SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS

Syllabus

SMRA3003	**ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS**	**L**	**T**	**P**	**Credits**	**Total Marks**
		3	**0**	**0**	**3**	**100**

COURSE OBJECTIVES

1. To understand Basic Concepts of Artificial Intelligence and Expert Systems.
2.To provide knowledge on Various Techniques and Tools involved in Artificial Intelligence.

UNIT 1 INTRODUCTION

Introduction: History, Definition of AI, Emulation of human cognitive process, knowledge search trade off, stored knowledge, semantic nets. An abstract view of modelling, elementary knowledge. Computational logic, analysis of compound statements using simple logic connectives, predicate logic, knowledge organization and manipulation, knowledge acquisition.

UNIT 2 PROGRAMMING AND LOGICS IN ARTIFICIAL INTELLIGENCE

LISP and other programming languages- introduction to LISP, syntax and numerical function, LISP and PROLOG distinction, input output and local variables, Interaction and recursion, property list and arrays alternative languages, formalized symbolic logics- properties of WFRS, non-deductive inference methods. Inconsistencies and uncertainties- Truth maintenance systems, default reasoning and closed world assumption, Model and temporary logic.

UNIT 3 SEARCH METHODS AND KNOWLEDGE REPRESENTATION

Fuzzy logic - concepts, Introduction to Fuzzy logic with examples, probabilistic reasoning, Bayesian probabilistic inference, Dempstor Shafer theory, possible world representation, Ad-Hoc methods. Structure knowledge: Graph, frames and related structures, Object oriented representation- object classes, message and methods, simulation examples using OOPS programs, OOP languages. Search and control strategies - Concepts, search problems, uniformed or Blined search, searching AND – OR graphs.

UNIT 4 KNOWLEDGE ORGANISATION AND COMMUNICATION IN EXPERT SYSTEMS

Matching techniques- Need for matching, matching problem, partial matching, Fuzzy matching, RETE matching algorithm. Knowledge organization- Indexing and retrieval techniques, integration of knowledge in memory organization systems, Perception, communication and Expert systems. Overview of Linguistics, Basic passim techniques, semantic analysis and representation structures, natural language generation and system.

UNIT 5 PATTERN RECOGNITION AND LEARNING TECHNIQUES

Pattern recognition system- understanding speech recognition, Image transformation, low level processing, medium and high level processing, vision system architecture, Rule based system architecture, knowledge acquisition and validation, knowledge system building tools, use of AI and ES in manufacturing and design, types of learning- general learning model, performance measures, learning automate genetic algorithm, learning by induction - LEX,ID3,INDUCE systems.

Max.45 Hours

Course Outcomes:

On completion of the course, student will be able to

CO1 - Understand the Basics about Artificial Intelligence and Expert Systems.
CO2 - Understand the Programming Logics in Artificial Intelligence.
CO3 - Understand Various search methods in Artificial Intelligence.
CO4 - Understand the Knowledge about the Expert Systems.
CO5 - Understand The Image processing and analysis.
CO6 - Understand the latest developments in Knowledge systems and Tools.

TEXT / REFERENCE BOOKS

1. Russel (Stuart), ‘Artificial Intelligence- Modern approach, Pearson Education series in AI’, 3rd Edition, 2009.
2. Dan W Patterson, ‘Introduction to Artificial intelligence and Expert systems’, Prentice Hall of India Pvt. Ltd,2001
3. Eugene Charniak, Drew Mc Dermot, ‘Introduction to Artificial intelligence’, Addison Wesley Longman Inc.,2009
4. George. F, William. A. Stubblefield, ‘Artificial intelligence and the design of expert systems’, The Benjamin Cummins Publishing Co., Inc 2nd Edition, 1992.
5. Robert J Schalkoff, ‘Artificial intelligence An Engineering Approach’, McGraw Hill International Edition, 1990

End Semester Examination Question Paper Pattern

Max. Marks: 100 Exam Duration: 3 Hrs
PART A: 10 Questions of 2 marks each – No Choice 20 Marks
PART B: 2 Questions from each units of internal choice, each carrying 16 marks 80 Marks

B.E. / B.Tech. - Regular REGULATIONS 2019

UNIT – I – ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS – SMRA3003

1 Introduction to Artificial Intelligence

1.1 Introduction

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

1.2 History of Artificial Intelligence

Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths. Following figure 1.1 are some milestones in the history of AI which defines the journey from the AI generation to till date development.

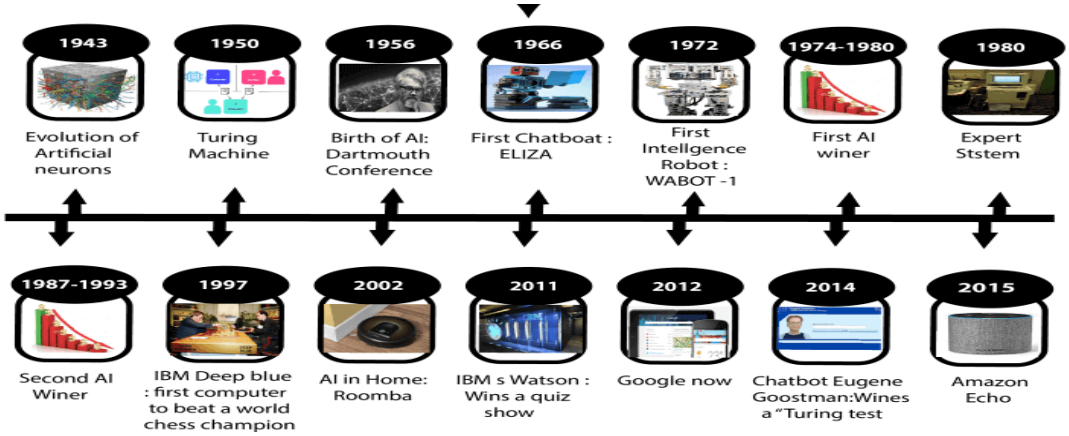


Figure 1.1 History of Artificial intelligence

1.3. Definition

The study of how to make computers do things at which at the moment, people are better. "Artificial Intelligence is the ability of a computer to act like a human being".

- Systems that think like humans
- Systems that act like humans
- Systems that think rationally
- Systems that act rationally.
- Intelligence - Ability to apply knowledge in order to perform better in an environment.
- Artificial Intelligence - Study and construction of agent programs that perform well in a given environment, for a given agent architecture.
- Agent - An entity that takes action in response to precepts from an environment.
- Rationality - property of a system which does the "right thing" given what it knows.
- Logical Reasoning - A process of deriving new sentences from old, such that the new sentences are necessarily true if the old ones are true.

1.4 Acting Humanly: Turning Test Approach

Figure 1.2 Illustrates the Turing Test, proposed by Alan Turing (1950), was designed to provide satisfactory operational definition of intelligence. A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written response.

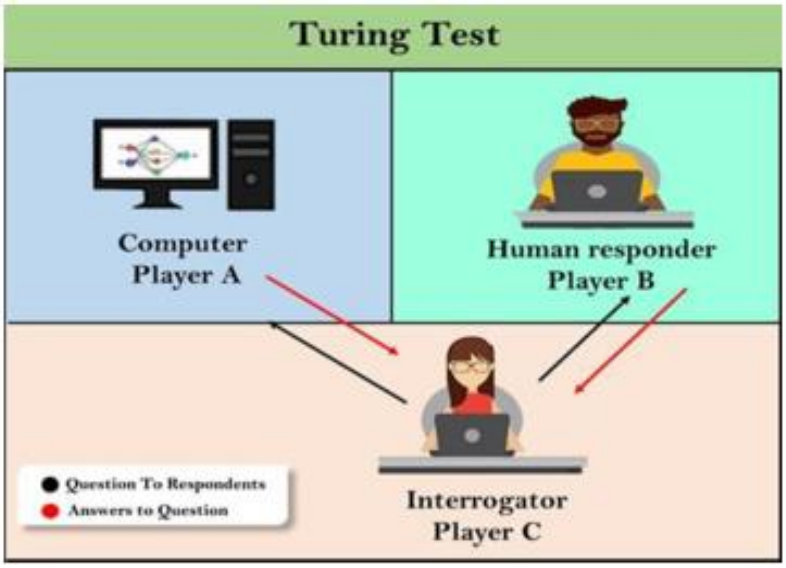


Figure 1.2 Turning Test Approach

- Natural language processing to enable it to communicate successfully in English;
- Knowledge representation to store what it knows or hears;
- Automated reasoning to use the stored information to answer questions and to draw new conclusions.
- Machine learning to adapt to new circumstances and to detect and extrapolate patterns.

Total Turing Test includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch." To pass the total Turing Test, the computer will need

Analyses how a given program thinks like a human, we must have some way of determining how humans think. The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind.

Although cognitive science is a fascinating field in itself, we are not going to be discussing it all that much in this book. We will occasionally comment on similarities or differences between AI techniques and human cognition. Real cognitive science, however, is necessarily based on experimental investigation of actual humans or animals, and we assume that the reader only has access to a computer for experimentation. We will simply note that

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

Acting rationally means acting so as to achieve one's goals, given one's beliefs. An agent is just something that perceives and acts.
The right thing: that which is expected to maximize goal achievement, given the available information Does not necessary involve thinking. For Example - blinking reflex- but should be in the service of rational action.

1.5 Future of Artificial Intelligence

- Transportation: Although it could take a decade or more to perfect them, autonomous cars will one day ferry us from place to place.
- Manufacturing: AI powered robots work alongside humans to perform a limited range of tasks like assembly and stacking, and predictive analysis sensors keep equipment running smoothly.
- Healthcare: In the comparatively AI-nascent field of healthcare, diseases are more quickly and accurately diagnosed, drug discovery is sped up and streamlined, virtual nursing assistants monitor patients and big data analysis helps to create a more personalized patient experience.
- Education: Textbooks are digitized with the help of AI, early-stage virtual tutors assist human instructors and facial analysis gauges the emotions of students to help determine who’s struggling or bored and better tailor the experience to their individual needs.
- Media: Journalism is harnessing AI, too, and will continue to benefit from it. Bloomberg uses Cobourg technology to help make quick sense of complex financial reports. The Associated Press employs the natural language abilities of Automated

Insights to produce 3,700 earnings reports stories per year — nearly four times more than in the recent past

- Customer Service: Last but hardly least; Google is working on an AI assistant that can place human-like calls to make appointments at, say, and your neighborhood hair salon. In addition to words, the system understands context and nuance.

1.6 Characteristics of Intelligent Agents

The agent receives some form of sensory input from its environment, and it performs some action that changes its environment in some way. Examples of environments: the physical world and the Internet.

- Autonomy

  The agent can act without direct intervention by humans or other agents and that it has control over its own actions and internal state.

- Adaptively

  The agent is capable of reacting flexibly to changes in its environment; taking goal-directed initiative (i.e., is pro-active), when appropriate; and Learning from its own experience, its environment, and interactions with others.

- Sociability The agent is capable of interacting in a peer-to-peer manner with other agents or humans

1.7 Agents and its type

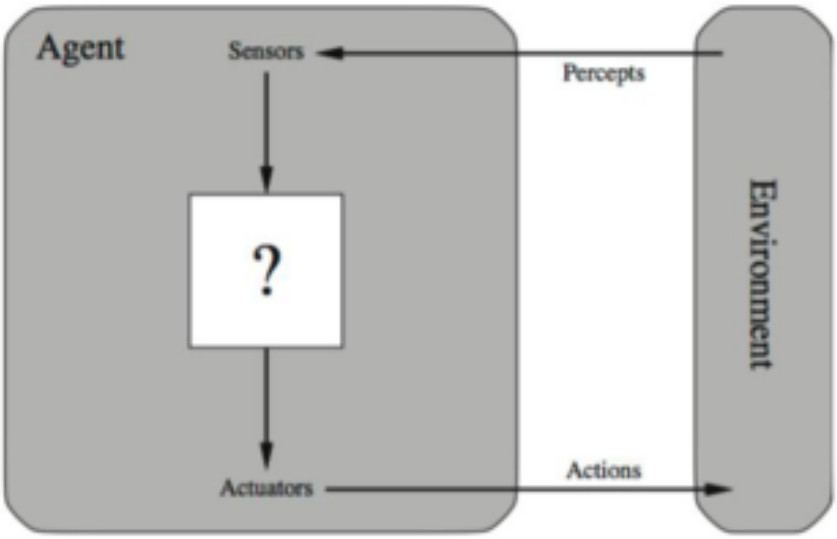


Figure 1.3 Agent types

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators Ref Figure (1.3)

- Human Sensors:
- Eyes, ears, and other organs for sensors.
- Human Actuators:
- Hands, legs, mouth, and other body parts.
- Robotic Sensors:
- Mic, cameras and infrared range finders for sensors
- Robotic Actuators:
- Motors, Display, speakers etc An agent can be:

Human-Agent: A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.

Robotic Agent: A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.

Software Agent: Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

Hence the world around us is full of agents such as thermostat, cell phone, camera, and even we are also agents. Before moving forward, we should first know about sensors, effectors, and actuators.

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

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

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

1.8 Properties of Environment

An environment is everything in the world which surrounds the agent, but it is not a part of an agent itself. An environment can be described as a situation in which an agent is present.

The environment is where agent lives, operate and provide the agent with something to sense and act upon it.

Fully observable vs Partially Observable:

If an agent sensor can sense or access the complete state of an environment at each point of time then it is a fully observable environment, else it is partially observable. A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.

An agent with no sensors in all environments then such an environment is called as unobservable.

Example: chess – the board is fully observable, as are opponent's moves. Driving – what is around the next bend is not observable and hence partially observable.

- Deterministic vs Stochastic

  If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.A stochastic environment is random in nature and cannot be determined completely by an agent. In a deterministic, fully observable environment, agent does not need to worry about uncertainty.
- Episodic vs Sequential

  In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action. However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

Single-agent vs Multi-agent

If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment. However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.

- Static vs Dynamic

  If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment. Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action. However for dynamic environment, agents need to keep looking at the world at each action. Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

- Discrete vs Continuous

  If in an environment there are a finite number of precepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.

  A chess game comes under discrete environment as there is a finite number of moves that can be performed. A self-driving car is an example of a continuous environment.

- Known vs Unknown

  Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.In a known environment, the results for all actions are known to the agent.

  While in unknown environment, agent needs to learn how it works in order to perform an action.It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.

- Accessible vs. Inaccessible

  If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.

  An empty room whose state can be defined by its temperature is an example of an accessible environment. Information about an event on earth is an example of Inaccessible environment. If in an environment there are a finite number of precepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment. A chess game comes under discrete environment as there is a finite number of moves that can be performed. A self-driving car is an example of a continuous environment.

1.9 PEAS: Performance Measure, Environment, Actuators, Sensors

Performance: The output which we get from the agent. All the necessary results that an agent gives after processing comes under its performance.

Environment: All the surrounding things and conditions of an agent fall in this section. It basically consists of all the things under which the agents work.

Actuators: The devices, hardware or software through which the agent performs any actions or processes any information to produce a result are the actuators of the agent.

Sensors: The devices through which the agent observes and perceives its environment are the sensors of the agent.

1.9.1 Characteristic of Rational Agent

The agent's prior knowledge of the environment.
The performance measure that defines the criterion of success.
The actions that the agent can perform.
The agent's percept sequence to date.

1.9.2 Problem Solving Approach to Typical Ai Problems

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

Some of the most popularly used problem solving with the help of artificial intelligence are:

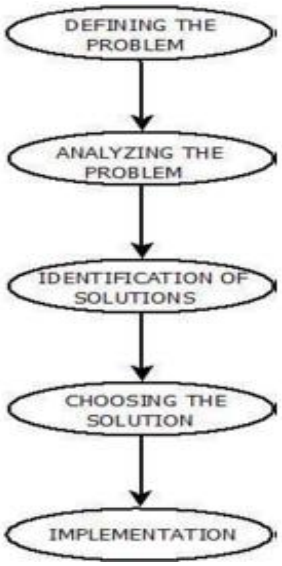
- Chess.
- Travelling Salesman Problem.
- Tower of Hanoi Problem.
- Water-Jug Problem.
- N-Queen Problem.

Problem Searching

- Searching is the most commonly used technique of problem solving in artificial intelligence.
- The searching algorithm helps us to search for solution of particular problem. In general, searching refers to as finding information one needs. 16

Problem: Problems are the issues which come across any system. A solution is needed to solve that particular problem.

Steps: Solve Problem Using Artificial Intelligence Flowchart 1.4



Flowchart 1.4

UNIT – II – ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS – SMRA3003

2 PROGRAMMING AND LOGICS IN ARTICIAL INTELLIGENCE

2.1 Lisp Programming

- LISP is the second oldest programming language in the world (1958), only one year younger than Fortran (1957).
- The term Artificial Intelligence was made up by John McCarthy who invented LISP.
- LISP was founded on the theory of Recursive Functions (a function appears in its own definition).
- Recursive Functions can be written as self-modifying functions, and this is very suitable for AI programs where "self-learning" is an important part of the program.

2.1.1 AI Programming languages

R Programming

- R is a programming language for Graphics and Statistical computing. R is supported by the R Foundation for Statistical Computing. R comes with a wide set of statistical and graphical techniques for:
- Linear Modeling
- Nonlinear Modeling
- Statistical Tests
- Time-series Analysis
- Classification
- Clustering

2.1.2 Python Programming

- Python is a general-purpose coding language. It can be used for all types of programming and software development.
- Python is typically used for server development, like building web apps for web servers.
- Python is also typically used in Data Science.
- An advantage for using Python is that it comes with some very suitable libraries:
- NumPy (Library for working with Arrays)
- SciPy (Library for Statistical Science)
- Matplotlib (Graph Plotting Library)
- NLTK (Natural Language Toolkit)
- TensorFlow (Machine Learning**)**

2.1.3 C++ Programming

- **C++** holds the title: "The worlds fastest programming language".
- Because of the speed, C++ is a preferred language when programming Computer Games.
- It provides faster execution and has less response time which is applied in search engines and development of computer games.

- Google uses C++ in AI programs for SEO (Search Engine Optimization).
- SHARK is a super-fast library with support for supervised learning algorithms, linear regression, neural networks, and clustering.
- MLPACK is a super-fast machine learning library written for C++.

2.1.4 Java Programming

- Java is another general-purpose coding language that can be used for all types of software development.
- For AI, Java is mostly used to create machine learning solutions, search algorithms, and neural networks

2.1.5 SQL Programming

- SQL (Structured Query Language) is the most popular language for managing data.
- Knowledge of SQL databases, tables and queries helps data scientists when dealing with data.
- SQL is very convenient for storing, manipulating, and retrieving data in databases.

2.1.6 Features of Common LISP

- It is machine-independent
- It uses iterative design methodology
- It has easy extensibility
- It allows to update the programs dynamically
- It provides high level debugging.

2.1.7 Applications Developed in LISP

- The following applications are developed in LISP: Large successful applications built in LISP.
- Emacs: It is a cross platform editor with the features of extensibility, customizability, self-document ability, and real-time display.
- G2
- AutoCAD
- Igor Engraver
- Yahoo Store

2.2 Problem Solving By Search

An important aspect of intelligence is goal-based problem solving.The solution of many problems can be described by finding a sequence of actions that lead to a desirable goal. Each action changes the state and the aim is to find the sequence of actions and states that lead from the initial (start) state to a final (goal) state. A well-defined problem can be described by:

What is Search?

Search is the systematic examination of states to find path from the start/root state to the goal state. The set of possible states, together with operators defining their connectivity constitute the search space. The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test.

Problem-solving agents

A Problem solving agent is a goal-based agent. It decides what to do by finding sequence of actions that lead to desirable states. The agent can adopt a goal and aim at satisfying it.

To illustrate the agent's behavior, let us take an example where our agent is in the city of Arad, which is in Romania. The agent has to adopt a goal of getting to Bucharest.

Goal formulation, based on the current situation and the agent's performance measure, is the first step in problem solving. The agent's task is to find out which sequence of actions will get to a goal state.

Problem formulation is the process of deciding what actions and states to consider given a goal.

Example: Route finding problem referring to figure

On holiday in Romania: currently in Arad. Flight leaves tomorrow from Bucharest Formulate goal: be in Bucharest

Formulate problem: states: various cities actions: drive between cities

Find solution: sequence of cities, e.g., Arad, Sibiu, Fagaras, and Bucharest.

2.3 Example Problems

The problem solving approach has been applied to a vast array of task environments. Some best known problems are summarized below. They are distinguished as toy or real-world problems. A toy problem is intended to illustrate various problem solving methods. It can be easily used by different researchers to compare the performance of algorithms. A real world problem is one whose solutions people actually care about.

Toy Problems

Vacuum World Example

States: The agent is in one of two locations, each of which might or might not contain dirt. Thus there are 2 x 22 = 8 possible world states.

Initial state: Any state can be designated as initial state. Successor function: This generates the legal states that results from trying the three actions (left, right, and suck). The complete state space is shown in figure

Goal Test: This tests whether all the squares are clean.

Path test: Each step costs one, so that the path cost is the number of steps in the path.

2.4 LISP Introduction

John McCarthy invented LISP in 1958, shortly after the development of FORTRAN. It was first implemented by Steve Russell on an IBM 704 computer. It is particularly suitable for Artificial Intelligence programs, as it processes symbolic information effectively.

Common Lisp originated, during the 1980s and 1990s, in an attempt to unify the work of several implementation groups that were successors to Maclisp, like ZetaLisp and NIL (New Implementation of Lisp) etc. It serves as a common language, which can be easily extended for specific implementation.

Programs written in Common LISP do not depend on machine-specific characteristics, such as word length etc.

2.4.1 Features of Common LISP

- It is machine-independent
- It uses iterative design methodology, and easy extensibility.
- It allows updating the programs dynamically.
- It provides high level debugging.
- It provides advanced object-oriented programming.
- It provides a convenient macro system.
- It provides wide-ranging data types like, objects, structures, lists, vectors, adjustable arrays, hash-tables, and symbols.
- It is expression-based.
- It provides an object-oriented condition system.
- It provides a complete I/O library.
- It provides extensive control structures.

2.4.2 Applications Built in LISP

Large successful applications built in Lisp.

- Emacs
- G2
- AutoCad
- Igor Engraver
- Yahoo Store

2.4.3 Basic Building Blocks in LISP

LISP programs are made up of three basic building blocks −

- atom
- list

An atom is a number or string of contiguous characters. It includes numbers and special characters.

Examples of some valid atoms

hello-from-tutorials-point

name

123008907

hello

Block#221

abc123

A list is a sequence of atoms and/or other lists enclosed in parentheses.

Examples of some valid lists –

(i am a list)

(a (a b c) d e fgh)

(father tom (susan bill joe))

(sun mon tue wed thur fri sat)

( )

A string is a group of characters enclosed in double quotation marks.

Examples of some valid strings –

" I am a string"

"a ba c d efg #$%^&!"

"Please enter the following details :"

"Hello from 'Tutorials Point'! "

2.4.4 Use of Single Quotation Mark

LISP evaluates everything including the function arguments and list members.

At times, we need to take atoms or lists literally and don't want them evaluated or treated as function calls.

To do this, we need to precede the atom or the list with a single quotation mark.

The following example demonstrates this.

Create a file named main. Lisp and type the following code into it.

```
(write-line "single quote used, it inhibits evaluation")
(write '(* 2 3))
(write-line " ")
(write-line "single quote not used, so expression evaluated")
(write (* 2 3))
```

2.5 Prolog in AI: Definition & Uses

Prolog as the name itself suggests, is the short form of Logical Programming. It is a logical and declarative programming language. Before diving deep into the concepts of Prolog, let us first understand what exactly logical programming is.

Logic Programming is one of the Computer Programming Paradigm, in which the program statements express the facts and rules about different problems within a system of formal logic. Here, the rules are written in the form of logical clauses, where head and body are present. For example, H is head and B1, B2, B3 are the elements of the body. Now if we state that "H is true, when B1, B2, B3 all are true", this is a rule. On the other hand, facts are like the rules, but without any body. So, an example of fact is "H is true".

Some logic programming languages like Data log or ASP (Answer Set Programming) are known as purely declarative languages. These languages allow statements about what the program should accomplish. There is no such step-by-step instruction on how to perform the task. However, other languages like Prolog, have declarative and also imperative properties. This may also include procedural statements like "To solve the problem H, perform B1, B2 and B3".

Some logic programming languages are given below −

- ALF (algebraic logic functional programming language).
- ASP (Answer Set Programming)
- CycL
- Datalog
- Fuzzy CLIPS
- Janus
- Parlog
- Prolog
- Prolog++
- ROOP

2.5.1 Logic and Functional Programming

We will discuss about the differences between Logic programming and the traditional functional programming languages. We can illustrate two using the below diagram fig 2.1

Some logic programming languages like Data log or ASP (Answer Set Programming) are known as purely declarative languages. These languages allow statements about what the program should accomplish. There is no such step-by-step instruction on how to perform the task. However, other languages like Prolog have declarative and also imperative properties.

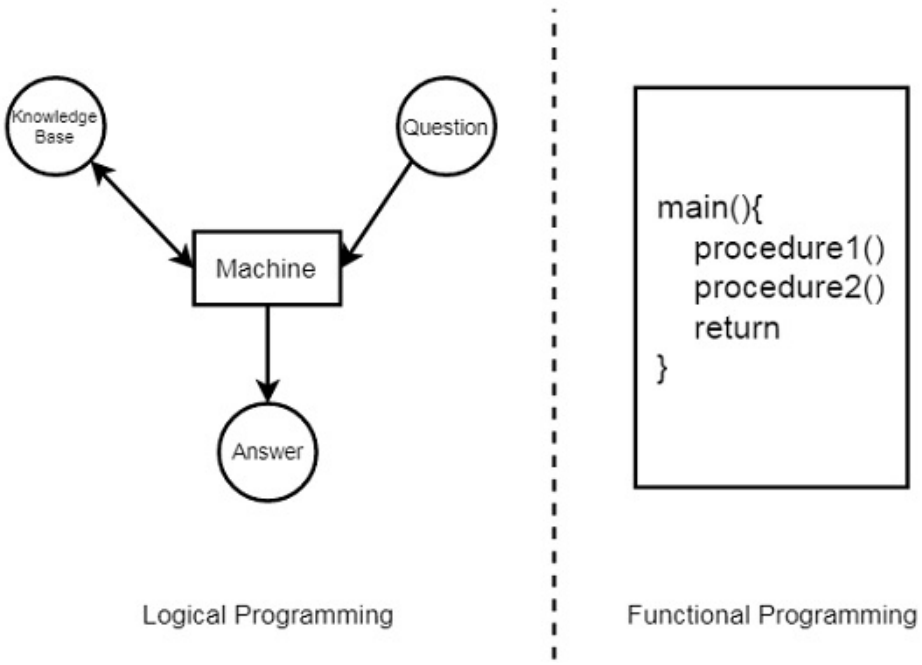


Figure 2.1 Logical Programming

2.5.2 What is Prolog?

Prolog or Programming in Logics is a logical and declarative programming language. It is one major example of the fourth generation language that supports the declarative programming paradigm. This is particularly suitable for programs that involve symbolic or non-numeric computation. This is the main reason to use Prolog as the programming language in Artificial Intelligence, where symbol manipulation and inference manipulation are the fundamental tasks.

In Prolog, we need not mention the way how one problem can be solved, we just need to mention what the problem is, so that Prolog automatically solves it. However, in Prolog we are supposed to give clues as the solution method.

Prolog language basically has three different elements −

Facts − The fact is predicate that is true, for example, if we say, “Tom is the son of Jack”, then this is a fact.

Rules − Rules are extinctions of facts that contain conditional clauses. To satisfy a rule these conditions should be met. For example, if we define a rule as –

Grandfather (X, Y) :- father(X, Z), parent(Z, Y)

This implies that for X to be the grandfather of Y, Z should be a parent of Y and X should be father of Z.

Questions − and to run a prolog program, we need some questions, and those questions can be answered by the given facts and rules.

2.5.3 Some Applications of Prolog

Prolog is used in various domains. It plays a vital role in automation system. Following are some other important fields where Prolog is used −

- Intelligent Database Retrieval
- Natural Language Understanding
- Specification Language
- Machine Learning
- Robot Planning
- Automation System
- Problem Solving

Prolog is also considered as a fourth generation programming language supporting the declarative programming paradigm. The well-known Japanese Fifth-Generation Computer Project, that was announced in 1981, adopted Prolog as a development language, and thereby grabbed considerable attention on the language and its capabilities. These ideas came together.

2.6 Uncertainty

To act rationally under uncertainty we must be able to evaluate how likely certain things are. With FOL a fact F is only useful if it is known to be true or false. But we need to be able to evaluate how likely it is that F is true. By weighing likelihoods of events (probabilities) we can develop mechanisms for acting rationally under uncertainty.

2.6.1 Dental Diagnosis example.

In FOL we might formulate

P. symptom (P, toothache) → disease (p, cavity) disease (p, gumDisease)

Disease (p, foodStuck)

When do we stop?

Cannot list all possible causes.

We also want to rank the possibilities. We don't want to start drilling for a cavity before checking for more likely causes first.

2.6.2 Axioms Of Probability

Given a set U (universe), a probability function is a function defined over the subsets of U that maps each subset to the real numbers and that satisfies the Axioms of Probability

1.Pr(U) = 1

2.Pr(A) [0,1]

3.Pr(A ∈B) = Pr(A) + Pr(B) –Pr(A ∩B)

Note if A ∩B = {} then Pr(A ∪B) = Pr(A) + Pr(B)

2.6.3 Review of Probability

- Natural way to represent uncertainty
- People have intuitive notions about probabilities
- Many of these are wrong or inconsistent
- Most people don't get what probabilities mean
- Understanding Probabilities
- Initially, probabilities are "relative frequencies
- This works well for dice and coin flips
- For more complicated events, this is problematic

2.6.4 Probabilities and Beliefs

- Suppose I have flipped a coin and hidden the outcome
- What is P(Heads)?
- Note that this is a statement about a belief, not a statement about the world
- The world is in exactly one state (at the macro level) and it is in that state with probability 1.
- Assigning truth values to probability statements is very tricky business
- Must reference speakers state of knowledge

2.6.5 Frequentism and Subjectivism

- Frequentists hold that probabilities must come from relative frequencies
- This is a purist viewpoint
- This is corrupted by the fact that relative frequencies are often unobtainable
- Often requires complicated and convoluted
- Assumptions to come up with probabilities
- Subjectivists: probabilities are degrees of belief

UNIT – III – ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS – SMRA3003

3 SEARCH METHODS AND KNOWLEDGE REPRESENTATION

3.1 Fuzzy Logic

(FL) is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.

The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO.

The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as

- Certainly YES
- Possible YES
- Cannot SAY
- Possibly NO
- Certainly NO

3.1.1 Implementation

- It can be implemented in systems with various sizes and capabilities ranging from small micro-controllers to large, networked, workstation-based control systems.
- It can be implemented in hardware, software, or a combination of both.

3.1.2 Why Fuzzy Logic?

Fuzzy logic is useful for commercial and practical purposes.

- It can control machines and consumer products.
- It may not give accurate reasoning, but acceptable reasoning.
- Fuzzy logic helps to deal with the uncertainty in engineering.

3.1.3 Fuzzy Logic Systems Architecture

- Fuzzification Module − It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps such as Figure 3.1
- Knowledge Base − It stores IF-THEN rules provided by experts.
- Inference Engine − It simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.
- Defuzzification Module − It transforms the fuzzy set obtained by the inference engine into a crisp value.

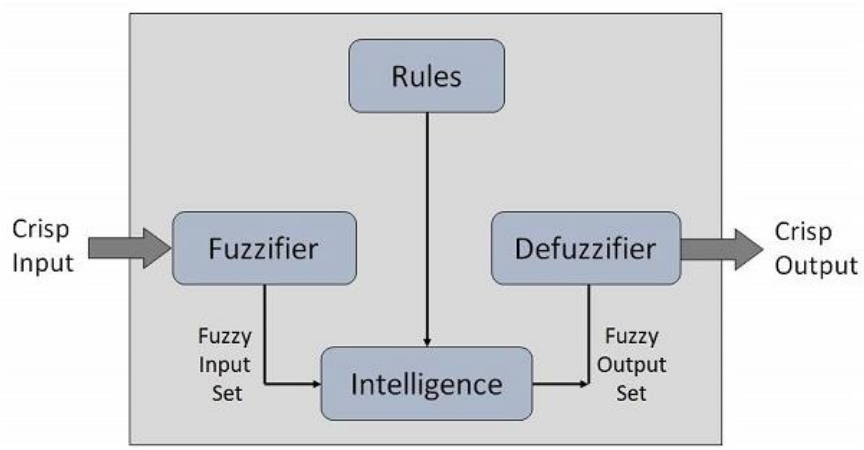


Figure 3.1 Fuzzy Logic Architecture

3.1.4 Membership Function

Membership functions allow you to quantify linguistic term and represent a fuzzy set graphically. A membership function for a fuzzy *set A* on the universe of discourse X is defined as μ_A:X → [0,1].

Here, each element of *X* is mapped to a value between 0 and 1. It is called membership value or degree of membership. It quantifies the degree of membership of the element in *X* to the fuzzy set *A*. Figure 3.2

x axis represents the universe of discourse.

y axis represents the degrees of membership in the [0, 1] interval.

.

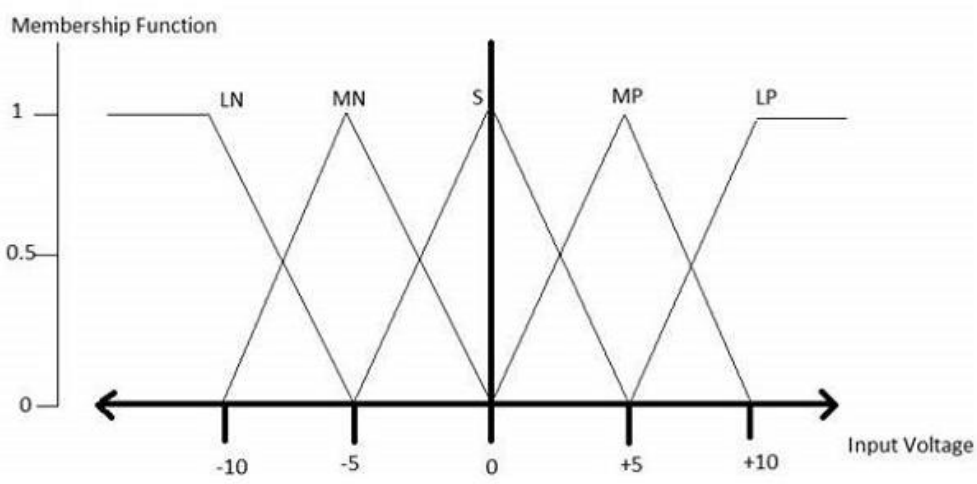


Figure 3.2 Membership Function

The triangular membership function shapes are most common among various other membership function shapes such as trapezoidal, singleton, and Gaussian.

3.1.5. Example of a Fuzzy Logic System

Let us consider an air conditioning system with 5-level fuzzy logic system. This system adjusts the temperature of air conditioner by comparing the room temperature and the target temperature value. Figure 3.3

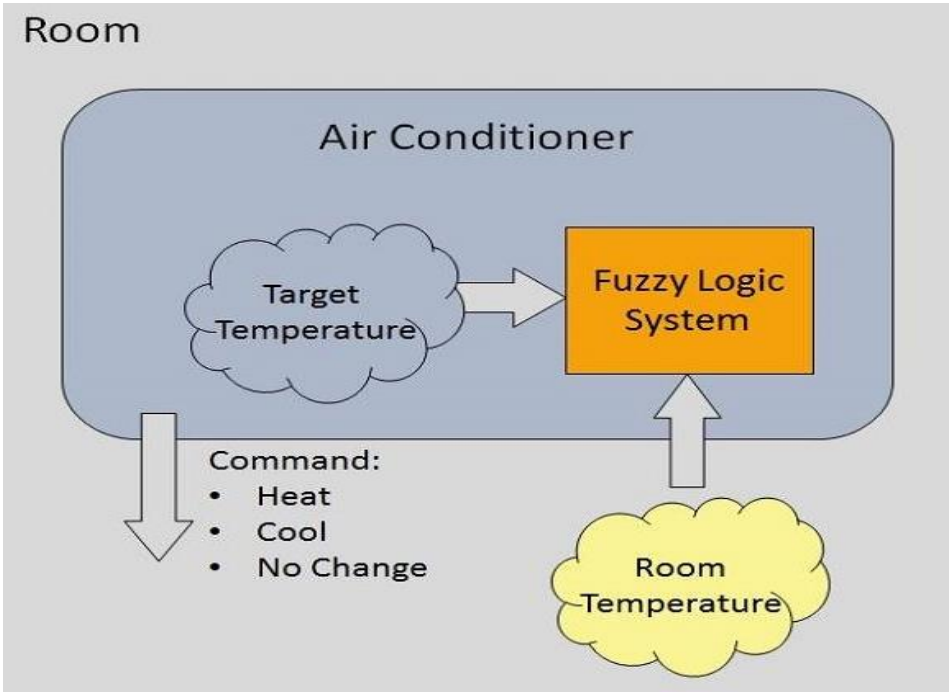


Figure 3.3 Example of Fuzzy Logic

3.1.6 Algorithm

- Define linguistic Variables and terms (start)
- Construct membership functions for them. (start)
- Construct knowledge base of rules (start)
- Convert crisp data into fuzzy data sets using membership functions. (fuzzification)
- Evaluate rules in the rule base. (Inference Engine)
- Combine results from each rule. (Inference Engine)
- Convert output data into non-fuzzy values. (defuzzification)

3.1.7 Application Areas of Fuzzy Logic

The key application areas of fuzzy logic are as given −

Automotive Systems

- Automatic Gearboxes
- Four-Wheel Steering
- Vehicle environment control

Consumer Electronic Goods

- Hi-Fi Systems
- Photocopiers
- Still and Video Cameras
- Television

Domestic Goods

- Microwave Ovens
- Refrigerators
- Toasters
- Vacuum Cleaners

- Washing Machines

Environment Control

- Air Conditioners/Dryers/Heaters
- Humidifiers

3.1.8 Advantages of FLSs

- Mathematical concepts within fuzzy reasoning are very simple.
- You can modify a FLS by just adding or deleting rules due to flexibility of fuzzy logic.
- Fuzzy logic Systems can take imprecise, distorted, noisy input information.
- FLSs are easy to construct and understand.
- Fuzzy logic is a solution to complex problems in all fields of life, including medicine, as it resembles human reasoning and decision making.

3.1.9 Disadvantages of FLSs

- There is no systematic approach to fuzzy system designing.
- They are understandable only when simple.
- They are suitable for the problems which do not need high accuracy.

3.2 Probabilistic Bayesian Networks Inference

Use of Bayesian Network (BN) is to estimate the probability that the hypothesis is true based on evidence.

- Bayesian Networks Inference:
- Deducing Unobserved Variables
- Parameter Learning
- Structure Learning

3.2.1 Deducing Unobserved Variables

With the help of this network, we can develop a comprehensive model that delineates the relationship between the variables. It is used to answer probabilistic queries about them. We can use it to observe the updated knowledge of the state of a subset of variables. For computing, the posterior distribution of the variables with the given evidence is called probabilistic inference. For detection applications, it gives universal statistics. When anyone wants to select values for the variable subset, it minimizes some expected loss function, for instance, the probability of decision error. A BN is a mechanism for applying Bayes' theorem to complex problems.

Popular inference methods are:

- Variable Elimination
  Variable Elimination eliminates the non-observed non-query variables. It eliminates one by one by distributing the sum over the product.
- Clique Tree Propagation
  It caches the computation to query many variables at one time and also to propagate new evidence.
- Recursive Conditioning
  Recursive conditioning allows a tradeoff between space and time. It is equivalent to the variable elimination method if sufficient space is available.

3.2.2 Parameter Learning

To specify the BN and thus represent the joint probability distribution, it is necessary to specify for each node X. Here, the probability distribution for the node X is conditional, based on its parents. There can be many forms of distribution of X. Discrete or Gaussian distributions simplifies calculations. Sometimes constraints on distribution are only known. To determine a single distribution, we can use the principle of maximum entropy. The only one who has the greatest entropy is given the constraints.

Conditional distributions include parameters from data and unknown. Sometimes by using the most likely approach, we can estimate the data. When there are unobserved variables, direct maximization of the likelihood is often complex. EMA refers to Expectation-maximization algorithm. It is for computing expected values of the unobserved variables by

performing the maximization of the likelihood with an assumption that the prior expectations are correct. This process converges on most likelihood values for parameters under mild condition.

To treat parameters as additional unobserved variables, Bayesian is an approach. We use BN to compute a posterior distribution conditional upon observed data and then to integrate out the parameters. This approach can be costly and lead to large dimension model. Thus, in real practice, classical parameter-setting are more common approaches.

3.2.3 Structure Learning

BN is specified by an expert and after that, it is used to perform inference. The task of defining the network is too complex for humans in other applications. The parameters of the local distributions and the network structure must learn from data in this case.

A challenge pursued that within machine learning is automatically learning the graph structure of a BN. After that, the idea went back to an algorithm developed by Rebane and Pearl (1987**)**. The triplets allowed in a Directed Acyclic Graph (DAG)**:**

X and Z are independent given Y. Represent the same dependencies by Type 1 and 2, so it is, indistinguishable. We can uniquely identify Type 3. All other pairs are dependent and X and Z are marginally independent. So, while the skeletons of these three triplets are identical, the direction of the arrows is somehow identifiable. When X and Z have common parents, the same distinction applies except that one condition on those parents. We develop the algorithm to determine the skeleton of the underlying graph. After that orient, all arrows whose directionality is estimated by the conditional independencies are observed.

3.3 Possible World Representation

An Artificial intelligence system has the following components for displaying intelligent behaviour: Figure 3.4

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

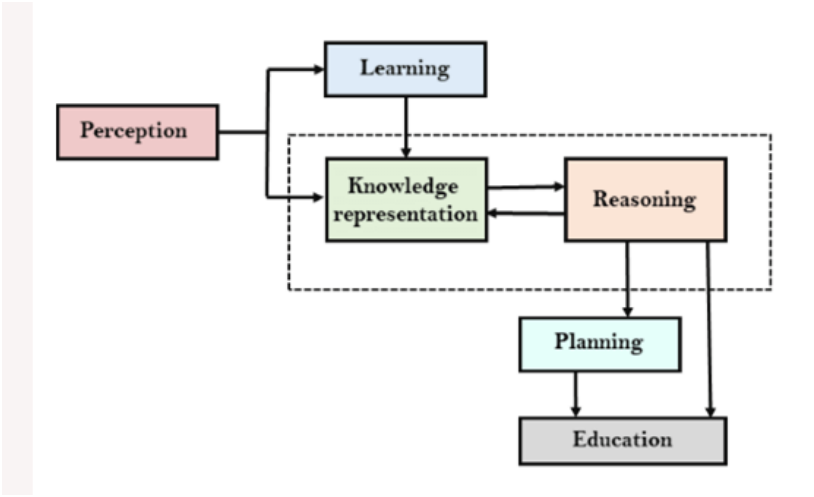


Figure 3.4 Possible World Representations

The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment.

In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

3.3.1 Approaches to knowledge representation:

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

- Simple relational knowledge:

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

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

This approach has little opportunity for inference.

Example: The following is the simple relational knowledge representation.

- Inheritable knowledge:

In the inheritable knowledge approach, all data must be stored into a hierarchy of classes. All classes should be arranged in a generalized form or a hierarchal manner. In this approach, we apply inheritance property.

Elements inherit values from other members of a class. This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation. Figure 3.5

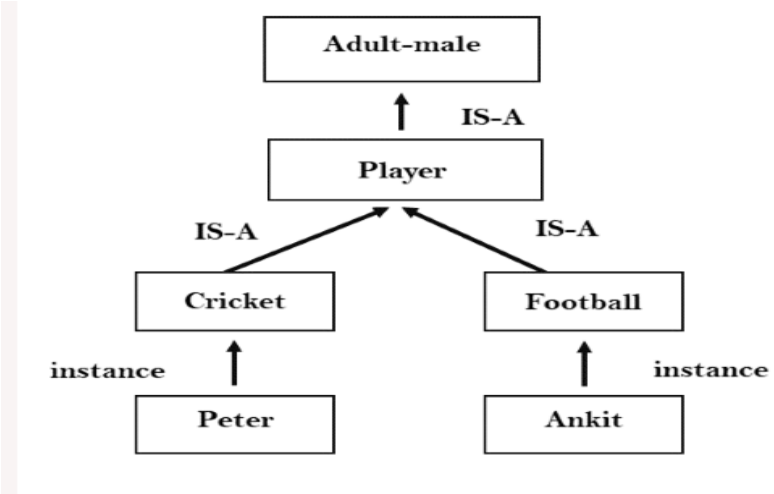


Figure 3.5 Inheritable knowledge

- Inferential knowledge:

  Inferential knowledge approach represents knowledge in the form of formal logics. This approach can be used to derive more facts.It guaranteed correctness.

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

  All men are mortal Then it can represent as**;** man(Marcus) ∀x = man (x) ----------> mortal (x)s

- Procedural knowledge:

Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed. In this approach, one important rule is used which is If-Then rule. In this knowledge, we can use various coding languages such as LISP language and Prolog language. We can easily represent heuristic or domain-specific knowledge using this approach. But it is not necessary that we can represent all cases in this approach.

- Requirements for knowledge Representation system:

Representational Accuracy**:** KR system should have the ability to represent all kind of required knowledge.

Inferential Adequacy: KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.

Inferential Efficiency: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.

Acquisition efficiency- The ability to acquire the new knowledge easily using automatic methods.

3.4 Search Algorithms in Artificial Intelligence

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

3.4.1 Search Algorithm Terminologies:

Search**:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:

Search Space: Search space represents a set of possible solutions, which a system may have. Start State: It is a state from where agent begins the search. Goal test**:** It is a function which observe the current state and returns whether the goal state is achieved or not.

Search tree: A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.

Actions: It gives the description of all the available actions to the agent.

Transition model: A description of what each action do, can be represented as a transition model.

Path Cost: It is a function which assigns a numeric cost to each path.

Solution: It is an action sequence which leads from the start node to the goal node.

Optimal Solution: If a solution has the lowest cost among all solutions.

3.4.2 Properties of Search Algorithms:

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

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

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

Time Complexity: Time complexity is a measure of time for an algorithm to complete its task. Space Complexity: It is the maximum storage space required at any point during the search, as the complexity of the problem.

3.4.3 Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms. Figure 3.6

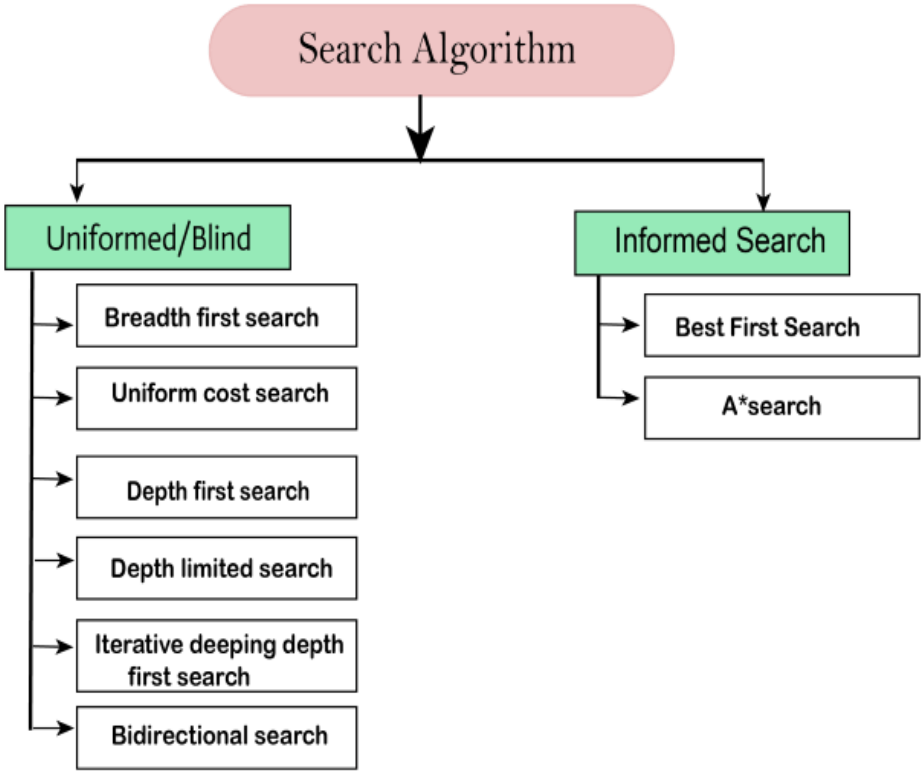


Figure 3.6 Types of Search Algorithm

3.4.4 Uninformed/Blind Search:

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

It can be divided into five main types:

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

3.4.5 Informed Search

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

A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time. Informed search can solve much complex problem which could not be solved in another way.An example of informed search algorithms is a traveling salesman problem.

3.4.6. Breadth-first Search:

Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search. Figure 3.7 Breadth-first search

BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level. The breadth-first search algorithm is an example of a general-graph search algorithm. Breadth-first search implemented using FIFO queue data structure.

A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time. Informed search can solve much complex problem which could not be solved in another way.An example of informed search algorithms is a traveling salesman problem

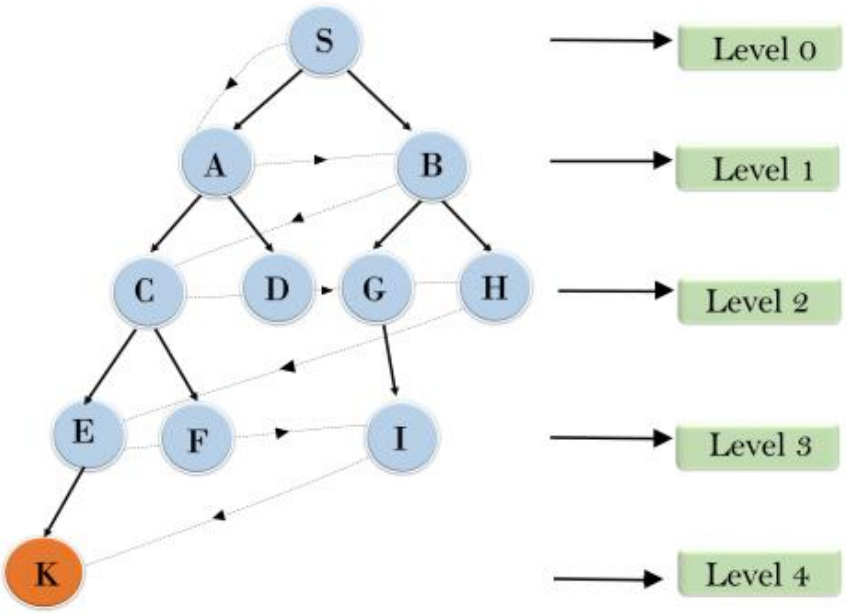


Figure 3.7 Breadth-first searches

3.4.7. Depth first Search:

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

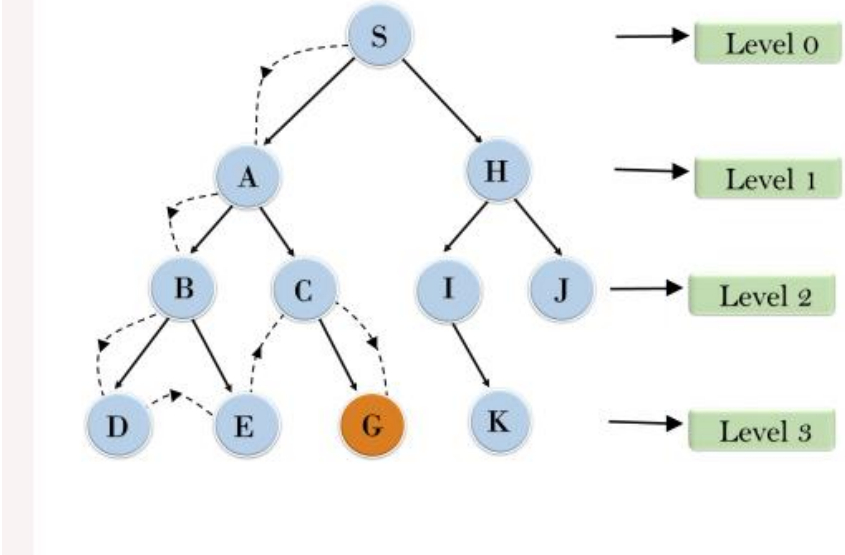


Figure 3.8 Breadth-first searches

UNIT – IV – ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS – SMRA3003

4 KNOWLEDGE ORGANISATION AND COMMUNICATION IN EXPERT SYSTEMS

4.1 Planning Classical Planning:

AI as the study of rational action, which means that planning—devising a plan of action to achieve one's goals—is a critical part of AI. We have seen two examples of planning agents so far the search-based problem-solving agent.

4.1.1 Definition of Classical Planning:

The problem-solving agent can find sequences of actions that result in a goal state.

But it deals with atomic representations of states and thus needs good domain-specific heuristics to perform well. The hybrid propositional logical agent can find plans without domain- specific heuristics because it uses domain-independent heuristics based on the logical structure of the problem but it relies on ground (variable-free) propositional inference, which means that it may be swamped when there are many actions and states. For example,in the world, the simple action of moving a step forward had to be repeated for all four agent orientations, T time steps, and n^2currentlocations.

In response to this, planning researchers have settled on a factored representation one in which a state of the world is represented by a collection of variables. We use a language called PDDL, the Planning Domain Definition Language that allows us to express all 4Tn2 actions with one action schema. There have been several versions of PDDL. we select a simple version and alter its syntax to be consistent with the rest of the book. We now show how PDDL describes the four things we need to define a search problem: the initial state, the actions that are available in a state, the result of applying an action, and the goal test.

Each state is represented as a conjunction of flaunts that are ground, functionless atoms. For example, Poor ∧ Unknown might represent the state of a hapless agent, and a state in a package delivery problem might be At(Truck 1, Melbourne) ∧ At(Truck 2, Sydney). Database semantics is used: the closed-world assumption means that any flaunts that are not mentioned are false, and the unique names assumption means that Truck 1 and Truck 2 are distinct.

A set of ground (variable-free) actions can be represented by a single action schema. The schema is a lifted representation—it lifts the level of reasoning from propositional logic to a restricted subset of first-order logic. For example, here is an action schema for flying a plane from one location to another: Action (Fly (p, from, to),

The schema consists of the action name, a list of all the variables used in the schema, a precondition and an effect.
A set of action schemas serves as a definition of a planning domain. A specific problem within the domain is defined with the addition of an initial state and a goal.
state is a conjunction of ground atoms. (As with all states, the closed-world assumption is used, which means that any atoms that are not mentioned are false?)

The goal is just like a precondition: a conjunction of literals (positive or negative) that may contain variables, such as At(p, SFO) ∧ Plane(p). Any variables are treated as existentially quantified, so this goal is to have any plane at SFO. The problem is solved when we can find a sequence of actions that end in a states that entails the goal. Example: Air cargo transport

An air cargo transport problem involving loading and unloading cargo and flying it from place to place. The problem can be defined with three actions: Load , Unload , and Fly . The actions affect two predicates: In(c, p) means that cargo c is inside plane p, and At(x, a) means that object x (either plane or cargo) is at airport a.
Note that some care must be taken to make sure the At predicates are maintained properly. When a plane flies from one airport to another, all the cargo inside the plane goes with it. In first-order logic it would be easy to quantify over all objects that are inside the plane. But basic PDDL does not have a universal quantifier, so we need a different solution.

The approach we use is to say that a piece of cargo ceases to be At anywhere when it is In a plane; the cargo only becomes At the new airport when it is unloaded. So At really means “available for use at a given location.”The complexity of classical planning :

We consider the theoretical complexity of planning and distinguish two decision problems. Plan SAT is the question of whether there exists any plan that solves a planning problem. Bounded Plan SAT asks whether there is a solution of length k or less; this can be used to find an optimal plan.

The first result is that both decision problems are decidable for classical planning. The proof follows from the fact that the number of states is finite. But if we add function symbols to the language, then the number of states becomes infinite, and PlanSAT becomes only semi decidable: an algorithm exists that will terminate with the correct answer for any solvable problem, but may not terminate on unsolvable problems. The Bounded PlanSAT problem remains decidable even in the presence of functionsymbols.

Both Plan SAT and Bounded Plan SAT are in the complexity class PSPACE, a class that is larger (and hence more difficult) than NP and refers to problems that can be solved by a deterministic Turing machine with a polynomial amount of space. Even if we make some rather severe restrictions, the problems remain quite difficult.

4.1.2 State-Space Search:

Now that we have shown how a planning problem maps into a search problem, we can solve planning problems with any of the heuristic search algorithms from Chapter 3 or a local search algorithm from Chapter 4 (provided we keep track of the actions used to reach the goal). From the earliest days of planning research (around 1961) until around 1998 it was assumed that forward state-space search was too inefficient to be practical. It is not hard to come up with reasons why.

First, forward search is prone to exploring irrelevant actions. Consider the noble task of buying a copy of AI: A Modern Approach from an online bookseller. Suppose there is an action schema Buy(isbn) with effect Own(isbn). ISBNs are 10 digits, so this action schema represents 10 billion ground actions. An uninformed forward-search algorithm would have to start enumerating these 10 billion actions to find one that leads to the goal.
Second, planning problems often have large state spaces.

Consider an air cargo problem with 10 airports, where each airport has 5 planes and 20 pieces of cargo. The goal is to move all the cargo at airport A to airport B. There is a simple solution to the problem: load the 20 pieces of cargo into one of the planes at A, fly the plane to B, and unload the cargo.

Finding the solution can be difficult because the average branching factor is huge: each of the 50 planes can fly to 9 other airports, and each of the 200 packages can be either unloaded (if it is loaded) or loaded into any plane at its airport (if it is unloaded). So in any state there is a minimum of 450 actions (when all the packages are at airports with no planes) and a maxi`

4.1.3 Backward (regression) relevant-states search:

In regression search we start at the goal and apply the actions backward until we find a sequence of steps that reaches the initial state. It is called relevant-states search because we only consider actions that are relevant to the goal (or current state).

As in belief-state search (Section 4.4), there is a set of relevant states to consider at each step, not just a single state. We start with the goal, which is a conjunction of literals forming a description of a set of states—for example, the goal ¬Poor ∧ Famous describes those states in which Poor is false, Famous is true, and any other fluent can have any value. If there are n ground flaunts in a domain, then there are 2n ground states (each fluent can be true or false), but 3n descriptions of sets of goal states (each fluent can be positive, negative, or not mentioned).

In general, backward search works only when we know how to regress from a state description to the predecessor state description. For example, it is hard to search backwards for a solution to the
problem because there is no easy way to describe the states that are one move away from the goal.

Happily, the PDDL representation was designed to make it easy to regress actions—if a domain can be expressed in PDDL, then we can do regression search on it.

The final issue is deciding which actions are candidates to regress over. In the forward direction we chose actions that were applicable—those actions that could be the next step in the plan. In backward search we want actions that are relevant—those actions that could be the last step in a plan leading up to the current goal state.

4.2 Heuristics for Planning:

Neither forward nor backward search is efficient without a good heuristic function. Recall from Chapter 3 that a heuristic function h(s) estimates the distance from a state s to the goal and that if we can derive an admissible heuristic for this distance one that does not over Estimate then we causes A search to find optimal solutions. An admissible heuristic can be derived by defining a relaxed problem that is easier to solve. The exact cost of a solution to this easier problem then becomes the heuristic for the original problem.

By definition, there is no way to analyze an atomic state, and thus it it requires some ingenuity by a human analyst to define good domain-specific heuristics for search problems with atomic states. Planning uses a factored representation for states and action schemas. That makes it possible to define good domain-independent heuristics and for programs to automatically apply a good domain- independent heuristic for a given problem.

4.3 Planning Graphs:

All of the heuristics we have suggested can suffer from inaccuracies. This section shows how a special data structure called a planning graph can be used to give better heuristic estimates. These heuristics can be applied to any of the search techniques we have seen so far. Alternatively, we can search for a solution over the space formed by the planning graph, using an algorithm called GRAPH PLAN. A planning problem asks if we can reach a goal state from the initial state. Suppose we are given a tree of all possible actions from the initial state to successor states, and their successors, and so on.

The planning graph can’t answer definitively whether G is reachable fromS0, but it can estimate how many steps it takes to reach G. The estimate is always correct when it reports the goal is not reachable, and it never overestimates the number of steps, so it is an admissible heuristic.

A planning graph is a directed graph organized into levels: first a level S0 for the initial state, consisting of nodes representing each fluent that holds in S0; then a level A0 consisting of nodes for each ground action that might be applicable in S0; then alternating levels Si followed by Ai; until we reach a termination condition (to be discussed later).

Roughly speaking, Si contains all the literals that could hold at time i, depending on the actions executed at preceding time steps. If it is possible that either P or ¬P could hold, then both will be represented in Si.

Also roughly speaking, Ai contains all the actions that could have their preconditions satisfied at time he possible negative interactions among actions; therefore, a literal might show up at level Sj when actually it could not be true until a later level, if at all. (A literal will never show up too late.) Despite the possible error, the level j at which a literal first appears is a good estimate of how difficult it is to achieve the literal from the initial state.

We now define mutex links for both actions and literals. A mutex relation holds between two actions at a given level if any of the following three conditions holds:

- Inconsistent effects: one action negates an effect of the other. For example, Eat(Cake) and the persistence of Have(Cake) have inconsistent effects because they disagree on the effect Have(Cake).
- Interference: one of the effects of one action is the negation of a precondition of the other. For example Eat(Cake) interferes with the persistence of Have(Cake) by its precondition.
- Competing needs: one of the preconditions of one action is mutually exclusive with a precondition of the other. For example, Bake (Cake) and Eat (Cake) are mutex because they compete on the value of the Have (Cake) precondition.

A mute relation holds between two literals at the same level if one is the negation of the other or if each possible pair of actions that could achieve the two literals is mutually exclusive. This condition is called inconsistent support. For example, Have (Cake) and Eaten (Cake) are mutex in S1 because the only way of achieving Have (Cake), the persistence action, is mutex with the only way of achieving Eaten (Cake), namely Eat(Cake). In S2 the two literals are not mutex, because there are new ways of achieving them, such as Bake(Cake) and the persistence of Eaten(Cake), that are not mutex. Other Classical Planning Approaches:

Currently the most popular and effective approaches to fully automated planning are:

- Translating to a Boolean satisfiability (SAT)problem
- Forward state-space search with carefully crafted heuristics
- Search using a planning graph (Section10.3)

These three approaches are not the only ones tried in the 40-year history of automated planning. Shows some of the top systems in the International Planning Competitions, which have been held every even year since 1998.

In this section we first describe the translation to a satisfiability problem and then describe three other influential approaches: planning as first-order logical deduction; as constraint satisfaction; and as plan refinement.

Classical planning as Boolean satisfy ability:

We saw how SATPLAN solves planning problems that are expressed in propositional logic. Here we show how to translate a PDDL description into a form that can be processed by SATPLAN. The translation is a series of straightforward steps:

- Proposition Alizethe actions replace each action schem awith a set of ground actions formed by substituting constants for each of the variables. These ground actions are not part of the translation, but will be used in subsequentsteps.
- Define the initial state:assert F0 for every fluent Fin the problem's initial state and for every fluent not mentioned in the initialstate.
- Proposition Alize the goal: for every variable in the goal, replace the literal that contain the variable with a disjunction over constants. For example, the goal of having block A on another block,
- Add successor-state axioms: For each fluent F , add an axiom of the form where Action CausesF is a disjunction of all the ground actions that have F in their add list, and Action CausesNotF is a disjunction of all the ground actions that have F in their delete list.

4.4 Analysis of Planning approaches:

Planning combines the two major areas of AI we have covered so far: search and logic. A planner can be seen either as a program that searches for a solution or as one that (constructively) proves the existence of a solution.

The cross-fertilization of ideas from the two areas has led both to improvements in performance amounting to several orders of magnitude in the last decade and to an increased use of planners in industrial applications. Unfortunately, we do not yet have a clear understanding of which techniques work best on which kinds of problems. Quite possibly, new techniques will emerge that dominate existing methods.

Planning is foremost an exercise in controlling combinatorial explosion. If there are n propositions in a domain, then there are 2n states. As we have seen, planning is PSPACE-hard. Against such pessimism, the identification of independent sub problems can be a powerful weapon. In the best case—full decomposability of the problem—we get an exponential speedup

Decomposability is destroyed, however, by negative interactions between actions. GRAPHPLAN records murexes to point out where the difficult interactions are. SATPLAN rep- resents a similar range of mute relations, but does so by using the general CNF form rather than a specific data structure.

Forward search addresses the problem heuristically by trying to find patterns (subsets of propositions) that cover the independent sub problems. Since this approach is heuristic, it can work even when the sub problems are not completely independent.

Sometimes it is possible to solve a problem efficiently by recognizing that negative interactions can be ruled out. We say that a problem has serialize able sub goals if there exists an order of sub goals such that the planner can achieve them in that order without having to undo any of the previously achieved sub goals.

For example, in the blocks world, if the goal is to build a tower (e.g., A on B, which in turn is on C, which in turn is on the Table, as in Figure 10.4 on page 371), then the sub goals are serialize able bottom to top: if we first achieve C on Table, we will never have to undo it while we are achieving the other sub goals. Planners such as GRAPHPLAN, SATPLAN, and FF have moved the field of planning forward, by raising the level of performance of planning systems.

4.5 Planning and Acting in the Real World:

This allows human experts to communicate to the planner what they know about how to solve the problem. Hierarchy also lends itself to efficient plan construction because the planner can solve a problem at an abstract level before delving into details. Presents agent architectures that can handle uncertain environments and inter leave deliberation with execution, and gives some examples of real- world systems.

4.5.1 Time, Schedules, and Resources:

The classical planning representation talks about what to do, and in what order, but the representation cannot talk about time: how long an action takes and when it occurs. For example, the planners of Chapter 10 could produce a schedule for an airline that says which planes are assigned to which flights, but we really need to know departure and arrival times as well. This is the subject matter of scheduling.

The real world also imposes many resource constraints; for example, an airline has a limited number of staff—and staff who are on one flight cannot be on another at the same time. This section covers methods for representing and solving planning problems that include temporal and resource constraints.

The approach we take in this section is "plan first, schedule later":that is, we divide the overall problem into a planning phase in which actions are selected, with some ordering constraints, to meet the goals of the problem, and a later scheduling phase, in which temporal information is added to the plan to ensure that it meets resource and dead line constraints.

This approach is common in real-world manufacturing and logistical settings, where the planning phase is often performed by human experts. The automated methods of Chapter 10 can also be used for the planning phase, provided that they produce plans with just the minimal ordering constraints required for correctness.

G RAPHPLAN (Section 10.3), SATPLAN (Section 10.4.1), and partial-order planners (Section 10.4.4) can do this; search-based methods (Section 10.2) produce totally ordered plans, but these can easily be converted to plans with minimal ordering constraints.

4.5.2 Hierarchical Planning:

The problem-solving and planning methods of the preceding chapters all operate with a fixed set of atomic actions. Actions can be strung together into sequences or branching networks; state-of-the-art algorithms can generate solutions containing thousands of actions.

For plans executed by the human brain, atomic actions are muscle activations. In very round numbers, we have about 103 muscles to activate (639, by some counts, but many of them have multiple subunits); we can modulate their activation perhaps 10 times per second; and we are alive and awake for about 109 seconds in all.

Thus, a human life contains about 1013 actions, give or take one or two orders ofmagnitude. Even if we restrict ourselves to planning over much shorter time horizons—for example, a two-week vacation in Hawaii—a detailed motor plan would contain around 1010 actions. This is a lot more than 1000.

To bridge this gap, AI systems will probably have to do what humans appear to do: plan at higher level of abstraction. Are a son able plan for the Hawaii vacation might begot San Francisco airport stake Hawaiian Airlines flight 11 to Honolulu; do vacation stuff for two Planning and Acting in Nondeterministic Domains: While the basic concepts are the same as in there are also significant differences.

These arise because planners deal with factored representations rather than atomic representations. This affects the way were present the agent's capability for action and observation and the way we represent belief states—the sets of possible physical states the agent might be in—for unobservable and partially observable environments.

We can also take ad- vantage of many of the domain-independent methods given in Chapter 10 for calculating search heuristics.

Consider this problem: given a chair and a table, the goal is to have them match—have the same color. In the initial state we have two cans of paint, but the colors of the paint and the furniture are unknown. Only the table is initially in the agent's field of view:

There are two actions: removing the lid from a paint can and painting an object using the paint from an open can. The action schemas are straightforward, with one exception: we now allow preconditions and effects to contain variables that are not part of the action's variable ist.That is, Paint(x,can) does not mention the variable c, representing the color of the paint in the can. In the fully observable case, this is not allowed—we would have to name the action Paint(x, can, c). But in the partially observable case, we might or might not know what color is in the can. (The variable c is universally quantified, just like all the other variables in an action schema.

To solve a partially observable problem, the agent will have to reason about the percepts it will obtain when it is executing the plan. The percept will be supplied by the agent's sensors when it is actually acting, but when it is planning it will need a model of its sensors. In Chapter 4, this model was given by a function, PERCEPT(s). For planning, we augment PDDL with a new type of schema, the percept schema:

4.6 Multi agent planning:

we have assumed that only one agent is doing the sensing, planning, and acting. When there are multiple agents in the environment, each agent faces a multi agent planning problem in which it tries to achieve its own goals with the help or hindrance of others.

Between the purely single-agent and truly multi agent cases is a wide spectrum of problems that exhibit various degrees of decomposition of the monolithic agent. An agent with multiple effectors that can operate concurrently—for example, a human who can type and speak at the same time—needs to do multi effector planning to manage each effector while handling positive and negative interactions among the effectors.

When the effectors are physically decoupled into detached units—as in a fleet of delivery robots in a factory—multi effectors planning becomes multimode planning. A multiband problem is still a "standard" single-agent problem as long as the relevant sensor information collected by each body can be pooled—either centrally or within each body—to form a common estimate of the world state that then informs the execution of the overall plan; in this case, the multiple bodies act as a single body

When a single entity is doing the planning, there is really only one goal, which all the bodies necessarily share. When the bodies are distinct agents that do their own planning, they may still share identical goals; for example, two human tennis players who form a doubles team share the goal of winning the match.

Even with shared goals, however, the multiband and multi agent cases are quite different. In a multimode robotic doubles team, a single plan dictates which body will go where on the court and which body will hit the ball. In a multi- agent doubles team, on the other hand, each agent decides what to do; without some method for coordination, both agents may decide to cover the same part of the court and each may leave the ball for the other to hit.

4.7 Planning with multiple simultaneous actions

For the time being, we will treat the multi effector, multibody, and multi agent settings in the same way, labeling them generically as multi actor settings, using the generic term actor to cove cover effectors, bodies, and agents.

The goal of this section is to work out how to define transition models, correct plans, and efficient planning algorithms for the multi actor setting.

A correct plan is one that, if executed by the actors, achieves the goal. (In the true multi agent setting, of course, the agents may not agree to execute any particular plan, but at least they will know what plans would work if they did agree to execute them.) For simplicity, we assume perfect synchronization: each action takes the same amount of time and actions at each point in the joint plan are simultaneous.

Having put the actors together into a multi actor system with a huge branching factor, the principal focus of research on multi actor planning has been to decouple the actors to the extent possible, so that the complexity of the problem grows linearly with n rather than exponentially.

If the actors have no interaction with one another—for example, n actors each playing a game of solitaire—then we can simply solve n separate problems. If the actors are loosely coupled, can we attain something close to this exponential improvement? This is, of course, a central question in many areas of AI.

The standard approach to loosely coupled problems is to pretend the problems are completely decoupled and then fix up the interactions. For the transition model, this means writing action schema as as if the actors act ed independently. Let's see how this works for the doubles tennis problem.Let'ssuppose that at one point in the game, the team has the goal of returning the ball that has been hit to them and ensuring that at least one of them is covering the net

4.8 Planning with multiple agents Cooperation and coordination:

Now let us consider the true multi agent setting in which each agent makes its own plan. To start with, let us assume that the goals and knowledge base are shared. One might think that this reduces to the multiband case—each agent simply computes the joint solution and executes its own part of that solution.Alas,the"the"in"the joint solution "is misleading. For our double steam, more than one joint solution exists:

If both agents can agree on either plan 1 or plan 2, the goal will be achieved. But if A chooses plan 2 and B chooses plan 1, then nobody will return the ball. Conversely, if A chooses 1 and B chooses 2, then they will both try to hit the ball.

One option is to adopt a convention before engaging in joint activity. A convention is any constraint on the selection of joint plans. For example, the convention "stick to you rside of the court "would rule out plan 1, causing the doubles partners to select plan 2. Drivers on a road face the problem of not colliding with each other; this is (partially) solved by adopting

the convention “stay on the right side of the road” in most countries; the alternative, “stay on the leftside,”works equally well as long as allagentsinan environment agree. Similar considerations apply to the development of human language, where the important thing is not which language each individual should speak, but the fact that a community all speaks the same language. When conventions are widespread, they are called social laws.

Conventions can also arise through evolutionary processes. For example, seed-eating harvester ants are social creatures that evolved from the less social wasps. Colonies of ants execute very elaborate joint plans without any centralized control—the queen’s job is to re-produce, not to do centralized planning—and with very limited computation,

Communication and memory capabilities in each ant (Gordon, 2000, 2007). The colony has many roles, including interior workers, patrollers, and foragers. Each ant chooses to perform a role ac- cording to the local conditions it observes. One final example of cooperative multi agent behaviour appears in the flocking behavior of birds.

We can obtain a reasonable simulation of a flock if each bird agent (sometimes called a bode) observes the positions of its nearest neighbours and then chooses the heading and acceleration that maximizes the weighted sum of these three components.

- Cohesion: a positive score for getting closer to the average position of the neighbours
- Separation: a negative score for getting too close to any one neighbour
- Alignment: a positive score for getting closer to the average heading of the neighbours

UNIT – V – ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS – SMRA3003

5 PATTERN RECOGNITION AND LEARNING TECHNIQUES

5.1 Introduction

It was the pre-corona period and I went to a movie theatre. Before the beginning of the movie, the national anthem was played. Everyone voluntarily stood up with respect. I saw some of them singing aloud. One man (maybe from Army) even saluted the flag shown on the screen. A foreigner who was perplexed, nevertheless, slowly stood up observing the audience.

In the above incident, I mentioned various actions (standing, singing, and saluting) of the audience that was triggered by playing a familiar music tune. Let's assume the theatre has 100 audiences and build an imaginary data table of actions.

If I replace the world Action with 'behaviour pattern', or simply 'pattern' you can see that the most common pattern is standing and the rare pattern is saluting. The foreigner, though confused, followed the most common pattern. Notice that our world is full of patterns. Nature is full of patterns.

It's fun to mimic the mannerisms and speech of other people like celebrities. This simply means capturing behaviour and language patterns.
I don't trust WORDS, I even question ACTIONS, but I don't doubt PATTERNS

5.1.1 What is a pattern?

A pattern is some phenomenon that repeats regularly based on a set rule or condition. In my previous blog article, I mentioned that Music is all about melodious patterns.

5.1.2 Human learning and pattern recognition

Learning is the constant disruption of an old pattern, a breakthrough that substitutes something old with something new

Humans and animals learn with the help of the senses. Learning helps in identifying and recognizing patterns around us. The process of pattern recognition involves matching the information received with the information already stored in the brain. Making the connection between memories and information perceived is a step of pattern recognition called identification.

5.1.3 Recognizing Patterns

Pattern recognition requires the repetition of experience. All discoveries and inventions to date are a result of the pattern recognition skills of humans.

Humans have a tendency to see patterns everywhere. They are important when making comparisons, judgments, and acquiring knowledge; we tend to be uneasy with chaos and chance, just like the foreigner who was perplexed in the movie theatre. Figure 5.1

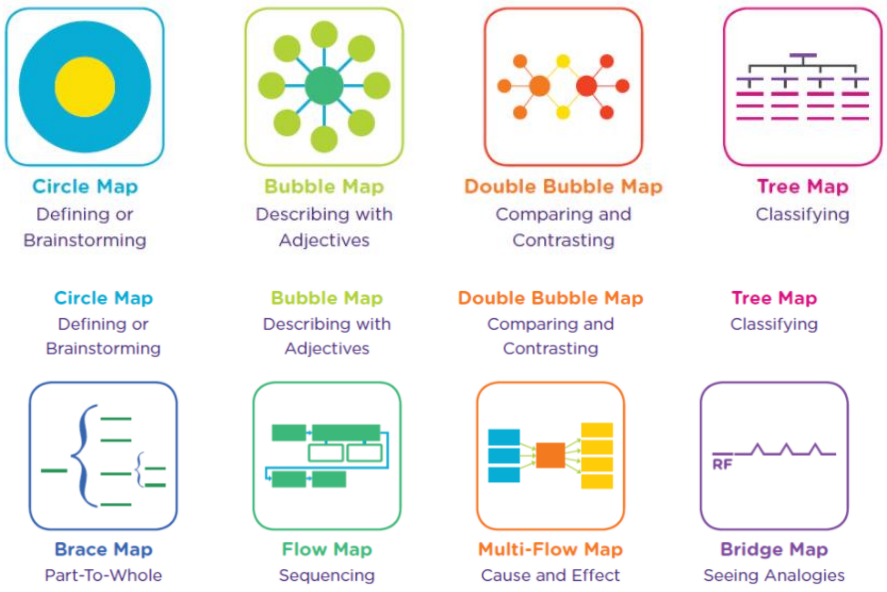


Figure 5.1 Recognizing Patterns

5.1.4 Why patterns?

Finding patterns is the essence of wisdom. Finding patterns is extremely important. Patterns make our task simpler. Finding and understanding patterns is crucial to mathematical thinking and problem-solving.

Let's take the simple example of the sum of numbers from 1 to 10, which is 55.
1+2+3+…+10 = 55
Now, the sum of 11 to 20 is 10×10+55 = 155
And similarly, the sum of 21 to 30 is 20×10+55 = 255.

We can go on with this pattern to do a sum of 10 consecutive numbers. The mathematical formulas are nothing but concise representations of patterns.

5.1.5 Types of patterns

Logic, number, sound, image, and word patterns are all around us. Logic patterns help us classify similar objects, while number patterns help us predict a sequence. Word patterns help us make sense of language. Here is an example of a logic pattern. Image patterns help in classifying information in images. Melodious music patterns can be identified with music sequences. Figure 5.2

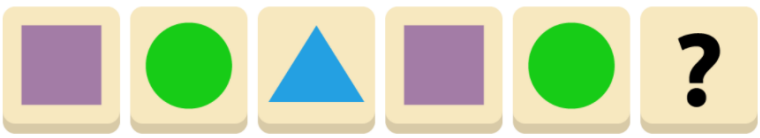


Figure 5.2 Types of Patterns

5.1.6 Design Patterns

Architect Christopher Alexander described first pattern language and design patterns. A Design Pattern is a blueprint that provides a general solution to the similar type of issues you will encounter over and over in any field of activity. Here is an example.

Iterating with the Python "for" loop

Python's for loop abstracts the Iterate Design Pattern so thoroughly that most Python programmers are never even aware of the object design pattern that it enacts beneath the surface. The for loop performs a repeated assignment, running its indented block of code once for each item in the sequence it is iterating over.

5.1.7 Mathematics and Patterns

Mathematics is sometimes called the science of patterns. The most important concept in mathematics is a function. A function is an abstract representation of a pattern. Similarly, every field of activity has patterns.

Here, the function y = f(x) = 3x is shown as number pattern above. The pattern shows values of the sequence, but a function can output any value of the pattern sequence directly.

5.1.8 Data Patterns in Statistics

Graphic displays like histograms in statistics are useful for seeing patterns in data. Patterns in data are commonly described in terms of centre, spread, shape, and unusual features.

Some common distributions have special descriptive labels, such as symmetric, bell-shaped, skewed, etc. This is useful in exploratory data analysis. Probability is used to anticipate the patterns in data Figure 5.3

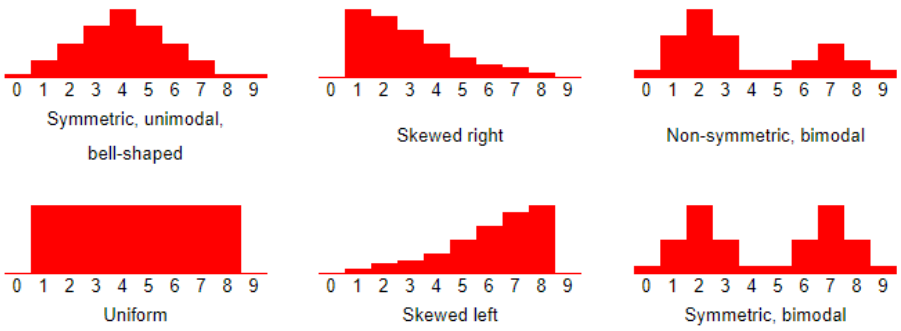


Figure 5.3 Data Patterns

5.1.9 Patterns in Data Mining

Data nowadays is in both structured (database, spreadsheet, etc) and unstructured form (images, documents, etc). Finding relevant data is a big challenge for stakeholders.

Data Mining tools perform data analysis to uncover important data patterns contributing greatly to business strategies and scientific research. Figure 5.4

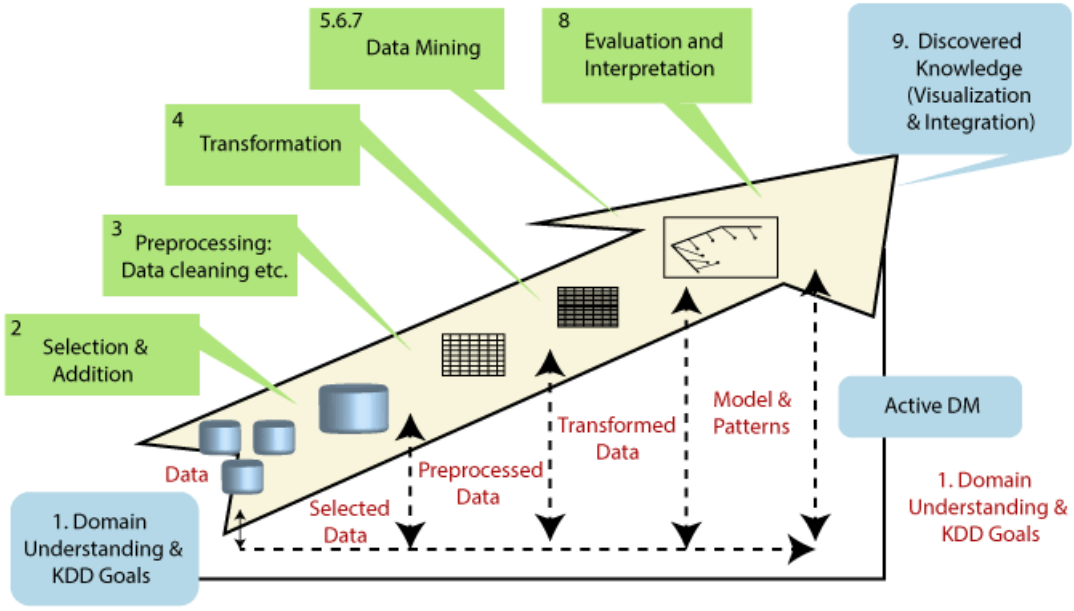


Figure 5.4 Patterns in Data Mining

5.2 Speech Recognition in AI

It is indisputable that speech recognition in AI (artificial intelligence) has come a long way since Bell Laboratories invented The Audrey, a device capable of recognizing a few spoken digits, in 1952.

A recent study by Capgemini demonstrates how ubiquitous speech recognition has become. 74% of consumers sampled reported that they use conversational assistants to research and buy goods and services, create shopping lists, and check order status.

We are all familiar with digital assistants such as Google Assistant, Cortana, Siri, and Alexa. Google Assistant and Siri are used by over 1 billion people globally, and Siri has over 40 million users in the US alone. But, have you ever wondered how these tools understand what you say? Well, they use speech to text AI.

5.2.1 Understanding Speech Recognition

In speech recognition, the computer takes input in the form of sound vibrations. This is done by making use of an analog to digital converter that converts the sound waves into a digital format that the computer can understand.

A series of complex algorithms are then run on the data to recognize the speech and return a text result. Depending on the end-goal, the data may also be converted into another form. For example, Google Voice typing converts spoken words into text while personal assistants like Siri and Google Assistant take the sound input but can also give a voice response. Essentially, you issue a command such as “How is the weather?” and the device responds with an audible answer.

Advanced speech recognition in AI also comprises AI voice recognition where the computer can distinguish a particular speaker’s voice.

5.2.2 Uses of Speech Recognition

Speech recognition technology has been deployed in digital personal assistants, smart speakers, smart homes, and a wide range of products and solutions.

The technology has allowed us to perform a wide range of voice-activated tasks. You can now use your voice to cook a turkey, dim the lights, turn on the stereo, and do a host of other things in the home.

In aviation, Amazon holds a patent for a voice-controlled drone that can change its behavior mid-flight based on voice commands and gestures. Militaries around the world are also experimenting with speech to text in the cockpit. This is to ensure pilots spend more time focusing on the mission as opposed to fiddling with the instruments.

5.2.3 Speech Recognition in Artificial Intelligence

The term “Artificial Intelligence” was first coined by John McCarthy (Dartmouth College), Claude Shannon (Bell Telephone Laboratories), Nathaniel Rochester (IBM), and Marvin Minsky (Harvard University) in a proposal to the Rockefeller Foundation in 1955. Artificial intelligence can be described as human intelligence shown by machines.

It was initially used to analyze and quickly compute data, but it is now used to perform tasks that previously could only be performed by humans.

Artificial intelligence is often confused with machine learning. Machine learning is a derivative of artificial intelligence and refers to the process of teaching a machine to recognize and learn from patterns rather than teaching it rules.

Computers are trained by feeding large volumes of data to an algorithm and then letting it pick out the patterns and learn. In the nascent days of machine learning, programmers had to write code for every object they wanted the computer to recognize – e.g., a cat vs. a human. These days, computers are shown numerous examples of each object. Over time, they learn without any human input.

Speech recognition, natural language processing, and translation use artificial intelligence today. Many speech recognition applications are powered by automatic speech recognition and Natural Language Processing (NLP). Automatic speech recognition refers to the conversion of audio to text, while NLP is processing the text to determine its meaning.

Humans rarely ever speak in a straightforward manner that computers can understand. Normal speech contains accents, colloquialisms, different cadences, emotions, and many other variations. It takes a great deal of natural language analysis to generate accurate text.

With such uptake, it isn’t surprising that the global speech recognition applications market is projected to grow to USD 3,505 million by 2024. Research by Gartner also shows that by 2022, 70% of white-collar workers will use conversational platforms daily.

The development of the Internet of Things (IoT) and big data is going to lead to deeper uptake of speech recognition technology

Jenn is an experienced marketer and Blogger. Product development and product launch projects have offered Jenn working experience with AI and Machine Learning. MarTech and data analysis are at the forefront of her daily activities.

Jenn is currently expanding her knowledge and experience in Cloud Computing and more. She is a fan of the Toronto Maple Leafs and is experienced in managing disappointment and is often heard saying "Next year!"

5.2.4 Challenges with Speech to Text AI

Despite the giant leap forward that AI speech to text has made over the last decade, there remain several challenges that stand in the way of true ubiquity.

The first of these is accuracy. The best applications currently boast a 95% accuracy rate – first achieved by Google Cloud Speech in 2017. Since then, many competitors have made great strides and achieved the same rate of accuracy.

While this is good progress, it means that there will always be a 5% error rate. This may seem like a small figure – and it is, where the issue at hand is a transcript that can be quickly edited by a human to correct errors. But, it is a big deal where voice is used to give a command to the computer. Imagine asking your car's navigator to search the map for a particular location, and it searches for something different and sends you on your way in the wrong direction because it didn't quite catch what you said. Or, imagine asking your smart home's conversational assistant to turn off the lights, but it instead hears a different command and turns off the heating in winter.

Such errors are caused by background noise, heavy accents, unknown dialects, and varied voice pitch in different speakers. The next generation of speech recognition in AI needs to surmount these challenges and attain 100% accuracy in order to reach the last mile of uptake.

The other challenge is that humans don't just listen to each other's voices to understand what is being said. They also observe non-verbal communication to understand what is being communicated but isn't being said. This includes facial expressions, gestures, and body language. So, while computers can hear and understand the content, we are a long way from getting to a point where they can pick up on non-verbal cues. The emotional robot that can hear, feel and interpret like a human is the holy grail of speech recognition.

5.2.5 Speech recognition in AI

In speech recognition, the computer takes input in the form of sound vibrations. This is done by making use of an analog to digital converter that converts the sound waves into a digital format that the computer can understand. Advanced speech recognition in AI also comprises AI voice recognition where the computer can distinguish a particular speaker's voice.

Machine learning is a form of pattern recognition which is basically the idea of training machines to recognize patterns and apply them to practical problems. Machine learning is a feature that can learn from data and iteratively keep updating itself to perform better but, Pattern recognition does not learn problems but, it can be coded to learn patterns.

5.3 Working of Machine Learning Image Processing

Typically, machine learning algorithms have a specific pipeline or steps to learn from data. Let's take a generic example of the same and model a working algorithm for an Image Processing use case.

Firstly, ML algorithms need a considerable amount of high-quality data to learn and predict highly accurate results. Hence, we'll have to make sure the images are well processed, annotated, and generic for ML image processing. This is where Computer Vision (CV) comes into the picture; it's a field concerning machines being able to understand the image data. Using CV, we can process, load, transform and manipulate images for building an ideal dataset for the machine learning algorithm.

For example, say we want to build an algorithm that will predict if a given image has a dog or a cat. For this, we'll need to collect images of dogs and cats and preprocess them using CV. The preprocessing steps include:

- Converting all the images into the same format.
- Cropping the unnecessary regions on images.
- Transforming them into numbers for algorithms to learn from them(array of numbers).

Computers see an input image as an array of pixels, and it depends on the image resolution. Based on the image resolution, it will see height * width * dimension. E.g., An image of a 6 x 6 x 3 array of a matrix of RGB (3 refers to RGB values) and an image of a 4 x 4 x 1 array of a matrix of the grayscale image.

These features (data that's processed) are then used in the next phase: to choose and build a machine-learning algorithm to classify unknown feature vectors given an extensive database of feature vectors whose classifications are known. For this, we'll need to choose an ideal

algorithm; some of the most popular ones include Bayesian Nets, Decision Trees, Genetic Algorithms, Nearest Neighbours and Neural Nets etc. Figure 5.5

Below is a screenshot of classic machine learning image processing workflow for image data:

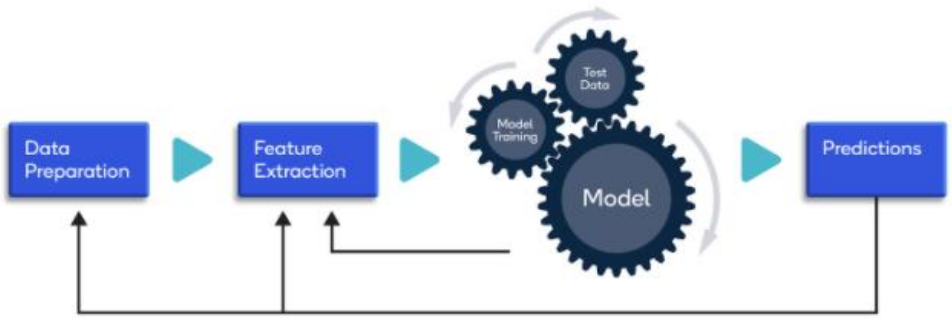


Figure 5.5 Image Processing

5.4 Introduction To Rule Based Systems:

In the past few years, technology has experienced a drastic change. Be it in the field of computer science (CS) or artificial intelligence (AI), new, more advanced research and inventions have given way machines that deliver excellence and help us imitate human knowledge.

However, these new inventions are a result of numerous technologies that work behind the scenes and enable them to deliver services that are on par with us humans. Rule-Based Expert System is one such technology, which is being used extensively across sectors to develop artificial intelligence applications and systems.

Therefore, today, we will explore the concept of the rule-based system (RBS) and try to understand its role in research and development.

5.4.1 What Are Rule-Based Systems (Rbs)?

Present in the heart of automated processes, Rule-Based System technology helps develop knowledge-based systems and applications, that is, intelligent programs and software capable of providing specialized problem-solving expertise in a specific subject by utilizing domain-specific knowledge.

In rule-based systems, knowledge is encoded in the form of facts, goals, and rules and is used to evaluate and manipulate data.

These are, in short, computer systems that use rules to perform a variety of tasks like diagnoses, solve a problem, interpretation, or to determine a course of action in a particular situation. Moreover, these are applied to systems involving human-crafted or curated rule nodes and can be used to perform lexical analysis to compile or interpret computer programs, or in natural language processing.

The rule-based expert systems consist of three important elements:

- Set of Facts: These are assertions or anything relevant to the beginning state of the system
- Set of Rules: It contains all actions that should be taken within the scope of a problem and specify how to act on the assertion set. Here, facts are represented in an IF-THEN form
- Termination Criteria or Interpreter: Determines whether a solution exists or not, as well as when to terminate the process.

5.4.2 Rule-Based System Example:

A domain-specific expert system that uses rules to make deductions or narrow down choices is one of the most popular as well as the classic example of rule-based systems. Furthermore, recent advancement in technology has given way to the development of modern machines and systems like:

- IKEA Virtual Assistant.
- Diagnostics Oriented Rockwell Intelligence System (DORIS)
- Machine for Intelligent Diagnosis (MIND)

5.4.3 Features of Rule-Based Systems:

Widely used in Artificial Intelligence, Rule-Based Expert System is not just only responsible for modeling intelligent behavior in machines and building expert system that outperform human expert(s) but also helps:

- Composed of combined knowledge of human experts in the problem domain.
- Represent knowledge in a highly declarative way.
- Enables the use of several different knowledge representations paradigms.
- Supports implementation of non-deterministic search and control strategies.
- It helps describe fragmentary, ill-structured, heuristic, judgemental knowledge.
- It is robust and can operate with uncertain or incomplete knowledge.
- Helps with rule-based decision making examples monitoring, control, diagnostics, service, etc.

5.4.4 Components of Rule-Based Systems:

The rule-based expert system architecture is an amalgamation of four (4) important components that are focused on different aspects of the problem in hand. From assessing the information to helping machines reach the goal state, these components are integral for the smooth functioning of rule-based systems. These are:

- Rule Base: This is a list of rules that is specific to a type of knowledge base, which can be rule-based vs. model-based, etc.
- Semantic Reasoner: Also known as the inference engine, it infers information or takes necessary actions based on input and the rule base in the knowledge base. Semantic reasoner involves a match-resolve-act cycle, wherein:
- Match: A section of the production rule system is matched with the contents of the working memory to obtain a conflict, which consists of various instances of the satisfied productions.
- Conflict-Resolution: When the production system is matched, one of the production instances in the conflict is chosen for execution, to determine the progress of the process.
- Act: Finally, the production instance executed in the above phase is executed, which impacts the contents of the working memory.
- Working Memory: Stores temporary information or data.
- User Interface: It is the connection to the outside world, input and output signals are sent and received.

Now that we have covered the basics of Rule-Based systems, let us try and answer “What is rule-based approach?”

5.4.5 Construction of Rule-Based Systems:

Before we move on to discuss the types of rule-based systems, we need to understand its construction, as it plays a crucial role in how the system evaluated the information. The construction of rule-based systems is based on a specific type of logic, such as Boolean logic, fuzzy logic, and probabilistic logic and is categorized into:

- Knowledge-Based Approach: It is a knowledge-based construction that follows a traditional engineering approach, which is domain-independent. Here, it is important to acquire requirements as well as necessary knowledge before identifying the relationships between attributes.
- Data-Based Approach: This data-based construction follows a machine learning approach, which, like the earlier approach, is domain-independent. This rule-based approach is subdivided into:
- Supervised Learning.
- Unsupervised Learning.

5.4.6 Types of Rule-Based Systems:

Like expert systems, rule-based systems can also be categorized into:

- Forward Chaining: Also known as data-driven reasoning, forward chaining is a data-driven technique that follows a deductive approach to reach a conclusion.
- Backward Chaining: Often used in formulating plans, backward chaining is an alternative to forward chaining. It is a goal-driven technique that follows an inductive approach or associative reasoning.

5.4.7 Advantages of Rule-Based Systems:

Being one of the core technologies responsible for making machines capable of rule-based learning, rule-based systems offer a range of advantages like:

- Rule-based programming is easy to understand.
- It can be built to represent expert judgment in simple or complicated subjects.

5.5 Machine Learning

With machine learning algorithms, AI was able to develop beyond just performing the tasks it was programmed to do. Before ML entered the mainstream, AI programs were only used to automate low-level tasks in business and enterprise settings.

This included tasks like intelligent automation or simple rule-based classification. This meant that AI algorithms were restricted to only the domain of what they were processed for. However, with machine learning, computers were able to move past doing what they were programmed and began evolving with each iteration.

Machine learning is fundamentally set apart from artificial intelligence, as it has the capability to evolve. Using various programming techniques, machine learning algorithms are able to process large amounts of data and extract useful information. In this way, they can improve upon their previous iterations by learning from the data they are provided.

We cannot talk about machine learning without speaking about big data, one of the most important aspects of machine learning algorithms. Any type of AI is usually dependent on the quality of its dataset for good results, as the field makes use of statistical methods heavily.

Machine learning is no exception, and a good flow of organized, varied data is required for a robust ML solution. In today's online-first world, companies have access to a large amount of data about their customers, usually in the millions. This data, which is both large in the number of data points and the number of fields, is known as big data due to the sheer amount of information it holds.

Big data is time-consuming and difficult to process by human standards, but good quality data is the best fodder to train a machine learning algorithm. The more clean, usable, and machine-readable data there is in a big dataset, the more effective the training of the machine learning algorithm will be.

We cannot talk about machine learning without speaking about big data, one of the most important aspects of machine learning algorithms. Any type of AI is usually dependent on the quality of its dataset for good results, as the field makes use of statistical methods heavily.

5.5.1 Types of Machine Learning.

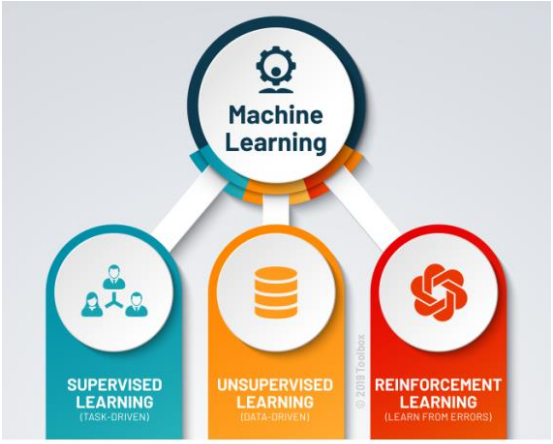


Figure 5.6 Machine Learning Types

5.5.2 Supervised learning

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

5.5.3 Unsupervised learning

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

5.5.3 Reinforcement learning

directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

5.5.4 Application of Machine Learning

Machine learning algorithms are used in circumstances where the solution is required to continue improving post-deployment. The dynamic nature of adaptable machine learning solutions is one of the main selling points for its adoption by companies and organizations across verticals.

Machine learning algorithms and solutions are versatile and can be used as a substitute for medium-skilled human labor given the right circumstances. For example, customer service executives in large B2C companies have now been replaced by natural language processing machine learning algorithms known as chatbots. These chatbots can analyze customer queries and provide support for human customer support executives or deal with the customers directly.

Machine learning algorithms also help to improve user experience and customization for online platforms. Face book, Netflix, Google, and Amazon all use recommendation systems to prevent content glut and provide unique content to individual users based on their likes and dislikes.

Facebook utilizes recommendation engines for its news feed on both Facebook and Instagram, as well as for its advertising services to find relevant leads. Netflix collects user data and recommends various movies and series based on the preferences of the user. Google utilizes machine learning to structure its results and for YouTube's recommendation system, among many other applications. Amazon uses ML to place relevant products in the user's field of view, maximizing conversion rates by recommending products that the user actually wants to buy.

However, as ML continues to be applied in various fields and use-cases, it becomes more important to know the difference between artificial intelligence and machine learning.

Facebook utilizes recommendation engines for its news feed on both Facebook and Instagram, as well as for its advertising services to find relevant leads. Netflix collects user data and recommends various movies and series based on the preferences of the user. Google utilizes machine learning to structure its results and for YouTube's recommendation system, among many other applications

5.6 The Basics of Genetic Algorithms in Machine Learning

A genetic algorithm is a search-based algorithm used for solving optimization problems in machine learning. This algorithm is important because it solves difficult problems that would take a long time to solve. It has been used in various real-life applications such as data centers, electronic circuit design, code-breaking, image processing, and artificial creativity.

This article will take the reader through the basics of this algorithm and explains how it works. It also explains how it has been applied in various fields and highlights some of its limitations.

5.6.1 Genetic algorithm (GA) explained

The following are some of the basic terminologies that can help us to understand genetic algorithms:

- Population: This is a subset of all the probable solutions that can solve the given problem.
- Chromosomes: A chromosome is one of the solutions in the population.
- Gene: This is an element in a chromosome.
- Allele: This is the value given to a gene in a specific chromosome.
- Fitness function: This is a function that uses a specific input to produce an improved output. The solution is used as the input while the output is in the form of solution suitability.
- Genetic operators: In genetic algorithms, the best individuals mate to reproduce an offspring that is better than the parents. Genetic operators are used for changing the genetic composition of this next generation.

A genetic algorithm (GA) is a heuristic search algorithm used to solve search and optimization problems. This algorithm is a subset of evolutionary algorithms, which are used in computation. Genetic algorithms employ the concept of genetics and natural selection to provide solutions to problems.

These algorithms have better intelligence than random search algorithms because they use historical data to take the search to the best performing region within the solution space.

GAs are also based on the behavior of chromosomes and their genetic structure. Every chromosome plays the role of providing a possible solution. The fitness function helps in providing the characteristics of all individuals within the population. The greater the function, the better the solution

5.6.2 Advantages of genetic algorithm

- It has excellent parallel capabilities.
- It can optimize various problems such as discrete functions, multi-objective problems, and continuous functions.
- It provides answers that improve over time.
- A genetic algorithm does not need derivative information.

5.6.3 How genetic algorithms work

Genetic algorithms use the evolutionary generational cycle to produce high-quality solutions. They use various operations that increase or replace the population to provide an improved fit solution.
Genetic algorithms follow the following phases to solve complex optimization problems:

- Initialization

  The genetic algorithm starts by generating an initial population. This initial population consists of all the probable solutions to the given problem. The most popular technique for initialization is the use of random binary strings.

- Fitness assignment

  The fitness function helps in establishing the fitness of all individuals in the population. It assigns a fitness score to every individual, which further determines the probability of being chosen for reproduction. The higher the fitness score, the higher the chances of being chosen for reproduction.

- Selection

  In this phase, individuals are selected for the reproduction of offspring. The selected individuals are then arranged in pairs of two to enhance reproduction. These individuals pass on their genes to the next generation. The main objective of this phase is to establish the region with high chances of generating the best solution to the problem (better than the previous generation). The genetic algorithm uses the fitness proportionate selection technique to ensure that useful solutions are used for recombination.

- Reproduction

  This phase involves the creation of a child population. The algorithm employs variation operators that are applied to the parent population. The two main operators in this phase include crossover and mutation.

  Crossover: This operator swaps the genetic information of two parents to reproduce an offspring. It is performed on parent pairs that are selected randomly to generate a child population of equal size as the parent population.

  Mutation: This operator adds new genetic information to the new child population. This is achieved by flipping some bits in the chromosome. Mutation solves the problem of local minimum and enhances diversification. The following image shows how mutation is done.