Er. Sujan Shrestha

Er. Sujan Shrestha received his MSc. in Computer Systems and Knowledge Engineering from Pulchowk Campus, IOE, TU. With more than five years of teaching experience, he is currently Senior Lecturer in the department of Electronics, Communication and Information Engineering, Kathmandu Engineering College, IOE, TU. He has also published his research papers in globally renowned publications.

Dr. Basanta Joshi

Dr. Basanta Joshi received a Doctor of Engineering from Osaka Sangyo University, Japan in 2013. He did both Bachelor of Electronics and Communication Engineering, and Masters of Science in Information and Communication Engineering from Pulchowk Campus, IOE, TU, Nepal. Currently, he is working as Assistant Professor in the Department of Electronics and Computer Engineering and Program Coordinator for Masters of Science in Information and Communication Engineering, Pulchowk Campus, IOE, TU. He has been involved as IT/Research Consultant in various national and international projects. He has been actively publishing national and international research papers.

Er. Santosh Giri

Er. Santosh Giri received his Masters in Computer Engineering from NCIT, Pokhara University, and currently a Ph.D. scholar at DOECE, Pulchowk Campus, IOE, TU. At present, he is working as Lecturer in the Department of Electronics and Computer Engineering, Pulchowk Campus, IOE, TU. With more than 5 years of teaching experience, he has published several papers in national and international journals.

Er. Shyam Dahal

Er. Shyam Dahal received MSc. in Computer Systems and Knowledge Engineering from Pulchowk Campus, IOE, TU. With more than 10 years of teaching experience, he is currently Associate Professor in the Department of Computer Engineering, Kathmandu Engineering College, IOE, TU. He is the main author of "Insights on Engineering Electromagnetics" and "Insights on Basic Electronics Engineering", and coauthor of "Insights on Microprocessors" and "Insights on Instrumentation II".

'Silence' and 'smile' are two powerful words. 'Smile' is the way to solve many problems and 'silence' is the way to avoid many problems.

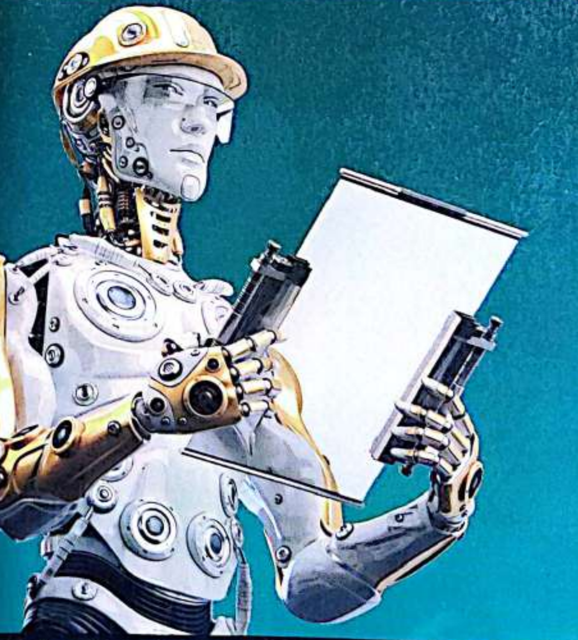
SYSTEM INCEPTION

Price Rs. 350/-

9789937085847

nsights on ARTIFICIAL NTELLIGENCE

B.E. (TU, PU, PoU, KU), BSc.CSIT, BCA, BEIT



. Sujan Shrestha
r. Basanta Joshi
Er. Santosh Giri
Er. Shyam Dahal

Insights on ARTIFICIAL INTELLIGENCE

Published by : System Inception

Authors : Er. Sujan Shrestha
Dr. Basanta Joshi
Er. Santosh Giri
Er. Shyam Dahal

© : Publisher

All rights reserved. This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the authors' prior written consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above, no part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), without the prior written permission of the copyright owner of the book.

First Edition: 2021 AD

Computer : Creation Graphics
Bagbazar, Kathmandu

PREFACE

Insights on Artificial Intelligence is a book about the science of artificial intelligence (AI). AI is the study of the design of intelligent computational agents. The book is structured as a textbook but it is designed to be accessible to a wide audience.

We wrote this book because we are excited about the emergence of AI as an integrated science. Here, concepts of AI are explained with examples. We develop the science of AI together with its engineering applications. We must build the science on solid foundations; we present the foundations, and give some examples of the complexity required to build useful intelligent systems.

AI research is expanding so rapidly now that the volume of potential new text material is vast. However, research teaches us not only what works but also what does not work so well, allowing us to be highly selective. We have included material on machine learning techniques that have proven successful.

The book can be used as an introductory text on artificial intelligence for advanced undergraduate or graduate students in computer science or related disciplines such as computer engineering, philosophy, cognitive science, or psychology. It will appeal more to the technically minded; parts are technically challenging, focusing on learning by doing: designing, building, and implementing systems. Any curious scientifically oriented reader will benefit from studying the book. Previous experience with computational systems is desirable, but prior study of the foundations upon which we build, including logic, probability, calculus, and control theory, is not necessary, because we develop the concepts as required.

The focus is on an intelligent agent acting in an environment. We start with simple agents acting in simple, static environments and gradually increase the power of the agents to cope with more challenging worlds. So, we introduce the simplest agents and then show how to add each of these complexities in a modular way.

Our approach, through the development of the power of the agent's capabilities and representation language, is both simpler and more powerful than the traditional approach of surveying and cataloging various applications of AI.

Remember that this book is not a survey of AI research. We invite you to join us in an intellectual adventure: building a science of intelligent agents.

We are thankful to all the people who had provided their support, encouragement and help directly or indirectly. All the mistakes remaining are ours.

Authors

Scanned with CamScanner

CONTENTS

Scanned with CamScanner

Chapter – 1

ARTIFICIAL INTELLIGENCE

1.1 Introduction

While watching the movie "The Terminator", we might have felt that robots that look amazingly like Arnold Schwarzenegger are going to hunt us down and kill us. That goes without saying. We have seen the movie. Failing that, we are sure that sophisticated hyper-intelligent machines as in the movie "The Matrix" are going to rule the planet and depending upon their mood, either enslave us in the Matrix for use as a power supply or simply exterminate us because we are annoying. Again, we are sure of this because we have seen the movie. Or they might lock us down for our own protection until Will Smith convinces one of them to think for itself and wink when he lies. If we have watched these two movies, we without going deeper into technology, can somehow get a feel of what "artificial intelligence" means. Unsurprisingly, these days, our daily activities are very much flooded with AI technology. Here are few examples to support this:

- When we are using Google search engine and write a search terms, Google makes recommendations to choose from, that is AI in action.
- The feeds that we see in our timeline to the notifications we receive from apps (Twitter, Facebook, Instagram), everything is gcurated with AI.
- When we are using Netflix or Youtube, AI is playing a big role by recommending us movies and music of our taste.
- The video games like Dota 2, PUBG in which we get locked into for many hours have AI inside them.
- Using Google or Apple Maps for finding locations when we apply for "Pathao" service is another beautiful example of AI inbuilt technology.

Scanned with CamScanner

- Siri (Apple), Alexa (Amazon), and Google Assistant (Google) are another AI-powered virtual assistants that use voice queries and a natural-language user interface to answer questions, make recommendations, and perform actions.

Above examples are only few among large areas of AI applications. AI currently encompasses a huge variety of fields. The application areas of AI will be felt as we go detail in the subsequent chapters. Now it is time to define AI.

John McCarthy, who coined the term "artificial intelligence" in 1955, defined artificial intelligence as:

"Artificial intelligence is the science and engineering of making intelligent machines".

Some other popular definitions of artificial intelligence are:

"The exciting new effort to make computers think .... machines with minds, in the full and literal sense."

(Haugeland, 1985)

"The art of creating machines that perform functions that require intelligence when performed by people."

(Kurzwell, 1990)

"AI is the study of how to make computers do things which at the moment, people do better." *(Rich and Knight, 1991)*

In general, artificial intelligence (AI) is a branch of Computer Science which deals with helping machines find solutions to complex problems in a more human-like fashion. This generally involves borrowing characteristics from human intelligence, and applying them as algorithms in a computer friendly way.

1.2 Brief History of Artificial Intelligence

The development of AI can be understood under the following headings:

1. **The gestation of artificial intelligence (1943-1955)**
   - The first work on AI was done by Warren McCulloch and Walter Pitts (1943). They drew on three sources: Knowledge of the basic physiology and function of neurons in the brain; a formal analysis of propositional logic due to Russel and Whitehead; and Turing's theory of computation. They showed for example, that any computable function could be computed by some network of connected neurons, and that all the logical connectives (and, or, not, etc.) could be implemented by simple net structures. McCulloch and Pits also suggested that suitably defined networks could learn.
   - In 1947, Alan Turing published an article "Computing Machinery and Intelligence" where he introduced the Turing Test, machine learning, genetic algorithms, and reinforcement learning.
   - In 1949, Donald Hebb demonstrated a simple updating rule (now known as *Hebbian learning*) for modifying the connection strengths between neurons.
   - In 1950, two graduates (Minsky and Edmonds) from Princeton college built the first neural network computer which was named "SNARC".
2. **The birth of artificial intelligence (1956)**
   - In 1956, McCarthy, Minsky, Claude Shannon, and Rochester organized a two-month workshop at Dartmouth college in Hanover, New Hampshire which brought together U.S. researchers interested in automata theory, neural networks, and the study of intelligence. The objective of the workshop was to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. There were altogether 10 attendees.

80 50

Scanned with CamScanner

- John McCarthy first coined the term "artificial intelligence" at the Dartmouth conference. He defined AI as the science and engineering of making intelligent machines.

3. **Early enthusiasm, great expectations (1952–1969)**
   - Newell and Simon developed General Problem Solver (GPS). This program could imitate human problem-solving protocols, but could only handle a limited class of puzzles.
   - In 1952, Arthur Samuel wrote a series of programs for checkers (draughts) that eventually learned to play at a strong amateur level.
   - In 1958 at MIT AI lab Memo No. 1, McCarthy defined the high-level AI programming language "Lisp". In the same year, McCarthy published a paper entitled "Programs with Common Sense" in which he described the Advice Taker, a hypothetical program that can be seen as the first complete AI system.
   - In 1959, Herbert Gelernter developed Geometry Theorem Prover which could prove theorems using explicitly represented axioms.
   - In 1963, McCarthy started the AI lab at Stanford.
   - In 1965, J. A. Robinson discovered resolution method which was a complete theorem-proving algorithm for first-order logic.
   - In 1963, James Slagle developed SAINT program which was able to solve closed-form calculus integration problems typical of first-year college courses.
   - In 1967, Daniel Bobrow developed STUDENT program which solved algebra story problems, such as the following:

   If the number of customers Rabindra gets is twice the square of 10% of the number of advertisements he runs, and the number of advertisements he runs is 50, what is the number of customers Rabindra gets?

   - Tom Evans' ANALOGY program (1968) solved geometry analogy problems that appear in IQ tests.
   - Hebb's learning methods were enhanced by Bernie Widrow (Widrow and Hoff, 1960; Widrow, 1962), who called his networks "adalines", and by Franke Rosenblatt (1962) with his "perceptrons".

4. **A dose of reality (1966 – 1973)**

   In 1957, Simon made concrete predictions that within 10 years, a computer would be a chess champion, and a significant mathematical theorem prover. These predictions turned out into reality within 40 years rather than 10. There were several reasons behind this:

   - The first reason was that most early programs knew nothing of their subject matter, they succeeded by means of simple syntactic manipulations. So, there occurred problems in early machine translation efforts.
   - The second reason was the intractability of many of the problems that AI was attempting to solve. Most of the early AI programs solved problems by trying out different combinations of steps until the solution was found. This strategy worked initially but later failed in many situations.
   - The third reason was that there were some fundamental limitations on the basic structure being used to generate intelligent behavior.

5. **Knowledge-based systems: The key to power? (1969 – 1979)**

   In this period, a more powerful, domain-specific knowledge was used that allowed larger reasoning steps.

Scanned with CamScanner

- The DENDRAL program (Buchanan et al., 1969) was developed at Stanford, and was the first successful knowledge-intensive system: its expertise derived from large number of special-purpose rules. The program was used to solve the problem of inferring molecular structure from the information provided by a mass spectrometer.
- Feigenbaum, Buchanan, and Dr. Edward Shortlife developed MYCIN to diagnose blood infections. With about 450 rules, MYCIN was able to perform as well as some experts.
- Winograd developed SHRDLU system for understanding natural language and was designed specifically for one area – the blocks world.
- At Yale, Roger Schank and his students built a series of programs that all had the task of understanding natural language. The emphasis was less on language in itself and more on the problems of representing and reasoning with the knowledge required for language understanding.
- A large number of different representation and reasoning languages were developed. Some were based on logic – The Prolog language (popular in Europe), and the PLANNER (popular in the United States). Others, following Minsky's idea of frames (1975), adopted a more structured approach.

6. **AI becomes an industry (1980 – present)**

- The first successful commercial expert system, R1 began operation at the Digital Equipment Corporation (McDermott, 1982). The program helped configure orders for new computer systems; by 1986, it was saving the company an estimated $40 million a year.

- Nearly every major U.S. corporation had its own AI group and was either using or investigating expert systems.
- In 1981, Japan announced the "Fifth Generation" project to build intelligent computers running Prolog. In response, the United States formed the Microelectronics and Computer Technology Corporation (MCC) as a research consortium.

Overall, the AI industry bloomed from a few million dollars in 1980 to billions of dollars in 1988, including hundreds of companies building expert systems, vision systems, robots, and software and hardware specialized for these purposes.

7. **The return of neural networks**

- In the mid-1980s, back-propagation learning was reinvented. The algorithm was applied to many learning problems in computer science and psychology.
- Modern neural network research has bifurcated into two fields, one concerned with creating effective network architectures and algorithms and understanding their mathematical properties, the other concerned with careful modeling of the empirical properties of actual neurons and ensembles of neurons.

8. **AI adopts the scientific method (1987 – present)**

AI has finally come firmly under the scientific method. It is now possible to replicate experiments by using shared repositories of test data and code.

- Approaches based on "hidden Markov models (HMMs)" have come to dominate the field of speech recognition. HMMs were based on a rigorous mathematical theory, and are generated by a process of training on a large corpus of real speech data.
- Machine translation follows the same course as speech recognition.

Scanned with CamScanner

- Neural networks also fit this trend. As neural networks have started using improved methodology and theoretical frameworks, "data mining" technology has spawned a vigorous new industry.
- The "Bayesian network" formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge.
- Similar gentle revolutions have occurred in robotics, computer vision, and knowledge representation.

9. **The emergence of intelligent agents (1995 – present)**
   - One of the most important environments for intelligent agents is the Internet. AI systems have become more common in Web-based applications. Moreover, AI technologies underlie many Internet tools such as search engines, recommender systems, and Web site aggregators.
   - The first conference on "Artificial General Intelligence (AGI)" was held in 2008. AGI looks for a universal algorithm for learning and acting in any environment.

10. **The availability of very large data sets (2001 – present)**
    - More emphasis on data than algorithm.
    - Increasing availability of very large data sources: for example, trillions of words of English and billions of images from the Web; billions of base pairs of genomic sequences.
    - Today, many thousands of AI applications are deeply embedded in the infrastructure of every industry.

1.3 Different Types of AI/ Approach of AI

AI can be understood as the human-like intelligence exhibited by a machine. The field of AI is inter-disciplinary in which several sciences and profession converse such as computer

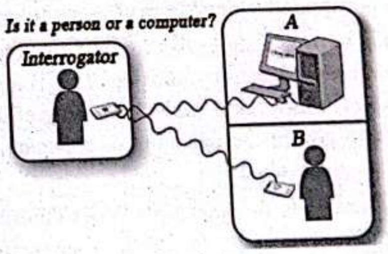
science, psychology, linguistics, neuroscience etc. AI can be defined in terms of the following four categories i.e., four views.

i. **Acting like human/acting humanly: Turing test approach**

The art of creating machines that perform functions which require intelligence when performed by people is understood as acting like a human. This approach is also called a *Turing test*. Turing test, purposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence. The Turing test measures the performance of intelligence machine against that of a human being.

The Turing test is a method for determining whether or not a computer is capable of thinking like a human. According to this test, a computer is deemed to have artificial intelligence if it can mimic human responses under specific conditions.

Consider the following scenario. There are two rooms A and B. One of the rooms contains a computer, and the other room contains a human. The interrogator is outside and does not know which room has a computer. He can ask questions through a teletype and receives answers from both rooms A and B. The interrogator needs to identify whether a human is in room A or in room B. To pass the Turing test, the computer has to fool the interrogator into believing that it is human.



The factors required to pass the Turing test are:

- **Natural language processing (NLP):** To enable it to communicate easier and successfully.
- **Knowledge representation:** To store information what it knows.
- **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.

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

This refers to the automation of activities that we associate with human thinking activities such as decision making, problem-solving, learning, etc. This phenomenon is also known as the *cognitive-based approach* in which the focus is not just on the behaviour and the input, but the output also looks at the reasoning process. Cognitive science brings together computer model forms AI and experimental technique from psychology to try to construct precise and testable theories of the working of the human mind.

This approach requires understanding how human thinks. A different way to determine how human thinks like experience, investigation, inspection, current perception etc.

E.g., General Problem Solver

iii. **Thinking rationally: The "laws of thought" approach**

The study of computations that make it possible to perceive reason and act is meant by thinking. This is also known as the *"laws of though" approach*. These laws of thought were supposed to govern the operation of the mind; their study initiated the field called logic.

A system is said to be rational if it does the right thing for what it knows. It attempts to codify the normative premises in term of logic, deriving the result as a right thinking. Logic provides precise notations or statements about all kinds of things and relations between them.

E.g., For the premises, All men are students. Ram is a man. The result is: Ram is a student.

iv. **Acting rationally: The rational agent approach**

Computational intelligence is the study of the design of an intelligence agent. This model is also known as the rational agent approach. A *rational agent* is one that acts so to achieve the best outcome or when there is uncertainty, the best-expected outcome.

A rational agent is more general than the laws of thought approach, which emphasizes correct interference. Making correct interference is sometimes part of being a rational agent because one way to act rationally is to reason logically to the conclusion that a givers action will achieve one's goals and then act on that conclusion.

E.g., reflex agent.

1.4 AI and Related Fields

i. **Logical AI**

A program knows about the world, in general, the facts of the specific situation in which it must act, and its goals are all represented by sentences of some mathematical, logical language. The program decides what to do by inferring that certain actions are appropriate for achieving its goals.

ii. **Search**

AI programs often examine large numbers of possibilities, e.g. moves in a chess game or inferences by a theorem proving program. Discoveries are continually made about how to do this more efficiently in various domains.

iii. **Pattern recognition**

When a program makes observations of some kind, it is often programmed to compare what it sees with a pattern. For example, a vision program may try to match a pattern of eyes and a nose in a scene to find a face. More complex patterns, e.g., in a natural language text, in a chess position, or the history of some event are also studied.

Scanned with CamScanner

iv. **Representation**

Facts about the world must be represented in some way. Usually, languages of mathematical logic are used.

v. **Inference**

From some facts, others can be inferred. The mathematical-logical deduction is adequate for some purposes, but new methods of non-monotonic inference have been added to logic since the 1970s. The simplest kind of non-monotonic reasoning is default reasoning in which a conclusion is to be inferred by default, but the conclusion can be withdrawn if there is evidence to the contrary. For example, when we hear of a bird, we can infer that it can fly, but this conclusion can be reversed when we hear that it is a penguin. It is the possibility that a conclusion may have to be withdrawn that constitutes the non-monotonic character of the reasoning. Ordinary logical reasoning is monotonic in that the set of conclusions that can the drawn from a set of premises is a monotonic increasing function of the premises.

vi. **Learning from experience**

Programs do that. The approaches to AI based on connectionism and neural nets specialize in that. There is also learning of laws expressed in logic. Programs can only learn what facts or behaviors their formalisms can represent, and unfortunately learning systems are almost all based on very limited abilities to represent information.

vii. **Ontology**

Ontology is the study of the kinds of things that exist. In AI, the programs and sentences deal with various kinds of objects, and we study what these are and what their basic properties are. Emphasis on ontology begins in the 1990s.

1.5 Importance and Applications of Artificial Intelligence

We are a privileged generation to live in this era full of technological advancements. Gone are the days when almost everything was done manually, and now we live in a time where a lot of work is taken over by machines, software, and various automatic processes. In this regard, artificial intelligence has a

special place in all the advancement made today. Artificial intelligence or AI is nothing but the science of computers and machines developing intelligence like humans. In this technology, the machines can do some of the simples to complex stuff that humans need to do regularly. The artificial intelligence applications help to get the work done faster and with accurate results. Error-free and efficient worlds are the main motives behind artificial intelligence.

A concise answer to what AI can do today is difficult because there are so many activities in so many subfields. There are numerous real-world applications of AI which signifies how important an AI is in today's world. Following are the applications which explains the importance of artificial intelligence in today's world:

i. Game playing
ii. Speech recognition
iii. Natural Language Processing
iv. Computer vision
v. Expert systems
vi. Heuristic classification
viii. Consumer marketing
ix. Intrusion detection
x. Autonomous planning and scheduling
xi. Medical diagnosis
xii. Media and entertainment
xiii. Scientific research
xiv. Finance

i. **Game playing**

These days, we can buy machines that can play master level games for a few hundred dollars. Some of the popular games where AI is extensively used are chess, FIFA, PES, PUBG, etc. Some great examples where AI machines had beaten genius player are:

Scanned with CamScanner

- The most famous battle was between "Deep Blue" and world chess champion, Garry Kasparov in 1997.
- The DeepMind system, developed by Google beat the world champion, Ke Jie in the game "The Go" in May 2017 to claim the title of the world's best player.

ii. **Speech recognition**

Before computers were instructed to perform certain tasks through keyboard and mouse. Nowadays, it has become possible to instruct computers through speech.

In the 1990s, United Airlines has replaced its keyboard tree for flight information by a system using speech recognition of flight numbers and city names, which was quite convenient.

AI systems where speech recognition is employed are Amazon Echo, Apple's Siri, Google Assistant, etc.

iii. **Natural Language Processing**

Some examples of AI in NLP are:

- **Skype translator**

  It offers on-the-fly translation to interpret live speech in real time.
- Optical character recognition

  It converts a written or printed text into data.

iv. **Computer vision**

Some examples of AI in computer vision are:

- **Entertainment:**

  Snapchat users love to overlay rabbit ears and fairy dust, for instance, on the images of friends but Such a simple activity actually relies on computer vision algorithms.

- **Banks**

  Around the world now use computer vision to deposit checks remotely. Banking customers take a photo of a paper check with their mobile device. Computer vision software in the banking app captures the image of the check destined for deposit in the bank, then verifies if the signature on the check is genuine. Funds typically become available for use within a business day of verification.
- Medical image analysis
- Manufacturing industries
- Marts

v. **Expert systems**

Expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning through bodies of knowledge, represented mainly as if-then rules. One of the first expert systems was MYCIN developed in 1974, which diagnosed bacterial infections of the blood and suggested treatments. It did better than medical students or practicing doctors.

We will discuss expert systems in detail in Chapter 7.

vi. **Heuristic classification**

It is any approach to problem solving that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals. For example, advising whether to accept a proposed credit card purchase or not can be done using heuristic classification. Information available about the owner of the credit card, his record of payment and the item he is buying are the parameters used for purchasing the credit card.

Scanned with CamScanner

vii. **Consumer marketing**

While using any kind of credit card or debit card during shopping, we have very likely been "input" to an AI algorithm. All this information is recorded digitally. Companies like Nielsen gather this information weekly and search for patterns general changes in consumer behaviour, tracking responses to new products, identifying customer segment (targeted marketing), e.g., they find out that consumers with sports cars who buy textbooks respond well to offers of new credit cards. Algorithms ("data mining") search data for patterns based on mathematical theories of learning.

viii. **Intrusion detection**

Network intrusion has become the biggest concern of this generation. A lot of data is stored on commercial and personal computers. Protection and confidentiality of such information is very crucial in organizations. The detection in such data can be done using artificial intelligence.

ix. **Autonomous planning and scheduling**

Automated planning and scheduling, sometimes denoted as simply AI planning, is a branch of artificial intelligence that concerns the realization of strategies or action sequences, typically for execution by intelligent agents, autonomous robots and unmanned vehicles. Unlike classical control and classification problems, the solutions are complex and must be discovered and optimized in multidimensional space. Planning is also related to decision theory.

x. **Medical diagnosis**

AI systems for diagnosis uses deep learning techniques to arrive at a diagnosis. These systems source image data and symptoms data from healthcare facilities to train on. Then, the systems apply the techniques to arrive at the diagnosis. The problem with deep learning techniques is transparency. Although the inputs and the outputs to the AI systems are

transparent, the way that the AI systems arrive at the diagnosis decision is unclear. The system is also dependent on the quality of the data to ensure accuracy.

xi. **Media and entertainment**

The maximum of artificial intelligence use cases in the media and entertainment domain can be categorized into four segments:

- **Advertising and marketing:** Artificial intelligence is aimed at assisting with the development of designing the promotion and advertising collaterals and the creation of film trailers.
- **Customer experience personalization:** Artificial intelligence apps are used by the entertainment providers for offering personalized content on the basis of users' data that is collected from their website behavior and activities.
- **Search friendliness:** In terms of search optimization, AI is used for increasing the efficiency and speed of the search and classification processes.
- **Immersive experiences through AR/VR:** An active implementation of artificial intelligence for enhancing the AR and VR takes the customer experience to the next level.

xii. **Scientific research**

A combination of human intellect and AI's deep learning technique, which refers to the learning approach of AI that emulates human intellect, can help us tap previously explored areas. New machine learning methods are tackling an almost-endless realm of options—like all the possible mutations in human DNA. Thus, the automation of science could make it possible to run large experiments competently. AI is also being used by pharmaceutical companies "to extract information from academic papers and other written

Scanned with CamScanner

materials, which can surface new hypotheses to test." This can lead to new discovery of newer and more potent drugs, treatment methods, etc. AI, thus, manifests the potential to help researchers reach out into areas that have been perceived as challenging.

xiii. **Finance**

Artificial intelligence in finance is transforming the way we interact with money. AI is helping the financial industry to streamline and optimize processes ranging from credit decisions to quantitative trading and financial risk management. Artificial intelligence has given the world of banking and the financial industry as a whole a way to meet the demands of customers who want smarter, more convenient, safer ways to access, spend, save and invest their money.

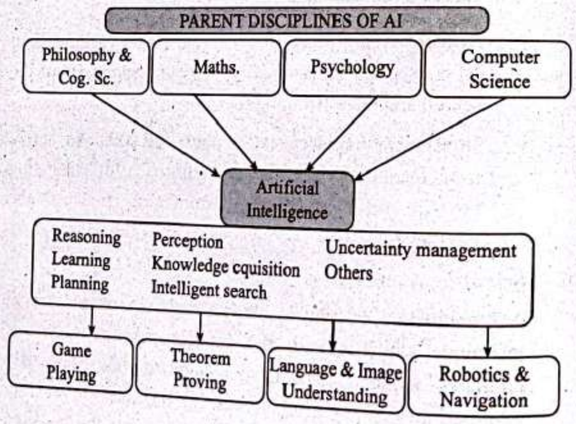


Figure 1.2: Application areas of AI

1.6 Definition and Importance of Knowledge and Learning

1.6.1 Data

Data are streams of raw facts representing events before they have been arranged into a form that people can understand and use.

1.6.2 Information

Information is that property of data, which represents and measures the effects of processing them. By information, we mean data that have been shaped into a form that is meaningful and useful to human beings.

1.6.3 Knowledge

i. **Knowledge definition**

Knowledge is understood as the fact or condition of knowing something with familiarity gained through experience or association. It is something that we come to know by seeing, hearing, touching, feeling, and tasting. It can be referred to as the information that we receive or acquire through any medium. It gives us the power to explore things and to take decisions accordingly. To solve many problems, it requires much knowledge, and this knowledge must be represented or stored in trusted devices.

ii. **Knowledge storing**

To acquire knowledge, we need to understand it first. We learn and gain knowledge in our own preferred language. Every human or domain has their own way of understanding the knowledge and stores it as per requirement. It is a natural language for people to understand knowledge.

Symbols for computer: The knowledge in computer is stored as a number or character string which represents an object or idea. (internal representation of the knowledge)

80 So

The core concepts: It means the mapping from facts to an internal computer representation and to a form that people can understand.

iii. **Knowledge representation**

There is an unlimited amount of knowledge in the whole world which holds meaning and is valuable in their own way. The important features of intelligence are to create knowledge from data.

Knowledge can be of various types:

- It can be a simple factor in a complex relationship.
- Mathematical formulas or rules for natural language syntax also represents knowledge.
- Associations between related concepts provide knowledge, which helps a lot to solve many problems.
- Inheritance hierarchies between classes of objects have made concepts of programming easily.

Knowledge is more general, which means it may be applied to situations we have not been programmed for it.

Importance of Knowledge

Sufficient amount of knowledge introduces to intelligence. If we have enough knowledge, then we can achieve intelligence. Knowledge plays a major role in building intelligence systems. Not only that, knowledge is very important, even in our daily lives. It makes us superior and gives us wisdom.

1.6.4 Learning

In simple terms, *learning* refers to the cognitive process of acquiring skill or knowledge. It is making a useful change in our minds. Learning can simply be understood as the process or phase of gaining knowledge or skills. Learning means of construct or modifying the representation of what is being experienced.

According to Herbert Simon (1983), "Learning is the phenomenon of knowledge acquisition in the absence of explicit programming". Learning denotes change in the system that is adaptive in such a way that they enable the system to do the same task or task which are drawn from the same population more efficiently and more effectively next time.

Learning basically involves three factors, and they are presented below :

- **Changes**: Learning changes leaner. For machine learning, the problem is determining the nature of the following changes and how to represent them in the best form.
- **Generalization**: Learning leads to generalization where performance must improve not only on the same task but on similar tasks.
- **Improvement:** Learning leads to improvements. Machine learning must address the possibility that the changes may degrade in term of performance and find a way to prevent it.

1.7 Intelligent Agents

1.7.1 Agents

An *agent* is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through the actuator.

- **Human agent:** Eyes, ears and other organs for sensors, hands, legs, mouth and another body part for actuators.
- **Robotic agent:** Cameras and infrared range finder for sensors; various motor for the actuator.

Scanned with CamScanner

Agents and Environments

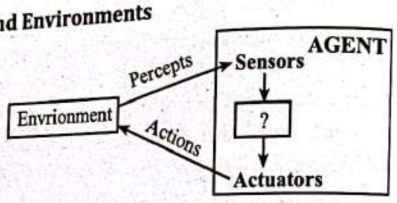


Figure 1.2: Agents interacting with the environment.

The term "percept" refers to the agent's perceptual input at any given instant, that is, location and state of the environment. An agent's percept sequence is the complete history of everything the agent has ever perceived.

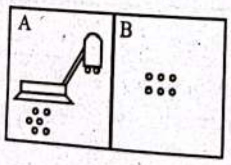
The agent function maps from percept histories to actions.

[f: P* → A]

The agent program runs on the physical architecture to produce f,

Agent = architecture + program

e.g., Vacuum-cleaner world



Percept sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck

1.7.2 Rational Agents

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, based on the evidence provided by the percept sequence and whatever built-in knowledge the agent has. Performance measure is an objective criterion for the success of an agent's behaviour. E.g., a performance measure of a vacuum-cleaner agent could be the amount of dirt cleaned up, amount of time taken, amount of electricity, amount of noise generated etc.

Rationality is distinct form omniscience. An omniscient agent knows the actual outcome of its actions and can act accordingly, but omniscience is impossible. Whereas, rationality maximizes expected performance.

Agents can act to modify future percepts to obtain useful information (information gathering, exploration). Agent is autonomous if its behaviour is determined by its own percepts and experiences (with the ability to learn and adapt) without depending solely on built-in knowledge.

In our discussion of the rationality of the simple vacuum-cleaner agent, we had to specify the performance measure, environment and agent's actuator and sensors. We will group all these together under the task environment.

Before we design an intelligent agent, we must specify its task environment (PEAS).

▪ Performance measure	**P**
▪ Environment	**E**
▪ Actuators	**A**
▪ Sensors	**S**

E.g., Agent: Taxi driver

- **Performance measure:** Safe, fast, legal, comfortable trip, maximise profits
- **Environment:** Roads, other traffic, customers

Scanned with CamScanner

- **Actuators:** Steering wheel, accelerator, brake, signal, horn
- **Sensors:** Camera, GPS, speedometer, engine sensor, keyboard.

Table 1.1: Examples of agents and their PEAS description

Agent Type	P	E	A	S
Medical Diagnosis System	Healthy patient, minimize costs.	Patient, Hospital staff	Display questions, tests, diagnoses, treatment, referrals.	Keyboard entry of symptoms, findings, patient's answers.
Interactive English Tutor	Maximize student's score on test.	Set of students testing.	Display exercises, suggestions.	Typed words.
Satellite Image Analysis System	Correct categorization	Downlink from orbiting satellite	Display categorization of scene.	Colour pixel arrays.
Refinery Controller	Maximize purity, yield, safety	Refinery Operators	Valves, pumps, heaters, display.	Temperature, pressure, chemical sensors
Taxi Driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrian, customers	Steering, accelerator, brake, signal, horn, display.	Cameras. Sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard.

1.7.3 Agent Types

There are five basic types of agents in order of increasing generality:

1. **Table-driven agents:** It uses a percept sequence/action table in memory to find the next action. They are implemented by a (large) lookup table.

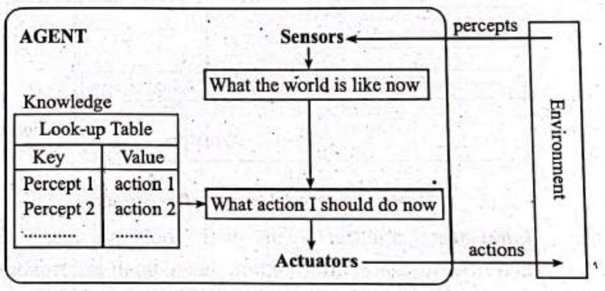


Figure 1.3: Table-driven agent

2. **Simple reflex agents:** Agents are based on condition-action rules, implemented with an appropriate production system. They are stateless devices which do not have a memory of past world states.

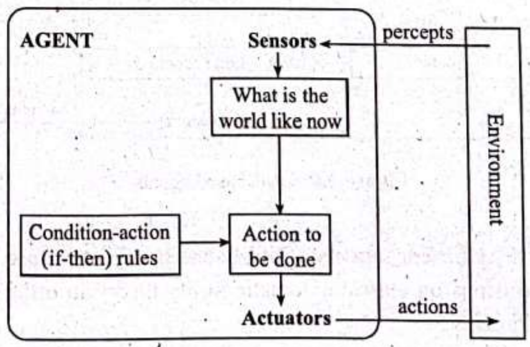


Figure 1.4: Simple reflex agents

Scanned with CamScanner

3. **Model-based reflex agents:** Agents with memory have an internal state, which is used to keep track of past states of the world.

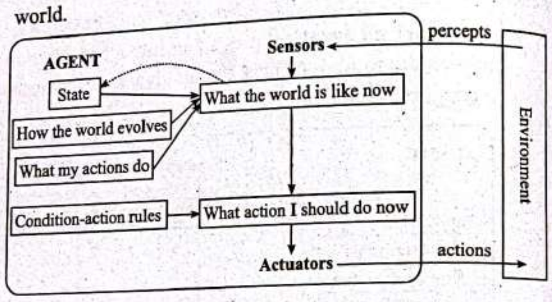


Figure 1.5: Model-based reflex agents

4. **Goal-based agents:** Agents with goals are agents that, in addition to state information, have goal information that describes desirable situations. Agents of this kind consider future events.

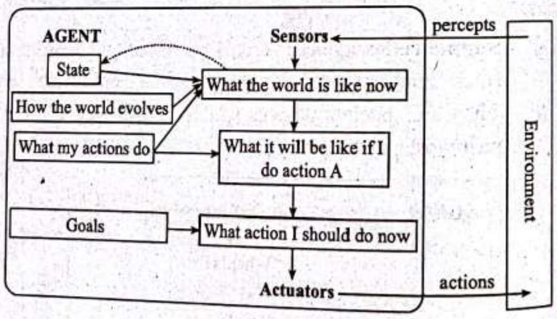
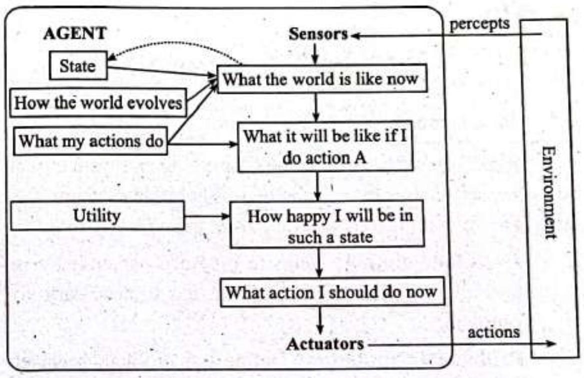


Figure 1.6: Goal-based agents

5. **Utility-based agents:** Utility-based agents base their decisions on classic axiomatic utility theory in order to act rationally.



AGENT
Sensors
percepts
State
What the world is like now
How the world evolves
What my actions do
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Actuators
actions
Environment

Figure 1.7: Utility-based agents

EXERCISE

1. Describe four views of artificial intelligence in details.
2. What is Turing test and total Turing test? Discuss any two fields of your daily life where artificial Intelligence has been applied.
3. Discuss brief history of AI with chronological development.
4. If the turning test is passed, does this show that computer exhibit intelligence? State your reasons.
5. What is an intelligent agent? How does learning agent work?
6. Explain different types of AI agents.

Chapter 2

PROBLEM SOLVING

2.1 Introduction

Problem solving is a goal-based agent system that finds sequence of actions that lead to desirable states from the initial state. The four steps of problem solving are:

i. **Goal formulation:** Helps to organize behavior by isolating and representing the task knowledge necessary to solve problem.

ii. **Problem formulation:** Define the problem precisely with initial states, final state and acceptable solutions.

iii. **Searching:** Find the most appropriate techniques of sequence among all possible techniques.

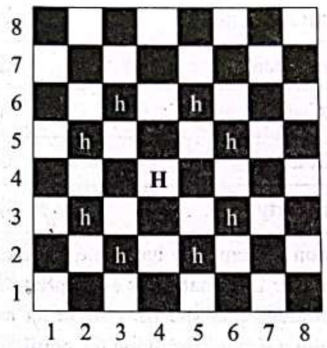
iv. **Execution:** Once the search algorithm returns a solution to the problem, the solution is then executed by the agent.

2.1.1 State Space Representation

A *state space* essentially consists of a set of nodes representing each state of the problem, arcs between nodes representing the legal moves from one state to another, an initial state and a goal state.

A problem is defined as *well-defined problem* if the problem is composed of initial state, actions (using successor function), goal test (to determine the goal state) and path cost. The actions and rules should be defined in as general way as possible. If the specific rules are made, the rule set becomes very large.

E.g., Chess world



Let the current position of horse: (4,4)

From rules of chess for moving a horse, next possible position can be any of the position: (2,3), (2,5), (3,2), (3,6), (5,2), (5,6), (6,3) and (6,5).

Current position of the horse can be any other location, we have to define another rule to find next possible position. By doing so, the rule set becomes very large. Therefore, specific pattern should be described for each object/item of the game. For above case, for moving a horse, next possible position is: *current position +/– ((2 vertical + 1 horizontal) or (1 vertical + 2 horizontal)) position.*

When a problem is defined with all its states, it is said to be a *complete state space.*

2.2 Production System

A *production system* consists of a set of rules, each consisting of a left-hand side (pattern) that determines the applicability of rules and a right side that describes the operation to be performed if the rule is applied. E.g.,vacuum robot

Pattern ⇒ Action (Operation)

Percept sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck

A production system may have one or more knowledgebase that contain whatever information is appropriate for the particular task. A control strategy that specifies the order in which the rules will be selected and a way of resolving the conflicts that arise when several rules matched at once i.e. it must have a Rule Applier for conflict resolution. The first requirement of a good control strategy is that it must cause motion. The second requirement of a good control strategy is that it should be systematic.

2.2.1 Problem Classification

i. **Ignorable:** Intermediate actions can be ignored. E.g., Water-jug problem.

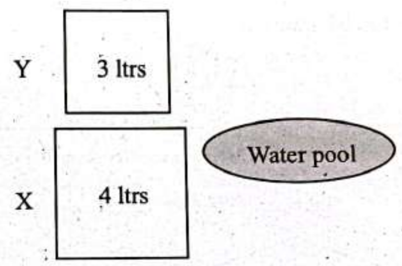
ii. **Recoverable:** The actions can be implemented to go the initial state. E.g., 8-puzzle game.

iii. **Irrecoverable:** The actions cannot help to reach the precious state. E.g., Tic-tac-toe.

iv. **Decomposable:** The problem can be broken into similar ones. E.g., Word puzzle game.

Problem 2.1

You are given two unlabeled empty water jugs X, Y that can hold 4 ltrs and 3 ltrs of water respectively. Now fill the water jug X with exactly 2 ltrs keeping jug Y empty from the water pool.



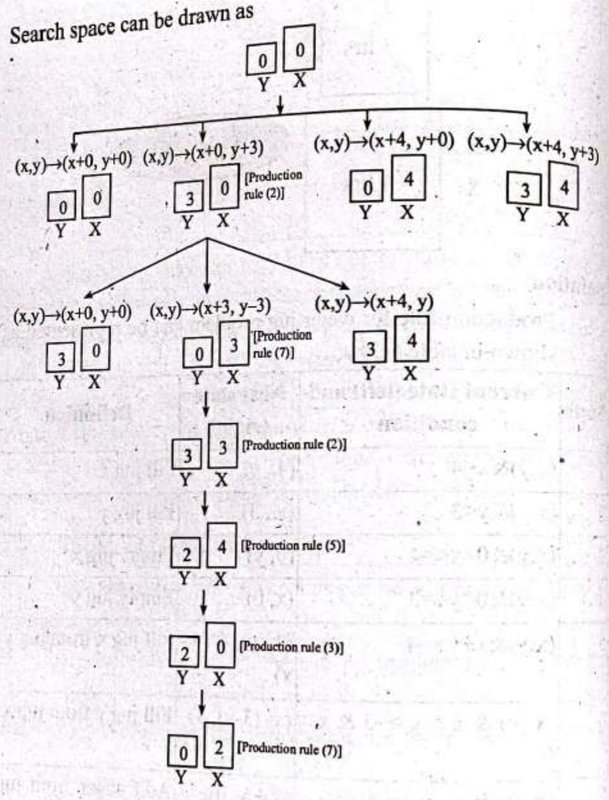
Solution:

Production rule for water jug problem can be represented as shown in table below:

State	Current state (left) and condition	Next state (right)	Definition
1	(x,y)&x<4	(4 , y)	Fill jug x
2	(x,y)&y<3	(x , 3)	Fill jug y
3	(x,y)&0<x<=4	(0, y)	Empty jug x
4	(x,y)&0<y<=3	(x, 0)	Empty jug y
5	(x,y)&x+y>=4	(4, y) – (4-x)	Fill jug x from jug y
6	(x, y) & x + y >=3 & x > 0	(x- (3-y), 3)	Fill jug y from jug x
7	(x, y) & x +y <= 4	(x + y, 0)	Add water from jug y to jug x
8	(x, y) & x + y <= 3	(0, x + y)	Add water from jug x to jug y

Scanned with CamScanner

Search space can be drawn as


0 0
Y X
(x,y)→(x+0, y+0)
(x,y)→(x+0, y+3)
(x,y)→(x+4, y+0)
(x,y)→(x+4, y+3)
0 0 Y X
3 0 Y X [Production rule (2)]
0 4 Y X
3 4 Y X
(x,y)→(x+0, y+0)
(x,y)→(x+3, y–3)
(x,y)→(x+4, y)
3 0 Y X
0 3 Y X [Production rule (7)]
3 4 Y X
3 3 Y X [Production rule (2)]
2 4 Y X [Production rule (5)]
2 0 Y X [Production rule (3)]
0 2 Y X [Production rule (7)]

In this way, we can solve the given water jug problem.

Problem 2.2

Solve the given 8-puzzle game as shown in the figure.

3	5	1	2
2		6	3
1	4	7	8
	A	B	C

Initial Position

3	1	2	3
2	4	5	6
1	7	8	
	A	B	C

Goal Position

Solution:

Each box for 8-puzzle problem is represented as shown in the figure.

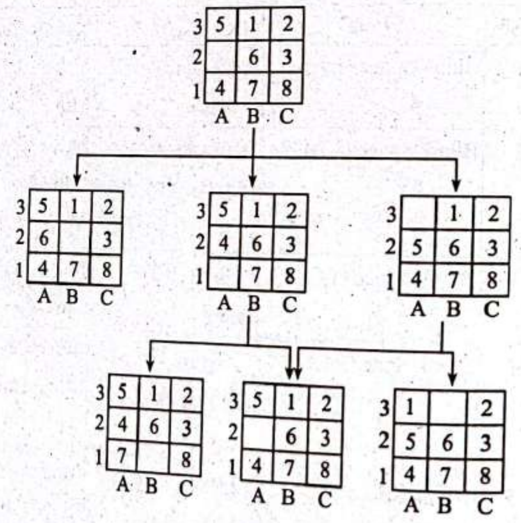
3			
2			2C
1			
	A	B	C

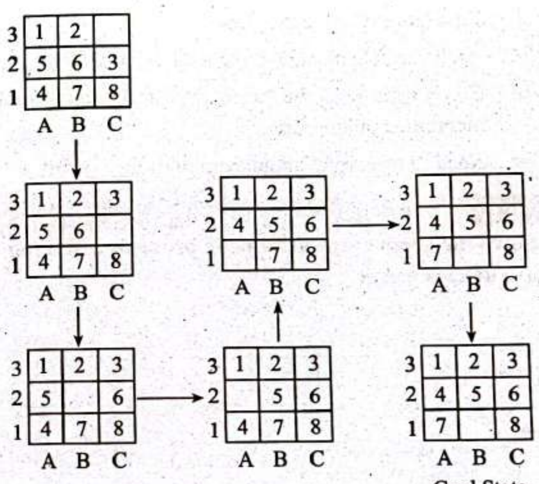
	SN	Left	Right	Definition
Production rules	1	Blank Space (1,A)	(1,A)→(1,B)	Move Right
			(1,A) → (2,A)	Move Up
	2	Blank Space (1,B)	(1,B)→(1,A)	Move left
			(1,B)→(2,B)	Move Up
			(1,B)→(1,C)	Move Right
	3	Blank Space (1,C)	(1,C)→(1,B)	Move Left
			(1,C)→(2,C)	Move Up
	4	Blank Space (2,A)	(2A)→(2,B)	Move Left
			(2A)→(1,A)	Move Down
			(2A)→(3,A)	Move Up
	5	Blank Space (2,B)	(2,B)→(1,B)	Move Down
			(2,B)→(3,B)	Move Up
			(2,B)→(2,A)	Move Left
			(2,B)→(2,C)	Move Right
	6	Blank Space (2,C)	(2,C)→(1,C)	Move Down
			(2,C)→(3,C)	Move Up
			(2,C)→(2,B)	Move Left

Scanned with CamScanner

Production rules	7	Blank Space (3,A)	(3,A)→(3,B) (3,A)→(2,A)	Move Right Move Down
	8	Blank Space (3,B)	(3,B)→(3,A) (3,B)→(3,C) (3,B)→(2,B)	Move Left Move Right Move Down
	9	Blank Space (3,C)	(3,C)→(3,B) (3,C)→(2,C)	Move Left Move Down

Search space can be drawn as





In this way, we can reach the goal state.

2.3 Constraints Satisfaction Problem

A search procedure that operates in a space of constraints is called *constraints satisfaction problem*. Constraints are discovered and propagated as far as possible throughout the system. A guess about something is made and added as a new constraint. The problem can be described as a set of variables (X1, X2, ...) and constraints (C1, C2, ...) and set of domains (D1, D2, ...) where constraints are applicable to variables having some relation with the help of domain D. Constraint propagation terminates for one of two reasons.

i. Contradiction detected i.e., no solution consistent with known constraints.

ii. Propagation has run off stream and there are no further changes that can be made on the basis of current knowledge.

For example, *crypt-arithmetic problem* is a constraints satisfaction problem. Crypt-arithmetic problem has certain rules:

Scanned with CamScanner

i. First letter must be non zero.
ii. Each variable must be assigned uniquely from domain.
iii. Carry over must be taken into account if nothing is mentioned on question.
iv. Need to be careful about symbol of the operation.

Problem 2.3

Solve the given crypt-arithmetic problem assuming there are no carry-over.

$$\begin{array}{cccc} & O & N & E \\ + & O & N & E \\ \hline & T & W & O \end{array}$$

Solution:

$$\begin{array}{cccc} C3=0 & C2=0 & C1=0 & \\ & O & N & E \\ + & O & N & E \\ \hline & T & W & O \end{array}$$

Set of variables X = {O, N, E, T, W}

Set of domain D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Set of constraints C are

i. Starting letters O, T must not be zero.
ii. Each variable must be assigned uniquely.
iii. $E + E = 10 \times C1 + O$
iv. $N + N = 10 \times C2 + W$
v. $O + O = 10 \times C3 + T$

O must be even as it is result of 2E where E can be any number from 0 to 9 i.e., 0,2,4,6,8 but 0 can't be zero because it violates constraints no. 1 and O can't be 6,8 because it will give carry over in 3rd column. So, O can be either 2 or 4. Let's say O = 2 with C2 as 0 which will result T = 4 and E = 1.

$$\begin{array}{cccc} C3=0 & C2=0 & C1=0 & \\ & O=2 & N & E=1 \\ + & O=2 & N & E=1 \\ \hline & T=4 & W & O=2 \end{array}$$

As 1st column doesn't produce carry over, $2 \times N = W$, if we take N = 3 then W = 6

$$\begin{array}{cccc} C3=0 & C2=0 & C1=0 & \\ & O=2 & N=3 & E=1 \\ + & O=2 & N=3 & E=1 \\ \hline & T=4 & W=6 & O=2 \end{array}$$

Now, we have obtained solution for all letters. Therefore, ONE + ONE = 231 + 231 = 462

Problem 2.4

Solve the given crypt-arithmetic problem.

$$\begin{array}{ccccc} & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

Solution:

We can rewrite the question including carry-over as

$$\begin{array}{ccccc} C4 & C3 & C2 & C1 & \\ & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

OR

$$\begin{array}{rr} & 1000 \times S + 100 \times E + 10 \times N + D \\ + & 1000 \times M + 100 \times O + 10 \times R + E \\ \hline & 10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y \end{array}$$

Set of variables are X = {S, E, N, D, M, O, R, Y}

Set of domains are D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Scanned with CamScanner

Set of constraints are C =

i. Value of variables must be unique.

ii. First value of word must not be zero i.e., $S = M = 0$

iii. $D + E = 10 \times C1 + Y$

iv. $N + R + C1 = 10 \times C2 + E$

v. $E + O + C2 = 10 \times C3 + N$

vi. $S + M + C3 = 10 \times C4 + O$

vii. $C4 = M$

Start in the 5th column. Since $9999 + 9999 < 20000$, we must have $M = 1$.

C4=1	C3	C2	C1	
	S	E	N	D
+	M=1	O	R	E
M=1	O	N	E	Y

Then go to the 4th column. Since $999 + 999 < 2000$, we either have $1 + S + 1 = O + 10$ or $S + 1 = O + 10$, meaning $S = O + 8$ or $S = O + 9$, and $O = 0$ or 1. Since S is a single digit, and $M = 1$, we must have $O = 0$.

C4=1	C3	C2	C1	
	S	E	N	D
+	M=1	O=0	R	E
M=1	O=0	N	E	Y

In the 3rd column, since E cannot equal to N, we cannot have $E + 0 = N$. Thus, we must have $1 + E + 0 = N$. Since, N cannot be 0 which is taken by O, we must also have E less than 9. So, there cannot be carryover in this column, and the 2nd column must have carryover.

C4=1	C3=0	C2=1	C1	
	S	E	N	D
+	M=1	O=0	R	E
M=1	O=0	N	E	Y

Returning to the 4th column (which has no carryover from the 3rd), we must have $S + 1 = 10$, which means $S = 9$.

C4=1	C3=0	C2=1	C1	
	S=9	E	N	D
+	M=1	O=0	R	E
M=1	O=0	N	E	Y

Now we know $1 + E = N$, and there must be carryover from the 2nd column. So, we have two cases: $N + R = E + 10$, or $N + R + 1 = E + 10$. We can substitute $1 + E = N$ in both cases to get $(1 + E) + R = E + 10 \rightarrow R = 9$ (but 9 is already taken), or we have $1 + E + R + 1 = E + 10 \rightarrow R = 8$. So, we must have $R = 8$.

C4=1	C3=0	C2=1	C1=1	
	S=9	E	N	D
+	M=1	O=0	R=8	E
M=1	O=0	N	E	Y

Now in the unit column $D + E = Y$, and it must give carryover. Since Y cannot be 0 or 1, we need $D + E \geq 12$. Since 9 and 8 are taken for S and R, we can have $5 + 7 = 12$ or $6 + 7 = 13$. So, either $D = 7$ or $E = 7$. If $E = 7$, then $E + 1 = N$. So $N = 8$ which is not possible since $R = 8$. So, we must have $D = 7$, meaning E is either 5 or 6.

C4=1	C3=0	C2=1	C1=1	
	S=9	E	N	D=7
+	M=1	O=0	R=8	E
M=1.	O=0	N	E	Y

Scanned with CamScanner

If E = 6, then N = 7 which is not possible as D = 7. So, we must have E = 5 and N = 6. This means D + E = 7 + 5 = 12, and thus Y = 2.

	C4=1	C3=0	C2=1	C1=1	
		S=9	E=5	N=6	D=7
+		M=1	O=0	R=8	E=5
	M=1	O=0	N=6	E=5	Y=2

Now, we have obtained solution for all letters. Therefore, SEND + MORE = 9567 + 1085 = 10652

Problem 2.5

Solve the given crypt-arithmetic problem.

			T	E	N
			T	E	N
+	F	O	R	T	Y
	S	I	X	T	Y

Solution:

We can write the given crypt-arithmetic problem including carry-over as:

	C4	C3	C2	C1	
			T	E	N
			T	E	N
+	F	O	R	T	Y
	S	I	X	T	Y

Set of variables X = {T, E, N, F, O, R, Y, S, I, X}

Set of domain D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Set of constraints C

i. First letter must not be Zero. i.e. T, F, S must not be 0.

ii. Each variable should be assigned uniquely.

iii. $C4 + F = S$

iv. $C3 + O = 10\times C4 + I$

v. $C2 + T + T + R = 10\times C3 + X$

vi. $C1 + E + E + T = 10\times C2 + T$

vii. $N + N + Y = 10\times C1 + Y$

As F and S can't be same and 0,C4 must be 1. C3 + O can't give carry-over greater than 1. So F, S must be consecutive numbers.

	C4=1	C3	C2	C1	
			T	E	N
			T	E	N
+	F	O	R	T	Y
	S	I	X	T	Y

C3 can't be 0 as it will result O and I same. So, C3 can be either 1 or 2. In 1st column, $2N + Y = 10\times C1 + Y$, and in 2nd column, $C1 + 2E + T = 10\times C2 + T$. Only two digits can give such numbers which can result same number even after adding by 2 time×variable i.e., 0, 5. So, N must be 0 and E must be 5 which will result C1 = 0, C2 = 1.

	C4=1	C3	C2=1	C1=0	
			T	E=5	N=0
			T	E=5	N=0
+	F	O	R	T	Y
	S	I	X	T	Y

As C3 + O must give carry over, O must be a big number i.e., 8 or 9.

If O = 8 gives I = 0 so O must be 9, then C3 must be 2, otherwise C3 + 9 = 10 and 0 is already taken for N.

This will result value of I = 1

80 Sol

Scanned with CamScanner

C4 = 1	C3 = 2	C2 = 1	C1 = 0	
		T	E = 5	N = 0
		T	E = 5	N = 0
+F	O = 9	R	T	Y
S	I = 1	X	T	Y

To get carry over 2 in 4th column, T can be 7, 8 . Let us take T as higher value i.e., 8 then $1 + 8 + 8 + R = 10 \times 2 + X$. R must be greater than 3 as it will result $17 + R < 20$, so R can be either 4, 6, 7. Let us take R = 7, then X = 4.

C4 = 1	C3 = 2	C2 = 1	C1 = 0	
		T=8	E = 5	N = 0
		T=8	E = 5	N = 0
+F	O = 9	R=7	T=8	Y
S	I = 1	X=4	T=8	Y

Only 2, 3, 6 are already taken for other variables. Now F must be equal to 2 to give S as 3 and Y = 6.

C4 = 1	C3 = 2	C2 = 1	C1 = 0	
		T=8	E = 5	N = 0
		T=8	E = 5	N = 0
+F=2	O = 9	R=7	T=8	Y=6
S=3	I = 1	X=4	T=8	Y=6

Now, we have obtained the solution for all letters. Therefore,

TEN + TEN + FORTY = 850 + 850 + 29786 = 31486

EXERCISE

1. What is well-defined problem? You are given two jugs, a 5-gallon jug and a 7-gallon jug, a pump which has unlimited water which you can use to fill the jug, and the ground on which water can be poured. Neither jug has any measuring markings on it. How can you get exactly 4 gallons of water in the 7-gallon jug? Solve the problem using production system.
2. There is a goat, a tiger, a man and a bundle of grass on one side of a river. There is a boat which can only carry two

items at a time and can only be sailed manually by the man. How can they cross the river? Solve the problem using production system.

3. Solve the following crypt-arithmetic problems:
   a) LOGIC + LOGIC = PROLOG
   b) WRONG + WRONG = RIGHT
   c) ONE + ONE + TWO = FOUR
   d) LETS + WAVE = LATER
   e) CROSS + ROADS = DANGER
   f) KYOTO + OSAKA = TOKYO
   g) BASE + BALL = GAMES
   h) RIGHT + RIGHT = WRONG
   i) APPLE + GRAPE = CHERRY
4. Given two unmarked jugs, one which holds 7 liters, and another which holds 11 liters, an unlimited supply of water, and no need to conserve, how do you measure exactly 6 liters?
5. Solve the 8 queens problem in chess world. Place the Queen such that no queen can attack each other.
6. Solve the tower of Hanoi problem with 4 tiles or disk (w, x, y, z) and 3 poles namely A, B, C from pole A to pole C.

80 Sol

Scanned with CamScanner

Chapter 3

SEARCHING TECHNIQUE

3.1 Introduction

Searching is the process of finding out the appropriate or the required state (goal state) among the possible states. Searching is performed in the state space of a specified problem. The search process is carried out by constructing a search tree.

Search is a universal problem solving technique. The search involves systematic trial and error exploration of an alternative solution. Many problems don't have a simple algorithmic solution. Casting this problem as search problems are often the easiest way of solving them. It is useful when the sequence of actions required to solve a problem is not known.

Every problem has these steps to solve:

1. **Define the problem**: Must include precise initial state & final state.
2. **Analyze the problem**: Select the most important features that can have an immense impact.
3. **Isolate and represent**: Convert these important features into knowledge representation.
4. **Problem solving technique(s)**: Choose the best technique and apply it to particular problem.

Some terminologies we need to know for searching technique are described below:

- **Search space**: It is the representation of all possible conditions and solutions.
- **Initial state**: It is the state where the searching process started.
- **Goal state**: It is the state where we target to reach in searching process.
- **Problem space**: "What to solve"

- **Searching strategy**: It is a method by which the search procedure is controlled.
- **Search tree**: The process of showing possible solutions from initial state is called *search tree*.

Searching through state space (explicitly using searching tree) involves node visiting and node expansion. *Node visiting* is theprocess of checking the selected node is goal node or not and *node expansion* is the process of generating new node related to previous nodes.

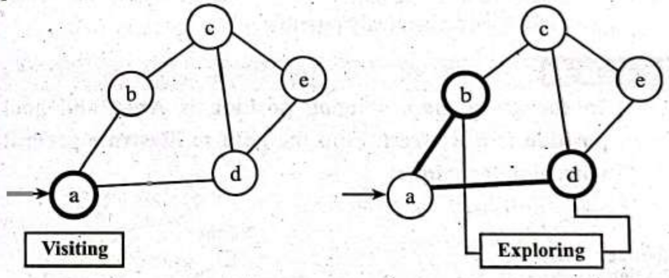


Figure 3.1: Visiting and exploring

3.2 Problem Formulation

A search problem is defined in terms of states, operators and goals.

3.2.1 State

A *state* is a complete description of the world at certain instance.

i. The initial state is the state the world is in when problem-solving begins.

ii. The goal state is a state in which the problem is solved or state meets the goal criteria. Goal state may have single or multiple possibilities.

In the eight-puzzle game, there is a single solution and a single goal state. In chess, there are many winning positions and hence, many goal states.

Scanned with CamScanner

3.2.2 Operator

An *operator* is an action that transforms one state of the world into another state.

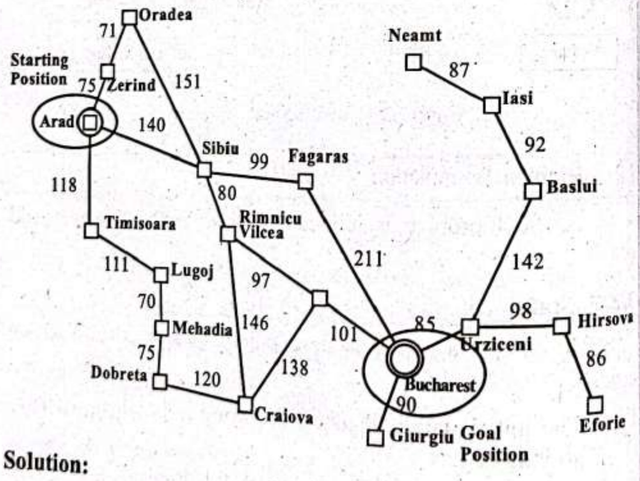
Applicable operators:

In general, not all operators can be applied in all states.

- In a given chess position, only some moves are legal (as defined by the rule of chess)
- In a given eight puzzle configurations, only some moves are physically possible.

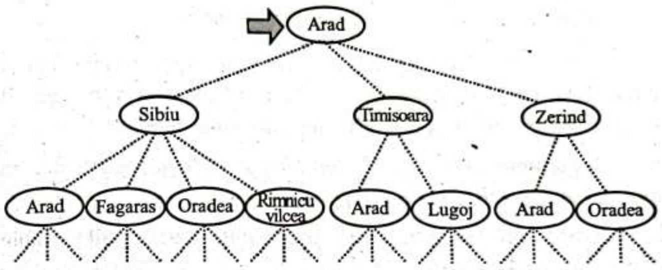
Problem 3.1

In the given map, starting position is Arad and goal position is Bucharest. Find the path to illustrate general searching technique.

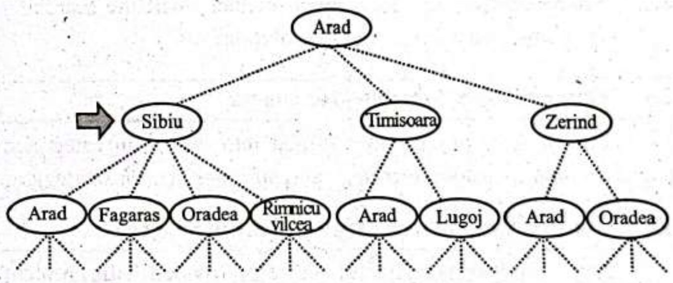


Solution:

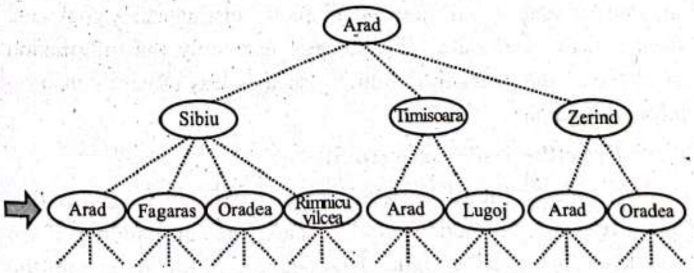
We start with the starting position from Arad which is checked for goal state and found to be false. Hence, we search its neighbor nodes (children nodes), (Zerind, Timisoara and Sibiu).



The process of selecting certain node and checking if it is goal node or not is called visiting while exploring process involves exploring that visited node for other children.



Again, we visit Sibiu and found it to be not a goal node so we explore it.



This process is continued until we reach the destination node.

Scanned with CamScanner

3.3 Performance Metrics for Searching Process

The output from problem-solving (searching) algorithm is either FAILURE or SOLUTION. There are four ways to check if the searching process is how efficient from one another:

i. **Completeness**: Will the searching technique guarantee to find a solution?

ii. **Optimality**: Does it find optimal solution among all possible solutions?

iii. **Time complexity**: How long time will the searching technique take to find the solution?

iv. **Space complexity**: How much memory will the searching technique consume to find the solution?

3.4 Classification of Searching Techniques

Searching technique is classified into two main categories namely *uninformedsearch strategy* and *informed search strategy*.

3.4.1 Uninformed Search Strategy (Blind Search)

These types of search strategies are provided with problem definition and these do not have other additional information about state space. The process of searching in this method is to expand the current state to get a new state and to distinguish a goal state from a main goal state. This strategy uses only the information available in the problem definition, so it is less effective than an informed search.

1. **Breadth-First Search (BFS)**

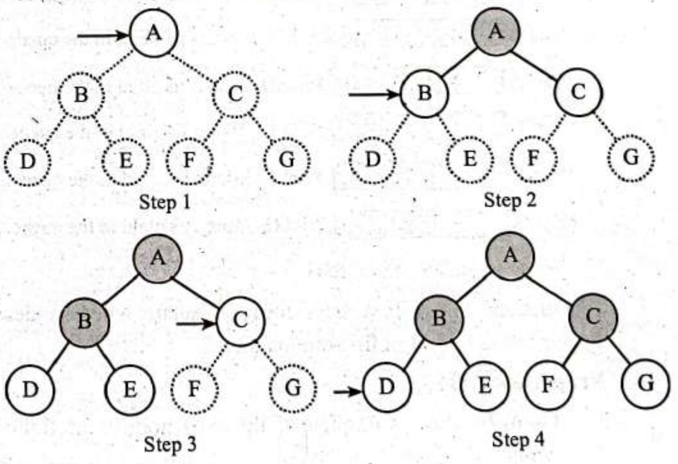
It proceeds level by level down the search trees. Starting from the root node (initial state)explores all children of the root node, left to right. If no solution is found, expand the first (leftmost) child of the root node, then expands the second node at depth 1 and so on.

Process:

i. Place the start node in the queue.

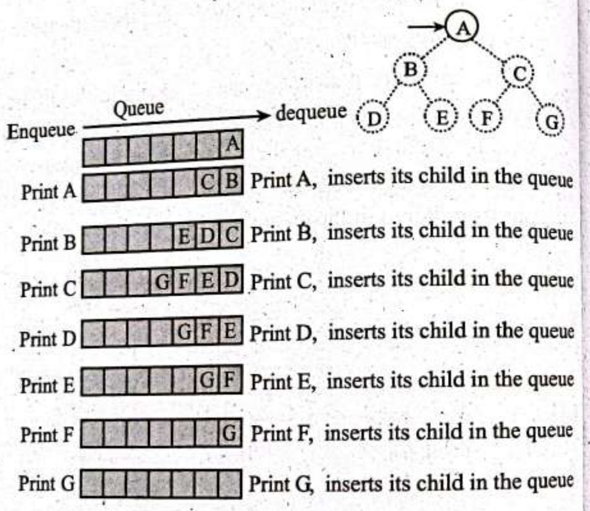
ii. Examine the node all the front of the queue.
- If the queue is empty, stop.
- If the node is the goal, stop.
- Otherwise, add the children of the node to the end of the queue.

Let us consider a search space as shown in the figure below where we need to find a path from A to D using breadth first search. At first we put the start node in the queue. Examine the first element of the queue. If it is the goal, stop otherwise put its children in the queue.



In breadth-first search, nodes are searched in a horizontal line and later checked for goal state. Exploring is done in first in first out manner same as in queue. In the figure, node A is explored for its child B, C as be is inserted earlier we explore B and C is done later. In breadth-first search we explore the node in further depth than where the goal state lies. If we assume goal node as C then we have to explore D. E doesn't need to be searched as we already reach the goal node.

If we have to show it in queue to reach from A to G then we follow the steps as shown in the figure below by updating the queue for every steps.


Queue
Enqueue
dequeue
A
B
C
D
E
F
G
Print A
C B
Print A, inserts its child in the queue
Print B
E D C
Print B, inserts its child in the queue
Print C
G F E D
Print C, inserts its child in the queue
Print D
G F E
Print D, inserts its child in the queue
Print E
G F
Print E, inserts its child in the queue
Print F
G
Print F, inserts its child in the queue
Print G
Print G, inserts its child in the queue

Flow pattern: ABCDEFG

The pattern shows how BFS forms a queue where nodes areexpanded in first in first out manner.

Properties of BFS:

i. **Completeness:** Complete if the goal node is at finite depth.

ii. **Optimality:** It is guaranteed to find the shortest path. If the goal node is available, the algorithm would already reach the node first. Hence, we can see that the algorithm is optimal.

iii. **Time complexity:**

- For a search tree of branching factor equal to 'b' expanding the root yield 'b' nodes at the 1st level.
- Again, expanding the 'b' nodes at the 1st level yield b^2 node at the second levels.

- If the goal is in d^{th} level, in the worst case, goal node would be the last node in d^{th} level.
- We should expand $(b^d - 1)$ nodes in d^{th} level (except goal node itself). Total nodes in d^{th} level $= b\ (b^d - 1) = b^{d+1} - b$.
- Total no. of nodes generated $= 1 + b + b^2 + b^3 + \ldots\ldots b^{d+1} - b$.
- Time complexity $= O(b^{d+1})$

iv. **Space complexity:** The time complexity and spacecomplexity for BFS would be the same as this search will have to store all the nodes in the memory until it finishes searching the whole tree. Hence space complexity would be $O(b^{d+1})$.

Weakness:

- High time and memory requirement.

2. Depth-First Search

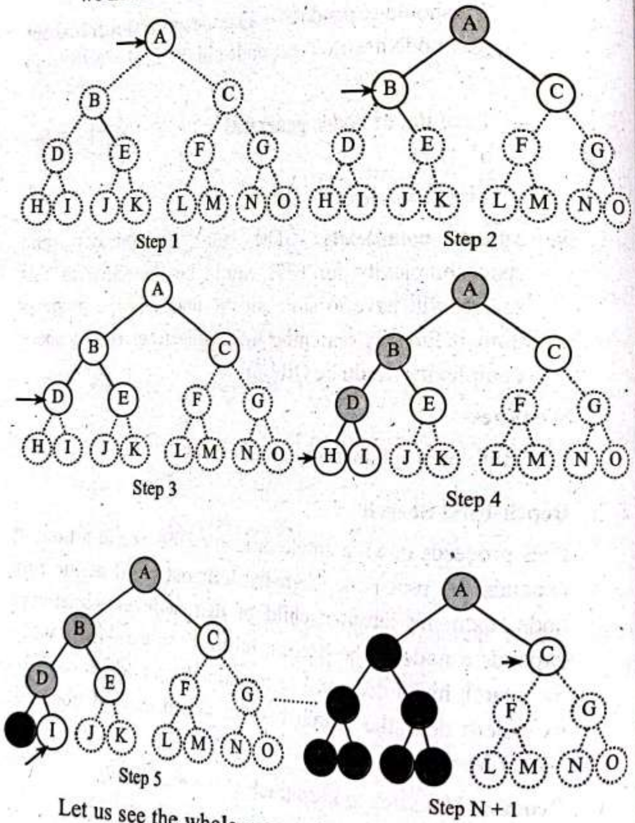
DFS proceeds down a single branch of the tree at a time. It expands the root node, then the leftmost child of the root node, then the leftmost child of that node etc. It always expands a node at the deepest level of the tree. Only when the search hits a dead end (a partial solution which can't be extended) does the search backtrack and expand nodes at higher levels.

Process: Use stack to keep track of nodes. (LIFO)

- Put the start node on the stack
- While stack is not empty
  - Pop the stack
  - If the top of the stack is the goal, stop
  - Otherwise push the nodes connected to the top of the stack on the stack (provided they are not already on the stack)

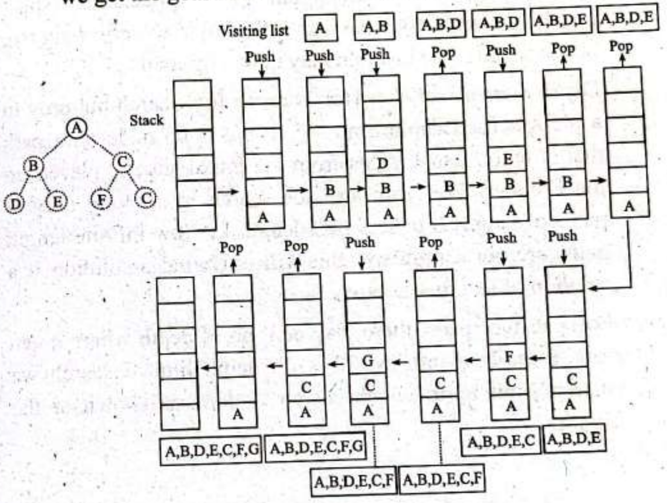
Scanned with CamScanner

Let us consider a search tree as shown in the figure where we need to find a path from A to L using DFS algorithm.



Let us see the whole process in stack form, first of all stack is started with empty and then first node to be visited is inserted in the stack i.e. Node A by push process, which is updated in visited list as {A}. Now, we explore it's children one at a time which is here taken as node B by pushing B and visited list is updated to {A, B} we further go for children of B i.e., D by pushing D. Now node D has no children so we back-track by doing popping D and visited

list is updated to {A, B, D} same process is continued until we get the goal node on the visited list.



Properties:

i. **Completeness:** Incomplete as it may get stuck down, going down an infinite branch that doesn't lead to a solution.

ii. **Optimality:** The first solution found by the DFS may not be the shortest.

iii. **Space complexity:** b as branching factor and d as tree depth level, space complexity = O(b×d)

iv. **Time complexity:**O(b×d)

Weakness:

In DFS,ceratin state form an infinite branch that doesn't lead to a solution.

As in the given figure, suppose our solution is on right branch at depth level 1, but DFS search proceeds down left the branch and continue its search up to 1000's of nodes. It might get stuck in that branch without a solution.

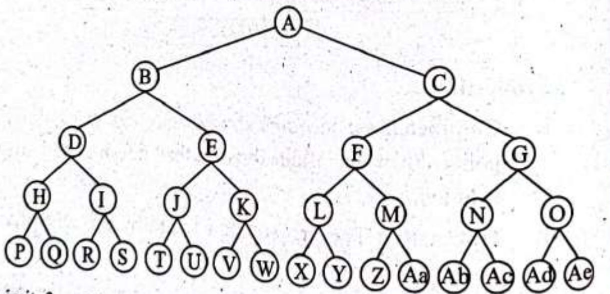
Scanned with CamScanner

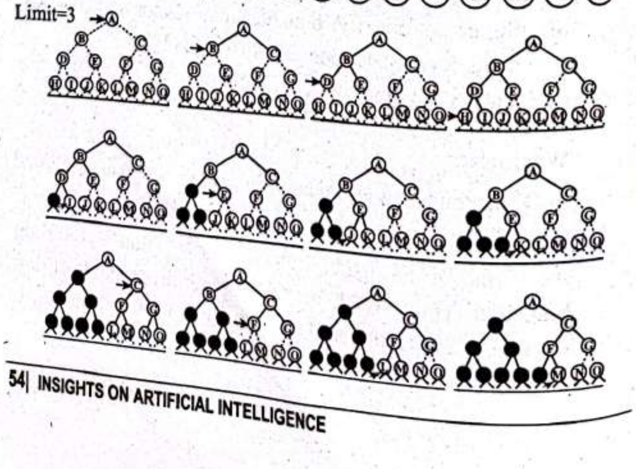
3. **Depth-Limited Search**

Breadth-first has computational, especially space problems.Depth-first can run off down a very long (or infinite) path. The solution may not be optimal.

Depth-limited search performs depth first search but only to a pre-specified depth limit "l". There is no node on a path that is more than L steps from the initial state is placed on the Frontier. We "truncate" the search by looking only at paths of length L or less then length L. Now infinite length paths are not a problem. But will only find a solution if a solution of length ≤ L exists.

For a search space, there may be n no of depth where n can vary from 1 to infinity. Thus, in depth limited search we state a limit up-to which search is done as shown in the figure.





Properties:

i. **Completeness:** Incomplete as the solution may be beyond the specified depth level.

ii. **Optimality:** not optimal

iii. **Space complexity:** b as branching factor and l as tree depth level, space complexity = O(b×l)

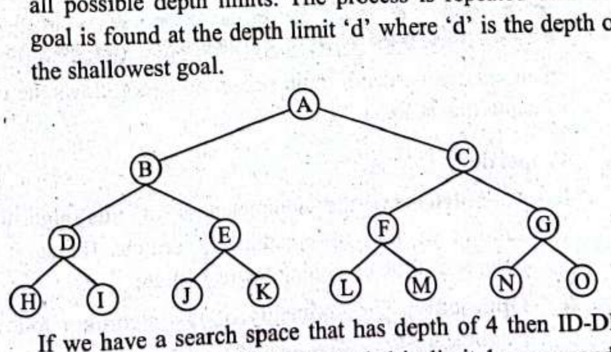
iv. **Time complexity:** O(b^l)

4. **Iterative Deepening DFS**

This searching technique is an extension of depth-limited search. Here, we start at depth limit L = 0, then iteratively increase the depth limit, performing a depth-limited search for each depth limit. The search stops if no solution is found or if the depth-limited search failed without cutting off any nodes because of the depth limit.

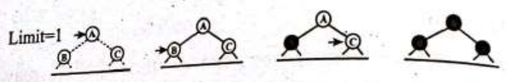
Iterative deepening search uses only linear space and not much more time than other uninformed algorithms.

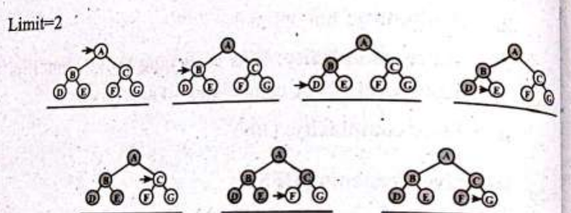
Search is helpful only if the solution is at a given depth level. The hard part of depth-limited search is to choose a good depth limit. This strategy addresses this issue by trying all possible depth limits. The process is repeated until the goal is found at the depth limit 'd' where 'd' is the depth of the shallowest goal.



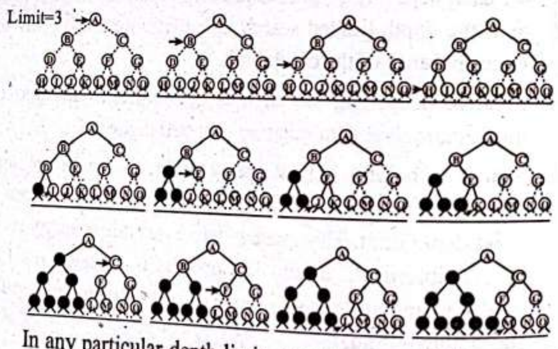
If we have a search space that has depth of 4 then ID-DFS starts with limit 1. All the included in limit 1 are expanded then the explored states are checked for goal state following the laws of depth first search.

Scanned with CamScanner





If we didn't get the result after increasing the depth at 1 then we increase the depth limit to 2 and search for goal and later we increase the depth limit to 3.



In any particular depth limit, search process follows the rule of depth first search.

Properties:

i. **Completeness:** the completeness of this algorithm follows as that of breadth-first search. Hence it is complete if the branching factor is finite.

ii. **Optimality:** The optimality of this algorithm follows that breadth-first search. Hence the path is optimal if the path cost is non decreasing function of the depth.

iii. **Space complexity:** b as branching factor and d as tree depth level, Space complexity = O(b×d)

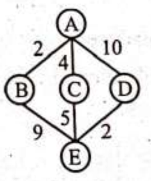
iv. **Time complexity:** $O(b^d)$

5. **Uniform Cost Search**

Uniform cost search can be used if the cost of travelling from one node to another is available. It always expands the lowest cost node on the finger (collection of nodes that are waiting to be expanded). In uniform cost search we have to maintain the open list and close list. Open list consists of nodes that are meant to be explored while close list contains nodes that are visited during searching. We stop the exploration process once it is time to explore the goal node

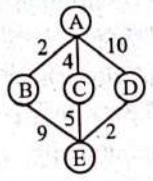
Problem 3.2

Find a path from A to E using uniform cost search in the figure below.



Solution:

First of all, open list and closed list are initialized as empty



- Expand A to B, C, D.

  Closed list = {A}

  Open list = {B, C, D}

Scanned with CamScanner

- The path to B, C and D are 2, 4 and 10 respectively. The path to B is the cheapest one with path cost 2. So, Exploring the node B update the list as

  Closed list = {A, B}.

  Open list = {C, D, E}

  Now, the cost of node C, D, E are 4, 10 and 11 respectively. The least cost of node at open list is C so we have to explore node C.

- Expand C to E.

  Closed list = {A, B, C}

  Open list = {D, E}

  Here, path cost of E is updated from 11 to 9 taking E to reachable via C with less cost. Now the cost of D and E are 9 and 10 so the least cost is of goal node we stop here.

  $\therefore$ Total path cost = $4 + 5 = 9$.

- This is optimal solution and the path is A-C-E which has past cost of 9.

Properties:

i. **Completeness:** Complete if the cost of every step is greater than or equal to some small positive constant 'e'.

ii. **Optimally:** Optimal if the cost of every step is greater than or equal to some small positive constant 'e'.

iii. **Time complexity:** $O(b^{C^*/e})$ where C^* is the cost of optimal path & e is small positive constant.

iv. **Space complexity:** $O(b^{C^*/e})$

Weakness:

- Does not care about 'no' of steps a path has but only about their cost. Hence it might get stuck in an infinite loop if it expands a node that has zero cost action leading back to the same state.

3.4.2 Informed Search

These search strategies have problem specific knowledge apart from the problem definition. These can find solutions more efficiently than uninformed search strategy by the use of heuristics.

Heuristics is a search technique that improves the efficiency of a search process. It guides the search process in the most profitable direction by suggesting which path to follow first when more than one is available. The heuristics search can be categorized into different categories, as follows:

1. **Best First Search**

In this type of search, a node is selected for expansion based on an evaluation function f(n). The node with the lowest evaluation function is expanded first. The measure of theheuristic, i.e., theevaluation must incorporate some estimate of the cost of the path from a state to the closest goal state.

The heuristic algorithm may have different evaluation function. On important parameter is the heuristic function h(n) where h(n) is the estimated cost of the cheapest path from node n to a goal node.

The search can be further divided into two major types, as follows:

i. **Greedy Best First Search**

In this strategy, the node whose state is judged to be the closest to the goal state is expanded first. It evaluates the nodes by using just the heuristic function. Hence, in this case

Evaluation function, f(n)=h(n)

h(n)=0 if n is the goal.

One example of the heuristic function may be the straight line distance to the goal in route finding problem.

Since this strategy prefers to take the biggest possible bite out of the remaining cost to reach the goal, without worrying

Scanned with CamScanner

about whether this will be the best in the long run, it is called "greedy search".

Properties:

i. **Completeness:** This algorithm can start down an infinite path and never return to any other possibilities. Hence this algorithm is not complete.

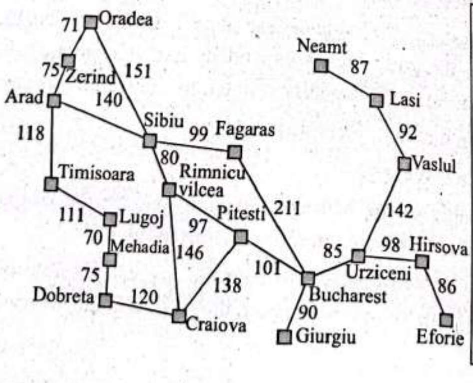
ii. **Optimality:** This algorithm looks for the immediate best choice and doesn't make a careful analysis of the long term options. Hence it may give longer solution even is a shorter solution exists. This algorithm is not optimal.

iii. **Time complexity:** The time complexity of this algorithm is $O(b^m)$,where m is the maximum depth of the search space.

iv. **Space complexity:** The space complexity of the algorithm is $O(b^m)$ too since all the nodes have to be kept in memory.

Problem 3.3

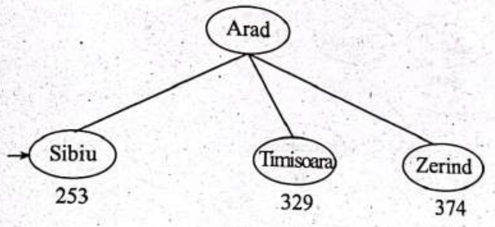
Given a map of city where the starting position is Arad and the destination is Bucharest along with the path cost required to reach from one city to another.



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Pradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisora	329
Urziceni	80
Vaslul	199
Zerind	374

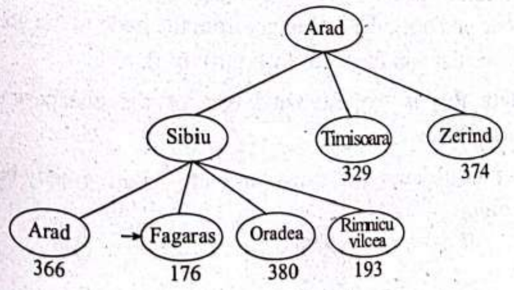
Solution:

First, we start from initial position (Arad) which has its children as Sibiu, Timisoara and Zerind with heuristic value as 253, 329 and 374 respectively.



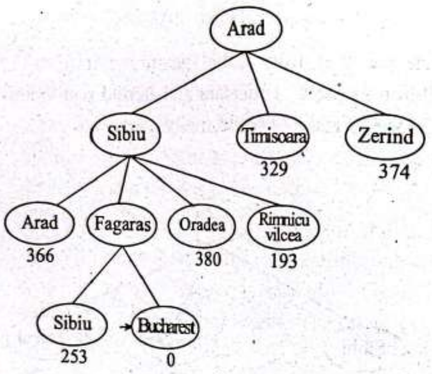
Step 1

Sibiu has the least heuristic cost so it is selected and explored for its children. The children nodes of Sibiu are Arad, Fagaras, Oradea and RimnicuVilcea with heuristic value as 366, 176, 380, 193 respectively as shown in the figure.



Step 2

Now, the node with least heuristic value is Fagaras, so it is selected and explored. Fagaras has children as Sibiu and Bucharest with heuristic cost 253 and 0 respectively.



Step 3

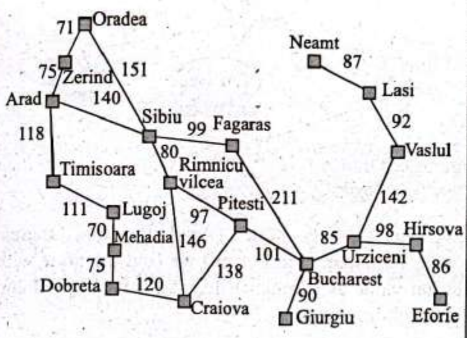
As we reach Bucharest, the path greedy-best search suggests is Arad→Sibiu→Fagaras→Bucharest. This path may or may not be optimal.

ii. A* Search

This algorithm is a best first search algorithm which evaluates nodes by combining g(n), the cost to reach the node and h(n), the cost to get from the node to the goal. Thus the evaluation function, $f(n)=g(n)+h(n)$.

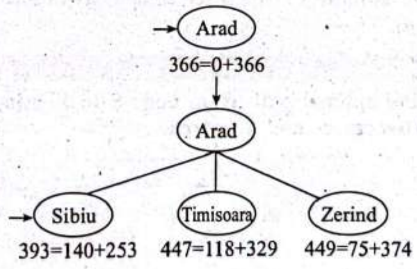
Here f(n) is the estimated cost of the cheapest solution through n.

Let us discuss the same problem as in greedy-best first search.



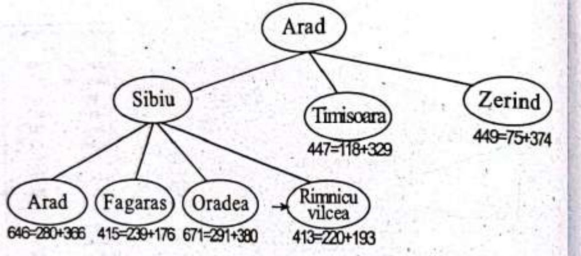
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Pradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisora	329
Urziceni	80
Vaslui	199
Zerind	374

At first, we start with initial position Arad which has its children nodes as Sibiu, Timisoara and Zerind with evaluation value as 393, 447 and 449 respectively as evaluation value is the result of sum of path travelled to reach the goal g(n) and path cost to reach the destination h(n).



Now, the node with least cost among all the nodes explored is Sibiu so it is selected and explored which has children nodes as Arad, Fagaras, Oradea and RimnicuVilcea with evaluation value as 646, 415, 671 and 413 respectively as shown in the figure.

Scanned with CamScanner



Again, if we check all the nodes (Arad, Fagaras, Oradea, Rimnicuvilcea, Timisoara and Zerind) we find the node with least evaluation value as RimnicuVilcea so it is selected and explored for its children node.

Properties:

i. **Completeness:** This algorithm is complete if the heuristic function, h(n) is admissible

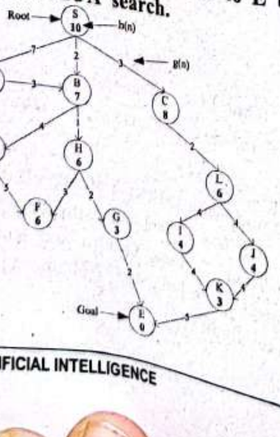
ii. **Optimality:** This algorithm is optimal too if the heuristic function, h(n) is admissible

iii. **Time complexity :** The time complexity of this algorithm is O(bd).

iv. **Space complexity :** The space complexity of this algorithm is O(bd).

Problem 3.4

Find the optimal path from node S to E using Greedy Best First search and A* search.



Solution:

Greedy

Step 1: Start by adding the start node 'S' to OPEN list with the path distance equal to zero (0).

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
S	10		

Step 2(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list.

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
A	9	S	
B	7		
C	8		

Step 2(b): Re-order the OPEN list in ascending order of the combined heuristic value

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
B	7	S	
C	8		
A	9		

Step 3(a): Move the first node in the OPEN list to the closed list and expand its immediate successors by adding them to the OPEN list.

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
C	8	S	
A	9	B	S
D	8		
H	6		

Scanned with CamScanner

Step 3 (b): Re-order the list in ascending order

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
H	6	S	
C	8	B	S
D	8		
A	9		

Step 4(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
C	8	S	
D	8	B	S
A	9	H	B
F	6		
G	3		

Step 4(b): Re-order the list in ascending order

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
G	3	S	
F	6	B	S
C	8	H	B
D	8		
A	9		

Step 5(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
F	6	S	
C	8	B	S
D	8	H	B
A	9	G	H
E	0		

Step 5(b): Re-order the list in ascending order

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
E	0	S	
F	6	B	S
C	8	H	B
D	8	G	H
A	9		

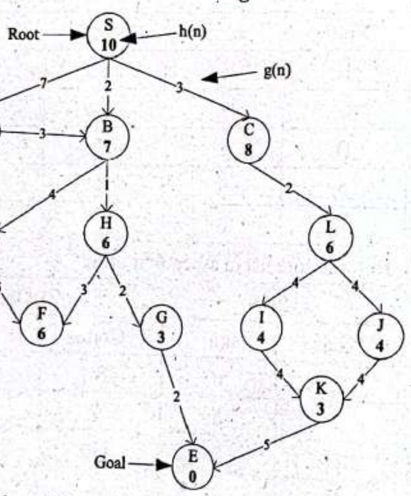
Step 6(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN		CLOSED	
Node	h(n)	Node	Parent Node
F	6	S	
C	8	B	S
D	8	H	B
A	9	G	H
		E	G

Step 6(b): Exit returning TRUE as the Goal node (E) is moved to the CLOSED list. Backtrack the closed list to get the optimal path (E→G→H→B→S)

Scanned with CamScanner

A* Searching



Step 1: Start by adding the start node 'S' to OPEN list with the path distance equal to zero (0).

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
S	10	0	10		

Step 2(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the OPEN list.

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
A	9	7	16	S	
B	7	2	9		
C	8	3	11		

Step 2(b): Re-order the OPEN list in ascending order of the combined heuristic value f(n)

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
B	7	2	9	S	
C	8	3	11		
A	9	7	16		

Step 3(a): Move the first node in the OPEN list to the closed list and expand its immediate successors by adding them to the OPEN list.

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
C	8	3	11	S	
A	9	7	16	B	S
D	8	6	14		
H	3	6	9		

Step 3(b): Re-order the list in ascending order

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
H	3	6	9	S	
C	8	3	11	B	S
D	8	6	14		
A	9	7	16		

Scanned with CamScanner

Step 4(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
C	8	3	11	S	
D	8	6	14	B	S
A	9	7	16	H	B
F	6	6	12		
G	3	5	8		

Step 4(b): Re-order the list in ascending order

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
G	3	5	8	S	
C	8	3	11	B	S
F	6	6	12	H	B
D	8	6	14		
A	9	7	16		

Step 5(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
C	8	3	11	S	
F	6	6	12	B	S
D	8	6	14	H	B
A	9	7	16	G	H
E	0	7	7		

Step 5(b): Re-order the list in ascending order

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
E	0	7	7	S	
C	8	3	11	B	S
F	6	6	12	H	B
D	8	6	14	G	H
A	9	7	16		

Step 6(a): Move the first node in the OPEN list to the CLOSED list and expand its immediate successors by adding them to the open list

OPEN				CLOSED	
Node	h(n)	g(n)	f(n)	Node	Parent Node
C	8	3	11	S	
F	6	6	12	B	S
D	8	6	14	H	B
A	9	7	16	G	H
				E	G

Step 6(b): Exit returning TRUE as the Goal node (E) is moved to the CLOSED list. Backtrack the closed list to get the optimal path (E→G→H→B→S)

3.5 Hill Climbing Algorithm

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a

Scanned with CamScanner

higher value. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of hillclimbing algorithm is travelingsalesman problem in which we need to minimize the distance traveled by the salesman. It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that. A node of hill climbing algorithm has two components which are state and value. Hill climbing is mostly used when a good heuristic is available.In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features of hill climbing algorithm:

Following are some main features of hill climbing algorithm:

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

State-space landscape for hill climbing:

The state-space landscape is a graphical representation of the hillclimbing algorithm which is showing a graph between various states of algorithm and objective function/cost.

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

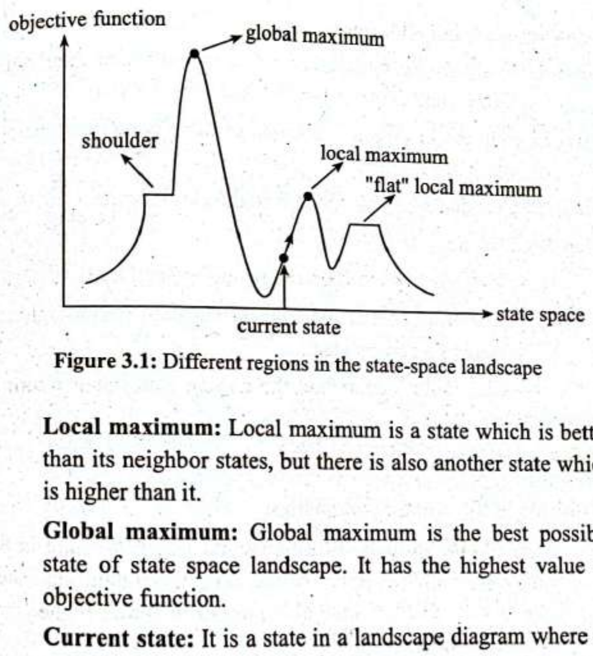


Figure 3.1: Different regions in the state-space landscape

- **Local maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.
- **Global maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
- **Current state:** It is a state in a landscape diagram where an agent is currently present.
- **Flat local maximum:** It is a flat space in the landscape where all the neighbor states of current states have the same value.
- **Shoulder:** It is a plateau region which has an uphill edge.

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

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

Scanned with CamScanner

Algorithm for hill climbing:

Step 1: Evaluate the initial state, if it is goal state then return success and stop.

Step 2: Loop until a solution is found or there is no new operator left to apply.

Step 3: Select and apply an operator to the current state.

Step 4: Check new state:

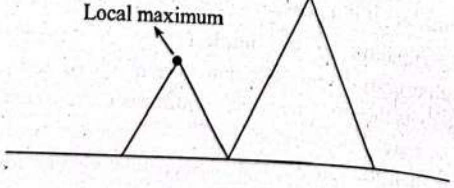
- If it is goal state, then return success and quit.
- Else if it is better than the current state then assign new state as a current state.
- Else if not better than the current state, then return to step 2.

Step 5: Exit

Problems in hill climbing algorithm:

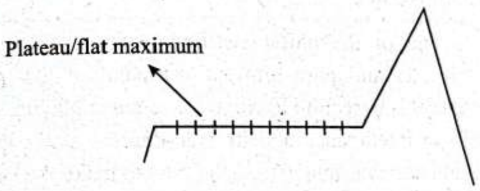
1. **Local maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

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



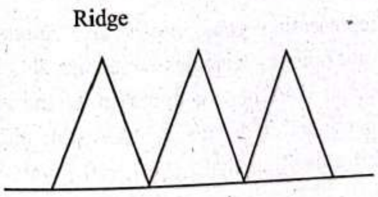
2. **Plateau:** A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

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



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

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



3.5.1 Simulated Annealing

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

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

3.6 Game Playing

It is one of the oldest sub-fields of AI. Game playing involves abstract and pure form of computation that seems to require that seems to require intelligence so game playing has close relationship to intelligence and its well-defined states and rules. Game playing has contributed ideas on how to make the best use of time to reach good decisions.

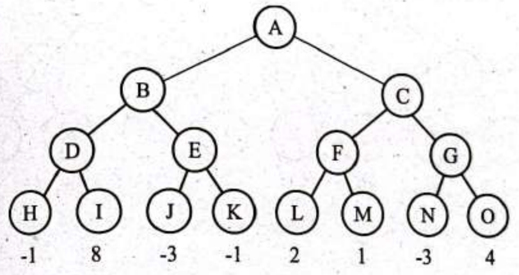
3.6.1 MIN-MAX Algorithm

The *min-max algorithm* is a way of finding an optional move in a two-player game. In the search tree for a two-player game, there are two kinds of nodes, nodes representing your moves and nodes representing your opponent's moves.

Nodes representing your moves are generally drawn as squares or upward pointing triangles which are also known as max nodes. The goal at a MAX node is to maximize the value of the sub tree rooted at that node. To do this, a MAX node chooses the child with the greatest value and that becomes the value of the MAX node.

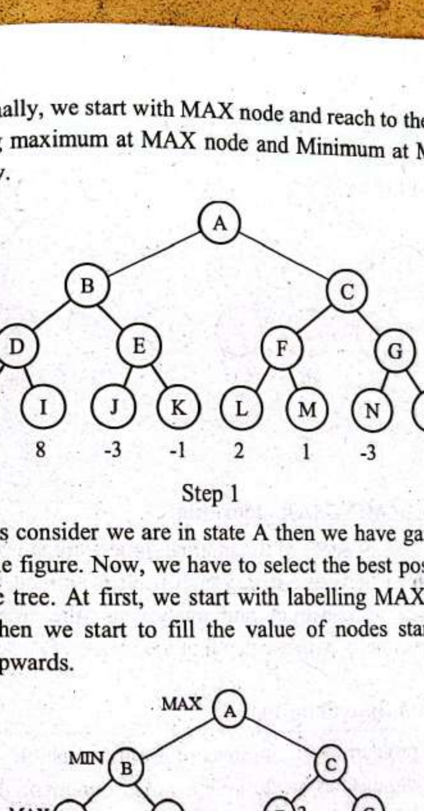
Nodes representing your opponent's moves are generally drawn as circles or downward pointing triangles which are also known as MIN nodes. The goal at a MIN node is to minimize the value of the sub-tree rooted at that node. To do this, a MIN node chooses the child with the least value and that becomes the value of the MIN node.

Normally, we start with MAX node and reach to the terminal by selecting maximum at MAX node and Minimum at MIN node alternatively.

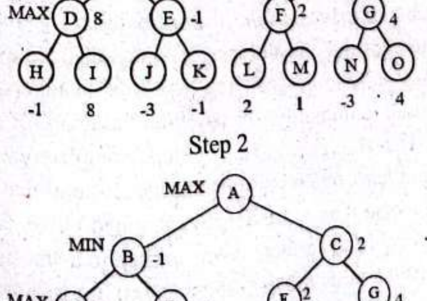


Step 1

Let us consider we are in state A then we have game tree as shown in the figure. Now, we have to select the best possible path in the game tree. At first, we start with labelling MAX and MIN label and then we start to fill the value of nodes starting from bottom to upwards.



Step 2



Step 3

Scanned with CamScanner

Step 4

Limitation of MIN-MAX algorithm:

1. The effectiveness of the minmax procedure is limited by the depth of the game-tree, which itself is limited by the time needed to construct and analyze it (the time increases exponentially with the depth of the tree.)

3.6.2 Alpha-Beta Pruning

The problem with minmax algorithm search is that the number of game states has to be examined exponentially with the greater number of moves. *Alpha-beta pruning* is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minmax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-Tac-Toe, chess, etc.). It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minmax tree, it returns the same move as minmax but prunes away branches that cannot possibly influence the final decision. So, alpha-beta pruning is a way of finding optimal minmax solution while avoiding searching sub trees of moves which won't be selected.

Alpha-beta pruning gets its name from two bounds that are passed along during the calculation, which restrict the set of possible solutions based on the portion of the search tree that has already been seen.

Beta (β) is the minimum upper bound of possible solutions and alpha (α) is the maximum lower bound of possible solutions.

Alpha pruning – Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX ancestors.

Beta pruning – Search can be stopped below any of its MAX node having an alpha value greater than or equal to the beta value of any of its MIN ancestors.

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called *pruning*. This involves two threshold parameter Alpha and beta for future expansion, so it is called *alpha-beta pruning*. It is also called as *alpha-beta algorithm*.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- The two-parameter can be defined as:

  Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.

Scanned with CamScanner

Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

- The alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Condition for alpha-beta pruning:

The main condition which required for alpha-beta pruning is $\alpha >= \beta$.

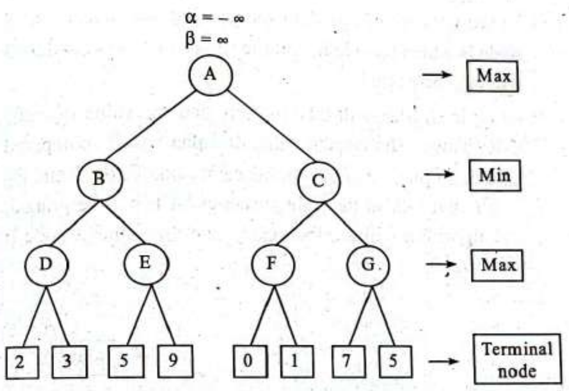
Key points about alpha-beta pruning:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.
- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

Working of alpha-beta pruning:

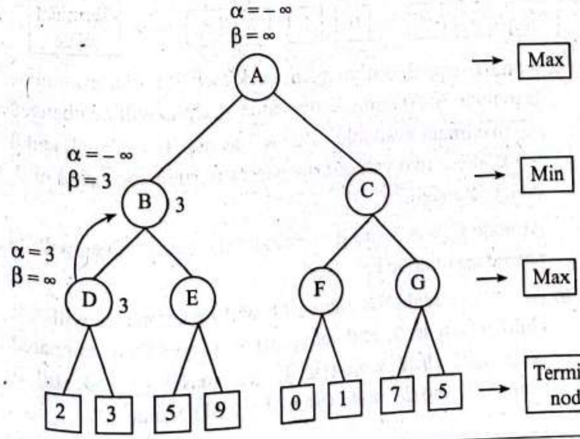
Let's take an example of two-player search tree to understand the working of alpha-beta pruning

Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$, and Node B passes the same value to its child D.



Step 2: At node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

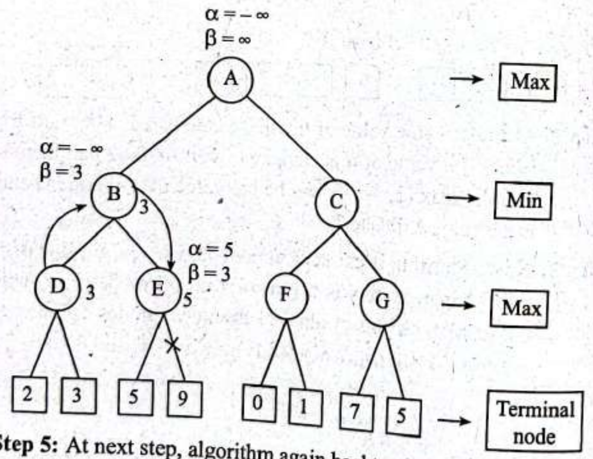
Step 3: Now algorithm backtracks to node B, where the value of β will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent nodes value, i.e. min $(\infty, 3) = 3$, hence at node B now $\alpha = -\infty$, and $\beta = 3$.



Scanned with CamScanner

In the next step, algorithm traverse the next successor of node B which is node E, and the values of α= −∞, and β= 3 will also be passed.

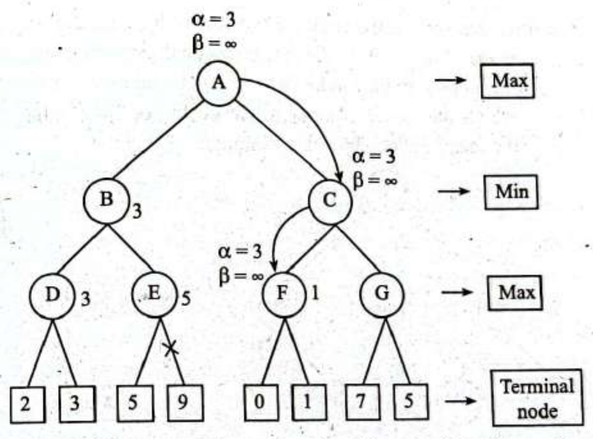
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (−∞, 5) = 5, hence at node E, α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



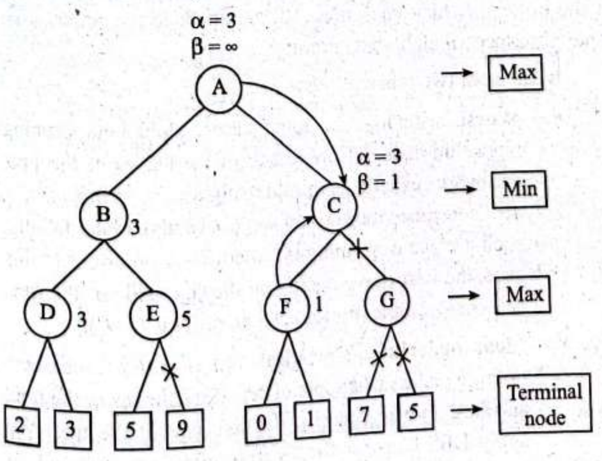
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (−∞, 3)= 3, and β = +∞, these two values now passes to right successor of A which is node C.

At node C, α = 3 and β = +∞, and the same values will be passed on to node F.

Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0) = 3, and then compared with right child which is 1, and max(3,1) = 3 still α remains 3, but the node value of F will become 1.

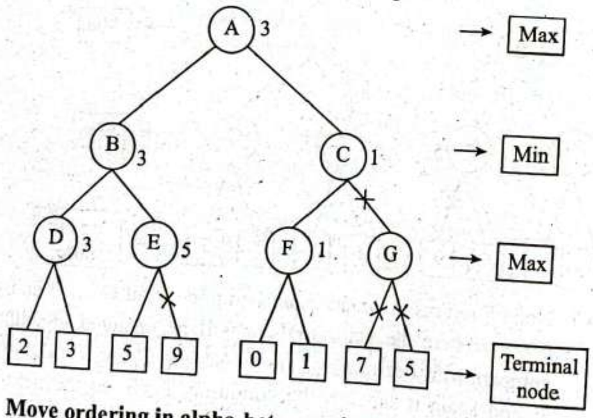


Step 7: Node F returns the node value 1 to node C, at C α = 3 and β = +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α = 3 and β = 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Scanned with CamScanner

Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Move ordering in alpha-beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is O(b^m).
- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimaxalgorithm in the same amount of time. Complexity in ideal ordering is O($b^{m/2}$).

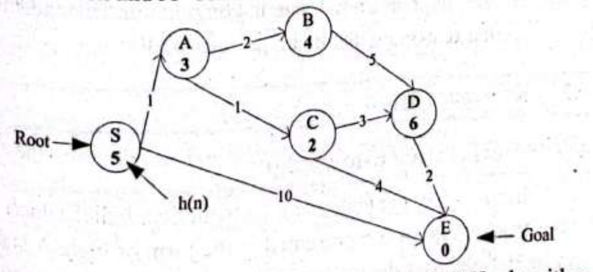
Rules to find good ordering:

Following are some rules to find good ordering in alpha-beta pruning:

- Occur the best move from the shallowest node.
- Order the nodes in the tree such that the best nodes are checked first.
- Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
- We can bookkeep the states, as there is a possibility that states may repeat.

EXERCISE

1. Compare BFS, DFS, DLS, DFID search strategy.
2. Discuss uniform cost search in detail.
3. Differentiate between best first search and A* search with example.
4. Discuss the hill-climbing search algorithm along with problems associated with it and discuss their solutions.
5. Find the optimal path from node S to E using Greedy Best first search and A* search



6. Solve the following diagram using MIN-MAX algorithm and Alpha-beta algorithm.

Scanned with CamScanner

Chapter 4

KNOWLEDGE REPRESENTATION

4.1 Introduction

Knowledge is an abstract term that attempts to capture an individual understanding of a given subject. It is a subset of information that has been extracted, filtered, or formatted.

$$\begin{tikzcd}\boxed{\text{Data}} \arrow[r] & \boxed{\text{Information}} \arrow[r] & \boxed{\text{Knowledge}} \arrow[r] & \boxed{\text{Intelligence}}\end{tikzcd}$$

Knowledge consists of facts, concepts, rules, etc. thatcan be presented in the form as mental images, as spoken sound or written text in human.

Knowledge representation is an area of AI whose fundamental goal is to represent knowledge in such a manner that facilitates inference i.e., drawing conclusion for knowledge. It analyzes how to think formally, how to use symbol to represent a domain of discourse along with function that allow inference about the objects. Knowledge representation helps to address the problem like:

- How do we represent fact about the world?
- How do we reason about them?
- What representations are appropriate for dealing with the real world?
- How to express knowledge in computer understandable form so that reasoning agent can perform well?

4.2 Knowledge Types

4.2.1 Declarative Knowledge

In this type, the statement, facts, notion, belief which can be either true or false are represented in the form of logic.A statement can be defined as a declarative sentence, or part of a sentence, that is capable of having a truth-value, such as being true or false.

Consider the following declarative sentences which has just one part.

- George W. Bush is the 43rd President of the United States.
- Paris is the capital of France.
- Everyone born on Monday has purple hair.

Sometimes, a declarative sentence can contain more than one statements as parts as shown below.

- Either Ganymede is a moon of Jupiter or Ganymede is a moon of Saturn.

While the above compound sentence is itself a statement, because it is true, the two parts, "Ganymede is a moon of Jupiter" and "Ganymede is a moon of Saturn", are themselves statements, because the first is true and the second is false.The term proposition is sometimes used synonymously with statement.

4.2.2 Heuristic Knowledge

It is the knowledge or rule which is related to particular domain. These rules or tricks are used to make judgement and simplify the solution for problem. Heuristic knowledge helps to achieve certain goal using some tricks.While playing chess game, to make opponent's king in danger position is an example of heuristic knowledge.

4.2.3 Procedural Knowledge (Imperative Knowledge)

Knowledge exercised in the performance of some task and processed by an intelligent agent is *procedural knowledge*. Here, the knowledge contains all the required steps to achieve certain goals or solve problems. For example, to arrange a numerical data set in ascending order, we require certain steps.So all these steps in sequence is procedural knowledge for this problem domain.

Scanned with CamScanner

4.3 Propositional Logic

Propositional logic (also known as *sentential logic* and *statement logic*) is a type of declarative knowledge. Propositional logic is the branch of logic that studies ways of joining and/or modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences, as well as the logical relationships and properties that are derived from these methods of combining or altering statements. In propositional logic, the simplest statements are considered as indivisible units, and hence, propositional logic does not study those logical properties and relations that depend upon parts of statements that are not themselves statements on their own, such as the subject and predicate of a statement. The most thoroughly researched branch of propositional logic is classical truth-functional propositional logic, which studies logical operators and connectives that are used to produce complex statements whose truth-value depends entirely on the truth-values of the simpler statements making them up, and in which it is assumed that every statement is either true or false and not both.

4.3.1 Syntax and Formation Rules of Propositional Logic

In any ordinary language, a statement would never consist of a single word, but would always at the very least consists of a noun or pronoun along with a verb. However, because propositional logic does not consider smaller parts of statements, and treats simple statements as indivisible wholes, the language propositional logic uses uppercase letters 'A', 'B', 'C', 'α', 'β', etc. in place of complete statements. The logical signs '∧', '∨', '→', '↔', and '˥' are used in place of the truth-functional operators 'and', 'or', 'if... then...', 'if and only if', and 'not' respectively.

Let us consider the following example.

- Paris is the capital of France and Paris has a population of over two million.

- Therefore, Paris has a population of over two million.

If we use the letter 'C' as our translation of the statement "Paris is the captial of France" in propositional logic, and the letter 'P' as our translation of the statement "Paris has a population of over two million", and use a horizontal line to separate the premise(s) of an argument from the conclusion, the above argument could be symbolized in language PL as follows:

$$\frac{C \land P}{\therefore P}$$

In addition to statement letters like 'C' and 'P' and the operators, the only other signs that sometimes appear in the language PL are parentheses which are used in forming even more complex statements. Consider the English compound sentence, "Paris is the most important city in France if and only if Paris is the capital of France and Paris has a population of over two million." If we use the letter 'I' in language PL to mean that Paris is the most important city in France, this sentence would be translated into PL as follows:

$$I \leftrightarrow (C \land P)$$

4.3.2 Well-Formed Formula

A *well-formed formula* (hereafter abbreviated as wff) of PL is defined recursively as follows:

Any statement letter is a well-formed formula,

1. if α is a well-formed formula, then so is˥ α.
2. if α and β are well-formed formulas, then so is (α ∧β).
3. if α and β are well-formed formulas, then so is (α ∨β).
4. if α and β are well-formed formulas, then so is(α →β).
5. if αand β are well-formed formulas, then so is (α↔ β).
6. nothing that cannot be constructed by successive steps of (1)-(6) is a well-formed formula.

Scanned with CamScanner

The notion of a well-formed formula should be understood as corresponding to the notion of a grammatically correct or properly constructed statement of language PL. This definition tells us, for example, that "ℸ (α ∧ℸ β)" is grammatical for PL because it is a well-formed formula, whereas the string of symbols "ℸ α∧(↔ℸ β)" is not grammatical because it is not well-formed.

4.3.3 Truth Table

So far we have in effect described the grammar of language PL. When setting up a language fully, however, it is necessary not only to establish rules of grammar, but also describe the meanings of the symbols used in the language. We have already suggested that uppercase letters are used as complete simple statements. Because truth-functional propositional logic does not analyze the parts of simple statements, and only considers those ways of combining them to form more complicated statements that make the truth or falsity of the whole dependent entirely on the truth or falsity of the parts, in effect, it does not matter what meaning we assign to the individual statement letters like 'P', 'Q' and 'R', etc., provided that each is taken as either true or false (and not both).

However, more must be said about the meaning or semantics, of the logical operators '∧', '∨', '→', '↔', and 'ℸ'. As mentioned above, these are used in place of the English words, 'and', 'or', 'if... then...', 'if and only if', and 'not', respectively. The signs '∧', '∨', '→', '↔', and 'ℸ' correspond respectively to the truth-functions conjunction, disjunction, implication, bimiplication(equivalence), and negation.

A	B	A∩B	A∪B	ℸA	ℸB	A→B	A↔B
0	0	0	0	1	1	1	1
0	1	0	1	1	0	1	0
1	0	0	1	0	1	0	0
1	1	1	1	0	0	1	1

4.3.4 Some Terminologies

a. **Tautology:** Propositional logic P is called *tautology* if it is true for all circumstances.

b. **Contradiction:** Propositional logic P is called *contradiction* if it contains only false (F).

c. **Contingency:** Propositional logic P is called *contingency* if its truth table contains at least 1 true and 1 false.

d. **Contrapositive:** Contrapositive of P tends to Q is

   (P→Q) is ℸ Q→ℸ P

e. **Inverse:** Inverse of P tends to Q is

   P→Q is ℸ P→ℸ Q

f. **Conjunctive Normal Form (CNF)**

   If different sentences are connected using AND, then it is CNF.

g. **Disjunctive Normal Form (DNF)**

   If different sentences are connected using OR, then it is DNF.

4.3.5 Rules of Inference

Here we give a list of intuitively valid rules of inference. The rules are stated in schematic form. Any inference in which any wff of language PL is substituted unformly for the schematic letters in the forms below constitutes an instance of the rule.

S. No.	Rules of inference	Tautological form	Common name
1.	$\frac{P}{\therefore P \lor Q}$	P→(P∨Q)	**Addition**
2.	$\frac{P \land Q}{\therefore P}$	(P∧Q)→P	**Simplification**
3.	$\frac{P \quad Q}{\therefore P \land Q}$	(P) ∧ (Q)→(P∧Q)	**Conjunction**

Scanned with CamScanner

S. No.	Rules of inference	Tautological form	Common name
4.	$P \to Q$ P $\therefore Q$	$[(P \to Q) \land P] \to Q$	**Modus ponens** (Implication-Elimination)
5.	$P \to Q$ $\neg Q$ $\therefore \neg P$	$[(P \to Q) \land (\neg Q)] \to \neg P$	**Modus tollens**
6.	$P \to Q$ $Q \to R$ $\therefore P \to R$	$[(P \to Q) \land (Q \to R)] \to (P \to R)$	**Hypothetical syllogism**
7.	$P \lor Q$ $\neg P$ $\therefore Q$	$[(P \lor Q) \land \neg P] \to Q$	**Disjunction syllogism**
8.	$P \lor Q$ $\neg P \lor R$ $\therefore Q \lor R$	$[(P \lor Q) \land (\neg P \lor R)] \to (Q \lor R)$	**Resolution**

4.4 First-Order Logic

In the topic of propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentences.

- Some humans are intelligent.
- All humans are intelligent.

To represent the above statements, PL logic is not sufficient, so we require some more powerful logic, such as first-order logic. *First-order logic* is another way of knowledge representation in

artificial intelligence. It is an extension to propositional logic. FOL is sufficiently expressive to represent the natural language statements in a concise way. First-order logic is also known as *predicate logic* or *first-order predicate logic*. First-order logic is a powerful language that develops information about the objects in an easy way and can also express the relationship between those objects. First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:

- **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, etc.
- **Relations:** It can be unary relation such as red, round, is adjacent, etc., or n-any relation such as the sister of, brother of, has color, comes between, etc.
- **Function:** Father of, best friend, third inning of, end of, etc.

As a natural language, first-order logic also has two main parts:

- Syntax
- Semantics

4.4.1 Syntax of First-Order Logic

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL. Following are the basic elements of FOL syntax:

Constants	1, 2, A, John, Mumbai, cat, ...
Variables	x, y, z, a, b, ...
Predicates	Brother, Father, >, ...
Functions	sqrt, LeftLegOf, ...
Connectives	$\land, \lor, \neg, \leftrightarrow, \to$
Equality	$==$
Quantifiers	$\forall, \exists$

Scanned with CamScanner

Atomic Sentences

Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms. We can represent atomic sentences as predicate (term1, term2, ..., term n).

Example:

- Ravi and Ajay are brothers.
  Brothers(Ravi,Ajay)
- Chinky is a cat.
  cat (Chinky)

Complex Sentences

Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "X is an integer".It consists of two parts: the first part "X" is the *subject* of the statement and second part "is an integer" is known as the*predicate.*

$$\underbrace{\text{X}}_{\text{Subject}}\ \underbrace{\text{is an integer.}}_{\text{Predicate}}$$

Predicate logic is also called 1st order predicate logic which allows quantifiers to range over object. 2nd order predicate logic allows quantifiers in range over relations or functions and 3rd order predicate allows quantifiers to range over predicates of predicate.

4.4.2 Quantifiers in First-Order Logic

A *quantifier* is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.These are the symbols that

permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:

- **Universal quantifier** – for all, everyone, everything
- **Existential quantifier** – for some, at least one

1. Universal Quantifier

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.The universal quantifier is represented by a symbol ∀, which resembles an inverted A.

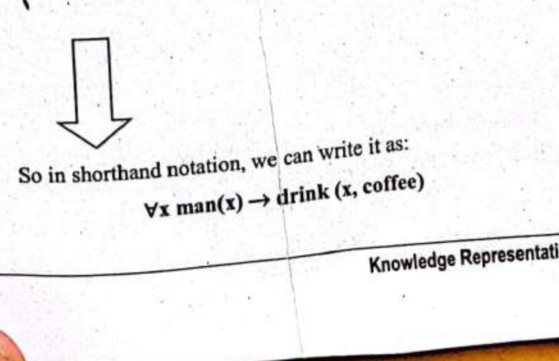
Note: In universal quantifier, we use implication "→".

If x is a variable, then ∀x is read as:

- For all x
- For each x
- For every x

Example:

- All man drink coffee.



So in shorthand notation, we can write it as:

$\forall x\ \text{man}(x) \rightarrow \text{drink}(x, \text{coffee})$

Scanned with CamScanner

It will be read as: There are all x where x is a man who drink coffee.

2. Existential Quantifier

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

Note: In existential quantifier, we always use AND or conjunction symbol (∧).

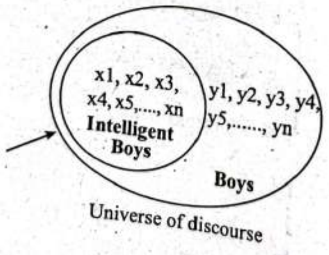
If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- There exists x
- For some x
- For at least one x

Example:

- **Some boys are intelligent.**

  - x1 is intelligent
  ∨
  - x2 is intelligent
  ∨
  - x3 is intelligent
  ∨
  - ⋮
  ∨
  - xn is intelligent



So in shorthand notation, we can write it as:

∃x: boys(x) ∧ intelligent(x)

It will be read as: There are some x where x is a boy who is intelligent.

Properties of quantifiers:

- In universal quantifier, ∀x∀y is similar to ∀y∀x.
- In existential quantifier, ∃x∃y is similar to ∃y∃x.
- ∃x∀y is not similar to ∀y∃x.

3. Free and Bound Variables in Quantifiers

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in first-order logic which are given below:

- **Free variable:** A variable is said to be a *free variable* in a formula if it occurs outside the scope of the quantifier.

  Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

- **Bound variable:** A variable is said to be a *bound variable* in a formula if it occurs within the scope of the quantifier.

  Example: ∀x [A (x) B(y)], where x and y are the bound variables.

Examples of FOL using quantifier:

- **All birds fly.**

  In this sentence, the predicate is "fly(bird)". Since there are all birds who fly, it will be represented as follows:

  ∀x bird(x) →fly(x).

- **Every man respects his parent.**

  In this question, the predicate is "respect(x, y)" where x=man and y= parent. Since there is every man, we will use ∀, and it will be represented as follows:

  ∀x man(x) → respects (x, parent)

- **Some boys play cricket.**

  In this question, the predicate is "play(x, y)" where x= boys, and y= game. Since there are some boys, we will use ∃, and it will be represented as follows:

  ∃x boys(x) ∧ play(x, cricket)

Scanned with CamScanner

- **Not all students like both Mathematics and Science.**

  In this question, the predicate is "like(x, y)" where x= student and y= subject. Since there are not all students, we will use ∀ with negation, and it will be represented as follows:

  ¬ ∀(x) [student(x)→like(x,Mathematics)∧ like(x, Science)]
- **Only one student failed in Mathematics.**

  In this question, the predicate is "failed(x, y)" where x= student and y= subject. Since there is only one student who failed in Mathematics, we will use ∃ and it will be represented as follows:

  ∃(x) [student(x) ∧ failed (x, Mathematics)]
- **John is a boy.**

  boy(John)
- **Ram likes Sita but not Gita.**

  likes(Ram,Sita)∧¬ likes(Ram,Gita)
- **Ram is a tall guy who likes Sita.**

  tall(Ram)∧likes(Ram,Sita)
- **Ram dislikes children who fight.**

  ∀x(children (x) ∧ fight(x))→ (¬ likes(Ram, x))
- **All basketball players are tall.**

  ∀x basketball(x) → tall(x)
- **All purple mushrooms are poisonous.**

  ∀x purple_mushroom(x) → poisonous(x)
- **Some basketball players are not tall.**

  ∃x basketplayer(x) ∧¬ tall(x)
- **All students are smart.**

  ∀x (student(x) →smart(x))

- **There exists a student.**

  ∃x student(x)
- **There exists a smart student.**

  ∃x (student(x) ∧ smart(x))
- **Bill is a student.**

  student(Bill)
- **Bill takes either analysis or geometry (but not both)**

  takes(Bill, analysis) ↔ ¬ takes(Bill, geometry)
- **Bill takes analysis or geometry (or both).**

  takes(Bill, analysis) ∨ takes(Bill, geometry)
- **Bill takes analysis and geometry.**

  takes(Bill, analysis) ∧ takes(Bill, geometry)
- **Bill does not take analysis.**

  ¬ takes(Bill, analysis)
- **No student loves Bill.**

  ¬ ∃x (student(x) ∧ loves(x, Bill))
- **Bill has at least one sister.**

  ∃x sisterof(x, Bill)
- **Bill has no sister.**

  ¬ ∃x sisterof(x, Bill)
- **Every student loves some student.**

  ∀x (student(x) →∃y (student(y) ∧loves(x, y)))
- **Every student loves some other student.**

  ∀x (student(x) →∃y (student(y) ∧¬ (x = y) ∧loves(x,y)))
- **There is a student who is loved by every other student.**

  ∃x (student(x) ∧∀y (student(y) ∧¬ (x = y) →loves(y,x)))

Scanned with CamScanner

- **Every student takes at least one course.**
  $\forall x\, (\text{student}(x) \rightarrow \exists y\, (\text{course}(y) \wedge \text{takes}(x,y)))$
- **Every student who takes analysis also takes geometry.**
  $\forall x(\text{student}(x) \wedge \text{takes}(x, \text{analysis}) \rightarrow \text{takes}(x, \text{geometry}))$
- **No student can fool all the other students.**
  $\neg \exists x(\text{student}(x) \wedge \forall y(\text{student}(y) \wedge \neg (x=y) \rightarrow \text{fools}(x,y)))$
- **All people who eats everything are fat and unhealthy.**
  $\forall x \forall y\, \text{eats}(x,y) \rightarrow (\text{fat}(x) \wedge \text{unhealthy}(x))$
  OR
  $\forall x \forall y\, \text{eats}(x,y) \rightarrow (\text{fat}(x) \wedge \neg \text{healthy}(x))$
- **Everyone who is lucky or studious can pass exam.**
  $\forall x\, (\text{lucky}(x) \vee \text{studious}\, (x)) \rightarrow \text{passexam}(x)$
  OR
  $\forall x\, (\text{lucky}(x) \vee \text{studious}\, (x)) \rightarrow \text{pass}(x,\text{exam})$
- **Everyone is a parent of anyone if everyone is a father of anyone or mother of anyone.**
  $\forall x \forall y\, (\text{mother}(x,y) \vee \text{father}(x,y)) \rightarrow \text{parent}(x,y)$
- **Everyone who loves all animals are loved by someone.**
  $\forall x\, [\forall y\, \text{animal}(y) \rightarrow \text{loves}(x,y)] \rightarrow [\exists z \text{loves}(z,x)]$

4.4.3 Rules of Inference for Quantifiers

S. No.	Rules	Common name
1.	$\dfrac{\forall x\, P(x)}{\therefore P(c) \text{ for all } c}$ c is any element of universe	Universal instantiation
2.	$\dfrac{\exists x\, P(x)}{\therefore P(c)}$ c is some element of universe for which P(c) is true	Existential instantiation
3.	$\dfrac{P(c) \text{ for all } x}{\forall x\, P(x)}$ x shouldn't be free in any of given parameters	Universal generalization
4.	$\dfrac{P(c)}{\exists x\, P(x)}$ c is some element in universe	Existential generalization

4.5 Unification

Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.It takes two literals as input and makes them identical using substitution.Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\Psi_1\sigma = \Psi_2\sigma$, then it can be expressed as UNIFY(Ψ_1, Ψ_2).

The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).Unification is a key component of all first-order inference algorithms.It returns fail if the expressions do not match with each other.The substitution variables are called most general unifier (MGU).

Examples:

1. **Find the most general unifier (MGU) for Unify{King(x), King(John)}.**

   Let Ψ_1 = King(x), Ψ_2 = King(John),

   Substitution set θ = {John/x} is a unifier for these atoms.Applying this substitution, and both expressions will be identical.

Scanned with CamScanner

2. **Find the most general unifier (MGU) for P(x, y), and P(a, f(z)).**

   First we need to make above both statements identical to each other. For this, we will perform the substitution

   P(x,y).........(i)

   P(a, f(z))......... (ii)

   Substitute x with a, and y with f(z) in the first expression, and it will be represented as a/x and f(z)/y.With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: θ = [a/x, f(z)/y].

3. **Find the most general unifier (MGU) for P(x, f(y)) and P(a, f(g(x))).**

   Substitution set θ = [a/x, g(x)/y] = [a/x, g(a)/y]

4. **Find the most general unifier (MGU) for Q(a, g(x, a), f(y)), Q (a, g(f(b), a), x)**

   Substitution set θ= [a/a,f(b)/x, f(b)/f(y)] =[a/a, f(b)/x, b/y]

4.6 Resolution

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or causal form. There are two ways to perform resolution:

i. Answer by contradiction method
ii. Answer by extraction method

4.6.1 Answer by Contradiction Method

Algorithm;

1. Convert all the premises into FOPL.

2. Convert the FOPL into CNF, DNF form.
3. Negate given problem and make it another premise.
4. Draw resolution graph and use rules of inference to deduct new facts.
5. If any given fact contradicts, then assumed fact is false else true.

There are some formulas for converting FOPL to CNF and DNFwhich are described below.

FOL to CNF and DNF

1. implies $P \rightarrow Q \equiv \neg P \vee Q$
2. implication : $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

   if and only if : $P \leftrightarrow Q \equiv (\neg P \vee Q) \wedge (\neg Q \vee P)$
3. Use demorgan's rule to remove negation

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

   $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
4. If there is any universal quantifier then we can simply remove it.

   $\forall x\, P(x) = P(c)$
5. Eliminate the existential quantifier.

   If the existential quantifier is alone or inside universal quantifier replace all the variables associated with existential quantifier by some arbitrary constant, this process is known as skolemization.

   $\exists x \exists y\, P(x, y) \wedge P(y, z) \equiv \exists x\, P(x, A) \wedge Q(A, z)$

Problem 4.1:

Assume the following facts:

1. **$P \vee Q$**
2. **$P \rightarrow S$**
3. **$\neg S$**

Prove Q

Solution:

Step 1: Converting statements into FOL

1. $P \vee Q$
2. $P \rightarrow S$
3. $\neg S$

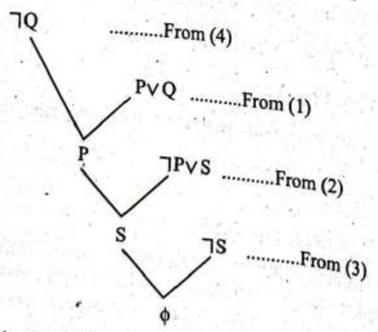
Step 2: Converting FOL into CNF

1. $P \vee Q$
2. $\neg P \vee S$
3. $\neg S$

Step 3: Negating the statement which has to be proved

4. $\neg Q$

Step 4: Drawing resolution graph



Assumed fact is contradicted with given fact while drawing resolution graph. So it is proved.

Problem 4.2:

Assume the following facts:

1. **Cats like fish.**
2. **Cats eat everything they like.**
3. **Upsa is a cat.**

Prove Upsa eats fish using resolution refutation.

Solution:

Step 1: Converting the given statements into FOL

1. $\forall x\ cat(x) \rightarrow like(x, fish)$
2. $\forall x \forall y (cat(x)) \wedge like(x, y) \rightarrow eat(x, y)$
3. cat (Upsa)

Step 2: Converting FOL into CNF

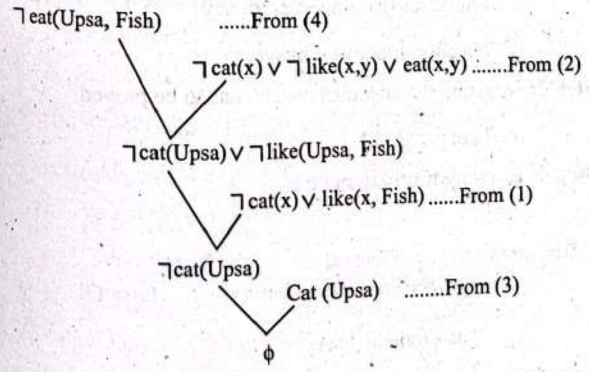
1. $\neg cat(x) \vee like(x, y)$
2. $\neg (cat(x)) \wedge like(x, y) \vee eat(x, y)$

   $\neg cat(x) \vee \neg like(x,y) \vee eat(x,y)$
3. $\neg$ cat (Upsa)

Step 3: Negating the statement which has to be proved

4. $\neg$ eat(Upsa, fish)

Step 4: Drawing resolution graph



¬eat(Upsa, Fish) ......From (4)
¬cat(x) ∨ ¬like(x,y) ∨ eat(x,y) ....From (2)
¬cat(Upsa) ∨ ¬like(Upsa, Fish)
¬cat(x) ∨ like(x, Fish) ......From (1)
¬cat(Upsa)
Cat (Upsa) ....From (3)
ϕ

Assumed fact is contradicted with given fact while drawing resolution graph. So "Upsa eats fish" is proved.

Scanned with CamScanner

Problem 4.3:

Assume the following facts:

1. **Rajesh is a megastar.**
2. **Megastars need more money.**
3. **More the money, you need to be more busy.**

Prove Rajesh is more busy.

Solution:

Step 1: Converting the given statements into FOPL

1. megastar (rajesh)
2. ∀x megastar (x) → need (x, money)
3. ∀x need(x, money) → busy (x)

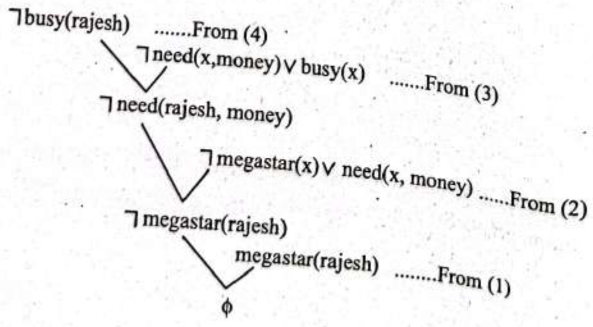
Step 2: Converting FOPL into CNF

1. megastar (rajesh)
2. ¬megastar(x) ∨ need(x, money) $\alpha \to \beta = \neg\alpha \vee \beta$
3. ¬need(x, money) ∨ busy(x)

Step 3: Negating the statement which has to be proved

4. ¬busy (rajesh)

Step 4: Drawing resolution graph


¬busy(rajesh) ....From (4)
¬need(x,money) ∨ busy(x) ....From (3)
¬need(rajesh, money)
¬megastar(x) ∨ need(x, money) ......From (2)
¬megastar(rajesh)
megastar(rajesh) ....From (1)
ϕ

Assumed fact is contradicted with given fact while drawing resolution graph. So "Rajesh is more busy" is proved.

Problem 4.4:

Assume the following facts:

1. **Ravi likes all kind of food.**
2. **Apple and chicken are food.**
3. **Anything anyone eats and is not killed by is food.**
4. **Ajay eats peanuts and is still alive.**
5. **Rita eats that Ajay eats.**

Prove Ravi likes peanuts.

Solution:

Step 1: Converting the given statements into FOPL

1. ∀x food(x) →likes(ravi, x)
2. food (apple) ∧food(chicken)
   i. food (apple)
   ii. food (chicken)
3. ∀x ∀y [eats (y, x) ∧¬killed (y)]→food(x)
4. eats(ajay, peanuts) ∧¬killed (ajay)
   i. eats(ajay, peanuts)
   ii. ¬killed (ajay)
5. ∀x eats(ajay, x) → eats(rita, x)

Step 2: Converting FOPL into CNF

1. ¬food(x) ∨ likes(ravi, x)
2. i. food (apple)
   ii. food (chicken)
3. ¬[eats(y, x) ∧¬killed(y)]∨food(x)

   [¬eats(y, x) ∨ food(x)]∧ [¬killed(y) ∨food(x)]

i. ¬eats(y, x) ∨ food(x)

ii. ¬killed(y) ∨food(x)

4. (i) eats(ajay, peanuts)

(ii)¬killed (ajay)

5. ¬eats(ajay, x) ∨ eats(rita, x)

Step 3: Negating the statement which has to be proved

6. ¬like(ravi, peanuts)

Step 4: Drawing resolution graph

¬like(ravi, peanuts) ....From (6)

¬food(x) ∨ likes(ravi,x) ....From (1)

¬food(peanuts)

¬eats(y,x)∨ ¬alive(y)∨ food(x) ......From (3)

¬eats(y,peanuts)∨ ¬alive(y)

eats(ajay,peanuts) ....From 4(ii)

¬alive(ajay)

alive(ajay) ....From 4(i)

ϕ

Assumed fact is contradicted with given fact while drawing resolution graph. So "Ravi likes peanuts" is proved.

Problem 4.5:

Assume the following facts:

a. Marcus is a man.

b. Marcus is a Pompian.

c. All Pompians are Roman.

d. Caesar is a ruler.

e. All Romans are either loyal to or hated Caesar.

f. Everyone is loyal to someone.

g. People only try to assassinate ruler if they aren't loyal.

h. Marcus tried to assassinate Caesar.

Prove Marcus hates Caesar.

Solution:

Step 1: Convert the given facts into FOL

1. man (marcus)
2. pompian (marcus)
3. ∀x :pompian (x)→ roman (x)
4. ruler (caesar)
5. ∀x : roman (x) → loyal (x, caesar) ∨ hate (caesar)
6. ∀x∀y loyal (x, y)
7. ∀x∀y assassinate(x,y) ↔¬ loyal(y,x)
8. assasinate (marcus, ceasar)

Step 3 ¬ hate (marcus, caesar)

¬hate (marcus,caesar) ....From (9)

¬ roman (x) ∨loyal(x,caesar)∨ hate(x,caesar) ....From (5)

¬roman (marcus)∨ loyal(marcus, caesar)

¬loyal (x,y)∨ assassinate (x,y) ......From (5)

¬roman (marcus)∨ assassinate (marcus, caresar)

assassinate(marcus, ceasar)........From (8)

¬roman (marcus)

¬pompian(x)∨ roman (x) ....From (3)

¬pompian(marcus)

pompian (marcus)........From (2)

ϕ

Scanned with CamScanner

Assumed fact is contradicted with given fact while drawing resolution graph. So "Marcus hates Caesar" is proved.

Problem 4.6:

Assume the following facts

1. **Everyone passing the exam and winning the lottery is happy.**
2. **Everyone how studies or lucky can pass the exam.**
3. **Ram didn't study but he is lucky.**
4. **Everyone who is lucky wins the lottery.**

Prove Ram is happy.

Solution:

Step 1: Convert the given facts into FOL

1. ∀x : (pass (x, exam)) ∧ win (x, lottery)) → happy(x)
2. ∀x : (pass (x, exam)) ∧ win (x, lottery)) → happy(x)
3. ˥ study (ram) ∧lucky (ram)
4. ∀x :lucky (x)→win (x, lottery)

Step 2: Converting FOL into CNF

1. ˥ pass (x, exam) ∨ ˥ win (x, lottery) ∨ happy (x)
2. (˥ studies (x) ∧ ˥ lucky (x)) ∨ pass (x, exam)

   (˥ study (x) ∨ pass (x, exam)) ∧ (˥ lucky (x) ∨ pass (x, exam))

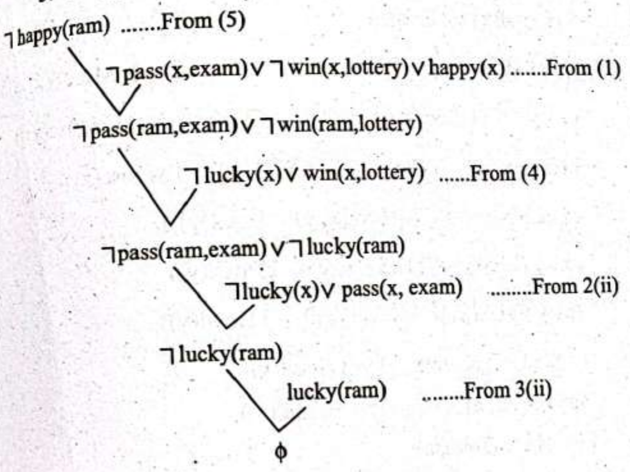
   (i) ˥ study (x) ∨ pass (x, exam)

   (ii) ˥ lucky (x) ∨ pass (x, exam)
3. (i) ˥ study (ram)

   (ii) lucky (ram)
4. ˥ lucky (x) ∨ wins (x, lottery)

Step 3: Negating the fact that has to be proved

5. ˥ happy (ram)


˥ happy(ram) ....From (5)
˥pass(x,exam) ∨ ˥ win(x,lottery) ∨ happy(x) ....From (1)
˥pass(ram,exam) ∨ ˥win(ram,lottery)
˥lucky(x) ∨ win(x,lottery) ......From (4)
˥pass(ram,exam) ∨ ˥lucky(ram)
˥lucky(x) ∨ pass(x, exam) ....From 2(ii)
˥lucky(ram)
lucky(ram) ....From 3(ii)
ϕ

Assumed fact is contradicted with given fact while drawing resolution graph. So "Ram is happy" is proved.

Problem 4.7:

Consider the following axioms.

1. **All hounds howl at night.**
2. **Anyone who has any cats will not have any mice**
3. **Light sleepers do not have anything which howls at night.**
4. **John has either a cat or a hound.**

Prove " if John is a light sleeper, then John does not have any mice" by using resolution refutation.

Solution:

Step 1: Convert the given statements into FOPL

1. ∀x (Hound(x) → Howl(x))
2. ∀x∀y (Have(x, y)∧Cat(y) → ˥ ∃z (Have(x, z) ∧ Mouse(z)))
3. ∀x(LS(x) → ˥ ∃y(Have(x, y) ∧ Howl(y)))
4. ∃x(Have(John, x) ∧ (Cat(x) ∨ Hound(x)))

Scanned with CamScanner

Step 2: Convert FOPL into CNF

1. $\neg \text{Hound}(x) \vee \text{Howl}(x)$
2. $\forall x \forall y\, (\text{Have}(x,y) \wedge \text{Cat}(y) \rightarrow \forall z \neg\, (\text{Have}(x, z) \wedge \text{Mouse}(z)))$

   $\forall x \forall y \forall z (\neg\, (\text{Have}(x,y) \wedge \text{Cat}(y)) \vee \neg\, (\text{Have}(x,z) \wedge \text{Mouse}(z)))$

   $\neg\, \text{Have}(x, y) \vee \neg\, \text{Cat}(y) \vee \neg\, \text{Have}(x, z) \vee \neg\, \text{Mouse}(z)$
3. $\forall x\, (\text{LS}(x) \rightarrow \forall y \neg\, (\text{Have}(x, y) \wedge \text{Howl}(y)))$

   $\forall x \forall y\, (\text{LS}(x) \rightarrow \neg\, \text{Have}(x, y) \vee \neg\, \text{Howl}(y))$

   $\forall x \forall y\, (\neg\, \text{LS}(x) \vee \neg\, \text{Have}(x)\, y) \vee \neg\, \text{Howl}(y))$

   $\neg\, \text{LS}(x) \vee \neg\, \text{Have}(x, y) \vee \neg\, \text{Howl}(y)$
4. $\text{Have}(\text{John}, a) \wedge (\text{Cat}(a) \vee \text{Hound}(a))$
   - i. Have (John, a)
   - ii. $\text{Cat}(a) \vee \text{Hound}(a)$

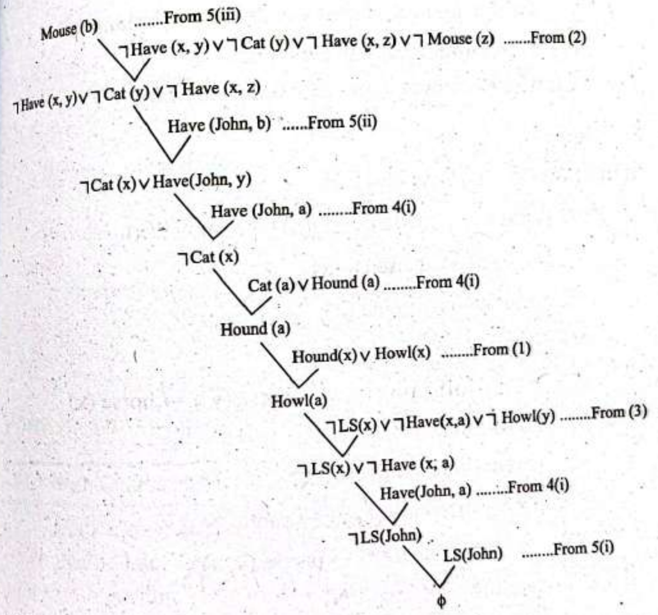
Step 3: Negating the statement that has to be proved.

5. $\neg\, [\text{LS}(\text{John}) \rightarrow \neg\, \exists y (\text{Have}(\text{John}, z) \wedge \text{Mouse}(z))]$

   $\neg\, [\neg\, \text{LS}(\text{John}) \vee \neg\, \exists z (\text{Have}(\text{John}, z) \wedge \text{Mouse}(z))]$

   $\text{LS}(\text{John}) \wedge \exists z (\text{Have}(\text{John}, z) \wedge \text{Mouse}(z)))$

   $\text{LS}(\text{John}) \wedge \text{Have}(\text{John}, b) \wedge \text{Mouse}(b)$
   - i. LS(John)
   - ii. Have(John, b)
   - iii. Mouse(b)


Mouse (b) ....From 5(iii)
¬Have (x, y) ∨ ¬ Cat (y) ∨ ¬ Have (x, z) ∨ ¬ Mouse (z) ....From (2)
¬Have (x, y) ∨ ¬ Cat (y) ∨ ¬ Have (x, z)
Have (John, b) ......From 5(ii)
¬Cat (x) ∨ Have(John, y)
Have (John, a) ....From 4(i)
¬Cat (x)
Cat (a) ∨ Hound (a) ....From 4(i)
Hound (a)
Hound(x) ∨ Howl(x) ....From (1)
Howl(a)
¬LS(x) ∨ ¬Have(x,a) ∨ ¬ Howl(y) ....From (3)
¬LS(x) ∨ ¬ Have (x; a)
Have(John, a) ....From 4(i)
¬LS(John)
LS(John) ....From 5(i)
ϕ

Assumed fact is contradicted with given fact while drawing resolution graph. So "if John is a light sleeper, then John does not have any mice" is proved.

4.6.2 Answer by Extraction Method

In extraction method, we start with given facts or statments and reach the conclusion. We don't need to negate the statment that has to be proved.

Problem 4.8:

Assume the following facts:

- **i. Horses, cow, pigs are mammals**
- **ii. An offspring of a horse is a horse**
- **iii. Bluebeard is a horse**
- **iv. Bluebeard is Charlie's parent**

Scanned with CamScanner

v. **Offspring and parent are inverse relations**

vi. **Every mammal has a parent**

Prove Charlie is a horse using extraction method

Solution:

Step 1: Converting facts into FOL

$\forall x$ horse (x) ∨ cow(x) ∨ Pig(x) → mammal(x)

$\forall x$ horse (x) → mammal (x)

$\forall x$ cow (x) → mammal (x)

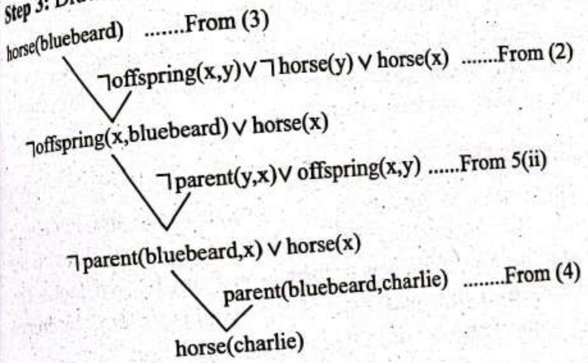
$\forall x$ pig (x) → mammal (x)

2. $\forall x \forall y$ [offspring (x, y) ∧ horse (y)] → horse (x)
3. horse (bluebeard)
4. parent (bluebeard, charlie)
5. $\forall x \forall y$ offspring (x, y) ↔ parent (y, x)
6. $\forall x \forall y$ mammal (x) → parent (y, x)

Step 2: Converting FOL into CNF

1. i. ¬ horse (x) ∨ mammal (x)
   ii. ¬ cow (x) ∨ mammal (x)
   iii. ¬ pig (x) ∨ mammal (x)
2. ¬ [offspring (x, y) ∧ horse (y)] ∨ horse (x)
   ¬ offspring (x, y) ∨ ¬ horse (y) ∨ horse (x)
3. horse (bluebeard)
4. parent (bluebeard, charlie)
5. [offspring (x, y) →parent (y, x)] ∧ [parent (y, x) → offspring (x, y)]
   i. ¬ offspring (x, y) ∨ parent (y, x)]
   ii. ¬ parent (y, x) ∨ offspring (x, y)]
6. ¬ mammal (x) ∨ parent (y, x)

Step 3: Draw resolution graph


horse(bluebeard) ....From (3)
¬offspring(x,y)∨¬horse(y) ∨ horse(x) ....From (2)
¬offspring(x,bluebeard) ∨ horse(x)
¬parent(y,x)∨ offspring(x,y) ......From 5(ii)
¬parent(bluebeard,x) ∨ horse(x)
parent(bluebeard,charlie) ....From (4)
horse(charlie)

Hence, we proved "Charlie is a horse".

4.7 Horn Clause and Definite Clause

Horn clause and *definite clause* are the forms of sentences, which enable knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require knowledge base in the form of the first-order definite clause.

Definite clause: A clause which is a disjunction of literals with exactly one positive literal is known as a *definite clause* or *strict horn clause*. Consider an example, (¬p ∨¬q∨ k). It has only one positive literal k. So, it is a definite clause.

Horn clause: A clause which is a disjunction of literals with atmost one positive literal is known as *horn clause*. Hence all the definite clauses are horn clauses. Consider examples, (¬p ∨¬q∨ k) and (¬p ∨ ¬q). Both of these FOL are horn clauses.

4.8 Rule-Based Deduction System

A *rule-based deduction system* is a set of "if-then" statements that uses a set of assertions, to which rules on how to act upon those assertions are created. Rule-based systems can be used to create software that will provide an answer to a problem in

Scanned with CamScanner

place of a human expert. These type of systems are also called *expert systems*. Rule-based systems are can be designed using forward chaining or back chaining. Before understanding forward and backward chaining, let us first understand inference engine from where these two terms came.

Inference Engine

The *inference engine* is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:

i. Forward chaining
ii. Backward chaining

4.8.1 Forward Chaining

Forward chaining is also known as a *forward deduction* or *forward reasoning method* when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties of forward chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.

- Forward-chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Consider the following famous example which we will use in both approaches:

Problem 4.9:

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

Prove that "Robert is criminal" using forward chaining.

Solution:

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts conversion into FOL:

- It is a crime for an American to sell weapons to hostile nations.

  Let's say p, q, and r are variables.

  $\forall p\ \forall q\ \forall r$ (American (p) $\wedge$ weapon(q) $\wedge$ sells (p, q, r) $\wedge$ hostile(r))$\rightarrow$ Criminal(p) ......(1)

- Country A has some missiles.

  $\forall p$ Owns(A, p) $\wedge$ Missile(p). It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

  Owns(A,T1) .....(2)
  Missile(T1) .....(3)

- All of the missiles were sold to country A by Robert. ......(4)
  $\forall p$Missiles(p)$\wedge$Owns(A,p) $\rightarrow$Sells(Robert, p, A)

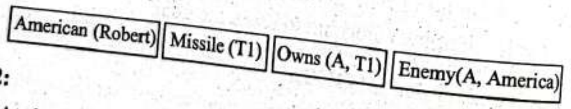
- Missiles are weapons.
  $\forall p$ Missile(p) $\rightarrow$ Weapons(p) ....(5)

- Enemy of America is known as hostile.
  Enemy(p, America) →Hostile(p) ....(6)
- Country A is an enemy of America.
  Enemy (A, America) ....(7)
- Robert is American.
  American(Robert). ....(8)

Forward chaining proof:

Step 1:

In the first step, we will start with the known facts and will choose the sentences which do not have implications, such as: American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1). All these facts will be represented as below.



Step 2:

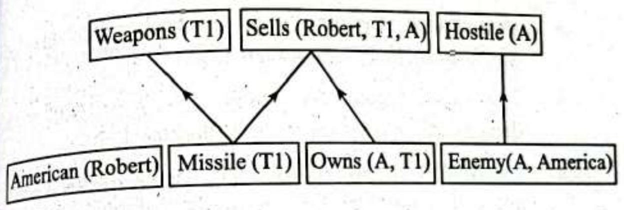
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

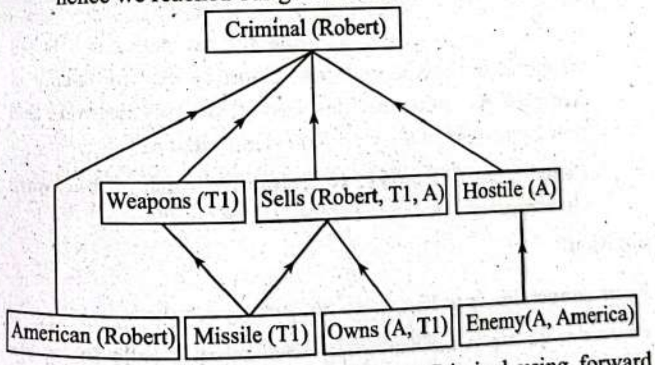
Rule-(4) satisfy with the substitution {p/T1}, so Sells (Robert, T1, A) is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



Step 3:

At step-3, as we can check Rule-(1) is satisfied with the substitution {p/Robert, q/T1, r/A}, so we can add Criminal(Robert) which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is Criminal using forward chaining approach.

4.8.2 Backward Chaining

Backward chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties of backward chaining:

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.

Scanned with CamScanner

- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward-chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- The backward-chaining method mostly uses a depth-first search strategy for proof.

Problem 4.10:

"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

Prove that "Robert is criminal" using backward chaining.

Solution:

Facts conversion into FOL:

- ∀p ∀q ∀r American (p) ∧ weapon(q) ∧ sells (p, q, r)∧ hostile(r)→ Criminal(p) ....(1)
- Owns(A, T1) ....(2)
- Missile(T1) ....(3)
- ∀p Missiles(p)∧Owns(A, p)→Sells(Robert, p, A) ......(4)
- Missile(p) → Weapons (p) ....(5)
- Enemy(p, America) →Hostile(p) ....(6)
- Enemy (A, America) ....(7)
- American(Robert) ....(8)

Backward chaining proof:

In backward chaining, we will start with our goal predicate, which is Criminal(Robert), and then infer further rules.

Step 1:

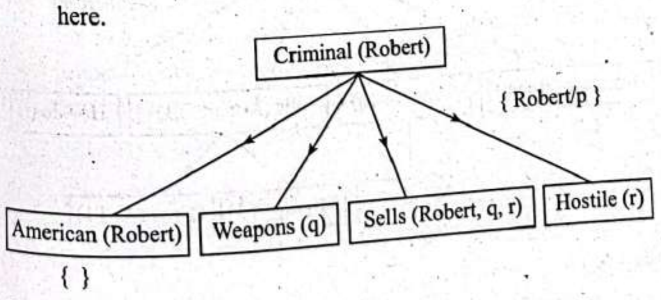
At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

$\boxed{\text{Criminal (Robert)}}$

Step 2:

At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.

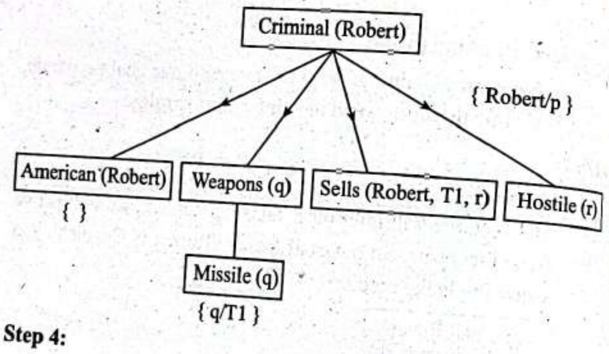
Here we can see American (Robert) is a fact, so it is proved here.



Step 3:

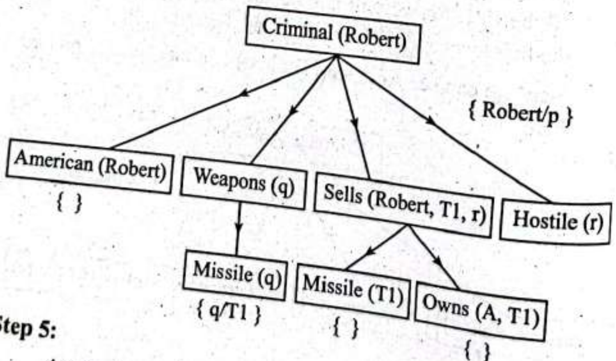
At this step, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.

Scanned with CamScanner



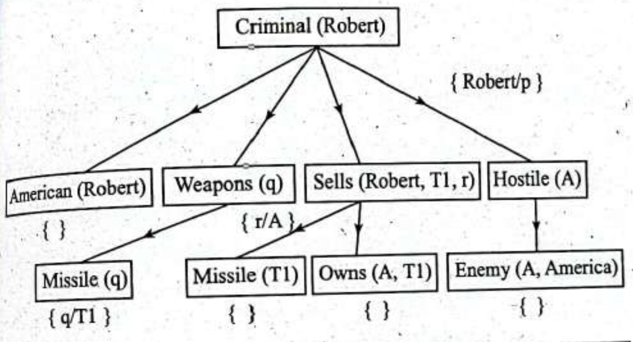
Step 4:

At step 4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the Rule- 4, with the substitution of A in place of r. So these two statements are proved here.



Step 5:

At step5, we can infer the fact Enemy (A, America) from Hostile(A) which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.



4.9 (Probabilistic) Statistical Reasoning

Reasoning is the process by which we use the knowledge and we have to draw conclusion or infer something new about a domain or interest. In the logic-based approach, we have assumed that everything is either true or false. However, it is often that the fact are probably true with probability 0.5, 0.6, etc. This is useful for dealing with problems where there is randomness or unpredictability.

Simply probability deals with unconditional events if we know probability of even A as P(A) and probability of event B as P(B), then probability that both occur is represented by

$$P(A \cap B) = P(A) \times P(B)$$

If two events are conditional or interdependent, the outcome of one affects that of other, then probability that both occur is represented by

$$P(A \cap B) = P(B) \times P\left(\frac{A}{B}\right).$$

Problem 4.11:

There are two bags A, B. Probability of selecting bag A and bag B are 0.7 and 0.3 respectively. Bag A contains 4 red balls and 2 blue balls whereas bag B contains 2 red balls and 4 blue balls. Find the probability of

Scanned with CamScanner

i. **Getting red ball given bag A**

ii. **Getting red ball from bag A**

iii. **Getting red ball**

iv. **Getting red ball from A if selected ball is red.**

Solution:

$\boxed{3R, 2B}$ $\qquad$ $\boxed{2R, 4B}$

P(A) = 0.7 $\qquad$ P(B) = 0.3

Probability of getting red ball given from A,

P(R/A) = 3/5 = 0.6

Probability of getting red ball from A,

$P(R\cap A) = P(A) \times P(R/A)$

$= 0.7 \times 0.6 = 0.42$

Probability of getting red ball,

$P(R) = P(A)\times P(R/A) + P(B) \times P(R/B)$

$= 0.7 \times 0.6 + 0.3 \times \frac{3}{7} = 0.5486$

Probability of getting red ball from A if selected ball is red

$$= \frac{P(A\cap R)}{P(A\cap R) + P(B\cap R)}$$

$$= \frac{P(A)\times P(R/A)}{P(A) \times P(R/A) + P(B) \times P(R/B)}$$

$$= \frac{0.7}{0.7\times\frac{3}{5} + 0.3\times\frac{3}{7}}$$

$$= 0.76$$

Problem 4.12:

Nepal is playing a cricket match with Bangladesh. The probability of winning the toss for Nepal is 0.7. If Nepal wins the toss then chances of winning the match is 80%. If Nepal loss the toss then the chances of winning the game is just 30 %. Find the probability of

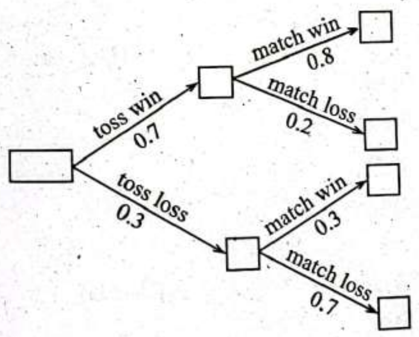
i. **winning the match**

ii. **winning the toss if match is already won.**

Solution:

P(T), P(~T) be the probability of winning the toss and not winning the toss respectively.

Similarly P(M) & P(~M) be the probability of winning the match and not winning the match respectively.



i. Probability of winning game

$= P(T) \times P(M/T) + P(\sim T) \times P(M/\sim T)$

$= 0.7 \times 0.8 + 0.3 \times 0.3 = 0.65$

ii. Probability of winning the toss if match is already won

$$= \frac{P(T) \times P(M/T)}{P(T) \times P(M/T) + P(\sim T) \times P(M/\sim T)}$$

$$= \frac{0.7 \times 0.8}{0.7 \times 0.5 + 0.3 \times 0.3} = 0.86$$

4.10 Bayesion Network/Boliet/Directed Arrow Graph

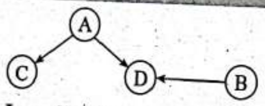
If node x_i has no parent, then its probability is said to be *unconditional* and it is written as P(x). The node having parents are called *conditional node* and probability of conditional node is written as:

Scanned with CamScanner

$P(x_i/\text{parent}(x_i))$

If value of node is observed, the node is *evidence node.*

Problem 4.13:



In a directed arrow graph, the probability of occurring different events are listed below:

P(A) = 0.3, P(B) = 0.7, P(C/A) = 0.4, P(C/~A) = 0.3, P(D/A,B) = 0.7, P(D/~A,B) = 0.2, P(D/A,~B) = 0.3, P(D/~A,~B) = 0.01.

Find probability of

i) all true

ii) all true except A

Solution:

P(all true) = P(A) × P(B) × P(C/A) × P(D/A, B)

= 0.0588

P(all true except A)

= P(B) × P(C/~A) × P (D/~A, B) × D(~A)

= 0.7 × 0.3 × 0.2 × (1 − 0.3)

= 0.0294

Problem 4.14:

In a house there is an alarm system which rings if burglary or earthquake occurs with probability as shown in the table 1 below. The two neighbours P_1 and P_2 calling if the alarm rings and doesn't ring are shown in the table 2 and 3 respectively. If the chances of burglary and Earthquake are 0.01 and 0.02 respectively. Find the probability of

i. $P_1$and P_2 call when no burglary & earthquake if alarm is ringing

ii. $P_1$and P_2 call when no burglary & earthquake if alarm isn't ringing

iii. All true

Table 1

B	E	P(A)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

Table 2

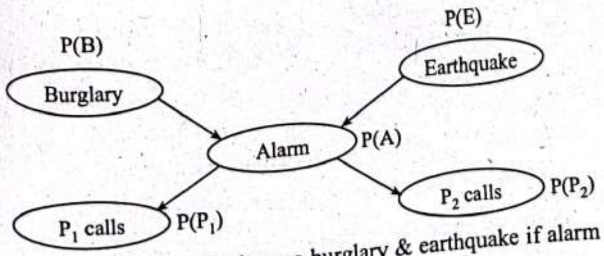
A	P_1
T	0.9
F	0.05

Table 3

A	P_2
T	0.7
F	0.01

Solution:

P(B) = 0.01, P(B) = 0.002



i. P(P_1 and P_2 call when no burglary & earthquake if alarm is ringing)

= $P(\sim B)\times P(\sim E)\times P(A/\sim B,\sim E)\times P(P_1/A)\times P(P_2/A)$

Scanned with CamScanner

$= 0.999 \times 0.998 \times 0.001 \times 0.9 \times 0.7$

$= 6.281 \times 10^{-4}$

ii. P(P_1 and P_2 call when no burglary & earthquake if alarm isn't ringing)

$= P(\sim B) \times P(\sim E) \times P(\sim A/\sim B,\sim E) \times P(P_1/\sim A) \times P(P_2/\sim A)$

$= 0.999 \times 0.998 \times 0.999 \times 0.05 \times 0.01$

$= 4.9 \times 10^{-4}$

iii. All true

$= P(E) \times P(B) \times P(A/E,B) \times P(P_1/A) \times P(P_2/A)$

$= 0.002 \times 0.001 \times 0.95 \times 0.9 \times 0.7$

$= 1.197 \times 10^{-6}$

EXERCISE

1. Translate the following sentences into FOPL:
   i. Ram likes all kind of foods.
   ii. Shyam likes everything john likes.
   iii. Fruits and vegetables are delicious.
   iv. God help those who help themselves.
   v. All students like good teachers.
   vi. All that glitter is not gold.
   vii. Some employees are sick today.
   viii. All employee earning Rs 400000 or more per year pay taxes.
   ix. You can fool some of the people all the time.
   x. Either Krishna or curiosity killed Kalpana.
2. What are merits and limitation of propositions.
3. Differentiate between propositions and predicate logic.
4. Write short notes on:
   a) Horn clause
   b) Resolution

   c) Unification
   d) Well-formed formula
   e) Skolemization
   f) Forward chaining and backward chaining
   g) Causal net
5. All married employees earning Rs 4,50,000 or more per year in Nepal pay taxes. All unmarried employees earning Rs 4,00,000 or more per year in Nepal pays taxes. The president of Nepal earns Rs 51,00,000 and has to may maximum taxes. No other employees earn more than the president. Some of the Nepalese citizens earn less than Rs 100 per day and they don't have to pay any taxes. Represent these sentences in First order logic and explain each step.
6. All oversmart persons are stupid. Children of oversmart persons are naughty. Ram is children of Hari. Hari is oversmart. Show that Ram is naughty. Using FOPL based resolution method.
7. In a village 1% of people have a certain genetic defect. 90% of test for gene defected people detect the defect. 9.6% of the test detect the positive result even if the person has no gene defect. If a person gets a positive test result, what are the odds they actually have the genetic defect?

Scanned with CamScanner

Chapter-5

STRUCTURED KNOWLEDGE REPRESENTATION

5.1 Introduction

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

5.2 Knowledge Model

A *model* is a world in which a sentence is true under a particular interpretation. There can be several models at once that have the same interpretation. First order predicate logic consists of objects, predicates on objects, connectives, and quantifiers. *Logic* is simple representation which helps to infer new fact from the existing information. *Predicates* are the relations between objects or properties of the objects. *Connectives* and *quantifiers* allow for universal sentences. Relation between objects can be true or false.

E.g., propositional logic.

We had already discussed logic representation in detail in chapter 4. Here in this chapter we focus much in structured knowledge representation namely semantic network, frame and scripts.

5.3 Semantic Network

Semantic network is an alternative to predicate logic as a form of knowledge representation. Semantic network is a declarative graphic representation that can be used to represent knowledge and support automated systems for reasoning about the knowledge. The structure of a semantic net is shown graphically in terms of nodes and the arcs connecting each other. Nodes are sometimes referred to as objects, events, subjects while arcs

represent the links or relations. The links are used to express relationships.

It is argued that this form of representation is closer to the way human's structure the knowledge as human also store the knowledge with relational link. For example; if a man lost bicycle key then he remembers where he had been, then he had used his bicycle last time etc, making this kind of relating scenario he goes to search those places.

Advantages of semantic web/network:

- This method is easy to visualize.
- It is efficient in space requirement.
- The objects are represented only once.

Disadvantages of semantic web/network:

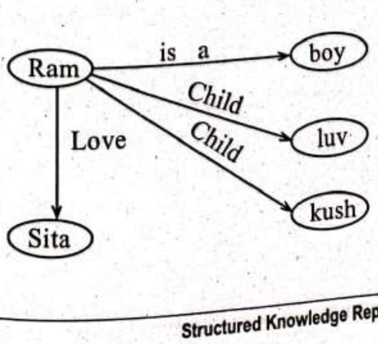
- It is unable to represent negation, quantification, disjunction etc.
- We cannot infer or deduct new information.

Example 5.1:

Show the following statements in semantic network.

1. **Ram is a boy.**
2. **Ram loves Sita.**
3. **Ram's children are luv and kush.**

Solution:



Example 5.2:

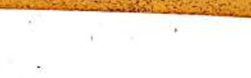
Show the following statements in semantic network

1. **Tom is a cat.**
2. **Tom caught a bird.**
3. **Tom is owned by John.**
4. **Tom is ginger in color.**
5. **Cats like cream.**
6. **The cat sat on the mat.**
7. **A cat is a mammal.**
8. **A bird is an animal.**
9. **All mammals are animals.**
10. **Mammals have fur.**

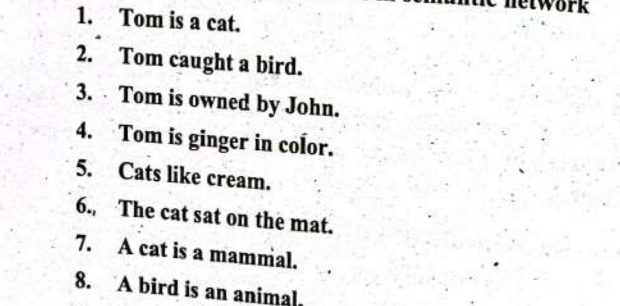
Solution:

We can make individual diagram with ease.

Tom is a cat

Tom caught a bird 

Representing all in a single picture we get the following diagram.

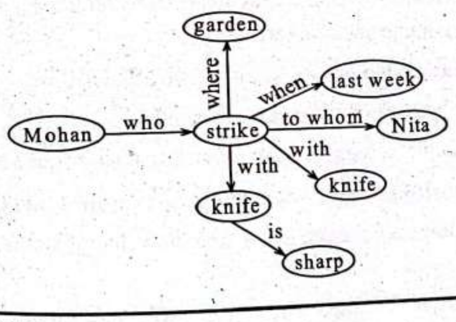


Example 5.3:

Show the following statement in semantic network.

Mohan struck Nita in the garden with a sharp knife last week.

Solution:



5.4 Frames

Frame is a static data structure used to represent well understood situation in a group of slots and slot fillers. Frame is similar to a record structure. It is used in many AI applications including vision and natural language processing that provides a convenient structure for representing. A single frame is not much useful. Frame systems usually have collection of frames connected to each other. Frames are also useful for representing commonsense knowledge. While semantic nets are basically a two-dimensional representation of knowledge, frames add a third dimension by allowing nodes to have structures. By using frame structure, we can use filler slots and inheritance, very powerful knowledge representation systems can be built. Frame-based expert systems are very useful for representing causal knowledge because their information is organized by cause and effect. Frames are generally designed to represent either generic or specific knowledge.

Scanned with CamScanner

Frame structure contains following information:

- **Frame identification name**

  It is field written in top of the frame structure where name of the frame is placed.

  Example: A frame which stores knowledge about a car can have frame name as car.

- **Relationship of this frame to the other frame**

  It relates different frame to each other.

  Example: A superclass of a frame (car) is a frame (vehicle).

- **Knowledge about an attribute of an object and its value**

  Attribute are written in slot and the value of slot is written in slot filler.

  Example: A frame (car) can have an attribute as no. of wheels with value 4.

- **Frame default information**

  These are slots values that are taken to be true when no evidence to the contrary of frame has been formed.

  Example: 'Ram's car' frame copies the slot and slot values of its parent frame 'car' like no of wheel, model name etc.

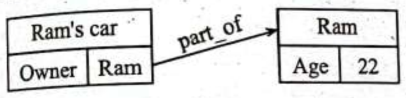
5.4.1 Types of Frame

1. **Class frame:** It is the main frame from which other frames can be inherited.
2. **Subclass frame:** It is frame inherited fromclass frame.
3. **Instance frame:** This frame is bottom frame from which no other frame can be derived. This frame may be derived from both subclass and class frame.

5.4.2 Types of Relationship

1. **Is-a relationship:** It relates subclass frame with a class frame or an instance frame with a subclass or class frame. In this case a subclass frame or instance frame inherits all slots from a class frame and it can also include new slots.

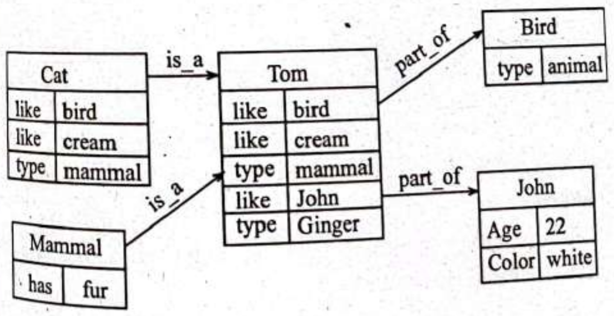
2. **Part of relationship:** It relates the slot values with its constituent parts:



3. **Semantic relationship:**

   It relates object with its attributes and frame structure can be represented in semantic network.

   Solution for the same example in frame can be shown in frame structure as shown below:



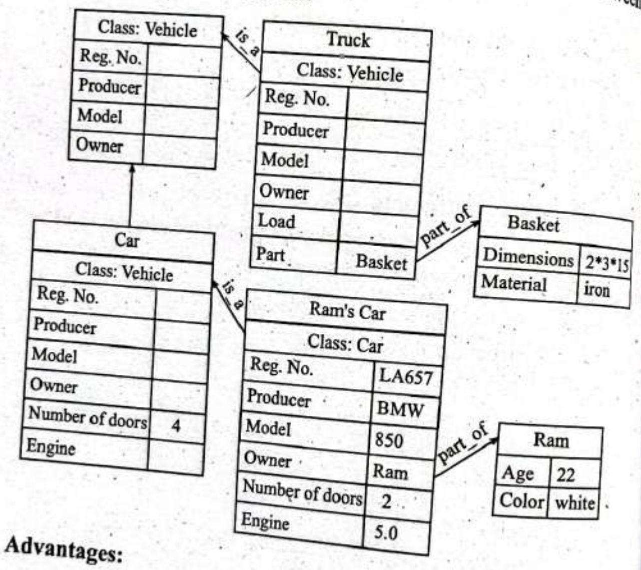
5.4.3 Frame Methods

Methods in frame are also called *demons* which are attached to slots. Demons are automatically invoked when a slot is accessed. Standard demons are:

1. **IF NEEDED:** It is invoked when it is necessary to acquire a slot value.
2. **IF CHANGED:** It is invoked when a value of slot is changed.
3. **IF ADDED:** It is invoked when a value is addedto a slot.
4. **IF REMOVED:** It is invoked when value of slot is deleted.

Scanned with CamScanner

Example below shows relations between relations between class, subclass, instance frame.



Advantages:

1. It makes programming easier by grouping related knowledge together.
2. Easy to setup new slots or new properties and relations.
3. Easy to include default information and detect missing values.

Disadvantages:

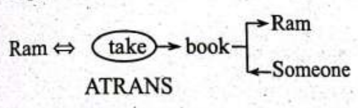
1. Frame for a room will be completely different person.
2. No associated inference mechanism.

5.5 Conceptual Dependency and Script

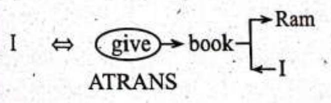
Conceptual dependency is a theory in which we try to be independent of the words. For any two or more sentences that are identical in meaning, there should be only one representation.A

script is a remembered precedent, consisting of tightly coupled, expectation-suggesting primitive-action and state change frames. A script is a structured representation describing a stereotyped sequence of events in a particular context i.e., extend frames by explicitly representing expectations of actions and state changes. Find primitives to describe the world like PTRANS for "transfer physical location of an object (= go)" and ATRANS for "transfer a relationship (= give)".

E.g., Ram took a book from someone.



I give a book to Ram.



Conceptual dependency provides:

- a structure into which nodes representing information can be placed
- a specific set of primitives
- at a given level of granularity.
- Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.
- The agent and the objects are represented.
- The actions are built up from a set of primitive acts which can be modified by tense.

Examples of primitive acts are:

- ATRANS - Transfer of an abstract relationship. e.g. give, take, lend, borrow.
- PTRANS - Transfer of the physical location of an object. e.g. go, went.

Scanned with CamScanner

- PROPEL - Application of a physical force to an object. e.g. push, pull.
- MTRANS - Transfer of mental information. e.g. tell.
- MBUILD - Construct new information from old. e.g. decide, conclude.
- SPEAK - Utter a sound. e.g. say, talk, play music.
- ATTEND - Focus a sense on a stimulus. e.g. listen, watch.
- MOVE - Movement of a body part by owner. e.g. punch, kick.
- GRASP - Actor grasping an object. e.g. clutch, collect.
- INGEST - Actor ingesting an object. e.g. eat, drink.
- EXPEL - Actor getting rid of an object from body. e.g. throw, sweat, cry, excrete.

Script is originally developed to represent knowledge acquired from natural language input. To implement script, we use conceptual dependency.

A script is composed of:

1. Entry condition must be true for further execution of the script..
2. Results or facts are true one the script is executed.
3. Props or things that make up the content of the script.
4. Roles are the actions that the individual participant performs.
5. Scene which represent temporal aspect of script.
6. Tracks are the variation on script. Different trick may share different component of same script.

E.g.,

Entry condition:

1. Man is Hungry.
2. Restaurant should be open.

Result:

1. Man has less money.
2. Man is less hungry.

Props:

1. Table (T)
2. Menu (ME)
3. Glass (G)
4. Door (D)
5. Chair (C)
6. Table (T)
7. Restaurant (R)

Roles:

1. Customer (M)
2. Waiter (W)
3. Chef (Ch)

Scene 1: Get to table

M PTRANS to D

M PROPEL D

M ATTEND T

M MBUILD For T

Scene 2: Eat the food

Track 1: for menu in the table

M ATTEND ME

M MBUILD F

M SPEAK to W

M ATRANS F from W

M INGEST F

Track 2: for menu not in the table

M SPEAK with W for ME

M ATRANS ME from W

M ATTEND ME

M MBUILD F

Scanned with CamScanner

M SPEAK to W

M ATRANS F from W

M INGEST F

Scene 3: Exit from the restaurant

M MOVE

M PTRANS to D

M PROPEL D

M PTRANS from R

EXERCISE

1. Elaborate the concept of semantic network.
2. What is frame? Explains its type, class, methods.
3. How are scripts written? Illustrate it with example.
4. What are the importance of the conceptual dependency in artificial intelligence?

Chapter 6

MACHINE LEARNING

6.1 Introduction

Learning is the process of acquiring new and better or updated information from existing knowledge, behaviors, skills, values, or preferences. The ability to learn is possessed by humans, animals, and some machines; there is also evidence for some kind of learning in some plants. Some learning is immediate, induced by a single event (e.g., being burned by a hot stove), but much skill and knowledge accumulates from repeated experiences. The changes induced by learning often last a lifetime, and it is hard to distinguish learned material that seems to be "lost" from that which cannot be retrieved.

Learning is the ability of an agent to improve its behavior based on experience. This could mean the following:

- The range of behaviors is expanded; the agent can do more.
- The accuracy on tasks is improved; the agent can do things better.
- The speed is improved; the agent can do things faster.

The ability to learn is essential to any intelligent agent. Learning involves an agent remembering its past in a way that is useful for its future. This chapter considers supervised learning i.e., given a set of training examples made up of input-output pairs, predict the output of a new input.

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender

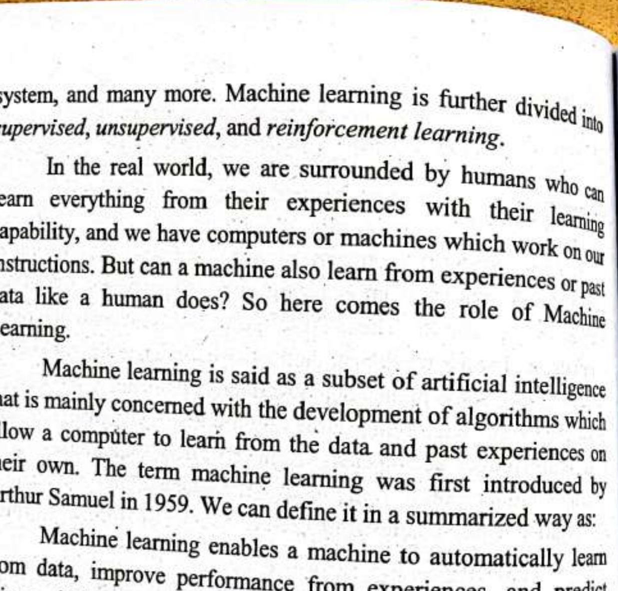
Scanned with CamScanner

system, and many more. Machine learning is further divided into *supervised*, *unsupervised*, and *reinforcement learning*.

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

Machine learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. We can define it in a summarized way as:

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.


Human
I can learn everything automatically from experiences. Can u learn?
Machine
Yes, I can also learn from past data with the help of Machine learning

With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science

and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

A machine has the ability to learn if it can improve its performance by gaining more data.

How does Machine Learning Work?

A machine learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The block diagram below explains the working of machine learning algorithm:

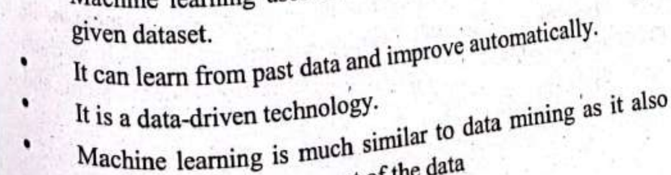


Figure 6.1: Steps involved in machine learning algorithm.

Features of machine learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data

Scanned with CamScanner

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases. Currently, machine learning is used in self-driving cars, cyber fraud detection, face recognition, and friend suggestion by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyze the user interest and recommend product accordingly.

Following are some key points which show the importance of machine learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data.

Now machine learning has got a great advancement in its research, and it is present everywhere around us, such as self-driving cars, Amazon Alexa, Chatbots, recommender system, and many more.

Modern machine learning models can be used for making various predictionsincluding weather prediction system, disease prediction system, stock market analysis, etc.

6.2 Learning Types

Learning is one of the fundamental building blocks of artificial intelligence (AI) solutions. From a conceptual standpoint, learning is a process that improves the knowledge of an AI program by making observations about its environment. From a technical/mathematical standpoint, learning processes focused on processing a collection of input-output pairs for a specific function and predicts the outputs for new inputs.

To understand the different types of AI learning models, we can use two of the main elements of human learning processes: knowledge and feedback. From the knowledge perspective, learning models can be classified based on the representation of input and output data points. In terms of the feedback, AI learning models can be classified based on the interactions with the outside environment, users and other external factors.

6.2.1 Based on Feedback Type

Based on feedback type, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

1. **Supervised Learning**

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output.

The system creates a model using labeled data to understand the datasets and learn about each data, once the training and

Scanned with CamScanner

processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is spam filtering.

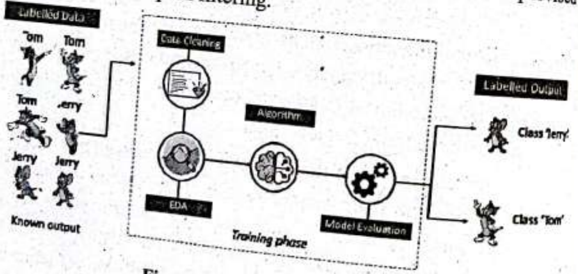


Figure 6.2: Supervised learning

We have initially taken some data and marked them as 'Tom' or 'Jerry'. This labeled data is used by the training supervised model, this data is used to train the model.

Once it is trained we can test our model by testing it with some test new mails and checking of the model can predict the right output.

Supervised learning can be grouped further in two categories of algorithms:

- **Regression:** It is a type of problem where the output variable is a real value, such as "dollars" or "weight".
- **Classification:** It is a type of problem where the output variable is a category, such as "red" or "blue" or "disease" and "no disease".

2. **Unsupervised Learning**

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data.

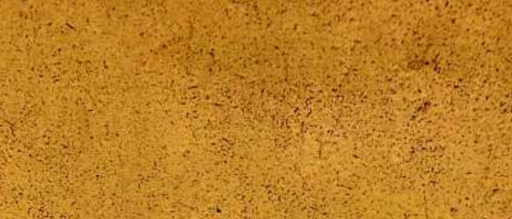


Figure 6.3: Supervised learning

We have given some characters to our model which are 'Ducks' and 'Not Ducks'. In our training data, we don't provide any label to the corresponding data. The unsupervised model can separate both the characters by looking at the type of data and models the underlying structure or distribution in the data to learn more about it.

It can be further r classifieds into two categories of algorithms:

- **Clustering:** A clustering problem is where we group similar data according to a pattern in data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where we want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Example 1:

An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc.) of newly admitted patients. A

Scanned with CamScanner

decision is needed: whether to put a new patient in an intensive-care unit. Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.

Problem: To predict high-risk patients and discriminate them from low-risk patients.

Example 2: A credit card company receives thousands of applications for new cards. Each application contains information about an applicant, age, and marital status, annual salary, outstanding debts, credit rating etc.

Problem: To decide whether an application should be approved, or to classify applications into two categories, approved and not approved.

3. **Reinforcement Learning**

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

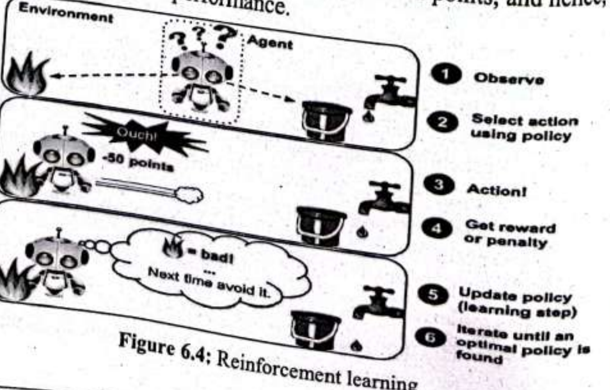

Environment
Agent
Ouch!
-50 points
= bad!
Next time avoid it.
1 Observe
2 Select action using policy
3 Action!
4 Get reward or penalty
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Figure 6.4: Reinforcement learning

The agent is given 2 options i.e., a path with water or a path with fire.

A reinforcement algorithm works on reward a system i.e. if the agent uses the fire path then the rewards are subtracted and the agent tries to learn that it should avoid the fire path.

If it had chosen the water path or the safe path then some points would have been added to the reward points, the agent then would try to learn what path is safe and what path isn't.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

6.2.2 AI Learning Models: Knowledge-Based Classification

AI learning models can be classified in two main types considering its representation of knowledge; inductive and deductive.

1. **Inductive Learning**

This type of AI learning model is based on inferring a general rule from datasets of input-output pairs. Algorithms such as knowledge based inductive learning (KBIL) are a great example of this type of AI learning technique. KBIL focused on finding inductive hypotheses on a dataset with the help of background information.

2. **Deductive Learning**

This type of AI learning technique starts with the series of rules and infers new rules that are more efficient in the context of a specific AI algorithm. Explanation-based learning (EBL) and relevance-based Learning (RBL) are examples of deductive techniques. EBL extracts general rules from examples by "generalizing" the explanation. RBL focuses on identifying attributes and deductive generalizations from simple example.

Scanned with CamScanner

6.3 Machine Learning Framework

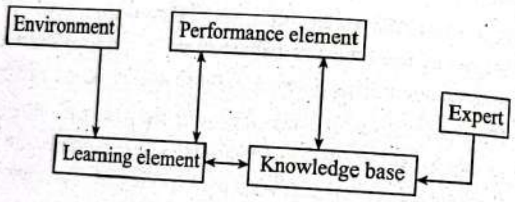


Figure 6.5: Learning framework

1. **Environment**

The *environment* refers the nature and quality of information given to the learning element. The nature of information depends on its level (the degree of generality with respect to the performance element). High level information is abstract, it deals with a broad class of problems. Low level information is detailed, it deals with a single problem, the quality of information involves noise free, reliable and ordered.

2. **Learning Elements**

Learning elements is the component that deals with the how the system should be trained so that this element can predict the output for new data. Learning elements are of four types on the basis of learning situations:

i. **Rote learning:** In this situation environment provides information at the required level

ii. **Learning by being told:** Here, information is too abstract, the learning element must hypothesize missing data.

iii. **Learning by example:** In this situation, information is too specific, the learning element must hypothesize more general rules.

iv. **Learning by analogy:** Information provided is relevant only to an analogous task, the learning element must discover the analogy.

3. **Knowledge Base**

Knowledge base is the main component in framework as it stores the rules that help to maintain the database required for making learning element. Knowledge acquired from the expert are maintained here. Knowledge base should have following features:

i. **Expressive:** The representation of knowledge in knowledge base contains the relevant knowledge in an easy fashion.

ii. **Modifiable:** The knowledge base must be easy to change the data in the knowledge base.

iii. **Extendibility:** The knowledge base must contain meta-knowledge (knowledge on how the data base is structured) so the system can change its structure.

4. **Performance Element**

Performance element is the component that deals with the evaluation of the algorithm designed. It helps to feedback the error occurred and gives the accuracy data to the learning component. It has following key features:

i. **Complexity:**For learning, the simplest task is classification based on a single rule while the most complex task requires the application of multiple rules in sequence

ii. **Feedback:**The performance element must send information to the learning system to be used to evaluate the overall performance

iii. **Transparency:** The learning element should have access to all the internal actions of the performance element.

Scanned with CamScanner

6.4 Genetic Algorithm

A *genetic algorithm* is a heuristic search method used in artificial intelligence. It is used for finding optimized solutions to search problems based on the theory of natural selection and evolutionary biology. Genetic algorithms are excellent for searching through large and complex data sets. They are considered capable of finding reasonable solutions to complex issues as they are highly capable of solving unconstrained and constrained optimization issues.

A genetic algorithm makes uses of techniques inspired from evolutionary biology such as selection, mutation, inheritance and recombination to solve a problem. The most commonly employed method in genetic algorithms is to create a group of individuals randomly from a given population. The individuals thus formed are evaluated with the help of the evaluation function provided by the programmer. Individuals are then provided with a score which indirectly highlights the fitness to the given situation. The best two individuals are then used to create one or more offspring, after which random mutations are done on the offspring. Depending on the needs of the application, the procedure continues until an acceptable solution is derived or until a certain number of generations have passed.

A genetic algorithm differs from a classical, derivative-based, optimization algorithm in two ways:

- A genetic algorithm generates a population of points in each iteration, whereas a classical algorithm generates a single point at each iteration.
- A genetic algorithm selects the next population by computation using random number generators, whereas a classical algorithm selects the next point by deterministic computation.

Compared to traditional artificial intelligence, a genetic algorithm provides many advantages. It is more robust and is

susceptible to breakdowns due to slight changes in inputs or due to the presence of noise. With respect to other optimization methods like linear programming, heuristic, first or breadth-first, a genetic algorithm can provide better and more significant results while searching large multi-modal state spaces, large state spaces or n-dimensional surfaces.

Genetic algorithms are widely used in many fields such as robotics, automotive design, optimized telecommunications routing, engineering design and computer-aided molecular design.

Criteria for efficient search algorithm for complex problems:

- randomly generate a population of potential solutions
- calculate fitness of each potential solution
- allow best individuals to breed and crossover ($p < 0.6$)
- allow low probability mutations ($p < 0.007$) to maintain diversity
- check for convergence of populations in the gene pool

Genetic Algorithm Components or Operators

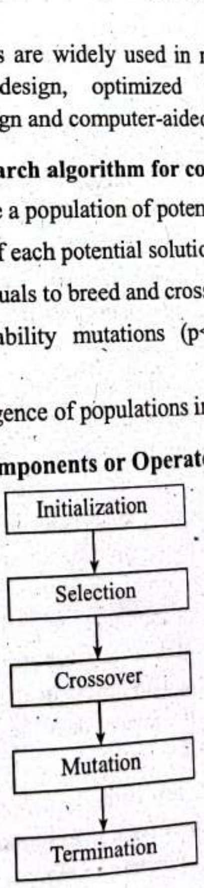


Figure 6.6: Genetic algorithm

Initialization

The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions in the search space. Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection

During each successive generation, a portion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem, one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise.

In some problems, it is hard or even impossible to define the fitness expression; in these cases, a simulation may be used to determine the fitness function value of a phenotype (e.g. computational fluid dynamics is used to determine the air resistance of a vehicle whose shape is encoded as the phenotype), or even interactive genetic algorithms are used.

Crossover and Mutation

The next step is to generate a second generation population of solutions from those selected through a combination of genetic operators: crossover (also called recombination), and mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired" i.e., generate higher quality chromosomes. It is done at random position and offspring produced have 30-70% of its parent's character. We randomly select a crossover point.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally, the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions. These, less fit solutions ensure genetic diversity within the genetic pool of the parents and therefore ensure the genetic diversity of the subsequent generation of children by mutation. In mutation we flip the value of the gene randomly. It is said that only 1% of data is mutated at single iteration. Mutation help to achieve the diversity in the solution.

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached

Scanned with CamScanner

- The highest ranked solution fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

When to Use Genetic Algorithm?

Genetic algorithms are not good for all kinds of problems. They're best for problems where there is a clear way to evaluate fitness. If our search space is not well constrained or our evaluation process is computationally expensive, it may not find solutions in a same amount of time. Genetic algorithm is a decent algorithm in place, but the "knobs" just need to be tweaked.

Example 6.1:

There are 4 items, each item is associated with some weight (w) and value of item (v). There is a knapsack (k) with limited capacity that can hold almost 12 kg.

Items	Weight	Value
A	5 kg	$ 12
B	3 kg	$ 5
C	7 kg	$ 10
D	2 kg	$,7

Problem statement:

The problem is that which item should be kept in the knapsack so that it will maximize knapsack value without breaking knapsack. Use genetic algorithm to solve this problem.

Solution:

Step 1: Chromosome Encoding

Chromosome: $\boxed{A}\boxed{B}\boxed{C}\boxed{D}$

Gene 0 $\rightarrow$ represent absence of item in the knapsack.

Gene 1 $\rightarrow$ represent presence of item in the knapsack.

4 bits are requested to represent chromosome so there can be $2^n = 16$.

Initial population is selected (chromosome) or created randomly.

C_1	0	0	1	0
C_2	0	1	1	0
C_3	1	1	0	1
C_4	1	1	1	1

Generation 1

Step 2: Knapsack capacity = 12,

Applyingfitness function:

For C_1: 0 0 1 0

Value of the knapsack = $10 = 10$

Weight of the knapsack = $7 = 7$ kg

here, $12 > 10$ soC_1 is accepted

For C_2: 0 1 1 0

$v = 5 + 10 = 15$

$w = 3 + 7 = 10$ kg

here, $12 > 10$, So C_2 accepted

For C_3: 1 1 0 1

$v = 12 + 5 + 7 = 24$

$w = 5 + 3 + 2 = 10$ kg

here, $12 > 10$, So C_3 accepted

For C_4: 1 1 1 1

$v = 12 + 5 + 10 + 7 = 34$

$w = 5 + 3 + 7 + 2 = 17$

here, $12 < 17$, So C_4 is not accepted

Scanned with CamScanner

Selection:

It is done by roulette wheel selection. Spin the roulette wheel and wherever the wheel stops, the individual get secreted at that point. The individual that has the highest fitness values get large sheer of the wheel total fitness value = $1\dot{0} + 15 + 24 = 49$

C_3 occupies almost half of the wheel, $\frac{24}{49}$. C_4 has 0 change of winning and C_3 has high chances of getting selected.

Crossover:

The crossover operation takes the selected chromosomes for mating and mixes the genetic material to produce offspring.

One-point crossover

$$\begin{array}{lccc|c} C_3 & 1 & 1 & 0 & 1 \\ C_2 & 0 & 1 & 1 & 0 \\ \hline OS_1 & 1 & 1 & 0 & 0 \\ OS_2 & 0 & 1 & 1 & 1 \end{array} \Big\} \text{offspring}$$

Mutation:

Mutation is done for 1 % of the gene so it is not mandatory to change the at each iteration. It introduces the diversely within the population so that search algorithm doesn't stuck at local maxima.

C_1:	0	0	1	0	Before mutation

$\downarrow$

C_1:	0	0	0	0	after mutation

Replace the chromosome that has lower fitness score with better chromosome or offspring. Now we have five chromosome $OS_1, OS_2, C_1, C_2, C_3$. Let's compare its value and weight.

Selection criteria:

$OS_1 = 1101$, value = $12 + 5 + 7 = 24$, weight = $5 + 3 + 2$

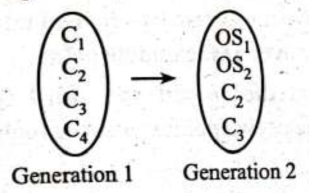
$OS_2 = 0101$, value = $5 + 7 = 12$, weight = $3 + 2$

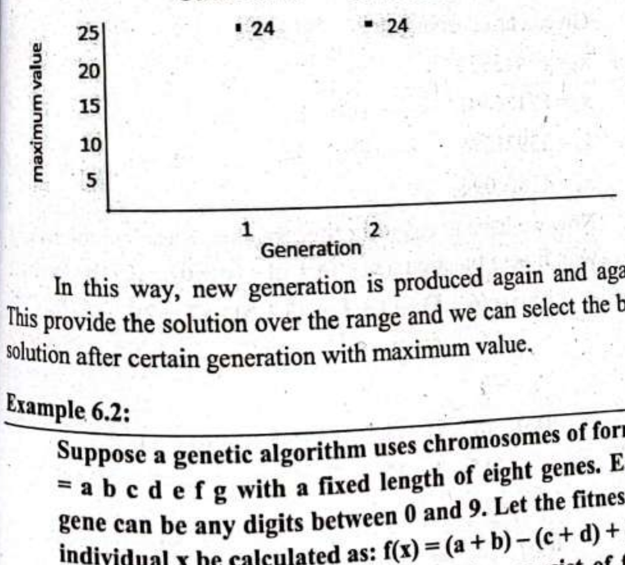
$C_3 = 1101$, value = 24, weight = 10

$C_2 = 0101$, value = $5 + 7 = 12$, weight = $3 + 2 = 5$

$C_1 = 0000$, value = 0, weight = 0

Among five chromosomes (OS_1, OS_2, C_1, C_2 and C_3), C_1 has least value so it is neglected by the rule of survival of the fittest.





In this way, new generation is produced again and again. This provide the solution over the range and we can select the best solution after certain generation with maximum value.

Example 6.2:

Suppose a genetic algorithm uses chromosomes of form x = a b c d e f g with a fixed length of eight genes. Each gene can be any digits between 0 and 9. Let the fitness of individual x be calculated as: f(x) = (a + b) – (c + d) + (e + f) – (g + h), & let the initial population consist of four individuals with the followin chromosomes:

x_1 = **65413532**

x_2 = **87126601**

x_3 = **23921285**

x_4 = **41852094**

a. Evaluate the fitness of each individual, showing all your workings and arragne them in order with the fittest first and the least fit last.

b. Perform the following crossover operations.

i. Cross the fittest two individuals using one-point crossover at the middle point.

ii. Cross the second and third fittest individuals using at two point crossover (points b and f).

Solution:

Given chromosomes are

$x_1 = 65413532$

$x_2 = 87126601$

$x_3 = 23921285$

$x_4 = 41852094$

Now we have to calculate fitness value of each chromosome with fitness function $f(x) = (a + b) - (c + d) + (e+f) - (y+h)$

a. $f(x_1) = (6 + 5) - (4 + 1) + (3 + 5) - (3 + 2)$
$= 11 - 5 + 8 - 5$
$= 9$

$f(x_2) = (8 + 7) - (1 + 2) + (6 + 6) - (0 + 1)$
$= 15 - 3 + 12 - 1$
$= 23$

$f(x_3) = (2 + 3) - (9 + 2) + (1 + 2) - (8 + 5)$
$= 5 - 11 + 3 - 13$
$= -16$

$f(x_4) = (4 + 1) - (8 + 5) + (2 + 0) - (9 + 4)$
$= 5 - 13 + 2 - 13$
$= -19$

Arranging the chromosomes in order with the filtest first and the least fit last is given below:

x_2, x_1, x_3, x_4

b. Crossover operations:

i. One point crossover at the middle point between two filtest chromosme result the offspring as

x_2	8712	6601
x_1	6541	3532
os_1	8712	3532
os_2	6541	6601

ii. Two point crossover (point's b & f) between second & third filtest chromosome result the offspring as.

x_1	65	4135	32
x_3	23	9212	85
os_1	65	9212	32
os_2	23	4135	85

Example 6.3:

You are given a coin. Find the generations using genetic algorithm so that you can get the number of heads above 30. Take 5 chromosomes made up of 10 genes to proceed the solution.

Solution:

Chromosome Encoding

Let us take

$1 \rightarrow$ Head

$0 \rightarrow$ Tail

Initialization

Let us take 5 random chromosome

$C_1 = 1\,0\,0\,0\,1\,0\,1\,1\,0\,0$

$C_2 = 1\,1\,1\,0\,0\,1\,0\,1\,1\,0$

$C_3 = 1\,0\,1\,0\,1\,1\,0\,0\,1\,0$

$C_4 = 0\,1\,1\,1\,1\,0\,1\,0\,1\,1$

Scanned with CamScanner

$C_5 = 0111001111$

Fitness score is the sum of head of individual chromosome's fitness score

$F(C_1) = 4$

$F(C_2) = 6$

$F(C_3) = 5$

$F(C_4) = 7$

$F(C_5) = 7$

So, total fitness score is 29.

Here C_4 and C_5 are selected for cross-over because of higher fitness score.

$$\begin{array}{l} C_4 = 011 \vdots 1101011 \\ \underline{C_5 = 011 \vdots 1001111} \\ OS_1 = 011 \vdots 1001111 \\ OS_2 = 011 \vdots 1101011 \end{array}$$

Now,

$F(C_1) = 4$

$F(C_2) = 6$

$F(C_3) = 5$

$F(C_4) = 7$

$F(C_5) = 7$

$F(OS_1) = 7$

$F(OS_2) = 7$

Selecting the best 5 chromosomes among 7 chromosomes i.e., $C_2, C_4, C_5, OS_1, OS_2$.

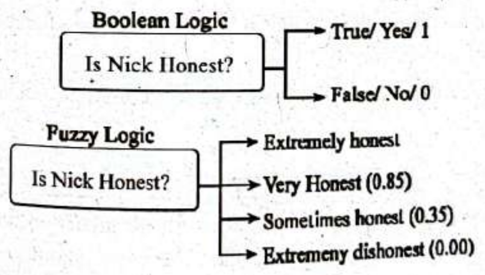
$$\begin{aligned} \text{Total fitness score} &= F(C_2) + F(C_4) + F(C_5) + F(OS_1) + F(OS_2) \\ &= 6 + 7 + 7 + 7 + 7 \\ &= 34 \end{aligned}$$

We reached the required objective in 2nd generation.

6.5 Fuzzy Logic

The term fuzzy mean things which are not very clear or vague. In real life, we may come across a situation where we can't decide whether the statement is true or false. At that time, fuzzy logic offers very valuable flexibility for reasoning. We can also consider the uncertainties of any situation.

Fuzzy logic algorithm helps to solve a problem after considering all available data. Then it takes the best possible decision for the given the input. The FL method imitates the way of decision making in a human which consider all the possibilities between digital values T and F.



6.5.1 History of Fuzzy Logic

Although, the concept of fuzzy logic had been studied since the 1920's. The term fuzzy logic was first used with 1965 by LoftiZadeh a professor of UC Berkeley in California. He observed that conventional computer logic was not capable of manipulating data representing subjective or unclear human ideas.

Fuzzy logic has been applied to various fields, from control theory to AI. It was designed to allow the computer to determine the distinctions among data which is neither true nor something similar to the process of human reasoning, like little dark, some brightness, etc.false.

Scanned with CamScanner

6.5.2 Characteristics of Fuzzy Logic

Here are some important characteristics of fuzzy logic:

- Flexible and easy to implement machine learning technique
- Helps you to mimic the logic of human thought
- Logic may have two values which represent two possible solutions
- Highly suitable method for uncertain or approximate reasoning
- Fuzzy logic views inference as a process of propagating elastic constraints
- Fuzzy logic allows you to build nonlinear functions of arbitrary complexity.
- Fuzzy logic should be built with the complete guidance of experts

6.5.3 When Not to Use Fuzzy Logic

However, fuzzy logic is never a cure for all. Therefore, it is equally important to understand that where we should not use fuzzy logic.

Here, are certain situations when you better not use fuzzy logic:

- If the problem is not convenient to map an input space to an output space
- Fuzzy logic should not be used when you can use common sense

6.5.4 Fuzzy Logic Architecture

Fuzzy logic architecture has three main parts as shown in the diagram:

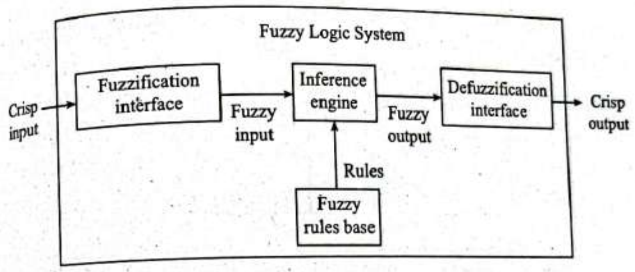


Figure 6.6: Architecture of fuzzy logic

Fuzzification:

Fuzzification step helps to convert inputs. It allows you to convert, crisp numbers into fuzzy sets. Crisp inputs measured by sensors and passed into the control system for further processing. Like room temperature, pressure etc. The main task of this step is to collect words that are needed to be assigned the values like very, little, nearly like words. For example,

Ram car is moving very slow.

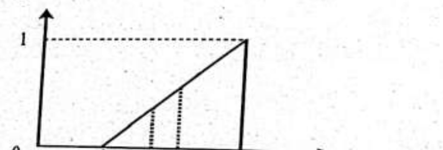
Shyam car looks a bit faster.

In above example, we see that very, looks a bit like terms that need to be addressed and hence in fuzzification, we collect these words.

Rule Base and Inference Engine:

Rule base contains all the rules and the if-then conditions offered by the experts to control the decision-making system. The recent update in fuzzy theory provides various methods for the design and tuning of fuzzy controllers. This updates significantly reduce the number of the fuzzy set of rules.

Inference engine helps you to determines the degree of match between fuzzy input and the rules. Based on the % match, it determines which rules we need to implement according to the given input field. After this, the applied rules are combined to develop the control actions. We define threshold to be taken what formula we need to write so that we can get better degree of membership.



Here 10 km/hr is threshold and below 7.5 km/hr we represent by 0, above 12.5 km/hr we indicate it by 1 and for the values in between we need to define degree of membership.

For the car moving at 10 km/hr we calculate as $\frac{10-7.5}{12.5-7.5} = \frac{10-7.5}{5} = 0.5$ so we can fuzzy value as (10, 0.5).

For the car moving at 9 km/hr we calculate as $\frac{9-7.5}{5} = 0.3$ and for this we write fuzzy value as (10, 0.3).

We see that in this step we define the threshold and gap should be taken so that any value can be represented in fuzzy value.

Defuzzification:

At last the *defuzzification* process is performed to convert the fuzzy sets into a crisp value. There are many types of techniques available, so you need to select it which is best suited when it is used with an expert system. It is the process of producing a quantifiable result in the fuzzy logic from fuzzy set and corresponding membership degree. Defuzzification is the interpretation of the membership degree for collected vague values in first step of fuzzification.

Example:

Very slow is indicated by (10, 0.1).

A bit faster is indicated by (10, 0.9).

6.5.5 Fuzzy Operations

Fuzzy set is the set that has the member with the threshold value and degree of membership. Let us consider fuzzy sets for two cars is given with threshold value of speed and degree of membership as

A = {(10, 0.2), (20, 0.3), (30, 0.4), (40, 0.5)}

B = {(10, 0.3), (20, 0.4), (30, 0.1), (40, 0.2)}

i. **Union:** Here, select the degree of membership with maximum values

$A \cup B$ = {(10, 0.3), (20, 0.4), (30, 0.4), (40, 0.5)}

ii. **Intersection:** In intersection, select the degree of membership with minimum values

$A \cap B$ = {(10, 0.2), (20, 0.3), (30, 0.1), (40, 0.2)}

iii. **Complement:** Simply subtract the degree of membership from 1.

$\overline{A}$ = {(10, 0.8), (20, 0.7), (30, 0.5), (40, 0.5)}

iv. **X-OR or bold union:**

$A \oplus B = \min\{1, M_A + M_B\}$

$A \oplus B$ = {(10, 0.5), (20, 0.7), (30, 0.5), (40, 0.7)}

v. **Bold intersection:**

$A \odot B = \max\{0, M_A + M_B - 1\}$

$A \odot B$ = {(10, 0), (20, 0), (30, 0), (40,0)}

vi. **A and B will be equal if all the members in both set are equal.**

6.5.6 Advantages and Disadvantages of Fuzzy Logic System

Advantages:

- The structure of fuzzy logic system is easy and understandable.
- Fuzzy logic is widely used for commercial and practical purposes.

Scanned with CamScanner

- Mostly robust as no precise inputs required
- It can be programmed to in the situation when feedback sensor stops working.
- It can easily be modified to improve or alter system performance.
- Inexpensive sensors can be used which helps you to keep the overall system cost and complexity low.
- It provides a most effective solution to complex issues.

Disadvantages:

- Fuzzy logic is not always accurate, so the results are perceived based on assumption, so it may not be widely accepted.
- Fuzzy systems don't have the capability of machine learning aswellas neural network type pattern recognition.
- Validation and verification of a fuzzy knowledge-based system needs extensive testing with hardware.
- Setting exact, fuzzy rules and, membership functions is a difficult task.
- Some fuzzy time logic is confused with probability theory and the terms.

6.5.7 Fuzzy Logic versus Probability

Fuzzy Logic	Probability
Ram's degree of membership within the set of old people is 0.90	There is a 90% chance that Ram is old.
Fuzzy logic takes truth degrees as a mathematical basis on the model of the vagueness phenomenon.	Probability is a mathematical model of ignorance.

6.5.8 Crisp versus Fuzzy

Crisp	Fuzzy
It has strict boundary T or F	Fuzzy boundary with a degree of membership
Some crisp time set can be fuzzy	It can't be crisp
True/False {0, 1}	Membership values on [0, 1]

6.6 ID3 (Iterative Dichotomizer 3)

ID_3 is the first of a series of algorithms created by Ross Quinlan to generate decision trees.

Characteristics:

- ID_3 does not guarantee an optimal solution; it can get stuck in local optimums
- It uses a greedy approach by selecting the best attribute to split the dataset on each iteration (one improvement that can be made on the algorithm can be to use backtracking during the search for the optimal decision tree)
- ID_3 can overfit to the training data (to avoid overfitting, smaller decision trees should be preferred over larger ones)
- This algorithm usually produces small trees, but it does not always produce the smallest possible tree
- ID_3 is harder to use on continuous data (if the values of any given attribute is continuous, then there are many more places to split the data on this attribute, and searching for the best value to split by can be time consuming).
- The ID_3 algorithm is used by training a dataset S to produce a decision tree which is stored in memory.
- At runtime, the decision tree is used to classify new unseen test cases by working down the tree nodes using the values of a given test case to arrive at a terminal node that tells you what class this test case belongs to.

Scanned with CamScanner

Metrics:

Entropy H(s) – measures the amount of uncertainty in the dataset

Information gain IG(A) – measures how much uncertainty in S was reduced, after splitting the (data) set S on an attribute

Algorithm of iterative dichotomizer 3:

Step I: Calculate the information gain before splitting.

$$IG_{before} = -\frac{P}{P+N}\log_2\frac{P}{P+N} - \frac{N}{P+N}\log_2\frac{N}{P+N}$$

Step II:Calculate entropy of each class

$$E_{Class} = \Sigma\left(\frac{P_i+N_i}{P+N}\right) IG_{subclass}$$

$N \rightarrow$ Negative, $P \rightarrow$ Positive

Where,

$$IG_{subclass} = -\frac{P_i}{P_i+N_i}\log_2\frac{P_i}{P_i+N_i} - \frac{N_i}{P_i+N_i}\log_2\frac{N_i}{P_i+N_i}$$

Step III

Calculate gain for each class

$\text{Gain} = IG_{before} - E_{class}$

Step IV

Compare all gain value & select the root node having the highest gain.

Step V

Again, start from step I to V for root node. Continue this unit it covers for all value.

Example 6.4:

A survey of 10 companies has been done and they are analyzed on the basis of age, competition, type and profit. Make decision tree diagram for the following dataset to evaluate the profit for new company.

Age	Competition	Type Software	Profit
Old	Yes	Software	Down
Old	No	Software	Down
Old	No	Hardware	Down
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up
New	Yes	Software	Up
New	No	Hardware	Up
New	No	Software	Up

Solution:

Step 1: Calculate information gain of the data set provided.

Information gain before (IG_{before})

$$= -\frac{P}{P+N}\log_2\frac{P}{P+N} - \frac{N}{P+N}\log_2\frac{N}{P+N}$$

$$= -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10} = 1$$

Step 2: Calculate the entropy of each class:

$$E_{age} = \Sigma\left(\frac{P_i+N_i}{P+N} IG_{subclass}\right)$$

$$IG_{old} = -\frac{P_i}{P_i+N_i}\log_2\frac{P_i}{P_i+N_i} - \frac{N_i}{P_i+N_i}\log_2\frac{N_i}{P_i+N_i}$$

For OLD subclass, $P_i = 0, N_i = 3$

Scanned with CamScanner

$IG_{old} = 0$

Similarly, for subclass MID IG_{mid}, $P_i = 2$, $N_i = 2$

$IG_{mid} = 1$

For subclass NEW IG_{New}, $P_i = 3$, $N_i = 0$

$IG_{New} = 0$

$$E_{age} = \frac{0+3}{10} \times 0 + \frac{2+2}{10} \times 1 + \frac{3+0}{10} \times 0 = +\frac{4}{10} = +0.4$$

Now for class competition,

For subclass YES, $P_i = 1$, $N_i = 3$

$$IG_{yes} = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.81$$

For subclass NO, $P_i = 4$, $N_i = 2$

$$IG_{no} = -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6} = 0.92$$

$$\text{Now, } E_{comp} = \frac{1+3}{10} \times 0.81 + \frac{4+2}{10} \times 0.92 = 0.87$$

Now for class type,

For subclass SOFTWARE, $P_i = 3$, $N_i = 3$

$IG_{software} = 1$

For subclass HARDWARE, $P_i = 2$, $N_i = 2$

$IG_{hardware} = 1$

$$E_{type} = \frac{3+3}{10} \times 1 + \frac{2+2}{10} \times 1 = 1$$

Step 3: Calculate gain for each class.

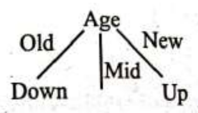
$Gain_{age} = 1 - 0.4 = 0.6$

$Gain_{comp} = 1 - 0.87 = 0.13$

$Gain_{type} = 1 - 1 = 0$

Step 4: Compare gains and select the highest one as root node.

Here, in age class, OLD results negative result and NEW give positive result so the diagram will be as follows:



Now, we see that there is still some randomness in Mid class so we have to repeat for those data from step 2 to 4.

Step 5:

Now, we need to look for only mid class,

Age	Competition	Type	Profit
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up

$P = 2$, $N = 2$

$IG_{before} = 1$

For competition Yes,

$P_i = 0$, $N_i = 2$

$IG_{yes} = 0$

For competition No,

$P_i = 2$, $N_i = 0$

$IG_{no} = 0$

$E_{comp} = 0$,

For type Software,

$P_i = 1$, $N_i = 1$

$IG_{software} = 1$

For type Hardware,

$P_i = 1$, $N_i = 1$

$IG_{hardware} = 1$

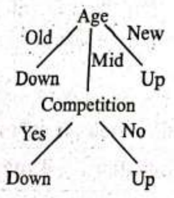
$E_{type} = 1$

Scanned with CamScanner

$\text{Gain}_{comp} = 1 - 0 = 1$

$\text{Gain}_{type} = 1 - 1 = 0$

Here, we find gain of competition to be high so,



EXERCISE

1. Distinguish between
   a) Supervised and unsupervised learning
   b) Training and testing
   c) Rule-based learning vs Fuzzy learning
2. Explain Fuzzy logic architecture in detail.
3. Write short notes on:
   a) Fuzzy operations
   b) Reinforcement learning
   c) Cross-over in genetic algorithm
   d) Mutation in genetic algorithm
4. Discuss the genetic operators in genetic algorithm.
5. Explain ID3 algorithm with example.

Chapter 7

APPLICATION OF AI

7.1 Neural Network

Artificial neural networks (ANN) are computing systems vaguely inspired by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the results to identify cats in other images. They do this without any prior knowledge about cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the learning material that they process.

Computers and the Brain: A Contrast

- **Arithmetic:** 1 brain $= \frac{1}{10}$ pocket calculator
- **Vision:** 1 brain = 1000 supercomputers
- **Memory of arbitrary details:** Computer wins
- **Memory of real-world facts:** Brain wins
- A computer must be programmed explicitly
- The brain can learn by experiencing the world

Scanned with CamScanner

Parameters	Computer	Brain
Complexity	Ordered structure serial processor	10^{10} neurons, 10^4 connections
Processor speed (operation per second)	10000000	100
Computational power	One operation at a time; 1 or 2 inputs at a time	Millions of operations at a time; 10^4 inputs at a time

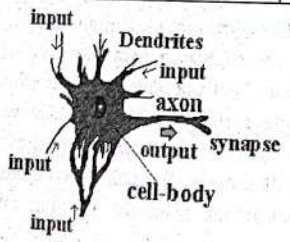


Figure 7.1: Biological neural network

Component Analogy Between Biological Neural Network and Artificial Neural Network

- **Neuron** –a biological information processor
- **Dendrites** – the receivers
- **Soma** – neuron cell body (sums input signals)
- **Axon** – the transmitter
- **Synapse** – point of transmission; neuron activates after a certain threshold is met

Learning occurs via electro-chemical changes in effectiveness of synaptic junction.

7.1.1 History of Artificial Neural Network

Warren McCulloch and Walter Pitts (1943) created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model paved the way for neural network research to split into two approaches. One approach focused on biological processes in the brain while the other focused on the application of neural networks to artificial intelligence. This work led to work on nerve networks and their link to finite automata.

In the late 1940s, D. O. Hebb created a learning hypothesis based on the mechanism of neural plasticity that became known as Hebbian learning. Hebbian learning is unsupervised learning. This evolved into models for long term potentiation. Researchers started applying these ideas to computational models in 1948 with Turing's B-type machines. Farley and Clark (1954) first used computational machines, then called "calculators", to simulate a Hebbian network.

Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition. With mathematical notation, Rosenblatt described circuitry not in the basic perceptron, such as the exclusive-or circuit that could not be processed by neural networks at the time.

Neural network research stagnated after machine learning research by Minsky and Papert (1969), who discovered two key issues with the computational machines that processed neural networks. The first was that basic perceptron was incapable of processing the exclusive-or(X-OR) circuit. The second was that computers didn't have enough processing power to effectively handle the work required by large neural networks. Neural network research slowed until computers achieved far greater processing power. Until in the late 1980s research expanded to low-level (sub-symbolic) machine learning, characterized by knowledge embodied in the parameters of a cognitive model.

A key trigger for renewed interest in neural networks and learning was Werbos's (1975) backpropagation algorithm that effectively solved the exclusive-or problem by making the training of multi-layer networks feasible and efficient. Backpropagation distributed the error term back up through the layers, by modifying the weights at each node.

Support vector machines and other, much simpler methods such as linear classifiers gradually overtook neural networks in machine learning popularity. However, using neural networks transformed some domains, such as the prediction of protein structures.

Hinton (2006) proposed learning a high-level representation using successive layers of binary or real-valued latent variables with a restricted Boltzmann machine to model each layer. Once sufficiently many layers have been learned, the deep architecture may be used as a generative model by reproducing the data when sampling down the model (an "ancestral pass") from the top-level feature activations. In 2012, Ng and Dean created a network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images taken from YouTube videos.

Earlier challenges in training deep neural networks were successfully addressed with methods such as unsupervised pre-training, while available computing power increased through the use of GPUs and distributed computing. Neural networks were deployed on a large scale, particularly in image and visual recognition problems. This became known as "deep learning".

7.1.2 Basic Neural Network Model

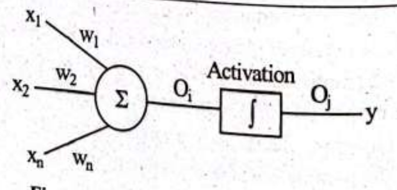


Figure 7.2: Basic neural network model

This is the information processing unit of artificialneuralnetwork. According to this model, a neuron consists of the following:

i. **Inputs:** Inputs are considered as features and each feature can support values. These values are taken from the environment or training dataset. For example, no. of rooms, area of house, location of house, etc. can be inputs for predicting the price of house. Each input can have different values like no. of room can be 4, 5, 6, etc.

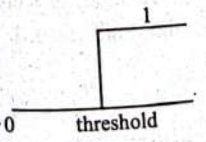
ii. **Weights:** They are the values which signify how important the input or features or dimensions is to make prediction. For example, location play more role to make decisions than no. of room and area of house, so we assign more weight to location.

iii. **Bias:** The bias 'b' has the effect of applying a transformation to the weighted sum and is an external parameter of the neuron.

iv. **Adder:** This component helps to collect or add all the values.

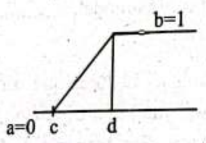
v. **Activation function:** Activation function limits the amplitude of the neuron which helps to analyze the result in a convenient way.

a. **Step function**



$$y = \begin{cases} 1 \text{ if } O_i \geq \text{threshold} \\ 0 \text{ if } O_i < \text{threshold} \end{cases}$$

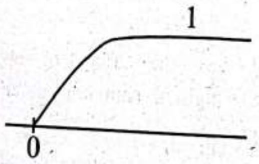
b. Ramp function



$$y = \begin{cases} 0 \text{ if } O_i < c \\ 1 \text{ if } O_i > d \\ a + \dfrac{(O_i - c)(b - a)}{d - c} \text{ otherwise} \end{cases}$$

Values of a and b are taken as 0, 1 repectively to limit the value of output(y) between 0 and 1.

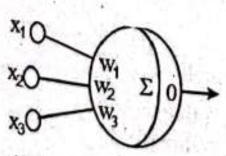
c. Sigmoid function



$$y = \frac{1}{1 + e^{-O_i}}$$

7.1.3 McCulloch/Pitts Neuron

It is one of the first neuron models that had been implemented. Its output is 1 (fired) or 0 (not fired). Each input is weighted with weights in the range −1 to +1. It has a threshold value, T.



The neuron fires if the following inequality is true:

$x_1 \times w_1 + x_2 \times w_2 + x_3 \times w_3 > T$

In neural network, we have to assign the weight of the neuron such that it satisfies the value for all of the dataset.

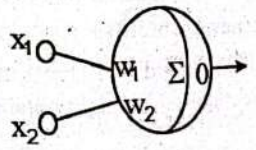
Example 7.1:

Construct an MCP (McCulloch/Pitts) neuron which will implement the function of OR gate below:

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Find the threshold T and the weights w_1 and w_2, where Y is 1 if $x_1 \times w_1 + x_2 \times w_2 > T$

Solution:



Each line of the function table places conditions on the unknown values:

If $x_1=0$, $x_2=0$, $w_1=w_2=$any value, then $x_1 \times w_1 + x_2 \times w_2 = 0 < T$

If $x_1=0$, $x_2=1$, $w_1=w_2=$any value, then $x_1 \times w_1 + x_2 \times w_2 = w_2 > T$

If $x_1=1$, $x_2=0$, $w_1=w_2=$any value then $x_1 \times w_1 + x_2 \times w_2 = w_1 > T$

If $x_1=1$, $x_2=1$, $w_1=w_2=$any value then $x_1 \times w_1 + x_2 \times w_2 = w_1+w_2 > T$

By hit and trial, we can get the solution as

w_1 and $w_2 = 0.7, T = 0.5$

There may be other solutions that satisfies above four equations.

Scanned with CamScanner

7.1.4 Hebbian Learning

The oldest and most famous of all learning rules is Hebb's postulate of learning. The key concept behind this is "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased".

Hebb's Algorithm

Step 0: Initialize all weights to 0

Step1: Given a training input(s) with its target output(t)set the activations of the input units:$x_i = s_i$

Step 2: Set the activation of the output unit to the target value: y = t from training set.

Step 3: Adjust the weights: $w_i(\text{new}) = w_i(\text{old}) + x_i \times y$

Step 4: Adjust the bias: $b(\text{new}) = b(\text{old}) + y$

Step 5: Continue until all the conditions are satisfied. Use the testing criteria to validate the weight obtained.

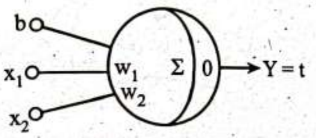
Example 7.2:

Construct a Hebb Net which performs like an AND function, that is, only when both features are "active" will the data be in the target class.

TRAINING SET (with the bias input always at 1):

x_1	x_2	Y
1	1	1
1	−1	−1
−1	1	−1
−1	−1	−1

Solution:



Initialize all the weights to 0 i.e.,$w_1 = w_2 = 0$.

Present the first input (1, 1) with a target of 1

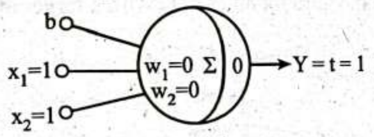


Fig: Before 1st training run

Update the weights:

$w_1(\text{new}) = w_1(\text{old}) + x_1 \times t = 0 + 1 = 1$

$w_2(\text{new}) = w_2(\text{old}) + x_2 \times t = 0 + 1 = 1$

$b(\text{new}) = b(\text{old}) + t = 0 + 1 = 1$

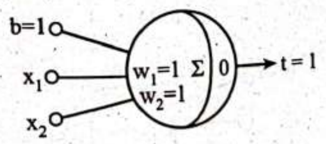


Fig.: After 1st training run

Training – second input

The second input: $(1, -1)$ with a target of -1

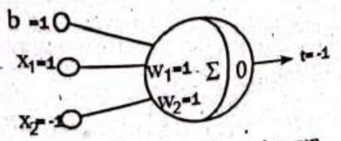


Fig.: Before 2nd training run

Update the weights:

$w_1(\text{new}) = w_1(\text{old}) + x_1 \times t = 1 + 1(-1) = 0$

$w_2(\text{new}) = w_2(\text{old}) + x_2 \times t = 1 + (-1)(-1) = 2$

$b(\text{new}) = b(\text{old}) + t = 1 + (-1) = 0$

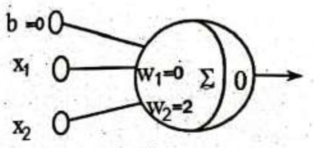


Fig.: After 2nd training run

Training – third input

Present the third input: (−1, 1) with a target of −1.

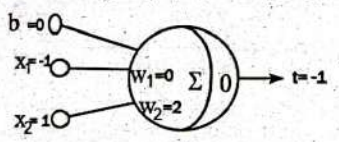


Fig.: Before 3rd training run

Update the weights:

$w_1(\text{new}) = w_1(\text{old}) + x_1 \times t = 0 + (-1)(-1) = 1$

$w_2(\text{new}) = w_2(\text{old}) + x_2 \times t = 2 + 1(-1) = 1$

$b(\text{new}) = b(\text{old}) + t = 0 + (-1) = -1$

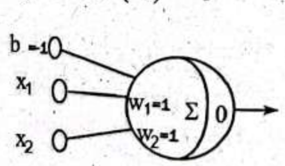


Fig.: After 3rd training run

Training – Fourth Input

Present the fourth input:(−1, −1) with a output of −1.

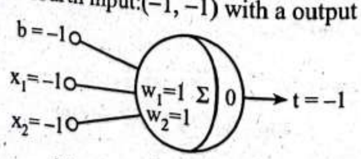


Fig.: Before 4th training run

Update the weights:

$w_1(\text{new}) = w_1(\text{old}) + x_1 \times t = 1 + (-1)(-1) = 2$

$w_2(\text{new}) = w_2(\text{old}) + x_2 \times t = 1 + (-1)(-1) = 2$

$b(\text{new}) = b(\text{old}) + t = -1 + (-1) = -2$

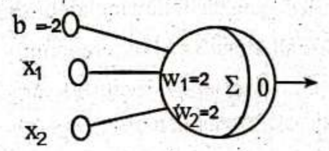
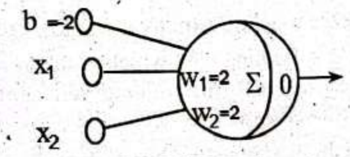


Fig.: After 4th training run

Validate if the given neural network works or not:



x_1	x_2	Y
1	1	1
1	−1	−1
−1	1	−1
−1	−1	−1

1st case: $1\times2 + 1\times2 + 1\times(-2) = 2>0$

2nd case: $1\times2 + (-1)\times2 + 1\times(-2) = -2<0$

3rd case: $(-1)\times2 + 1\times2 + 1\times(-2) = -2<0$

4th case: $(-1)\times2 + (-1)\times2 + 1\times(-2) = -6<0$

Hence this satisfies all the truth table. So we stop here.

7.1.5 Perceptron

The perceptron was suggested by Rosenblatt in 1958. It uses an iterative learning procedure which can be proven to converge to the correct weights for linearly separable data. It has a bias and a threshold function.

Scanned with CamScanner

Perceptron Learning Rule

Weights are changed only when an error occurs. The weights are updated using the following algorithm:

Step 0: Initialize all weights and bias randomly.

Step 1: Given a training input(s) with its target output(t) set the activations of the input units: $x_i = s_i$, t is either +1 or −1, x_i is input.

Step 2: Set the activation of the output unit to the target value: y = t from training set.

Step 3: Adjust the weights: $w_i(\text{new}) = w_i(\text{old}) + \alpha \times t \times x_i$

where α is learning rate, which is taken normally 0.1 for gate realization. If the training set increase, we take α using relation $0.01 < n\alpha < 1$.

Step 4: Adjust the bias: $b(\text{new}) = b(\text{old}) + \alpha \times y$

Step 5: Continue until all the conditions are satisfied. If an error does not occur, the weights aren't changed. We test the error using the formula,

$x_1 \times w_1 + x_2 \times w_2 > T$

Perceptron can work for linearly separable functions like AND, OR, NOT, NAND, NOR, etc.

For example, AND gate:

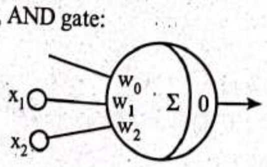


Fig.: AND gate

Table: Truth table of AND gate

x_1	x_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

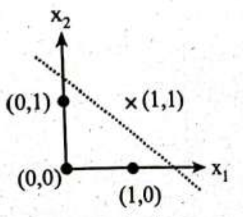


Fig.: Graph for AND gate

We can separate the output with 1 and 0 with a line as shown in figure.

Limitations of the Perceptron

The perceptron can only learn to distinguish between classifications if the classes are linearly separable. If the problem is not linearly separable then the behaviour of the algorithm is not guaranteed. If the problem is linearly separable, there may be a number of possible solutions. The algorithm as stated gives no indication of the quality of the solution found.

X-OR Problem

A perceptron network can't implement an XOR function.

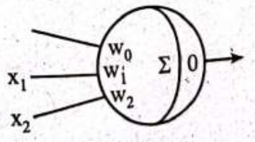


Fig.: XOR gate

Table: Truth Table XOR

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Scanned with CamScanner

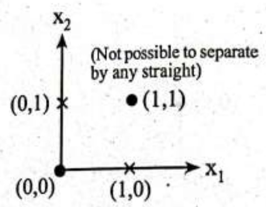


Fig.: Graph for X-OR gate

There is no assignment of values to w_0, w_1, and w_2 that satisfies above inequalities. X-OR cannot be represented.

7.1.6 Adaline Network

It is slightly variation on the perceptron network. Inputs are +1 or −1 and outputs are +1 or −1 and uses a bias input.

Adaline network is trained using the delta rule which is also known as the least mean squares (LMS) or Widrow-Hoff rule. The activation function during training is identity function. After training, the activation function is a threshold function.

Adaline Algorithm

Step 0: Initialize the weights to small random values and select a learning rate, α.

Step 1: For each input vector s, with target output t, set the inputs to node.

Step 2: Compute the neuron inputs:

$y\_in = b + \Sigma (x_i \times w_i)$

Step 3: Use the delta rule to update the bias and weights

$b(new) = b(old) + \alpha(t - y\_in)$

$w_i(new) = w_i(old) + \alpha(t - y\_in)x_i$

Step 4: Stop if the largest weight change across all the training samples is less than a specified tolerance, otherwise cycle through the training set again.

The performance of an ADALINE neuron depends heavily on the choice of the learning rate.

- If it is too large, the system will not converge
- If it is too small, the convergence will take too long

Typically, α is selected as trial and error with following assumption.

- Typical range: $0.01 < \alpha < 10.0$
- Often start at 0.1
- Sometimes it is suggested that$0 < n\alpha < 1.0$ where n is the number of inputs.

Example 7.3:

Construct an AND function for an ADALINE neuron. Consider α = 0.1.

x_1	x_2	Y
1	1	1
1	−1	−1
−1	1	−1
−1	−1	−1

Solution:

Initial conditions:

Set the weights to small random values.

Let $b = 0.1$, $w_1 = 0.2, w_2 = 0.3$

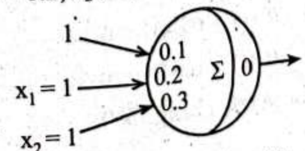


Fig.: Before 1st training run

First training run:

Apply the input (1,1) with output 1

The net input is:

$y\_in = 0.1 + 0.2 \times 1 + 0.3 \times 1 = 0.6$

$y\_in = b + \Sigma (x_i \times w_i)$

Delta rule:

$b(new) = b(old) + \alpha(t - y\_in)$

$w_i(new) = w_i(old) + \alpha(t - y\_in)x_i$

The new weights are:

$b = 0.1 + 0.1(1 - 0.6) = 0.14$

$w_1 = 0.2 + 0.1(1 - 0.6)1 = 0.24$

$w_2 = 0.3 + 0.1(1 - 0.6)1 = 0.34$

The largest weight change is 0.04.

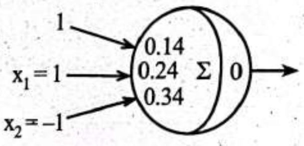


Fig.: After 1st training run

Second training run:

Apply the second training set (1, –1) with output –1.

The net input is:

$y\_in = 0.14 + 0.24 \times 1 + 0.34 \times (-1) = 0.04$

The new weights are:

$b = 0.14 - 0.1(1 + 0.04) = 0.04$

$w_1 = 0.24 - 0.1(1 + 0.04)1 = 0.14$

$w_2 = 0.34 + 0.1(1 + 0.04)1 = 0.44$

The largest weight change is 0.1.

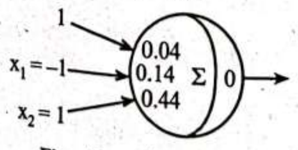


Fig.: After 2nd training run

Third training run:

Apply the third training set (–1, 1) with output –1.

The net input is:

$y\_in = 0.04 - 0.14 \times 1 + 0.44 \times 1 = 0.34$

The new weights are:

$b = 0.04 - 0.1(1 + 0.34) = -0.09$

$w_1 = 0.14 + 0.1(1 + 0.34)1 = 0.27$

$w_2 = 0.44 - 0.1(1 + 0.34)1 = 0.31$

The largest weight change is 0.13

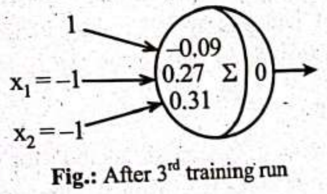


Fig.: After 3rd training run

Fourth training run:

Apply the fourth training set (–1, –1) with output –1.

The net input is:

$y\_in = -0.09 - 0.27 \times 1 - 0.31 \times 1 = -0.67$

The new weights are:

$b = -0.09 - 0.1(1 + 0.67) = -0.27$

$w_1 = 0.27 + 0.1(1 + 0.67)1 = 0.43$

$w_2 = 0.31 + 0.1(1 + 0.67)1 = 0.47$

The largest weight change is 0.16.

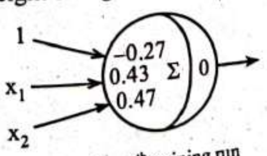


Fig.: After 4th training run

In this way, we update our network till all the data in the truth table are satisfied. This may take hundreds of iterations to reach the desirable weight.

7.1.7 Multilayer Artificial Neural Network

After Rosenblatt perceptron was developed in the 1950s, there was a lack of interest in neural networks until 1986, when Dr. Hinton and his colleagues developed the backpropagation algorithm to train a multilayer neural network.

Today it is a hot topic with many leading firms like Google, Facebook, and Microsoft which invest heavily in applications using deep neural networks. A fully connected multilayer neural network is called a *multilayer perceptron (MLP)*.

Backpropagation

Backpropagation is a common method for training a neural network. Here, how it works is explained with a concrete example. Consider an example with two inputs, two hidden neurons, and two output neurons. Additionally, the hidden and output neurons will include a bias.

Here's the basic structure to implement X-OR gate:

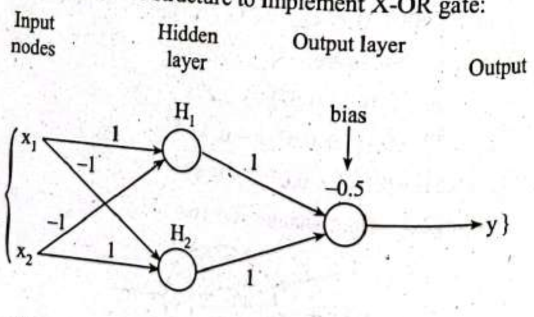


Figure 7.3: Feed forward neural network for XOR gate

Table: Working of an feed foward neural network for XOR gate

Input		Output of hidden nodes		Output node	x_1 XOR x_2
x_1	x_2	H_1	H_2		
0	0	0	0	$-0.5 \to 0$	0
0	1	$-1 \to 0$	1	$0.5 \to 1$	1
1	0	1	$-1 \to 0$	$0.5 \to 1$	1
1	1	0	0	$-0.5 \to 0$	0

Backpropagation Algorithm

Step 1: Initialize all weightsand bias randomly.

Step 2: Consider training data set and corresponding input as $I_i = O_i$ and $T = Y_i$.

Step 3: Do forward pass to calculate hidden layer values

$$I_j = \Sigma(w_{ij} \times O_i) + b_j$$

If we take sigmoid function as activation function then

$$O_j = \frac{1}{1 + e^{-I_j}}$$

Step 4: Calculate for output layer,

$$I_k = \Sigma(w_{jk} \times O_j) + b_k$$

$$O_k = \frac{1}{1 + e^{-I_k}}$$

Step 5: Calculate errors:

error in output layer, $E_k = O_k(1 - O_k)(T - O_k)$

error in hidden layer; $E_j = O_j(1 - O_j)\,\Sigma(E_k \times w_{jk})$

Step 6: Hidden layer's weight and bias are updated using following formula:

$$\Delta w_{ij} = \alpha \times E_j \times O_i$$

$$w_{ij} = w_{ij} + \Delta w_{jk}$$

$$\Delta b_j = \alpha \times E_j$$

Scanned with CamScanner

$b_j = b_j + \Delta b_j$

Output layer weight and bias are updated using following formula:

$\Delta w_{jk} = \alpha \times E_k \times O_j$

$W_{jk(New)} = W_{jk(old)} + \Delta w_{jk}$

$\Delta b_k = \alpha \times E_k$

$b_k = b_k + \Delta b_k$

Step 7: Continue from steps (ii) to (vi) until all training sets values are satisfied.

Example 7.4:

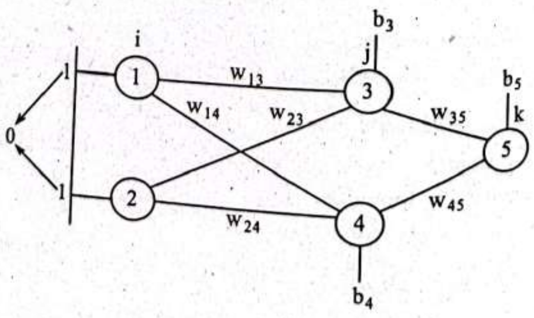
Find the weight for X-OR gate for just one iteration taking first training data from the table given below.

x_1	x_2	Y
1	1	0
0	1	1
1	0	1
0	0	0

Take sigmoid function as activation function

Solution:

Taking training set as $x_1 = x_2 = 1$ and target T as 0.



Step 1: Initializing the weights and bias randomly

$b_3 = 0.1$	$b_4 = 0.35$	$w_{35} = 0.35$
$w_{13} = 0.15$	$w_{14} = 0.25$	$w_{45} = 0.4$
$w_{23} = 0.2$	$w_{24} = 0.3$	$b_5 = 0.45$

Step 2: $I_1 = 1$, $I_2 = 1$, So $O_1 = O_2 = 1$ and $T = 0$ (from training set taking ist dataset from the table)

Step 3: Input at node 3 is $I_3 = w_{13} \times I_1 + w_{23} \times I_2 + b_3$

$= 0.15 \times 1 + 0.2 \times 1 + 0.1 = 0.45$

Output from node 3 is $O_3 = \frac{1}{1 + e^{-0.45}} = 0.6106$

Input at node 4 is $I_4 = w_{14} \times I_1 + w_{24} \times I_2 + b_4 = 0.9$

Output from node 4 is $O_4 = 0.7109$

Step 4: Input from node is $I_5 = w_{35} \times O_3 + w_{45} \times O_4 + b_5 = 0.9481$

Output from node is $O_5 = 0.7207$

Step 5: Error at node 5 is $E_5 = O_5 (1 - O_5)(T - O_5) = 0.7207(1 - 0.7207)(0 - 0.7207) = -0.1450$

Error at node 3 is $E_3 = O_3 (1 - O_3) E_5 \times w_{35} = -0.0121$

Error at node 4 is $E_4 = O_4 (1 - O_4) E_5 \times w_{45} = -0.0119$

Step 6: For updating weight and bias,

$\Delta w_{13} = \alpha \times E_3 \times O_1 = 0.1 \times -0.0121 \times 1 = -0.00121$

$w_{13new} = -0.00121 + 0.15 = 0.1488$

$\Delta b_3 = 0.1 \times -0.0121 = -0.00121$

$b_{3new} = 0.1 + (-0.00121) = 0.09879$

$\Delta w_{14} = 0.1 \times E_4 \times O_1 = 0.1 \times -0.0119 \times 1 = -0.00119$

$w_{14new} = -0.00119 + 0.25 = 0.24881$

$\Delta b_4 = 0.1 \times -0.0119 = -0.00119$

$b_{4new} = 0.1 + (-0.00119) = 0.09881$

$\Delta w_{24} = 0.1 \times E_4 \times O_2 = 0.1 \times (-0.0119) \times 1 = -0.00119$

$w_{24new} = -0.00119 + 0.3 = 0.29881$

Scanned with CamScanner

$\Delta b_4 = 0.1 \times -0.0119 = -0.00119$

$b_{4new} = 0.1 + (-0.00119) = 0.09881$

$\Delta w_{23} = 0.1 \times E_3 \times O_2 = -0.00121$

$w_{23new} = 0.19879$

$\Delta b_3 = -0.00121$

$b_{3new} = 0.09879$

$\Delta w_{45} = -0.0103$

$w_{45new} = 0.3897$

$\Delta b_5 = -0.0145$

$b_{5new} = 0.4355$

$\Delta w_{35} = -0.00885$

$w_{35new} = 0.34115$

$\Delta b_5 = -0.0145$

$b_{5new} = 0.4355$

We repeat all the steps for other training dataset and later we continue this for hundreds of epochs and then we get the stable solution.

7.1.8 Recurrent Neural Networks

Recurrent neural networks (RNNs) are popular models that have shown great promise in many NLP tasks. The idea behind RNNs is to make use of sequential information. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If we want to predict the next word in a sentence we better know which words came before it. RNNs are called *recurrent* because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to

looking back only a few steps (more on this later). Here is what a typical RNN looks like:

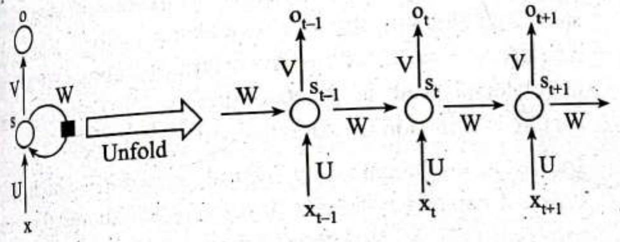


Figure 7.4: Recurrent neural network

A recurrent neural network and the unfolding in time of the computation involved in its forward computation.

The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in RNN are as follows:

- X_t is the input at time step t. For example, X_1 could be a one-hot vector corresponding to the second word of a sentence.
- S_t is the hidden state at time step t. It's the "memory" of the network. S_t is calculated based on the previous hidden state and the input at the current step.

  $S_t = F(U \times X_t + W \times S_{t-1})$

  The function F usually is a non-linearity such as tanh or ReLU. S_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- O_t is the output at step t. For example, to predict the next word in a sentence it would be a vector of probabilities across our vocabulary.

  $O_t = \text{softmax}(V \times S_t)$

Scanned with CamScanner

There are a few things to note here:

- The hidden state $S_t$is the memory of the network. S_t captures information about what happened in all the previous time steps. The output at step O_t'is calculated solely based on the memory at time t. As briefly mentioned above, it's a bit more complicated in practice because S_t typically can't capture information from too many time steps ago.
- Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (U, V, W above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn.
- The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence.

Applications of RNN:

i. **Language modeling and generating text**

Given a sequence of words we want to predict the probability of each word given the previous words. Language models allow us to measure how likely a sentence is, which is an important input for machine translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a generative model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is, we can generate all kind of stuff. In language modeling, our input is typically a sequence of

words (encoded as one-hot vectors for example), and our output is the sequence of predicted words.

i. **Machine translation**

Machine translation is similar to language modeling in that our input is a sequence of words in our source language (e.g., German). We want to output a sequence of words in our target language (e.g., English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.

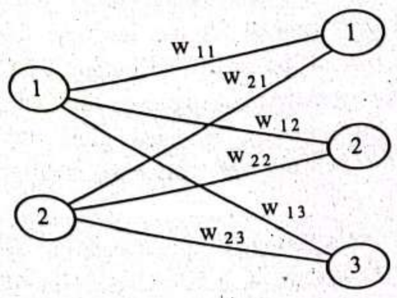
ii. **Speech recognition**

Given an input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.

7.1.9 Kohonen Neural Network

This network. is also known as *self-organizing map*. Kohonenself-organizing feature map (SOM) refers to a neural network, which is trained using competitive learning. Basic competitive learning implies that the competition process takes place before the cycle of learning. The competition process suggests that some criteria select a winning processing element. After the winning processing element is selected, its weight vector is adjusted according to the used learning law (Hecht Nielsen, 1990).

Let us take an example of neural network as shown in the figure below:



Scanned with CamScanner

Let us take the weight as

$w_{11} = 0.2$, $w_{22} = 0.35$

$w_{21} = 0.25$, $w_{13} = 0.4$

$w_{12} = 0.3$, $w_{23} = 0.45$

Taking training set input values as $x_1 = 1$ and $x_2 = .1$

Calculate $d_1 = \sqrt{(x_1 - w_{11})^2 + (x_2 - w_{21})^2} = 1.096$

$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = 0.955$

$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = 0.8139$

According to Kohenen, update is done with least value to corresponding weights.

Here, d_3 has least value so only weights of w_{13} and w_{23} are updated.

$\Delta w_{13} = \alpha\,(x_1 - w_{13}) = 0.1\,(1 - 0.4) = 0.06$

$\Delta w_{23} = \alpha\,(x_2 - w_{23}) = 0.055$

Update weight

$$\begin{bmatrix} w_{13} \\ w_{23} \end{bmatrix}_{new} = \begin{bmatrix} w_{13} \\ w_{23} \end{bmatrix}_{old} + \begin{bmatrix} \Delta w_{13} \\ \Delta w_{23} \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.45 \end{bmatrix} + \begin{bmatrix} 0.06 \\ 0.055 \end{bmatrix} = \begin{bmatrix} 0.46 \\ 0.505 \end{bmatrix}$$

Here, we see that weights are updated by competition (least value obtained during calculation)

7.1.10 Hopfield Network

Hopfield network is a special kind of neural network whose response is different from other neural networks. It is calculated by converging iterative process. It has just one layer of neurons relating to the size of the input and output, which must be the same. When such a network recognizes, for example, digits, we present a list of correctly rendered digits to the network. Subsequently, the network can transform a noise input to the relating perfect output.

In 1982, John Hopfield introduced an artificial neural network to collect and retrieve memory like the human brain. Here, a neuron is either on or off the situation. The state of a neuron(on +1 or off 0) will be restored, relying on the input it receives from the other neuron. A Hopfield network is at first prepared to store various patterns or memories. Afterward, it is ready to recognize any of the learned patterns by uncovering partial or even some corrupted data about that pattern, i.e., it eventually settles down and restores the closest pattern. Thus, similar to the human brain, the Hopfield model has stability in pattern recognition.

A Hopfield network is a single-layered and recurrent network in which the neurons are entirely connected, i.e., each neuron is associated with other neurons. If there are two neurons i and j, then there is a connectivity weight w_{ij} lies between them which is symmetric $w_{ij} = w_{ji}$.

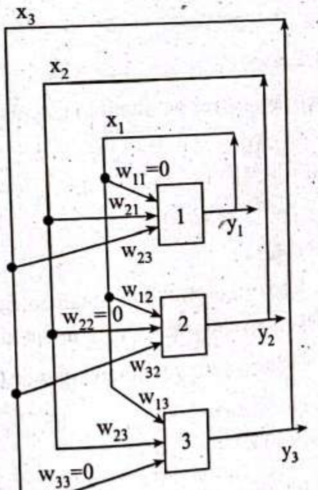


Figure 7.5: Hopfield network

Training algorithm:

For storing bipolar input patterns weight matrix is given by

$$W_{ij} = \sum_{P=1}^{P} s_i(P)\, s_j(P) \text{ for } i \neq j$$

where $s_j(P)$ is input pattern, $s_i(P)$ is transverse of $s_j(P)$ & weights have no self-connection

Testing algorithm:

Step 0: Initialize weights to store pattern.

Step 1: When activation of network is not converged, performs step 2-6.

Step2: Make initial activations of network equal to external input vectors x:

$y_i = x_i\,[i = 1 \text{ to } n]$

Step 3: Perform steps 5-6 for each unit y. (units are updated in random order)

Step 4: Calculate net input of the network.

$y_{in} = y_i + \Sigma_j y_j w_{ji}$

Apply activation over net input to calculate output.

$$y_i = \begin{cases} 1 \text{ if } y_{in} > \theta_i \\ y_i \text{ if } y_{in} = \theta_i \\ 0 \text{ if } y_{in} < \theta_i \end{cases}$$

θ_i is 0 basically.

Step 5: Now feedback (transmit) the obtained output to all other units. Thus, activation vectors are updated.

Step 6: Finally, test the network for convergence (are input vectors equal).

Example 7.5:

Construct an auto associative discrete hop field network with input vector [1 1 1 –1]. Test discrete Hopfield Network with missing entries in first and second components of stored vector.

Solution:

Step 1: Input vector is x = [1 1 1 –1] in bi-polar representation. On binary representation it is written as x = [1 1 1 0].

Now, weight matrix,

$$W = \Sigma\, S^T(P)\, S(P) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} [1\ 1\ 1\ -1] = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

Weight matrix with no self-connection is W= W – I. So,

$$W = W - I = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Binary representation for [1 1 1 –1] is [1 1 1 0]

Step 2: From the question, it says first and second components are missing so we have it as zero. So, input vector is y = [0 0 1 0]. We have to find the missing entries with the help of weight matrix.

Step 3: Choosing unit y_1, for updating its activation.

$$y_{in1} = y_1 + \sum_{j=1}^{4} y_i w_{ji} = 0 + [0\ 0\ 1\ 0] \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix} = 0 + 1 = 1$$

Applying activation $y_{in1} > 0$, $y_1 = 1$. Broadcasting y, we get y = [1 0 1 0] → No convergence (this output is not equal to desired vector i.e., [1 1 1 0]).

Step 4: Choosing unit y_4 for updating its activation.

$$y_{in4} = y_4 + \sum_{j=1}^{4} y_i w_{j4}$$

$$= 0 + [1\ 0\ 1\ 0] \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \end{bmatrix} = 0 - 1 - 1 = -2 < 0.$$

$y_{in4} < 0 \Rightarrow y_4 = 0$, therefore y = [1 0 1 0] → No convergence

Scanned with CamScanner

Step 5: Choosing unit y_2 for updating, its activation,

$$y_{in2} = y_2 + \sum_{j=1}^{4} y_j w_{j2}$$

$$= 0 + [1\ 0\ 1\ 0]\begin{bmatrix}1\\0\\1\\-1\end{bmatrix} = 0 + 2 = 2$$

$y_{in} > 0 \Rightarrow y_2 = 1,$

Therefore $y = [1\ 1\ 1\ 0]$

Now, it converges with vector x.

In this way, we can converge the missing entries with the help of Hopfield Neural Network.

7.1.11 Boltzmann Machine

These are stochastic learning processes having recurrent structure and are the basis of the early optimization techniques used in ANN. Boltzmann machine was invented by Geoffrey Hinton and Terry Sejnowski in 1985. More clarity can be observed in the words of Hinton on Boltzmann machine.

"A surprising feature of this network is that it uses only locally available information. The change of weight depends only on the behavior of the two units it connects, even though the change optimizes a global measure" - Ackley, Hinton 1985.

Some important points about Boltzmann Machine are:

- They use recurrent structure.
- They consist of stochastic neurons, which have one of the two possible states, either 1 or 0.
- Some of the neurons in this are adaptive (free state) and some are clamped (frozen state).
- If we apply simulated annealing on discrete Hopfield network, then it would become Boltzmann machine.

The following diagram shows the architecture of Boltzmann machine. It is clear from the diagram, that it is a two-dimensional array of units. Here, weights on interconnections between units are -p where p > 0. The weights of self-connections are given by b where b > 0.

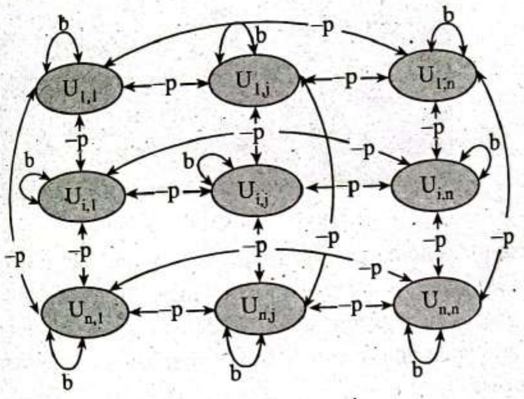


Figure 7.6: Boltzmann machine

Objective of Boltzmann machine:

The main purpose of Boltzmann machine is to optimize the solution of a problem. It is the work of Boltzmann machine to optimize the weights and quantity related to that particular problem.

7.2 Expert System

An *expert (human)* is an individual who has a superior capability understanding of a problem. Examples: a doctor, financial advisor, an expert in car engines, etc. So, human collect the information from expert and designed a machine which can perform like that of expert. Hence, an *expert system* is a computer system whose performance is guided by specific, expert knowledge in solving problems. This expert system it simulates the decision-making process of a human expert in a specific domain. Expert system is one of the early (large-scale) successes of artificial

Scanned with CamScanner

intelligence. So collectively we can say an expert system is an "intelligent" program that solves problems in a narrow problem area by using high-quality, specific knowledge rather than an algorithm.

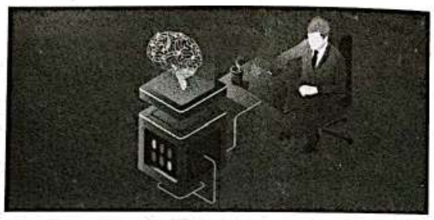


Figure 7.7: Expert system

Expert systems are used by most of the large or medium sized organization as a major tool for improving productivity and quality. An expert system's knowledge is obtained from expert sources and code in a form which is suitable for the system to use in its process.

Expert systems are designed to solve complex problems by reasoning about knowledge, like an expert, and not by following the procedure of a developer as is the case in conventional programming.

Features of an expert system:

Expert system has the following important features:

- **High performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

7.2.1 History of Expert Systems

In 1966, DENDRAL was developed at Stanford which analyzes mass spectrographic, NMR data and infers possible structures of unknown chemicals. Later in 1971, MYCIN was developed at Stanford diagnose and treat infectious blood diseases. In 1980, XCON was developed at CMU which is one of the first commercial expert systems. Some other popular expert system are PROSPECTO which analyzes geological data and GASOIL was designed to gas-oil separation systems for offshore oil platforms.

7.2.2 Comparison of Human Expert and Expert System

Parameters	Human Expert	Expert System
Time availability	Working day	Any time
Geographical	Specific location	Wherever
Security	Not replaceable	Can be replaced
Perishable	Yes	No
Performance	Variable	Consistent
Speed	Variable	Consistent and more fast

The fundamental reason why the expert system was developed are discussed below:

1. **No memory limitations:** It can store as much data as required and can memorize it at the time of its application. But for human experts, there are some limitations to memorize all things at every time.
2. **High efficiency:** If the knowledge base is updated with the correct knowledge, then it provides a highly efficient output, which may not be possible for a human.
3. **Expertise in a domain:** There are lots of human experts in each domain, and they all have different skills, different experiences, and different skills, so it is not easy to get a final output for the query. But if we put the knowledge

Scanned with CamScanner

gained from human experts into the expert system, then it provides an efficient output by mixing all the facts and knowledge

4. **Not affected by emotions:** These systems are not affected by human emotions such as fatigue, anger, depression, anxiety, etc.. Hence the performance remains constant.

5. **High security:** These systems provide high security to resolve any query.

6. **Considers all the facts:** To respond to any query, it checks and considers all the available facts and provides the result accordingly. But it is possible that a human expert may not consider some facts due to any reason.

7. **Regular updates improve the performance:** If there is an issue in the result provided by the expert systems, we can improve the performance of the system by updating the knowledge base.

7.2.3 Comparison of Conventional Systems and Expert System

Parameters	Conventional System	Expert System
Component placement	Information and processing combined in a single sequential program	Knowledge base is separated from the processing block or inference.
Accuracy	Program is never wrong	Program can make a mistake
Data requirement	Need all the input data	Not necessarily need all inputs data or facts
Modification	Modification in program is inconvenient	Modification in the program can be done with ease
Completeness	The system works if it is complete	The system can work with little rules
Main objective	Efficiency	Effectiveness

2.4 Components of Expert System

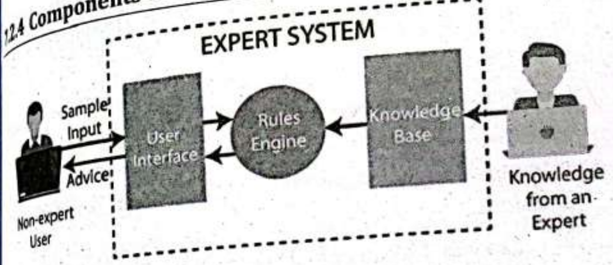


Figure 7.7: Architecture of an expert system

For a workable expert system, we need more than a set of rules. We need mainly three components:

- User interface
- Inference engine
- Knowledge base

1. **User interface**

   With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

2. **Inference engine (rules of engine)**

   The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.With the help of an inference engine, the system extracts the knowledge from the knowledge base.There are two types of inference engine:

- **Deterministic inference engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on facts and rules.
- **Probabilistic inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

- **Forward chaining:** It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.
- **Backward chaining:** It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

3. **Knowledge base**

The *knowledge base* is a type of storage that stores knowledge acquired from the different experts of the particular domain. It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the expert system.It is similar to a database that contains information and rules of a particular domain or subject. One can also view the knowledge base as collections of objects and their attributes. For e.g., a lion is an object and its attributes are "it is a mammal", "it is not a domestic animal" etc.

Components of knowledge base:

- **Factual knowledge:** The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.
- **Heuristic knowledge:** This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge representation: It is used to formalize the knowledge stored in the knowledge base using the if-else rules.

Knowledge acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

Types of knowledge:

- **Procedural knowledge:** "Knowing how" i.e., procedures include step by step sequences and how-to types of instructions. It may contain explanations. E.g.,a mobile robot that navigates building contains procedures such as "navigate to room", "plan a path", etc.
- **Declarative knowledge:** "Knowing what" i.e., descriptive representation of knowledge. It tells us facts, about truths and associations. E.g., "there is positive association between smoking and cancer". E.g., A mobile robot with facts to move forward, backward, turn left, turn right. Important in initial stage of knowledge acquisition.
- **Episodic knowledge:** Experiential
- **Meta-knowledge:** Knowledge about knowledge

Knowledge Base Size

Frederick Hayes-Roth developed the following heuristics about knowledge bases:

- A convincing demonstration of a knowledge system's power requires about 250 rules.
- An expert level of competence in a narrow area requires about 500 to 1000 rules.
- Expertise in a profession requires about 10,000 rules.
- The limit of human expertise is about 100,000 rules.

Scanned with CamScanner

Rule Types

- Relationship (FACT): IF the battery is dead THEN the car will not start
- Recommendation: IF the car will not start THEN take a cab
- Directive: IF the car will not start AND the fuel system is ok THEN check out the electrical system
- Heuristic: IF the car will not start AND the car is a 1957 Ford THEN check the float

Rule Uncertainty

- Vagueness/uncertainty rules: IF inflation is HIGH THEN interest rates might be high
- Re-write using a confidence quantifier: IF inflation is HIGH THEN interest rates are high (CQ = 0.8)

Meta Rules

- Rules that express knowledge about how other knowledge should be used.

  E.g., **IF** the car isn't starting **AND** the electrical system is operating properly **THEN** use fuel system.

7.2.5 Steps in Expert System Development

The steps for the development of expert system are explained as follows:

1. **Knowledge acquisition**

   Knowledge acquisition is the first step or process where we define the rules and ontologies required for a knowledge-based system. It helps to describe the initial tasks associated with developing an expert system that include finding and interviewing domain experts and capturing their knowledge via rules, objects, and frame-based ontologies. The expert sources can be domain specialist, articles, journal, database, etc.

Sources of knowledge:

- **Expert:** It is primary source.
- **End users:** They usually have a good overview of the problem domain and may provide valuable insight during initial investigations.
- **Secondary/tertiary experts:** They can provide specialized knowledge on sub-problems. Some time they may give rise to conflicting advice also.
- **Literature:** This is another source which is taken from reports, guidelines, books, manuals, etc. It provides background and insight in early stages.

2. **Knowledge representation**

   Collected data and information about the world need to be represented in such a from which will help a computer system to understand and later utilize it to solve complex task. It is a set of ontological commitment. We can use declarative knowledge to represent the acquired knowledge.

3. **Knowledge inference**

   It refers to acquiring new knowledge from existing facts based on certain rules and constraints. Mostly rule-based reasoning (forward chaining/backward chaining) is used for inferencing.

4. **Knowledge transfer**

   Knowledge transfer is the practical problem of transferring knowledge from one part of the organization to another. Knowledge transfer seeks to organize, create, capture or distribute knowledge and ensure its availability for future users.

7.2.6 Participants in the Development of Expert System

There are three primary participants in the building of expert system:

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

7.2.7 Advantages and Disadvantages of Expert System

Expert systems have the following advantages:

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

However, the disadvantages of expert systems are:

- The response of the expert system may get wrong if the knowledge base contains the wrong information.
- Like a human being, it cannot produce a creative output for different scenarios.
- Its maintenance and development costs are very high.
- Knowledge acquisition for designing is much difficult.

- For each domain, we require a specific ES, which is one of the big limitations.
- It cannot learn from itself and hence requires manual updates.

7.2.8 Applications of Expert System

Following are the applications of expert systems:

- Business
- Manufacturing
- Medicine
- Engineering
- Applied science
- Military
- Space
- Transportation
- Education
- Image analysis
- Chemical structure
- Agriculture
- Robotics and many more

7.2.9 Some Popular Expert Systems

1. **DENDRAL**

It is the first expert system developed in late 1965 by Edward Feigenbaum (AI researcher) and Joshua Lederberg (Geneticist) which was designed to analyze mass spectra. The substance to be analyzed might be a complicated compound of carbon, hydrogen, and nitrogen. Starting from spectrographic data obtained from the substance, DENDRAL would hypothesize the substance's molecular structure. DENDRAL's performance rivaled that of chemist expert and the program was used in industry and in academia. The main concept behind this project is to use heuristic knowledge obtained from experienced chemists and used forward chaining for reasoning

2. **MYCIN**

   It is an expert system designed in 1972 at Stanford University for treating blood infections. It diagnoses patients based on reported symptoms and medical test results. It could ask for more information and lab test results for diagnosis and recommend a course of treatment. If patient request for explanation then, MYCIN would explain the reasoning that lead to its diagnosis and recommendation. It used 500 production rules, MYCIN operated roughly the same level of competence as human specialists in blood infections. In this project backward chaining was used for reasoning.

7.3 Natural Language Processing

Language is the means for communication for humans. By studying language, we come to understand more about the world. If we can succeed at building computational mode of language, we will have a powerful tool for communicating around the world. We look at how we can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems. *Natural language processing (NLP)* is the process of computer analysis of input provided in a human language (natural language), and conversion of this input into a useful form of representation.

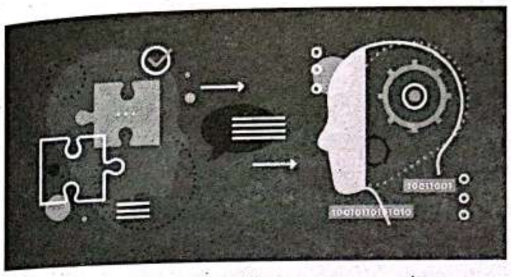


Figure 7.8: Natural language processing

NLP is one field of AI that processes or analyzes written or spoken language. It involves processing of speech, grammar and meaning. NLP is composed of two part: NLU (Natural Language Understanding) and NLG (Natural Language generation). Understanding of any language requires detailed knowledge of that language.

7.3.1 Components of NLP

There are two components of NLP:

1. **Natural Language Understanding (NLU)**

   Natural language understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

   NLU mainly used in business applications to understand the customer's problem in both spoken and written language.

   NLU involves the following tasks:

   - It is used to map the given input into useful representation.
   - It is used to analyze different aspects of the language.

2. **Natural Language Generation (NLG)**

   Natural language generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Scanned with CamScanner

7.3.2 Natural Language Processing Steps/Processes

The input of NLP system can be in the form of written text or speech. Quality of input decides the possible errors in language processing i.e., high quality input leads to correct language understanding and vice-versa. After that inputs are segmented in which text inputs are divided in segments (chunks) and the segments are analyzed. Each chunk is called a *frame*.

After that each segment is analyzed through various step as shown in figure below:

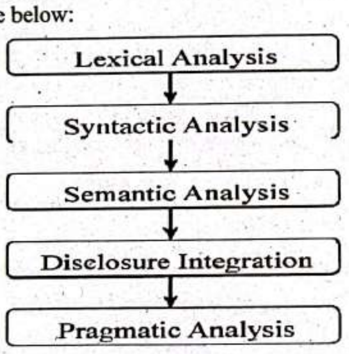
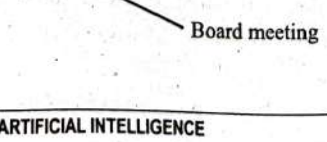


Figure 7.9: Steps in natural language processing

1. **Lexical or Morphological Analysis**

   The lexicon of a language is its vocabulary that includes its words and expressions. Morphology is the identification, analysis and description of structure of words. The lexical analysis objective is to divide the text into paragraphs, sentences, words and analyze the structure of words. The lexical analysis cannot be performed in isolation from morphological and syntactic analysis.



For example board can have different meaning on different context; board might refers to the board in which we write or the board meeting(meeting of the board members of the organization).

2. **Syntactic Analysis**

   Syntactic analysis takes an input sentence and produces a representation of its grammatical structure. A grammar describes the valid parts of speech of a language and how to combine them into phrases. The English grammar is nearly context free. A computer grammar specifies which sentences are in a language and their parse trees. A parse tree is a hierarchical structure that shows how the grammar applies to the input. Each level of the tree corresponds to the application of one grammar rule. The sentences such as "the school goes to boy" is rejected by English syntactic analyzer. For example,

   The red bird pecks the grain.

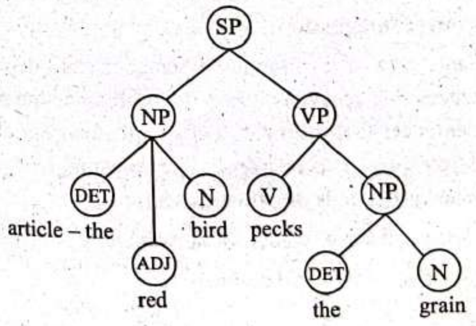


Figure 7.10: Parse tree

3. **Semantic Analysis**

   Semantic analysis is a process of converting the syntactic representations into a meaningful representation. This analysis involves the following tasks:

   - Word sense determination
   - Sentence level analysis

Scanned with CamScanner

Word sense:

Words have different meanings in different contexts.

E.g., Ram loves fish.

Here the meaning of loves can be 'likes' and also 'eats'.

E.g., Mary had a bat in her office.

Here, bat can be 'a baseball thing' or 'a flying mammal'.

Sentence level meaning

Once the words are understood, the sentence must be assigned some meaning.

E.g., Ram ate hot ice-cream.

Semantic analyzer rejects this sentence as it states hot ice-cream would make no sense.

E.g., Colorless blue idea.

This would be rejected by the analyzer as colorless blue do not make any sense together.

4. **Discourse integration**

The meaning of an individual sentence may depend on the sentences that precedes it and may influence the meaning of the sentences that follow it. It covers the concept of pronouns and referring expressions. The meaning of individual sentence is depends on previous sentence.

E.g., Bill had a red balloon. John wanted it.

Here, "it" means "a red ballon".

5. **Pragmatic Analysis**

Pragmatics comprises aspects of meaning that depend upon the context or upon facts about real world. These aspects include:

- Logical inferences, that can be drawn from the meanings of a set of propositions.

E.g., Close the window

Here, it should have been interpreted as a request rather than an order.

E.g., Jack fell. Jill brought him a band-aid.

Here, it should be understood as: Jack got hurt and Jill wanted to help.

7.3.3 Natural Language Processing AIDS or Tools

NLP can be understood or analyzed using some aids or tools.

1. **Lexicon or Dictionary**

A *lexicon* defines the words of a language that a system knows about. This includes common words and words that are specific to the domain of the application. Entries include meanings for each word and its syntactic and morphological behavior.

2. **Morphological Analysis System**

Morphological analysis is the process of recognizing the suffixes and prefixes that have been attached to a word. We do this by having a table of affixes and trying to match the input as: prefixes + root + suffixes.

For example: adjective +ly mean adverb.

"-s, -es" can be either a plural noun or a verb.

"-d, -ed" can be either a past tense or a perfect participle.

Following are some examples in which a word can have multiple form or changes it's form.

E.g., <u>Green</u> can be transformed into <u>greenness</u> (adjective, noun)

E.g., <u>Boat</u> can be <u>bóats</u> (singular, plural)

E.g., <u>Bill</u> slept. <u>Bill's</u> bed. (subjective case, possessive case)

E.g., I <u>walk</u>. I am <u>walking</u>. (present, progressive)

E.g., I <u>walked</u>. I <u>will walk</u>. I <u>had been walking</u>. (past, future, past progressive).

Scanned with CamScanner

E.g., I walk. They walk. (first person singular, third person plural)

Problems in morphological analysis system:

- doubled consonants (sit, sitting)
- changing y to i (fly, flies)
- abbreviations (CAN)
- naming word (Cook)
- wrong spelled word (Hare instead hair)
- unrecognized words:
  - real words not in the dictionary
  - names, etc that cannot be listed (names, account nos., library call nos)

3. **Syntactic Analysis System**

   It takes an input sentence and produces a representation of its grammatic structure. Grammar (mostly context free grammar) and parser are used.

4. **Semantic Analysis System**

   It converts the syntactic representation into a meaning representation.

   It involves:

   - Word sense determination
   - Sentence level analysis
   - Knowledge Representation

5. **Pragmatic Analysis System**

   In this system, we study how knowledge about the world and language conventions interacts with the literal meaning which depends upon context and facts about real world. Pragmatic analysis system includes:

   - Pronouns and referring expressions.
   - Logical interfaces that can be drawn from meaning of a set of propositions.

- Inference about what the producer of a sentence is trying to do.

7.3.4 Natural Language Analysis Techniques

There are two commonly used techniques in natural language processing. They are :

1. **Keyword analysis or pattern matching**

   In this analysis, accept the given sentence as input then segment the sentence and identify keywords in each segment.

   - If keyword is present
     - If only one keyword is present, give suitable reply as of keyword
     - If more keywords are present, prioritize them and give suitable reply as of keyword
   - If no keyword present in the segment, give a random reply.

2. **Syntactic driven parsing technique**

   In his technique, words can fit together in higher level units such as phrases, clauses and sentences so interpretations of larger groups of words are built up out of the interpretation of their syntactic constituent words or phrases. As interpretation of input is done as a whole it can be obtained by application of grammar that determines what sentences are legal in the language that is being parsed.

7.3.5 Natural Language Processing Problems

1. The same expression means different things in different context.

   - Where's the water? (Chemistry lab? Must be pure)
   - Where's the water? (Thirsty? Must be drinking water)
   - Where's the water? (Leaky roof? It can be dirty)

Scanned with CamScanner

2. No natural language program can be complete because of new words, expression, and meaning can be generated quite freely
   - I'll fax it to you
3. There are lots of ways to say the same thing.
   - Ram was born on October 11.
   - Ram's birthday is October 11.
4. Sentence and phrases might have hidden meanings
   - Out of sight, out of mind means invisible, idiot respectively.
5. Problem due to syntax and semantics
6. Problem due to extensive use of pronouns (semantic issue)
   - E.g., Ravi went to the supermarket. He found his favorite brand of coffee in rack. He paid for it and left.
   - It denotes??
7. Use of grammatically incorrect sentence
   - He rice eats. (syntax issue)
8. Use of conjunctions to avoid repetition of phrases cause problem in NLP
   - Ram and hari went to restaurant. Ram had a cup of coffee while hari had tea.

7.3.5 Natural Language Processing Applications

Following are the applications of NLP:

i. **Question answering**

Question answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

ii. **Spam detection**

Spam detection is used to detect unwanted e-mails getting to a user's inbox.

iii. **Sentiment analysis**

Sentiment analysis is also known as *opinion mining*. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (natural language processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)

iv. **Machine translation**

Machine translation is used to translate text or speech from one natural language to another natural language.

Example: Google translator

v. **Spelling correction**

Microsoft corporation provides word processor software like MS-Word, PowerPoint for spelling correction.

vi. **Speech recognition**

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

vii. **Chatbot**

Implementing the Chàtbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

viii. **Information extraction**

Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

Scanned with CamScanner

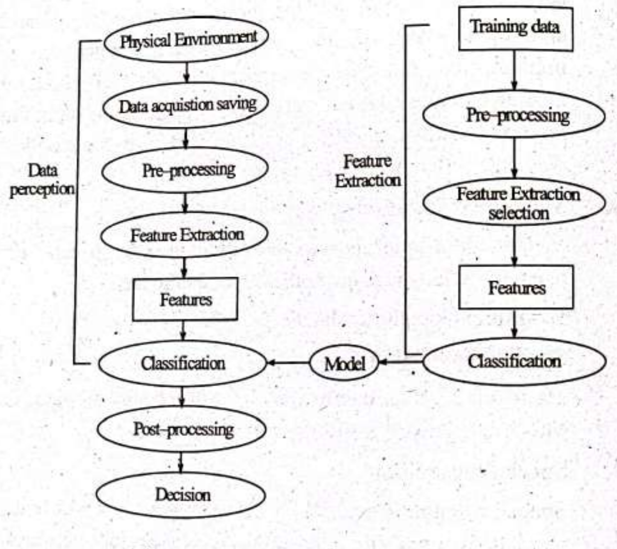


Figure 7.11: Pattern recognition system

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms.

Example: The colors on the clothes, speech pattern etc. In computer science, a pattern is represented using vector features values.

Pattern recognition is the process of recognizing patterns by using machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of the pattern recognition is its application potential. Examples: Speech recognition, speaker identification, multimedia document recognition (MDR), automatic medical diagnosis.

Data Acquisition

In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves classification and cluster of patterns.

In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.

Clustering generates a partition of the data which helps decision making for activity of interest to us. Clustering is used in an unsupervised learning.

Feature Extraction

Features may be represented as continuous, discrete or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

Example: Consider our face, then eyes, ears, nose, etc. are features of the face.

A set of features that are taken together, forms the features vector.

Example: In the above example of face, if all the features (eyes, ears, nose,etc) taken together then the sequence is feature vector ([eyes, ears, nose]). Feature vector is the sequence of a features represented as a d-dimensional column vector. In case of speech, MFCC (MelfrequencyCepstralCoefficent) is the spectral features of the speech. Sequence of first 13 features forms a feature vector.

Pattern recognition possesses the following features:

- Pattern recognition system should recognize familiar pattern quickly and accurate
- Recognize and classify unfamiliar objects

- Accurately recognize shapes and objects from different angles
- Identify patterns and objects even when partly hidden
- Recognize patterns quickly with ease, and with automaticity.

Training and Learning in Pattern Recognition

Learning is a phenomenon through which a system gets trained and becomes adaptable to give result in an accurate manner. Learning is the most important phase as how well the system performs on the data provided to the system depends on which algorithms used on the data. Entire dataset is divided into two categories, one which is used in training the model i.e. Training set and the other that is used in testing the model after training, i.e. testing set.

- **Training set**

  Training set is used to build a model. It consists of the set of images which are used to train the system. Training rules and algorithms used give relevant information on how to associate input data with output decision. The system is trained by applying these algorithms on the dataset, all the relevant information is extracted from the data and results are obtained. Generally, 80% of the data of the dataset is taken for training data.

- **Testing set**

  Testing data is used to test the system. It is the set of data which is used to verify whether the system is producing the correct output after being trained or not. Generally, 20% of the data of the dataset is used for testing. Testing data is used to measure the accuracy of the system. Example: a system which identifies which category a particular flower belongs to, is able to identify seven categories of flowers correctly out of ten and rest others wrong, then the accuracy is 70 %.

7.4.1 Real-Time Examples and Explanations

A pattern is a physical object or an abstract notion. While talking about the classes of animals, a description of an animal would be a pattern. While talking about various types of balls, then a description of a ball is a pattern. In the case balls considered as pattern, the classes could be football, cricket ball, table tennis ball etc. Given a new pattern, the class of the pattern is to be determined. The choice of attributes and representation of patterns is a very important step in pattern classification. A good representation is one which makes use of discriminating attributes and also reduces the computational burden in pattern classification.

An obvious representation of a pattern will be a vector. Each element of the vector can represent one attribute of the pattern. The first element of the vector will contain the value of the first attribute for the pattern being considered.

Example: While representing spherical objects, (25, 1) may be represented as a spherical object with 25 units of weight and 1 unit of diameter. The class label can form a part of the vector. If spherical objects belong to class 1, the vector would be (25, 1, 1), where the first element represents the weight of the object, the second element, the diameter of the object and the third element represents the class of the object.

7.4.2 Advantages and Disadvantages of Pattern Recognition

Advantages:

- Pattern recognition solves classification problems
- Pattern recognition solves the problem of fake bio-metric detection.
- It is useful for cloth pattern recognition for visually impaired blind people.
- It is used in speaker diarization.
- We can recognize particular object from different angle.

Disadvantages:

- Syntactic pattern recognition approach is complex to implement and it is very slow process.
- Sometime to get better accuracy, larger dataset is required.
- It cannot explain why a particular object is recognized. Example: my face vs my friend's face.

7.4.2 Applications of Pattern Recognition

Following are the applications of pattern recognition:

i. **Image processing, segmentation, and analysis**

Pattern recognition is used to give human recognition intelligence to machine which is required in image processing.

ii. **Computer vision**

Pattern recognition is used to extract meaningful features from given image/video samples and is used in computer vision for various applications like biological and biomedical imaging.

iii. **Seismic analysis**

Pattern recognition approach is used for the discovery, imaging and interpretation of temporal patterns in seismic array recordings. Statistical pattern recognition is implemented and used in different types of seismic analysis models.

iv. **Radar signal classification/analysis**

Pattern recognition and Signal processing methods are used in various applications of radar signal classifications like AP mine detection and identification.

v. **Speech recognition**

The greatest success in speech recognition has been obtained using pattern recognition paradigms. It is used in various algorithms of speech recognition which tries to avoid the

problems of using a phoneme level of description and treats larger units such as words as pattern.

vi. **Fingerprint identification**

The fingerprint recognition technique is a dominant technology in the biometric market. A number of recognition methods have been used to perform fingerprint matching out of which pattern recognition approaches is widely used.

7.5 Machine Vision

Machine vision is the ability of a computer to see with one or more digital cameras and performing analog to digital conversion (ADC) and digital signal processing (DSP). The resulting data will be passed onto a computer or a robot controller.

Why are we imparting our primary sense to machines? For machines to relate to human mind they should also perceive the visual world like us. This can be in the form of a small camera which helps the machine to see and understand the world around them. Machine vision is a mushrooming branch of AI that aims to give machine a sense of vision similar to that of humans. Integrating specialized neural network to machine helps them to identify and understand images from the real world.

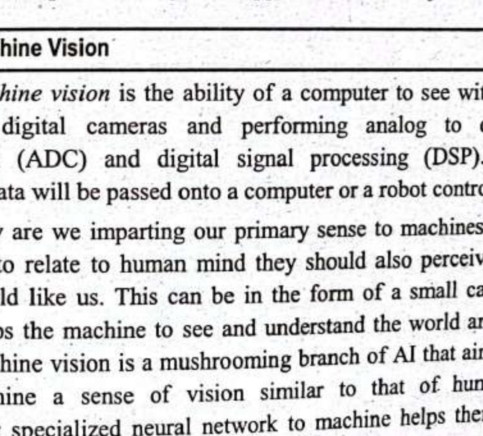


Figure 7.12: Machine vision

The basic image classification is easier for machines but it challenges them to extract meaning or information from abstract images. A common example to understand the difference between machine vision and human vision is comparing flight of plane and that of bird. Both will depend on principle of physics which help them to lift to air but that doesn't mean that plane would flap its wings to fly.

7.5.1 Some Examples of Machine Vision

Recently AI techniques have been integrated into several machine vision systems. Most of the application had tried to overcome the aforementioned challenges. The following are the applications of AI in machine vision with examples:

1. **PulnixZiCam** is a smart camera which doesn't require but presented in good and bad parts and learns how to differentiate them using a hardware neural network. It consists of a recognition engine that extracts 64 features including histogram, profiles and pixel samples, These features are passed to the neural network, which has 74 outputs. So instead of giving just a pass or fail output, ZiCAM can be trained to separate products into up to 74 classes.
2. **Smart Search** is an adaptive pattern-locating software, developed by Coreco Imaging as part of their Sherlock and MVTools vision packages. It comprises of a training wizard that is based on AI to facilitate system set-up. The developer only provides good and bad examples of the object to be checked and then Smart search learns the characteristics of that object automatically.
3. **Neuro Check Compact** is a smart camera programmed using mouse. The system consists of neurocheck package that is able to compute a number of object features from regions of interest in an image. These features are then passed to the classification module, which determines the type of the region being observed.
4. **The Sightech Eyebot** has features which uses a combination of fuzzy logic and neural networks, known as Neuro-Ram.

The Eyebot is available in two formats, the Shape Eyebot and the Spectrum Eyebot. The Shape Eyebot recognizes the shape of objects that are placed in front of the camera and can then detect the deviation from the learned shape. The Spectrum Eyebot learns the color of objects that are shown to it. The output of the system can be an integer from 0 to 99 that indicates the status of the shown product.

5. **ILIB software** contains pattern recognition, statistical, fuzzy logic and neural network tools. The neural network function of ILIB is provided by a multi-layer perceptron neural network with many configurable parameters, including learning rate, momentum, number of neurons, number of layers and activation functions. ILIB also contains pattern recognition techniques, such as minimum distance and K-nearest neighbour classification. Also, the distributions of feature vectors can be analyzed by many statistical techniques such as ANOVA. The software also incorporates fuzzy logic classification functions and genetic imaging. In genetic imaging, the user provides the system with the original image and the image that is desired from image processing. Then, ILIB uses genetic algorithms to derive the convolution filter or sequence of filters required to convert the original image to the target one.

The Common Vision Blox tool MANTO is developed based on research in AI, particularly work on the statistical theory of learning. It is able to cope with high degrees of image noise and is recommended for application in the food industry or for security technology.

The inclusion of technologies like neural networks, fuzzy logic and specialized hardware vision is bridging the gap between human vision and machine vision. This would revolutionize several industries like automotive, security and health. But eventually, robots will attain visual capabilities beyond human vision, making them more powerful to solve

Scanned with CamScanner

challenging task and operate autonomously. Time will tell whether AI is the best or worst thing to happen to humanity.

7.5.1 Applications of Machine Vision

Following are the applications of machine vision:

- Robotics
- Medicine
- Remote sensing
- Meteorology
- Quality inspection

EXERCISE

1. Distinguish between:
   a) Biological neural network and artificial neural network
   b) Hebb learning and perceptron
   c) Expert system vs management information system
2. Design neural network on the following criteria:
   a) McCulloch and Pitt for OR gate.
   b) Hebb neural network for AND gate.
   c) Perceptron for OR gate.
   d) Adaline neural network for AND gate.
   e) Back propagation for X-OR gate for data set $(1, 0) \rightarrow 1$
3. Write short notes on:
   a) Hopfield neural network
   b) Recurrent neural network
   c) Boltzmann machine
   d) Expert system
4. Explain steps involved in natural language processing with example.
5. Explain the importance of AI in
   a) Pattern recognition
   b) Machine vision
   c) Robotics

Scanned with CamScanner