CONTENTS

Publishers : **KAMAL PUBLISHING HOUSE**
110-201, Laxmi Tower,
Near R.K. Mission School & Hospital
R.K. Nagar, G.T. Road
Kanpur – 208 012
(Uttar Pradesh)
Contact No. +91 9559401840
Email kamalpubhouse@gmail.com
Website www.kphouse.in

Composed By : Komal Graphics
Kanpur

SYLLABUS

	Topic
1.	**Artificial Intelligence** : Introduction to artificial intelligence, Historical development and foundation areas of artificial intelligence, Tasks and application areas of artificial intelligence. Introduction, types and structure of intelligent agents, Computer Vision, Natural language processing
2.	**Searching Techniques** : Introduction, Problem solving by searching, Searching for solutions, Uniformed searching techniques, Informed searching techniques, Local search algorithms, Adversarial search methods, Search techniques used in games, Alpha Beta pruning
3.	**Knowledge Representation and Reasoning** : Propositional logic, Predicate logic, First order logic, Inference in first order logic, Clause form conversion, Resolution Chaining concept, forward chaining and backward chaining, Utility theory and Probabilistic reasoning, Hidden Markov model, Bayesian networks
4.	**Machine Learning** : Introduction, types and application areas, Decision trees, Statistical learning methods, Learning with complete data - concept and Naive Bayes models, Learning with hidden data - concept and EM algorithm, Reinforcement learning
5.	**Pattern Recognition** : Introduction and design principles, Statistical pattern recognition, Parameter estimation methods - Principle component analysis and Linear discrimination analysis, Classification techniques - Nearest neighbor rule and Bayes classifier, K-means clustering, Support vector machine

UNIT – ONE
INTRODUCTION

Q.1. Name the elements of an agent. (2018-19)
OR What is an intelligent agent? Discuss any two types of intelligent agents. (2017-18)
OR Give the structure of an intelligent agent and draw schematic diagrams of a model based reflex agent and a learning agent.

Ans. The term "agent" and "intelligent agent" are used in two different but related senses which are often confused. In computer science, an Intelligent Agent (IA) is a software agent that assists users and will act on their behalf, in performing non-repetitive computer-related tasks, in the sense of a "representative agent", like an insurance agent or travel agent. Intelligent agents are used for operator assistance or data mining (Sometimes referred to as bots). While they are often based on fixed pre-programmed rules, "intelligent" in this context is often taken to imply the ability to adapt and learn.

In artificial intelligence, an intelligent agent is used for intelligent actors which observe and act upon an environment, in the sense of a rational agent : an entity that is capable of perception, action and goal directed behavior. Such an agent might be a robot or an embedded real time software system - and is intelligent if it interacts with its environment in a manner that would normally be regarded as intelligent if that interaction were carried out by a human being.

There are no reasons why these two notions of intelligent agents need to be related. An intelligent agent in the first sense might be implemented using conventional software techniques and display no more intelligence than a conventional computer program. On the other hand, an intelligent agent in the second sense might be wholly autonomous, carried out its own agenda, and acting as an agent for no one.

Intelligent Agents in Artificial Intelligence : The academic community has been very active in defining the concept of agents. Practically the concepts of agents are associated to software entities which have some ability to [illegible] environment, to act on the

environment, to have some level of autonomy (via a set of internal goals). In the artificial intelligence sense of the term, there are multiple types of agents and sub-agents. For example :

Physical Agents : A physical agent is an entity which percepts through sensors and acts through actuators.

Temporal Agents : A temporal agent may use time based stored information to offer instructions or data acts to a computer program or human being and takes program inputs percepts to adjust its next behaviors

Believable Agents : An agent exhibiting a personality via the use of an artificial character (the agent is embedded) for the interaction. A simple agent program can be defined mathematically as an agent function which maps every possible percepts sequence to a possible action the agent can perform or to a coefficient, feedback element, function or constant that affects eventual actions :

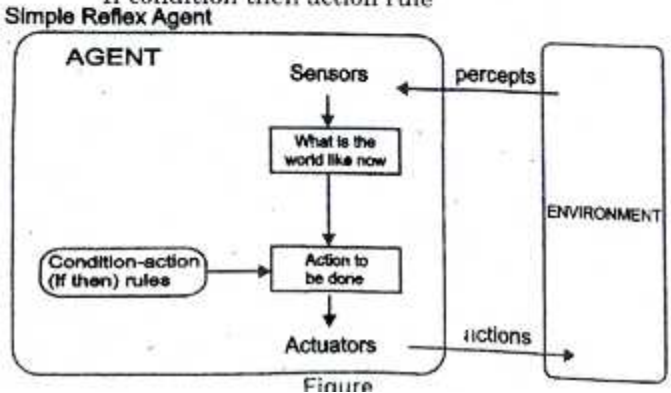
$$f : P^* \rightarrow A$$

The program agent, instead, maps every possible percept to an action. It is possible to group agents into five classes based on their degree of perceived intelligence and capability :

(1) Simple reflex agents;
(2) Model-based reflex agents;
(3) Goal-based agents;
(4) Utility-based agents;
(5) Learning agents.

(1) **Simple Reflex Agents :** Simple reflex agents acts only on the basis of the current percept. The agent function is based on the condition-action rule :

If condition then action rule



Figure

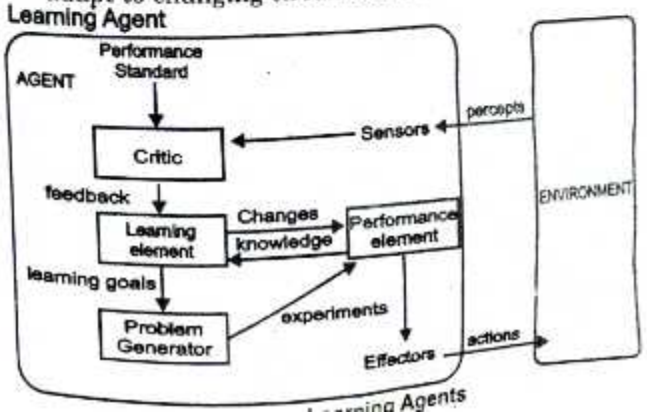
This agent function only succeeds when the environment is fully observable. Some reflex agents can also contain information on their current state which allows them to disregard conditions whose actuators are already triggered

(2) **Model-Based Reflex Agents :** Model-based agents can handle partially observable environments. Its current state is stored inside the agent maintaining some kind of structure which describes the part of the world which cannot be seen. This behavior requires information on how they would behave and works. This additional information completes the "World View" model.

(3) **Goal-Based Agents :** Goal-based agents are model-based agents which store information regarding situations that are desirable. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.

(4) **Utility-Based Agents :** Goal-based agents only distinguish between goal states and non-goal states. It is possible to define a measure of how desirable a particular state is. This measure can be obtained through the use of a utility function which maps a state to a measure of the utility of the state.

(5) **Learning Agents :** In some literature IAs are also referred to as autonomous intelligent agents, which means they act independently, and will learn and adapt to changing circumstances.



Learning Agents

According to Nikola Kasabov, IA systems should exhibit the following characteristics :

(a) Learn and improve through interaction with the environment (embodiment).
(b) Adapt online and in real time.
(c) Learn quickly from large amounts of data.
(d) Accommodate new problem solving rules incrementally.
(e) Have memory based exemplar storage and retrieval capacities.
(f) Have parameters to represent short and long-term memory, age, forgetting, etc.
(g) Be able to analyze itself in terms of behavior error and success.

To actively perform their functions, intelligent agents today are normally gathered in a hierarchical structure containing many "Sub-agents". Intelligent sub-agents process and perform lower level functions

(1) Temporal Agents (for time-based decisions);
(2) Spatial Agents (that relate to the physical real-world);
(3) Input Agents (that process and make sense of sensor inputs-example neural network based agents neural network);
(4) Agents (that solve a problem like speech recognition)
(5) Decision Agents (that are geared to decision making)
(6) Learning Agents (for building up the data structure and database of other intelligent agents);
(7) World Agents (that incorporate a combination of all the other classes of agents to allow autonomous behaviors).

Intelligent Agent in Computer Science : A very limited set of agents, that might be classified as semi-intelligent due to their lack of complexity, decision making, extremely limited world view and learning capacity can be found in the reference : Third Canadian Edition of "Management Information Systems for the Information Age" The document suggests that there are only four essential types of Intelligent Agents :

(1) Buyer agents or shopping bots
(2) User or personal agents

(3) Monitoring-and-surveillance agents
(4) Data mining agents.

Q.2. Summarize the factors that make up rationality. (2018-19)

Ans. An agent which acts in a way that is expected to maximize to its performance measure, given the evidence provided by what it perceived and whatever built-in knowledge it has. The performance measure defines the criterion of success for an agent. Such agents are also known as Rational Agents. The rationality of the agent is measured by its performance measure, the prior knowledge it has, the environment it can perceive and actions it can perform.

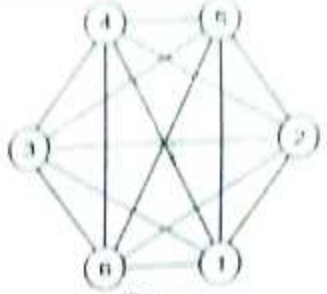
This concept is central in Artificial Intelligence. The above properties of the intelligent agents are often grouped in the term PEAS, which stands for Performance, Environment, Actuators and Sensors. So, for example a self driving car would be having following PEAS :

(1) **Performance :** Safety, time, legal drive, comfort.
(2) **Environment :** Roads, other cars, pedestrians, road signs.
(3) **Actuators :** Steering, accelerator, brake, signal, horn.
(4) **Sensors :** Camera, sonar, GPS, speedometer, odometer, accelerometer, engine sensors, keyboard.
(5) **Rationality :** What is rational at any given time depends on four things:
   (a) The performance measure that defines the criterion of success.
   (b) The agent's prior knowledge of the environment.
   (c) The actions that the agent can perform
   (d) The agent's percept sequence to date.

This leads to a definition of a rational agent (ideal rational agent).

Q.3. How suited would PROLOG be in implementing the search algorithms? Comment on how this might be done and what difficulties might exist. (2018-19)

Ans. A prolog program consists of facts and rules. The following are some examples of facts. car(rusty). car(dino). colour(rusty, blue). colour(dino, red). These facts state that rusty and dino are both cars. They also state the colour of each car. Notice that a fact has two parts, a property or



Figure

Let us now solve the following task. Which nodes are reachable from a particular node, in a reflexive and transitive way? Or slightly more generally, what is the reflexive transitive closure of a set of nodes? In Prolog, we can write this as

```
reachable(_, _, From, From).
reachable(Adjs, Visited, From, To) :-
    maplist(dif(Next), Visited),
    member(From-As, Adjs),
    member(Next, As),
    reachable(Adjs, [From|Visited], Next, To).
```

To compute the set of solutions, we can use setof/3. For example, all nodes that are reachable from node 1:

```
?- k_n(3, Adjs),
   setof(To, reachable(Adjs, [], 1, To), Tos).
Adjs = [1-[2, 3], 2-[1, 3], 3-[1, 2]],
Tos = [1, 2, 3].
```

The major drawback of this approach is that it doesn't scale well. In particular, we have:

```
?- list_length(_, N), portray_clause(N),
   k_n(N, Adjs),
   time(setof(To, reachable(Adjs, [], 1, To), Tos)),
   false.
...
      6% 110,079 inferences, 0.012 CPU in 0.012 seconds
(98% CPU, 9334266 Lips)
      7% 914,953 inferences, 0.093 CPU in 0.098 seconds
(95% CPU, 9816460 Lips)
      8 % 8,446,107 inferences, 0.823 CPU in 0.837 seconds
(98% CPU, 10265329 Lips)
      9% 85,982,043 inferences, 8.086 CPU in 8.135 seconds
(99% CPU, 10633230 Lips)
```

This is because the number of paths in this graph increases super-exponentially in the number of nodes, and the naive solution traverses all paths. This approach

quickly becomes too slow, no matter which language y use to implement it.

In this concrete case, we can solve the task in a mu more efficient way. For example, we can use Warshal algorithm for computing the transitive closure, with co similar to :

```
warshall(Adjs, Nodes0, Nodes) :-
    phrase(reachables(Nodes0, Adjs), Nodes1, Nodes0),
    sort(Nodes1, Nodes2),
    if_(Nodes2 = Nodes0,
        Nodes = Nodes2,
        warshall(Adjs, Nodes2, Nodes)).
reachables([], _) --> [].
reachables([Node|Nodes], Adjs) -->
    { member(Node-Rs, Adjs) },
    Rs,
    reachables(Nodes, Adjs).
```

Note how sort/2 is used to remove duplicates from list, and to obtain a canonical representation of the set nodes that have already been found. Sample query :

```
?- k_n(9, Adjs),
   time(warshall(Adjs, [1], Tos)).
% 322 inferences, 0.000 CPU in 0.000 seconds (93% CP
4735294 Lips)

Tos = [1, 2, 3, 4, 5, 6, 7, 8, 9].
```

This is clearly much more efficient. By implementin intelligent strategies, you can obtain elegant and efficien Prolog solutions for many search tasks.

Q.4. What is meant by the term Artificial Intelligence How it is different from natural intelligence? **(2017-18**

OR **What is artificial intelligence? How it is differen than general intelligence?** **(2015-16**

OR **List the characteristics of intelligence.** **(2015-16**

OR **Define the terms-natural intelligence and artificia intelligence. How do you differentiate between th two?** **(2014-15**

OR **What do you understand by Artificial Intelligence Why AI is important?**

Ans. Artificial Intelligence : The branch of computer scienc concerned with making computers behave like human The term was coined in 1956 by John McCarthy at th Massachusetts Institute of Technology. Artifici intelligence includes :

Games Playing : Programming computers to play games such as chess and checkers.

Expert Systems : Programming computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms).

Natural Language : Programming computers to understand natural human languages.

Neural Networks : Systems that simulate intelligence by attempting to reproduce the types of physical connections that occur in animal brains.

Robotics : Programming computers to see and hear and react to other sensory stimuli. Today, the hottest area of artificial intelligence is neural networks, which are proving successful in a number of disciplines such as voice recognition and natural-language processing. There are several programming languages that are known as AI languages because they are used almost exclusively for AI applications. The two most common are LISP and PROLOG.

Importance of AI : Almost every branch of science and engineering currently shares the tools and techniques available in the domain of artificial intelligence. Here few typical applications, where AI plays a significant and decisive role in engineering automation are mentioned :

(1) **Expert Systems :** In this example, we illustrate the reasoning process involved in an expert system for a weather forecasting problem with special emphasis to its architecture. An expert system consists of a knowledge base, database and an inference engine for interpreting the database using the knowledge supplied in the knowledge base. The reasoning process of a typical illustrative expert system is described in figure PR 1 in figure represents i^{th} production rule. The inference engine attempts to match the antecedent clauses (IF parts) of the rules with the data stored in the database. When all the antecedent clauses of a rule are available in the database, the rule is fired, resulting in new inferences. The resulting inferences are added to the database for activating subsequent firing of other rules. In order to keep limited data in the database, a few rules that contain an explicit consequent (THEN)

clause to delete specific data from the databases employed in the knowledge base. On firing of s rules, the unwanted data clauses as suggested by rule are deleted from the database.

Here PR_1 fires as both of its antecedent clau are present in the database. On firing of PR_1, consequent clause "it-will-rain" will be added to database for subsequent firing of PR_2.

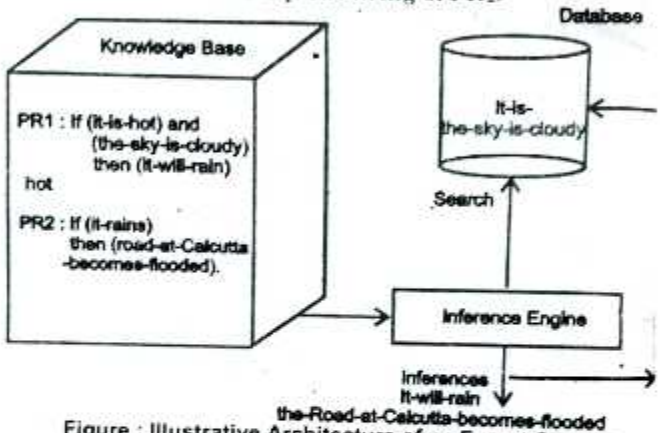


Figure : Illustrative Architecture of an Expert System

(2) **Image Understanding and Computer Vision :** digital image can be regarded as a two-dimensiona array of pixels containing gray levels correspondin to the intensity of the reflected illumination receive by a video camera. For interpretation of a scene, it image should be passed through three basi processes : low, medium and high level vision.

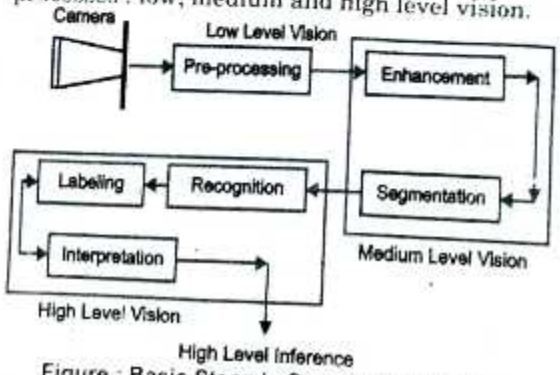


Figure : Basic Steps in Scene Interpretation

The importance of low level vision is to pre-process the image by filtering from noise. The medium level vision system deals with enhancement of details and segmentation (i.e., partitioning the image into objects of interest). The high level vision system includes three steps : recognition of the objects from the segmented image, labeling of the image and interpretation of the scene. Most of the AI tools and techniques are required in high level vision systems. Recognition of objects from its image can be carried out through a process of pattern classification, which at present is realized by supervised learning algorithms. The interpretation process, on the other hand, requires knowledge-based computation.

(3) **Speech and Natural Language Understanding :** Understanding of speech and natural languages is basically two classical problems. In speech analysis, the main problem is to separate the syllables of a spoken word and determine features like amplitude, and fundamental and harmonic frequencies of each syllable. The words then could be identified from the extracted features by pattern classification techniques. Recently, artificial neural networks have been employed to classify words from their features. The problem of understanding natural languages like English, on the other hand, includes syntactic and semantic interpretation of the words in a sentence, and sentences in a paragraph.

(4) **Scheduling :** In a scheduling problem, one has to plan the time schedule of a set of events to improve the time efficiency of the solution. For instance in a class-routine scheduling problem, the teachers are allocated to different classrooms at different time slots, and we want most classrooms to be occupied most of the time. In a flow-shop scheduling problem, a set of jobs n and n_2 (say) are to be allocated to a set of machines m_1, m_2 and m_3 (say).

We assume that each job requires some operations to be done on all these machines in a fixed order say, M_1, M_2 and M_3. Now, what should be the

schedule of the jobs (J_1 - J_2) or (J_2 - J_1), so that the completion time of both the jobs, called the make-span, is minimized? Let the processing time of jobs $J1$ and $J2$ on machines M_1, M_2 and M_3 be (5, 8, 7) and (8, 2, 3) respectively. The Gantt charts in fig. (a) and (b) describe the make-spans for the schedule of jobs J_1 - J_2 and J_2 - J_1 respectively. It is clear from these figures that J_1-J_2 schedule requires less make-span and is thus preferred.

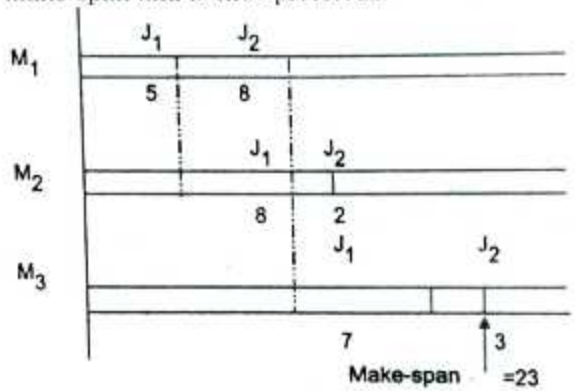


Figure : The $J_1 - J_2$ Schedule

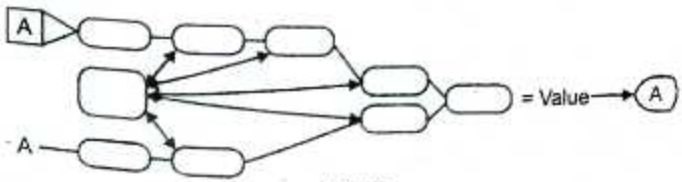
Intelligent Control : In process control, the controller is designed from the known models of the process and the required control objective. When the dynamics of the plant is not completely known, the existing techniques for controller design no longer remain valid. Rule-based control is appropriate in such situations. In a rule-based control system, the controller is realized by a set of production rules intuitively set by an expert control engineer. The antecedent (premise) part of the rules in a rule-based system is searched against the dynamic response of the plant parameters. The rule whose antecedent part matches with the plant response is selected and fired. When more than one rule is firable, the controller resolves the conflict by a set of strategies. On the other hand, there exist situations when the antecedent part of no rules exactly matches with the plant responses.

Figure : The Gantt Charts for the Flow-shop Scheduling Problem with 2 Jobs and 3 Machines

Such situations are handled with fuzzy logic, which is capable of matching the antecedent parts of rules partially/ approximately with the dynamic plant responses. Fuzzy control has been successfully used in many industrial plants. One typical application is the power control in a nuclear reactor. Besides design of the controller, the other issue in process control is to design a plant (process) estimator, which attempts to follow the response of the actual plant, when both the plant and the estimator are jointly excited by a common input signal. The fuzzy and artificial neural network-based learning techniques have recently been identified as new tools for plant estimation.

General Intelligence : GI on the other hand is a system where the domain is unknown and the GI process has to figure out the correct choices. The training process is the same as teaching any animal (human) in that the developer must allow the GI system to review the input(s) and then tell it when it has a correct outcome.

AGI



Figure

Input 1 -> cortical correlates -> outcome
Input 2 /

With GI, the domain is everything we present as inputs and there are no pre-defined outcome. The system

learns when we tell it that it is correct and solidifies the neuronal correlates. This requires a means to give positive and negative reinforcement and can have less than optimal results.

Difference Between Artificial and Natural Intelligence : Artificial intelligence (AI) is the established name for the field, but the term "artificial intelligence" is a source of much confusion because artificial intelligence may be interpreted as the opposite of real intelligence. For any phenomenon, you can distinguish real versus fake, where the fake is non-real. You can also distinguish natural versus artificial. Natural means occurring in nature and artificial means made by people.

Natural intelligence (NI) is the opposite of artificial intelligence : it is all the systems of control present in biology. Normally when we think of NI we think about how animal or human brains function, but there is more to natural intelligence than neuroscience. Nature also demonstrates non-neural control in plants and protozoa, as well as distributed intelligence in colony species like ants, hyenas and humans. Our behaviour co-evolves with the rest of our bodies, and in response to our changing environment. Understanding natural intelligence requires understanding all of these influences on behaviour and their interactions.

One of the best methods for understanding how NI systems work is to try to replicate their behaviour in simulation. Just as learning to paint forces you to understand the details of what you are seeing, building a working model forces you to understand the intricacies of what the target intelligent system is doing. For example :

(1) What environment does it work in?
(2) What aspects of that environment does it rely on?
(3) What does it need to do itself?
(4) How much does it need to learn and remember?
(5) What can it learn just from its senses?
(6) How much does it need to innovate?

Difference between Natural Intelligence and Artificial Intelligence :

Features	Natural Intelligence	Artificial Intelligence
Sensor using capability.	High	Low
Creative and imaginative ability.	High	Low
Ability to learn from the past experiences.	High	Low
Adaptive ability.	High	Low
Ability to afford the cost of the acquiring intelligence.	High	Low
Ability to use the information source varieties.	High	High
Ability of acquiring the nigh amount of the external information.	High	High
Ability of making the complex calculations.	Low	High
Information transferring ability.	Low	High
Ability to make a series of the different types of the calculations at a good high speed and all this in a very accurate way.	Low	High

Q.5. Discuss the historical development of artificial intelligence. (2017-18)
OR **Write down the foundation of AI.** (2014-15)
OR **What do you understand by History of Artificial Intelligence?**

Ans. The History of Artificial Intelligence :

The Gestation of Artificial Intelligence (1943-1955) : There were a number of early examples of work that can be characterized as AI, but it was Alan Turing who first articulated a complete vision of AI in his 1950 article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.

The Birth of Artificial Intelligence (1956) : McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help him bring together U.S. researchers interested in automata theory, neural nets, and the study of intelligence. They organized a two-month workshop at

Dartmouth in the summer of 1956. Perhaps the longest-lasting thing to come out of the workshop was an agreement to adopt McCarthy's new name for the field : artificial intelligence.

Early Enthusiasm, Great Expectations (1952-1969) : The early years of AI were full of successes-in a limited way.

General Problem Solver (GPS) : GPS was a computer program created in 1957 by Herbert Simon and Allen Newell to build a universal problem solver machine. The order in which the program considered sub-goals and possible actions was similar to that in which humans approached the same problems. Thus, GPS was probably the first program to embody the "thinking humanly" approach.

Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute of Technology (MIT). In 1963, McCarthy started the AI lab at Stanford. Tom Evans's ANALOGY program (1968) solved geometric analogy problems that appear in IQ tests, such as the one in Figure

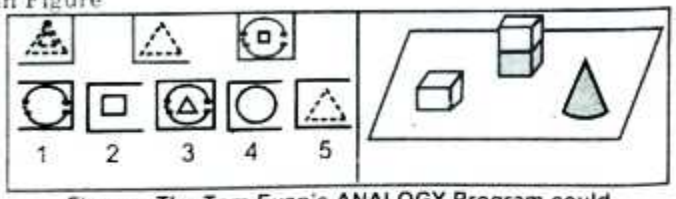


Figure : The Tom Evan's ANALOGY Program could Solve Geometric Analogy Problems as Shown

A Dose of Reality (1966-1973) : From the beginning, AI researchers were not shy about predictions of their coming successes. The following statement by Herbert Simon in 1957 is often quoted

"It is not my aim to surprise or shock you-but the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until-in a visible future-the range of problems they can handle will be coextensive with the range to which the human mind has been applied."

Knowledge-based Systems : The Key to Power? (1969-1979) : Dendral was an influential pioneer project in

artificial intelligence (AI) of the 1960s, and the computer software expert system that it produced. Its primary aim was to help organic chemists in identifying unknown organic molecules, by analyzing their mass spectra and using knowledge of chemistry. It was done at Stanford University by Edward Feigenbaum, Bruce Buchanan, Joshua.

AI Becomes an Industry (1980-present) : In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build intelligent computers running PROLOG. Overall, the AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988.

The Return of Neural Networks (1986-present) : Psychologists including David Rumelhart and Geoff Hinton continued the study of neural-net models of memory.

AI Becomes a Science (1987-present) : In recent years, approaches based on Hidden Markov Models (HMMs) have come to dominate the area. Speech technology and the related field of handwritten character recognition are already making the transition to widespread industrial and consumer applications.

The Emergence of Intelligent Agents (1995-present) : One of the most important environments for intelligent agents is the Internet.

Q.6. What do you understand by Natural Language Processing? (2015-16)
***OR* What is natural language processing? (2014-15)**
***OR* Discuss some most prominent applications of Artificial Intelligence and describe natural language processing in brief.**

Ans. Artificial intelligence is the study of how to make computers do things intelligent is the aim of AI. Some of the prominent application of AI are :

(1) Perception (both vision and speech)
(2) Natural language processing (Understanding, generation and translation)
(3) Common sense reasoning
(4) Robotics
(5) Game Playing
(6) Expert takes such as engineering (design, fault finding, manufacturing planning) Scientific analysis,

AI has changed our lives. Be it our mobile devices, washing machines, electrical appliances etc, all are making use of AI and neural network techniques. Applications have already been proven in such areas as medicine, law, economics, banking, defense and aerospace. Natural language processing is one of the prime applications of AI. It covers the processing of both written text and spoken (speech) language.

The steps involved in NLP are :

(1) **Morphological Analysis :** Individual words are analyzed into their components and non-taken words such as punctuation are separated from the words.

(2) **Syntactic Analysis :** Syntax is checked against the grammar rules. Parse tree is generated and sentences with wrong syntactic structures are rejected.

(3) **Semantic Analysis :** After syntactic analysis is done, semantic analysis is done to check if the meaning of the sentence is correct. Ex : Monkey ate the Banana and Banana ate the Monkey. Both are syntactically correct but semantically Banana ate the monkey is incorrect. Semantic analysis rejects such type of wrong meaning sentences.

(4) **Discourse Integration :** The meaning of an individual sentence may depend on the meaning of sentences that follow it. Example, "John wanted it". The meaning of 'it' depends on the prior sentences.

(5) **Pragmatic Analysis :** This analysis determines what the real meaning of the sentence is.

These steps are involved in the process of natural language processing.

Q.7 Describe the role of computer vision in artificial intelligence. (2015-16)

OR **Write a short note on Computer vision.** (2014-15)

Ans. Computer Vision : Accurate machine vision opens up a new realm of computer application. These applications include mobile robot navigator, complex manufacturing tasks, analysis of satellite images and medical image processing. A video camera provides a computer with an image represented as 2*D* grid of intensity levels. Each grid element or pixel may store a single bit of information or many bits (color information). A visual image is composed

(1) **Signal Processing :** Enhancing the image

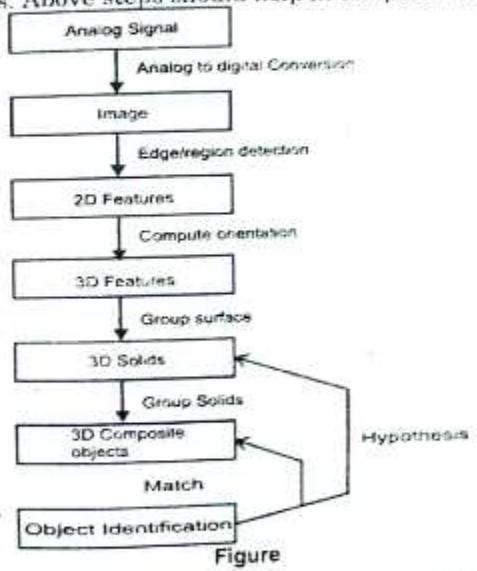
(2) **Measurement Analysis :** For images containing a single object, DETERMINING the 2*D* extent of object depicted

(3) Patten recognition

(4) Image understanding.

Here is one possible architecture for image understanding.

Computer vision is a hard and difficult process. 2*D* images are often ambiguous making vision a difficult process. Above steps should help in computer vision.



Figure

Q.8. Define and differentiate between weak and strong AI. (2014-15)

Ans. Difference between Weak and Strong AI : Weak AI and Strong AI are two types of AI, classified based on the goals that the corresponding sets of researchers are focused on achieving.

(1) Weak AI is focused towards the technology which is capable of carrying out pre-planned moves based on some rules and applying these to achieve a certain goal but, Strong AI is based on coming up with a technology that can think and function very similar

(2) So, the applications of Weak AI make the humans feel as that the machines are acting intelligently but they are not. Contrastingly, the applications of Strong AI will someday actually act and think just as a human, as opposed to just making the humans feel that the machines are intelligent.

Strong Artificial Intelligence : Strong artificial intelligence research deals with the creation of some form of computer-based artificial intelligence that can truly reason and solve problems; a strong form of AI is said to be sentient, or self-aware. In theory, there are two types of strong AI :

(1) Human-like AI, in which the computer program thinks and reasons much like a human mind.

(2) Non-human-like AI, in which the computer program develops a totally non-human sentience, and a non-human way of thinking and reasoning.

❑❑

UNIT – TWO

INTRODUCTION TO SEARCH

Q.1. What do you infer from hill-climbing search algorithm? (2018-19)

OR **Explain Steepest-ascent hill climbing algorithm. What are the problems with hill climbing algorithm?** (2017-18)

OR **When will Hill climbing search technique fail? Do steepest ascent hill climbing always find solutions? How might some problem be overcome in search?**

Ans. Steepest Ascent Hill Climbing : Steepest Hill climbing algorithm by choosing the best successor rather than the first successor that is better. This indicates that it has elements of the breadth first algorithm.

The algorithm proceeds Steepest ascent Hill climbing algorithm

(1) Evaluate the initial state.

(2) If it is goal state. Then quit otherwise make the current state this initial state and proceed.

(3) Repeat

(a) Set Target to be the state that any successor of the current state can better.

(b) For each operator that can be applied to the current state apply the new operator and create a new state

(c) Evaluate this state

(d) If this state is goal state Then quit

(e) Otherwise compare with Target

(f) If better set Target to this value

(g) If Target is better than current state set current state to Target

(h) Until a solution is found or current state does not change

Hill Climbing Technique and Steepest Hill Climbing Technique Fails When : In simple hill climbing, the first closer node is chosen, whereas in steepest ascent hill climbing all successors are compared and the closest to the solution is chosen. Both forms fail if there is no closer node, which may happen if there are local maxima in the search

Steepest ascent hill climbing is similar to best-fi search, which tries all possible extensions of the curre path instead of only one. So, steepest hill climbi technique also does not find solution always.

Other Techniques to Overcome the Problem : T other methods through which some of the problems mig be solved are :

Stochastic Hill Climbing : Stochastic hill climbing do not examine all neighbors before deciding how to mov Rather, it selects a neighbor at random, and decides (base on the amount of improvement in that neighbor) wheth to move to that neighbor or to examine another.

Random-Restart Hill Climbing : Random-restart h climbing is a meta-algorithm built on top of the hi climbing algorithm. It is also known as Shotgun hi climbing. It iteratively does hill-climbing, each time with random initial condition x_0. The best x_m is kept : if a ne run of hill climbing produces a better x_m than the store state, it replaces the stored state.

Random-restart hill climbing is a surprising effective algorithm in many cases. It turns out that it i often better to spend CPU time exploring the space, tha carefully optimizing from an initial condition.

Q.2. Trace the constraint satisfaction procedure to solv the following cryptarithmetic problem : (2018-19

CROSS
+ROADS
DANGER

Ans. Many problems in AI can be considered as problems o constraint satisfaction, in which the goal state satisfies a given set of constraint. constraint satisfaction problems can be solved by using any of the search strategies. The general form of the constraint satisfaction procedure is as follows :

Until a complete solution is found or until all paths have led to lead ends, do.

(1) Select an unexpanded node of the search graph.

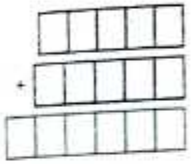
(2) Apply the constraint inference rules to the selected node to generate all possible new constraints.

(3) If the set of constraints contains a contradiction, then report that this path is a dead end.

(4) If the set of constraints describes a complete solution then report success.

(5) If neither a constraint nor a complete solution has been found then apply the rules to generate new partial solutions. Insert these partial solutions into the search graph.

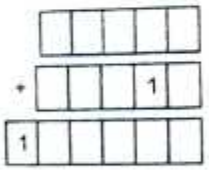
Cryptarithmetic Problem



	C5	C4	C3	C2	C1	
		C	R	O	S	S
+		R	O	A	D	S
D	A	N	G	E	R	

Character	Code
D	
C	
R	
A	
O	
N	
G	
S	
E	

Cryptarithmetic Problem



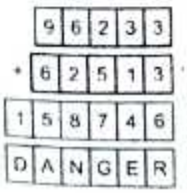
	C5	C4	C3	C2	C1	
		C	R	O	S	S
+		R	O	A	D	S
D	A	N	G	E	R	

Character	Code
D	1
C	
R	
A	
O	
N	
G	
S	
E	

SUM of two same numbers are EVEN

S + S = R, so R is Even R = 0
R = 2
R = 4
R = 6
R = 8

Cryptarithmetic Problem



	C5	C4	C3	C2	C1	
		C	R	O	S	S
+		R	O	A	D	S
D	A	N	G	E	R	

Character	Code
D	1
C	9
R	6
A	5
O	2
N	8
G	7
S	3
E	4

Figure

So the final answer is :

CROSS	96233
ROADS	62513
DANGER	158746

Q.3. Implement the Search Algorithms described in the lecture in LISP and/or C. Comment on how suitable each language would be for each type of search.

Ans. The Search algorithms in LISP is implemented as : (2018-19)

Mode : Lisp -*-

Simple Search Algorithms : Here we define the general search function, and then a set of search functions that follow specific search strategies. None of these algorithms worries about repeated states in the search.

```
(defun general-search (problem queuing-fn)
"Expand nodes according to the specification of PROBLEM until we find
a solution or run out of nodes to expand. The QUEUING-FN decides which
nodes to look at first. [p 73]"
(let ((nodes (make-initial-queue problem queuing-fn))
      node)
(loop (if (empty-queue? nodes) (RETURN nil))
(setq node (remove-front nodes))
(if (goal-test problem (node-state node)) (RETURN node))
(funcall queuing-fn nodes (expand node problem)))))
(defun breadth-first-search (problem)
"Search the shallowest nodes in the search tree first. [p 74]"
(general-search problem #'enqueue-at-end))
(defun depth-first-search (problem)
"Search the deepest nodes in the search tree first. [p 78]"
(general-search problem #'enqueue-at-front))
(defun iterative-deepening-search (problem)
"Do a series of depth-limited searches, increasing depth each time
[p 79]"
(for depth = 0 to infinity do
(let ((solution (depth-limited-search problem depth)))
(unless (eq solution :cut-off) (RETURN solution)))))
(defun depth-limited-search (problem &optional (limit infinity)
(node (create-start-node problem)))
"Search depth-first, but only up to LIMIT branches deep in the tree."
(cond ((goal-test problem node) node)
((>= (node-depth node) limit) :cut-off)
(t (for each n in (expand node problem) do
```



```
(defun best-first-search (problem eval-fn)
"Search the nodes with the best evaluation first. [p 93]"
(general-search problem #'(lambda (old-q nodes)
(enqueue-by-priority old-q nodes eval-fn))))
(defun greedy-search (problem)
"Best-first search using H (heuristic distance to goal). [p 93]"
(best-first-search problem #'node-h-cost))
(defun tree-a*-search (problem)
"Best-first search using estimated total cost, or (F = G + H). [p 97]"
(best-first-search problem #'node-f-cost))
(defun uniform-cost-search (problem)
"Best-first search using the node's depth as its cost. Discussion on
[p 75]"
(best-first-search problem #'node-depth))
;;;;Utility Function
(defun make-initial-queue (problem queuing-fn)
(let ((q (make-empty-queue)))
(funcall queuing-fn q (list (create-start-node problem)))
q))
```

There are a few reasons that helps in using Lisp.

(1) Homoiconic code. This allows structured self-modifying code.

(2) Syntax-aware macros. They allow rewriting of boilerplate code.

(3) Pragmatism. Common Lisp is designed to get stuff done by working professionals. Most functional languages aren't, as a rule.

(4) Flexibility. It can do a lot of different things, all at reasonable speeds.

(5) Wariness. The real world is messy. Pragmatic coding winds up having to either use or invent messy constructs. Common Lisp has sufficient wariness that it can get stuff done.

Q.4. Discuss how constraint satisfaction might work it implemented its search strategy via. (2018-19)

(1) Depth first search

(2) Breadth first search

(3) Best first search

OR **Discuss Simulated Annealing search algorithm with its advantages and disadvantages.** (2017-18)

OR **Give an example of a problem for which breadth first search would work better than depth first search. Write the difference between these two approaches.** (2015-16)

OR **What do you mean by constraint satisfaction problem? Explain constraint satisfaction algorithm.** **(2014-15)**

OR **Explain various types of heuristic.**

Ans. Constraint satisfaction problems (CSPs) are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over a variable, which is solved by constraint satisfaction methods. CSPs are the subject of intense research in both artificial intelligence and operations research, since the regularity in their formulation provides a common basis to analyze and solve problems of many seemingly unrelated families. CSPs often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods to be solved in a reasonable time. The Boolean satisfiability problem (SAT), the satisfiability modulo theories (SMT) and answer set programming (ASP) can be roughly thought of as certain forms of the constraint satisfaction problem.

Examples of simple problems that can be modeled as a constraint satisfaction problem

(1) Eight queens puzzle
(2) Map coloring problem
(3) Sudoku, Futoshiki, Kakuro (Cross Sums), Numbrix, Hidato and many other logic puzzles

Backtracking Algorithm :

Function BACKTRACKING
Input : A CSP (X, D, C)
Output : Either a solution, or a decision that the CSP has no solution

```
d[i] ← Di for 1 <i <n (copy all domains)
i ← 1 (initialize variable counter)
while 1 <= i <= n do
reorder variables xi
                xn according to variable ordering heuristic (select variable)
xi ← SELECT-VALUE (select value)
di ← xi
if xi = null then (no value left in domain)
if i = 1 then
restore di to its state before xi = I was last instantiated
i ← i - 1 (backtrack)
else
i ← i + 1 (step forward)
end while
if i = 0 then
return "no solution"
else
return instantiated values of {x1, ..., xn}
```


(1) **Depth First Search :** The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows :

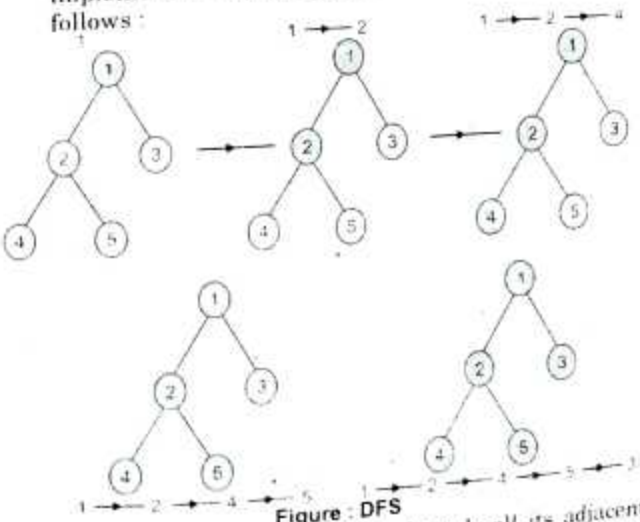


Figure : DFS

Pick a starting node and push all its adjacent nodes into a stack.

Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.

Repeat this process until the stack is empty. However, ensure that the nodes that are visited are

marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

Time Complexity O(V + E), When Implemented Using and Adjacency List.

(2) Breadth First Search : Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. It uses the opposite strategy as depth-first search, which instead explores the highest-depth nodes first before being forced to backtrack and expand shallower nodes.

There are many ways to traverse graphs. BFS is the most commonly used approach.

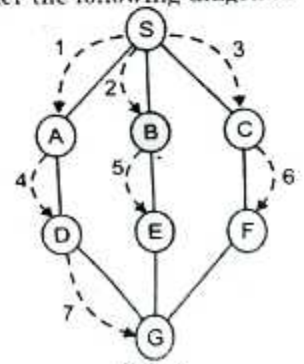
BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer wise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

As the name BFS suggests, you are required to traverse the graph breadth wise as follows :

(a) First move horizontally and visit all the nodes of the current layer

(b) Move to the next layer

Consider the following diagram.



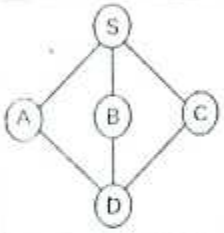
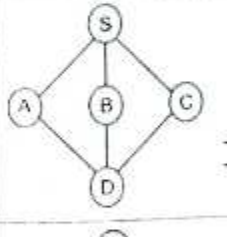
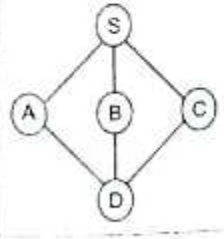
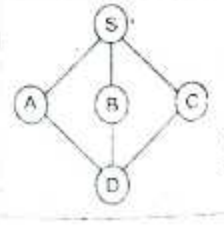
Figure

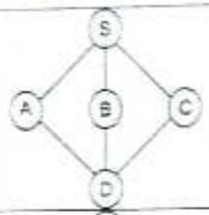
As in the example given above, BFS algorithm traverses from A to B to E to F first then to G and lastly to D. It employs the following rules.

Rule 1 : Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.

Rule 2 : If no adjacent vertex is found, remove the first vertex from the queue.

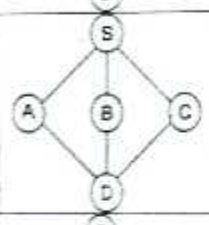
Rule 3 : Repeat Rule 1 and Rule 2 until the queue is empty.

Step	Traversal	Description
1	 Queue	Initialize the queue.
2	 Queue	We start from visiting S (starting node), and mark it as visited.
3	 A Queue	We then see an unvisited adjacent node from S. In this example, we have three nodes but alphabetically we choose A, mark it as visited and enqueue it.
4	 B A Queue	Next, the unvisited adjacent node from S is B. We mark it as visited and enqueue it.



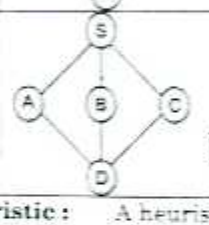
C B A
Queue

Next, the unvisited adjacent node from S is C. We mark it as visited and enqueue it.



C B
Queue

Now, S is left with no unvisited adjacent nodes. So we dequeue and find A.



D C B
Queue

From A we have D as unvisited adjacent node. We mark it as visited and enqueue it.

Heuristic : A heuristic is method that

(1) Might not always find the best solution.

(2) But is guaranteed to find a good solution in reasonable time.

(3) By sacrificing completeness it increases efficiency.

(4) Useful in solving tough problems.

The classic example of heuristic search methods is the traveling salesman problem. Various types of heuristics are the following :

Heuristic Search Methods Generate and Test Algorithm :

(1) Generate a possible solution which can either be a point in the problem space or a path from the initial state.

(2) Test to see if this possible solution is a real solution by comparing the state reached with the set of goal states.

(3) If it is a real solution, return Otherwise repeat from 1.

(4) This method is basically a depth first search as complete solutions must be created before testing. It is often called the British Museum method as it is like looking for an exhibit at random. A heuristic is needed to sharpen up the search. Consider the

problem of four 6-sided cubes, and each side of the cube is painted in one of four colours. The four cubes are placed next to one another and the problem lies in arranging them so that the four available colours are displayed whichever way the 4 cubes are viewed. The problem can only be solved if there are at least four sides coloured in each colour and the number of options tested can be reduced using heuristics if the most popular colour is hidden by the adjacent cube.

Hill Climbing : Here the generate and test method is augmented by an heuristic function which measures the closeness of the current state to the goal state.

(1) Evaluate the initial state if it is goal state quit otherwise current state is initial state.

(2) Select a new operater for this state and generate a new state.

(3) Evaluate the new state
   (a) If it is closer to goal state than current state makes it current state.
   (b) If it is no better ignore

(4) If the current state is goal state or no new operators available, quit. Otherwise repeat from 2.

In the case of the four cubes a suitable heuristic is the sum of the number of different colours on each of the four sides, and the goal state is 16 four on each side. The set of rules is simply choose a cube and rotate the cube through 90 degrees. The starting arrangement can either be specified or is at random.

Simulated Annealing : This is a variation on hill climbing and the idea is to include a general survey of the scene to avoid climbing false foot hills. The whole space is explored initially and this avoids the danger of being caught on a plateau or ridge and makes the procedure less sensitive to the starting point. There are two additional changes, we go for minimization rather than creating maxima and we use the term objective function rather than heuristic. It becomes clear that we are valley descending rather than hill climbing. The title comes from the metallurgical process of heating metals and then letting them cool until they reach a minimal energy steady final state. The probability that the metal will jump to a higher energy level is given by $p = e^{\delta E / kT}$ where k is

Boltzmann's constant . The rate at which the system is cooled is called the annealing schedule. ΔE is called the change in the value of the objective function and kT is called T a type of temperature. An example of a problem suitable for such an algorithm is the traveling salesman.

The Simulated Annealing algorithm is based upon the physical process which occurs in metallurgy where metals are heated to high temperatures and are then cooled. The rate of cooling clearly affects the finished product. If the rate of cooling is fast, such as when the metal is quenched in a large tank of water the structure at high temperatures persists at low temperature and large crystal structures exist, which in this case is equivalent to a local maximum. On the other hand if the rate of cooling is slow as in an air based method then a more uniform crystalline structure exists equivalent to a global maximum. The probability of making a large uphill move is lower than a small move and the probability of making large moves decreases with temperature. Downward moves are allowed at any time.

(1) Evaluate the initial state.
(2) If it is goal state then quit otherwise make the current state this initial state and proceed.
(3) Make variable best state to current state.
(4) Set temperature T according to the annealing schedule.
(5) Repeat
ΔE=difference between the values of current and new state.
(1) If this new state is goal state then quit.
(2) Otherwise compare with the current state.
(3) If better set best state to the value of this state and make this current the new state.
(4) If it is not better then make it the current state with probability p'. This involves generating a random number in the range 0 to 1 and comparing it with p. If it is less than p then do nothing and if it is greater than p then accept this state as the next current be exact.
(5) Revise T as the annealing schedule dictates.
(6) If [illegible]

Best First Search : A combination of depth first and breadth first searches. Depth first is good because a solution can be found without computing all nodes and breadth first is good because it does not get trapped in dead ends. The best first search allows us to switch between paths thus gaining the benefit of both approaches. At each step the most promising node is chosen.

If one of the nodes chosen generates nodes that are less promising it is possible to choose another at the same level and in effect the search changes from depth to breadth. If on analysis these are no better than this previously unexpanded node and branch is not forgotten and the search method reverts to the descendants of the first choice and proceeds, backtracking as it were.

This process is very similar to steepest, but in hill climbing once a move is chosen and the others rejected are never reconsidered whilst in best first they are saved to enable revisits if an impasses occurs on the apparent best path. Also the best available state is selected in best first even its value is worse than the value of the node just explored whereas in hill climbing the progress stops if there are no better successor nodes.

The best first search algorithm will involve on OR graph which avoids the problem of node duplication and assumes that each node has a parent link to give the best node from which it came and a link to all its successors. In this way if a better node is found this path can be propagated down to the successors.

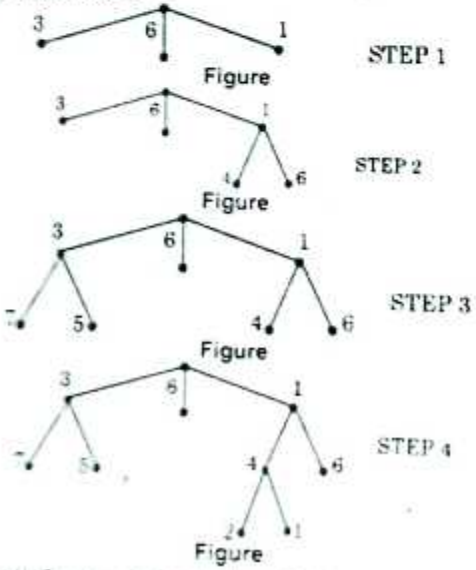
This method of using an OR graph requires 2 lists of nodes. OPEN is a priority queue of nodes that have been evaluated by the heuristic function but which have not yet been expanded into successors. The most promising nodes are at the front. CLOSED are nodes that have already been generated and these nodes must be stored because a graph is being used in preference to a tree.

Heuristics in order to find the most promising nodes a heuristic function is needed called f where f is an approximation to f and is made up of two parts g and h where g is the cost of going from the initial state to the current node. g is considered simply in this context to be the number of arcs traversed each of which is treated as being of unit weight. h is an estimate of the initial cost of getting form the current node to the goal state.

The function f is the approximate value or estimate of getting form the initial state to the goal state. Both g and h are positive valued variables. Best First: The Best First algorithm is a simplified form of the A^* algorithm. From A^* we note that $f = g + h'$ where g is a measure of the time taken to go from the initial node to the current node and h is an estimate of the time taken to solution from the current node. Thus f is an estimate of how long it takes to go from the initial node to the solution. As an and we take the time to go from one node to the next to be a constant at 1

Best First Search Algorithm :

(1) Start with OPEN holding the initial state
(2) Pick the best node on OPEN
(3) Generate its successors
(4) For each successor *Do*
  (a) If it has not been generated before evaluate it add it to OPEN and record its parent
  (b) If it has been generated before change the parent if this new path is better and in that case update the cost of getting to any successor notes
(5) If a goal is found or no more nodes left in OPEN, quit, else return to 2



All figures indicate "cost" of move

The A^* Algorithm : Best first search is a simplified A

(1) Start with OPEN holding the initial nodes
(2) Pick the BEST node on OPEN such that $f = g + h'$ is minimal
(3) If BEST is goal node quit and return the path from initial to BEST otherwise.
(4) Form OPEN and all of BESTs children, labeling each with its path from initial node.

Graceful Decay of Admissibility : If h' rarely overestimates h by more than d then the A^* algorithm will rarely find a solution whose cost is d greater than the optimal solution.

Advantage and Disadvantages of Simulated Annealing :

Advantages :

(1) Can find global optimum(given sufficient time)
(2) Well – suited for detailed placement.

Disadvantages :

(1) Very slow.
(2) To achieve high-quality implementation, laborious parameter tuning is necessary.
(3) Randomized, chaotic algorithms – small changes in the input lead to large changes in the output.

Q.5. Give the heuristic function for shortest path problem. (2018-19)

Ans. In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. $A*$ algorithm depends on the heuristics. $A*$ is faster and give good results if we have a good heuristic. The function is $f(n') = g(n') + h(n')$.

Q.6. You have three jugs measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You need to measure out exactly one gallon. (2018-19)

Ans. State Space : all configurations of the three jugs with different amounts of water in each jug. Initial State: 1 empty 12-gallon jug, 1 empty 8 gallon jug, 1 empty 3 gallon jug

Actions :

(1) **Action 1 :** Fill a jug from the faucet until the jug is

(2) **Action 2 :** Dump water from one jug A into anoth jug B until B is full

(3) **Action 3 :** Dump all the water from one jug onto t ground

Goal : A jug has 1 gallon of water in it and the other ju are empty

Cost : A reasonable approach would be to make the cost each action a unit cost so that all actions have the sam cost. Another possibility is to define the cost of an action be equal to the number of gallons of water transferred b the action. Furthermore, one could think of making th cost of Action 3 (dumping water onto the group) higher t try to minimize wasting water.

Q.7. Discuss Branch-and-bound search algorithm.
(2017-18

***OR* State and explain the Branch-and-bound searchin technique.** **(2014-15**

Ans. Branch and bound (BB or B&B) is an algorithm desig paradigm for discrete and combinatorial optimizatio problems, as well as general real valued problems. A branch-and-bound algorithm consists of a systemati enumeration of candidate solutions by means of state spac search: the set of candidate solutions is thought of as forming a rooted tree with the full set at the root.

The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm. The algorithm depends on the efficient estimation of the lower and upper bounds of a region/branch of the search space and approaches exhaustive enumeration as the size (*n*-dimensional volume) of the region tends to zero.

Q.8. Differentiate between local search and global search.
(2017-18)

Ans. ***Difference between Local Search and Global Search :***

Local Search	Global Search
Single monolithic game state	Partition game into sum of subgames
Full board evaluation	Local analysis
Single game tree, minimax backup	Problems : how to evaluate local results

Q.9. What are the steps to define a problem? Explain. Also discuss various components of a problem.
(2017-18)

Ans. Steps to Define a Problem : Defining problems is simple and any difficulty that arises is because it requires patience, repetition and thorough examination. It is the most important element of critical thinking.

You can define proble s correctly in just three steps :

(1) **Explore the Current Situation** : Paint a picture in words by including the "presenting problem," the impact it is having, the consequences of not solving the problem, and the emotions the problem is creating for those involved.

(2) **Explain :** Once you have examined and clearly explained the situation, draft a simple problem statement by filling in the blank The problem that we are trying to solve is Distill the problem to its simplest form possible.

(3) **Ask Yourself :** "Why is that a problem?" If the answer is another problem, then congratulate yourself for moving from the "presenting problem" to a deeper problem. Then ask yourself again, "Why is that a problem?" Do that repeatedly until you either land on what is obviously the source of all of the problems you've identified or you identify unexpected consequences of not solving the problem. If you land on unexpected consequences, the problem you identified right before that is likely your "source problem."

Problem Solving : The key Components of the process of problem solving are :

(1) **The Problem :** A situation or scenario creating difficulty.

(2) **The Answer :** A remedy for the difficulty that the problem requires.

(3) **The Method :** The means used to get an answer.

(4) **The Solution :** The whole process of solving a problem, including the method of obtaining an answer and the answer itself.

Components of Problem : Three components of problem solving communication were assessed :

(1) Orientation to other's point of view,

(2) Communication of essential information

(3) Verification of solutions.

These components were used to account for communication accuracy and to reexamine previously reported relationships between children's social characteristics and communication accuracy.

Q.10. Explain the concept of Alpha-beta pruning. Write Alpha-beta search algorithm. (2017-18)

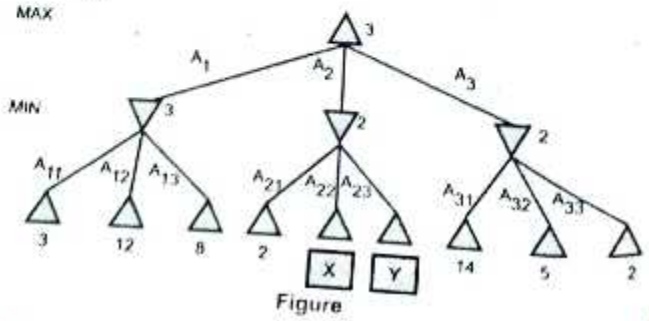
OR **Describe alpha-beta pruning and give the other modifications to the min-max procedure to improve its performance. (2015-16)**

OR **Write a short note on Alpha-beta pruning. (2014-15)**

OR **What do you understand by Adversarial Search with suitable examples?**

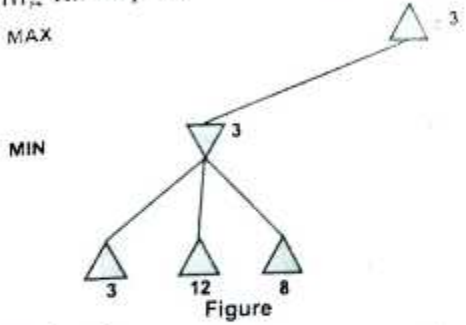
Ans. α-β Pruning : It is possible to compute the correct minimax decision without looking at every node in the game tree.

$$MINIMAX\text{-}VALUE(root) = \max(\min(3,12,8),\min(2,x,y),\min(14,5,2))$$
$$= \max(3, \min(2,x,y),2)$$
$$= \max(3,z,2) \quad \text{where } z <= 2$$
$$= 3$$



Figure

α-β Pruning Example :



Figure

Properties of α-β :

(1) Pruning does not affect final result

(2) Good move ordering improves effectiveness of pruning

(3) With "perfect ordering" time complexity = $O(b^{m/2})$
   (i) Doubles depth of search

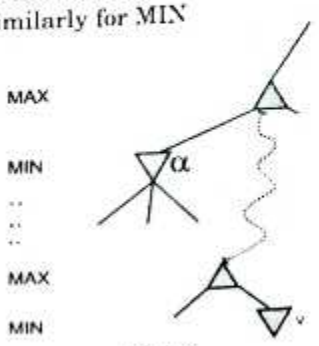
(4) A simple example of the value of reasoning about which computations are relevant (a form of meta-reasoning)

Why is It Called α-β :

(1) α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for MAX

(2) If v is worse than α, MAX will avoid it.
   Prune that branch

(3) Define β similarly for MIN



Figure

The α-β Algorithm :

function ALPHA-BETA-SEARCH(state) returns an action
inputs : state, current state in game

v← MAX-VALUE(state, -∞, +∞)
return the action in SUCCESSORS(state) with value v
function MAX-VALUE(state,α,β) returns a utility value
inputs : state, current state in game
α, the value of the best alternative for MAX along the path to state
β, the value of the best alternative for MIN along the path to state
if TERMINAL-TEST(state)then return UTILITY(state)
v← -∞
for a, s in SUCCESSORS(state) do
v← MAX(v, MIN-VALUE(s,α,β))
if v ≥β then return v
α ← MAX(α,v)
return v

Adversarial Search :

(1) Game Tree Search :

(a) Initial State : Initial board position and player

(b) Operators : One for each legal move

(c) Goal States : Winning board positions

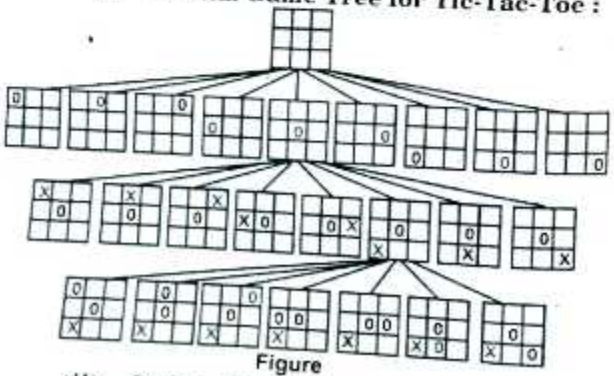
(d) Scoring Function : Assigns numeric value to states

(e) Game Tree : Encodes all possible games

(f) We are not looking for a path, only the next move to make (that hopefully leads to a winning position)

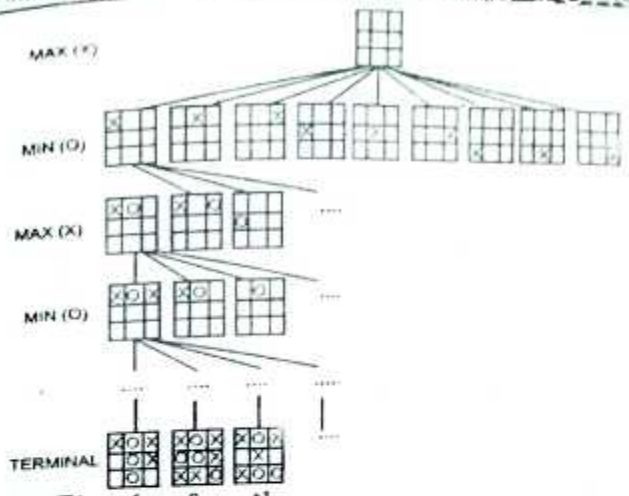
(g) Our best move depends on what the other player does

(i) Partial Game Tree for Tic-Tac-Toe :



Figure

(ii) Game Tree (2-player, Deterministic Turns) :



Figure

(2) Optimal Strategies :

(a) In a normal search problem, the optimal solution would be a sequence of moves leading to a goal state-a terminal state that is a win

(b) In a game, MIN has something to say about it and therefore MAX must find a contingent strategy, which specifies

(i) MAX's move in the initial state,

(ii) Then MAX's moves in the states resulting from every possible response by MIN,

(iii) Then MAX's moves in the states resulting from every possible response by MIN to those moves

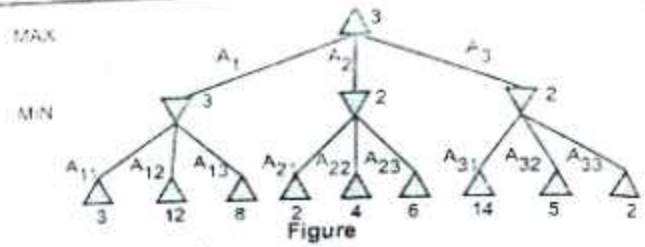
(c) An optimal strategy leads to outcomes at least as good as any other strategy when one is playing an infallible opponent

(3) Minimax :

(a) Perfect play for deterministic games

(b) Idea : Choose move to position with highest minimax value = Best achievable payoff against best play

(c) E.g., 2-ply game :



Figure

(4) Minimax-Value :

(a) Given a game tree, the optimal strategy can be determined by examining the minimax value of each node (MINIMAX-VALUE(n))

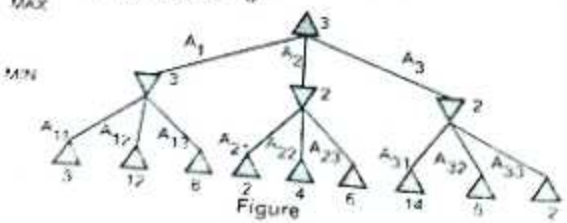
(b) The minimax value of a node is the utility of being in the corresponding state, assuming that both players play optimally from there to the end of the game

(c) Given a choice, MAX prefer to move to a state of maximum value, whereas MIN prefers a state of minimum value

(5) Minimax Algorithm :

```
function MINIMAX-DECISION(state) returns an action
v← MAX-VALUE(state)
return the action in SUCCESSORS(state) with value v
function MAX-VALUE(state) returns a utility value
if TERMINAL-TEST(state)then return UTILITY(state)
v← -∞
for a, s in SUCCESSORS(state) do
v←MAX(v, MIN-VALUE(s))
return v
function MIN-VALUE(state) returns a utility value
if TERMINAL-TEST(state)then return UTILITY(state)
v← ∞
for a, s in SUCCESSORS(state) do
v← MIN(v, MIN-VALUE(s))
return v
```

(6) Minimax : MINIMAX-VALUE(root) = max(min(3, 12, 8), min(2, 4, 6), min(14, 5, 2))
= max(3, 2, 2) = 3



Figure

(7) Properties of Minimax :

(a) Complete? Yes (if tree is finite)

(b) Optimal? Yes (against an optimal opponent)

(c) Time Complexity? $O(b^m)$

(d) Space Complexity? $O(bm)$ (depth-first exploration)

(e) For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games. Exact solution completely infeasible

Q.11. What is "overfitting"? How do we overcome overfitting? (2015-16)

Ans. Overfitting : In overfitting, a statistical model describes random error or noise instead of the underlying relationship. Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the training data.

The possibility of overfitting exists because the criterion used for training the model is not the same as the criterion used to judge the efficacy of a model. In particular, a model is typically trained by maximizing its performance on some set of training data. However, its efficacy is determined not by its performance on the training data but by its ability to perform well on unseen data. Overfitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from trend. As an extreme example, if the number of parameters is the same as or greater than the number of observations, a simple model or learning process can perfectly predict the training data simply by memorizing the training data in its entirety, but such a model will typically fail drastically when making predictions about new or unseen data, since the simple model has not learned to generalize at all.

The potential for overfitting depends not only on the number of parameters and data but also the conformability of the model structure with the data shape, and the magnitude of model error compared to the expected level of noise or error in the data.

Even when the fitted model does not have an excessive number of parameters, it is to be expected that the fitted relationship will appear to perform less well on a new data set than on the data set used for fitting. In

particular, the value of the coefficient of determination will shrink relative to the original training data.

In order to avoid overfitting, it is necessary to use additional techniques (e.g. cross-validation, regularization, early stopping, pruning, Bayesian priors on parameters or model comparison), that can indicate when further training is not resulting in better generalization. The basis of some techniques is either to explicitly penalize overly complex models, or to test the model's ability to generalize by evaluating its performance on a set of data not used for training, which is assumed to approximate the typical unseen data that a model will encounter.

To Avoid Overfitting : To avoid overfitting add the regularization if there are many features. Regularization forces the magnitudes of the parameters to be smaller (shrinking the hypothesis space). For this add a new term to the cost function

$$J(\theta) = 1/2M \sum_{i=1}^{M} (h(x^i) - y^i)^2$$

Which penalizes the magnitudes of the parameters like as.

Q.12. Describe A* search technique. Prove that A* is complete and optimal. **(2015-16)**

OR **Explain *A** algorithm with using the Tower of Hanoi Problem.**

Ans. The *A Algorithm :** Given a suitable problem, we represent the initial conditions of the problem with an appropriate initial state, and the goal conditions as the goal state. For each action that is performed, generate successor states to represent the effects of the action. If this continues at some point one of the generated successor states is goal state, then the path from the initial state to the goal state is the solution to the problem. What A-Star does is generate and process the successor states in a certain way. Whenever it is looking for the next state to process, A-Star employs a heuristic function to try to pick the best state to process next. If the heuristic function is good, not only will A-Star find a solution quickly, but it can also find the best solution possible.

Working of A* :

(1) The algorithm requires two lists :

(a) **OPEN List :** The OPEN list keeps track of those

(b) **CLOSED List :** The CLOSED list keeps track of nodes that have already been examined.

(2) Initially, the OPEN list contains just the initial node, and the CLOSED list is empty. Each node n maintains the following :

$g(n)$ = The cost of getting from the initial node to n.

$h(n)$ = The estimate, according to the heuristic function, of the cost of getting from n to the goal node.

$f(n) = g(n) + h(n)$; intuitively, this is the estimate of the best solution that goes through n.

(3) Each node also maintains a pointer to its parent, so that later the best solution if found can be retrieved. A-Star has a main loop that repeatedly gets the node, call it n, with the lowest $f(n)$ value from the OPEN list. If n is the goal node, then we are done, and the solution is given by backtracking from n. Otherwise, n is removed from the OPEN list and added to the CLOSED list. Next all the possible successor nodes of n are generated.

(4) For each successor node n, if it is already in the CLOSED list and the copy there has an equal or lower f estimate, and then we can safely discard the newly generated n and move on. Similarly, if n is already in the OPEN list and the copy there has an equal or lower f estimate, we can discard the newly generated n and move on.

If no better version of n exists on either the CLOSED or OPEN lists, we remove the inferior copies from the two lists and set n as parent of n. We also have to calculate the cost estimates for n as follows :

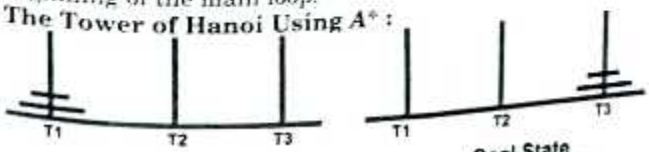
Set $g(n)$ to $g(n)$ plus the cost of getting from n to n.

Set $h(n)$ to the heuristic estimate of getting from n to the goal node.

Set $f(n)$ to $g(n)$ plus $h(n)$.

Lastly, add n to the OPEN list and return to the beginning of the main loop.

The Tower of Hanoi Using *A :**



Figure

There are three disks placed on the tower according to order of size as shown in an above figure. All disks should be moved in the same order to the third tow using second tower as a secondary media for transf Above figure shows three towers T_1, T_2, T_3. In the ini state all the three disks are on the first tower. They are be moved to third tower. T_2 can be used as intermedia tower for transfer. The transferring condition is that a bigger disk should not be placed on the smaller one dur transfer.

Figure below shows the state space representation the problem. The graph has 27 nodes each represents one of legal configurations of disks on pegs. The notatio (*ijk*) labeling each node describes a state in which disk (the largest) is on tower 1, disk B on T_2 and disk A is smallest) on T_3. If more than one disk is on the same tow than it is assumed that the largest is in the bottom. T arcs connecting pairs of node mean that a disk can transferred such that the configuration represented by a node in the pair is changed to that represented by th other.

(111) : All the three disks are on tower 1

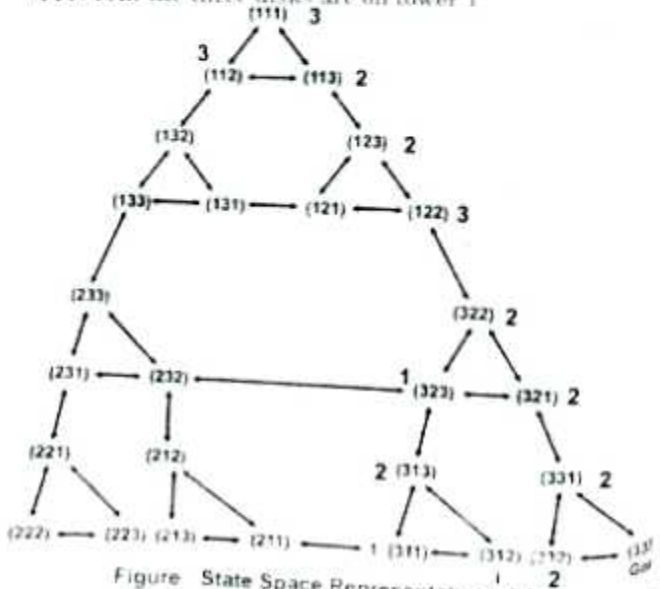


Figure State Space Representation of The Tower of Hanoi Problem

Q.13. Differentiate between simple hill climbing and steepest-ascent hill climbing algorithms. (2014-15)

Ans. These are both greedy local search algorithms, but are greedy in different ways. Classic hill climbing is more naive: it takes the first step that it encounters which improves the objective function. steepest ascent hill climbing examines all possible next steps and chooses the best one. In this way, it is analogous to the use of best-first search to traverse graphs. It is important to note that the fact that SAHC chooses the "best" next step at each iteration does not necessarily mean that it finds the global maximum. It has the same problem of languishing in local maxima as other hill climbing algorithms. As an aside, this is also true of best-first search, which uses a heuristic to pick the "most promising" next node.

In the case of SAHC, "most promising" is defined to be "steepest ascent." In depth-first search, you add the current node's children to the front of the queue (a stack). In breadth-first search, you add the current node's children to the back of the queue. Think for a moment about how this leads to the right behavior for those algorithms.

Now, in hill-climbing search, you sort the current node's children before adding them to the queue. In best-first search, you add the current node's children to the queue in any old order, and then sort the entire queue. If you think about the effect that might have on the order in which nodes are searched, you should get an idea of the practical difference.

Q.14. Discuss MIN-MAX algorithm for game playing. (2014-15)

Ans. The Min-Max algorithm is applied in two player games, such as tic-tac-toe, checkers, chess and go, and so on. All these games have at least one thing in common, they are logic games. This means that they can be described by a set of rules and premises. With them, it is possible to know from a given point in the game, what are the next available moves. So they also share other characteristic, they are "full information games". Each player knows everything about the possible moves of the adversary.

Before explaining the algorithm, a brief introduction to search trees is required. Search trees are a way to represent searches. You can a representation of a search tree. The squares are known as nodes and they represent

points of the decision in the search. The nodes are connected with branches. The search starts at the root node. At each decision point, nodes for the available search paths are generated, until no more decisions are possible. The nodes that represent the end of the search are known as leaf nodes.

There are two players involved, MAX and MIN. A search tree is generated, depth-first, starting with the current game position upto the end game position. Then, the final game position is evaluated from MAX's point of view. Afterwards, the inner node values of the tree are filled bottom-up with the evaluated values. The nodes that belong to the MAX player receive the maximum value of its children. The nodes for the MIN player will select the minimum value of its children. The algorithm is described here.

So what is happening here? The values represent how good a game move is. So the MAX player will try to select the move with highest value in the end. But the MIN player also has something to say about it and he will try to select the moves that are better to him, thus minimizing MAX's outcome.

Algorithm for Game Playing :

```
MinMax (GamePosition game){
return MaxMove (game);
}
MaxMove (GamePosition game) {
if (GameEnded(game)) {
return EvalGameState(game);
}
else {
best_move <- {};
moves <- GenerateMoves(game);
ForEach moves {
move <- MinMove(ApplyMove(game));
if (Value(move) > Value(best_move)) {
best_move <- move;
}
}
return best_move;
}
}
```



```
MinMove (GamePosition game) {
best_move <- {};
moves <- GenerateMoves(game);
ForEach moves {
move <- MaxMove(ApplyMove(game));
if (Value(move) > Value(best_move)) {
best_move <- move;
}
}
return best_move;
```

Q.15. Write a short note on Control strategy. (2014-15)

Ans. Control Strategy :

(1) How to decide which rule to apply next during the process of searching for a solution to a problem.

(2) Requirement for a good Control Strategy

It should Cause Motion : In water jug problem, if we apply a simple control strategy of starting each time from the top of rule list and choose the first applicable one, then we will never move towards solution. It should explore the solution space in a systematic manner. If we choose another control strategy, let us say, choose a rule randomly from the applicable rules then definitely it causes motion and eventually will lead to a solution. But one may arrive to same state several times. This is because control strategy is not systematic

(1) Breadth First Search

(2) Depth First Search

(3) Heuristic Search

Q.16. Explain searching for solutions.

Ans. Searching for Solutions : The typical way for solving problem in state subspace.

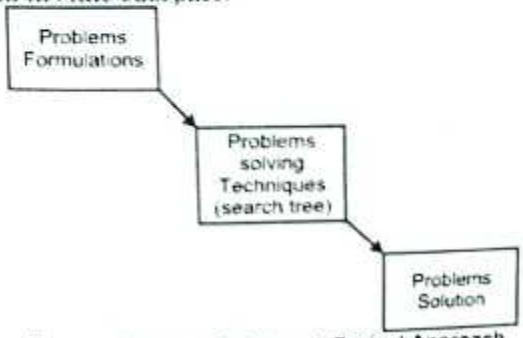


Figure : Problem Solving : A Typical Approach

Search Techniques :

(1) The objective of the search techniques is to search optimal solution for a specific problem.

(2) With the application of an explicit search tree which is generated by the initial state and successor module.

(3) The initial state and successor module are combined together which form the 'state space'.

(4) In broad sense we can use "a search graph", instead of "a search tree", for specific state.

(5) The specific state is reached from multiple paths.

(6) A search graph can be used instead of a search tree for specific application.

Search Strategy :

(1) The selection of one state, by keeping aside other state is the essence of search.

(2) The goal state cannot always be achieved in the first choice.

(3) The choice of state remain a question in problem solve. But the "search strategy" plays an important intrinsic role in state selection.

Tree Search Algorithm : The perfect search algorithm avoids expansion of respected paths.

The generalized tree search algorithm is as follows :

function TREE-SEARCH (problem, search strategy)
returns a solution (goal), or failure.
initialize the tree using the origin state and problem in (0).
loop 'do' if there is no node for expansion according to search strategy.
If the node contains a goal state then return the solution
else expand the node and add the generated nodes to the search tree.

Difference between the State Space and Search Tree :

	The State Space	The Search Tree
1.	The initial state and successor function implicitly are combined together, forms the state space.	The search tree is generated by the initial state and the successor function which are combined together defines the state space.
2.	The state space is represented in a graph.	The search tree is explicit tool for search techniques. The problem is represented in search tree form. Some time graphs may be used.
3.	The nodes represent states and the ones represent actions.	The search node represent states and expanding states generate a new set of states.

Node as a Data Structure : The node can be represented in various form, there are several ways to represent node. The node is a data structure. The node is represented with five components.

(1) **State :** It is the state in the state space, the node belongs.

(2) **Parent Node :** It is an node that generates n nodes the search tree.

(3) **Action :** It is the action implied to parent node to generate n nodes.

(4) **Path cost :** The path cost is shown by $g(n)$. The path cost of the path from the initial state.

(5) **Depth :** These are the number of steps required along the path from the initial state.

Nodes, States , Fringe and Leaf Node :

Node : A node is information keeping data structure which is used to represent the search tree.

State : A state is mapping a state action pair into state. It reflects world configuration.

(1) Nodes are projected by PARENT-NODE pointer. Nodes are present in typical path.

(2) States are not represented through PARENT-NODE.

Fringe :

(1) It is graphical, hierarchical structure of nodes, which is an expanded.

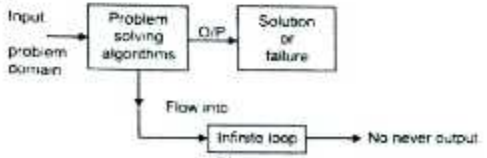
(2) It is collection of nodes that is just generated but which is in unexpanded state.

Leaf Node :

(1) The each and every element of the fringe is a leaf node.

(2) The leaf node is not having any successor in the tree.

Measurement of the Performance of the Problem Solving :



Figure

The performance of the problem solving algorithm is basically measured by following four factors.

(1) **Completeness of Solution :** With this factor completeness of solution is checked.

(2) **Optimality of Solution :** This factor checks whether optimal solution of the problem can be obtained or not.

(3) **Time Complexity :** This factor evaluates how much time is required to get a solution?

(4) **Space Complexity :** The evaluation of memory space required for any problem (to search) is performed?

(a) The difficult problems require more time and space complexity than simpler one.

(b) The size of state space graph is an important aspect in theory of computation as the graph is an explicit data structure.

(c) The static space graph is an input to each search program.

Branching Factor, Search Cost and Total Cost :

Figure

Graph is an implicit data structure which is represented with initial state and successor function.

(1) **Branching Factor :** The average number of children of a tree is counted as branching factor.

(2) **Complexity of Graph :** The complexity of graph is evaluated by three quantities.

(a) Branching factor or maximum number of successor (b).

(b) The depth of the shallowest goal node (d).

(c) The longest path length in the state space (m)

Search Cost : The node expansion is major activity in searching. The maximum number of nodes expanded and generated within specific time domain will generate search cost. Even the memory factor is also counted in search cost.

Total Cost : The total cost can be calculated by considering two factors,

(1) Time complexity of search algorithm.

(2) Space complexity of problem domain/solution.

The search cost and the path cost altogether form a total cost.

For Example : For the problem to find a route from Pune to Banglore, the such cost is the amount of time taken by the search and the solution cost is the complete length of the path in kilometers.

Total cost = Search cost in milliseconds + Solution cost in kilometers

Here kilometers are converted into milliseconds by using an estimate of the vehicle's average speed.

Uniformed Search Strategies or Blind Search Strategies :

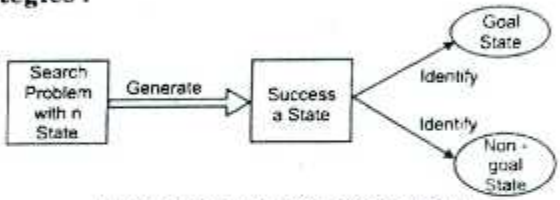


Figure : Typical Uniformed Search

Blind Search State :

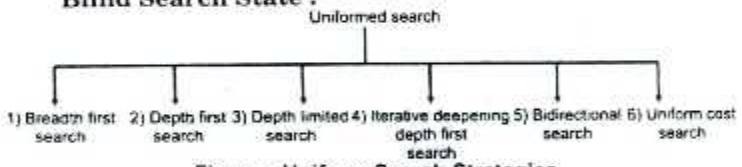


Figure : Uniform Search Strategies

Q.17. Explain the effect of over-estimation and under – estimation on A* algorithm. (2014-15)

Ans. A* maintains a priority queue of options that it's considering, ordered by how good they might be. It keeps searching until it finds a route to the goal that's so good that none of the other options could possibly make it better. How good an alternative might be is based on the heuristic and on actual costs found in the search so far.

If the heuristic underestimates, the other options will look better than they really are. A* thinks those other options might improve the route, so it checks them out. If the heuristic only underestimates by a little bit, maybe some of those routes will turn out to be useful. On the other hand, if the heuristic overestimates, A* can think that the alternatives to the route already has are all

terrible, so it won't bother to look at them. But th heuristic overestimates so they might be much better tha they seem.

For example, suppose you're trying to drive fro Chicago to New York and your heuristic is what you friends think about geography. If your first friend says "Hey, Boston is close to New York" (underestimating), the you'll waste time looking at routes via Boston. Before long you'll realize that any sensible route from Chicago t Boston already gets fairly close to New York befor reaching Boston and that actually going via Boston jus adds more miles. So you'll stop considering routes vi Boston and you'll move on t find the optimal route. Your underestimating friend cost you a bit of planning time but in the end, you found the right route.

Suppose that another friend says, "Indiana is a million miles from New York!" Nowhere else on earth i more than 13,000 miles from New York so, if you take your friend's advice literally, you won't even considering any route through Indiana. This makes you drive for nearly twice as long and cover 50% more distance.

Q.18. Explain the Local search algorithms with applications and describe its Terminology.

Ans. Local Search :

(1) In some problem path to the goal is not much important. There is a class of algorithm which neglect path at all.

(2) Local algorithm operates on above theory. Local algorithm considers a single "current state" instead of multiple paths and the move is applied to the neighbours of that state.

(3) The path followed by search is not remembered or not recorded in the memory.

(4) Hence, the search of this type is not systematic. But it has two key benefits

(a) As the explored path is not recorded it requires very less memory.

(b) This algorithm can be used to find solution of classic problem for which systematic search algorithm are not useful. Thus local search gives reasonable solution.

Application of Local Search : The local search algorithms are applicable for solving pure optimization problems. In the optimization problem it gives best goal state according to objective function. In the variety of problem domain, many optimization problem cannot fit the standard search model.

(1) An objective function of the problem can be derived from environment (or nature).

(2) For an optimization problem there is no specific "goal test" and no "path cost." The reproductive fitness can be achieved by Darwinian evolution.

Terminology of Local Search :

(1) **State Space Landscape :** A landscape is formed by location and elevation. The location means the state and the elevation means the value of heuristic cost function or objective function. The state space landscape is shown in diagram.

(2) **Global Minimum :** The lowest valley is an elevation which belong to cost is called as global minimum.

(3) **Global Maximum :** The highest peak is an elevation which belongs to an objective function called as a global maximum.

The conversion of global maximum to global minimum or vice versa is possible just by inserting a minus sign.

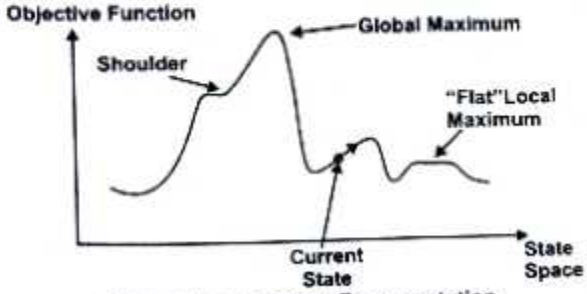


Figure : Local Search Representation

The above landscape is explored by local search algorithm. The goal is achieved by a complete local search algorithm. A global minimum/maximum can be searched by an optimal algorithm.

Q.19. You have three jugs measuring 12 gallons, 8 gallo and 3 gallons, and a water tap. You can fill the ju up or empty them out from one to another or o the ground. Your objective is to measure out exac one gallon. Give the complete state space and set all applicable/feasible rules.

Ans. The state space for this problem can be described as a of ordered pains of integer (x, y, z) such that $x = 0, 1, 2, \ldots, 12$, $y = 0, 1, 2, \ldots, 8$ and $z = 0, 1, 2$ and 3. Where represents water in 12 gallon jug, y represents water i gallon jug and z represents water in 3 gallon jug. The st state is (0, 0, 0). The goal state is $(1, m, n)$ for any value m and n (since the problem doesn't specify how ma gallons need to be in 8 and 3 gallon jug respectively).

On Solution to Water Jug Problem :

Gallons in 12 Gallons	Gallons in 8 Gallons jug	Gallons in 3 Gallons	Rule Applied
0	0	0	→ Initial State
0	0	3	→ Fill 3 gallons jug
3	0	3	→ Pour/empty 3 gallons jug into 12 gallons jug and fill 3 gallons
6	0	0	→ Empty 3 gallons jug into 12 gallons jug
6	0	3	→ Fill 3 gallons jug
9	0	0	→ Empty 3 gallons jug into 12 gallons jug
Goal State ↓	8 ↓	0	→ Fill 8 gallons jug into 8 gallons jug
(1)	0 Empty it	0	→ Empty 8 gallons jug

Production Rules for Water Jug Problem :

(1)	(x, y, z)	→	(12, y, z)	Fill 12 gallon jug
	If x < 12			
(2)	(x, y, z)	→	(x, 8, z)	Fill 8 gallon jug
	If y < 8			
(3)	(x, y, z)	→	(x, y, 3)	Fill 3 gallon jug
	If z < 3			
(4)	(x, y, z)	→	(x–d, y, z)	Pour some water out of 12 gallon jug
	If x > 0			
(5)	(x, y, z)	→	(x, (y–d), z)	Pour some water out of 8 gallon jug
	If y > 0			

(6)	(x, y, z)	→	(x, y, (z–d))	Pour some water out of 3 gallon jug
	If z > 0			
(7)	(x, y, 3)	→	(3, y, 0)	Empty/Pour water from 3 gallon jug into 12 gallon jug
(8)	(x–8, y, z)	→	(x, 8, z)	Pour water into 8 gallon jug till its ful
(9)	(x, y, z)	→	(x, 0, z)	Empty 8 gallon jug

The rules / production rules are applied as mentioned in the solution given. Fully 12 gallon jug has one gallon water and 8 gallon jug is empted on the ground.

Q.20. Discuss the problem of "Water Jug Problem" with Heuristic Search Techniques.

Ans. Heuristic Search : A heuristic is a method that :

(1) Might not always find the best solution

(2) But is guaranteed to find a good solution in reasonable time.

(3) By sacrificing completeness it increases efficiency.

(4) Useful in solving tough problems which :

(a) Could not be solved any other way.

(b) Solutions take an infinite time or very long time to compute.

The classic example of heuristic search methods is the travelling salesman problem.

Heuristic Search Methods Generate and Test Algorithm :

(1) Generate a possible solution which can either be a point in the problem space or a path from the initial state.

(2) Test to see if this possible solution is a real solution by comparing the state reached with the set of goal states.

(3) If it is a real solution, return. Otherwise repeat from 1.

Water Jug Problem : In water jug problem there are two jugs given a 4 gallon one and a 3-gallon one. Neither have any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can we fill exactly 2 gallons water in 4 gallon jug?

The state space for this problem can be described as set of ordered pairs of integers (x, y) such that x = 0,1,2,3 4 and y = 0, 1, 2 or 3. The state is (0, 0). The goal stat (2, n).

One such heuristic solution is given below :

Gallons in 4-gallon Jug	Gallons in 2-gallon Jug
0	0
0	3
3	0
3	3
4	2
0	2
2	0

2 Gallons in 4-Gallon Jug

UNIT – THREE
KNOWLEDGE REPRESENTATION & REASONING

Q.1. Compare propositional logic and predicate logic. (2018-19)

Ans. Difference Between Propositional Logic and Predicate Logic : Propositional logic is the study of propositions, where a proposition is a statement that is either true or false. Predicate calculus includes predicates, variables and quantifiers, and a predicate is a characteristic or property that the subject of a statement can have.

In AI or artificial intelligence, propositional logic deals with the determination of the truth of a sentence. The allowable sentence is known as the syntax of proposition. A sentence or syntax holds different propositional symbols. Each symbol holds a proposition which can be either true or false. For instance, a proposition like 'Socrates is a man' can be represented as 'S' in propositional logic. In predicate logic the subject and predicate are symbolized separately. Often logicians symbolize subjects or objects with lowercase letters and symbolize predicates with uppercase letters.

	Propositional Logic	Predicate Logic
Atomic Symbols	Concrete objects, AND, OR, NOT, IF, THEN	Propositional logic + variables + quantifiers : for all, exists.
Examples of formalizable statements	My son is at home and my dad is not. It either rains or it does not. If the president sleeps then a war cannot start.	All husbands cheat. If all colleges are bad, CMU is bad. At least one chinchilla is smarter than at least one human.

Propositional logic is associated with finite models while predicate logic is related to both finite and infinite structures.

Q.2. Explain the concept of forward and backward state space search in detail. (2018-19)

Ans.(1) Forward State-Space Search : Planning with forward state-space search is similar to the problem-

solving approach. It is sometimes called progressi planning, because it moves in the forward direction

We start with the problem's initial sta con ring sequences of actions until we reach a go stat

The Formulation of Planning Problem a State-Space Search Problems is as Follows :

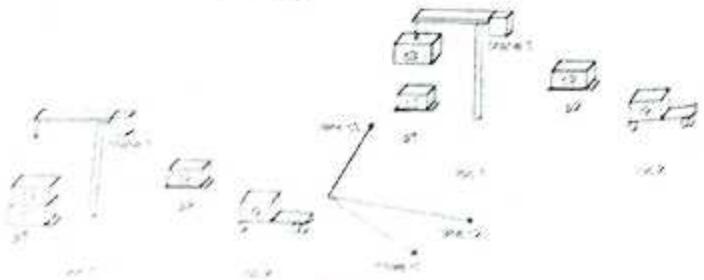
(a) The initial state of the search is the initial sta from the planning problem. In general each sta will be set of positive ground literals; literals no appearing are false.

(b) The actions which are applicable to a state ar all those whose preconditions are satisfied. Th successor state resulting from an action generated by adding the positive effect litera and deleting the negative effect literals.

(c) The goal test checks whether the state satisfi the goal of the planning problem.

(d) The step cost of each action is typically Although it would be easy to allow different cos for different actions, this was seldom done b STRIPS planners.

```
Forward-search (O, s, g)
    s ← %
    π ← the empty plan
    loop
        if s satisfies g then return
        E ← {a|a is a ground instance an operator in O, an
        precond(a) is true in s}
        if E = Ø then return failure
        nondeterministically choose an action a ∈ E
        s ← γ (s, a)
        π ← π . a
```



Figure

(2) Backward State-Space Search : Backward search can be difficult to implement when the goal states are described by a set of constraints which are not listed explicitly. In particular, it is not always obvious how to generate a description of the possible predecessors of the set of goal states. The STRIPS representation makes this quite easy because sets of states can be described by the literals which must be true in those states.

The main advantage of backward search is that it allows us to consider only relevant actions. An action is relevant to a conjunctive goal if it achieves one of the conjuncts of the goal. For example, the goal in our 10-airport air cargo problem is to have 20 pieces of cargo at airport B, or more precisely,

$At(C1, B) \wedge At(C2, B) \ldots\ldots\ldots\ldots\ldots\ldots At(C20, B)$

Now consider the conjunct At $(C1, B)$. Working backwards, we can seek those actions which have this as an effect,

There is only one :

Unload $(C1p, B)$,

where plane p is unspecified.

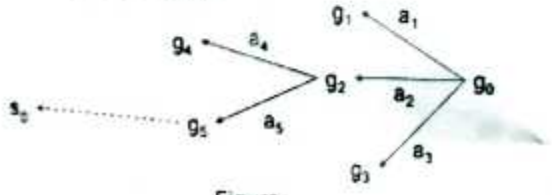
We may note that there are many irrelevant actions which can also lead to a goal state. For example, we can fly an empty plane from Mumbai to Chennai; this action reaches a goal state from a predecessor state in which the plane is at Mumbai and all the goal conjuncts are satisfied. A backward search which allows irrelevant actions will still be complete, but it will be much less efficient. If a solution exists, it should be found by a backward search which allows only relevant action.

This restriction to relevant actions only means that backward search often has a much lower branching factor than forward search. For example, our air cargo problem has about 1000 actions leading forward from the initial state, but only 20 actions working backward from the goal. Hence backward search is more efficient than forward searching.

It is a search in the reverse direction start with the goal state, expand the graph by computing parents. The parents are computed by regressing actions given a ground goal description g and a ground action a, the regression from g over a is

$g' : g' = (g - ADD(a)) \cup PRECOND(a)$. The regress represents the effects that don't have to be true in previous step because they were added have to true in the previous step because they are preconditions of the action

```
Backward-search (O, s0, g)
    π ← the empty plan
    loop
        if s0 satisfies g then return π
        A ← {a|a is a ground instance an operator in O
        γ⁻¹(g, a) is defined}
        If A = ∅ then return failure
        nondeterministically choose an action a ∈ A
        π ← a . π
        g ← γ⁻¹ (g, a)
```



Figure

Q.3. Justify the usage of universal and existentia quantifier with an example. (2018-19)

Ans. Universal Quantifier : A Universal Quantifier is a logical statement that applies to all elements of a set.

An Existential Quantifier is a logical statement tha applies to at least one element of a set.

The universal quantifier ∀ is used to expres universal claims, those we express in English using quantified phrases like everything, each thing, all things and anything.

Existential Quantifier : The existential quantifier ∃ i used to express existential claims, those we express i English using such phrases as something, at least on thing, a, and an.

In the "for all x" case I'm saying tha "(something about x)" is true for any x you could pick that's universal quantification. For example "for all x, x i an even number" is a false statement.

In the "there exists an x such that" case I'm saying that there is a possible choice for x so tha "(something about x)" is true (I'm not saying what that

choice is, just that there is one). This is called existential quantification. As an example "there exists an x such that x is an even number" is a true statement.

Q.4. Which algorithm is more similar to backward chaining algorithm? Write its algorithm. (2018-19)

Ans. Depth first search or DFS is similar to backward chaining algorithm. The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking which resembles backward chaining algorithm. Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

Algorithm : Depth First Search :

(1) If the initial state is a goal state, quit and return success.

(2) Otherwise, loop until success or failure is signaled.

(a) Generate a state, say E, and let it be the successor of the initial state. If there is no successor, signal failure.

(b) Call Depth-First Search with E as the initial state.

(c) If success is returned, signal success. Otherwise continue in this loop.

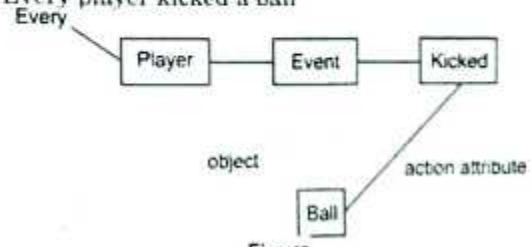
Q.5. Represent the following in partitioned semantic networks : (2018-19)

(1) Every player kicked a ball.

(2) All players like the referee.

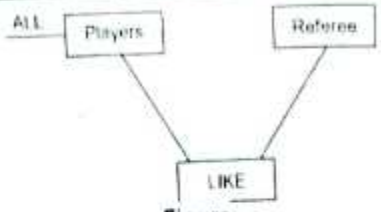
(3) Andrew believes that there is a fish with lungs.

Ans.(1) Every player kicked a ball



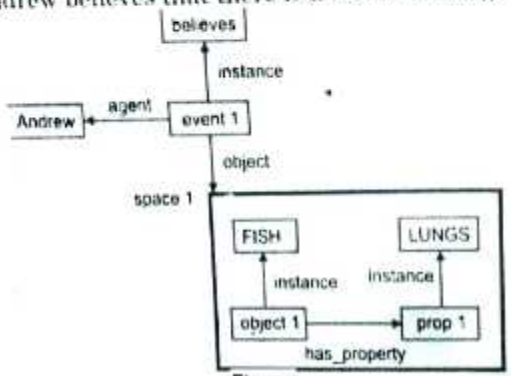
Figure

(2) All players like the referee



Figure

(3) Andrew believes that there is a fish with lungs



Figure

Q.6. Discuss the different design issues to be solved t use hidden markov model for real world application (2018-1

OR **Describe Hidden Markov model with suitabl example. Also discuss its role in probabilisti reasoning. (2017-1**

OR **Write short notes on :**

(1) Probabilistic Reasoning.

(2) Hidden Markov Models.

Ans. (1) Probabilistic Reasoning : Probabilities, Randor Variables, Probability Distribution, Condition Probability, Joint Distributions, Bayes Theorem

Probability of an Event : Consider an experime that may have different outcomes and what is th probability of a particular set of outcomes. Let samp space S is the set of all possible outcomes. Let Eve A be any subset of S

Definition : Probability(A) = (Number of outcomes A)/(Total number of outcomes)

$$P(A) = |A|/|S|$$

i.e. the probability of A is equal to the number outcomes of interest divided by the number of possible outcomes.

$P(A)$ is called prior (or unconditional) probability of A

$P(\neg A)$ is the probability event A not to take place

Example : The probability to pick a spade card out of a deck of 52 cards is 13/52 = 1/4. The probability to pick an Ace out of a deck of 52 cards is 4/52 = 1/13.

(2) Hidden Markov Models : A hidden Markov model (HMM) is an augmentation of the Markov chain to include observations. Just like the state transition of the Markov chain, an HMM also includes observations of the state. These observations can be partial in that different states can map to the same observation and noisy in that the same state can stochastically map to different observations at different times.

The assumptions behind an HMM are that the state at time $t+1$ only depends on the state at time t, as in the Markov chain. The observation at time t only depends on the state at time t. The observations are modeled using the variable O_t for each time t whose domain is the set of possible observations. The belief network representation of an HMM is depicted in Figure. Although the belief network is shown for four stages, it can proceed indefinitely.

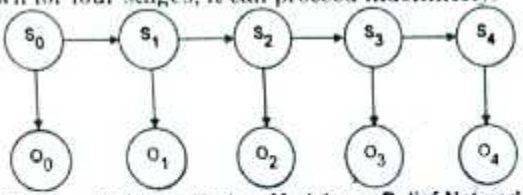


Figure : A Hidden Markov Model as a Belief Network

A stationary HMM includes the following probability distributions :

(1) $P(S_0)$ specifies initial conditions.

(2) $P(S_{t+1} | S_t)$ specifies the dynamics.

(3) $P(O_t | S_t)$ specifies the sensor model.

There are a number of tasks that are common for HMMs. The problem of filtering or belief-state monitoring is to determine the current state based on the current and previous observations, namely to determine

$$P(S_t | O_0, \ldots, O_t)$$

Note that all state and observation variables after S_t are irrelevant because they are not observed and can be ignored when this conditional distribution is computed.

The problem of smoothing is to determine a state based on past and future observations. Suppose an agent has observed up to time k and wants to determine the state at time i for $i<k$; the smoothing problem is to determine

$$P(S_i \mid O_0,\ldots,O_k)$$

All of the variables S_i and V_i for $i>k$ can be ignored.

Q.7. Pick a problem area and represent the knowledge in frame based system. (2018-19)

Ans. Using Semantic Networks for representing knowledge has particular advantages :

(1) They allow us to structure the knowledge to reflect the structure of that part of the world which is being represented.

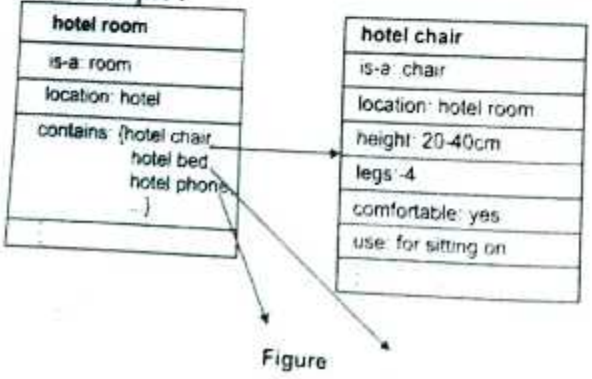
(2) The semantics, i.e. real world meanings, are clearly identifiable.

(3) There are very powerful representational possibilities as a result of "is a" and "is a part of" inheritance hierarchies.

(4) They can accommodate a hierarchy of default values (for example, we can assume the height of an adult male to be 178cm, but if we know he is a baseball player we should take it to be 195cm).

(5) They can be used to represent events and natural language sentences. Clearly, the notion of a semantic network is extremely general. However, that can be a problem, unless we are clear about the syntax and semantics in each case. A frame consists of a selection of slots which can be filled by values, or procedures for calculating values, or pointers to other frames.

For Example :



Figure

A complete frame based representation will consist of a whole hierarchy or network of frames connected together by appropriate links/pointers.

Frames as a Knowledge Representation : The simplest type of frame is just a data structure with similar properties and possibilities for knowledge representation as a semantic network, with the same ideas of inheritance and default values. Frames become much more powerful when their slots can also contain instructions/procedures for computing things from information in other slots or in other frames. The original idea of frames was due to Minsky (1975) who defined them as "data-structures for representing stereotyped situations", such as going into a hotel room. This type of frames are now generally referred to as Scripts. Attached to each frame will then be several kinds of information. Some information can be about how to use the frame. Some can be about what one can expect to happen next, or what one should do next. Some can be about what to do if our expectations are not confirmed. Then when one encounters a new situation, one can select from memory an appropriate frame and this can be adapted to fit reality by changing particular details as necessary.

It is easy to construct frames for each node of a semantic net by reading off the links, e.g.

Figure

Baseball player	Fielder	Pee-Wee-Reese
is a Adult Male	is a Baseball player	instance Baseball player
batting average 252 bats equal to handed	Batting average 262	team Brooklyn Dodgers
team		

Q.8. Give the completeness proof of resolution. (2018-19)

***OR* Describe Resolution in brief.**

Ans. Resolution : Resolution is a rule of inference leading to a refutation theorem-proving technique for sentences in propositional logic and first-order logic. In other words, iteratively applying the resolution rule in a suitable way allows for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable; this method may prove the satisfiability of a first-order satisfiable formula, but not always, as it is the case for all methods for first-order logic. Resolution was introduced by John Alan Robinson in 1965.

Resolution in Propositional Logic : The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals. A literal is a propositional variable or the negation of a propositional variable. **Two literals** are said to be complements if one is the negation of the other. In the following, a_i is taken to be the complement to b_j. The resulting clause contains all the literals that do not have complements. Formally :

$$\frac{a_1 \vee \ldots \vee a_i \vee \ldots \vee a_m,\ b_1 \vee \ldots \vee b_j \vee \ldots \vee b_n}{a_1 \vee \ldots \vee a_{i-1} \vee a_{i+1} \vee \ldots \vee a_m \vee b_1 \vee \ldots \vee b_{j-1} \vee b_{j+1} \vee \ldots \vee b_n}$$

Where all as and bs are literals, a_i is the complement to b_j, and the dividing line stands for entails. The clause produced by the resolution rule is called the resolvent of the two input clauses. When the two clauses contain more than one pair of complementary literals, the resolution rule can be applied (independently) for each such pair. However, only the pair of literals that are resolving upon can be removed : all other pair of literals remain in the resolving clause.

***A Resolution Technique* :** When coupled with a complete search algorithm, the resolution rule yields a sound and complete algorithm for deciding the satisfiability of a propositional formula, and, by extension, the validity of a sentence under a set of axioms. This resolution technique uses *proof by contradiction* and is based on the fact that any sentence in propositional logic can be transformed into

(1) All sentences in the knowledge base and the negation of the sentence to be proved (the conjecture) are conjunctively connected.

(2) The resulting sentence is transformed into a conjunctive normal form with the conjuncts viewed as elements in a set, S of clauses.

For Example : ($A_1 \vee A_2$) $\wedge$ ($B_1 \wedge B_2$) $\wedge$ B_3) $\vee$ (C_1)

Would give rise to a set

$S = \{ A_1 \vee A_2, B_1 \vee B_2 \vee B_3, C_1 \}$ <= "p=" style="margin : 0em; padding : 0em;">

(3) The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the sentence contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set S, it is added to S, and is considered for further resolution inferences.

(4) If after applying a resolution rule the empty clause is derived, the complete formula is unsatisfiable (or contradictory), and hence it can be concluded that the initial conjecture follows from the axioms.

(5) If, on the other hand, the empty clause cannot be derived, and the resolution rule cannot be applied to derive any more new clauses, the conjecture is not a theorem of the original knowledge base.

One instance of this algorithm is the original Davis–Putnam algorithm that was later refined into the DPLL algorithm that removed the need for explicit representation of the resolvents. This description of the resolution technique uses a set S as the underlying data-structure to represent resolution derivations. Lists, Trees and Directed Acyclic Graphs are other possible and common alternatives. Tree representations are more faithful to the fact that the resolution rule is binary.

Together with a sequent notation for clauses, a tree representation also makes it clear to see how the resolution rule is related to a special case of the cut-rule, restricted to atomic cut-formulas. However, tree representations are not as compact as set or list representations, because they explicitly show redundant subderivations of clauses that are used more than once in the derivation of the empty clause. Lists or sets can be as compact in the

number of clauses as list representations and they also store structural information regarding which clauses were resolved to derive each resolve.

Example : $\frac{a \vee b, \neg a \vee c}{b \vee c}$

In English : if a or b is true, and a is false or c is true, then either b or c is true. If a is true, then for the second premise to hold, c must be true. If a is false, then for the first premise to hold, b must be true. So regardless of a, if both premises hold, then b or c is true.

Q.9. Define forward chaining and backward chaining with example. (2017-18)

OR **Explain Meta knowledge. Under what conditions would it make sense to use both forward and backward chaining? Give an example where both are used.**

Ans. (1) Meta Knowledge : Meta-knowledge may be loosely defined as "knowledge about knowledge". Meta-knowledge includes information about the knowledge the system possesses, about the efficiency of certain methods used by the system, the probabilities of the success of past plans, etc. The meta-knowledge is generally used to guide future planning or execution phases of a system.

(2) Backward Chaining : Backward chaining (or backward reasoning) is an inference method used in automated theorem proves, proof assistants and other artificial intelligence applications. It is one of the two most commonly used methods of reasoning with inference rules and logical implications the other is forward chaining. Backward chaining is implemented in logic programming by SLD resolution. Both rules are based on the modus ponens inference rule.

Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal. If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals (in order for one's goal to be confirmed one must also provide data that confirms this new rule). For example, suppose that the goal is to conclude the color of my pet Fritz, given that he croaks and eats flies, and that the rule base contains the following four rules :

(a) If X croaks and eats flies then X is a frog.
(b) If X chirps and sings then X is a canary.
(c) If X is a frog then X is green.
(d) If X is a canary then X is yellow.

This rule base would be searched and the third and fourth rules would be selected, because their consequent (Then Fritz is green, Then Fritz is yellow) match the goal (to determine Fritz's color). It is not yet known that Fritz is a frog, so both the antecedents (If Fritz is a frog, If Fritz is a canary) are added to the goal list. The rule base is again searched and this time the first two rules are selected, because their consequents (Then X is a frog, Then X is a canary) match the new goals that were just added to the list. The antecedent (If Fritz croaks and eats flies) is known to be true and therefore it can be concluded that Fritz is a frog, and not a canary. The goal of determining Fritz's color is now achieved (Fritz is green if he is a frog, and yellow if he is a canary, but he is a frog since he croaks and eats flies; therefore, Fritz is green).

The goals always match the affirmed versions of the consequents of implications (and not the negated versions as in modus tolling) and even then, their antecedents are then considered as the new goals (and not the conclusions as in affirming the consequent) which ultimately must match known facts (usually defined as consequents whose antecedents are always true); thus, the inference rule which is used is modus ponens. Because the list of goals determines which rules are selected and used, this method is called goal-driven, in contrast to data-driven forward-chaining inference. The backward chaining approach is often employed by expert systems.

(3) Forward Chaining : Forward chaining is one of the two main methods of reasoning when using inference

rules (in artificial intelligence). It is referred philosophical circle as modus ponens. The oppose forward chaining is backward chaining. Forw chaining starts with the available data and u inference rules to extract more data (from an u user for example) until a goal is reached. An inferen engine using forward chaining searches the inferen rules until it finds one where the antecedent clause) is known to be true. When found it c conclude, or infer, the consequent (Then claus resulting in the addition of new information to i data. Inference engines will iterate through th process until a goal is reached. For example, suppos that the goal is to conclude the color of a pet name Fritz, given that he croaks and eats flies, and that th rule base contains the following four rules :

- **(a)** If X croaks and eats flies then X is a frog.
- **(b)** If X chirps and sings then X is a canary.
- **(c)** If X is a frog then X is green.
- **(d)** If X is a canary then X is yellow.

This rule base would be searched and the first rul would be selected, because its antecedent (If Fritz croak and eats flies) matches our data. Now the consequen (Then X is a frog) is added to the data. The rule base i again searched and this time the third rule is selected because its antecedent (If Fritz is a frog) matches our data that was just confirmed. Now the new consequent (Then Fritz is green) is added to our data. Nothing more can be inferred from this information, but we have now accomplished our goal of determining the color of Fritz. Because the data determines which rules are selected and used, this method is called data-driven, in contrast t goal-driven backward chaining inference. The forward chaining approach is often employed by expert systems such as CLIPS. One of the advantages of forward-chaining over backward-chaining is that the reception of new data can trigger new inferences, which makes the engine better suited to dynamic situations in which conditions are likely to change.

Choice between Forward and Backward Chaining : Forward chaining is often preferable in case where there are many rules with the same conclusions. A well-known category of such rule systems are taxonomic

hierarchies. Example : The taxonomy of the animal kingdom includes such rules

```
animal(X) : sponge(X).
animal(X) : arthropod(X).
animal(X) : vertebrate(X).
...
vertebrate(X) : fish(X).
vertebrate(X) : mammal(X).
...
mammal(X) : carnivore(X).
...
carnivore(X) : dog(X).
carnivore(X) : cat(X).
...
```

Now, suppose we have such a knowledge base of rules, we add the fact "dog(fido)" and we query whether "animal(fido)". In forward chaining, we will successively add "carnivore(fido)", "mammal (fido)", "vertebrate(fido)", and "animal(fido)". The query will then succeed immediately. The total work is proportional to the height of the hierarchy. By contrast, if you use backward chaining, the query "~ animal(fido)" will unify with the first rule above, and generate the subquery "~sponge(fido)", which will initiate a search for Fido through all the subdivisions of sponges, and so on. Ultimately, it searches the entire taxonomy of animals looking for Fido.

Condition Under which Forward Chaining and Backward Chaining Both : In some cases, it is desirable to combine forward and backward chaining. For example, suppose we augment the above animal with features of these various categories :

```
breathes(X) : animal(X).
...
backbone(X) : vertebrate(X).
has(X,brain) : vertebrate(X).
...
furry(X) : mammal(X).
warm_blooded(X) : mammal(X).
...
```

If all these rules are implemented as forward chaining, then as soon as we state that Fido is a dog, we have to add all his known properties to Gamma: that he breathes, is warm-blooded, has a liver and kidney, and so

on. The solution is to mark these property rules as backward chaining and mark the hierarchy rules as forward chaining. You then implement the knowledge base with both the forward chaining algorithm, restricted to rules marked as forward chaining, and backward chaining rules, restricted to rules marked as backward chaining. However, it is hard to guarantee that such a mixed inference system will be complete.

Q.10. For each of the following agents, develop a PEAS description of the task environment- (2017-18)

(1) Mathematician's theorem proving assistant

(2) Satellite image analysis system

(3) Internet book shopping agent

(4) Medical diagnosis system

Ans. Agents and Development of a PEAS Description of the Task Environment :

Agent Type	Performance Measure	Environment	Actuators	Sensors
Mathematician's Theorem-Proving Assistant	Theorems proved good math knowledge, new theorems discovered, time requirement, and degree of correction	CPU, the theorem to prove, existing axioms, internet, library.	Display to user, accept the right theorem, reject the wrong theorem, infer based on axioms and facts.	User input (keyboard, file system) input device that reads the theorem to prove.
Satellite Image Analysis System	Correct image categorization	Downlink from orbiting satellite	Display categorization of scene	Color pixel arrays
Internet Book-Shopping Agent	Obtain requested /interesting books, minimize expenditure	Internet	Follow link, enter/submit data in fields, display to user	Web pages user requests
Medical Diagnosis System	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff.	Screen display (questions, tests, diagnoses, treatments, referrals)	Keyboard (entry of symptoms, findings, patient's answers)

Q.11. Transform the following formula to Prenex Normal form (2017-18)

$$\forall x : \forall y : (\exists z : P(x,z) \wedge P(y,z)) \rightarrow \exists u : Q(x,y,u)$$

Ans. $\forall x \forall y \forall z \exists u(\neg P(x,z) \vee \neg P(x,z) \vee Q(x,y,u))$

Q.12. Discuss Maximum-likelihood parameter learning for complete data with discrete models. (2017-18)

OR **Explain Maximum likelihood hypothesis and Maximum a posteriori. (2015-16)**

Ans. Maximum A Posteriori : One way to trade off model complexity and fit to the data is to choose the model that is most likely, given the data. That is, choose the model that maximizes the probability of the model given the data, $P(\text{model}|\text{data})$. The model that maximizes $P(\text{model}|\text{data})$ is called the maximum a posteriori probability model, or the MAP model.

The probability of a model (or a hypothesis) given some data is obtained by using Bayes' rule :

$P(\text{model}|\text{data}) = (P(\text{data}|\text{model}) \times P(\text{model}))/(P(\text{data}))$.

The likelihood, $P(\text{data}|\text{model})$, is the probability that this model would have produced this data set. It is high when the model is a good fit to the data, and it is low when the model would have predicted different data. The prior $P(\text{model})$ encodes the learning bias and specifies which models are a priori more likely. The prior probability of the model, $P(\text{model})$, is required to bias the learning toward simpler models. Typically simpler models have a higher prior probability. The denominator $P(\text{data})$ is a normalizing constant to make sure that the probabilities sum to 1.

Because the denominator of above equation is independent of the model, it can be ignored when choosing the most likely model. Thus, the MAP model is the model that maximizes

$P(\text{data}|\text{model}) \times P(\text{model})$.

One alternative is to choose the maximum likelihood model - the model that maximizes $P(\text{data}|\text{model})$. The problem with choosing the most likely model is that, if the space of models is rich enough, a model exists that specifies that this particular data set will be produced, which has $P(\text{data}|\text{model}) = 1$. Such a model may be a priori very unlikely. However, we do not want to exclude it, because it may be the true model. Choosing the maximum-likelihood

...equivalent to choosing the maximum a pos...
...uniform prior over hypotheses.

Maximum Likelihood (ML) Learning : Choose ... $P(h_i)$. **For Example :** Simply get the ... identical to MAP for uniform prior (which ... if all hypotheses are of the same complexity. ML is the "standard" (non-Bayesian) statistical learning ...

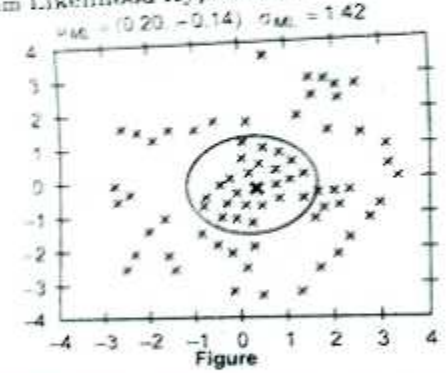
(1) Re-introduce ... distrust the subjective nature ... hypothesis priors.

(2) Hypotheses are of the same complexity.

(3) Hypotheses priors is of less important when data ...

(4) Hypothesis space of the hypotheses ...

Maximum Likelihood Hypothesis :



Figure

Q.13. Discuss algorithm of conversion to clause form. Convert the following to clause form using algorithm (2017)

$$\forall x[Brick(x) \rightarrow (\exists y[On(x,y) \wedge \neg Pyramid(y)] \wedge \neg\exists y[On(x,y) \wedge On(y,x)] \wedge \forall y[\neg Brick(y) \rightarrow \neg Equal(x,y)])]$$

Ans. Algorithm of Conversion to Clause Form :

(1) Eliminate $\rightarrow$, using the fact that $a \rightarrow b$ is equivalent to $\neg a \vee b$.

$\forall x$ [Roman(x) ∨ knows(x, Marcus)] ∧ [Hate(x, Caesar) ∨ (∀y ∃z hate(y, z) ∨ thinkcrazy(x, y))]

(2) Reduce the scope of ...

...using the fact that $\neg(\neg p) = p$.

...De-Morgan's Laws

$\neg(a \wedge b) \equiv \neg a \vee \neg b$ and $\neg(a \vee b) \equiv \neg a \wedge \neg b$

The correspondence between quantifiers

$\neg \forall x\, P(x) \equiv \exists x \neg P(x)$

$\neg \exists x\, P(x) \equiv \forall x \neg P(x)$

Applying these to the wff from step 1 yields

$\forall x$ [¬Roman(x) ∨ ¬knows(x, Marcus)] ∧ [∀x ∀y ∀z ¬hate(y, z) ∨ ¬hate(x, Caesar) ∨ (∀y ∃z thinkcrazy(x, y))]

(3) Standardise variables so that each quantifier binds a unique variable.

$\forall x\, P(x) \vee \forall x\, Q(x)$ is converted to $\forall x\, P(x) \vee \forall y\, Q(y)$

Done in preparation for step 4

(4) Move all quantifiers to the left of the formula without changing the relative order. Done on the wff gives :

$\forall x \forall y \forall z$ [¬Roman(x) ∨ ¬knows(x, Marcus)] ∧ [hate(x, Caesar) ∨ (¬hate(y, z) ∨ thinkcrazy(x, y))]

This is called prenex normal form. It consists of a prefix of quantifiers followed by a matrix which is quantifier free.

Numerical Solution :

$\forall X$ [brick(X) ⇒ (∃Y [on(X, Y) ∧ ¬pyramid(Y)] ∧ ¬∃Y [on(X, Y) ∧ on(Y, X)] ∧ ∀Y [¬brick(Y) ⇒ ¬equal(X, Y)])]

(1) Eliminate ⇒ using $A \Rightarrow B = \neg A \vee B$

$\forall X$ [¬brick(X) ∨ (∃Y [on(X, Y) ∧ ¬pyramid(Y)] ∧ ¬∃Y [on(X, Y) ∧ on(Y, X)] ∧ ∀Y [¬¬brick(Y) ∨ ¬equal(X, Y)])]

(2) Put into negation normal form : negation only occurs immediately before propositions

$\forall X$ [¬brick(X) ∨ (∃Y [on(X, Y) ∧ ¬pyramid(Y)] ∧ ∀Y [¬on(X, Y) ∨ ¬on(Y, X)] ∧ ∀Y [brick(Y) ∨ ¬equal(X, Y)])]

(3) Replace ∃ using Skolem functors (abstract names for objects, functor has to be new) :

$\forall X$ [¬brick(X) ∨ (on(X, sup(X)) ∧ ¬pyramid(sup(X))] ∧ ∀Y [¬on(X, Y) ∨ ¬on(Y, X)] ∧ ∀Y [brick(Y) ∨ ¬equal(X, Y)])]

(4) Standardize all variables apart such that e quantifier has its own unique variable

$\forall X[\neg brick(X) \vee ([on(X, sup(X)) \wedge \neg pyramid(sup(X))$

$\forall Y[\neg on(X, Y) \vee \neg on(Y, X)$

$\forall Z[brick(Z) \vee \neg equal(X, Z)$

(5) Move $\forall$ to the front

$\forall X \forall Y \forall Z[\neg brick(X) \vee ([on(X, sup(X))$

$\wedge \neg pyramid(sup(X))$

$[\neg on(X, Y) \vee \neg on(Y, X)$

$[brick(Z) \vee \neg equal(X, Z)$

(6) Convert to conjunctive normal form usi $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

$\forall X \forall Y \forall Z[(\neg brick(X) \vee ([on(X, sup(X)) \wedge$

$\neg pyramid(sup(X))$

$(\neg brick(X) \vee [\neg on(X, Y) \vee \neg on(Y, X)) \wedge$

$(\neg brick(X) \vee [brick(Z) \vee \neg equal(X, Z)$

$\forall X \forall Y \forall Z[((\neg brick(X) \vee on(X, sup(X))) \wedge$

$(\neg brick(X) \vee \neg pyramid(sup(X)))$

$(\neg brick(X) \vee [\neg on(X, Y) \vee \neg on(Y, X)]$

$(\neg brick(X) \vee [brick(Z) \vee \neg equal(X, Z)$

$\forall X \forall Y \forall Z[(\neg brick(X) \vee on(X, sup(X))] \wedge$

$[\neg brick(X) \vee \neg pyramid(sup(X))]$

$[\neg brick(X) \vee \neg on(X, Y) \vee \neg on(Y, X)]$

$[\neg brick(X) \vee brick(Z) \vee \neg equal(X, Z)]$

(7) Split the conjuncts in clauses (a disjunction o literals)

$\forall X \quad \neg brick(X) \vee on(X, sup(X))$

$\forall X \quad \neg brick(X) \vee \neg pyramid(sup(X))$

$\forall X \forall Y \quad \neg brick(X) \vee \neg on(X, Y) \vee \neg on(Y, X)$

$\forall X \forall Z \quad \neg brick(X) \vee brick(Z) \vee \neg equal(X, Z)$

(8) Convert to clausal syntax (negative literals to bod positive ones to head)

on(X, sup(X)) :- brick(X),

:- brick(X), pyramid(sup(X)),

:- brick(X), on(X, Y), on(Y, X),

brick(X) :- brick(Z), equal(X, Z).

Q.14. Represent the following sentence in the Predicate form "All the children like sweets". **(2015-16)**

Ans. $\forall x$: like (Children, x) $\rightarrow$ Sweets (x).

Q.15. Write four properties a good system should possess for the knowledge representation in a particular domain. **(2014-15, 2015-16)**

Ans. Four Properties a Good System should Possess for the Knowledge Representation in a Particular Domain :

(1) **Representational Adequacy :** The ability to represent all the different kinds of knowledge that might be needed in that domain.

(2) **Inferential Adequacy :** The ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.

(3) **Inferential Efficiency :** The ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.

(4) **Acquisitional Efficiency :** The ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a knowledge engineer would be acceptable.

Q.16. What is Bayesian reasoning? What does a Bayesian network represent? Explain. **(2014-15, 2015-16)**

OR Write short note on Bayesian Networks.

Ans. Bayesian Probability : Bayesian probability is one interpretation of the concept of probability. In contrast to interpreting probability as frequency or propensity of some phenomenon, Bayesian probability is a quantity that we assign to represent a state of knowledge, or a state of belief. In the Bayesian view, a probability is assigned to a hypothesis, whereas under frequentist inference, a hypothesis is typically tested without being assigned a probability. The Bayesian interpretation of probability can be seen as an extension of propositional logic that enables reasoning with hypotheses, i.e., the propositions whose truth or falsity is uncertain.

Bayesian probability belongs to the category of evidential probabilities; to evaluate the probability of a hypothesis the Bayesian probabilist specifies some prior probability which is then updated in the light of new relevant data (evidence). The Bayesian interpretation provides a standard set of procedures and formulae to perform this calculation. The term "Bayesian" derives from the 18th century mathematician and theologian Thomas Bayes, who provided the first mathematical treatment of a non-trivial problem of Bayesian inference. Mathematician Pierre-Simon Laplace pioneered and popularized what is now called Bayesian probability. Broadly speaking, there are two views on Bayesian probability that interpret the probability concept in different ways.

According to the objectivist view, the rules of Bayesian statistics can be justified by requirements of rationality and consistency and interpreted as an extension of logic. According to the subjectivist view, probability quantifies a "personal belief".

Bayesian Network : A Bayesian network is a directed acyclic graph with :

(1) Random variables as nodes.

(2) Links that specify "directly influences" relationships.

(3) Probability distributions $P(x_i \mid \text{parents}(x_i))$ for each node x_i.

Graph structure asserts conditional independencies :

P (Mary Calls | John Calls, Alarm, Earthquake, Burglary) = P (Mary Calls | Alarm)

Bayesian Networks as Joint Probabilities :

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i, \ldots, x_{i-1})$$

$$= \prod_{i} P(x_i \mid \text{parents}(x_i))$$

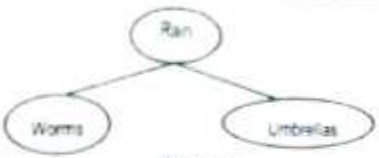
For parents $(x_i) \subseteq \{x_1, \ldots, x_{i-1}\}$

Burglary Example :

$$P(b, \neg e, a, \neg m, j) = P(b) P(\neg e) P(a \mid b, \neg e) P(\neg m \mid a) P(j \mid a)$$

$$= 0.001 \times 0.998 \times 0.94 \times 0.3 \times 0.9$$
$$= 0.0002$$

A Simple Bayesian Network Example :



Figure

P (Rain, Worms, Umbrellas) = P (Worms | Rain) P (Umbrellas | Rain) P (Rain). With conditional independence, need only right-hand side to represent joint distribution.

Q.17. Convert the following sentence into predicate logic and then prove "Marcus is Dead" using resolution :

(1) Marcus was a man.

(2) Marcus was a Pompeian.

(3) Marcus was born in 40 AD.

(4) All men are mortal.

(5) All Pompeian's dead when the volcano erupted in 1979.

(6) No mortal lives more than 150 years.

(7) It is now 1991.

(8) Alive means not dead. **(2015-16)**

Ans. → Man (Marcus)

→ Pompeian (Marcus)

→ Born (Marcus, 40)

→ $\forall X$: Men (X) → Mortal (X)

→ Erupted (volcano, 79) ∧ $\forall X$: [Pompeian (X) → Died (X, 79)]

→ $\forall X$: $\forall t_1$: $\forall t_2$: Mortal (X) ∧ Born (X, t_1) ∧ $gt(t_2 - t_1, 150)$ → dead (X, t_2)

→ Now = 1991

→ $\forall X$: $\forall t$: [alive (X, t) → ¬ dead (X, t)] ∧ [¬ dead (X, t) → alive (X, t)]

Now that fact's are represented, they can now be used for reasoning

Now we have to prove that Marcus is dead this can be represented in predicate logic as dead (Marcus, now) or ¬ alive (Marcus, now).

¬ alive (Marcus, now)

|(9, substitution)

Dead (Marcus, now)

|(10, substitution)

Pompeian (Marcus) ∧ gt (now, t_1)

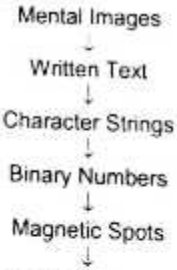
|(5, substitution)

Pompeian (Marcus) ∧ *gt* (now, 79)
|{2}
gt (now, 79)
|{8, substitute Equals}
gt (1991, 79)
True

Q.18. What is the difference between knowledge representation and knowledge acquisition? (2014-15)

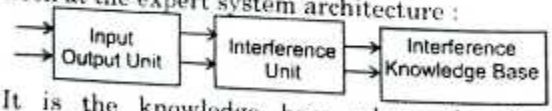
***OR* Describe meaning of knowledge representation and knowledge acquisition.**

Ans. Given the fact that knowledge is important and is in fact essential for intelligent behaviour, the representation of knowledge has become one of AI's top research priorities. As defined previously, knowledge consists of facts, concepts, rules etc. It can be represented in different forms such as mental images in one's thoughts, as spoken or written words in some language, as graphical or other pictures and as character strings or collection of magnetic spots stored in a computer. Figure shows different levels of knowledge representation



The most common of representation scheme is first order predicate logic. Propositional logic is another method of representation. The other representation schemes include frames, and associative networks, fuzzy logic, model logic and object oriented methods.

Look at the expert system architecture :



It is the knowledge base where the facts and knowledge are represented and stored. Representation means that the facts/knowledge is stored in knowledge base using propositional/predicate logic, frames, associative networks etc.

Example : we want to represent some facts as out the car 'Ford' using frames. We can do like this

Ford	(AKO	(VALUE Car))
	(COLOR	(VALUE Silver))
	(MODEL	(VALUE 4-door))
	(WEIGHT	(VALUE 2600))
	(FUEL – CAP	(VALUE 18))

These facts are represented in knowledge base. We can store facts using predicate logic too. Consider sentences

(1) John likes all kind of food.
(2) Apples are food.
(3) Jill is married to marry.

These are represented as

(1) $\forall x$: likes (John, x) → food (x)
(2) Food (apple)
(3) Married to (Jill, Marry)

So, we see that knowledge representation is a way of representing/storing facts/knowledge in knowledge base.

Knowledge Acquisition : One of the greatest bottlenecks in building knowledge rich systems is the acquisition and validation of the knowledge. Knowledge can come from various sources such as experts, text books, reports, technical articles etc. To be useful, the knowledge must be accurate, presented at the right level for encoding, complete in the sense that all essential facts and rules are included, free of inconsistencies and so on. Eliciting facts, heuristics, procedures and rules from an expert is a slow and tedious process. This has led to the development of some advanced acquisition tools including a variety of intelligent editors. Editors which provide help to knowledge engineers and system users.

The acquisition problem has also stimulated much research in machine learning systems i.e. systems which can learn new knowledge without the help of humans. Since knowledge based systems depend on large quantities of high quality knowledge for their success, it is essential that better methods of acquisition, refinement and validation be developed. The ultimate goal is to develop techniques that permit systems to learn new knowledge automatically and continually improve the quality of the knowledge they process. Thus, knowledge acquisition is the process of acquiring knowledge that is to be represented in

a knowledge base. Accumulation of knowledge is the other word for knowledge acquisition.

Q.19. Convert the following sentences to FOPL : (2014-15)

(1) Jack owns a dog

(2) Every dog owner is an animal lover.

(3) No animal lover kills an animal.

(4) Either Jack or Curiosity killed the cat, who is named Tuna.

Also prove by resolution – Did curiosity kill the cat.

Ans. (1) $(\exists x)\ \text{Dog}(x) \wedge \text{Owns}(\text{Jack}, x)$

(2) $(\forall x)\ ((\exists y)\ \text{Dog}(y) \wedge \text{Owns}(x, y)) \Rightarrow \text{AnimalLover}(x)$

(3) $(\forall x)\ \text{AnimalLover}(x) \Rightarrow (\forall y)\ \text{Animal}(y) \Rightarrow \neg\text{Kills}(x, y)$

(4) $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$

(5) $\text{Cat}(\text{Tuna})$

(6) $(\forall x)\ \text{Cat}(x) \Rightarrow \text{Animal}(x)$

Did curiosity kill the cat

(a) $\text{Dog}(\text{spike})$

(b) $\text{Owns}(\text{Jack}, \text{spike})$

(c) $\neg\text{Dog}(y) \vee \neg\text{Owns}(x, y) \vee \text{AnimalLover}(x)$

(d) $\neg\text{AnimalLover}(x1) \vee \neg\text{Animal}(y1) \vee \neg\text{Kills}(x1, y1)$

(e) $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$

(f) $\text{Cat}(\text{Tuna})$

(g) $\neg\text{Cat}(x2) \vee \text{Animal}(x2)$

Q.20. Using propositional linear resolution, show the following propositional sentence is unsatisfiable. Convert this sentence to clause form and derive the empty clause using resolution : (2014-15)

$$(P \vee Q \vee \neg R) \vee ((\neg R \vee Q \vee P) \Rightarrow (R \vee Q) \wedge \neg Q \wedge \neg P)$$

Ans. $(P \wedge Q \vee \neg R) \vee ((R \vee Q \vee P) \Rightarrow (R \vee Q) \wedge \neg Q \wedge \neg P)$

Elimination $\Rightarrow$

$(P \wedge Q \vee \neg R) \vee (\neg(\neg R \vee Q \vee P) \wedge (R \vee Q) \wedge \neg Q \wedge \neg P)$

$(P \wedge Q \vee \neg R) \vee (\neg\neg R \vee \neg Q \vee \neg P) \vee \neg(R \vee Q) \wedge \neg\neg Q \wedge \neg\neg P)$

Elimination negation $\neg$ and applying De Morgan law

$(P \wedge Q \vee \neg R) \vee (R \wedge \neg Q \wedge \neg P) \wedge (R \vee Q) \vee Q \vee P)$

Clause :

(1) $(P \wedge Q \vee \neg R) \vee (R \wedge \neg Q \wedge \neg P)$

(2) $(R \vee Q) \vee Q \vee P)$

Empty Clause :

(1), (2) $(P \wedge Q \vee \neg R) \vee (R \wedge \neg Q \wedge \neg P), (R \vee Q) \vee Q \vee P)$

Q.21. Explain the various terms used in Propositional Logic (PL).

Ans. Propositional Logic (PL) : A proposition is a statement, which in English would be a declarative sentence. Every proposition is either TRUE or FALSE.

Examples :

(1) The sky is blue.

(2) Snow is cold

(3) $12 \times 12 = 144$

(4) Propositions are "sentences", either true or false but not both.

(5) A sentence is smallest unit in propositional logic.

(6) If proposition is true, then truth value is "true".

(7) If proposition is false, then truth value is "false".

Example :

Sentence	Truth value	Proposition (Y/N)
"Grass is green"	"true"	Yes
"2 + 5 = 5"	"false"	Yes
"Close the door"	-	No
"Is it hot outside?"	-	No
"x > 2" where x is variable	-	No (since x is not defined)
"x = x"	-	No (Don't know what is "x" and "=","3 = 3" or "air is equal to air" or "Water is equal to water has no meaning")

(a) Propositional logic is fundamental to all logic.

(b) Propositional logic is also called propositional calculus, sentential calculus, or Boolean algebra

(c) Propositional logic tells the way of joining and/or modifying entire propositions, statements or sentence, as well as the logical relationships and properties that are derived from the methods of combining or altering statements.

Statement, Variables and Symbols : These and few more related terms, such as, connective, truth value, contingencies, tautologies, contradictions, antecedent, consequent, argument are explained below.

(1) **Statement :** Simple statements (sentences), TRUE or FALSE, that does not contain any other statement as a part, are basic propositions; lower-case letters, p, q,

r are symbols for simple statements. Large compound or complex statement are constructed from basic propositions by combining them with connectives.

(2) **Connective or Operator :** The connectives join simple statements into compounds, and join compounds into larger compounds. Table below indicates, the basic connectives and their symbols :

(a) Listed in decreasing order of operation priority.

(b) Operations with higher priority are solved first.

Example of a formula : $(((a \wedge \neg b) \vee c \rightarrow d) \leftrightarrow \neg(a \vee c))$

Connectives and Symbols in decreasing order of operation priority :

Connective	Symbols					Read as
Assertion	p					"p is true"
Negation	¬p	~	!		NOT	"p is false"
Conjunction	p ∧ q	*	&&	&	AND	"both p and q are true"
Disjunction	p ∨ q	| |	|		OR	"either p is true, or q is true, or both"
Implication	p→q	⊃	⇒		if ..then	"If p is true, then q is true" "p implies q"
Equivalence	↔	≡	⇔		if and only if	" p and q are either both true or both false"

Figure

Note : The propositions and connectives are the basic elements of propositional logic.

(3) **Truth Value :** The truth value of a statement is its TRUTH or FALSITY.

Example :

p is either TRUE or FALSE.

$\neg p$ is either TRUE or FALSE.

$p \vee q$ is either TRUE or FALSE, and so on.

Use "T" or "1" to mean TRUE.

Use "F" or "0" to mean FALSE

Truth Table defining the Basic Connectives :

p	q	¬p	¬q	p ∧ q	p ∨ q	p→q	p ↔ q	q→p
T	T	F	F	T	T	T	T	T
T	F	F	T	F	T	F	F	T
F	T	T	F	F	T	T	F	F
F	F	T	T	F	F	T	T	T

(4) **Tautologies :** A proposition that is always true is called a "tautology".

E.g. $(p \vee \neg p)$ is always true regardless of the truth value of the proposition p.

(5) **Contradictions :** A proposition that is always false is called a "contradiction".

E.g. $(p \wedge \neg p)$ is always true regardless of truth value of the proposition p.

(6) **Contingencies :** A proposition is called a "contingency", if that proposition is neither a tautology nor a contradiction.

E.g. $(p \vee q)$ is a contingency.

(7) **Antecedent, Consequent :** These two are parts of conditional statements.

In the conditional statements, $p \rightarrow q$, the 1st statement or "if-clause" (here p) is called antecedent. 2nd statement or "then-clause" (here q) is called consequent.

(8) **Argument :** An argument is a demonstration or a proof of some statement.

Example : "That bird is a crow; therefore, it's black.". Any argument can be expressed as a compound statement. In logic, an argument is a set of one or more meaningful declarative sentence (or "proposition") known as the premises along with another meaningful declarative sentences (or "propositions") known as the conclusion.

(a) **Premise :** Premise is a proposition which gives reasons, grounds, or evidence for accepting some other proposition, called the conclusion.

(b) **Conclusion :** Conclusion is a proposition, which is purported to be established on the basis of other propositions.

Take all the premises, conjoin them and make that conjunction the antecedent of a conditional and make the conclusion the consequent. This implication statement is called the corresponding conditional of the argument.

(1) An argument is valid "if and only if" its corresponding conditional is a tautology.

(2) Two statements are consistent "if and only if" their conjunction is not a contradiction.

(3) Two statements are logically equivalent "if and only if" their truth table columns are identical.

"if and only if" the statement of their equivalence using " ≡ " is a tautology.

Q.22.Define and describe the difference between Knowledge, Belief, Hypothesis and Data.

Ans. Definition :

Knowledge : Knowledge is defined as :

(1) Expertise, and skills acquired by a person through experience or education; the theoretical or practical understanding of a subject.

(2) What is known in a particular field or in total; facts and information?

(3) Awareness or familiarity gained by experience of a fact or situation.

Belief : Belief is the psychological state in which an individual holds a proposition or premise to be true. Belief can be true or false.

Hypothesis : A hypothesis consists either of a suggested explanation for a phenomenon or of a reasoned proposal suggesting a possible correlation between multiple phenomena. The term derives from the Greek, hypothesis meaning "to put under" or "to suppose."

Data : Data are raw facts and figures.

Difference between Knowledge, Belief, Hypothesis and Data : We define belief as essentially any meaningful and coherent expression that can be represented. Thus, a belief may be true or false. A hypothesis is a justified belief that is not known to be true. Thus, a hypothesis is a belief which is backed up with some supporting evidence, but it may still be false. Knowledge is a true justified belief and data are raw facts and figures which is the base for knowledge, hypothesis and belief.

Q.23.Describe the First-Order Logic model.

Ans. Models for First-Order Logic :

(1) It is an organization of formal structure representing possible world under different assumptions.

(2) The propositional logic's model is very simple which holds set of truth values for the propositional symbol.

(3) The most interesting models for first-order logic is shown in figure.

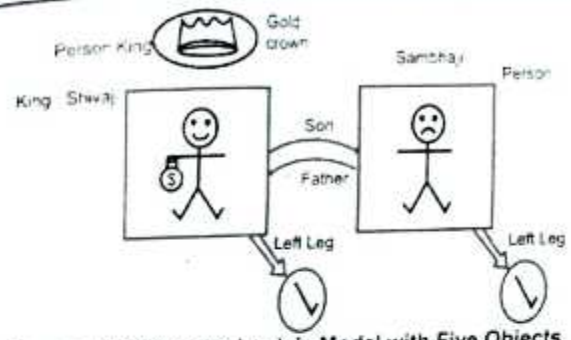


Figure : A First-order Logic's Model with Five Objects, Five Relations, One Unary Function, Left Leg Two Binary Relations, Three Unary Relations

(4) **Domain :** It is set of objects which are part of model. All objects are sometime called as domain elements.

(5) **Five Objects Are :** King Shivaji, Sambhaji (son of Shivaji), left leg of Shivaji, left leg of Sambhaji and Gold crown.

(6) **Five Relations Are :**

(a) **Binary :**
- (i) Sambhaji is great son of Shivaji
- (ii) Shivaji is great father of Sambhaji

<King Shivaji, Sambhaji><Sambhaji, King Shivaji>

(b) **Unary :**
- (i) On head- one tuple <the crown, King Shivaji>
- (ii) Person –King as an person

(c) **Person :** Sambhaji as one person.
Tuple : Relation is a set of tuples of the object which are related.

(d) **Function :** Left leg is a unary function
<King Shivaji> → Shivaji's left leg.
<Sambhaji> → Sambhaji left leg.

(e) **Total Function :** Total function must is first-order logic. An input tuple must be with a value.

Q.24.Consider the following sentences :

(1) **John likes all kinds of food.**
(2) **Apples are food.**
(3) **Children are food.**
(4) **Anything anyone eats and is not killed by is food.**
(5) **Jack eats peanuts and is still alive.**
(6) **Jill eats everything jack eats.**

Represent these sentences in predicate logic and prove that "John likes peanuts" using deduction or resolution.

Ans. (1) $\forall x$: likes (John, x) $\rightarrow$ food (x)

(2) food (apple)

(3) food (Children)

(4) $\forall x$: $\forall y$ eats (x, y) $\wedge^{\sim}$ killed (x) $\rightarrow$ food (y)

(5) eats (jack, Peanuts) $\rightarrow$ alive (Jack)

(6) $\forall x$: eats (Jill, x) $\rightarrow$ eats (Jack, x)

Now that facts are represented, they can now be used for reasoning. Let's assume "John likes peanuts" to be true and proceed.

Likes (John, Peanuts)

↓ rule 1 & substitute x with Peanuts

food (Peanuts)

↓ rule 4 & substitution

eats (x, Peanuts) $\wedge^{\sim}$ killed (x)

↓ rule 5 & substitution

alive (Jack) $\wedge^{\sim}$ killed (Jack)

↓ true

Which proves that John likes Peanuts as alive (Jack) $\wedge^{\sim}$ killed (Jack) would return TRUE. TRUE means that the final output is TRUE for our assumption that John likes Peanuts.

□□

UNIT – FOUR

MACHINE LEARNING

Q.1. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state. (2018-19)

Ans. For the case in which each movement costs one point, the agent should stop after checking both squares A and B and removing any dirt found in order to reduce needless movements and loss of points. The corresponding agent program would require internal state in this case, as it would need to remember checking both squares before it stops performing its task. If it finds dirt at square A, cleans it, then moves to square B and finds no dirt, and then does not remember whether or not it checked square A, it will check A again, and then repeat with square B and continue making unnecessary movements, thus losing points.

If the squares did not permanently remain clean, however, the agent could stop for some fixed amount of time after checking and cleaning both squares and repeat its task. It could work at some time interval, checking both squares, cleaning any dirt found, stopping for 1 hour or so, and then repeating the process. This would allow the agent to minimize movements and point loss while keeping the squares clean if they were to become dirty again.

Things to consider are; "the performance measure that defines the criterion of success, the agent's prior knowledge of the environment, the actions that the agent can perform, and the agent's percept sequence to date" (Russell & Norvig, 37). The given assumptions of the vacuum-cleaner agent are listed below :

(1) "The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.

(2) The "geography" of the environment is know a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.

(3) The only available actions are Left, Right, and Suck.
(4) The agent correctly perceives its location and whether that location contains dirt".

Remember that "a rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome"

The agent knows everything about it's environment, but not it's initial location or the dirt distribution. The agent understands its location and whether it contains dirt or not. The only option for the agent is to move left, right, and suck. There are only 2 squares in the environment, so the only actions the agent can take are; is the square dirty? yes -> suck, no move left or right since it doesn't know it's initial location -> moves left bump goes right, moves right bump goes left, then is the square dirty? yes -> suck. Both squares are now clean. Clean squares will stay clean, and now the agent is done.

Based on the above assumptions then, and the definition of a rational agent, you can clearly see that the vacuum-cleaner agent is indeed rational. The best outcome for the agent would be to have both squares clean, and since they remain clean once they've been cleaned the agent is done.

Yes, the agent program would require an internal state, that way the agent would know some more information about the environment and could reduce movement. An internal state "depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state" (Russell & Norvig, 50).

This kind of agent is known as a model-based reflex agent.

Each movement costs one point, and to minimize the movement and have a high clean rate, it makes sense to have an internal state. If we are still using the above vacuum-cleaner agent, there are only 2 squares, so by maintaining an internal state it will remember if it has visited and cleaned a square. With a different vacuum-cleaner agent that had more area to cover, this would also make sense because it would not have to revisit squares unless over time dirt accumulated and it needed to be re-cleaned. But initially during it's first cleaning, it would get the whole area with minimum movements and would keep it's point cost low.

Q.2. Assume two players, min and max, play nim (as described above). Min plays first. If a terminal state in the search tree developed above is a win for min, a utility function of zero is assigned to that state. A utility function of 1 is assigned to a state if max wins the game. Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree. (2018-19)

Ans. Game Tree : It is a structure in the form of a tree consisting of all the possible moves which allow you to move from a state of the game to the next state.

A game can be defined as a search problem with the following components :

(1) **Initial State :** It comprises the position of the board and showing whose move it is.

(2) **Successor Function :** It defines what the legal moves a player can make are.

(3) **Terminal State :** It is the position of the board when the game gets over.

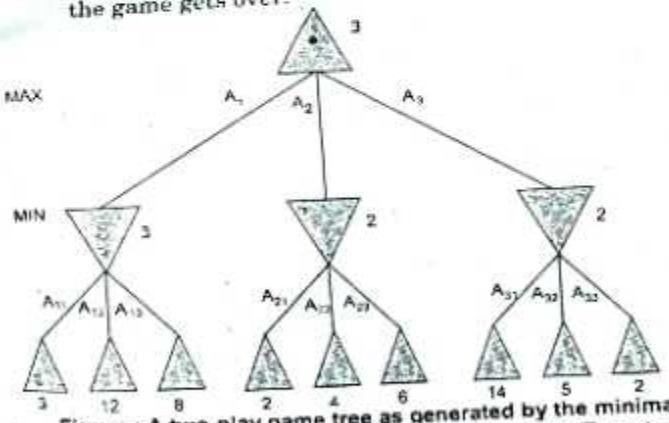


Figure : A two-play game tree as generated by the minimax algorithm. The △ nodes are moves by MAX and the ▽ nodes are moves by MIN. The terminal nodes show the utility value for MAX computed by the utility function (i.e., by the rules of the game), whereas the utilities of the other nodes are computed by the minimax algorithm from the utilities of their successors. MAX's best move is A_1 and MIN's best reply is A_{11}.

(4) **Utility Function :** It is a function which assigns a numeric value for the outcome of a game. For instance, in chess or tic-tac-toe, the outcome is either

a win, a loss, or a draw, and these can be represented by the values +1, -1, or 0, respectively. There are games that have a much larger range of possible outcomes; for instance, the utilities in backgammon varies from +192 to -192. A utility function can also be called a payoff function.

Minimax Algorithm :

```
function MINIMAX-DECISION (state) returns an action
inputs : state, current state in game
  v ← MAX - VALUE(state)
  return the action in SUCCESSOR (state) with value v
function MAX-VALUE (state) returns a utility value
if TERMINAL-TEST (state) then return UTILITY (state)
then return UTILITY (state)
  v ← ∞
  for a, s in SUCCESSORS (state) do
  v ← MAX(v, MIN-VALUE(s))
  return v
function MIN-VALUE(state) returns a utility value
if TERMINAL-TEST(state) then return UTILITY (state)
  v ← ∞
  for a, s in SUCCESSORS (state) do
  v ← MIN(v, MAX-VALUE(s))
  return v
```

Minimax – backward induction

Max-and Min-Value are dual recursive

Q.3. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn? (2018-19)

Ans. For the case in which clean squares can become dirty, the agent could work at some fixed time interval. It could check and clean both squares A and B, then stop for a fixed amount of time, then start up and repeat the process in order to keep the squares clean while minimizing needless moves and performance measurement point loss. If the geography of the environment is unknown, the agent would need to explore the environment instead of simply oscillating back and forth between squares A and B. I would say it would make sense for the agent to learn from its experience in these cases. It could learn the geography of the environment by exploring it, as [illegible] learning which squares it has checked and clean [illegible] it may temporarily stop once it has finished checking learning all of them.

Yes it makes sense for the agent to learn from its experience, so that it can eventually learn the geography of its location. Also it could learn how long a square stays clean or before a square would get dirty again. It would maximize it's efficiency, and make it a rational agent.

If not agent couldn't learn and there were performance measures as in part b, then it would have a high point count, because it would just go from square to square searching for dirt to clean. And the agent wouldn't be considered very effective, and also wouldn't be consider rational. On top of that say it didn't remember if it cleaned a square, the agent would never stop until it was drained of power.

Q.4. Explain in brief the concept of reinforcement learning. (2017-18)

OR **Describe major steps involved in a learning process. Also discuss how learning systems are classified. (2017-18)**

OR **What is machine learning? How many types of learning are there? (2015-16)**

OR **What is machine learning? Differentiate between supervised, unsupervised and reinforcement learning. (2014-15)**

Ans. Steps Involved in Learning Process : It follows the following steps in sequence as shown below :

(1) **Stimulus :** Stimulus refers those factors that motivate or inspire or induce learners to learn the required skills and knowledge. The factors that affect the learning process must be understood by the learner. Then only they can motivate themselves to learn. Stimulus exists in the environment in which the person lives and behaves.

(2) **Response :** It means the amount of interaction by the learners. Response emphasizes that there should be a positive response from the learners with regard to learning process and program. Because the regular and timely response provides opportunity to determine the level of teaching. This also helps to explore to what extent learners are improving.

(3) **Motivation :** Stimulus and response are not enough for an effective learning process, rather the learners must be motivated to impart required skills and

knowledge from the training. Hence, an effective motivation package consisting of rewards and prizes should be provided to the learners to motivate them.

(4) **Reward :** It is an incentive which satisfies the learner from learning. If a motivated learner is rewarded, he/she will be inspired for further performance. Reward further stimulates the learners.

Machine Learning : Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Differences :

Supervised Learning :

(1) A human builds a classifier based on input and output data

(2) That classifier is trained with a training set of data

(3) That classifier is tested with a test set of data

(4) Deployment if the output is satisfactory

To be used when, "I know how to classify this data, I just need you (the classifier) to sort it."

Point of Method : To class labels or to produce real numbers

Unsupervised Learning :

(1) A human builds an algorithm based on input data

(2) That algorithm is tested with a test set of data (in which the algorithm creates the classifier)

(3) Deployment if the classifier is satisfactory

To be used when, "I have no idea how to classify this data, can you (the algorithm) create a classifier for me?"

Point of Method : To class labels or to predict (PDF)

Reinforcement Learning :

(1) A human builds an algorithm based on input data

(2) That algorithm presents a state dependent on the input data in which a user rewards or punishes the algorithm via the action the user took, this continues over time

(3) That algorithm learns from the reward/punishment and updates itself, this continues

(4) It's always in production, it needs to learn real data to be able to present actions from states

To be used when, "I have no idea how to classify this data, can you classify this data and I'll give you a reward if it's correct or I'll punish you if it's not.

Q.5. Discuss various application domains of machine learning. **(2017-18)**

Ans. Application Domains of Machine Learning : There are many various application of machine learning :

(1) **Financial Services :** Companies in the financial sector are able to identify key insights in financial data as well as prevent any occurrences of financial fraud, with the help of machine learning technology. The technology is also used to identify opportunities for investments and trade. Usage of cyber surveillance helps in identifying those individuals or institutions which are prone to financial risk, and take necessary actions in time to prevent fraud.

(2) **Marketing and Sales :** Companies are using machine learning technology to analyze the purchase history of their customers and make personalized product recommendations for their next purchase. This ability to capture, analyze, and use customer data to provide a personalized shopping experience is the future of sales and marketing.

(3) **Government :** Government agencies like utilities and public safety have a specific need FOR Ml, as they have multiple data sources, which can be mined for identifying useful patterns and insights. For example sensor data can be analyzed to identify ways to minimize costs and increase efficiency. Furthermore, ML can also be used to minimize identity thefts and detect fraud.

(4) **Healthcare :** With the advent of wearable sensors and devices that use data to access health of a patient in real time, ML is becoming a fast-growing trend in healthcare. Sensors in wearable provide real-time patient information, such as overall health condition, heartbeat, blood pressure and other vital parameters. Doctors and medical experts can use this informatio

to analyze the health condition of an individual, find a pattern from the patient history, and predict the occurrence of any ailments in the future. This technology also empowers medical experts to analyze data to identify trends that facilitate better diagnosis and treatment.

(5) **Transportation :** Based on the travel history and pattern of traveling across various routes, machine learning can help transportation companies predict potential problems that could arise on certain routes and accordingly advise their customers to opt for a different route. Transportation firms and delivery organizations are increasingly using machine learning technology to carry out data analysis and data modeling to make informed decisions and help their customers make smart decisions when they travel.

(6) **Oil and Gas :** This is perhaps the industry that needs the application of machine learning the most. Right from analyzing underground minerals and finding new energy sources to streaming distribution, ML applications for this industry are vast and are still expanding.

Q.6. Explain the statistical nature of the learning process. (2015-16)

***OR* Explain how Bayesian statistics provide reasoning under various kinds of uncertainty.** (2015-16)

***OR* Write short notes on :**

(1) Statistical Learning.

(2) Naive Bayes Model.

(3) Reinforcement Learning.

Ans. (1) Statistical Learning :

(a) Training Set, Data : Evidence-instantiation of all or some of the random variables describing the domain.

(b) Hypotheses : Hypotheses are probabilistic theories of how the domain works.

Example : Suppose there are five kinds of bags of candies : 10% are h_1 : 100% cherry candies, 20% are h_2 : 75% cherry candies + 25% lime candies, 40% are h_3 : 50% cherry candies + 50% lime candies, 20% are h_4 : 25% cherry candies + 75% lime candies, 10% are h_5 : 100% lime candies.

(c) Bayesian Learning : Calculates the probability of each hypothesis, given the data, and makes predictions on that basis.

(d) The predictions are made by using all the hypotheses, weighted by their probabilities, rather than by using a single "best" hypothesis.

(e) Thus, learning is reduced to probabilistic inference.

View learning as Bayesian updating of a probability distribution over the hypothesis space. H is the hypothesis variable, values h_1, h_2, ... prior (unconditional) probabilities $P(H)$ j^{th} observation d_j gives the outcome of random variable D_j training data $d=d_1,.....d_N$. Given the data so for, each hypothesis has a posterior (conditional) probability :

$$P(h_i \mid d) = \alpha\, P(d \mid h_i) P(h_i)$$

Where $P(d \mid h_i)$ is called the likelihood of the data under each hypothesis. Predictions of unknown quantity X, use a likelihood-weighted average over the hypothesis :

$$P(X \mid d) = \sum_i P(X \mid d, h_i) P(h_i \mid d)$$

$$= \sum_i P(X \mid h_i) P(h_i \mid d)$$

Where we assume that each hypothesis determines a probability distribution over X.

(2) Naive Bayes Model : Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n. Bayes' theorem states the following relationship : $P(y \mid x_1,\ldots,x_n) = \dfrac{P(y)P(x_1,\ldots,x_n \mid y)}{P(x_1,\ldots,x_n)}$

Using the naive independence assumption that $P(x_i \mid y, x_1,\ldots,x_{i-1}, x_{i+1},\ldots,x_n) = P(x_i \mid y)$.

For all i, this relationship is simplified to

$$P(y \mid x_1,\ldots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1,\ldots,x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

And we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set. The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Gaussian Naive Bayes : GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

(3) **Reinforcement Learning :** Reinforcement learning refers to a class of problems in machine learning which postulate an agent exploring an environment in

(a) The agent perceives its current state and takes actions.

(b) The environment provides a reward to the agent (may be positive or negative).

(c) The algorithm adapts the agent's behavior so as to maximize expected cumulative reward for the agent over the course of the problem.

In other words, the definition of reinforcement learning is :

"A computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment."

Reinforcement learning is "a way of programming agents by reward and punishment without needing to specify how the task is to be achieved".

Q.7. Explain the expectation and maximization (EM) algorithm for finding the maximum likelihood with hidden variables. **(2014-15, 2015-16)**

OR **Explain EM (Expectation-Maximization) algorithm with limitations.**

Ans. EM Algorithm : Expectation-Maximization (EM) is a technique used in point estimation. Given a set of observable variables X and unknown (latent) variables Z we want to estimate parameters θ in a model.

Notation : X Observed variables.

Z Latent (unobserved) variables.

θ^t The estimate of the parameters at iteration t.

$l(\theta)$ The marginal log-likelihood $\log p(x \mid \theta)$.

$\log p(x, z \mid \theta)$ The complete log-likelihood, i.e., when we know the value of Z.

$q(z \mid x, \theta)$ Averaging distribution, a free distribution that EM gets to vary.

$Q(\theta \mid \theta^t)$ The expected complete log-likelihood $\sum_z q(z \mid x, \theta) \log p(x, z \mid \theta)$.

$H(q)$ Entropy of the distribution $q(z \mid x, \theta)$.

We could directly maximize $l(\theta) = \sum_x \log p(x, z \mid \theta)$ using a gradient method (e.g. gradient ascent, conjugate gradient, quasi-Newton) but sometimes the gradient is hard to compute, hard to implement, or we do not want to bother adding in a black-box optimization routine. If we have the following inequality

$$l(\theta) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta) \quad ...(1)$$

$$= \log \sum_z q(z \mid x, \theta) \frac{p(x, z \mid \theta)}{q(z \mid x, \theta)} \quad ...(2)$$

$$\geq \sum_z q(z \mid x, \theta) \log \frac{p(x, z \mid \theta)}{q(z \mid x, \theta)} \equiv F(q, \theta) \quad ...(3)$$

Where $q(z \mid x, \theta)$ is an arbitrary density over Z. This inequality is foundational to what are called "variational methods" in the machine learning literature. Instead of maximizing $l(\theta)$ directly, EM maximizes the lower-bound $F(q, \theta)$ via coordinate ascent

E-step : $q^{(t+1)} = \underset{2}{\arg\max}\, F(q, \theta^{(t)})$ ...(4)

M-step : $\theta^{(t+1)} = \underset{\theta}{\arg\max}\, F(q^{t+1}, \theta)$ ...(5)

Starting with some initial value of the parameters $\theta^{(0)}$, one cycles between the E and M-steps until $\theta^{(t)}$ converges to a local maxima. Computing equation (4) directly involves fixing $\theta = \theta^{(t)}$ and optimizing over the space of distributions, which looks painful. However, it is possible to show that $q^{(t+1)} = p(z \mid x, \theta^{(t)})$. We can stop worrying about q as a variable over the space of distributions, since we know the optimal q is a distribution that depends on $\theta^{(t)}$. To compute equation 5 we fix q and note that

$$l(\theta) \geq F(q, \theta) \quad ...(6)$$

$$= \sum_z q(z \mid x, \theta) \log \frac{p(x, z \mid \theta)}{q(z \mid x, \theta)} \quad ...(7)$$

$$= \sum_z q(z \mid x, \theta) \log p(x, z \mid \theta) - \sum_z q(z \mid x, \theta) \log q(z \mid x, \theta) \quad ...(8)$$

$$= Q(\theta \mid \theta^{(t)}) + H(q) \quad ...(9)$$

So maximizing $F(q, \theta)$ is equivalent to maximizing the expected complete log-likelihood. Obscuring these details, which explain what EM is doing, we can re-express equations 4 and 5 as

E-step : Compute $Q(\theta \mid \theta^{(t)}) = E_{p(z \mid x, \theta^{(t)})} [\log p(x, z \mid \theta)]$ ...(10)

M-step : $\theta^{(t+1)} = \underset{\theta}{\arg\max}\, E_{p(z \mid x, \theta^{(t)})} [\log p(x, z \mid \theta)]$ ...(11)

Limitations of EM : EM is useful several reasons : conceptual simplicity, ease of implementation, and the fact that each iteration improves $l(\theta)$. The rate of convergence on the first few steps is typically quite good, but can become excruciatingly slow as you approach local optima. Generally, EM works best when the fraction of missing information is small and the dimensionality of the data is not too large. EM can require much iteration, and higher dimensionality can dramatically slow down the E-step.

Q.8. Explain Decision Tree with example.

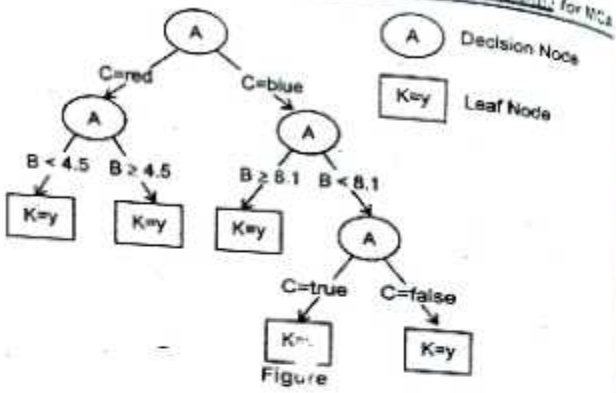
Ans. Decision Trees : Decision trees are powerful tools for classification and prediction. Decision trees represent rules. Rules are easily expressed so that humans can understand them or even directly use in a database access language like SQL so that records falling into a particular category may be retrieved.

Description :

(1) Decision tree is a classifier in the form of a tree structure where each node is either a leaf or decision node.
   - **(a) Leaf Node :** Indicates the target attribute (class) values of examples.
   - **(b) Decision Node :** Specify test to be carried on an attribute-value.

(2) Decision tree is a typical inductive approach to learn knowledge on classification. The conditions are :
   - **(a) Attribute-value Description :** Object or case must be expressible as a fixed collection of properties or attributes having discrete values.
   - **(b) Predefined Classes :** Categories to which examples are to be assigned must already be defined (i.e., supervised data).
   - **(c) Discrete Classes :** Classes must be sharply delineated; continuous classes broken up into vague categories as "hard", "flexible", "soft".
   - **(d) Sufficient Data :** Enough training cases to distinguish valid patterns.

Example : A simple decision tree



Figure

UNIT – FIVE

PATTERN RECOGNITION

Q.1. Discuss back propagation algorithm for learning in multilayer neural network. (2018-19)

Ans. Back Propagation : Back propagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks). But, some of you might be wondering why we need to train a Neural Network or what exactly is the meaning of training.

While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.

Any complex system can be abstracted in a simple way, or at least dissected to its basic abstract components. Complexity arises by the accumulation of several simple layers. The goal of this post, is to explain how neural networks work with the most simple abstraction. We will try to reduce the machine learning mechanism in NN to its basic abstract components.

Backpropagation Algorithm :

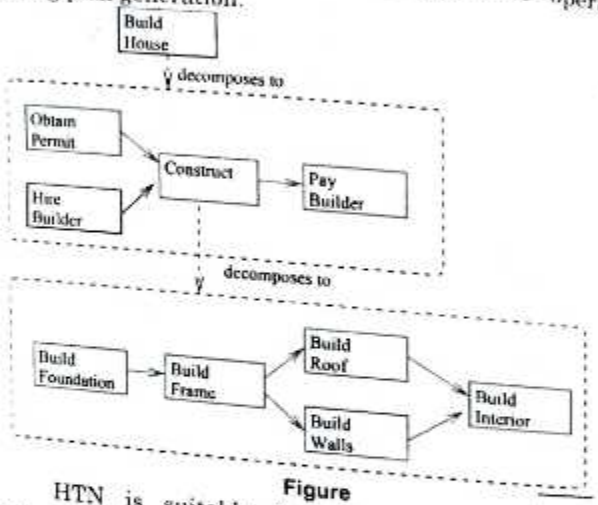
```
initialize network weights (often small random values)
do
  forEach training example named ex
    prediction = neural-net-output(network, ex)  // forward pass
    actual = teacher-output(ex)
    compute error (prediction - actual) at the output units
  compute (displaystyle Delta w_{h}) for all weights from hidden layer to output layer // backward pass
    compute (displaystyle Delta w_{i}) for all weights from input layer to hidden layer  // backward pass continued
    update network weights // input layer not modified by error estimate
  until all examples classified correctly or another stopping criterion satisfied
  return the network
```

Q.2. Describe the planning method based on hierarchical task networks with an example. (2018-19)

Ans. Idea : Many tasks in real life already have a built-in hierarchical structure

For Example : A computational task, a military mission, an administrative task It would be a waste of time to construct plans from individual operators. Using the built-in hierarchy help escape from exponential explosion

Running Example : The activity of building a house consists of obtaining the necessary permits, finding a builder, constructing the exterior/interior. HTN approach use abstract operators as well as primitive operators during plan generation.



Figure

HTN is suitable for domains where tasks are naturally organized in a hierarchy. Uses abstract operators to start a plan. Use partial-order planning techniques and action decomposition to come up with the final plan The final plan contains only primitive operators. What is to be considered primitive is subjective: what an agent considers as primitive can be another agent's plans.

Algorithm :

(1) Input a planning problem P.
(2) If P contains only primitive tasks, then resolve the con icts in P and return the result. If the con icts cannot be resolved, return failure.
(3) Choose a non-primitive task t in P.
(4) Choose an expansion for t.
(5) Replace t with the expansion.

(6) Use critics to find the interactions among the tasks in P, and suggest ways to handle them.
(7) Apply one of the ways suggested in step 6.
(8) Go to step 2.

In artificial intelligence, hierarchical task network (HTN) planning is an approach to automated planning in which the dependency among actions can be given in the form of hierarchically structured networks.

Planning problems are specified in the hierarchical task network approach by providing a set of tasks, which can be :

(1) Primitive tasks, which roughly correspond to the actions of STRIPS;
(2) Compound tasks, which can be seen as composed of a set of simpler tasks.
(3) Goal tasks, which roughly corresponds to the goals of STRIPS, but are more general.

A solution to an HTN problem is then an executable sequence of primitive tasks that can be obtained from the initial task network by decomposing compound tasks into their set of simpler tasks, and by inserting ordering constraints.

A primitive task is an action that can be executed directly given the state in which it is executed supports its precondition. A compound task is a complex task composed of a partially ordered set of further tasks, which can either be primitive or abstract. A goal task is a task of satisfying a condition. The difference between primitive and other tasks is that the primitive actions can be directly executed. Compound and goal tasks both require a sequence of primitive actions to be performed; however, goal tasks are specified in terms of conditions that have to be made true, while compound tasks can only be specified in terms of other tasks via the task network outlined below.

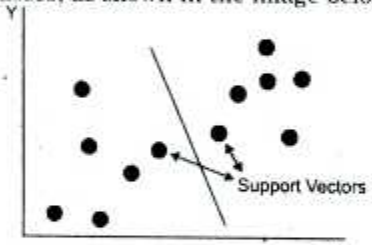
Constraints among tasks are expressed in the form of networks, called (hierarchical) task networks. A task network is a set of tasks and constraints among them. Such a network can be used as the precondition for another compound or goal task to be feasible. This way, one can express that a given task is feasible only if a set of other actions (those mentioned in the network) are done, and they are done in such a way that the constraints among

them (specified by the network) are satisfied. One particular formalism for representing hierarchical task networks that has been fairly widely used is TAEMS

Q.3. Write a short note on Support Vector Machine. (2017-18)

OR **Write short note on Support vector machine (SVM). (2015-16)**

Ans. SVM : A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems and as such, this is what we will focus on in this post. SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.



Figure

Support Vectors : Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

Advantages of Support Vector Machines :

(1) Accuracy
(2) Works well on smaller cleaner datasets
(3) It can be more efficient because it uses a subset of training points

Disadvantages of Support Vector Machines :

(1) Isn't suited to larger datasets as the training time with SVMs can be high
(2) Less effective on noisier datasets with overlapping classes

Uses of Support Vector Machines : SVM is used for text classification tasks such as category assignment, detecting spam and sentiment analysis. It is also commonly used for

image recognition challenges, performing particularly well in aspect-based recognition and color-based classification. SVM also plays a vital role in many areas of handwritten digit recognition, such as postal automation services.

Q.4. What is pattern recognition? Explain various steps involved in the designing of a pattern recognition system with the help of a diagram. (2017-18)

OR **What do you mean by classification? Discuss the process of classification with the help of a diagram. (2017-18)**

OR **Design principles of pattern recognition system. (2015-16)**

OR **What do you understand by pattern recognition? Differentiate between structured description and symbolic description. (2015-16)**

OR **With the help of a suitable diagram, explain the classification process in a pattern recognition system. (2014-15)**

OR **Discuss the complete process of pattern recognition and give various methods of object classification.**

Ans. Pattern Recognition : Pattern recognition is "the act of taking in raw data and taking an action based on the category of the pattern". Most research in pattern recognition is about methods for supervised learning and unsupervised learning. "Pattern recognition is the research area that studies the operation and design of systems that recognize patterns in data. It encloses sub-disciplines like discriminant analysis, feature extraction, error estimation, cluster analysis (together sometimes called statistical pattern recognition), grammatical inference and parsing (sometimes called syntactical pattern recognition). Important application areas are image analysis, character recognition, speech analysis, man and machine diagnostics, person identification and industrial inspection."

Pattern recognition aims to classify data (patterns) based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. This is in contrast to pattern matching, where the pattern is rigidly specified. A

complete pattern recognition system consists of a sensor that gathers the observations to be classified or described, a feature extraction mechanism that computes numeric or symbolic information from the observations, and a classification or description scheme that does the actual job of classifying or describing observations, relying on the extracted features.

The classification or description scheme is usually based on the availability of a set of patterns that have already been classified or described. This set of patterns is termed the training set, and the resulting learning strategy is characterized as supervised learning. Learning can also be unsupervised, in the sense that the system is not given an a priori labeling of patterns, instead it itself establishes the classes based on the statistical regularities of the patterns.

The classification or description scheme usually uses one of the following approaches : statistical (or decision theoretic) or syntactic (or structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Syntactical (or structural) pattern recognition is based on the structural interrelationships of features. A wide range of algorithms can be applied for pattern recognition, from simple naive bayes classifiers and neural networks to the powerful KNN decision rules.

Pattern recognition is more complex when templates are used to generate variants. For example, in English, sentences often follow the "N-VP" (Noun - Verb Phrase) pattern, but some knowledge of the English language is required to detect the pattern. Pattern recognition is studied in many fields, including psychology, ethology, cognitive science and computer science. Holographic associative memory is another type of pattern matching where a large set of learned patterns based on cognitive meta-weight is searched for a small set of target patterns.

Within medical science, pattern recognition is the basis for Computer Aided Diagnosis (CAD) systems. CAD describes a procedure that supports the doctor's interpretations and findings. Typical applications are automatic speech recognition, classification of text into several categories (e.g. spam/non-spam email

messages), the automatic recognition of handwritten postal codes on postal envelopes, or the automatic recognition of images of human faces. The last two examples form the subtopic image analysis of pattern recognition that deals with digital images as input to pattern recognition systems.

Example :

Fingerprint Recognition : "Fingerprinting is an authentication technique that has helped law enforcement officials identify potential criminals for decades, but recently it has started to gain wider usage. The technique is emerging as the most popular form of biometrics, and much of the budding interest is coming from government agencies looking to enhance physical security, such as access to buildings. Corporations are also making a move toward using fingerprinting technology to provide more reliable identification of employees, business partners and customers.

Symbolic Description : Symbolic processing involves attributes and non algorithmic processing. For example, the "production system" is an ideal vehicle for symbolic processing, where information is stored as IF-THEN rules.

Structured Description : Representing knowledge using logical formalism, like predicate logic, has several advantages. They can be combined with powerful inference mechanisms like resolution, which makes reasoning with facts easy. But using logical formalism complex structures of the world, objects and their relationships, events, sequences of events etc. cannot be described easily.

A good system for the representation of structured knowledge in a particular domain should posses the following four properties :

(1) **Representational Adequacy :** The ability to represent all kinds of knowledge that are needed in that domain.

(2) **Inferential Adequacy :** The ability to manipulate the represented structure and infer new structures.

(3) **Inferential Efficiency :** The ability to incorporate additional information into the knowledge structure that will aid the inference mechanisms.

(4) **Acquisitional Efficiency :** The ability to acquire new information easily, either by direct insertion or by program control.

Classification : Classification is a technique by which you determine to what group a certain observation belongs, such as when biologists categorize plants, animals, and other life forms into different taxonomies. It is one of the primary uses of data science and machine learning.

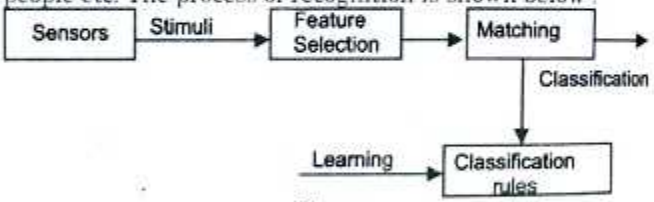
In order to determine the correct category for a given observation, machine learning technology does the following :

(1) Applies a classification algorithm to identify shared characteristics of certain classes.

(2) Compares those characteristics to the data you're trying to classify.

(3) Uses that information to estimate how likely it is that observation belongs to a particular class.

Classification Process in a Pattern Recognition System : Recognition is the process of establishing a close match between some new stimulus and previously stored stimulus patterns. Through visual sensing and recognition, we identify many special objects such as home, facts of people etc. The process of recognition is shown below :



Figure

The steps involved are :

Step 1 : Stimuli produced by objects are perceived by sensory devices. The more prominent attributes (such as size, shape, color, texture) produce the strongest stimuli.

Step 2 : A subset of attributes whose value provide cohesive object grouping or clustering, consistent with some goals anociated with object classification are selected. Attributes selected are those which produce high interclass and low interclass grouping. This subset represents a reduction in the attribute space dimensionality and hence simplifies the classification process.

Step 3 : Using the selected attribute values, objects or class characterization models are learned by forming generalized photo type descriptions, classification rules or decision functions. These models are used in subsequence recognition.

Step 4 : Recognition of familiar objects is achieved through application of rules learned in step 3 by comparison and matching of object features with the stored models. Refinements and adjustments can be performed continually thereafter to improve the quality and speed of recognition.

These are two basic methods to object classification :

(1) Decision theoretic method

(2) Syntactic approach

(1) **Decision theoretic** approach is based on the use of decision functions to classify objects.

More formally, this can be started as :

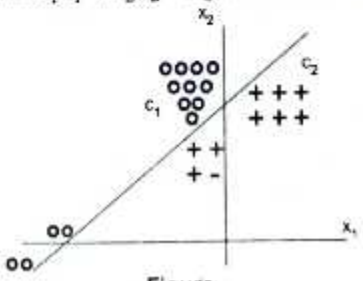
(a) Given a universe of objects $O = \{O_1, O_2, ...O_n\}$ let each 0_i here k observable attributes and relations expressible as a vector $v = (v_1, v_2, ...v_k)$.

(b) Determine :

(a) A subset $m \le k$ of the v_i say $x = (x_1, x_2, ...x_n)$ whose values uniquely characterize O_i and

(b) $c \ge 2$ grouping or classification of O_i which exhibits high interclass and low interclass similarities such that a decision variable $d(x)$ can be found which partition D into C disjoint regions. The regions are used to classify each O_i as belonging to at most one of the C classes.

When these are two classes C_1 and C_2, the value of the objects pattern vector may tend to cluster into two disjoint groups. $d(x) = w_1x_1 + w_2x_2 + w_3$ (Decision function).



Figure

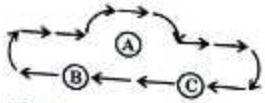
$\leftarrow d(x) = x_1w_1 + x_2w_2 + w_3 = 0$

The decision function approach described above is examples of deterministic recognition otherwise we will have to choose probabilistic models to deal with will non-deterministic recognition.

(2) **Syntactic Classification :** This approach is based on the uniqueness of syntactic "structure" among the object classes. Instead of defining the grammar in term of alphabet, vocabulary is based on shape primitives :

V_T :

Figure



A = a a f a g a a d c i c c i c f
B = e f g h
C = e f g h

Q.5. Explain Nearest Neighbor rule used for classification. **(2017-18)**

OR **Write short note on Linear Discriminant Analysis.** **(2015-16)**

OR **Explain Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA).** **(2015-16)**

OR **Write short notes on :**

(1) LDA.

(2) PCA.

(3) Nearest Neighbor Rule.

Ans. (1) Linear Discriminant Analysis (LDA) : Linear discriminant analysis (LDA) and the Fisher linear discriminant method is used to find the statistics and to learn specific features or separate two or more classes of objects or events a linear combination of machines. The resulting combination can be used as linear classifier, or more commonly, in the future to reduce the peacekeeping category.

LAD is closely related to ANOVA (analysis of variance) and regression analysis, which also tried to express a variable for other functions or a linear combination of measurements. In the other two methods, however, the dependent variable is a numerical quantity, and the LAD is an absolute variable (i.e., class label). Logistic regression and probability of return is more similar to the LAD, because they also show a classification variable. These other methods in the application, preferably it is unreasonable to assume that, since the variable is normally distributed, which is a basic assumption of the LAD method.

LAD is also closely related to principal component analysis (PCA) and factor analysis in a linear combination of these two variables to see if the best interpretation of data. The Department clear attempt to model the class differences between the data. On the other hand, the Permanent Court of Arbitration does not consider any class differences, and factor analysis, based on the basis of functional differences, rather than the same combination. Discriminant analysis is also different from the factor analysis, this is not an interdependent technologies : an independent variable and the difference between the dependent variable (also known as the standard variables) must be made. LAD work, each observed variable measurement is achieved by a continuous number. When combined with an absolute independent variables treatment, equivalent technical Discriminant Correspondence Analysis.

Scatter Matrices :

(a) Within Class :

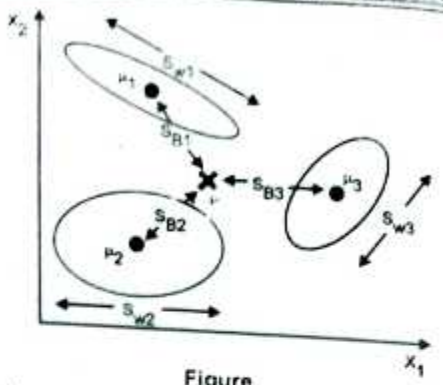
$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

(b) Between Class :

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

(c) Then Maximize Ratio :

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$$



Figure

Solution : Optimal projections are the eigenvectors of the largest eigen values of the generalized eigen value problem.

$$(S_B - \lambda_i S_W)w_i = 0$$

Notes :

(a) S_B is the sum of c matrices of rank one are less, and the mean vectors are constrained by $\sum_i \mu_i = \mu$.

(b) Therefore, S_B will be at most of rank C-1, and LDA produces at most C-1 feature projections.

Limitations :

(a) Over fitting

(b) Information not in the mean of the data

(c) Classes significantly non Gaussian

(2) Principle Component Analysis (PCA) : PCA can be justified in several ways.

The Variance Preservation View : Let's consider a projection onto a line going through the origin. Such a line can be specified by a vector $w \in R^D$. The projection of x is

$$\frac{w^T x}{||w||} \quad \ldots (1)$$

For simplicity, let us consider w with unit length. The variance of the projected dataset is

$$\frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 = w^T S w \quad \ldots (2)$$

Where $S = \frac{1}{n}\sum_i x_i x_i^T$ ... (3)

Is the sample covariance matrix since we assume the dataset is centered. The goal of PCA is to find the w that maximizes the variance, in the hope that it maximally preserves the distinction among points. This leads to the following optimization problem

$$\text{Max} w = w^T S w \quad \ldots (4)$$

$$\text{s.t.} ||w|| = 1 \quad \ldots (5)$$

Let's solve it by forming the Lagrangian

$$w^T S w + \lambda(1 - w^T w) \quad \ldots (6)$$

The gradient w.r.t. w is

$$\nabla = 2Sw - 2\lambda w \quad \ldots (7)$$

Setting to zero, we find that

$$Sw = \lambda w \quad \ldots (8)$$

I.e., the desired direction w is an eigenvector of S! But which one? Recall the projected variance is

$$w^T S w = w^T \lambda w = \lambda, \quad \ldots (9)$$

Dimensionality Reduction : We see that we want λ to be the largest eigen value of S and w the corresponding eigenvector. In other words, let $\lambda_1, \ldots \lambda_n$ be the eigen values of S in non-increasing order, and $o_1, \ldots, o_n$ be the corresponding eigenvectors. Then o_1 is the maximum variance preserving direction, and the resulting variance is simply λ_1. This is PCA with d=1 : a D-dimensional point x is projected to a scalar $o_1^T x$. Note that when top eigen value has multiplicity larger than one, i.e. $\lambda_1 = \lambda_2$, then PCA is not unique; any unit vector in span(o_1, o_2) can be the PCA direction. If we want d>1, it can be shown that we want to project x onto the first d eigenvectors

$$x \to (o_1^T x, \ldots, o_d^T x)^T \quad \ldots (10)$$

Recall that one can view $o_1 \ldots o_D$ as the D major-to-minor axes of an ellipsoid represented by the sample covariance matrix (NB this does not assume that the underlying distribution is Gaussian). Clearly

, if $d=D$ then e_1 ... e_D is a basis for R^D, and the PCA projection amounts to a rotation of the coordinate system (align them with the eigenvectors) without any loss of information.

(3) *k*-Nearest Neighbors(kNN) Classifier :

(a) kNN is a very intuitive method that classifies unlabeled examples based on their similarity to examples in the training set

Given unlabeled example x_u, find the k closest examples in the training set and assign x_u to the most represented class among the k-subset

(b) The kNN classifier only requires :

(i) An integer k

(ii) A set of labeled examples(training data)

(iii) A "closeness" measure, generally Euclidean

Advantages :

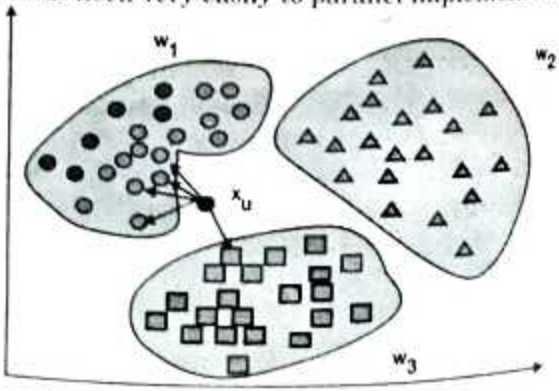
(1) Analytically tractable

(2) Simple implementation

(3) Nearly optimal in the large sample limit ($n \to \infty$) $P[\text{error}]_{Bayes} < P[\text{error}]_{1NN} < 2P[\text{error}]_{Bayes}$

(4) Uses local information, which can yield highly adaptive behavior

(5) Lends itself very easily to parallel implementations



Figure

Disadvantages :

(1) [illegible]

Q.6. To Which category of clustering schemes does the k-means algorithm belong? What is its major advantage? Which are the factors that influence the computation duration of this algorithm? (2015-16)

***OR* Explain the *k*-means clustering with example.**

Ans. *k*-Means Clustering : The k-means algorithm is a simple procedure that attempts to group a collection of unlabeled examples $X = \{x_1 \ldots x_n\}$ into one of C clusters

(1) k-means seeks to find compact clusters, measured as

$$J_{MSE} = \sum_{c=1}^{C} \sum_{x \in w_c} \| x - \mu_c \|^2 ; \mu_c = \frac{1}{n_c} \sum_{x \in w_c} x$$

(2) It can be shown that k-means is a special case of the GMM-EM algorithm.

Procedure :

(1) Define the number of clusters.

(2) Initialize clusters by

(a) An arbitrary assignment of examples to clusters

or

(b) An arbitrary set of cluster centers (i.e., use some examples as centers).

(3) Compute the sample mean of each cluster.

(4) Reassign each example to the cluster with the nearest mean.

(5) If the classification of all samples has not changed, stop, else go to step 3.

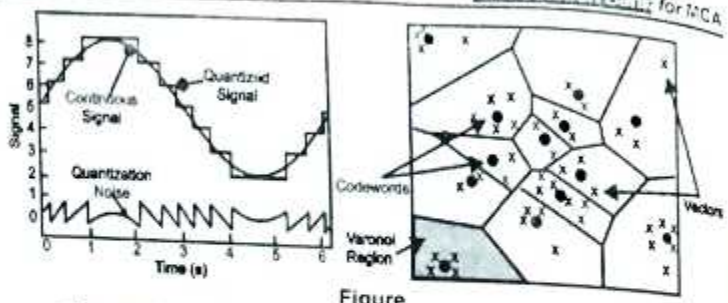
***k*-means is Widely used in DSP for Vector Quantization :**

(1) Unidimensional signal values are usually quantized into a number of levels (typically a power of 2 so the signal can be transmitted in binary format)

(2) The same idea can be extended for multiple channels.

However, rather than quantizing each separate channel, we can obtain a more efficient signal coding by quantizing the overall multidimensional vector into a small number of multidimensional prototypes (cluster centers)

(3) Each cluster center is called a codeword and the collection of codeword is called a codebook



Figure

Example : An agent has observed the following (X,Y) pairs :

(0.7,5.1), (1.5,6), (2.1,4.7), (2.4,5.5), (3,4.4), (3.5,5), (4.5,1.5), (5.2,0.7), (5.3,1.8), (6.2,1.7), (6.7,2.5), (8.5,9.2), (9.1,9.7), (9.5,8.5).

These data points are plotted in part (a) of Figure. Suppose the agent wants to cluster the data points into two classes.

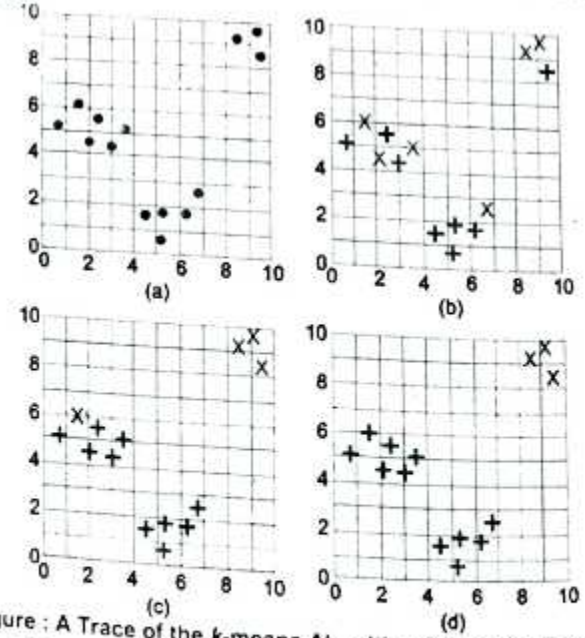


Figure : A Trace of the k-means Algorithm for k=2 for Example

In part (*b*), the points are randomly assigned into the classes; one class is depicted as + and the other as ×. The mean of the points marked with + is (4.6, 3.65). The mean of the points marked with × is (5.2, 6.15). In part (*c*) the

points are reassigned according to the closer of the two means. After this reassignment, the mean of the points marked with + is then (3.96, 3.27). The mean of the points marked with × is (7.15, 8.34).

In part (*d*), the points are reassigned to the closest mean. This assignment is stable in that no further reassignment will change the assignment of the examples. A different initial assignment to the points can give different clustering. One clustering that arises in this data set is for the lower points (those with a Y-value less than 3) to be in one class, and for the other points to be in another class.

Running the algorithm with three classes would separate the data into the top-right cluster, the left-center cluster, and the lower cluster. One problem with the k-means algorithm is the relative scale of the dimensions. For example, if one feature is height, another feature is age, and another is a binary feature, you must scale the different domains so that they can be compared. How they are scaled relative to each other affects the classification. To find an appropriate number of classes, an agent can search over the number of classes.

Note that k+1 classes can always result in a lower error than k classes as long as more than k different values are involved. A natural number of classes would be a value k when there is a large reduction in error going from k-1 classes to k classes, but in which there is only gradual reduction in error for larger values. Note that the optimal division into three classes, for example, may be quite different from the optimal division into two classes.

Q.7. Write short note on Bayesian Classifier. (2015-16)

OR Explain how classification is done by using Bayes classifier. (2014-15)

Ans. Bayes Classifier : Bayes classifier is popular in pattern recognition because it is an optimal classifier. It is possible to show that the resultant classification minimizes the average probability of error. Bayes classifier is based on the assumption that information about classes in the form of prior probabilities and distributions of patterns in the class are known. It employs the posterior probabilities to assign the class label to a test pattern: a pattern is

assigned the label of the class that has the maximum posterior probability. The classifier employs Bayes theorem to convert the prior probability into posterior probability based on the pattern to be classified, using the likelihood values.

Bayes Theorem : Let X be the pattern whose class label is unknown. Let H be some hypothesis which indicates the class to which X belongs. For example, H_i is the hypothesis that a pattern belongs to class C. Let us assume that the prior probability of H_i, given by $P(H_i)$ is known. In order to classify X, we need to determine $P(H_i|X)$ which is the probability that the hypothesis H_i holds, given the observed pattern X.

$P(H_i|X)$ is the posterior probability of H_i conditioned on X. In contrast, $P(H_i)$ is the prior probability, or apriori probability, of H_i. It is the probability of the hypothesis regardless of the value of X. The posterior probability, $P(H_i|X)$ is based on X and other information (such as background knowledge), whereas the prior probability, $P(H_i)$ is obtained before observing X. Similarly, $P(X|H_i)$ is the probability of X conditioned on H_i. Note that $P(X)$ is defined as a weighted combination of $P(X|H_i)$ and is

$$P(X) = \sum_i P(X|H_i)P(H_i) \quad \ldots(1)$$

Bayes theorem is useful in that it provides a way of calculating the posterior probability, $P(H_i|X)$, from $P(H_i)$, $P(X)$ and $P(X|H_i)$. It states

$$P(H_i|X) = \frac{P(X|H_i)P(H_i)}{P(X)} \quad \ldots(2)$$

Q.8. To which category of clustering schemes does the k-means algorithm belong? What is its major advantage? Which are the factors that influence the computation duration of this algorithm? (2014-15)

Ans. k-means clustering is a method of vector quantization, originally from signal processing that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

Standard Algorithm : The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the *k*-means algorithm; it is also referred to as Lloyd's algorithm, particularly in the computer science community. Given an initial set of k means $m_1(1),\ldots,m_k(1)$ (see below), the algorithm proceeds by alternating between two steps :

Assignment Step : Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean. (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means).

Update Step : Calculate the new means to be the centroids of the observations in the new clusters. Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares (WCSS) objective. The algorithm has converged when the assignments no longer change. Since both steps optimize the WCSS objective, and there only exists a finite number of such partitioning, the algorithm must converge to a (local) optimum. There is no guarantee that the global optimum is found using this algorithm.

The algorithm is often presented as assigning objects to the nearest cluster by distance. The standard algorithm aims at minimizing the WCSS objective, and thus assigns by "least sum of squares", which is exactly equivalent to assigning by the smallest Euclidean distance. Using a different distance function other than (squared) Euclidean distance may stop the algorithm from converging. Various modifications of k-means such as spherical k-means and k-medoids have been proposed to allow using other distance measures.

Advantages : k-means clustering, in particular when using heuristics such as Lloyd's algorithm, is rather easy to implement and apply even on large data sets. As such, it has been successfully used in various topics, including market segmentation, computer vision, geo-statistics, astronomy and agriculture. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

(1) Vector quantization
(2) Cluster analysis
(3) Feature learning

Factors that Influence the Computation :

(1) Initialization
(2) Complexity
(3) The Triangle Inequality
(4) Variations

❑❑

MCA

(SEM. IV) EXAMINATION, 2015–16

NMCA – 413 : ARTIFICIAL INTELLIGENCE

Time : 3 Hours *Total Marks : 100*

Note : Attempt questions from ***all*** Sections as per directions.

SECTION – A

1. **Attempt *all* the parts. All parts carry equal marks. Write answer of each part in short.** **(10 × 2 = 20)**
   (a) What is machine learning? How many types of learning are there?
   (b) List the characteristics of intelligence.
   (c) What is "overfitting"? How do we overcome overfitting?
   (d) Explain the statistical nature of the learning process.
   (e) Represent the following sentence in the Predicate form "All the children like sweets".
   (f) What do you understand by Natural Language Processing?
   (g) What is artificial intelligence? How it is different than general intelligence?
   (h) Describe the role of computer vision in artificial intelligence.
   (i) Write four properties a good system should possess for the knowledge representation in a particular domain.
   (j) Explain Maximum likelihood hypothesis and Maximum a posteriori.

SECTION – B

2. **Attempt *any five* questions from this section.** **(5 × 10 = 50)**
   (a) Explain how Bayesian statistics provide reasoning under various kinds of uncertainty.
   (b) Describe A* search technique. Prove that A* is complete and optimal.
   (c) Give an example of a problem for which breadth first search would work better than depth first search. Write the difference between these two approaches.
   (d) What is Bayesian reasoning? What does a Bayesian network represent? Explain.
   (e) Describe alpha-beta pruning and give the other modifications to the min-max procedure to improve its performance.
   (f) Explain the expectation and maximization (EM) algorithm for finding the maximum likelihood with hidden variables.
   (g) Design principles of pattern recognition system. Explain Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA).
   (h) What do you understand by pattern recognition? Differentiate between structural description and symbolic description.

SECTION – C

Note : Attempt *any two* questions from this section. **(2 × 15 = 30)**

3. To Which category of clustering schemes does the k-means algorithm belong? What is its major advantage? Which are the factors that influence the computation duration of this algorithm?

4. Convert the following sentence into predicate logic and then prove "Marcus is Dead" using resolution :
   (a) Marcus was a man.
   (b) Marcus was a Pompeian.
   (c) Marcus was born in 40 AD.
   (d) All men are mortal.
   (e) All Pompeian's dead when the volcano erupted in 1979.
   (f) No mortal lives more than 150 years.
   (g) It is now 1991.
   (h) Alive means not dead.

5. Write short note on the following :
   (a) Support vector machine (SVM)
   (b) Linear Discriminant Analysis
   (c) Bayesian Classifier

□□

MCA

(SEM – IV) EXAMINATION, 2016 – 17

NMCA – 413 : ARTIFICIAL INTELLIGENCE

Time : 3 Hours *Max. Marks : 100*

Note : Be precise in your answer. In case of numerical problem assume data wherever not provided.

SECTION – A

1. **Explain the following :** **(10 × 2 = 20)**
   (a) Name the elements of an agent.
   (b) Summarize the factors that make up rationality.
   (c) What do you infer from hill – climbing search algorithm?
   (d) Compare propositional logic and predicate logic.
   (e) Justify the usage of universal and existential quantifier with an example.
   (f) Give the heuristic function for shortest path problem.
   (g) Which algorithm is more similar to backward chaining algorithm? Write its algorithm.
   (h) What do you mean by hybrid Bayesian network?
   (i) Which value is assigned to alpha and beta in the alpha – beta pruning?
   (j) List few decision tree algorithms.

SECTION – B

2. **Attempt *any five* of the following questions :** **(5 × 10 = 50)**
   (a) Discuss the structure of an intelligent agent. Give an example.
   (b) Compare the uniformed and informed search strategies with respect to all factors.
   (c) Solve the 8 – puzzle problem using Hill climbing. Write down the heuristic function.

Initial State		
1	2	3
8	6	5
7	4	

Goal State		
1	2	3
4	5	6
7	8	

   (d) The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono an enemy of America, has some missiles and all of its missiles were sold to it by Colonel West, who is American. Prove that Col. West is criminal by forward and backward chaining.
   (e) Assume two players, min and max, play min (as describe above). Min plays first. If a terminal state in the search tree developed above is a win for min, a utility function of zero is assigned to that state. A utility function of 1 is assigned to a state if max wins the game. Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree.
   (f) Consider the following data set :

Feature 1	Feature 2	Feature 3	Class
0	0	0	0
1	0	1	1
1	0	0	0
1	1	1	1
0	1	1	1
0	1	1	0

Assume the test pattern as feature 1 as 0, feature 2 as 0 and feature 3 as 1, classify the pattern using NNC and Bayes classifier.

(g) Write down the evaluation procedure of HMM.

(h) How a k – means clustering algorithm works? Give an example.

SECTION – C

Attempt *any two* of the following questions : **(2 × 15 = 30)**

3. (a) Represent the following sentences in first – order logic, using a consistent vocabulary (which you must define)
   (i) Not all students take both History and biology.
   (ii) Only one student failed history.
   (iii) Only one student failed both History and Biology.
   (iv) The best score in History was better than the best score in Biology.
   (v) Every person who dislikes all vegetarians is smart.
   (vi) No person likes a smart vegetarian.
   (vii) There is a woman who likes all men who are not vegetarians.
   (viii) There is a barber who shaves all men in town who do not shave themselves.
   (ix) No person likes a professor unless the professor is smart.
   (x) Politicians can fool some of the people all of the time and they can fool all of the people some of the time, but they can't fool all of the people all of the time.

   (b) (i) Give a predicate calculus sentence such that every world in which it is true contains exactly one object.
   (ii) Represent the sentence "All Germans speak the same languages" in predicate calculus. Use Speaks (x, l), meaning that person x speaks language.

4. Write in detail about the decision trees and decision lists with an example.

5. How parameter estimation is done? Explain with an example.

MCA

(SEM. IV) THEORY EXAMINATION, 2017-18
RCA – 403 : ARTIFICIAL INTELLIGENCE

Time : 3 Hours *Total Marks : 70*

Note : Attempt ***all*** Sections. If require any missing data, then choose suitably.

SECTION – A

1. **Attempt *all* questions in brief.** **(2 × 7 = 14)**
   (a) What is meant by the term Artificial Intelligence? How it is different from natural intelligence?
   (b) Discuss Branch-and-bound search algorithm.
   (c) Differentiate between local search and global search.
   (d) Transform the following formula to Prenex Normal form-
   $$\forall x : \forall y : (\exists z : P(x,z) \cap P(y,z)) \rightarrow \exists u : Q(x,y,u)$$
   (e) Define forward changing and backward chaining with example.
   (f) Explain in brief the concept of reinforcement learning.
   (g) Write a short note on Support Vector Machine.

SECTION – B

2. **Attempt *any three* of the following :** **(7 × 3 = 21)**
   (a) What is an intelligent agent? Discuss any two types of intelligent agents.
   (b) Explain Steepest-ascent hill climbing algorithm. What are the problems with hill climbing algorithm?
   (c) Describe Hidden Markov model with suitable example. Also discuss its role in probabilistic reasoning.
   (d) Discuss Maximum-likelihood parameter learning for complete data with discrete models.
   (e) What do you mean by classification? Discuss the process of classification with the help of a diagram.

SECTION – C

3. **Attempt *any one* part of the following :** **(7 × 1 = 7)**
   (a) Discuss the historical development of artificial intelligence.
   (b) For each of the following agents, develop a PEAS description of the task environment-
   (i) Mathematician's theorem proving assistant
   (ii) Satellite image analysis system
   (iii) Internet book shopping agent
   (iv) Medical diagnosis system

4. **Attempt *any one* part of the following :** **(7 × 1 = 7)**
   (a) Discuss Simulated Annealing search algorithm with its advantages and disadvantages.
   (b) What are the steps to define a problem? Explain also discuss various components of a problem.

5. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Discuss algorithm of conversion to clause form. Convert the following to clause form using algorithm-

$$\forall x[Brick(x) \rightarrow (\exists y\{On(x,y) \frown \neg Pyramid(y)\} \frown \neg \exists y\{On(x,y) \frown On(y,x)\} \frown \forall y\{\neg Brick(y) \rightarrow \neg Equal(x,y)\})]$$

(b) Explain the concept of Alpha-beta pruning. Write Alpha-beta search algorithm.

6. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Discuss various application domains of machine learning.

(b) Describe major steps involved in a learning process. Also discuss how learning systems are classified.

7. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) What is pattern recognition. Explain various steps involved in the designing of a pattern recognition system with the help of a diagram.

(b) Explain Nearest Neighbor rule used for classification.

❑❑

MCA

(SEM. IV) THEORY EXAMINATION, 2018-19

RCA – 403 : ARTIFICIAL INTELLIGENCE

Time : 3 Hours *Total Marks : 70*

Note : Attempt ***all*** Sections. If require any missing data; then choose suitably.

SECTION – A

1. **Attempt *all* questions in brief.** (2 × 7 = 14)

(a) Name the elements of an agent.

(b) Summarize the factors that make up rationality.

(c) What do you infer from hill-climbing search algorithm?

(d) Compare propositional logic & predicate logic.

(e) Justify the usage of universal and existential quantifier with an example.

(f) Give the heuristic function for shortest path problem.

(g) Which algorithm is more similar to backward chaining algorithm? Write its algorithm.

SECTION – B

2. **Attempt *any three* of the following :** (7 × 3 = 21)

(a) You have three jugs measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You need to measure out exactly one gallon.

(b) Describe the planning method based on hierarchical task networks with an example.

(c) Discuss the different design issues to be solved to use hidden markov model for real world application.

(d) Assume two players, min and max, play (as described above). Min plays first. If a terminal state in the search tree developed above is a win for min, a utility function of zero is assigned to that state. A utility function of 1 is assigned to a state if max wins the game. Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree.

(e) Give the completeness proof of resolution.

SECTION – C

3. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Implement the Search Algorithms described in this lecture in LISP and/or C. Comment on how suited each language would be for each type of search.

(b) How suited would PROLOG be in implementing the search algorithms? Comment on how this might be done and what difficulties might exist.

4. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Trace the constraint satisfaction procedure to solve the following cryptarithmetic problem :

```
  CROSS
+ROADS
-------
 DANGER
```

(b) Discuss how constraint satisfaction might work it implemented its search strategy via;
(i) depth first search
(ii) breadth first search
(iii) best first search

5. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Represent the following in partitioned semantic networks :
(i) Every player kicked a ball.
(ii) All players like the referee.
(iii) Andrew believes that there is a fish with lungs.

(b) Pick a problem area and represent the knowledge in frame based system.

6. **Attempt *any one* part of the following :** (7 × 1 = 7)

(a) Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?

(b) Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn.

7. **Attempt *any one* parts of the following :** (7 × 1 = 7)

(a) Discuss back propagation algorithm for learning in multilayer neural network.

(b) Explain the concept of forward and backward state space search in detail.