science. The field can be said to have started at a workshop in September 1956 at MIT. This is just two months after the conference at which **AI itself was "born."**

6. Linguistics :

Modem linguistics and AI were born at about the same time, and grew up together, intersecting in a hybrid field called **computational linguistics** or **Natural Language Processing**. The problem of understanding language soon turned out to be considerably more complex than it seemed in 1957. Understanding language requires an understanding of the subject matter and context, not just an understanding of the structure of sentences.

History of Artificial Intelligence

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

Gestation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.
- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

The birth of Artificial Intelligence (1952-1956)

- **Year 1955:** An Allen Ne-well and Herbert A. Simon created the "first artificial intelligence program"Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field. At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden years-Early enthusiasm (1956-1974)

- **Year 1966:** The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizmann created the first chat bot in 1966, which was named as ELIZA.
- **Year 1972:** The first intelligent humanoid robot was built in Japan which was named as WABOT-1.

The first AI winter (1974-1980): The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches. During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

- **Year 1980:** After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.
- In the Year 1980, the first national conference of the American Association of Artificial Intelligence **was held at Stanford University**.

The second AI winter (1987-1993)

- The duration between the years 1987 to 1993 was the second AI Winter duration.Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

The emergence of intelligent agents (1993-2011)

- **Year 1997:** In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
- **Year 2002:** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
- **Year 2006:** AI came in the Business world till the year 2006. Companies
- like Facebook, Twitter, and Netflix also started using AI.

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

- **Year 2011:** In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
- **Year 2014:** In the year 2014, Chat bot "Eugene Goostman" won a competition in the infamous "Turing test."
- **Year 2018:** The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well. Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.
- Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom.

Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

The State of the Art

1. Autonomous planning and scheduling : A hundred million miles from Earth, NASA's Remote Agent Program became the first onboard autonomous planning program. It control the scheduling of operations for a spacecraft Remote Agent generated plans from high-level goals specified from the ground, and it monitored the operation of the spacecraft as the plans were executed-detecting, diagnosing, and recovering from problems as they occurred.

2. Game playing : IBM's Deep Blue became the first computer program to defeat the world champion in a chess match when it bested Garry Kasparov by a score of 3.5 to 2.5 in an exhibition match. Kasparov said that he felt a "new kind of intelligence" across the board from him. Newsweek magazine described the match as "The brain's last stand." The value of IBM's stock increased by $18 billion.

3. Autonomous control : The ALVINN computer vision system was trained to steer a car to keep it following a lane. Video cameras that transmit road images to ALVINN, which then computes the best direction to steer, based on experience from previous training runs.

4. Diagnosis : Medical diagnosis programs based on probabilistic analysis have been able to perform at the level of an expert physician in several areas of medicine. Heckerman (1991) describes a case where a leading expert on lymph-node pathology scoffs at a program's diagnosis of an especially difficult case. The creators of the program suggest he ask the computer for an explanation of the diagnosis. The machine points out the major factors

influencing its decision and explains the subtle interaction of several of the symptoms in this case. Eventually, the expert agrees with the program.

5. Logistics Planning : During the Persian Gulf crisis of 1991, U.S. forces deployed a Dynamic Analysis and Re-planning Tool, DART (Cross and Walker, 1994), to do automated logistics planning and scheduling for transportation. This involved up to 50,000 vehicles, cargo, and people at a time, and had to account for starting points, destinations, routes, and conflict resolution among all parameters. The AI planning techniques allowed a plan to be generated in hours that would have taken weeks with older methods. The Defense Advanced Research Project Agency (DARPA) stated that this single application more than paid back DARPA's 30-year investment in AI.

6. Robotics : Many surgeons now use robot assistants in microsurgery. Computer vision techniques are used to create a three-dimensional model of a patient's internal anatomy and then uses robotic control to guide the insertion of a hip replacement prosthesis.

7. Language understanding and problem solving : PROVERB is a computer program that solves crossword puzzles better than most humans, using constraints on possible word fillers, a large database of past puzzles, and a variety of information sources including dictionaries and online databases such as a list of movies and the actors that appear in them.

Agents and Environments

An "agent" is an independent program or entity that interacts with its environment by perceiving its surroundings via sensors, then acting through actuators.Agents use their actuators to run through a cycle of perception, thought, and action. Examples of agents in general terms include:

a. **Software:** This Agent has file contents, keystrokes, and received network packages that function as sensory input, then act on those inputs, displaying the output on a screen.

b. Human: Yes, we're all agents. Humans have eyes, ears, and other organs that act as sensors, and hands, legs, mouths, and other body parts act as actuators.

c. Robotic: Robotic agents have cameras and infrared range finders that act as sensors, and various servos and motors perform as actuators.

Intelligent agents in AI are autonomous entities that act upon an environment using sensors and actuators to achieve their goals. In addition, intelligent agents may learn from the environment to achieve those goals. Driver less cars and the Siri virtual assistant are examples of intelligent agents in AI.There are five different types of intelligent agents used in AI. They are defined by their range of capabilities and intelligence level:

- ✓ **Simple Reflex Agents:** These agents work here and now and ignore the past. They respond using the event-condition-action rule. The ECA rule applies when a user initiates an event, and the Agent turns to a list of preset conditions and rules, resulting in pre-programmed outcomes.
- ✓ **Model-based Agents:** These agents choose their actions like reflex agents do, but they have a better comprehensive view of the environment. An environmental model is programmed into the internal system, incorporating into the Agent's history.
- ✓ **Goal-based agents:** These agents build on the information that a model-based agent stores by augmenting it with goal information or data regarding desirable outcomes and situations.
- ✓ **Utility-based agents:** These are comparable to the goal-based agents, except they offer an extra utility measurement. This measurement rates each possible scenario based on the desired result and selects the action

that maximizes the outcome. Rating criteria examples include variables such as success probability or the number of resources required.

- ✓ **Learning agents:** These agents employ an additional learning element to gradually improve and become more knowledgeable over time about an environment. The learning element uses feedback to decide how the performance elements should be gradually changed to show improvement.

Agent Terminology

1. **Performance Measure of Agent** − It is the criteria, which determines how successful an agent is.
2. **Behavior of Agent** − It is the action that agent performs after any given sequence of precepts.
3. **Percept** − It is agent's perceptual inputs at a given instance.
4. **Percept Sequence** − It is the history of all that an agent has perceived till date.
5. **Agent Function** − It is a map from the precept sequence to an action.

Good Behavior : The concept of Rationality

Rationality is nothing but status of being reasonable, sensible, and having good sense of judgment. Rationality is concerned with expected actions and results depending upon what the agent has perceived. Performing actions with the aim of obtaining useful information is an important part of rationality.

What is Ideal Rational Agent?

An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of −

- Its percept sequence
- Its built-in knowledge base

Rationality of an agent depends on the following :

a. The **performance measures**, which determine the degree of success.
b. Agent's **Percept Sequence** till now.
c. The agent's **prior knowledge about the environment**.
d. The **actions** that the agent can carry out.

A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept

sequence. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).

The Nature of Environments

Some programs operate in the entirely **artificial environment** confined to keyboard input, database, computer file systems and character output on a screen.

In contrast, some software agents (software robots or soft bots) exist in rich, unlimited domains. The software agent needs to choose from a long array of actions in real time. A soft bot designed to scan the online preferences of the customer and show interesting items to the customer works in the **real** as well as an **artificial** environment.

The most famous **artificial environment** is the **Turing Test environment**, in which one real and other artificial agents are tested on equal ground. This is a very challenging environment as it is highly difficult for a software agent to perform as well as a human.

Properties of Environment

1. **Fully Observable / Partially Observable** − If it is possible to determine the complete state of the environment at each time point from the percepts it is observable; otherwise it is only partially observable.
2. **Single agent / Multiple agents** − The environment may contain other agents which may be of the same or different kind as that of the agent.
3. **Deterministic / Non-deterministic** − If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.
4. **Episodic / sequential** − In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. Subsequent episodes do not depend on the actions in

the previous episodes. Episodic environments are much simpler because the agent does not need to think ahead.

5. **Static / Dynamic** – If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.

6. **Discrete / Continuous** – If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete (For example, chess); otherwise it is continuous (For example, driving).

7. **Known/ Unknown** – If the agent's sensory apparatus can have access to the complete state of the environment, then the environment is accessible to that agent.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Figure 2.6 Examples of task environments and their characteristics.

PEAS System is used to categorize similar agents together. The PEAS system delivers the performance measure with respect to the environment, actuators, and sensors of the respective agent. Most of the highest performing agents are Rational Agents. PEAS stand for Performance, Environment , Actuators and Sensors. An example PEAS system of an automated taxi driver is shown in the figure below :

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Structure of Agents

Agent's structure can be viewed as the combination of architecture that the agent is using along with the program it has to execute.

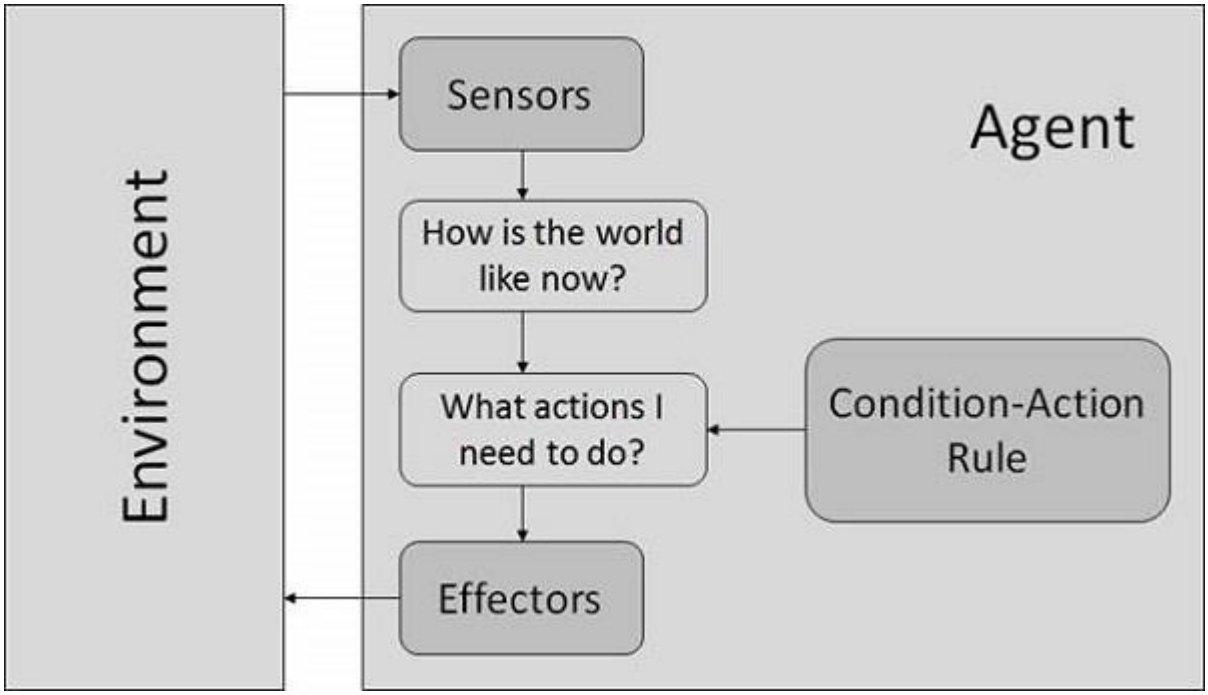
Agent = Architecture + Agent Program

Architecture is the machinery that an agent executes on and that of A**gent Program** is an implementation of an agent function.

Simple Reflex Agents

- They choose actions only based on the current percept.
- They are rational only if a correct decision is made only on the basis of current precept.
- Their environment is completely observable.

Condition-Action Rule – It is a rule that maps a state (condition) to an action.



Model Based Reflex Agents

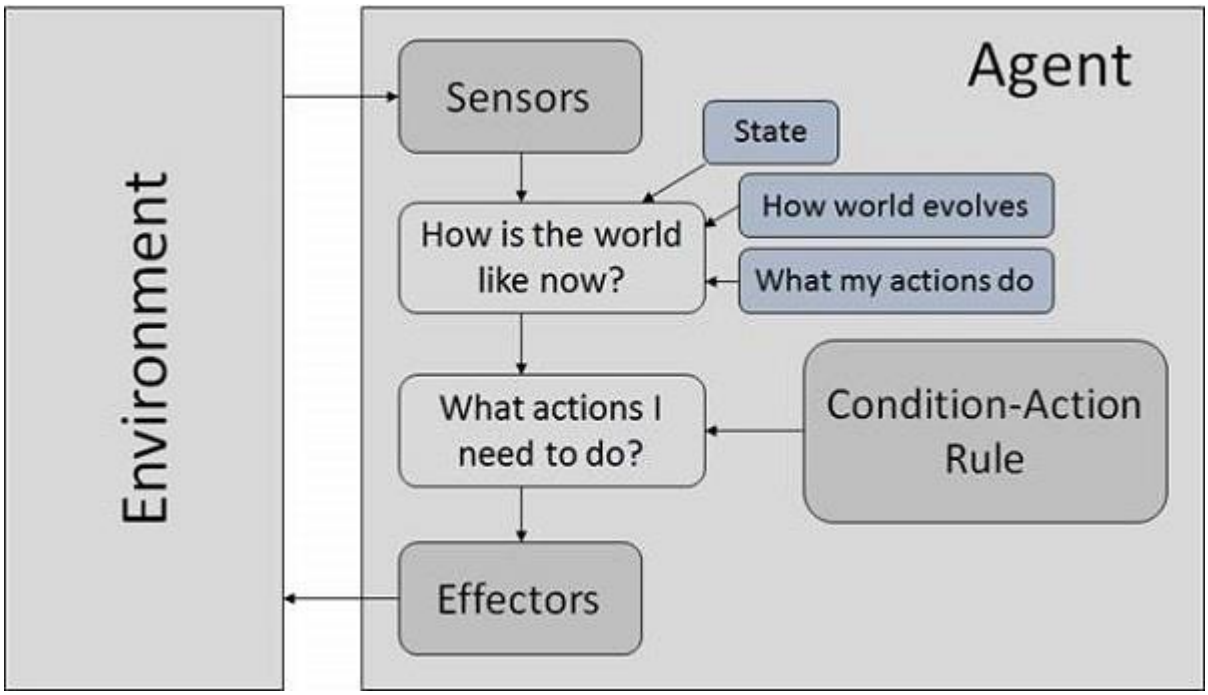
They use a model of the world to choose their actions. They maintain an internal state.

Model – knowledge about "how the things happen in the world".

Internal State – It is a representation of unobserved aspects of current state depending on percept history.

Updating the state requires the information about –

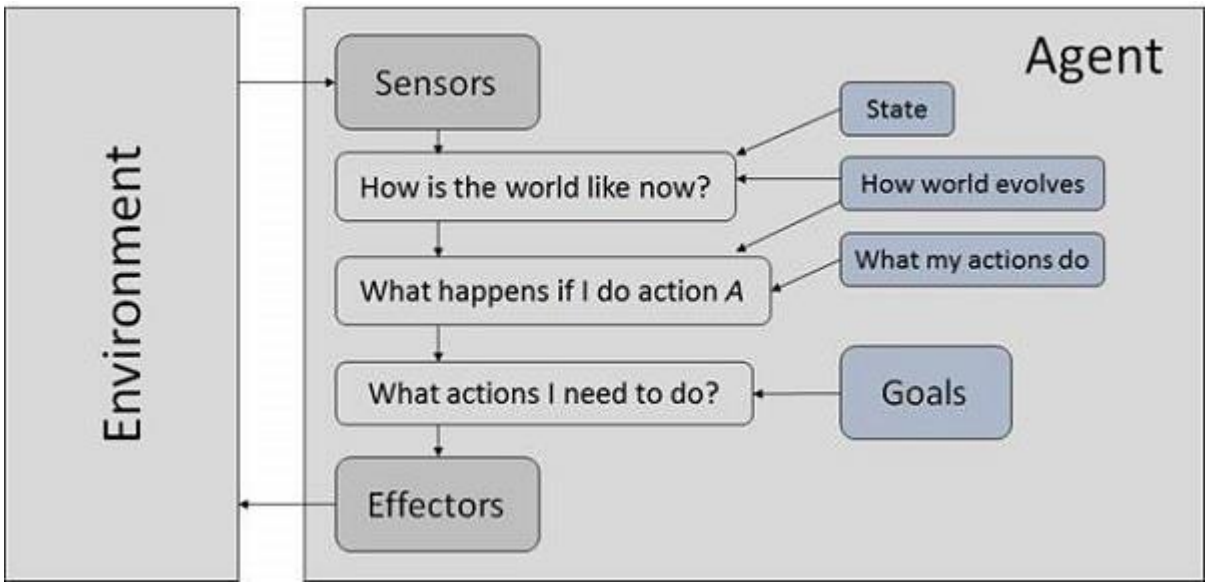
- How the world evolves.
- How the agent's actions affect the world.



Goal Based Agents

They choose their actions in order to achieve goals. Goal-based approach is more flexible than reflex agent since the knowledge supporting a decision is explicitly modeled, thereby allowing for modifications.

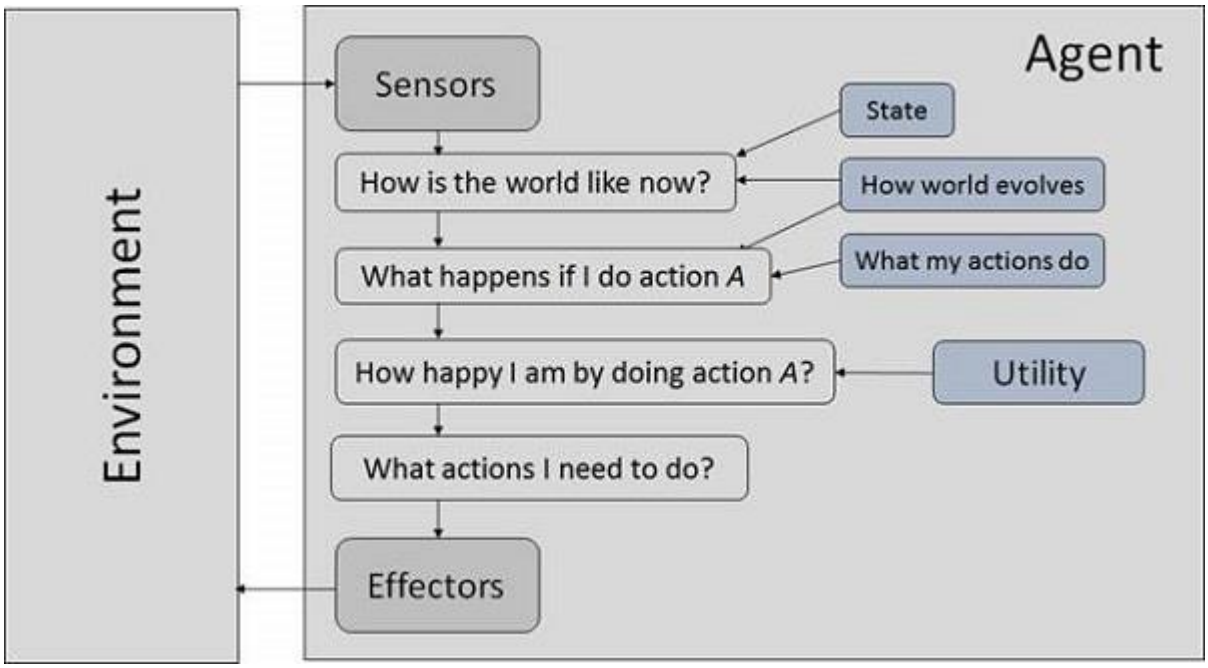
Goal – It is the description of desirable situations.



Utility Based Agents

They choose actions based on a preference (utility) for each state.

Goals are inadequate when there are conflicting goals, out of which only few can be achieved. Goals have some uncertainty of being achieved and you need to weigh likelihood of success against the importance of a goal.



Learning Agents

A learning agent can be divided into four conceptual components, as shown in figure below. The most important distinction is between the **learning element**, which is responsible for making improvements, and the p**erformance element**, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance CRITIC element should be modified to do better in the future.

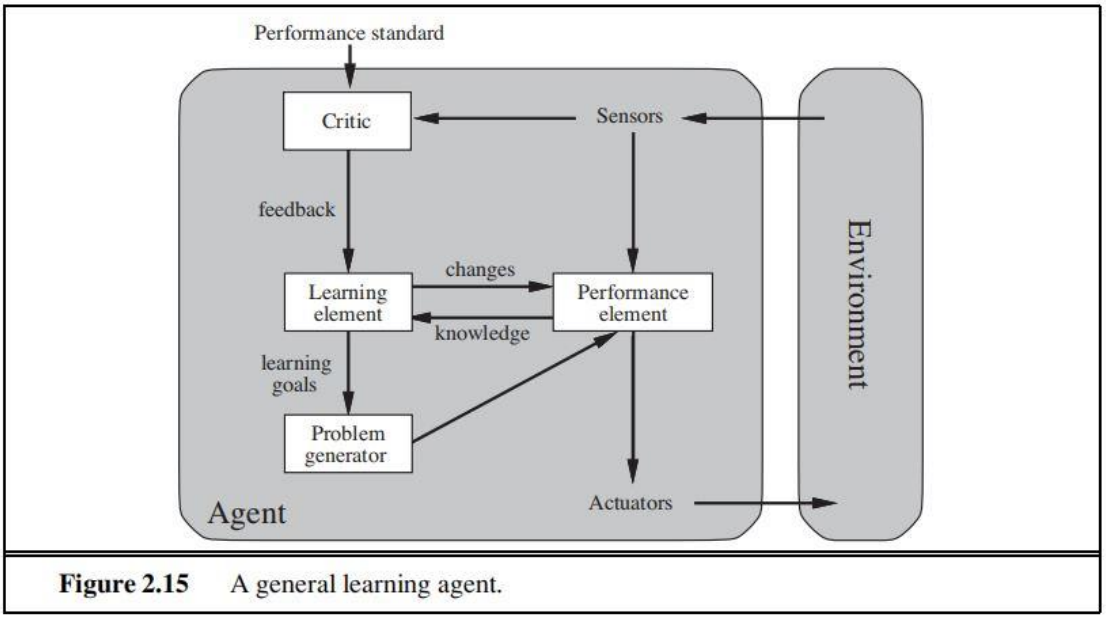


Figure 2.15 A general learning agent.

How the Components of Agent Programs work

The agent programs consists of various components that represent environment that the agent inhabits. To draw user's attention the components will make various representations about the environments that the agent inhabits. In general,we can place the representations along an axis of increasing complexity and expressive power—**atomic**, **factored**, and **structured** as shown in the below figure :

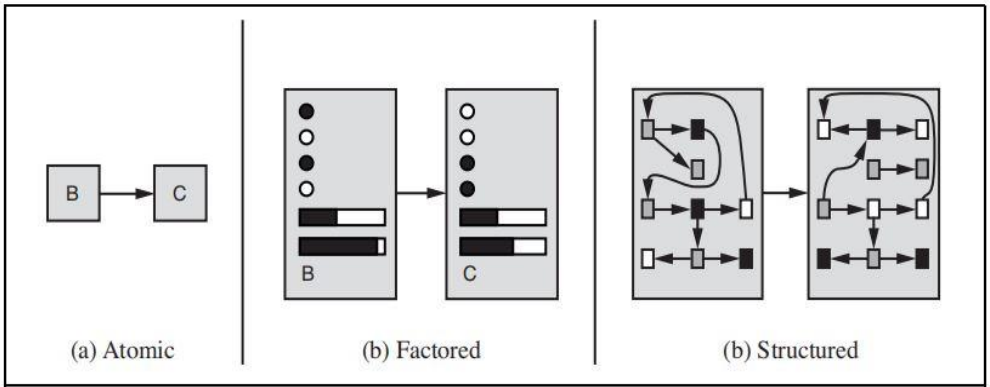


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

In an **atomic representation** each state of the world is indivisible—it has no internal structure.A single atom of knowledge called a "black box" whose only

discernible property is that of being identical to or different from another black box is visible in the structure. The algorithms underlying **search** and **game-playing, Hidden Markov models** , and **Markov decision processes** all work with atomic representations—or, at least, they treat representations *as if* they were atomic.

A **factored representation** splits up each state into a fixed set of **variables** or **attributes**, each of which can have a **value**. Two different factored states can share some attributes (such as being at some particular GPS location) and not others (such as having lots of gas or having no gas); this makes it much easier to work out how to turn one state into another. With factored representations, we can also represent *uncertainty*—for example, ignorance about the amount of gas in the tank can be represented by leaving that attribute blank. Many important areas of AI are based on factored representations, including **constraint satisfaction** algorithms, **propositional logic**, **planning, Bayesian networks**, and the **machine learning** algorithms..

structured representation, in which objects and their various and varying relationships can be described explicitly.Structured representations underlie **relational databases** and **first-order logic**,**first-order probability models**, **knowledge-based learning** and much of **natural language understanding**. In fact, almost everything that humans express in natural language concerns objects and their relationships.

VISHNU INSTITUTE OF TECHNOLOGY :: BHIMAVARAM

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

UNIT 2

Topics :

1. **Problem Solving Agents**
2. **Example Problems**
    a) **Two State Vaccum Cleaner**
    b) **8 Queens**
    c) **8 Puzzle**
3. **Real World Problems**
    a) **Route Finding Problems**
    b) **Touring Problem**
    c) **Travelling Salesman Problem**
    d) **VLSI Layout Problem**
4. **Searching for Solutions**
5. **Uninformed Search Strategies**
    a) **Breadth First Search**
    b) **Depth First Search**
    c) **Uniform Cost Search**
    d) **Depth Limited Search**
    e) **Iterative Deepening Depth Limited Search**
    f) **Bidirectional Search**
6. **Informed Search Strategies**
    a) **Best First Search**
    b) **A* Search**
7. **Local Search Algorithms and Optimization Problems**
8. **Searching with Non Deterministic Actions**

Problem Solving Agents

Problem Solving Agents decide what to do by finding a sequence of actions that leads to a desirable state or solution. An agent may need to plan when the best course of action is not immediately visible. They may need to think through a series of moves that will lead them to their goal state. Such an agent is known as a **problem solving agent**, and the computation it does is known as a **search**.Intelligent agents are supposed to maximize its **performance measure**. Achieving this can be simplified if the agent can adopt a **goal** and aim to satisfy it.

Setting goals help the agent organize its behavior by limiting the objectives that the agent is trying to achieve and hence the actions it needs to consider. This **Goal formulation** based on the current situation and the agent's performance measure is the first step in problem solving.We consider the agent's goal to be a set of states. The agent's task is to find out actions in the present and in the future that could reach the goal state from the present state.

The problem solving agent follows this four phase problem solving process:

1. **Goal Formulation:** This is the first and most basic phase in problem solving. It arranges specific steps to establish a target/goal that demands some activity to reach it. AI agents are now used to formulate goals.
2. **Problem Formulation:** It is one of the fundamental steps in problem-solving that determines what action should be taken to reach the goal.
3. **Search:** After the Goal and Problem Formulation, the agent simulates sequences of actions and has to look for a sequence of actions that reaches the goal. This process is called **search,** and the sequence is called a **solution**. The agent might have to simulate multiple sequences that do not reach the goal, but eventually, it will find a

solution, or it will find that no solution is possible. A search algorithm takes a problem as input and outputs a sequence of actions.

4. **Execution:** After the search phase, the agent can now execute the actions that are recommended by the search algorithm, one at a time. This final stage is known as the execution phase.

Formulating the Problem : Before we move into the problem formulation phase, we must first define a problem in terms of problem solving agents. A formal definition of a problem consists of five components:

1. Initial State
2. Actions
3. Transition Model
4. Goal Test
5. Path Cost

Initial State : It is the agent's starting state or initial step towards its goal. For example, if a taxi agent needs to travel to a location(B), but the taxi is already at location(A), the problem's initial state would be the location (A).

Actions : It is a description of the possible actions that the agent can take. Given a state s, Actions(s) returns the actions that can be executed in s. Each of these actions is said to be appropriate in s.

Transition Model : It describes what each action does. It is specified by a function Result(s, a) that returns the state that results from doing action an in states.

The initial state, actions, and transition model together define the **state space** of a problem, a set of all states reachable from the initial state by any sequence of actions. The state space forms a graph in which the nodes are states, and the links between the nodes are actions.

Goal Test : It determines if the given state is a goal state. Sometimes there is an explicit list of potential goal states, and the test merely verifies whether the provided state is one of them. The goal is sometimes expressed via an abstract attribute rather than an explicitly enumerated set of conditions.

Path Cost : It assigns a numerical cost to each path that leads to the goal. The problem solving agents choose a cost function that matches its performance measure. Remember that the optimal solution has the lowest path cost of all the solutions.

After **Goal formulation** and **problem formulation**, the agent has to look for a sequence of actions that reaches the goal. This process is called **Search**. A search algorithm takes a problem as input and returns a sequence of actions as output. After the search phase, the agent has to carry out the actions that are recommended by the search algorithm. This final phase is called **execution** phase.

Formulate --> Search --> Execute

Thus the agent has a formulate, search and execute design to it.

Example Problems

The problem solving approach has been used in a wide range of work contexts. There are tw0o kinds of problem approaches

1. **Standardized/ Toy Problem:** Its purpose is to demonstrate or practice various problem solving techniques. It can be described concisely and precisely, making it appropriate as a benchmark for academics to compare the performance of algorithms.

2. **Real-world Problems:** It is real-world problems that need solutions. It does not rely on descriptions, unlike a toy problem, yet we can have a basic description of the issue.

The **solution** to the problem is an action sequence that leads from initial state to goal state and the solution quality is measured by the path cost function. An optimal solution has the lowest path cost among all the solutions.

An Example Toy Problem 1: Vaccum World Problem

There is a vacuum cleaner agent and it can move left or right and its jump is to suck up the dirt from the floor.

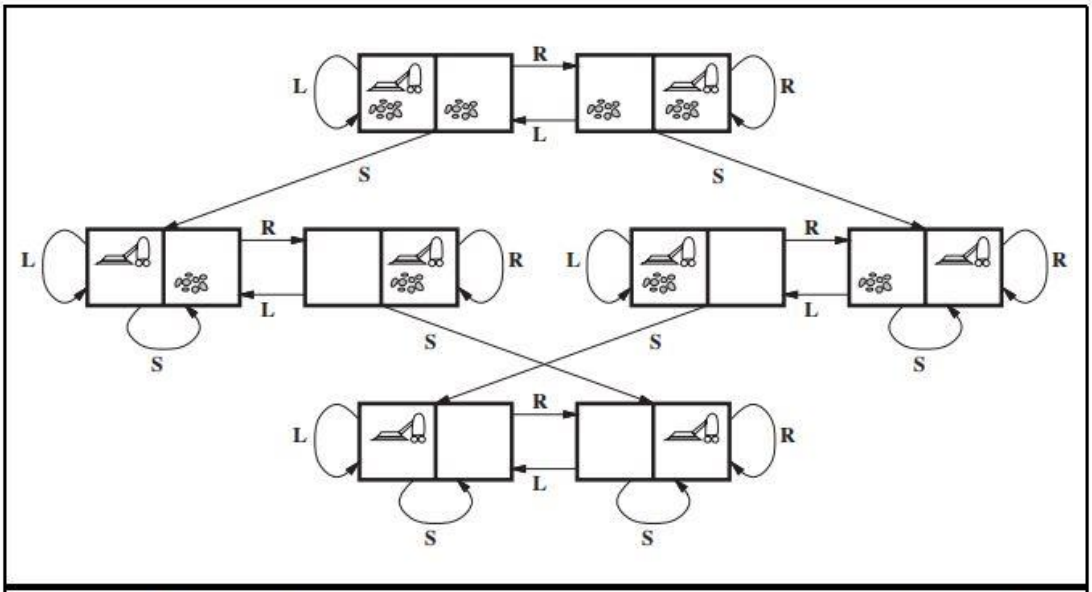


Figure 3.3 The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

The problem for vacuum world can be formulated as follows:

States: The state is determined by both the agent location and the dirt location. The agent is in one of two locations, each of which might or might not contain dirt. Therefore, there are 2 x 2^2 **= 8** possible world states.

A larger environment would have n x 2 to the power of n states.

Initial State: Any state can be assigned as the initial state in this case.

Action: In this environment there are three actions, *Move Left* , *Move Right* , *Suck up the dirt*.

Transition Model: All the actions have expected effects, except for when the agent is in leftmost square and the action is *Left*, when the agent is in rightmost square and the action is *Right* and the square is clean when the action is to *Suck*.

Goal Test: Goal test checks whether all the squares are clean.

Path Cost: Each step costs 1, so the path cost is the number of steps in the path.

The vacuum world problem is a toy problem and involves only discrete locations, discrete dirt etc. Therefore, this problem is a **Toy Problem**. There are many **Real-World Problems** like the automated taxi world. Try to formulate problems of real world and see what would be the states be and what actions could be chosen etc.

An Example Toy Problem 2: 8 Puzzle Problem

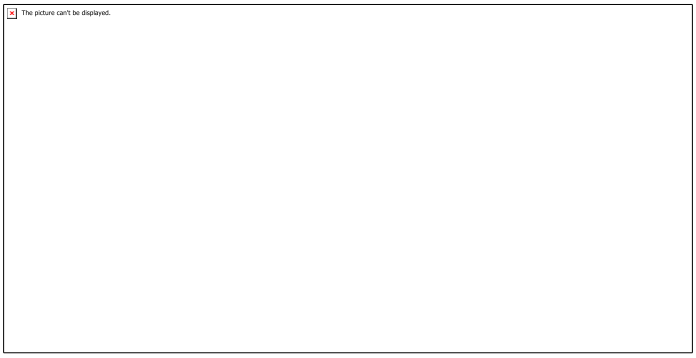
In a **sliding-tile puzzle**, a number of tiles (sometimes called blocks or pieces) are arranged in a grid with one or more blank spaces so that some of the tiles can slide into the blank space. One variant is the Rush Hour puzzle, in which cars and trucks slide around a 6 x 6 grid in an attempt to free a car from the traffic jam. Perhaps the best-known variant is the **8- puzzle** (see Figure below), which consists of a 3 x 3 grid with eight numbered tiles and one blank space, and the **15-puzzle** on a 4 x 4 grid. The object is to reach a specified goal state, such as the one shown on the right of the figure. The standard formulation of the 8 puzzles is as follows:

STATES: A state description specifies the location of each of the tiles.

INITIAL STATE: Any state can be designated as the initial state. (Note that a parity property partitions the state space—any given goal can be reached from exactly half of the possible initial states.)

ACTIONS: While in the physical world it is a tile that slides, the simplest way of describing action is to think of the blank space moving **Left**, **Right**, **Up**, or **Down**. If the blank is at an edge or corner then not all actions will be applicable.

TRANSITION MODEL: Maps a state and action to a resulting state; for example, if we apply Left to the start state in the Figure below, the resulting state has the 5 and the blank switched.



A typical instance of the 8-puzzle

GOAL STATE: It identifies whether we have reached the correct goal state. Although any state could be the goal, we typically specify a state with the numbers in order, as in the Figure above.

ACTION COST: Each action costs 1.

An Example Toy Problem 3: 8 Queens Problem

The 8-queens problem can be defined as follows: Place 8 queens on an (8 by 8) chess board such that none of the queens attacks any of the others. A configuration of 8 queens on the board is shown in figure 1, but this does not represent a solution as the queen in the first column is on the same diagonal as the queen in the last column.

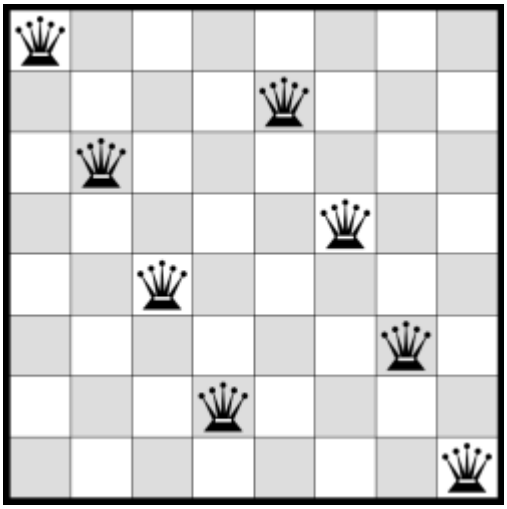


Figure 1: Almost a solution of the 8-queens problem

This problem can be solved by searching for a solution. The initial state is given by the empty chess board. Placing a queen on the board represents an action in the search problem. A goal state is a configuration where none of the queens attacks any of the others. Note that every goal state is reached after exactly 8 actions.

This formulation as a search problem can be improved when we realize that, in any solution, there must be exactly one queen in each of the columns. Thus, the possible actions can be restricted to placing a queen in the next column that does not yet contain a queen. This reduces the branching factor from (initially) 64 to 8.

Furthermore, we need only consider those rows in the next column that are not already attacked by a queen that was previously on the board. This is because the placing of further queens on the board can never remove the mutual attack and turn the configuration into a solution.

Real World Problems

1. Route Finding Problem : It is defined in terms of specified locations and transitions along links between them. Route-finding algorithms are used in a variety of applications. Some, such as Web sites and in-car systems that provide driving directions are relatively straightforward. Others, such as

routing video streams in computer networks, military operations planning, and airline travel-planning

systems, involve much more complex specifications. Consider the airline travel problems that must be solved by a travel-planning Web site:

- **States**: Each state obviously includes a location (e.g., an airport) and the current time. Furthermore, because the cost of an action (a flight segment) may depend on previous segments, their fare bases, and their status as domestic or international, the state must record extra information about these "historical" aspects.
- **Initial state**: This is specified by the user's query.
- **Actions**: Take any flight from the current location, in any seat class, leaving after the current time, leaving enough time for within-airport transfer if needed.
- **Transition model**: The state resulting from taking a flight will have the flight's destination as the current location and the flight's arrival time as the current time.
- **Goal test**: Are we at the final destination specified by the user?
- **Path cost**: This depends on monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer mileage awards, and so on.

2. Touring Problem : Touring problems are closely related to route-finding problems, but with an important difference. Consider, for example, the problem "Visit every city in Figure below, at least once, starting and ending in Bucharest." As with route finding, the actions correspond to trips between adjacent cities. The state space, however, is quite different. Each state must include not just the current location but also the *set of cities the agent has visited*. So the initial state would be In(Bucharest), Visited({Bucharest}), a typical intermediate state would be In(Vaslui), Visited({Bucharest, Urziceni,

Vaslui}), and the goal test would check whether the agent is in Bucharest and all 20 cities have been visited.

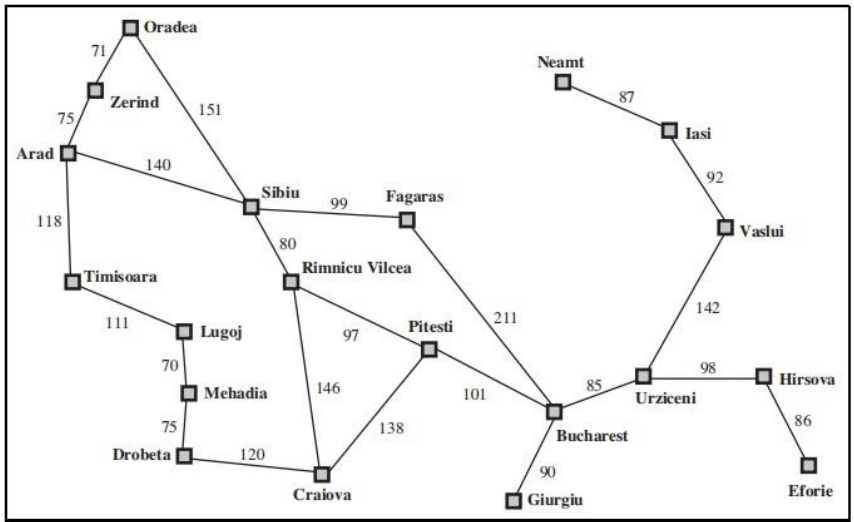


Figure 3.2 A simplified road map of part of Romania.

3. Travelling Salesman Problem : The **traveling salesperson problem** (TSP) is a touring problem in which each city must be visited exactly once. The aim is to find the *shortest* tour. The problem is known to be NP-hard, but an enormous amount of effort has been expended to improve the capabilities of TSP algorithms. In addition to planning trips for traveling salespersons, these algorithms have been used for tasks such as planning movements of automatic circuit-board drills and of stocking machines on shop floors

4. VLSI Layout Problem : A **VLSI layout** problem requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitance's, and maximize manufacturing yield. The layout problem comes after the logical design phase and is usually split into two parts: **cell layout** and **channel routing**. In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognized function. Each cell has a fixed footprint (size and shape) and requires a certain number of connections to each of the other cells. The aim is to place the cells on the chip so that they do not overlap and so that there is room for the connecting wires to be placed between the cells. Channel routing

finds a specific route for each wire through the gaps between the cells. These search problems are extremely complex, but definitely worth solving.

5. **Robot Navigation Problem : Robot Navigation is a** generalization of the route finding problem described earlier. Rather than following a discrete set of routes, a robot can move in a continuous space with an infinite set of possible actions and states. For a circular robot moving on a flat surface, the space is essentially two-dimensional. When the robot has arms and legs or wheels that must also be controlled, the search space becomes many-dimensional. Advanced techniques are required just to make the search space finite.

Searching for Solutions

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.

Terminologies:

- **Search:** Searching is 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:

1. 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.

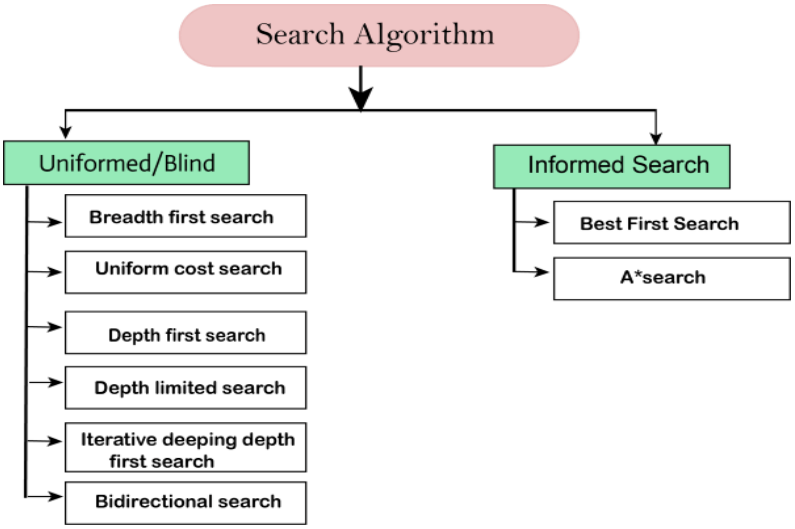
2. Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

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

4. 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.



******** **Notes on Uninformed Search Strategies and Informed Search Strategies are given as a Part-II Notes *******

Uninformed Search Strategies

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**

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.

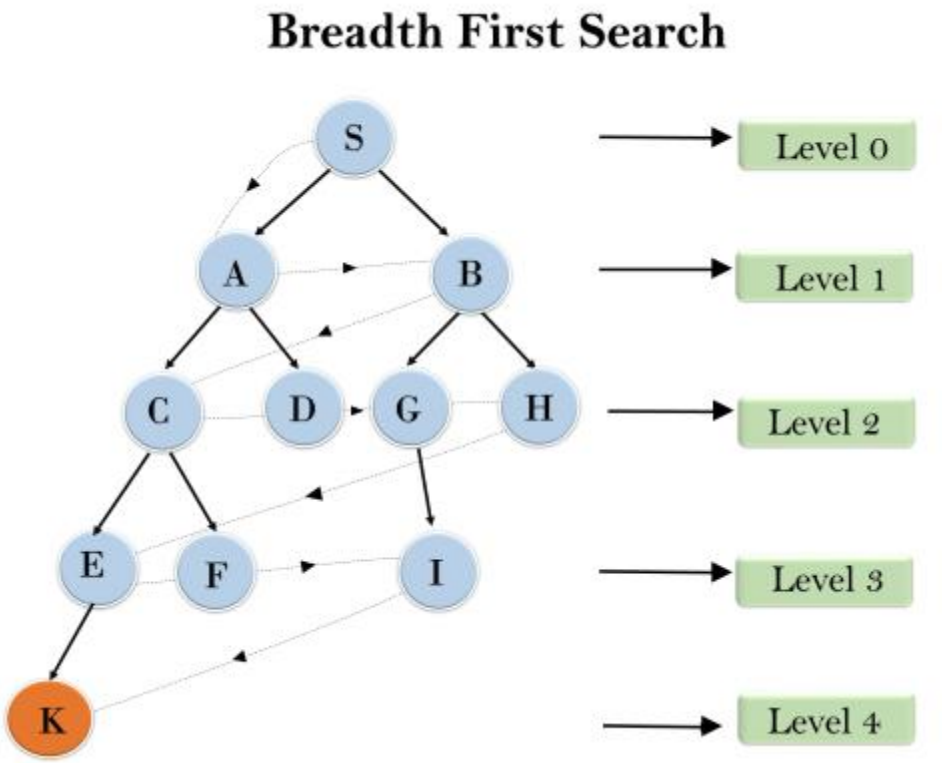
Disadvantages:

- 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.

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.

Depth-first Search

- Depth-first search is a 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.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.

- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantage:

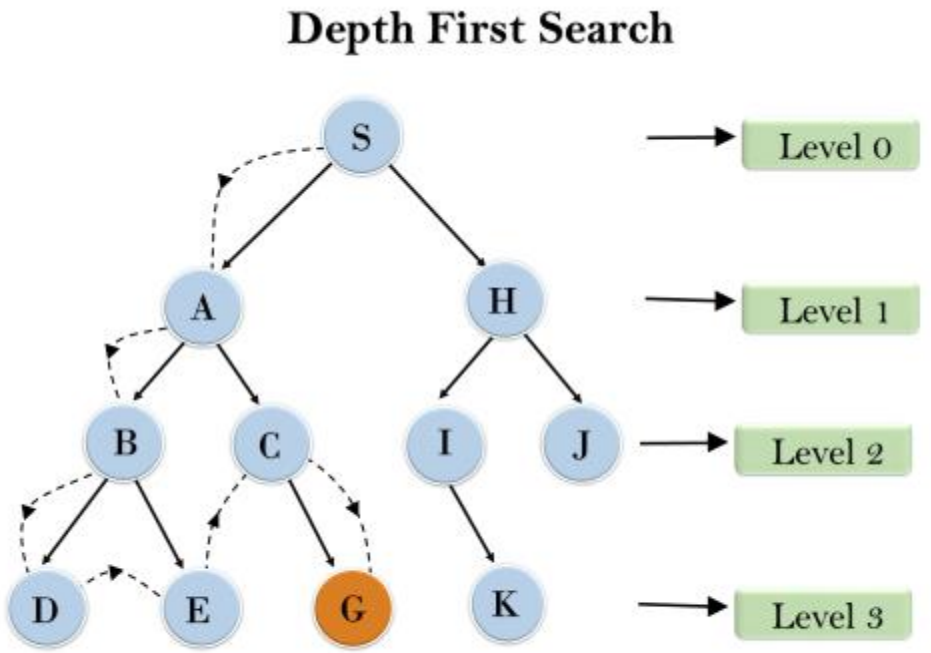
- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Example:

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

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

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



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

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

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

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

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

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

Depth-Limited Search Algorithm

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

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

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

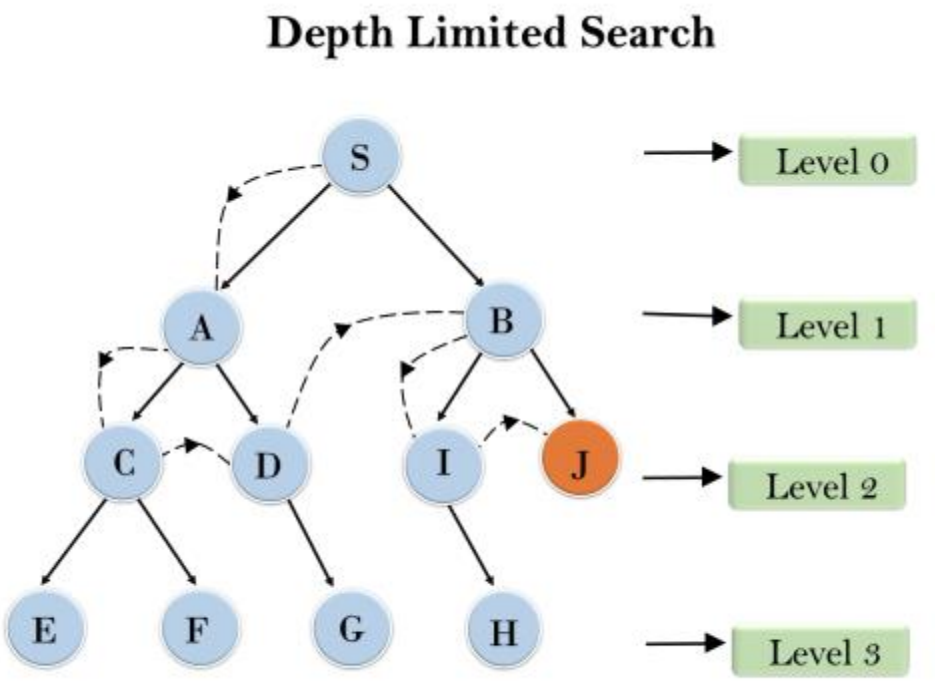
Advantages:

Depth-limited search is Memory efficient.

Disadvantages:

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

Example:



Completeness: DLS search algorithm is complete if the solution is above the depth-limit.
Time Complexity: Time complexity of DLS algorithm is **O(b^{ℓ})**.
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 $\ell > d$.

Uniform-cost Search Algorithm

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

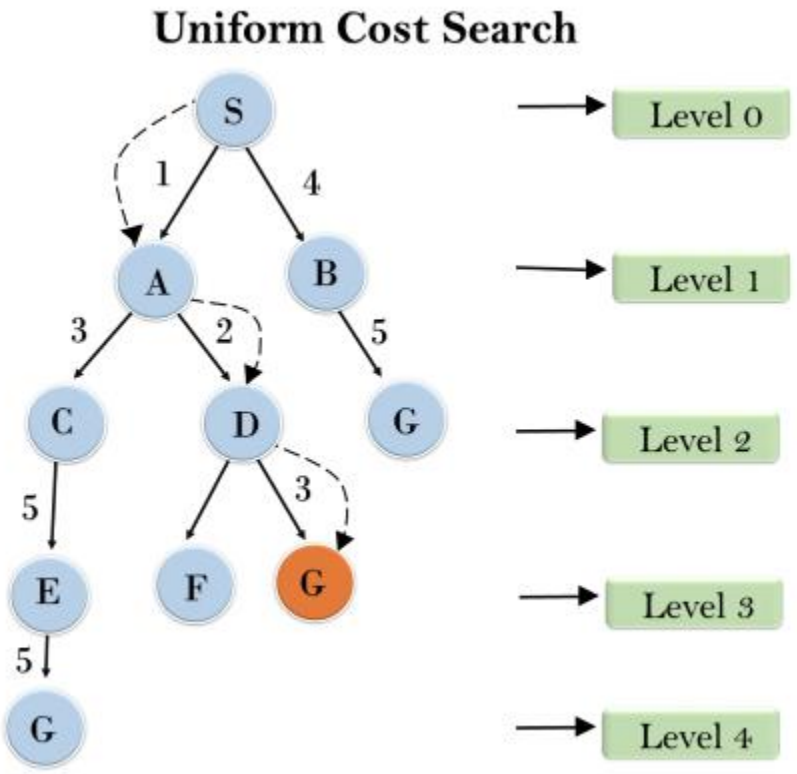
Advantages:

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

Disadvantages:

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

Example:



Completeness:

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

Time Complexity:

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

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

Space Complexity:

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

Optimal:

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

Iterative deepening depth-first Search

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

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

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

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

Advantages:

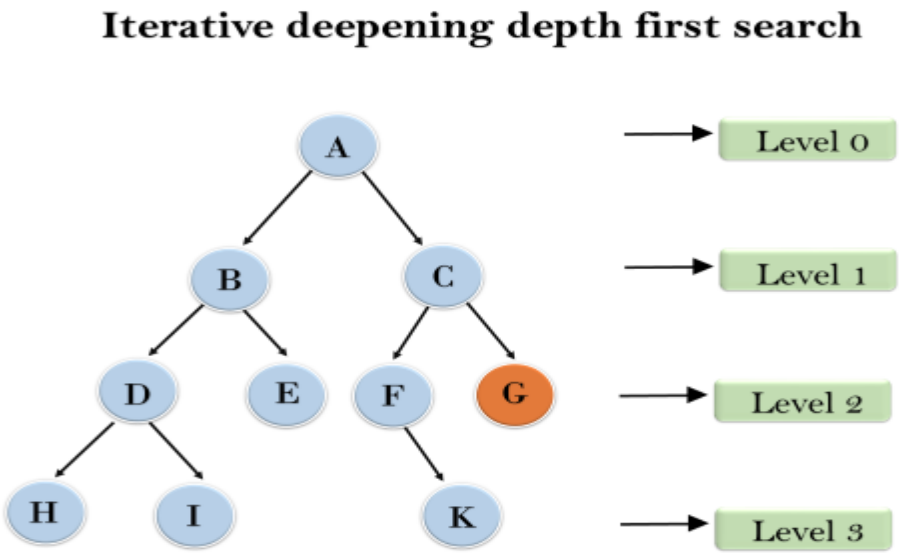
- Itcombines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

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

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.

Bidirectional Search Algorithm

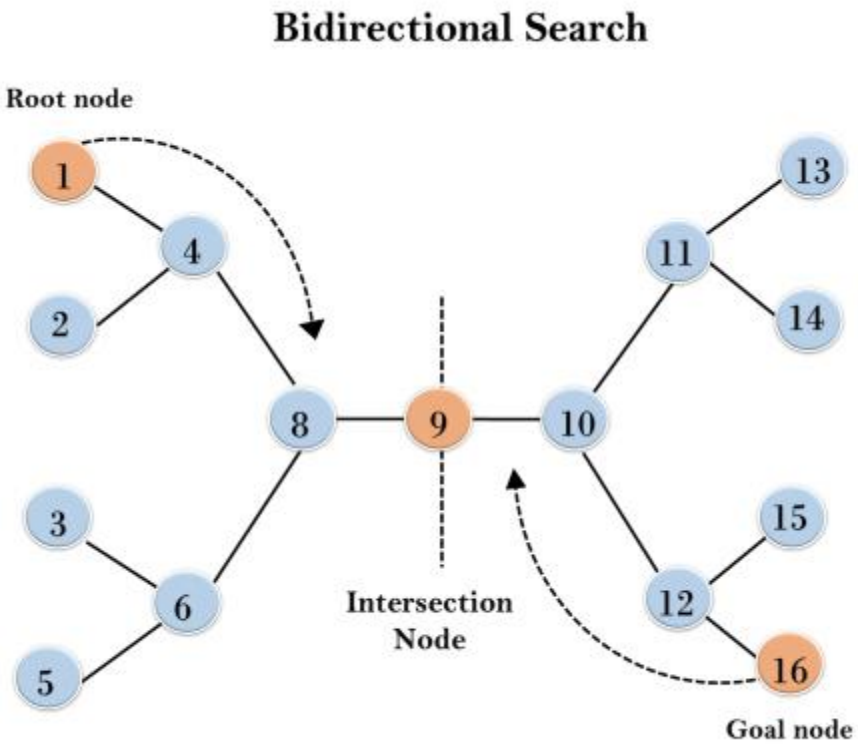
Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from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

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

Disadvantages:

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

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



Completeness: Bidirectional Search is complete if we use BFS in both searches.
Time Complexity: Time complexity of bidirectional search using BFS is **$O(b^d)$**.
Space Complexity: Space complexity of bidirectional search is **$O(b^d)$**.
Optimal: Bidirectional search is Optimal.

Informed (Heuristic) Search Strategies

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.

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

Admissibility of the heuristic function is given as:

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

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

Pure Heuristic Search:

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

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

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

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

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

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic

function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

1.f(n)= g(n).

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

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

Best first search algorithm:

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

Advantages:

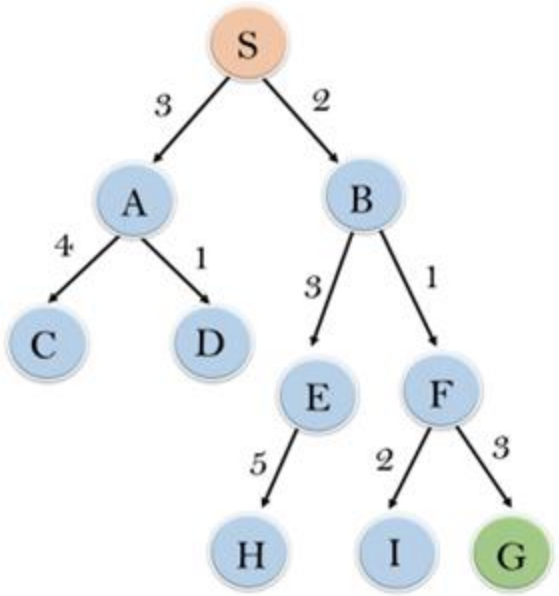
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

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

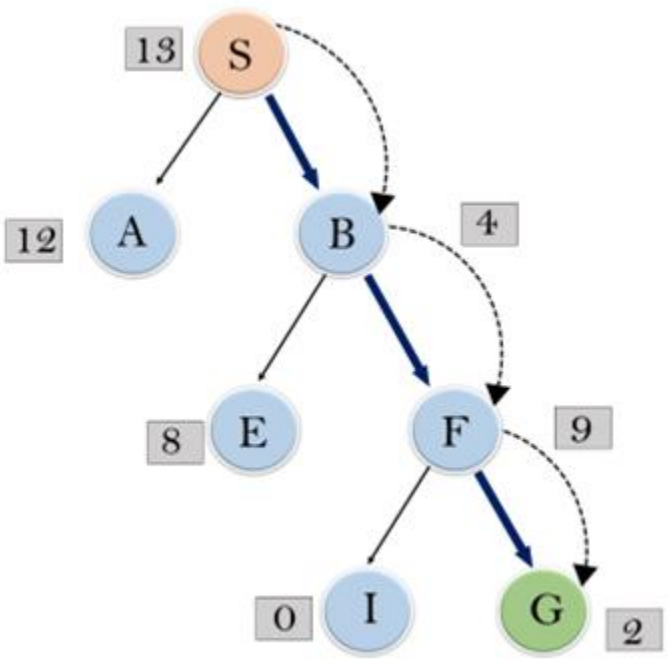
Example:

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



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

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



Expand the nodes of S and put in the CLOSED list

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

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

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

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

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

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

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

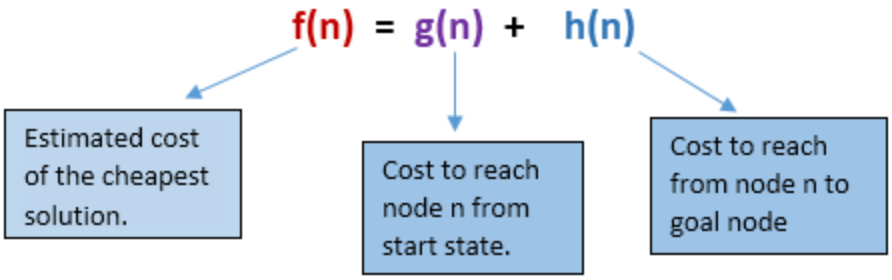
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

2.) A* Search Algorithm:

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

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



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

Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

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

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

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

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

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

Advantages:

- A* search algorithm is the best algorithm than other search algorithms.

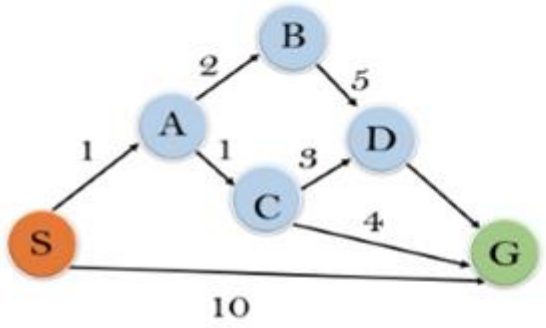
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Disadvantages:

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

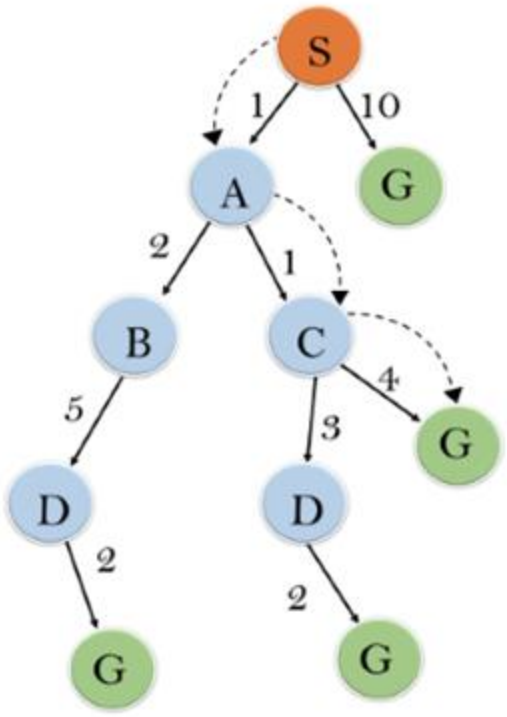
Example:

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



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

Solution:



Initialization: {(S, 5)}

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

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

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

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

Points to remember:

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

Complete: A* algorithm is complete as long as:

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

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

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

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

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

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

VISHNU INSTITUTE OF TECHNOLOGY :: BHIMAVARAM

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

UNIT 3

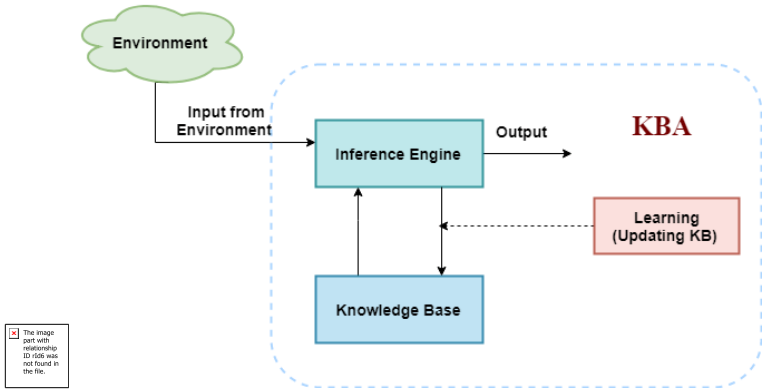
Topics :

1. **Knowledge Based Agents**
2. **Logic, Propositional Logic**

Knowledge Based Agents

- An intelligent agent needs knowledge about the real world for taking decisions and reasoning to act efficiently.
- Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.
- Knowledge-based agents are composed of two main parts:
  - ✓ **Knowledge-base and**
  - ✓ **Inference system**.
- **A knowledge-based agent must able to do the following:**
  - An agent should be able to represent states, actions, etc.
  - An agent Should be able to incorporate new percepts
  - An agent can update the internal representation of the world
  - An agent can deduce the internal representation of the world
  - An agent can deduce appropriate actions.

Architecture of knowledge-based agent:



1. The above diagram is representing a generalized architecture for a knowledge-based agent. The knowledge-based agent (KBA) take input from the environment by perceiving the environment. The input is taken by the inference engine of the agent and which also communicate with KB to decide as per the knowledge store in KB. The learning element of KBA regularly updates the KB by learning new knowledge.
2. **Knowledge base:** Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences (here 'sentence' is a technical term and it is not identical to sentence in English). These sentences are expressed in a language which is called a knowledge representation language. The Knowledge-base of KBA stores fact about the world.Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.Inference system
3. **Inference System :** Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information. Inference system generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given as:
    a) **Forward chaining**
    b) **Backward chaining**
4. **Operations Performed by KBA :**Following are three operations which are performed by KBA in order to show the intelligent behavior:
    a) **TELL:** This operation tells the knowledge base what it perceives from the environment.
    b) **ASK:** This operation asks the knowledge base what action it should perform.
    c) **Perform:** It performs the selected action.

Various levels of knowledge-based agent:

A knowledge-based agent can be viewed at different levels which are given below:

1. Knowledge level

Knowledge level is the first level of knowledge-based agent, and in this level, we need to specify what the agent knows, and what the agent goals are. With these specifications, we can fix its behavior. For example, suppose an automated taxi agent needs to go from

a station A to station B, and he knows the way from A to B, so this comes at the knowledge level.

2. Logical level:

At this level, we understand that how the knowledge representation of knowledge is stored. At this level, sentences are encoded into different logics. At the logical level, an encoding of knowledge into logical sentences occurs. At the logical level we can expect to the automated taxi agent to reach to the destination B.

3. Implementation level:

This is the physical representation of logic and knowledge. At the implementation level agent perform actions as per logical and knowledge level. At this level, an automated taxi agent actually implement his knowledge and logic so that he can reach to the destination.

Approaches to designing a knowledge-based agent:

There are mainly two approaches to build a knowledge-based agent:

1. **1. Declarative approach:** We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with. This approach is called Declarative approach.
2. **2. Procedural approach:** In the procedural approach, we directly encode desired behavior as a program code. Which means we just need to write a program that already encodes the desired behavior or agent.

However, in the real world, a successful agent can be built by combining both declarative and procedural approaches, and declarative knowledge can often be compiled into more efficient procedural code.

Logic, Propositional Logic : A Very Simple Logic

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

Example:

a) It is Sunday.

b) The Sun rises from West (False proposition)

c) 3+3= 7(False proposition)

d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and **logical connectives**.
- These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.
- A proposition formula which is always false is called **Contradiction**.
- A proposition formula which has both true and false values is called
- Statements which are questions, commands, or opinions are not propositions such as "**Where is Rohini**", "**How are you**", "**What is your name**", are not propositions.

Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

a. **Atomic Propositions**
b. **Compound propositions**

- **Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example:

a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.

b) "The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

a) "It is raining today, and street is wet."

b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has ∧ connective such as, **P ∧ Q** is called a conjunction.
   Example: Rohan is intelligent and hardworking. It can be written as,
   P= Rohan is intelligent,
   Q= Rohan is hardworking. → P∧ Q.
3. **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P ∨ Q**.
4. **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as
   If it is raining, then the street is wet.
   Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
5. **Biconditional:** A sentence such as **P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive**
   P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.

Following is the summarized table for Propositional Logic Connectives:

Connective symbols	Word	Technical term	Example
∧	AND	Conjunction	A ∧ B
∨	OR	Disjunction	A ∨ B
→	Implies	Implication	A → B
⇔	If and only if	Biconditional	A⇔ B
¬ or ~	Not	Negation	¬ A or ¬ B

Truth Table:

In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called **Truth table**. Following are the truth table for all logical connectives:

For Negation:

P	¬ P
True	**False**
False	**True**

For Conjunction:

P	Q	P∧ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

For disjunction:

P	Q	P ∨ Q.
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

For Implication:

P	Q	P→ Q
True	True	**True**
True	False	**False**
False	True	**True**
False	False	**True**

For Biconditional:

P	Q	P⇔ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**True**

Truth table with three propositions:

We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

P	Q	R	¬R	P∨ Q	P∨Q→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

Precedence of connectives:

Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

Precedence	Operators
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction(AND)
Fourth Precedence	Disjunction(OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

Logical equivalence:

Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.

Let's take two propositions A and B, so for logical equivalence, we can write it as A⇔B. In below truth table we can see that column for ¬A∨ B and A→B, are identical hence A is Equivalent to B

A	B	¬A	¬A∨ B	A→B
T	T	F	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	F	T	**T**	**T**

Properties of Operators:

- **Commutativity:**
  - P∧ Q= Q ∧ P, or
  - P ∨ Q = Q ∨ P.
- **Associativity:**
  - (P ∧ Q) ∧ R= P ∧ (Q ∧ R),
  - (P ∨ Q) ∨ R= P ∨ (Q ∨ R)
- **Identity element:**
  - P ∧ True = P,
  - P ∨ True= True.
- **Distributive:**
  - P∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).
  - P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).
- **DE Morgan's Law:**
  - ¬ (P ∧ Q) = (¬P) ∨ (¬Q)
  - ¬ (P ∨ Q) = (¬ P) ∧ (¬Q).
- **Double-negation elimination:**
  - ¬ (¬P) = P.

Limitations of Propositional logic:

- We cannot represent relations like ALL, some, or none with propositional logic.
- Example:

1. **All the girls are intelligent.**
2. **Some apples are sweet.**

- Propositional logic has limited expressive power.
- In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

VISHNU INSTITUTE OF TECHNOLOGY :: BHIMAVARAM

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

UNIT 4

Topics :

1. **Introduction to Machine Learning**
2. **Well Posed Learning Problems**
3. **Designing a Learning System**
4. **Perspectives and Issues in Machine Learning**
5. **Introduction to Concept Learning**
6. **Concept Learning as Task, Concept Learning as Search**
7. **Find-S Algorithm for Finding a Maximally Specific Hypothesis**
8. **Version Spaces and the Candidate Elimination Algorithm**
9. **Remarks on Version Spaces and Candidate Elimination**
10. **Inductive Bias**

(Note : The first Four topics above belongs to Chapter 1 of 4th Unit and the Remaining topics belongs to 2nd Chapter of 4th Unit)

Introduction to Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to "self-learn" from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

In traditional programming, a computer engineer writes a series of directions that instruct a computer how to transform input data into a desired output. Instructions are mostly based on an IF-THEN structure: when certain conditions are met, the program executes a specific action.On the other hand, Machine Learning is an automated process that enables

machines to solve problems with little or no human input, and take actions based on past observations.

While artificial intelligence and machine learning are often used interchangeably, they are two different concepts. AI is the broader concept – machines making decisions, learning new skills, and solving problems in a similar way to humans – whereas machine learning is a subset of AI that enables intelligent systems to autonomously learn new things from data.

Instead of programming machine learning algorithms to perform tasks, you can feed them examples of labeled data (known as training data), which helps them make calculations, process data, and identify patterns automatically.

Put simply, Google's Chief Decision Scientist describes machine learning as a fancy labeling machine. After teaching machines to label things like apples and pears, by showing them examples of fruit, eventually they will start labeling apples and pears without any help – provided they have learned from appropriate and accurate training examples.

Machine learning can be put to work on massive amounts of data and can perform much more accurately than humans. It can help you save time and money on tasks and analyses, like solving customer pain points to improve customer satisfaction, support ticket automation, and data mining from internal sources and all over the internet.

To understand how machine learning works, you'll need to explore different machine learning methods and algorithms, which are basically sets of rules that machines use to make decisions. Below, you'll find the five most common and most used types of machine learning:

- ✓ Supervised Learning
- ✓ Unsupervised Learning
- ✓ Semi Supervised Learning
- ✓ Reinforcement LEarning
- ✓ Deep Learning

Well Posed Learning Problems

A computer program is said to learn from experience E in context to some task T and some performance measure P, if its performance on T, as was measured by P, upgrades with experience E.

Any problem can be segregated as well-posed learning problem if it has three traits –

- Task
- Performance Measure
- Experience

Certain examples that efficiently defines the well-posed **learning problem are :**

1. To better filter emails as spam or not

- Task – Classifying emails as spam or not

- Performance Measure – The fraction of emails accurately classified as spam or not spam
- Experience – Observing you label emails as spam or not spam

2. A checkers learning problem

- Task – Playing checkers game
- Performance Measure – percent of games won against opposer
- Experience **–** playing implementation games against itself

3. Handwriting Recognition Problem

- Task – Acknowledging handwritten words within portrayal
- Performance Measure – percent of words accurately classified
- Experience – a directory of handwritten words with given classifications

4. A Robot Driving Problem

- Task – driving on public four-lane highways using sight scanners
- Performance Measure – average distance progressed before a fallacy
- Experience – order of images and steering instructions noted down while observing a human driver

5. Fruit Prediction Problem

- Task – forecasting different fruits for recognition
- Performance Measure – able to predict maximum variety of fruits
- Experience – training machine with the largest datasets of fruits images

6. Face Recognition Problem

- Task – predicting different types of faces
- Performance Measure – able to predict maximum types of faces
- Experience – training machine with maximum amount of datasets of different face images

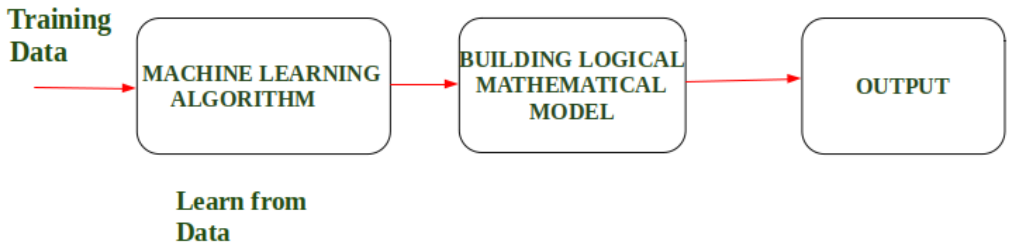
7. Automatic Translation of documents

- Task – translating one type of language used in a document to other language
- Performance Measure – able to convert one language to other efficiently
- Experience – training machine with a large dataset of different types of languages

Designing a Learning System

According to Arthur Samuel "Machine Learning enables a Machine to Automatically learn from Data, Improve performance from an Experience and predict things without explicitly programmed."

In Simple Words, When we fed the Training Data to Machine Learning Algorithm, this algorithm will produce a mathematical model and with the help of the mathematical model, the machine will make a prediction and take a decision without being explicitly programmed. Also, during training data, the more machine will work with it the more it will get experience and the more efficient result is produced.



Example : In Driverless Car, the training data is fed to Algorithm like how to Drive Car in Highway, Busy and Narrow Street with factors like speed limit, parking, stop at signal etc. After that, a Logical and Mathematical model is created on the basis of that and after that, the car will work according to the logical model. Also, the more data the data is fed the more efficient output is produced.

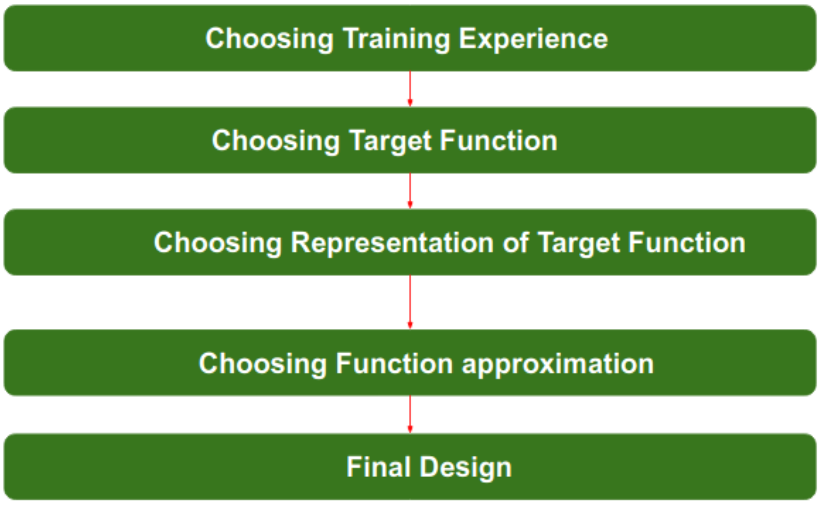
Designing a Learning System in Machine Learning :

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

Example: In Spam E-Mail detection,

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

Steps for Designing Learning System are:



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

Below are the attributes which will impact on Success and Failure of Data:

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

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

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

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

Step 5- Final Design: The final design is created at last when system goes from number of examples , failures and success , correct and incorrect decision and what will be the next step etc. Example: Deep Blue is an intelligent computer which is ML-based won chess game

against the chess expert Garry Kasparov, and it became the first computer which had beaten a human chess expert.

Perspectives and Issues in Machine Learning

"Machine Learning" is one of the most popular technology among all data scientists and machine learning enthusiasts. It is the most effective Artificial Intelligence technology that helps create automated learning systems to take future decisions without being constantly programmed. It can be considered an algorithm that automatically constructs various computer software using past experience and training data. It can be seen in every industry, such as healthcare, education, finance, automobile, marketing, shipping, infrastructure, automation, etc. Almost all big companies like Amazon, Facebook, Google, Adobe, etc., are using various machine learning techniques to grow their businesses. But everything in this world has bright as well as dark sides. Similarly, Machine Learning offers great opportunities, but some issues need to be solved.

Although machine learning is being used in every industry and helps organizations make more informed and data-driven choices that are more effective than classical methodologies, it still has so many problems that cannot be ignored. Here are some common issues in Machine Learning that professionals face to inculcate ML skills and create an application from scratch.

1. Inadequate Training Data

The major issue that comes while using machine learning algorithms is the lack of quality as well as quantity of data. Although data plays a vital role in the processing of machine learning algorithms, many data scientists claim that inadequate data, noisy data, and unclean data are extremely exhausting the machine learning algorithms. Data quality can be affected by some factors as follows:

- **Noisy Data-** It is responsible for an inaccurate prediction that affects the decision as well as accuracy in classification tasks.
- **Incorrect data-** It is also responsible for faulty programming and results obtained in machine learning models. Hence, incorrect data may affect the accuracy of the results also.
- **Generalizing of output data-** Sometimes, it is also found that generalizing output data becomes complex, which results in comparatively poor future actions.

2. Poor quality of data

Data plays a significant role in machine learning, and it must be of good quality as well. Noisy data, incomplete data, inaccurate data, and unclean data lead to less accuracy in classification and low-quality results. Hence, data quality can also be considered as a major common problem while processing machine learning algorithms.

3. Non-representative training data

To make sure our training model is generalized well or not, we have to ensure that sample training data must be representative of new cases that we need to generalize. The training data must cover all cases that are already occurred as well as occurring.

Further, if we are using non-representative training data in the model, it results in less accurate predictions. A machine learning model is said to be ideal if it predicts well for generalized cases and provides accurate decisions. If there is less training data, then there will be a sampling noise in the model, called the non-representative training set. It won't be accurate in predictions. To overcome this, it will be biased against one class or a group.

4. Over-fitting and Under-fitting

Over-fitting:

Over-fitting is one of the most common issues faced by Machine Learning engineers and data scientists. Whenever a machine learning model is trained with a huge amount of data, it starts capturing noise and inaccurate data into the training data set. It negatively affects the performance of the model. We can overcome over-fitting by using linear and parametric algorithms in the machine learning models.

Under-fitting:
Under-fitting is just the opposite of over-fitting. Whenever a machine learning model is trained with fewer amounts of data, and as a result, it provides incomplete and inaccurate data and destroys the accuracy of the machine learning model.

5. Monitoring and maintenance

As we know that generalized output data is mandatory for any machine learning model; hence, regular monitoring and maintenance become compulsory for the same. Different results for different actions require data change; hence editing of codes as well as resources for monitoring them also become necessary.

6. Data Bias

Data Biasing is also found a big challenge in Machine Learning. These errors exist when certain elements of the dataset are heavily weighted or need more importance than others. Biased data leads to inaccurate results, skewed outcomes, and other analytical errors. However, we can resolve this error by determining where data is actually biased in the dataset. Further, take necessary steps to reduce it.

7. Slow implementations and results

This issue is also very commonly seen in machine learning models. However, machine learning models are highly efficient in producing accurate results but are time-consuming. Slow programming, excessive requirements' and overloaded data take more time to provide accurate results than expected. This needs continuous maintenance and monitoring of the model for delivering accurate results.

8. Irrelevant features

Although machine learning models are intended to give the best possible outcome, if we feed garbage data as input, then the result will also be garbage. Hence, we should use relevant features in our training sample. A machine learning model is said to be good if training data has a good set of features or less to no irrelevant features.

Find-S Algorithm for Finding a Maximally Specific Hypothesis

The find-S algorithm is a basic concept learning algorithm in machine learning. The find-S algorithm finds the most specific hypothesis that fits all the positive examples. We have to note here that the algorithm considers only those positive training example. The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.

Important Representation :

1. **?** indicates that any value is acceptable for the attribute.
2. specify a single required value (e.g., Cold) for the attribute.
3. **ϕ**indicates that no value is acceptable.
4. The most **general hypothesis** is represented by: **{?, ?, ?, ?, ?, ?}**
5. The most **specific hypothesis** is represented by: **{ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}**

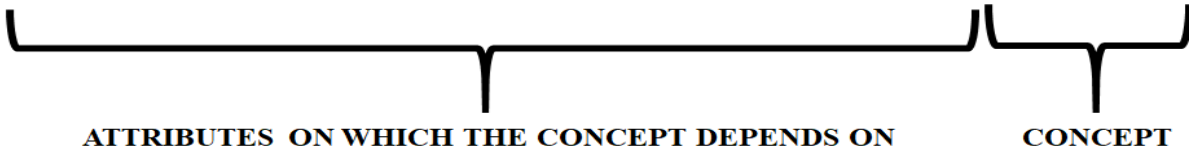
Steps Involved In Find-S :

1. Start with the most specific hypothesis.
   h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}
2. Take the next example and if it is negative, then no changes occur to the hypothesis.
3. If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to a general condition.
4. Keep repeating the above steps till all the training examples are complete.
5. After we have completed all the training examples we will have the final hypothesis when can use to classify the new examples.

Example :

Consider the following data set having the data about which particular seeds are poisonous.

EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES



First, we consider the hypothesis to be a more specific hypothesis. Hence, our hypothesis would be :
h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}

Consider example 1 :

The data in example 1 is { GREEN, HARD, NO, WRINKLED }. We see that our initial hypothesis is more specific and we have to generalize it for this example. Hence, the hypothesis becomes :
h = { GREEN, HARD, NO, WRINKLED }

Consider example 2 :
Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.
h = { GREEN, HARD, NO, WRINKLED }

Consider example 3 :
Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.
h = { GREEN, HARD, NO, WRINKLED }

Consider example 4 :
The data present in example 4 is { ORANGE, HARD, NO, WRINKLED }. We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (" ? "). After doing the process the hypothesis becomes :
h = { ?, HARD, NO, WRINKLED }

Consider example 5 :
The data present in example 5 is { GREEN, SOFT, YES, SMOOTH }. We compare every single

attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (" ? "). After doing the process the hypothesis becomes :

h = { ?, ?, ?, ? }

Since we have reached a point where all the attributes in our hypothesis have the general condition, example 6 and example 7 would result in the same hypothesizes with all general attributes.

h = { ?, ?, ?, ? }

Hence, for the given data the final hypothesis would be :

Final Hypothesis: h = { ?, ?, ?, ? }

Algorithm :

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   For each attribute constraint a, in h
      If the constraint a, is satisfied by x
      Then do nothing
      Else replace a, in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Limitations of the Find-S algorithm:

Find-S algorithm for concept learning is one of the most basic algorithms of machine learning with some limitation and disadvantages. Some of them are listed here:

1. No way to determine if the only final hypothesis (found by Find-S) is consistent with data or there are more hypothesis that is consistent with data.

2. Inconsistent sets of training examples can mislead the finds algorithm as it ignores negative data samples, so an algorithm that can detect inconsistency of training data would be better to use.

3. A good concept learning algorithm should be able to backtrack the choice of hypothesis found so that the resulting hypothesis can be improved over time. Unfortunately, Find-S provide no such method.

Many of the limitations can be removed in one most important algorithm of concept learning called **Candidate elimination algorithm.**

Version Spaces and the Candidate Elimination Algorithm

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples. The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example. The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

- You can consider this as an extended form of the Find-S algorithm.
- Consider both positive and negative examples.

- Actually, positive examples are used here as the Find-S algorithm (Basically they are generalizing from the specification).
- While the negative example is specified in the generalizing form.

Terms Used:

- **Concept learning:** Concept learning is basically the learning task of the machine (Learn by Train data)
- **General Hypothesis:** Not Specifying features to learn the machine.
- **G = {'?', '?','?','?'...}:** Number of attributes
- **Specific Hypothesis:** Specifying features to learn machine (Specific feature)
- **S= {'pi','pi','pi'...}: The number** of pi depends on a number of attributes.
- **Version Space:** It is an intermediate of general hypothesis and Specific hypothesis. It not only just writes one hypothesis but a set of all possible hypotheses based on training data-set.

Algorithm:

Step1: Load Data set
Step2: Initialize General Hypothesis and Specific Hypothesis.
Step3: For each training example
Step4: If example is positive example

```
if attribute_value == hypothesis_value:
   Do nothing
else:
   replace attribute value with '?' (Basically generalizing it)
```

Step5: If example is Negative example
Make generalize hypothesis more specific.

Example:

Consider the dataset given below:

Sky	Temperature	Humid	Wind	Water	Forest	Output
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Algorithmic steps:

Initially : G = [[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?]]
S = [Null, Null, Null, Null, Null, Null]

For instance 1 : <'sunny','warm','normal','strong','warm ','same'> and positive output.

G1 = G

S1 = ['sunny','warm','normal','strong','warm ','same']

For instance 2 : <'sunny','warm','high','strong','warm ','same'> and positive output.

G2 = G

S2 = ['sunny','warm',?,'strong','warm ','same']

For instance 3 : <'rainy','cold','high','strong','warm ','change'> and negative output.

G3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, 'same']]

S3 = S2

For instance 4 : <'sunny','warm','high','strong','cool','change'> and positive output.

G4 = G3

S4 = ['sunny','warm',?,'strong', ?, ?]

At last, by synchronizing the G4 and S4 algorithm produce the output.

Output :

G = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?]]

S = ['sunny','warm',?,'strong', ?, ?]

The Candidate Elimination Algorithm (CEA) is an improvement over the Find-S algorithm for classification tasks. While CEA shares some similarities with Find-S, it also has some essential differences that offer advantages and disadvantages. Here are some advantages and disadvantages of CEA in comparison with Find-S:

Advantages of CEA over Find-S:

1. Improved accuracy: CEA considers both positive and negative examples to generate the hypothesis, which can result in higher accuracy when dealing with noisy or incomplete data.

2. Flexibility: CEA can handle more complex classification tasks, such as those with multiple classes or non-linear decision boundaries.
3. More efficient: CEA reduces the number of hypotheses by generating a set of general hypotheses and then eliminating them one by one. This can result in faster processing and improved efficiency.
4. Better handling of continuous attributes: CEA can handle continuous attributes by creating boundaries for each attribute, which makes it more suitable for a wider range of datasets.

Disadvantages of CEA in comparison with Find-S:

1. More complex: CEA is a more complex algorithm than Find-S, which may make it more difficult for beginners or those without a strong background in machine learning to use and understand.
2. Higher memory requirements: CEA requires more memory to store the set of hypotheses and boundaries, which may make it less suitable for memory-constrained environments.
3. Slower processing for large datasets: CEA may become slower for larger datasets due to the increased number of hypotheses generated.
4. Higher potential for over-fitting: The increased complexity of CEA may make it more prone to over-fitting on the training data, -especially if the dataset is small or has a high degree of noise.

Remarks on Version Spaces and Candidate Elimination

The version space method is still a trial and error method. The program does not base its choice of examples, or its learned heuristics, on an *analysis* of what works or why it works, but rather on the simple assumption that what works will probably work again.Unlike the decision tree ID3 algorithm,

- Candidate-elimination searches an *incomplete* set of hypotheses (ie. only a subset of the potentially teachable concepts are included in the hypothesis space).
- Candidate-elimination finds every hypothesis that is consistent with the training data, meaning it searches the hypothesis space *completely*.
- Candidate-elimination's inductive bias is a consequence of how well it can represent the subset of possible hypotheses it will search. In other words, the bias is a product of its *search space*.
- No additional bias is introduced through Candidate-elimination's *search strategy*.

Advantages of the version space method:

- Can describe all the possible hypotheses in the language consistent with the data.
- Fast (close to linear).

Disadvantages of the version space method:

- Inconsistent data (noise) may cause the target concept to be pruned.

- Learning disjunctive concepts is challenging.

The version space learned by the CandidateElimination Algorithm will converge toward the hypothesis that correctly describes the target concept provided:

(1) There are no errors in the training examples;

(2) There is some hypothesis in *H* that

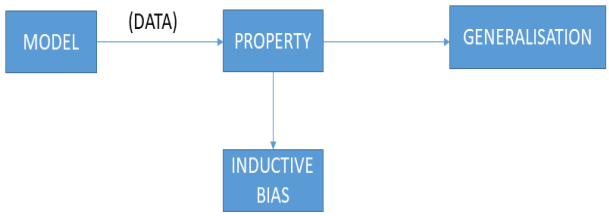
• Convergence can be speed up by presenting the data in a strategic order. The best examples are those that satisfy exactly half of the hypotheses in the current version space.

• Version-Spaces can be used to assign certainty scores to the classification of new examples

Inductive Bias

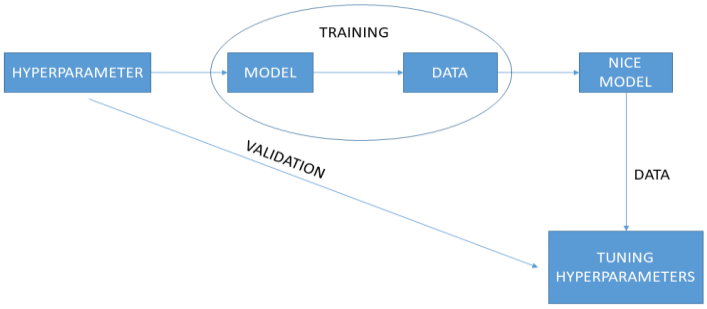
Before learning a model given a data and a learning algorithm, there are a few assumptions a learner makes about the algorithm. These assumptions are called the inductive bias. It is like the property of the algorithm.

For eg. in the case of decision trees, the depth of the tress is the inductive bias. If the depth of the tree is too low, then there is too much generalization in the model. Similarly, if the depth of the tree is too much, there is too less generalization and while testing the model on a new example, we might reach a particular example used to train the model. This may give us incorrect results.



Hyperparameters : In machine learning the hyperparameters are used to control the learning process as compared to parameters which are obtained after training the model on the data. The hyperparameters are independent of the data.

Hyperparameters are set manually before the training of the model. Generally the hyper parameter is chosen using the inductive bias. After we set out hyper parameter, we train on the data and get a trained model. We used the trained model on separate data to "validate" it. Using this, we tune our hyperparameters as per our requirement. A basic flowchart is given below:



The image part with relationship ID rId11 was not found in the file.

Introduction to Concept Learning

- Concept learning is Acquiring the definition of a general category from given sample positive and negative training examples of the category.
- *Concept Learning* can seen as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- The hypothesis space has a *general-to-specific ordering* of hypotheses, and the search can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space.
- Much of human learning involves acquiring general concepts from past experiences. For an example, humans identify different vehicles among all the vehicles based on some specific set of features defined over a large set of features. This special set of features differentiates the subset of cars in the set of vehicles. This set of features that differentiate cars, can be called a concept.
- Similarly, machines can also learn from the concepts to identify whether an object belongs to a specific category or not by processing past/training data to find a hypothesis that best fits the training examples.
- Target Concept:The set of items/objects over which the concept is defined is called the set of instances and denoted by X. The concept or function to be learned is called the target concept and denoted by c. It can be seen as a boolean valued function defined over X and can be represented as:

$$c: X \rightarrow \{0, 1\}$$

VISHNU INSTITUTE OF TECHNOLOGY :: BHIMAVARAM

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

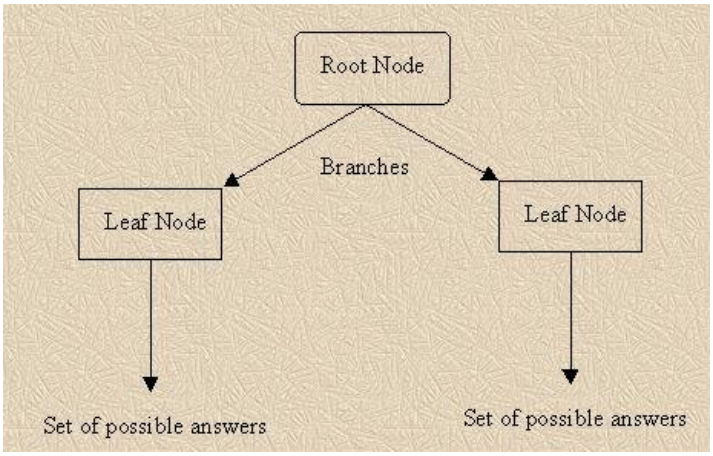
UNIT 5

Topics :

1. **Introduction to Decision Tree**
2. **Decision Tree Representation**
3. **Appropriate Problems for Decision Tree Learning**
4. **The Basic Decision Tree Learning Algorithm**
5. **Hypothesis Space Search in Decision Tree Learning**
6. **Inductive Bias in Decision Tree Learning**
7. **Issues in Decision Tree Learning**

1. Introduction to Decision Tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.



As you can see from the diagram above, a decision tree starts with a root node, which does not have any incoming branches. The outgoing branches from the root node then feed into the internal nodes, also known as decision nodes. Based on the available features, both node types conduct evaluations to form homogenous subsets, which are denoted by leaf nodes, or terminal nodes. The leaf nodes represent all the possible outcomes within the dataset.

Advantages of Decision Tree :

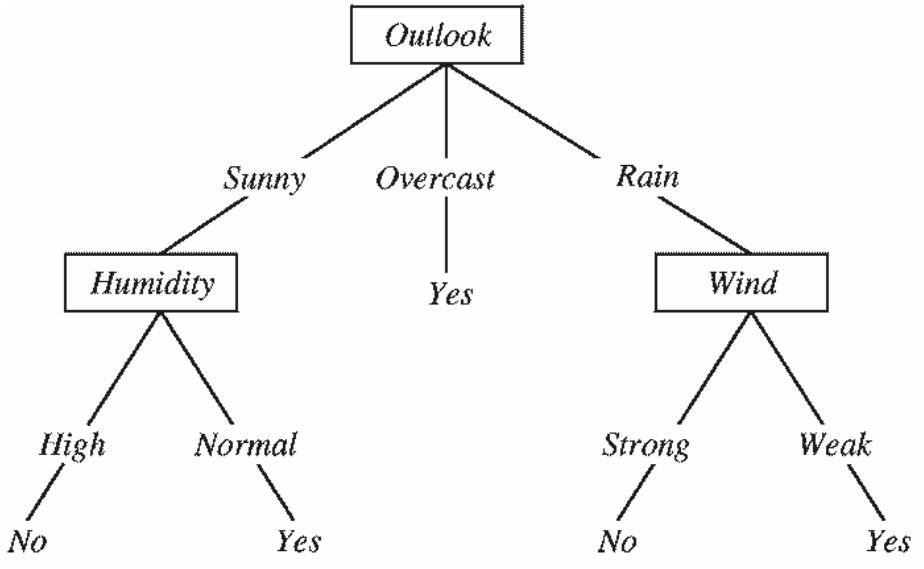
- ✓ **Easy to interpret:** The Boolean logic and visual representations of decision trees make them easier to understand and consume. The hierarchical nature of a decision tree also makes it easy to see which attributes are most important, which isn't always clear with other algorithms, like neural networks.
- ✓ **Little to no data preparation required:** Decision trees have a number of characteristics, which make it more flexible than other classifiers. It can handle various data types—i.e. discrete or continuous values, and continuous values can be converted into categorical values through the use of thresholds. Additionally, it can also handle values with missing values, which can be problematic for other classifiers, like Naïve Bayes.
- ✓ **More flexible:** Decision trees can be leveraged for both classification and regression tasks, making it more flexible than some other algorithms. It's also insensitive to underlying relationships between attributes; this means that if two variables are highly correlated, the algorithm will only choose one of the features to split on.

Disadvantages of Decision Tree :

- ✓ **Prone to overfitting:** Complex decision trees tend to overfit and do not generalize well to new data. This scenario can be avoided through the processes of pre-pruning or post-pruning. Pre-pruning halts tree growth when there is insufficient data while post-pruning removes subtrees with inadequate data after tree construction.
- ✓ **High variance estimators:** Small variations within data can produce a very different decision tree. Bagging, or the averaging of estimates, can be a method of reducing variance of decision trees. However, this approach is limited as it can lead to highly correlated predictors
- ✓ **More costly:** Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.
- ✓ **Not fully supported in scikit-learn:** Scikit-learn is a popular machine learning library based in Python. While this library does have a Decision Tree module (DecisionTreeClassifier, link resides outside of ibm.com), the current implementation does not support categorical variables.

2. Decision Tree Representation

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. Below diagram illustrate the basic flow of decision tree for decision making with labels (Rain(Yes), No Rain(No)).



8.

Fig. Decision Tree for Rain Forecasting

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric **supervised learning** method used for both **classification** and **regression** tasks.Tree models where the target variable can take a discrete set of values are called **classification trees**. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**. Classification And Regression Tree (CART) is general term for this.

Approach to make decision tree

While making decision tree, at each node of tree we ask different type of questions. Based on the asked question we will calculate the **Information Gain** corresponding to it.**Information gain** is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information. For each node of the tree, the information value measures how much information a feature gives

us about the class. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes are pure, or until the information gain is 0.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

Decision Tree Terminology

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

3. Appropriate Problems for Decision Tree Learning

Although a variety of decision-tree learning methods have been developed with somewhat differing capabilities and requirements, decision-tree learning is generally best suited to problems with the following characteristics:

1. Instances are represented by attribute-value pairs.

"Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot). The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold). However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically)."

2. *The target function has discrete output values.*

*"The decision tree is usually used for Boolean classification (e.g., **yes** or **no)** kind of example. Decision tree methods easily extend to learning functions with more than two possible output values. A more substantial extension allows learning target functions with real-valued outputs, though the application of decision trees in this setting is less common."*

3. *Disjunctive descriptions may be required.*

"In general, decision trees represent a disjunction of conjunctions of constraints on the attribute-values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions". Hence, Decision trees naturally represent disjunctive expressions.

4. *The training data may contain errors.*

"Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples."

5. *The training data may contain missing attribute values.*

*"Decision tree methods can be used even when some training examples have unknown values (e.g., if the **Humidity** of the day is known for only some of the training examples)."*

4. The Basic Decision Tree Learning Algorithm

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

The core algorithm for building decision trees is called ID3 Algorithm.The Construction of a decision tree given the data consisting a set of attributes and set of discrete values follows the basic decision tree learning Algorithm(ID3) :

Step 1: Identify the dependent Node or Target Node
Step 2: Calculate the Information Gain for the Target Node.
Step 3 : For all Non Target Attributes, Calculate Entropy
Step 4 : Calculate the Final Gain for each Non Target Attribute.
Step 5 : Construct the Decision Tree by using the attribute with highest Final Gain as the root node.

Information Gain : The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches). The Information Gain for a particular can be calculated as

$$\textbf{Information Gain} = -(P/P+N)\log_2^{(P/P+N)} - (N/N+P)\log_2^{(N/N+P)}$$

Entropy : A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one. Entroyp can be calculated as

Entropy(Attribute1)=Information Gain(Attribute1) * Probability(Attribute1)

5. Hypothesis space Search in Decision Tree Learning

Hypothesis space is defined as a set of all possible legal hypotheses; hence it is also known as a hypothesis set. It is used by supervised machine learning algorithms to determine the best possible hypothesis to describe the target function or best maps input to output.

It is often constrained by choice of the framing of the problem, the choice of model, and the choice of model configuration.

Hypothesis (h):

It is defined as the approximate function that best describes the target in supervised machine learning algorithms. It is primarily based on data as well as bias and restrictions applied to data.Hence hypothesis (h) can be concluded as a single hypothesis that maps input to proper output and can be evaluated as well as used to make predictions.

The hypothesis (h) can be formulated in machine learning as follows:

y= mx + b Where,

y: Range ; x: domain; m: Slope of the line which divided test data or changes in y divided by change in x.

Unlike machine learning, we cannot accept any hypothesis in statistics because it is just an imaginary result and based on probability. Before start working on an experiment, we must be aware of two important types of hypotheses as follows:

- **Null Hypothesis:** A null hypothesis is a type of statistical hypothesis which tells that there is no statistically significant effect exists in the given set of observations. It is also known as conjecture and is used in quantitative analysis to test theories about markets, investment, and finance to decide whether an idea is true or false.
- **Alternative Hypothesis:** An alternative hypothesis is a direct contradiction of the null hypothesis, which means if one of the two hypotheses is true, then the other must be

false. In other words, an alternative hypothesis is a type of statistical hypothesis which tells that there is some significant effect that exists in the given set of observations.

6. Inductive Bias in Decision Tree Learning

The **inductive bias** (also known as **learning bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.Inductive bias is anything which makes the algorithm learn one pattern instead of another . In machine learning, one aims to construct algorithms that are able to *learn* to predict a certain target output. To achieve this, the learning algorithm is presented some training examples that demonstrate the intended relation of input and output values. Then the learner is supposed to approximate the correct output, even for examples that have not been shown during training. Without any additional assumptions, this problem cannot be solved since unseen situations might have an arbitrary output value. The kind of necessary assumptions about the nature of the target function are subsumed in the phrase *inductive bias*. A classical example of an inductive bias is Occam's razor, assuming that the simplest consistent hypothesis about the target function is actually the best. Here *consistent* means that the hypothesis of the learner yields correct outputs for all of the examples that have been given to the algorithm. Approaches to a more formal definition of inductive bias are based on mathematical logic. Here, the inductive bias is a logical formula that, together with the training data, logically entails the hypothesis generated by the learner. However, this strict formalism fails in many practical cases, where the inductive bias can only be given as a rough description, or not at all.

Types of Inductive Bias :

1. **Restriction Bias**
2. **Preference Bias**

Restriction Bias :

The Candidate elimination algorithm searches an incomplete hypothesis space for certain hypothesis that best fits the training data. Its inductive bias is solely the consequence of the power of its hypothesis representation and hence adhere to a particular category. Hence, the bias is of type Restrictive.

Preference Bias :

The ID3 algorithm searches a complete hypothesis space. It searches incompletely through this space from simple to complex hypothesis until its termination conditions met. Its hypothesis space introduces no additional bias. The inductive bias of ID3 is thus a preference for certain hypothesis over others and is called as Preference Bias.

7. Issues in Decision Tree Learning

Decision Tree models are sophisticated analytical models that are simple to comprehend, visualize, execute, and score, with minimum data pre-processing required. These are supervised learning systems in which input is constantly split into distinct groups based on specified factors. They also have limitations which we are going to discuss; when there are few decisions and consequences in the tree, decision trees are generally simple to understand. Typical examples include the inability to measure attribute values, the high cost and complexity of such measures, and the lack of availability of all attributes at the same time.

1. Avoiding Overfitting the Data

- Definition: Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.
- Why might overfitting occur? There might be noise in the training data. There might be coincidental regularities in the data.
- One solution is to stop growing the tree early.
- Another solution is to post-prune the tree. This approach has been found to be more successful in practice.
- To make decisions for either of these approaches, we should split the known data into training data and validation data.

Node Pruning

- Remove the subtree rooted at that node, make it a leaf node and assign the most common classification to it. Only do this if the resulting tree performs no worse on the validation data.
- Figure 3.7 illustrates this process.

Rule Pruning (C4.5)

1. Build a decision tree.
2. Convert the tree into an equivalent set of rules.
3. Generalize each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy and use them in this sequence when classifying subsequent instances.

2. Incorporating Continuous-Valued Attributes

- A binary solution is to order the n pieces of data and consider the n-1 middle points that partition the data. Calculate the information gain that results from making the test attribute ≤ value.
- The binary solution can be extended to multiple intervals.

3. Alternative Measures for Selecting Attributes

- Problem: Attributes with large numbers of values will partition the data perfectly. For example, the attribute date-and-time-of-event-in-milliseconds-since-year-0.
- Solution: Modify information gain to penalize such attributes.

4. Handling Missing Attribute Values

- Solution: Assign the value to be the most commonly occurring value among the set of examples at this node.
- Solution: Assign the value to be a distribution of probabilities.