S.Y. MCA

SEMESTER - IV (CBCS)

ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING

SUBJECT CODE: MCA405

Dr. Suhas Pednekar
Vice Chancellor
University of Mumbai, Mumbai

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai

Dr. Prakash Mahanwar
Director,
IDOL, University of Mumbai

Programme Co-ordinator : **Shri Mandar Bhanushe**
Head, Faculty of Science and Technology IDOL,
Univeristy of Mumbai – 400098

Course Co-ordinator : **Reshma Kurkute**
Assistant Professor,
Department - MCA, IDOL
University of Mumbai- 400098

Course Writers : **Mr. Ashish Shah**
B. Sc. I. T. Coordinator,
J. M. Patel College of Commerce

: **Mr. Sandeep Kamble**
Assistant Professor,
Cosmopolitan's Valia College

: **Ms.Anjali Gaikwad**
Assistant Professor
JES college of Commerce,
Science & IT Jogeshwari

: **Ms.Gauri Ulhas Ansurkar**
Assistant Professor,
KSD's Model College (Autonomous)

May 2022, Print - 1

Published by : Director,
Institute of Distance and Open Learning,
University of Mumbai,
Vidyanagari,Mumbai - 400 098.

DTP composed and Printed by:
Mumbai University Press
Vidyanagari, Santacruz (E), Mumbai- 400098

CONTENTS

S.Y. MCA

SEMESTER - IV (CBCS)

ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING

SYLLABUS

Sr. No.	Module	Detailed Contents	Hrs
1	**Introduction to AI**	**Artificial Intelligence** : Role of AI in engineering, AI in daily life, Intelligence and Artificial Intelligence, Different task domains of AI, Programming methods, Limitations of AI **Intelligent Agent:** Agent, Performance Evaluation, task environment of agent, Agent classification, Agent architecture	**05**
2	**Problem Solving**	**Problems, problem spaces and search:** Define the problem as a state space search, Production systems, Problem characteristics, Production system characteristic, Issues in design of search program **Search Techniques:** DFS, BFS, Hill Climbing	**06**
3	**Knowledge Representati on**	**Knowledge Representation**: Need to represent knowledge, Knowledge representation with mapping scheme, Properties of good knowledge-based system, Knowledge representation issues, AND-OR graph, Types of knowledge	**09**
4	**Concepts of Soft Computing**	**Soft Computing:** Hard computing Vs Soft Computing, Soft computing constituents – ANN, Fuzzy Logic, GA Applications of Soft Computing	**02**

5	**Neural Network**	**Artificial Neural Network:** Introduction,Fundamental Concept, Artificial Neural Network, Brain vs. Computer - Comparison Between Biological Neuron and Artificial Neuron, Basic Models of Artificial Neural Network **Supervised Learning Network-**Linear Separability, Perceptron Networks, Adaptive Linear Neuron (Adaline), Multiple Adaptive Linear Neurons, Back-Propagation Network. **Unsupervised Learning Networks-** MaxNet	**12**
6	**Fuzzy Logic**	**Introduction to Fuzzy Logic, Classical Sets and Fuzzy Sets:**Introduction to Fuzzy Logic, Classical Sets (Crisp Sets),Fuzzy Sets **Classical Relations and Fuzzy Relations:** Introduction, Cartesian Product of Relation, Classical Relation, Fuzzy Relations **Membership Functions:** Introduction, Features of the Membership Functions, Fuzzification, Methods of Membership Value Assignments **Defuzzification:** Introduction, Lambda-Cuts for Fuzzy Sets (Alpha-Cuts), Lambda-Cuts for Fuzzy Relations, Defuzzification Methods	**10**
7	**Fuzzy Inference System**	**Fuzzy Inference System**: Truth Values and Tables in Fuzzy Logic, Fuzzy Propositions, Formation of Rules, Decomposition of Rules (Compound Rules), Aggregation of Fuzzy Rules, Fuzzy Inference Systems (FIS)-Construction and Working Principle of FIS, Methods of FIS, Overview of Fuzzy Expert System	**04**
8	**Genetic Algorithm**	**Genetic Algorithm**: Basic concepts, Difference between genetic algorithm and traditional methods, Simple genetic algorithm, Working principle, Procedures of GA, Genetic operators- reproduction, Mutation, crossover.	**04**

1

INTRODUCTION TO AI

Unit Structure

1.1 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks-as, for example, discovering proofs for mathematical theorems or playing chess with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition.

1.2 ROLE OF AN AI ENGINEER

What is an AI engineer?

Artificial intelligence engineers are responsible for developing, programming and training the complex networks of algorithms that make up AI so that they can function like a human brain. This role requires combined expertise in software development, programming, data science and data engineering. Though this career is related to data engineering, AI engineers are rarely required to write the code that develops scalable data sharing. Instead, artificial intelligence developers locate and pull data from a variety of sources, create, develop and test machine learning models and then utilize application program interface (API) calls or embedded code to build and implement AI applications.

What does an AI engineer do?

AI engineers are primarily responsible for using various programming algorithms so that they can build, test and deploy AI models. Some of the other duties commonly found in an artificial intelligence engineer job description include:

- Coordinating with other team members
- Creating and managing the AI development process and overall infrastructure of the product
- Conducting statistical analysis and interpreting the results so that they can guide the organization's decision-making process
- Automating important infrastructure for the data science team
- Developing infrastructures for data transformation and ingestion
- Building AI models
- Explaining the usefulness of the AI models they create to a wide range of individuals within the organization, including stakeholders and product managers
- Transforming the machine learning models into APIs that other applications can interact with.

Why are AI engineers important?

Machine learning and artificial intelligence are developing specialties that can have a large impact on the overall success of an organization. That's because information technology professionals that decide to pursue a career in AI can develop advanced machine learning models that have the ability to provide valuable recommendations and insight into future issues or decisions. Some of the fields that are utilizing this technology include:

- **Finance:** In the finance industry, many organizations are beginning to use AI to learn the habits of users so that they can better identify suspicious and fraudulent activity.

- **Manufacturing:** Manufacturing organizations have begun using AI models to rethink the supply chain, predict maintenance issues and integrate with technological systems. As a result, these companies can manufacture products more safely and inexpensively.

- **Health care:** The health. care industry has many processes that monopolize a great deal of time and resources. By using AI, organizations can reduce the cost and time associated with things like drug discovery.

- **Enterprises:** Many businesses are beginning to utilize AI so that they can identify important insights in otherwise unstructured data, such as social media.

1.3 AI IN DAILY LIFE

You probably heard the term “artificial intelligence” multiple times and thought you had nothing to do with it but that’s not the case. Artificial intelligence has a wide range of applications in our daily lives. While some refer to this as the “robots taking over the world in an evil genius way” scenario, there is no doubt that artificial intelligence has simplified our lives by saving us time, money, and energy.

Here are examples of artificial intelligence that you’re likely to come across regularly but aren’t aware of their AI aspect.

1. Social Media

Many individuals check their social media accounts, including Facebook, Twitter, Instagram, and others, daily. Not only is AI working behind the scenes to customize what you see on your feeds, but it’s also figuring out friend recommendations, recognizing and filtering out fake news.

2. Music and Media Streaming Services

The music and media streaming services that we use on a daily basis are other wonderful examples of AI. AI is making decisions for you whether you’re using Spotify, Netflix, or YouTube. These platforms create recommendations according to your tastes. The next time you play a recommended video on YouTube or watch a suggested show on Netflix, or any other media for that matter; remember that AI is at work.

3. Search Engines

The majority of us can’t go a day without looking for an answer or a product on search engines like Google, Yandex, or Bing. Without artificial intelligence, search engines would be unable to scan the whole internet and offer what you want. Those ads that appear to track your every move? These are enabled by AI, based on your search history, and personalized to you with the goal of presenting you with items that the algorithms believe you will value.

4. Navigation Apps

You need Artificial intelligence even on your everyday journey to and from work. Navigation applications such as Google Maps use AI to evaluate the speed of movement of traffic. It also analyses user-reported events, such as traffic accidents or road construction, to estimate how long it will take you to get to your location and recommend the quickest route.

5. Banking

In the banking and finance business, artificial intelligence is a hot topic. AI has an influence on the banking business, from fraud detection to customer service and investing. Automated emails from banks out of the ordinary transaction are an example of the use of AI.

6. Smart Devices

Many of the smart home products use artificial intelligence to learn our habits and automatically modify settings to make our experience as

seamless as possible. It may take a while to have a perfect AI-powered house but some steps will take us there. For example, there are smart thermostats that change the temperature according to your preferences, as well as smart lights that alter the colour and intensity of lights based on time. It won't be long before our primary interactions with all of our smart home gadgets are conducted entirely through AI.

7. Video Games

Various components driven by AI or similar applications may be found in a wide range of video games, including racing games, shooting games, and strategy games. The fundamental goal of incorporating AI into gaming is to provide a realistic gaming experience for players to compete on a digital platform. Some companies are currently developing computer games with the goal of studying their patterns in order to improve their algorithms.

8. Chatbots

Chatbots identify words and phrases in order to provide relevant information to clients with typical inquiries. Chatbots can be so accurate that it appears as though you're chatting with a real person. Chatbots try to imitate natural language by mimicking conversations while assisting with daily tasks like scheduling appointments, accepting orders, and responding to billing questions.

9. Healthcare

Artificial intelligence has provided new opportunities in healthcare. It's gotten a little simpler to identify and diagnose diseases thanks to the development of AI-powered robots. Furthermore, it contributes significantly by making the treatment and management processes more simplified. As a result, hospitals and healthcare organizations are quickly adopting AI-enabled technology to help with anything from research to disease diagnosis.

10. Security and Surveillance

While we may all argue about the ethics of employing a large surveillance system, there's no doubt that it's being used, and AI is playing a significant role in it. Since it is not possible for humans to keep monitoring multiple monitors at the same time, using AI makes perfect sense. With technological advances such as object recognition and facial recognition, it won't be long until all security camera feeds are watched by AI rather than humans. While it will take some time for AI to be completely integrated, it is going to be our future.

12. Smart Personal Assistants

With advancements in speech recognition technologies, we can now hold a full conversation with digital assistants, and the technology is nearly flawless. Voice assistants, such as Google Assistant, Alexa, and Siri, are the greatest AI examples in real life. They take your inquiry via voice, process it with your phone's Speech Recognition and Natural Language Processing technologies, and then deliver the results as speech or text. As their function gets increasingly important, these smart digital assistants

manage our digital lives. They help with a wide range of tasks. The following are some of the most typical activities they help with:

- Online shopping
- Controlling internet-enabled devices
- Setting alarms and reminder
- Reservations for taxis, planes, and trains
- Playing media

1.4 INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is intelligence?

All but the simplest human behaviour is ascribed to intelligence, while even the most complicated insect behaviour is never taken as an indication of intelligence. What is the difference? Consider the behaviour of the digger wasp, *Sphex ichneumoneus*. When the female wasp returns to her burrow with food, she first deposits it on the threshold, checks for intruders inside her burrow, and only then, if the coast is clear, carries her food inside. The real nature of the wasp's instinctual behaviour is revealed if the food is moved a few inches away from the entrance to her burrow while she is inside: on emerging, she will repeat the whole procedure as often as the food is displaced. Intelligence—conspicuously absent in the case of *Sphex*—must include the ability to adapt to new circumstances.

psychologists generally do not characterize human intelligence by just one trait but by the combination of many diverse abilities. Research in AI has focused chiefly on the following components of intelligence: learning, reasoning, problem solving, perception, and using language.

What is artificial intelligence?

According to the father of Artificial Intelligence, John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs".

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think. AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

Philosophy of AI

While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, *"Can a machine think and behave like humans do?"*

Thus, the development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

- To Create Expert Systems − The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.
- To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

1.5 DIFFERENT TASK DOMAINS OF AI

Artificial intelligence is a computer system that is able to perform tasks that ordinarily require human intelligence. Artificial intelligence systems are critical for companies that wish to extract value from data by automating and optimizing processes or producing actionable insights.

There are certain domains of artificial intelligence on which we can create our expertise

1. Machine learning
2. Deep learning
3. Robotics
4. Expert systems
5. Fuzzy logic
6. Natural language processing
7. Computer vision

1. Machine learning

Machine learning is a subset of artificial intelligence. Machine learning enables computers or machines to make data-driven decisions rather than being explicitly programmed for a certain task. These programs or algorithms are designed in a way that they learn and improve over time when are exposed to new data.

Different types of machine learning models

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Use cases

- Product recommendation on a shopping website.
- spam filter on email.
- Chatbots

2. Deep learning

Deep learning is artificial intelligence (AI) function that imitates the working of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has network capable of learning unsupervised

from data that is unstructured or unlabelled also known as deep neural learning or deep neural network.

Different types of deep learning models

- Autoencoders
- Deep belief net
- Convolutional neural network
- Recurrent neural network
- Reinforcement learning to neural network

Use cases

- Driverless vehicles
- Virtual assistants
- chatbots
- Medical research
- Facial recognition

3. Robotics

Robotics is a branch of engineering that involves the conception, design, manufacture, and operation of robots. This fields overlaps with electronics, computer science, artificial intelligence, mechatronics, nanotechnology and bioengineering.

Different types of robots

- Pr-programmed robots
- Humanoid robots
- Autonomous robots
- Teleoperated robots
- Augmenting robots

Use cases

- Manufacturing
- Logistics
- Healthcare
- Home

4. Expert system

An expert system is a program that uses artificial intelligence technology to simulate the knowledge and judgement of humans. Expert systems usually include a subject-specific knowledge base and can have additional modules added to expand their capacities.

Different types of expert systems

- Rule-based systems
- Frame-based systems
- Hybrid systems
- Model-based systems
- Off the shelf systems
- Custom made systems

Use cases

- In the medical field
- In the agriculture field
- In the education field

5. Fuzzy logic

Fuzzy logic is a method of reasoning that resembles human reasoning. The approach of fuzzy logic imitates the way of decision making in humans that involves all intermediate possibilities between digital values yes or no. The conventional logic block that a computer can understand takes precise input and produces a definite output as true or false which is equivalent to human's yes or no.

Different types of fuzzifier

- Singleton fuzzifier
- Gaussian fuzzifier
- Trapezoidal or triangular fuzzifier

Use cases

- Psychology
- Pattern recognition and classifications
- Securities
- Medical
- Marine
- Finance

6. Natural language processing

Natural language processing is a branch of artificial intelligence that helps the computers understand interpret and manipulate human language. Natural language processing draws from many disciplines including computers science and computational linguistics in its pursuit to fill the gap between human communication and computer understanding.

Different types of Natural Language Processing (NLP)

- Optical character recognition
- Speech recognition

- Machine translation
- Natural language generation
- Sentiment analysis
- Semantic search
- Machine learning

Use cases

- Email filter
- Smart assistants
- Search results
- Predictive text
- Language translation
- Digital phone calls
- Text analytics

7. Computer vision

Today, computer vision is one of the hottest subfields of artificial intelligence and machine learning given its wide variety of applications and tremendous potential. It's a goal to replicate the powerful capacities of human vision. Computer vision system must recognize the present objects and their characteristics such as shapes textures, colours, sizes, spatial arrangement, among other things to provide a description as complete as possible of the image.

Different techniques of computer vision

- Image classification
- Object detection
- Object tracking
- Semantic segmentation
- Instance segmentation

Use cases

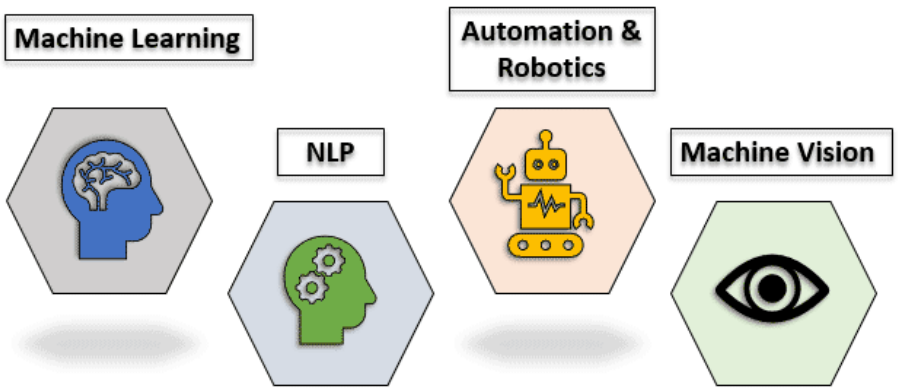
- Defect detection
- Metrology
- Intruder detection
- Assembly verification
- Screen reader

1.6 Programming methods

Artificial Intelligence can be divided into different categories based on the machine's capacity to use past experiences to predict future decisions, memory, and self-awareness. IBM came up with Deep Blue, a chess program that can identify the pieces in the chessboard. But it does not have the memory to predict future actions. This system though useful, but it cannot be adapted to another situation. Another type of AI system that

uses past experiences and has the bonus of a limited memory to predict the decisions. An example of this kind of AI system can be found in the functions of decision making in case of the self-driving cars. Here the observations help in the actions to be taken shortly, which does not get stored permanently as the observations change frequently. At the same time with the advancement in technology, it might be possible to have machines with a sense or consciousness where the machines understand the current state of things, which can be used to infer what is to be done. But such systems do not exist.

Below are the various categories of Artificial Intelligence:



1. Machine Learning

It is one of the applications of AI where machines are not explicitly programmed to perform certain tasks; rather, they learn and improve from experience automatically. Deep Learning is a subset of machine learning based on artificial neural networks for predictive analysis. There are various machine learning algorithms, such as Unsupervised Learning, Supervised Learning, and Reinforcement Learning. In Unsupervised Learning, the algorithm does not use classified information to act on it without any guidance. In Supervised Learning, it deduces a function from the training data, which consists of a set of an input object and the desired output. Reinforcement learning is used by machines to take suitable actions to increase the reward to find the best possibility which should be taken in to account.

2. NLP (Natural Language Processing)

It is the interactions between computers and human language where the computers are programmed to process natural languages. Machine Learning is a reliable technology for Natural Language Processing to obtain meaning from human languages. In NLP, the audio of a human talk is captured by the machine. Then the audio to text conversation occurs, and then the text is processed where the data is converted into audio. Then the machine uses the audio to respond to humans. Applications of Natural Language Processing can be found in IVR (Interactive Voice Response) applications used in call centres, language translation applications like Google Translate and word processors such as Microsoft Word to check the accuracy of grammar in text. However, the nature of human languages makes the Natural Language Processing difficult because of the rules

which are involved in the passing of information using natural language, and they are not easy for the computers to understand. So NLP uses algorithms to recognize and abstract the rules of the natural languages where the unstructured data from the human languages can be converted to a format that is understood by the computer.

3. Automation and Robotics

The purpose of Automation is to get the monotonous and repetitive tasks done by machines which also improve productivity and in receiving cost-effective and more efficient results. Many organizations use machine learning, neural networks, and graphs in automation. Such automation can prevent fraud issues while financial transactions online by using CAPTCHA technology. Robotic process automation is programmed to perform high volume repetitive tasks which can adapt to the change in different circumstances.

4. Machine Vision

Machines can capture visual information and then analyse it. Here cameras are used to capture the visual information, the analogue to digital conversion is used to convert the image to digital data, and digital signal processing is employed to process the data. Then the resulting data is fed to a computer. In machine vision, two vital aspects are sensitivity, which is the ability of the machine to perceive impulses that are weak and resolution, the range to which the machine can distinguish the objects. The usage of machine vision can be found in signature identification, pattern recognition, and medical image analysis, etc.

1.7 LIMITATIONS OF AI

1. One of the main barriers to implementing AI is **the availability of data**. Data is often siloed or inconsistent and of poor quality, all of which presents challenges for businesses looking to create value from AI at scale. To overcome this, you should have a clear strategy from the outset for sourcing the data that your AI will require.

2. Another key roadblock to AI adoption is the **skills shortage** and the availability of technical staff with the experience and training necessary to effectively deploy and operate AI solutions. Research suggests experienced data scientists are in short supply as are other specialised data professionals skilled in machine learning, training good models, etc.

3. **Cost** is another key consideration with procuring AI technologies. Businesses that lack in-house skills or are unfamiliar with AI often have to outsource, which is where challenges of cost and maintenance come in. Due to their complex nature, smart technologies can be expensive and you can incur further costs for repair and ongoing maintenance. The computational cost for training data models etc can also be an additional expense.

4. Software programs need regular upgrading to adapt to the changing business environment and, in case of breakdown, present a **risk** of losing code or important data. Restoring this is often time-consuming and costly. However, this risk is no greater with AI than with other software development. Provided that the system is designed well and that those procuring AI understand their requirements and options, these risks can be mitigated.

❄❄❄❄❄❄❄

2

INTELLIGENT AGENT

Unit Structure

2.1 AGENT

An intelligent agent (IA) is an entity that make decisions, that enables artificial intelligence to be put into action. It can also be described as a software entity that conducts operations in the place of users or programs after sensing the environment. It uses actuators to initiate action in that environment.

What is an Intelligent Agent (IA)?

This agent has some level of autonomy that allows it to perform specific, predictable, and repetitive tasks for users or applications. It's also termed as 'intelligent' because of its ability to learn during the process of performing tasks. The two main functions of intelligent agents include perception and action. Perception is done through sensors while actions are initiated through actuators. Intelligent agents consist of sub-agents that form a hierarchical structure. Lower-level tasks are performed by these sub-agents. The higher-level agents and lower-level agents form a complete system that can solve difficult problems through intelligent behaviours or responses.

Intelligent agents have the following distinguishing characteristics:

They have some level of autonomy that allows them to perform certain tasks on their own. They have a learning ability that enables them to learn even as tasks are carried out. They can interact with other entities such as agents, humans, and systems. New rules can be accommodated by intelligent agents incrementally. They exhibit goal-oriented habits. They are knowledge-based. They use knowledge regarding communications, processes, and entities.

The structure of intelligent agents:

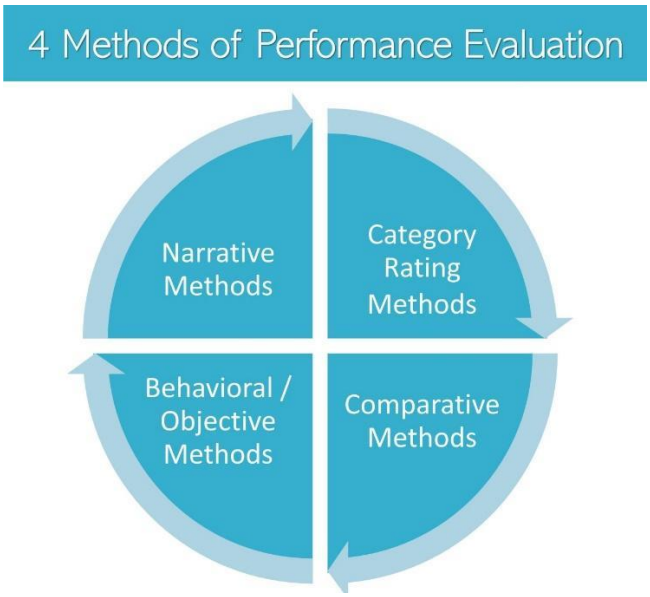
The IA structure consists of three main parts: architecture, agent function, and agent program.

1. **Architecture:** This refers to machinery or devices that consists of actuators and sensors. The intelligent agent executes on this machinery. Examples include a personal computer, a car, or a camera.
2. **Agent function:** This is a function in which actions are mapped from a certain percept sequence. Percept sequence refers to a history of what the intelligent agent has perceived.
3. **Agent program:** This is an implementation or execution of the agent function. The agent function is produced through the agent program's execution on the physical architecture.

2.2 PERFORMANCE EVALUATION

Methods of Performance Evaluation:

Performance evaluation is a systematic process of evaluating how well employees are performing their jobs. The appraisal is based on results obtained by the employee in his/her job, not on the employee's personality characteristics.



4 Methods of Performance Evaluation:

1. Category rating methods.
2. Comparative methods.
3. Behavioural/objective methods.
4. Narrative methods.

1. Category Rating Methods

The simplest methods for appraising performance are category rating methods, which require a manager to mark an employee's level of performance on a specific form divided into categories of performance.

The graphics rating scale and checklist are common category rating methods.

a. Graphics Rating Scale.
b. Checklist.

a. Graphics Rating Scale

Graphic rating scale is a scale that lists a number of traits and a range of performance for each that is used to identify the score that best describes an employee's level of performance for each trait.

b. Checklist

The checklist is composed of a list of statements or words. Raters check statements most representative of the characteristics and performance of employees. This method requires the rater to select statements or words that describe the employee's performance and characteristics. He does not evaluate employee performance. The rater is usually the immediate supervisor. He just supplies report about Performance Appraisals and the final rating is done by the HR department.

But without rater's knowledge, the HR department may assign weights to different items on the checklists according to each item's importance. The result is called a weighted checklist. The weight is the average score of the raters prior to use the checklist. The weights allow the rating to be quantified so total scores can be determined. The limitations of this method include use of personality criteria instead of performance criteria, misinterpretation of checklist items and the usage of improper weights by the HR department.

These statements are ordered progressively in terms of more or less of some property. An example-

i. David always goes to John.
ii. David often goes to John.
iii. David sometimes goes to John.
iv. David never goes to John.

Limitations of checklist

- It suffers from biases on the part of the rater because he cannot distinguish between positive and negative questions.
- A separate checklist must be prepared for different classes of jobs. It is expensive and time-consuming.

2. Comparative Methods

Comparative methods require that managers directly compare the performance of their employees against one another. For example, a data-entry operator's performance would be compared with that of other data-

entry operators by the computing supervisor. Comparative techniques include ranking, paired comparison, and forced distribution.

i. **Ranking:** The ranking method consists of listing all employees from highest to lowest in performance. The primary drawback of the ranking method is that the size of the differences among individuals is not well defined. For example, there may be little difference in performance between individuals ranked second and third, but a big difference in performance between those ranked third and fourth. Ranking scale gives rank by value, such as-

Very	Good	(5)
Good		(4)
Not	Bad	(3)
Bad		(2)
Very Bad		(1)

ii. **Forced Distribution:** Forced distribution performance appraisal method in which ratings of employees' performance are distributed along a bell-shaped curve. It is Similar to grading on a curve; predetermined percentages of rates are placed in various performance categories.

- Example:
  - 15% high performers.
  - 20% of high-average performers.
  - 35% average performers.
  - 20% of low-average performers.
  - 15% low performers.

3. Behavioural/Objective Methods

In an attempt to overcome some of the difficulties of the methods just described, several different behavioural approaches have been used. Behavioural approaches hold promise for some situations in overcoming some of the problems with other methods.

i. **Behavioural Rating Approaches:** Behavioural rating approaches attempt to assess an employee's behaviours instead of other characteristics. Some of the different behavioural approaches are behaviourally anchored rating scales (BARS), behavioural observation scales (BOS), and behavioural expectation scales (BES).

ii. **Management by Objectives (MBO): Management by objectives (MBO) specifies the performance goals** that individual hopes to attain within an appropriate length of time. The objectives that each manager sets are derived from the overall goals.

4. Narrative Methods

Managers and HR specialists frequently are required to provide written appraisal information.

Documentation and description are the essence of the critical incident, the essay, and the field review methods.

1. Critical Incident.
2. Essay.
3. Field Review.
4. 360° Feedback or Multi-source Appraisal.

These records describe an employee's actions rather than indicating an actual rating.

1. Critical Incident

In the critical incident method, the manager keeps a written record of both highly favourable and unfavourable actions in an employee's performance. When a "critical incident" involving an employee occurs, the manager writes it down. A list of critical incidents is kept during the entire rating period for each employee. The critical incident method can be used with other methods to document the reasons why an employee was rated in a certain way.

Critical Incident Method was first used by the US Army during World War Two. Now it is widely used in the business organizations to appraise employee performance. Under this method, the manager keeps a written record of highly favourable and unfavourable employee actions. The focus is on the key behaviours that make the difference between doing a job effectively or ineffectively. The statements are called critical incidents.

The supervisor records these incidents during the evaluation- period for each employee. Both positive and negative incidents are recorded. Supervisor keeps a log with positive and negative examples (critical incidents) of subordinate's work behaviour.

Advantages of Critical Incident method

1. This method is extremely useful for giving employees' job-related feedback.
2. It also reduces recency biases.
3. It identifies even rare events that might be missed by other methods which only focus on common and everyday events.
4. Data are collected from the respondent's perspective and in his or her own words.
5. It provides an objective basis for conducting a discussion of an individual's performance.
6. It forces the supervisor to evaluate subordinates on an ongoing performance basis. Drawbacks.

This method suffers from some limitations, which are listed below:

1. It is difficult to rate or rank employees relative to one another,
2. There may occur variations in how managers define a 'critical incident',
3. Most employee actions are not observed and may become different if observed,
4. Supervisors often do not record incidents as they occur,
5. Negative incidents are more noticeable than positive ones,
6. Very close supervision is required, which employees may not like,
7. It is time-consuming and burdensome for managers to write down critical behaviors of a large number of subordinates throughout the year.
8. Respondents may not be accustomed to or willing to take the time to say (or write) a complete story when describing a critical incident.

2.2.1 Essay

The essay, or "free-form," appraisal method requires the manager to write a short essay describing each employee's performance during the rating period. The rater usually is given a few general headings under which to categorize comments. The intent is to allow the rater more flexibility than other methods do. As a result, the essay is often combined with other methods. The rater writes a narrative describing an employee's strengths, weaknesses, past performance, potential and suggestions for improvement. It is simple and requires no complex forms or extensive training to complete. It can provide a good deal of information about an employee.

Because essays are unstructured, they are likely to vary widely in terms of length and content. Some raters are better writers than others are. It contains a subjective evaluation of the reported behaviour of an individual and may affect such important decisions as promotion and layoff. No attempt is made to evaluate an employee in a quantitative manner.

2.2.2 Field Review

This approach can include the HR department as a reviewer, or a completely independent reviewer outside the organization. In the field review, the outside reviewer becomes an active partner in the rating process. The outsider interviews the manager about each employee's performance and then compiles the notes from each interview into a rating for each employee. Then the rating is reviewed by the supervisor for needed changes. This method assumes that the outsider knows enough about the job setting to help supervisors give more accurate and thorough appraisals.

In this method, a skilled representative of the human resource department goes into the field and assists supervisors with their ratings. The personnel specialists solicit from the immediate supervisor specific information about the employee's performance. Then the experts prepare an evaluation based on this information. The evaluation is sent to the supervisor for

review, changes, approval, and discussion with an employee who was rated.

Since the skilled professional is completing the evaluation form, reliability and comparability are more likely, but the usage of skilled professionals makes this approach costly and impractical for many firms. And since the supervisor is the primary source of information, bias may still exist.

2.2.3 360° Feedback or Multi-source Appraisal

360-degree feedback, also known as multi-rater feedback, multisource feedback, or multi-source assessment. Most often, 360- degree feedback will include direct feedback from an employee's subordinates, peers, and supervisor(s), as well as a self-evaluation. It can also include, in some cases, feedback from external sources, such as customers and suppliers or other interested stakeholders.

360-degree evolution means the evaluation of an employee will be assessed based on ideas of many other different people for example customers, suppliers, peers, and direct reports. If the assessee is a manager, his/her staff will be often asked for feedback on how that manages is doing his task. It can of using 360-degree evolution it is vital that the process is implemented by the managers of the Human Resources Department. So that the subordinate reviewer's (or staff) are made sure that all their assessments on performance are kept anonymous.

The components of this evaluation are as follows;

- Self-evaluation
- Subordinate's appraisal
- Peer appraisal and
- Superior's appraisal

Advantages of this evaluation are as follows;

- Offer a more comprehensive view of the performance of employees.
- Improve Credibility of performance appraisal.
- Such a colleague's feedback will help strengthen self-development.
- Increase the responsibility of employees to their customers.
- The mix of ideas can give a more accurate assessment.
- Opinions gathered from lots of staff are sure to be more persuasive.
- Not only a manager should make assessments on its staff performance but other colleagues should do too.
- People who undervalue themselves are often motivated by feedback from others.
- If more staff takes part in the process of performance evaluation, the organizational culture of the company will become more honest.

2.3 TASK ENVIRONMENT OF AGENT

An environment in artificial intelligence is the surrounding of the agent. The agent takes input from the environment through sensors and delivers the output to the environment through actuators. There are several types of environments:

1. Fully Observable vs Partially Observable
2. Deterministic vs Stochastic
3. Competitive vs Collaborative
4. Single-agent vs Multi-agent
5. Static vs Dynamic
6. Discrete vs Continuous
7. Episodic vs Sequential

1. Fully Observable vs Partially Observable

When an agent sensor is capable to sense or access the complete state of an agent at each point in time, it is said to be a fully observable environment else it is partially observable. Maintaining a fully observable environment is easy as there is no need to keep track of the history of the surrounding. An environment is called **unobservable** when the agent has no sensors in all environments.

Examples:

Chess – the board is fully observable, so are the opponent's moves.

Driving – the environment is partially observable because what's around the corner is not known.

2. Deterministic vs Stochastic

When a uniqueness in the agent's current state completely determines the next state of the agent, the environment is said to be deterministic. The stochastic environment is random in nature which is not unique and cannot be completely determined by the agent.

Examples:

Chess – there would be only a few possible moves for a coin at the current state and these moves can be determined.

Self-Driving Cars – the actions of a self-driving car are not unique, it varies time to time.

3. Competitive vs Collaborative

An agent is said to be in a competitive environment when it competes against another agent to optimize the output. The game of chess is competitive as the agents compete with each other to win the game which is the output. An agent is said to be in a collaborative environment when multiple agents cooperate to

produce the desired output. When multiple self-driving cars are found on the roads, they cooperate with each other to avoid collisions and reach their destination which is the output desired.

4. Single-agent vs Multi-agent

An environment consisting of only one agent is said to be a single-agent environment. A person left alone in a maze is an example of the single-agent system. An environment involving more than one agent is a multi-agent environment. The game of football is multi-agent as it involves 11 players in each team.

5. Dynamic vs Static

An environment that keeps constantly changing itself when the agent is up with some action is said to be dynamic. A roller coaster ride is dynamic as it is set in motion and the environment keeps changing every instant. An idle environment with no change in its state is called a static environment. An empty house is static as there's no change in the surroundings when an agent enters.

6. Discrete vs Continuous

If an environment consists of a finite number of actions that can be deliberated in the environment to obtain the output, it is said to be a discrete environment. The game of chess is discrete as it has only a finite number of moves. The number of moves might vary with every game, but still, it's finite. The environment in which the actions performed cannot be numbered i.e. is not discrete, is said to be continuous. Self-driving cars are an example of continuous environments as their actions are driving, parking, etc. which cannot be numbered.

7. Episodic vs Sequential

In **Episodic task environment**, each of the agent's action is divided into an atomic incidents or episodes. There is no dependency between current and previous incident. In each incident agent receives input from environment and then performs corresponding action.

Example: Consider an example of **Pick and Place robot**, which is used to detect defective parts from conveyer belt. Here, every time robot(agent) will make decision on current part i.e. there is no dependency between current and previous decision.

In **Sequential environment**, previous decision can affect all future decisions. The next action of agent depends on what action he has taken previously and what action he is supposed to take in future.

Example:

Checkers- Where previous move can affect all the following moves.

2.4 AGENT CLASSIFICATION

Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below:

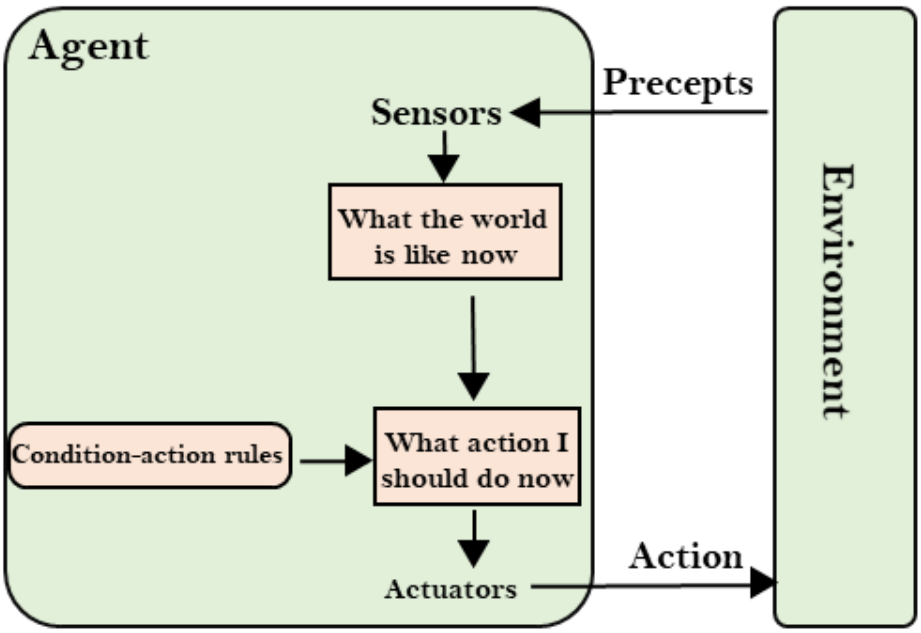
1. Simple Reflex Agent
2. Model-based reflex agent
3. Goal-based agents
4. Utility-based agent
5. Learning agent

1. Simple Reflex agent:

The Simple reflex agents is the simplest agents. These agents take decisions on the basis of the current percept and ignore the rest of the percept history. These agents only succeed in the fully observable environment. The Simple reflex agent does not consider any part of percept history during their decision and action process. The Simple reflex agent works on Condition-action rule, which means it maps the current state to action. Such as a Room Cleaner agent, it works only if there is dirt in the room.

Problems for the simple reflex agent design approach:

- They have very limited intelligence
- They do not have knowledge of non-perceptual parts of the current state
- Mostly too big to generate and to store.
- Not adaptive to changes in the environment.



2. Model-based reflex agent

The Model-based agent can work in a partially observable environment, and track the situation. A model-based agent has two important factors:

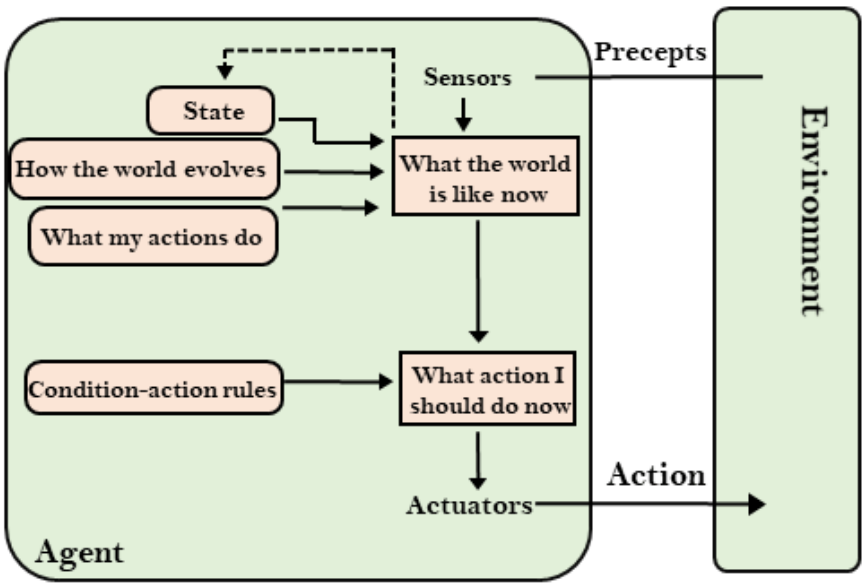
Model: It is knowledge about "how things happen in the world," so it is called a Model-based agent.

Internal State: It is a representation of the current state based on percept history.

These agents have the model, "which is knowledge of the world" and based on the model they perform actions.

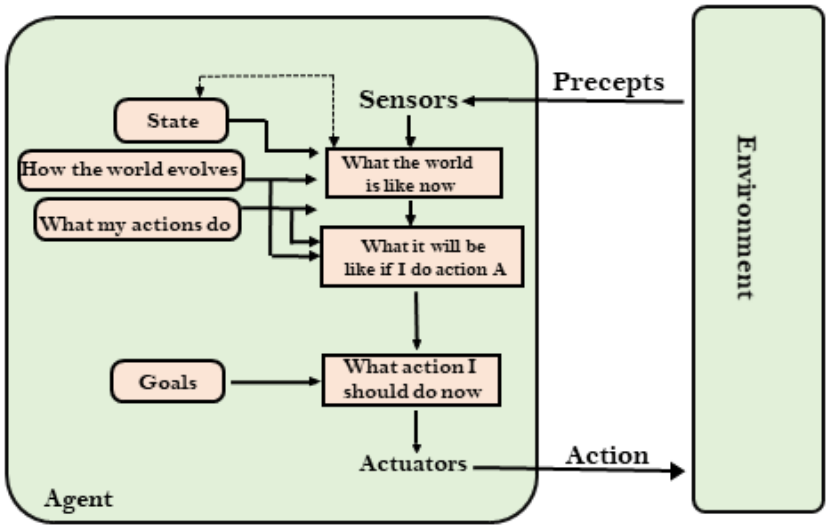
Updating the agent state requires information about:

- How the world evolves
- How the agent's action affects the world.



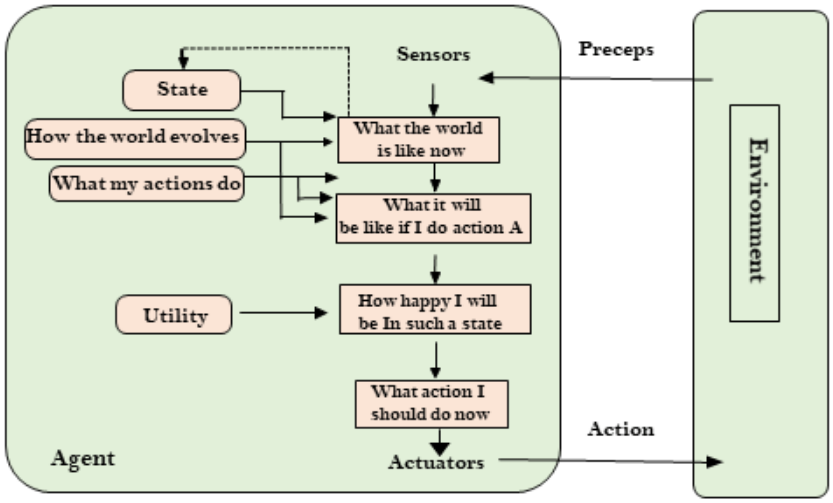
3. Goal-based agents

The knowledge of the current state environment is not always sufficient to decide for an agent to what to do. The agent needs to know its goal which describes desirable situations. Goal-based agents expand the capabilities of the model-based agent by having the "goal" information. They choose an action, so that they can achieve the goal. These agents may have to consider a long sequence of possible actions before deciding whether the goal is achieved or not. Such considerations of different scenario are called searching and planning, which makes an agent proactive.



4. Utility-based agents

These agents are similar to the goal-based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state. Utility-based agent act based not only goals but also the best way to achieve the goal. The Utility-based agent is useful when there are multiple possible alternatives, and an agent has to choose in order to perform the best action. The utility function maps each state to a real number to check how efficiently each action achieves the goals.



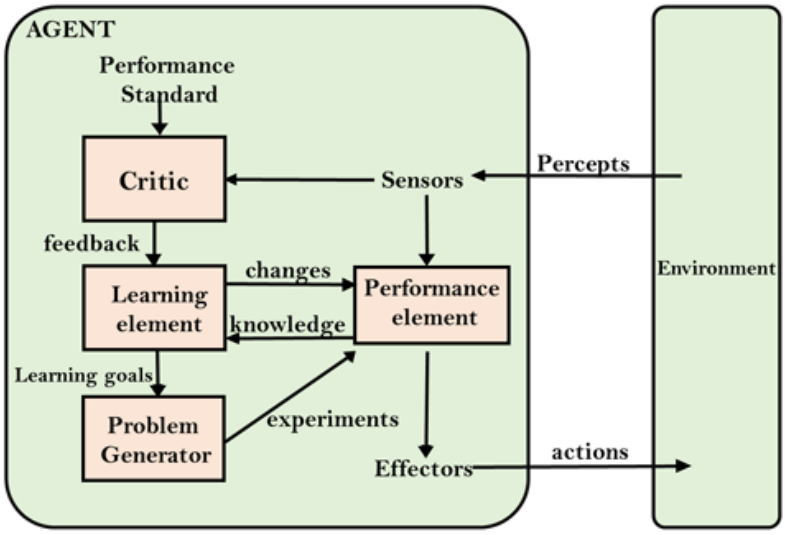
5. Learning Agents

A learning agent in AI is the type of agent which can learn from its past experiences, or it has learning capabilities. It starts to act with basic knowledge and then able to act and adapt automatically through learning.

A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from environment
2. **Critic:** Learning element takes feedback from critic which describes that how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action
4. **Problem generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.

Hence, learning agents are able to learn, analyse performance, and look for new ways to improve the performance.



2.5 AGENT ARCHITECTURE

Agent architecture in computer science is a blueprint for software agents and intelligent control systems, depicting the arrangement of components. The architectures implemented by intelligent agents are referred to as cognitive architectures.[1] The term agent is a conceptual idea, but not defined precisely. It consists of facts, set of goals and sometimes a plan library.

Following are the types of agent architecture :

- Reactive architectures
- Deliberative reasoning architectures
- Layered/hybrid architectures
- Cognitive architectures

Reactive architecture

Reactive Architecture is nothing more than the combination of reactive programming and software architectures. Also known as reactive systems,

the goal is to make the system responsive, resilient, elastic, and message driven. A Reactive system is an architectural style that allows multiple individual applications to coalesce as a single unit, reacting to its surroundings while aware of each other, and enable automatic scale up and down, load balancing, responsiveness under failure, and more.

Reactive Architecture can elastically scale in the face of varying incoming traffic. Scaling usually serves one of two purposes: either we need to scale out (by adding more machines) and up (by adding beefier machines), or we need to scale down, reducing the number of resources occupied by our application. An interesting scaling pattern popularized by the likes of Netflix is predictive scaling, in which we know when spikes are going to hit so we can proactively provision servers for that period, and once traffic starts going down again, decrease the cluster size incrementally.

Reactive Architecture Benefits

- Be responsive to interactions with its users
- Handle failure and remain available during outages
- Strive under varying load conditions
- Be able to send, receive, and route messages in varying network conditions.

Deliberative reasoning architectures

We define a deliberative agent or agent architecture to be one that contains an explicitly represented, symbolic model of the world, and in which decisions (for example about what actions to perform) are made via logical (or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation. The idea of deliberative agents based on purely logical reasoning is highly seductive: to get an agent to realise some theory of agency one might naively suppose that it is enough to simply give it logical representation of this theory and `get it to *do a bit of theorem proving*' [Shardlow, 1990]. If one aims to build an agent in this way, then there are at least two important problems to be solved:

1. The transduction problem: that of translating the real world into an accurate, adequate symbolic description, in time for that description to be useful.
2. The representation/reasoning problem: that of how to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful.

Layered/Hybrid architecture

A hybrid architecture is one that combines or adapts one of the previously discussed systems. For example, system manufacturers will connect multiple SMP machines using a high-speed interconnect to create a hybrid system with a communications model involving two different levels of service. *On-node* communication (where a node is a single SMP machine)

is significantly faster than *cross-node* communication. Another configuration might connect small MPP (i.e., 16-node) machines, each of which shares some memory with other small MPP machines within a single box. The use of a hybrid architecture may be very dependent on the specific application, because some systems may be better suited to the concurrency specifics associated with each application.

Cognitive architecture

A cognitive architecture is a hypothesis about the fixed structures that provide a mind, whether in natural or artificial systems, and how they work together – in conjunction with knowledge and skills embodied within the architecture – to yield intelligent behaviour in a diversity of complex environments. A grand unified architecture integrates across (nominally symbolic) higher-level thought processes plus any other (nominally sub-symbolic) aspects critical for successful behaviour in human-like environments, such as perception, motor control, and emotions. A generically cognitive architecture spans both the creation of artificial intelligence and the modelling of natural intelligence, at a suitable level of abstraction. A functionally elegant architecture yields a broad range of capabilities from the interactions among a small general set of mechanisms – essentially what can be thought of as a set of *cognitive Newton's laws*. A sufficiently efficient architecture executes quickly enough for its anticipated applications; for example, taking no more than 50m sec per cognitive cycle for real-time virtual humans.

Multiple Choice Questions

1) Artificial Intelligence is about_____.

   a. Playing a game on Computer
   b. Making a machine Intelligent
   c. Programming on Machine with your Own Intelligence
   d. Putting your intelligence in Machine

2) Who is known as the -Father of AI"?

   a. Fisher Ada
   b. Alan Turing
   c. John McCarthy
   d. Allen Newell

3) Which of the given language is not commonly used for AI?

   a. LISP
   b. PROLOG
   c. Python
   d. Perl

4) An AI agent perceives and acts upon the environment using___.

a. Sensors
b. Perceiver
c. Actuators
d. Both a and c

5) Ways to achieve AI in real-life are_____.

a. Machine Learning
b. Deep Learning
c. Both a & b
d. None of the above

6) The best AI agent is one which_____

a. Needs user inputs for solving any problem
b. Can solve a problem on its own without any human intervention
c. Need a similar exemplary problem in its knowledge base
d. All of the above

7) If a robot is able to change its own trajectory as per the external conditions, then the robot is considered as the__

a. Mobile
b. Non-Servo
c. Open Loop
d. Intelligent

8) Which rule is applied for the Simple reflex agent?

a. Simple-action rule
b. Simple &Condition-action rule
c. Condition-action rule
d. None of the above

9) Which agent deals with the happy and unhappy state?

a. Utility-based agent
b. Model-based agent
c. Goal-based Agent
d. Learning Agent

10) The exploration problem is where_____.

a. Agent contains the knowledge of State and actions.
b. Agent does not contain the knowledge of State and actions.
c. Only actions are known to the agent.
d. None of the above

❄❄❄❄❄❄❄

3

PROBLEM SOLVING - I

Unit Structure

3.0 Introduction
3.1 Problems,
3.2 Problem spaces and search: Define the problem as a state space search

3.0 INTRODUCTION

With reference to Artificial Intelligence, Problem is defined as a statement who clearly describes the problem in real world as well as beginning state of a problem that's need to be solved.

This statement indicates properties of problem like as

- The task to be solved
- The current performance of existing systems, and
- Experience with the current system

In other word, one can state Problem as, it is a query, needs to get solve. For solving a query, Problem must be clearly defined, means clear understanding of begin state, goal or final state as well as valid states and how it changes its states from beginning to final states as well it undergoes many stages which during the process of solving the query.

3.1 Problems :

As defined in section 3.0. Every Problems needs to be having its start stat, end state as well as intermediate state which it may acquire during the process of solving any real-world problem. The solution to any problem is a fixed sequence of actions.

The problem-solving approach has been applied to a vast array of task environments. We list some of the best known here, distinguishing between *toy* and *real-world* problems. A **toy problem** is intended to illustrate or exercise various problem-solving methods. It can be given a concise, exact description and hence is usable by different researchers to compare the performance of algorithms.

A **real-world problem** is one whose solutions people actually care about. Such problems tend not to have a single agreed-upon description, but we can give the general flavour of their formulations.

The process of looking for a sequence of actions that reaches the goal is called **search**.

A search algorithm takes a problem as input and returns a **solution** in the form of an action sequence. Once a solution is found, the actions it recommends can be carried out. This is called the **execution** phase.

Thus, we have a simple "formulate, search, execute" design for the agent, as shown in algorithm 3.1. After formulating a goal and a problem to solve, the agent calls a search procedure to solve it.

It then uses the solution to guide its actions, doing whatever the solution recommends as the next thing to do—typically, the first action of the sequence—and then removing that step from the sequence. Once the solution has been executed, the agent will formulate a new goal.

Well-defined problems and solutions:

A problem can be defined formally by five components:

1. The **initial state** that the agent starts in. (**agent** could be anything that makes decisions)

2. A description of the possible **actions** available to the agent. Given a particular state "s", ACTIONS(s) return the set of actions that can be executed in s. We say that each of these actions are applicable in s.

   A description of what each action does; the formal name for this is the TRANSITION MODEL, specified by a function RESULT(s, a) that returns the state that results from SUCCESSOR doing action "a" in state "s". We also use the term successor to refer to any state reachable from a given state by a single action.

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

4. A **path** in the state space is a sequence of states connected by a sequence of actions.

   The **goal test**, which determines whether a given state is a goal state. Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state is one of them.

   A **path cost** function that assigns a numeric cost to each path. The problem-solving agent chooses a cost function that reflects its own performance measure.

   The preceding elements define a problem and can be gathered into a single data structure that is given as input to a problem-solving algorithm.

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

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept ) returns an action persistent: seq, an action sequence, initially empty state, some description of the current world state goal, a goal, initially null problem, a problem formulation

        state←UPDATE-STATE(state, percept )
        if seq is empty then
                goal ←FORMULATE-GOAL(state)
                problem ←FORMULATE-PROBLEM(state, goal )
                seq ←SEARCH(problem)
        if seq = failure then return a null action
                action ←FIRST(seq)
                seq ←REST(seq)
        return action
```

Algorithm 3.1.

Above algorithm is a demonstration of a simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

What is Abstraction:

With reference to solve problem of travelling from source to destination, as well as many state of object in between then, for such a case "Once we start moving from one source to reach to destination, All the considerations are left out of our source state descriptions because they are irrelevant to the problem of finding a route to Destination. The process of removing detail from a representation is called **abstraction"**.

Following are Some examples shows problems initialisation state, goal state and mid-state arises during processing of problem to achieve target goal state.

Touring problems:

Touring problems are closely related to route-finding problems, but with an important difference. As with route finding, the actions correspond to trips between adjacent cities. The state space, however, may quite different, and the goal test would check whether the agent is achieved a target-states or not along with all mid cities must have been visited.

The **traveling salesperson problem** (TSP):

The **traveling salesperson problem** (TSP) is a touring problem in which each city must be visited exactly once. The aim is to find the *shortest* tour. The problem is known to be NP-hard, but an enormous amount of effort has been expended to improve the capabilities of TSP algorithms. In addition to planning trips for traveling salespersons, these algorithms have

been used for tasks such as planning movements of automatic circuit-board drills and of stocking machines on shop floors.

Very Large Scale Integrated (VLSI) Circuits (VLSI) Layout Problem:

A **VLSI layout** problem requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield.

The layout problem comes after the logical design phase and is usually split into two parts: **cell layout** and **channel routing**. In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognized function.

Each cell has a fixed footprint (size and shape) and requires a certain number of connections to each of the other cells. The aim is to place the cells on the chip so that they do not overlap and so that there is room for the connecting wires to be placed between the cells.

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

Robot navigation:

Robot navigation is a generalization of the route-finding problem described earlier. Rather than following a discrete set of routes, a robot can move in a continuous space with (in principle) an infinite set of possible actions and states.

For a circular robot moving on a flat surface, the space is essentially two-dimensional. When the robot has arms and legs or wheels that must also be controlled, the search space becomes many-dimensional. Advanced techniques are required just to make the search space finite.

Automatic assembly sequencing:

Automatic assembly sequencing of complex objects by a robot was first demonstrated by FREDDY (Michie, 1972). Progress since then has been slow but sure, to the point where the assembly of intricate objects such as electric motors is economically feasible.

In assembly problems, the aim is to find an order in which to assemble the parts of some object. If the wrong order is chosen, there will be no way to add some part later in the sequence without undoing some of the work already done. Checking a step in the sequence for feasibility is a difficult geometrical search problem closely related to robot navigation.

Thus, the generation of legal actions is the expensive part of assembly sequencing. Any practical algorithm must avoid exploring all but a tiny fraction of the state space. Another important assembly problem is **protein design**, in which the goal is to find a sequence of amino acids that will fold into a three-dimensional protein with the right properties to cure some disease.

TOY PROBLEM:

The problem-solving approach has been applied to a vast array of task environments. We list some of the best known here, distinguishing between toy and real-world problems. A toy problem is intended to illustrate or exercise various problem-solving methods. It can be given a concise, exact description and hence is usable by different researchers to compare the performance of algorithms.

A real-world problem is one whose solutions people actually care about. Such problems tend not to have a single agreed-upon description, but we can give the general flavour of their formulations.

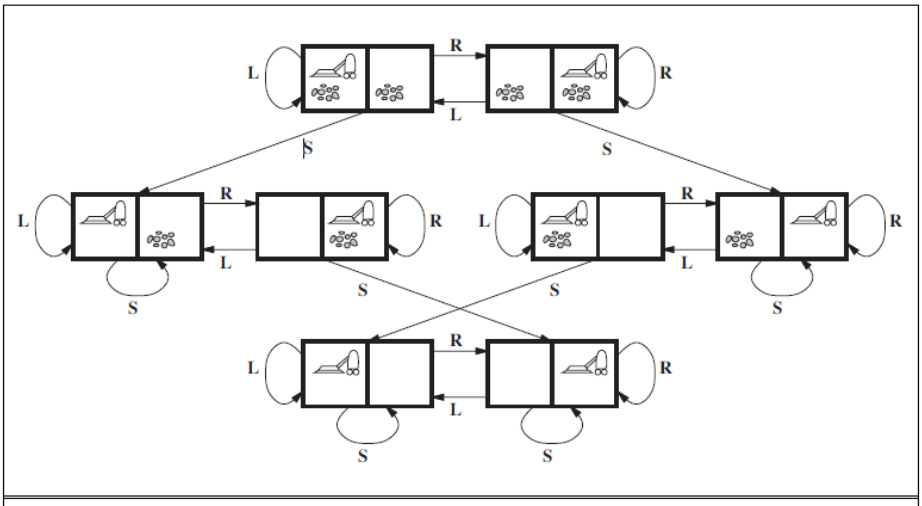


Figure 2.3.0 The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

The first example we examine is the **vacuum world,** this can be formulated as a problem as follows:

- **States**: The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt.

Thus, there are $2 * 2^2 = 8$ possible world states. A larger environment with n locations has $n * 2n$ states.

- **Initial state**: Any state can be designated as the initial state.
- **Actions**: In this simple environment, each state has just three actions: *Left*, *Right*, and *Suck*. Larger environments might also include *Up* and *Down*.
- **Transition model**: The actions have their expected effects, except that moving *Left* in the leftmost square, moving *Right* in the rightmost square, and *Suck*ing in a clean square have no effect. The complete state space is shown in Figure 3.3.0
- **Goal test**: This checks whether all the squares are clean.
- **Path cost**: Each step costs 1, so the path cost is the number of steps in the path.

Conclusion: "Compared with the real world, this toy problem has discrete locations, discrete dirt, reliable cleaning, and it never gets any dirtier."

3.2 PROBLEM SPACES AND SEARCH: DEFINE THE PROBLEM AS A STATE SPACE SEARCH

As per definition of Problem (Ref. 3.1) with reference to an Artificial Intelligence, definition of start state (Begin State), goal state (End State), other valid states and transitions must be clearly stated before problem solving process will be carried out.

Whenever state of any problem needs to be represented, a state space representation allows for the formal definition of a problem which makes the movement from start state to the end state very acceptable and understanding way. So, we can say that various problems like planning, learning, algorithm creation etc. are all mandatory a search problem only.

State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with goal of finding a goal state with a desired property.

Following example demonstrate problem and state space search :

An Eight-Tile Puzzle Problem:

The 08-tile puzzle consist of a 3X3 (3 by 3) tiles arranged in matrix format frame board which hold 08 movable tiles numbered from one (01) to eight (08).

One square is left vacant, allowing the adjacent tiles to be shifted from all direction. The objective of the puzzle is to find a sequence of tile movements that leads from starting configuration to a goal configuration.

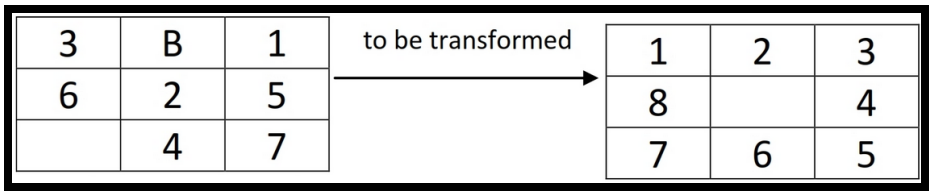


Fig. 3.1 Problem statement for 08 tile puzzle

The state of 08 tile puzzle are the different permutations of the tile within frame.

The formulation of this problem can be stated as follows:

States: The location of each 08 tiles and the blank in one of the nine squares.

Initial state : Any state can be designated as the initial state.

Goal : Many goal configurations are possible one such is shown in the figure

Rule of Legal moves or state : They generate legal states that result from trying the four actions as "Blank Tile Moves in any of the four directions as Left, Right, Up, Down"

Cost of Path : Every step cost unique 01, so the path cost is the number of steps in the path.

The tree diagram showing the search space is shown in figure:

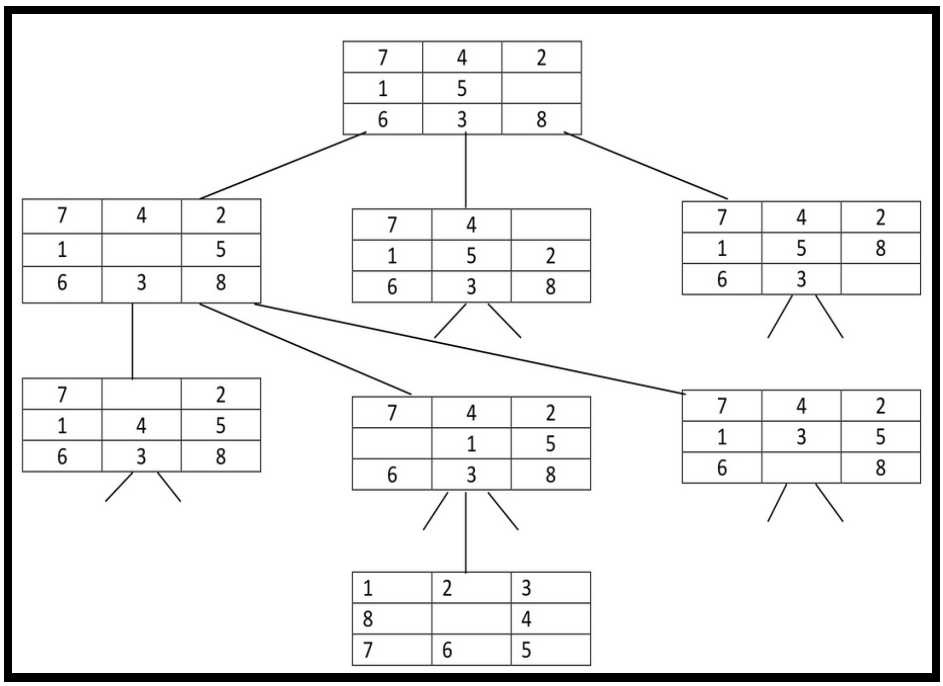


Fig. 3.2 Initial stat, Steps wise states and Goal state of Problem statement for 08 tile puzzle

What abstractions have we included here? The actions are abstracted to their beginning and

final states, ignoring the intermediate locations where the block is sliding. We have abstracted away actions such as shaking the board when pieces get stuck and ruled out extracting the pieces with a knife and putting them back again. We are left with a description of the rules of the puzzle, avoiding all the details of physical manipulations.

The 8-puzzle belongs to the family of **sliding-block puzzles**, which are often used as test problems for new search algorithms in AI. This family is known to be NP-complete, so one does not expect to find methods significantly better in the worst case than the search algorithms.

The 8-puzzle has 9!/2=181, 440 reachable states and is easily solved.

The 15-puzzle (on a 4×4 board) has around 1.3 trillion states, and random instances can be solved optimally in a few milliseconds by the best search algorithms.

The 24-puzzle (on a 5 × 5 board) has around 1025 states, and random instances take several hours to solve optimally.

The 8-queens problem:

The goal of the 8-queens problem is to place eight queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.)

Figure 3.3 shows an attempted solution that fails: the queen in the rightmost column is attacked by the queen at the top left.

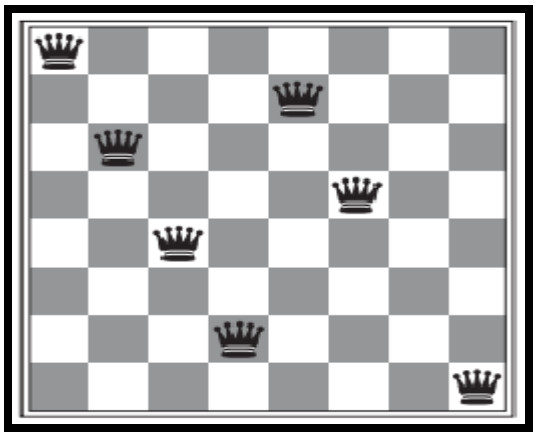


Fig. 3.3 Solution to the 08-queens problem

- Although efficient special-purpose algorithms exist for this problem and for the whole n-queens family, it remains a useful test problem for search algorithms.
- There are two main kinds of formulation. An incremental formulation involves operators that augment the state description, starting with an empty state; for the 8-queens problem, this means that each action adds a queen to the state.
- A complete-state formulation starts with all 8 queens on the board and moves them around. In either case, the path cost is of no interest because only the final state counts. The first incremental formulation one might try is the following:

- States: Any arrangement of 0 to 8 queens on the board is a state.
- Initial state: No queens on the board.
- Actions: Add a queen to any empty square.
- Transition model: Returns the board with a queen added to the specified square.
- Goal test: 8 queens are on the board, none attacked. In this formulation, we have $64 \cdot 63 \cdots 57 \approx 1.8 \times 1014$ possible sequences to investigate. A better formulation would prohibit placing a queen in any square that is already attacked:
- States: All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.
- Actions: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.

Real-world problems:

Following are some of the examples of Real-World problems:

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

Searching for solutions:

A solution is an action sequence, so search algorithms work by considering various possible action sequences.

The possible action sequences starting at the initial state form a search tree with the initial state NODE at the root; the branches are actions and the nodes correspond to states in the state space of the problem. Refer algorithm below for general search algorithm.

function TREE-SEARCH(problem) **returns** a solution, or failure initialize the frontier using the initial state of problem **loop do if** the frontier is empty **then return** failure choose a leaf node and remove it from the frontier **if** the node contains a goal state **then return** the corresponding solutionexpand the chosen node, adding the resulting nodes to the frontier

Algorithm 3.2

function GRAPH-SEARCH(problem) **returns** a solution, or failure initialize the frontier using the initial state of problem ***initialize the explored set to be empty*** **loop do if** the frontier is empty **then return** failure choose a leaf node and remove it from the frontier **if** the node contains a goal state **then return** the corresponding solution ***add the node to the explored set*** expand the chosen node, adding the resulting nodes to the frontier ***only if not in the frontier or explored set***

Algorithm 3.3

Above algorithms (3.2 & 3.3) shows an informal description of the general tree-search and graph-search algorithms.

The parts of GRAPH-SEARCH marked in bold italic are the additions needed to handle repeated states.

Infrastructure for search algorithms:

Search algorithms require a data structure to keep track of the search tree that is being constructed. Refer Figure 3.4.

For each node n of the tree, we have a structure that contains four components:

- n.STATE: the state in the state space to which the node corresponds;
- n.PARENT: the node in the search tree that generated this node;
- n.ACTION: the action that was applied to the parent to generate the node;
- n.PATH-COST: the cost, traditionally denoted by g(n), of the path from the initial state to the node, as indicated by the parent pointers.

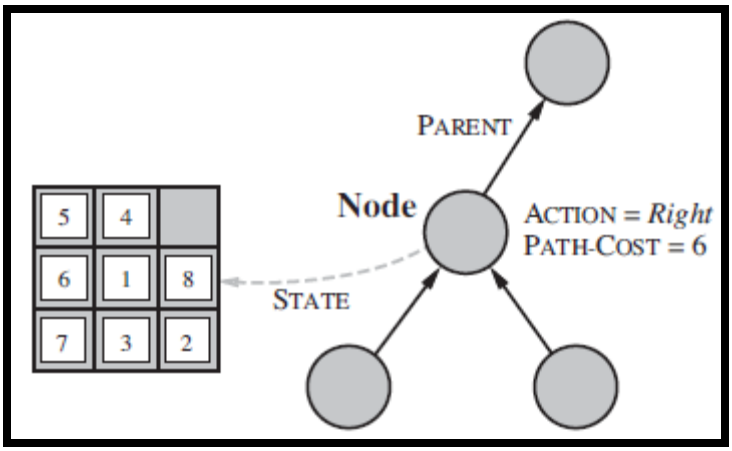


Fig. 3.4 : Nodes are the data structures from which the search tree is constructed.

Each has a parent, a state, and various bookkeeping fields. Arrows point from child to parent.

Given the components for a parent node, it is easy to see how to compute the necessary components for a child node. The function CHILD-NODE takes a parent node and an action and returns the resulting child node:

function CHILD-NODE(problem, parent , action) **returns** a node
return a node with
STATE = problem.RESULT(parent.STATE, action),
PARENT = parent, ACTION = action,
PATH-COST = parent.PATH-COST + problem.STEP-COST(parent.STATE, action)

The node data structure is depicted in Figure 3.4. Notice how the PARENT pointers string the nodes together into a tree structure. These pointers also allow the solution path to be extracted when a goal node is found; we use the SOLUTION function to return the sequence of actions obtained by following parent pointers back to the root.

Up to now, we have not been very careful to distinguish between nodes and states, but in writing detailed algorithms, it's important to make that distinction. A node is a bookkeeping data structure used to represent the search tree.

A state corresponds to a configuration of the world. Thus, nodes are on particular paths, as defined by PARENT pointers, whereas states are not. Furthermore, two different nodes can contain the same world state if that state is generated via two different search paths.

Now that we have nodes, we need somewhere to put them. The frontier needs to be stored in such a way that the search algorithm can easily choose the next node to expand according to its preferred strategy. The appropriate data structure for this is a **queue**. The operations on a queue are as follows:

- EMPTY(queue) returns true only if there are no more elements in the queue.
- POP(queue) removes the first element of the queue and returns it.
- INSERT(element, queue) inserts an element and returns the resulting queue.

Queues are characterized by the *order* in which they store the inserted nodes. Three common variants are the first-in, first-out or **FIFO queue**, which pops the *oldest* element of the queue; the last-in, first-out or **LIFO queue** (also known as a **stack**), which pops the *newest* element of the queue; and the **priority queue**, which pops the element of the queue with the highest priority according to some ordering function.

The explored set can be implemented with a hash table to allow efficient checking for repeated states. With a good implementation, insertion and lookup can be done in roughly constant time no matter how many states are stored. One must take care to implement the hash table with the right notion of equality between states.

Measuring problem-solving performance:

Before we get into the design of specific search algorithms, we need to consider the criteria that might be used to choose among them. We can evaluate an algorithm's performance in four ways:

- Completeness: Is the algorithm guaranteed to find a solution when there is one?
- Optimality: Does the strategy find the optimal solution?
- Time complexity: How long does it take to find a solution?
- Space complexity: How much memory is needed to perform the search?

Time and space complexity are always considered with respect to some measure of the problem difficulty. In theoretical computer science, the typical measure is the size of the state space graph, |V | + |E|, where V is the set of vertices (nodes) of the graph and E is the set of edges (links). This is appropriate when the graph is an explicit data structure that is input to the search program.

In AI, the graph is often represented implicitly by the initial state, actions, and transition model and is frequently infinite.

For these reasons, complexity is expressed in terms of three quantities:

1. b, the branching factor or maximum number of successors of any node;
2. d, the depth of the shallowest goal node (i.e., the number of steps along the path from the root); and
3. m, the maximum length of any path in the state space.

Time is often measured in terms of the number of nodes generated during the search, and space in terms of the maximum number of nodes stored in memory. For the most part, we describe time and space complexity for search on a tree; for a graph, the answer depends on how "redundant" the paths in the state space are in kilo meters. Thus, to compute the total cost, we have to add milliseconds and kilo meters.

There is no "official exchange rate" between the two, but it might be reasonable in this case to convert kilo meters into milliseconds by using an estimate of the car's average speed (because time is what the agent cares about). This enables the agent to find an optimal trade off point at which further computation to find a shorter path becomes counterproductive.

To assess the effectiveness of a search algorithm, we can consider just the **search cost**- which typically depends on the time complexity but can also include a term for memory usage or we can use the **total cost**, which combines the search cost and the path cost of the solution found. For the problem of finding a route from Source to Destination, the search cost is the amount of time taken by the search and the solution cost is the total length of the path becomes counterproductive.

Bibliography:

1. Artificial Intelligence, Modern Approach, By Stuart Russel & Peter Norving
2. A First Course in Artificial Intelligence, By Deepak Khemani, TMH
3. Artificial Intelligence: A Rational Approach, By Rahul Deva, Shroff publishers

Web Reference:

1. http://www.simplynotes.in/state-space-search/

Exercise:

Answer the following:

1. Explain Goal and problem formulation.
2. Explain Search, Solution and Execution with reference to AI.
3. Explain following terms:
   i) State space of problem
   ii) Path in state space
   iii) Goal test
   iv) Path cost
   v) Solution to problem
4. Explain 8-puzzle game problem.
5. Explain real world problem.
6. Explain Infrastructure for search algorithms.
7. Explain how algorithm's performance can be evaluated?

❄❄❄❄❄❄❄

4

PROBLEM SOLVING - II

Unit Structure

4.1 PRODUCTION SYSTEMS (P. S.)

With reference to Artificial Intelligence, a production system is a special type of software program that provides artificial intelligence based on a set of rules.

A production system is also referred as a production rule system. It is a kind of rational or cognitive architecture that is used to implement search algorithms and imitate human problem-solving skills.

This problem-solving knowledge is encoded in the system in the form of little quanta which is known as productions. It consists of two components: rule and action.

The process starts with the rules, which identifies the condition, and the action part has the knowledge of how to deal with the condition. In general, the production system in AI contains a set of rules which are defined by the left side and right side of the system. The left side contains a set of things to watch for (condition), and the right side contains the things to do (action).

Following are the important elements of P.S.:

1. Global Database
2. A set of Production Rules
3. Control System

Detailed description of elements of P.S. are as follows:

1. **Global Database:** The primary database which contains all the information necessary to successfully complete a task. It is further broken down into two parts: temporary and permanent. The temporary part contains information relevant to the current situation only whereas the permanent part contains information about the fixed actions.

2. **A set of Production Rules:** A set of rules that operates on the global database. Each rule consists of a precondition and postcondition that the global database either meets or not. For example, if a condition is met by the global database, then the production rule is applied successfully.
3. **Control System:** A control system that acts as the decision-maker, decides which production rule should be applied. The Control system stops computation or processing when a termination condition is met on the database.

The Features of a P. S.?

A P.S. has the following features:

1. **Simplicity:** Due to the use of the IF-THEN structure, each sentence is unique in the production system. This uniqueness makes the knowledge representation simple to enhance the readability of the production rules.

2. **Modularity:** The knowledge available is coded in discrete pieces by the production system, which makes it easy to add, modify, or delete the information without any side effects.

3. **Modifiability:** This feature allows for the modification of the production rules. The rules are first defined in the skeletal form and then modified to suit an application.

4. **Knowledge-intensive:** As the name suggests, the system only stores knowledge. All the rules are written in the English language. This type of representation solves the semantics problem.

Classification of P.S.: Refer Fig. 4.1

- **Monotonic Production System:**

  In this type of P.S., the use of one rule never prevents the involvement of another rule when both the rules are selected at the same time. Hence, it enables the system to apply rules simultaneously.

- **Partially Commutative Production System:**

  In this type of P.S., if a set of rules is used to change state A to state B then any allowable combination of these rules will also produce the same results (convert state A to state B).

- **Non-Monotonic Production System:**

  In this type of P.S., increases the problem-solving efficiency of the machine by not keeping a record of the changes made in the previous search process. These types of production systems are useful from an implementation point of view as they do not

backtrack to the previous state when it is found that an incorrect path was followed.

- **Commutative Production System:**

  In this type of P.S., is used when the order of operation is not important, and the changes are reversible.

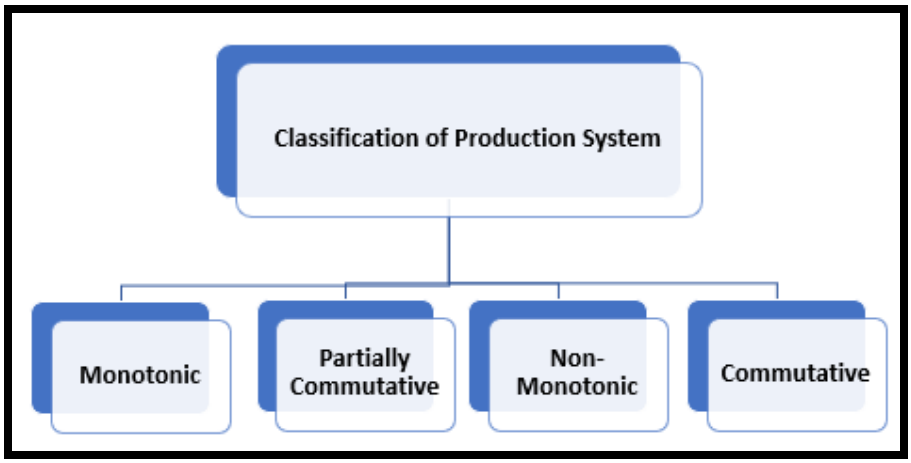


Fig. 4.1 Classification of P.S

Contribution of P.S. in A.I.:

- Offers modularity as all the rules can be added, deleted, or modified individually.
- Separate control system and knowledge base.
- An excellent and feasible model that imitates human problem-solving skills.
- Beneficial in real-time applications and environment.
- Offers language independence

4.2 PROBLEM CHARACTERISTICS

As artificial intelligence (AI) is mainly concerned to the **search process**, it is important to have some methodology to choose the best possible solution.

To choose an appropriate method for a particular problem first we need to categorize the problem based on the following characteristics.

1. Is the problem decomposable into small sub-problems which are easy to solve?

- Answer to this question is that, to solve any problems based on Search, A.I. , first breaking down the bigger problem into smaller problems to be solved independently?
- The decomposable problem can be solved easily.

Example: In this case, the problem is divided into smaller problems. The smaller problems are solved independently. Finally, the result is merged to get the final result.

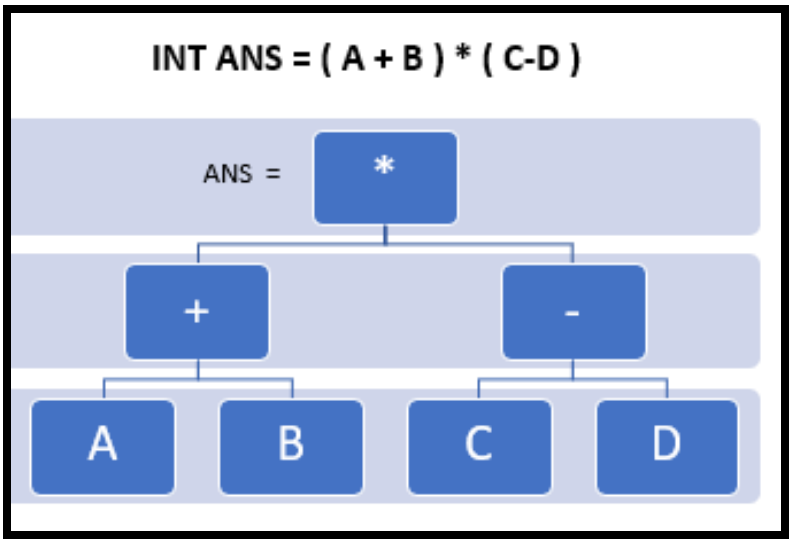


Fig. 4.2 problem decomposable into small sub-problems by A.I.

As shown in above Figure: left side and right side of the equation will be solved independently of each other than results of both sides will be clubbed together for multiplication and result of multiplication will get stored in ANS.

2. Can solution steps be ignored or undone?

In the Theorem Proving problem, a lemma that has been proved can be ignored for the next steps. Such problems are called **Ignorable** problems. Eg. In the 8-Puzzle, Moves can be undone and backtracked.

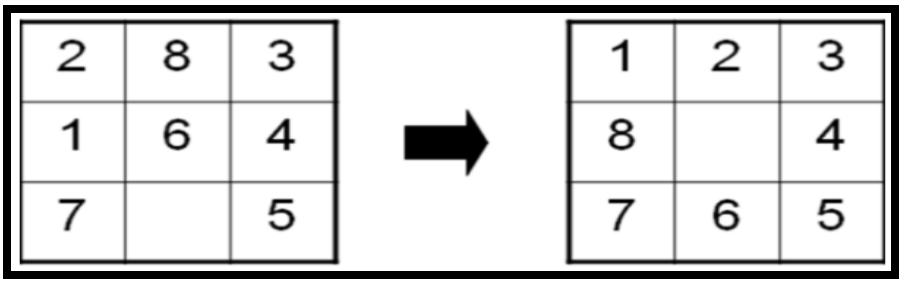


Fig. 4.3 Eight (08)-Puzzle, Moves can be undone and backtracked

Such problems are called **Recoverable** problems. **Ignorable** problems can be solved using a simple control structure that never backtracks. **Recoverable** problems can be solved using backtracking. **Irrecoverable** problems can be solved by recoverable style methods via planning.

3. Is the universe of the problem being predictable?

In Playing Bridge, we cannot know exactly where all the cards are or what the other players will do on their turns.

Uncertain outcome!

For **certain-outcome problems**, planning can be used to generate a sequence of operators that is guaranteed to lead to a solution.

For **uncertain-outcome problems**, a sequence of generated operators can only have a good probability of leading to a solution. Plan revision is made as the plan is carried out and the necessary feedback is provided.

4. Is a good solution to the problem is absolute or relative?

The Travelling Salesman Problem, we have to try all paths to find the shortest one. Any path problem can be solved using heuristics that suggest good paths to explore. For best-path problems, a much more exhaustive search will be performed.

5. Is the solution to the problem a state or a path:

The Water Jug Problem, the path that leads to the goal must be reported. A path-solution problem can be reformulated as a state-solution problem by describing a state as a partial path to a solution. The question is whether that is natural or not.

6. What is the role of knowledge in solving a problem using artificial intelligence?

Playing Chess

Consider again the problem of playing chess. Suppose you had unlimited computing power available. How much knowledge would be required by a perfect program? The answer to this question is very little—just the rules for determining legal moves and some simple control mechanism that implements an appropriate search procedure.
Additional knowledge about such things as good strategy and tactics could of course help considerably to constrain the search and speed up the execution of the program. Knowledge is important only to constrain the search for a solution.

Reading Newspaper

Now consider the problem of scanning daily newspapers to decide which are supporting the Democrats and which are supporting the Republicans in some upcoming election. Again, assuming unlimited computing power, how much knowledge would be required by a computer trying to solve this problem? This time the answer is a great deal.

- It would have to know such things as: The names of the candidates in each party. The fact that if the major thing you want to see done is have taxes lowered, you are probably supporting the Republicans.

  The fact that if the major thing you want to see done is improved education for minority students, you are probably supporting the Democrats. The fact that if you are opposed to big government, you are probably supporting the Republicans. And so on …

  Note: Knowledge is required even to be able to recognize a solution.

7. Does the task of solving a problem require human interaction?

Sometimes it is useful to program computers to solve problems in ways that the majority of people would not be able to understand.

This is fine if the level of the interaction between the computer and its human users is problem-in solution-out.

But increasingly we are building programs that require intermediate interaction with people, both to provide additional input to the program and to provide additional reassurance to the user.

The **solitary problem**, in which there is no intermediate communication and no demand for an explanation of the reasoning process.

The **conversational problem,** in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

4.3 PRODUCTION SYSTEM CHARACTERISTIC

Main characteristics of the P.S. in AI that is

- simplicity,
- modifiability,
- modularity, and
- knowledge-intensive.

Simplicity

The production rule in AI is in the form of an 'IF-THEN' statement. Every rule in the production system has a unique structure. It helps represent knowledge and reasoning in the simplest way possible to solve real-world problems. Also, it helps improve the readability and understanding of the production rules.

Modularity

The modularity of a production rule helps in its incremental improvement as the production rule can be in discrete parts. The production rule is made from a collection of information and facts that may not have dependencies

unless there is a rule connecting them together. The addition or deletion of single information will not have a major effect on the output. Modularity helps enhance the performance of the production system by adjusting the parameters of the rules.

Modifiability

The feature of modifiability helps alter the rules as per requirements. Initially, the skeletal form of the production system is created. We then gather the requirements and make changes in the raw structure of the production system. This helps in the iterative improvement of the production system.

Knowledge-intensive

Production systems contain knowledge in the form of a human spoken language, i.e., English. It is not built using any programming languages. The knowledge is represented in plain English sentences. Production rules help make productive conclusions from these sentences.

Disadvantages of a Production System

Many remarkable disadvantages are there in a P.S. in A.I. as given below:

Opacity

Communication between the rule interpreter and the production rules creates difficulty for the understanding of the control system and its strategies. This condition arises due to the impact of the combined operation of the control program. There exist difficulties in understanding the hierarchy of operations.

Inefficiency

There are various rules that we employ for solving a problem. The rules can be effective in different ways. There are conditions where multiple rules get activated during execution. All the individual rules apply exhaustive searches in each cycle that reduces the efficiency of the production system.

Inability to Learn

A simple production system based on certain rules is not capable of learning through experience, unlike advanced AI systems. They are simply bound to specific rules for actions. We can understand the rules and break them.

Conflict Resolution

To satisfy a condition, various production rules are employed. The condition may arise when there is a triggering of more than one rule. In that condition, the control system has to determine the best possible rule from the set of conflicting rules. This may reduce the efficiency of the production system

4.4 ISSUES IN DESIGN OF SEARCH PROGRAM

Search problem consists of following:

- A Start State. The state from where the search begins.
- A State Space. Set of all possible states where you can be.
- A Goal Test. A function that looks at the current state returns whether or not it is the goal state.

While solving search problem, object undergoes above stats, Following issues are faced during design of search program:

Issues in the design of search programs

1. The direction in which to conduct search (forward versus backward reasoning). If the search proceeds from start state towards a goal state, it is a forward search or we can also search from the goal.

2. How to select applicable rules (Matching). Production systems typically spend most of their time looking for rules to apply. So, it is critical to have efficient procedures for matching rules against states.

3. How to represent each node of the search process (knowledge representation problem).

Above issues can be minimized by following Control strategies to decide which rule to apply next during the process of searching for a solution to a problem.

Good control strategy should:

1. It should cause motion
2. It should be Systematic

Control strategies are classified as:

1. Uninformed/blind search control strategy.
2. Informed/Direct Search Control Strategy.

Uninformed/blind search control strategy:

- Do not have additional information about states beyond problem definition.
- Total search space is looked for solution.
- Example: Breadth First Search (BFS), Depth First Search (DFS), Depth Limited Search (DLS).

Informed/Directed Search Control Strategy:

- Some information about problem space is used to compute preference among the various possibilities for exploration and expansion.

- Examples: Best First Search, Problem Decomposition, A*, Mean end Analysis.

4.5 SEARCH TECHNIQUES: DFS, BFS, HILL CLIMBING

Introduction to uninformed search / Blind search:

- The term means that the strategies have no additional information about states beyond that provided in the problem definition.
- All they can do is generate successors and distinguish a goal state from a non-goal state.
- All search strategies are distinguished by the order in which nodes are expanded.
- Strategies that know whether one non-goal state is "more promising" than another are called informed search or heuristic search strategies

Breadth-first search:

- Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.
- In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.
- Breadth-first search is an instance of the general graph-search algorithm as given below, in which the shallowest unexpanded node is chosen for expansion.

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
if problem.GOAL-TEST(node.STATE) then return SOLUTIONS(node)
frontier ← a FIFO queue with node as the only element
explored ← an empty set
loop do
if EMPTY?(frontier) then return failure
node ← POP(frontier)
/* chooses the shallwest node in frontier */
add node.STATE to explored

for each action in problem.ACTIONS(node.STATE)
do
child ← CHILD-NODE(problem, node, action)
if child.state is not in explored or frontier then
if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
frontier ← INSERT(child,frontier)
```

Algorithm: Breadth-first search on a graph

- This is achieved very simply by using a FIFO queue for the frontier.
- Thus, new nodes (which are always deeper than their parents) go to the back of the queue, and old nodes, which are shallower than the new nodes, get expanded first.

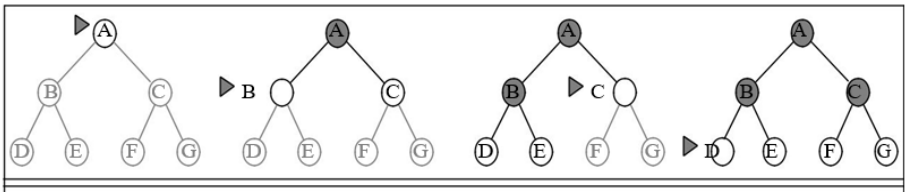


Figure 2.4.4 Breadth-first search on a simple binary tree. At each stage, the node to be expanded next is indicated by a marker.

Depth-First Search:

- It is implemented in recursion with LIFO stack data structure. It creates the same set of nodes as Breadth-First method, only in the different order.
- As the nodes on the single path are stored in each iteration from root to leaf node, the space requirement to store nodes is linear. With branching factor *b*and depth as *m*, the storage space is *bm.* (Refer Fig. 4.5)

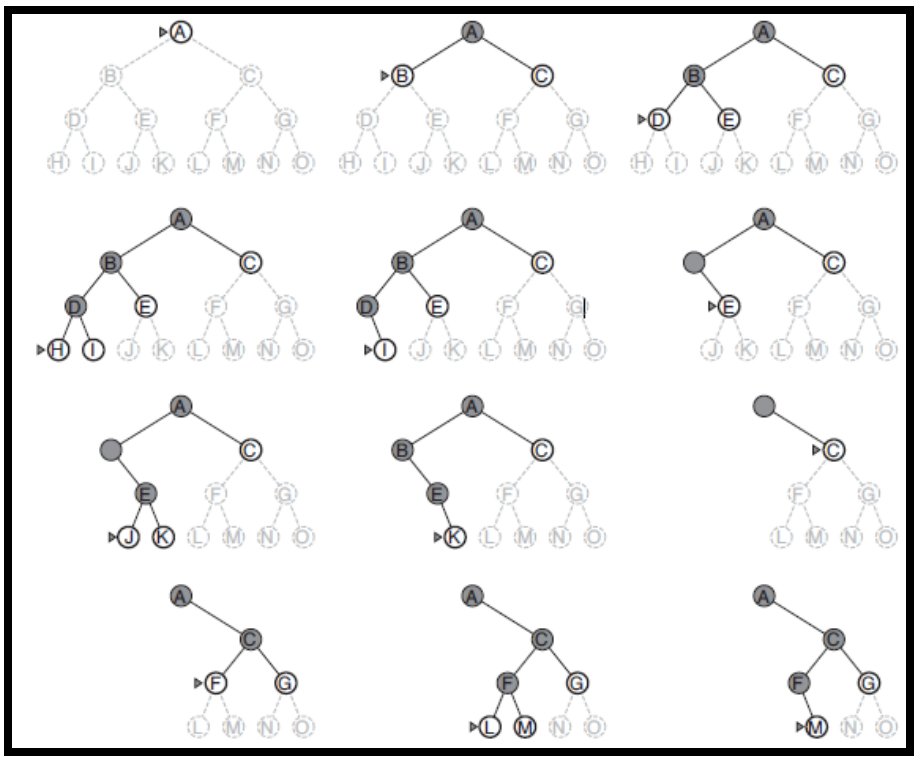


Fig. 4.5: Depth-first search on a binary tree. The unexplored region is shown in light- gray. Explored nodes with no descendants in the frontier are removed from memory. Nodes at depth 3 have no successors and M is the only goal node.

Disadvantage of DFS:

This algorithm may not terminate and go on infinitely on one path. The solution to this issue is to choose a cut-off depth. If the ideal cut-off is*d*, and if chosen cut-off is lesser than *d*, then this algorithm may fail. If chosen cut-off is more than *d*, then execution time increases.

- Its complexity depends on the number of paths. It cannot check duplicate nodes.

Hill Climbing:

To understand local search, we find it useful to consider the **state-space landscape** (as in Figure 4.6). A landscape has both "location" (defined by the state) and "elevation" (defined

by the value of the heuristic cost function or objective function). If elevation corresponds to cost, then the aim is to find the lowest valley—a **global minimum**; if elevation corresponds to an objective function, then the aim is to find the highest peak—a **global maximum**. (You can convert from one to the other just by inserting a minus sign.) Local search algorithms explore this landscape. A **complete** local search algorithm always finds a goal if one exists; an **optimal** algorithm always finds a global minimum/maximum

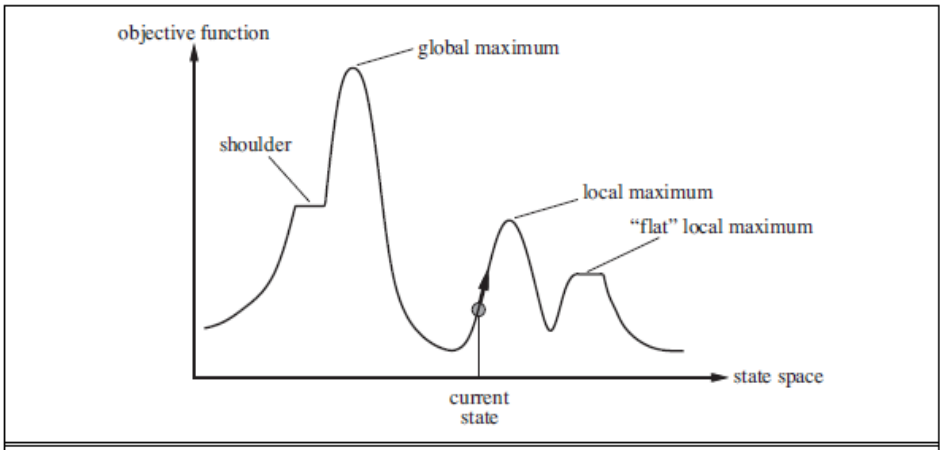


Figure: 4.6 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.

- In addition to finding goals, local search algorithms are useful for solving pure **optimization problems, in which the aim is to find the best state according to an objective function.**
- To understand local search, we find it useful to consider the **state-space landscape** A landscape has both "location" (defined by the state) and "elevation" (defined by the value of the heuristic cost function or objective function).

- If elevation corresponds to cost, then the aim is to find the lowest valley—a **global minimum; if elevation corresponds** to an objective function, then the aim is to find the highest peak—a **global maximum.**
- Local search algorithms explore this landscape. A **complete local search algorithm always finds a goal if one exists;** an **optimal algorithm always finds a global minimum/maximum.** Refer Figure 4.6 & Algorithm stated below for The hill-climbing search algorithm.

Hill-climbing search:

- The hill-climbing search algorithm (steepest-ascent version) is shown in Figure.
- It is STEEPEST ASCENT simply a loop that continually moves in the direction of increasing value—that is, uphill.
- It terminates when it reaches a "peak" where no neighbour has a higher value.
- Hill-climbing algorithms typically choose randomly among the set of best successors if there is more than one.

function HILL-CLIMBING (problem) **returns** a state that is a local maximum
current ←MAKE-NODE (problem. INITIAL-STATE)
loop do
neighbor ←a highest-valued successor of current
if neighbor.VALUE ≤ current. VALUE **then return** current. STATE
current←neighbor

Algorithm: The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

Greedy local search:

- Hill climbing is sometimes called greedy local search because it grabs a good neighbour state without thinking ahead about where to go next.
- Although greed is considered one of the seven deadly sins, it turns out that greedy algorithms often perform quite well.
- Hill climbing often makes rapid progress toward a solution because it is usually quite easy to improve a bad state.
- Unfortunately, hill climbing often gets stuck for the following reasons:

- **Local maxima: a local maximum is a peak that is higher than each of its neighbouring** states but lower than the global maximum.

- Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn upward toward the peak but will then be stuck with nowhere else to go. Refer Figure 4.7

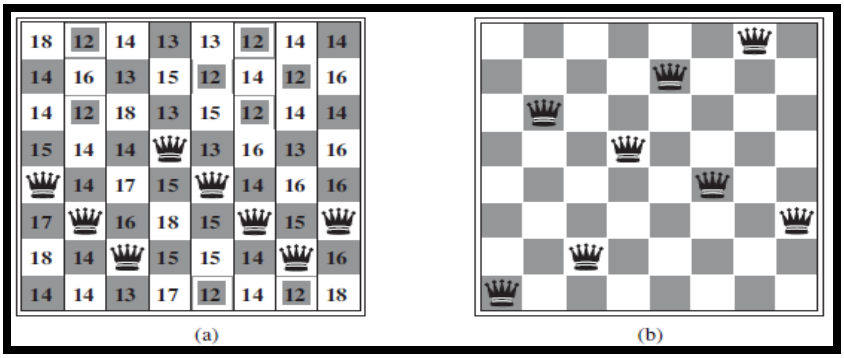


Fig. 4.7 (a) An 8-queens state with heuristic cost estimate h=17, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked. (b) A local minimum in the 8-queens state space; the state has h=1 but every successor has a higher cost.

- **Ridges: a ridge is shown in Figure 4.8 result in a sequence of local maxima** that is very difficult for greedy algorithms to navigate.
- **Plateaux**: a plateau is a flat area of the state-space landscape. It can be a flat local maximum, from which no uphill exit exists, or a **shoulder**, from which progress is possible. (Refer Figure 4.8) A hill-climbing search might get lost on the plateau.

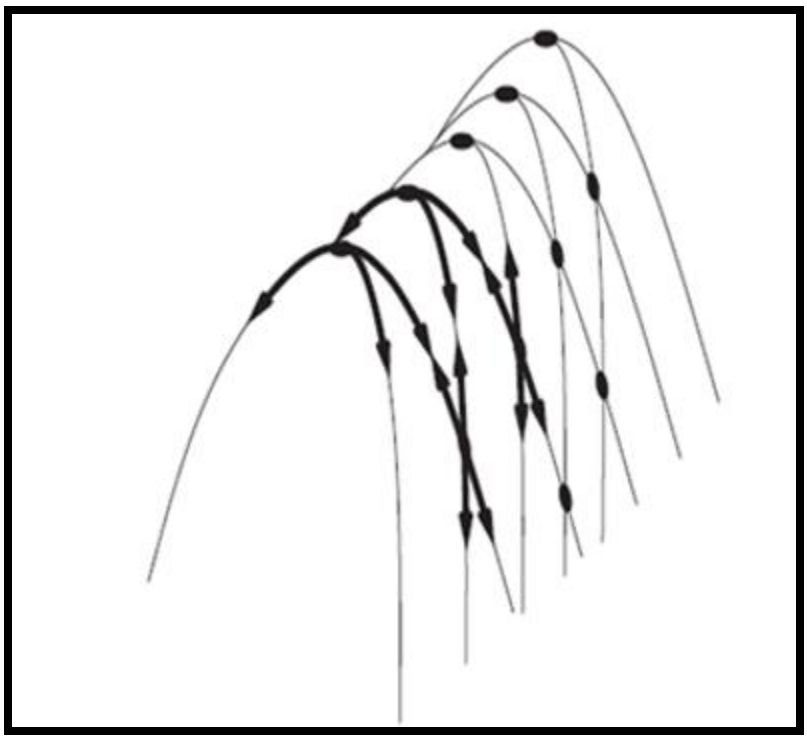


Fig. 4.8 Illustration of why ridges cause difficulties for hill climbing. The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.

Stochastic hill climbing:

- Many variants of hill climbing have been invented. **Stochastic hill climbing** chooses at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move.
- This usually converges more slowly than steepest ascent, but in some state landscapes, it finds better solutions.

Bibliography:

1. Artificial Intelligence, Modern Approach, By Stuart Russel & Peter Norving
2. A First Course in Artificial Intelligence, By Deepak Khemani, TMH
3. Artificial Intelligence: A Rational Approach, By Rahul Deva, Shroff publishers

Web Reference:

1. http://www.simplynotes.in/state-space-search/

Exercise:

Answer the following:

Q-1. With reference to A.I. explain What is meant by Production System. Explain the important elements of P.S.
Q-2. Explain features and classification of production system.
Q-3. Explain the characteristics based on which problem can be characterised.
Q-4. Explain characteristics and disadvantages of production system.
Q-5. Explain issues of design a search program.
Q-6. Write a short note of the following:
a. Depth first search
b. Breadth first search
c. Hill climbing algorithm

❄❄❄❄❄❄❄

5

KNOWLEDGE REPRESENTATION

Unit Structure

5.0 OBJECTIVE

This chapter will able you to understand the following concept

- What is knowledge? How it will be used in our day to day life
- Types of knowledge
- Methods/ approaches of knowledge representation
- Ways of knowledge representation

- Properties of good knowledge based system
- Issues of Knowledge representing
- AND OR search graph
- AI life cycle

5.1 INTRODUCTION

Knowledge level is based level of an agent, which consist of domain specific content.in this level agent has facts/information about surrounding environment in which they are working, it does not consider the actual implementation. Knowledge based agent is crucial to use in observable environments. Before choosing any action, knowledge based agents make use of the existing knowledge along with the current inputs from the environment in order to infer hidden aspects of the current state.

As we have learnt that knowledge base is a setoff representations of facts/information about surroundings. Every single representation in the set is called a sentence and sentence are expressed with the help of formal representation language. We can say that sentence is a statement which is a set of words that express some truth about real world. The knowledge level describes agent by saying what it knows.

5.2 KNOWLEDGE REPRESENTATION

Humans are having the ability to understand things, reasoning and knowledge interpreting. Human have knowledge about the things and they behave as per their knowledge. They act in real life based upon situation as per their knowledge. But in this technology era all the work is replaced by intelligent machine which can perform all the activities which can man done. This act of machine tells about knowledge representation and reasoning.

- In artificial intelligence Knowledge representation and reasoning are the most important feature with which we can identify how the machine can contribute in thinking and acting like human. This specify the intelligent bbehaviour of agents.
- It represents the real world activity by which computer can understand, think over the situation and will act like human with his prior knowledge. This will helpful to solve the problem in medical world like diagnosis the medical condition or it can also communicate with human in natural language.
- It also tells how human can talk with the machine and getting the knowledge based answer form it which shows the intelligence of the machine. Knowledge representation is come from stored data in machine which make the machine more intelligent and it will look like an intelligent human.

Types of knowledge present in AI systems:

- Object: facts that represents the real information about the world object. E.g., mango, banana grapes are fruits.
- Events: the actions in our world represent Events.
- Performance: the prescribe steps using the present knowledge in machine to do the things which also can define the behaviour of the agent.
- Meta-knowledge: as metadata is data about data as same the knowledge is also defining as the facts that we know in respect to any object.
- Facts: the truth about any data represent in real world is called Facets.
- Knowledge-Base: it can be also known as KB. Many sentence come together to make the knowledge. The sentence can be logical or natural language sentence. The main component of knowledge base agent is knowledge.

Knowledge: Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.

5.3 TYPES OF KNOWLEDGE

5.3.1. Declarative Knowledge:

- To know about something is called as Declarative knowledge.
- It includes concepts, facts, and objects.
- With declarative sentence it can express. It is also called descriptive knowledge
- It is simpler than procedural language.

5.3.2. Procedural Knowledge

- It is also known as imperative knowledge.
- The steps to do something is described as procedure and Procedural knowledge is define the process or procedure of any action.
- Any task can be done through the procedure.
- It includes rules, strategies, procedures, agendas, etc.
- The task dependent procedure is there for all type task hence the Procedural knowledge depends on the task

5.3.3. Meta-knowledge:

- As metadata is data about data as same the knowledge is also defining as the facts that we know in respect to any object.

5.3.4. Heuristic knowledge:

- Subject experts are heir for different field as they have the more knowledge and experience about the field. This type of knowledge is called as Heuristic knowledge.
- It can be gained with rules of thumb based on previous experiences, awareness of approaches, and which are good to work.

5.3.5. Structural knowledge:

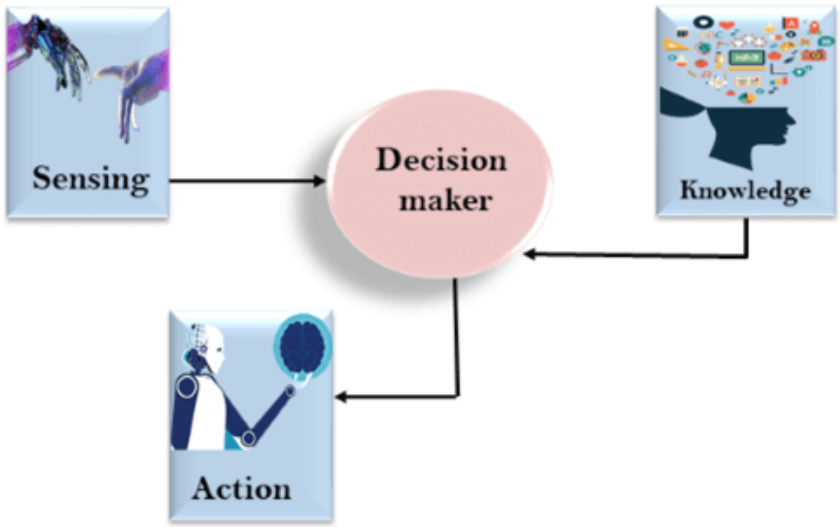
- to solve a problem, we need to have some steps of action, this structure is called as Structural knowledge.
- It describes relationships among various concepts such as type of, part of, and grouping of something.
- The relationship between concepts or objects can be described by structural knowledge.

5.4 HOW KNOWLEDGE & INTELLIGENCE CAN BE MAPPED EACH OTHER

Artificial intelligence can be creating by Knowledge of real-worlds objects which plays a vital role in demonstrating intelligent behaviour of AI agents. On some input agent should accurate act with applying knowledge or experience about that input.

Let's suppose if you act on a situation where the lady who is trying to cross the road by following signals, and she did not turn up t cross the road, one need to help her to cross the road likewise the intelligent agent also takes the decision from the knowledge that he have within himself and do the action as human.

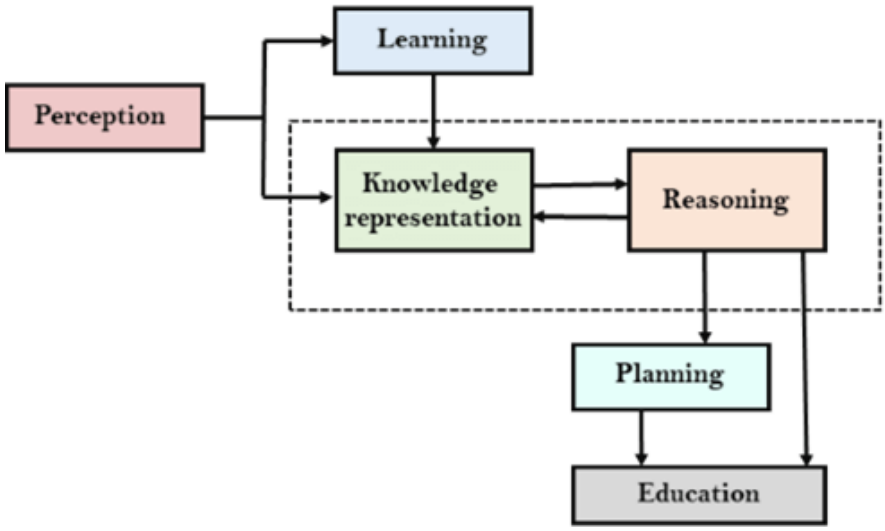
Following figure can show that decision can be taken from sensing the environment and using the knowledge which the agent has. The intelligent behaviour can be display only with the knowledge.



5.4.1 AI knowledge cycle:

In Artificial intelligence, intelligent behaviour can be display through the following components:

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



The above diagram shows that the interaction among the AI system and how can it relate with the real world object and how the components help each other to get expected result. Through perception AI system perceives or retrieves information from the environment. It can be audio, visual, image or data captured by the sensor. All the inputs are transfer t learning agent, the agent will form a fact full data from the sensory input captured by perception component. Now the data will transfer to knowledge representation and reasoning module. Where the agent twill decide the steps of action performed by analysing the data captured at learning phase. Reasoning will make the system to work as human or think as human. These two components work together. And then the decision will forward to the planning and execution component where the actual implementation of decision will do.

5.5 APPROACHES OF KNOWLEDGE REPRESENTATION

5.5.1. Simple relational knowledge:

- In relational model sorting the facts the simplest which uses the relational method, and each object is explain their facts of information systematically in columns.
- The relationship between different entities in database is represented by this famous approach.

- This approach has little opportunity for inference.

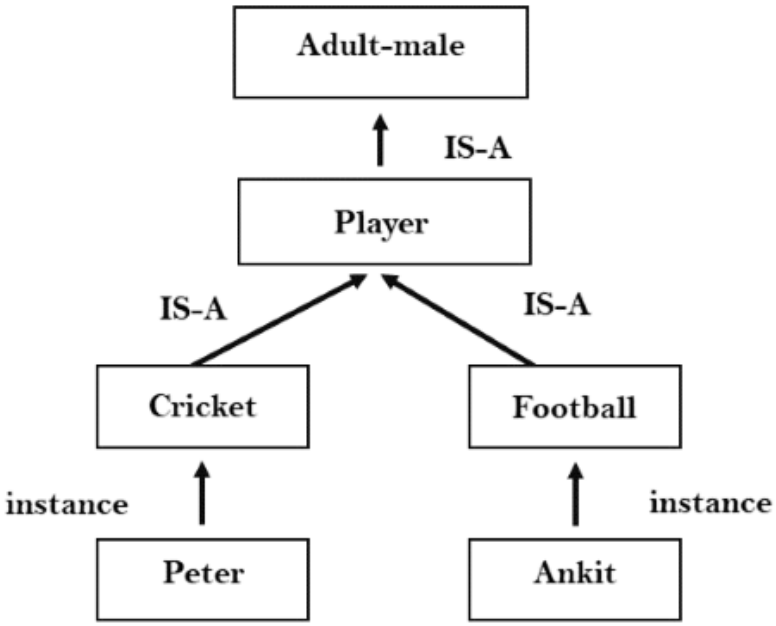
Example: simple relational knowledge representation.

Player	Age	No. Of Matches Played
Player 1	50	56
Player 2	60	122
Player 3	45	80

5.5.2. Inheritable knowledge:

- The data must be stored into a hierarchy of classes, said about the inheritable knowledge approach.
- All classes and objects must be arranged in a hierarchal manner or a generalized form.
- The inheritance property will be applying in this method.
- object inherit values from other class as well as other members of a class.
- In this approach inheritable knowledge is apply, which indicates a relation between object and class, and it is called instance/object relation.
- Collection of attributes and values can be represented by individual frame.
- objects and values are represented in Boxed nodes as shown in diagram.
- objects to their values direction can be indicate by arrow.

Example:



5.5.3. Inferential knowledge:

- knowledge in the form of formal logics represents by Inferential knowledge approach.
- to derive more facts this approach is used.
- Correctness can be achieved through this approach.

Example: Let's suppose there are two statements:

a. Marcus is a man
b. All men are mortal

man(Marcus)
∀x = man (x) ----------> mortal (x)

5.5.4. Procedural knowledge:

Step wise process of hoe to do specific thing and how to proceed with the object is known as Procedural knowledge approach. This uses small programs and codes to proceed the things.

If-Then rule is the most important rule in this approach.

- In this knowledge, LISP language and Prolog language is used to represent the language.
- In this approach heuristic or domain-specific knowledge can represent easily.

1. Representational Accuracy:

- the ability to represent all type of required knowledge as data in provide in KR system.

2. Inferential Adequacy:

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

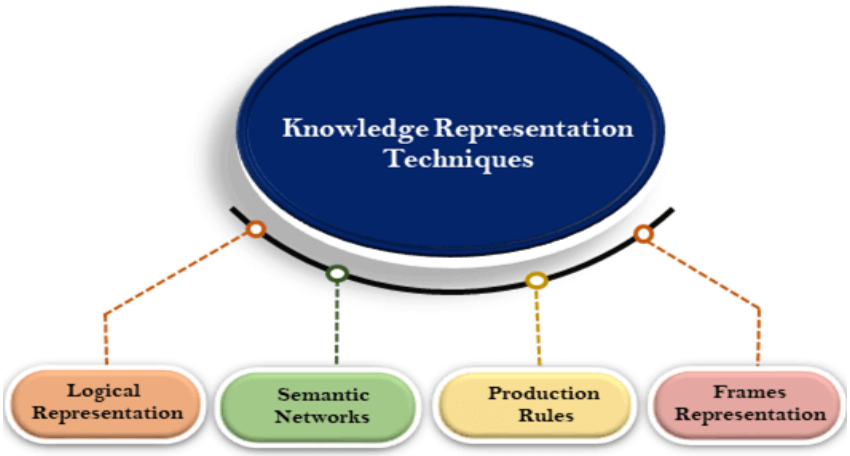
3. Inferential Efficiency:

- The conversion between the inferential knowledge mechanism to the most productive directions by storing appropriate guides can be done through this approach.

4. Acquisitional efficiency- The latest knowledge can be easily upgrade using automatic method is the most important feature of AI

5.6 WAYS OF KNOWLEDGE REPRESENTATION USING RULES

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules



5.6.1. Logical Representation

In our day to day life normal language we have to implement in logical format as system knows only logical language. Logical representation can be done with concrete rules which deals with preposition and it does not have any confusion, with all this thing we can use logical language. To draw a conclusion, we need on the basis of various condition is Logical representation. Communication rules are important rule to use logical expression. It works on precisely defined syntax and semantics which supports the sound inference.

Syntax:

- Set of rules which define how the syntaxes will work. And it will show how to construct the sentence with logical connectives.
- The symbols used in logical language determines knowledge representation.

Semantics:

- Interpretation between sentence and logic are called as semantics.
- The meaning of sentence should not change when we apply logic to the sentence.

Two main categories of Logical Representation

Propositional Logics: a very simple logic called prepositional logic. This runs with the syntax and semantics with truth sentences. The syntax of proportional logic defines the allowable sentence. Each such symbol stands for a preposition that can be true or false.

Predicate logics: models of logical language are the formal structures that constitute the possible world under consideration.

Advantages of logical representation:

1. logical reasoning can be done with logical representation.
2. The programming languages are used for logic representation.

Disadvantages of logical Representation:

1. Some restriction made Logical representations complex.
2. Inference may not be so efficient and logical representation technique may not be easy.

5.6.2. Semantic Network Representation

Alternative of predicate logic for knowledge representation is known as semantics. In Semantic networks, in graphical format we can represent our knowledge. The relationship between the objects may be represent by this network many different applications are present with which can determine Semantic networks. It is very easy to understand and easy to implement also.

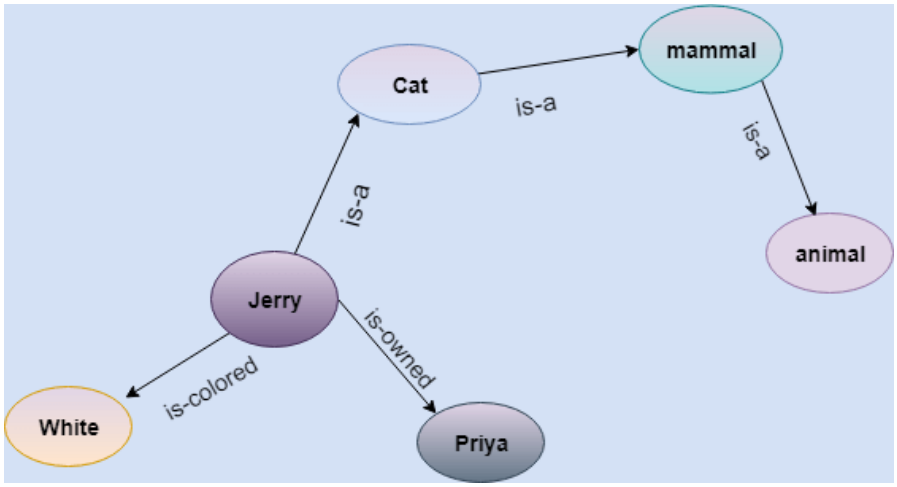
Two types of relations:

IS-A relation (Inheritance)
Kind-of-relation

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

Statements:

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.
4. Jerry is brown colored.
5. All Mammals are animal.



The diagram shows that the representation about the objects which are connected to each other with the help of nodes as entity and line as relationship among them.

Drawbacks in Semantic representation:

1. Semantic networks require more running time to traverse the complete network tree to answer any questions. In some case it

happens that after traversing entire tree there will be no answer for some questions or the solution might not be exist in the network.

2. Semantic networks always try to store the information like human memory, but in some condition, it seems not possible to develop such type of network model.
3. All these sentences cannot be representing with help of normal quantifier for more explanation we need adequate equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition.

Advantages of Semantic network:

1. natural representation of knowledge can come under Semantic networks.
2. The transparent transaction can be done through Semantic networks.
3. It is very simple and easily understandable network representation.

5.6.3. Frame Representation

A frame is a collection of attributes and its respective value and description of an entity record in the world. Frames can divide knowledge into substructures by representing stereo types situations in AI data structure, which holds a collection of slots and slot values. The size and type of slots can be varying with data. Facets are having slots with names and values.

Facets: The various aspects of a slots are known as **Facets**. In some cases we need to put some constraints on the frames which can be a feature of Facets. Example: IF-NEEDED facts are called when data of any particular slot is needed. Any number of slots may have in frame. followed by any number of slot may have in facets followed same as any number of slot facet may have values. A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example of a person Ram,
(Ram)
(PROFESSION (VALUE Professor))
(AGE (VALUE 50))
(WIFE (VALUE Sita))
(CHILDREN (VALUE Lav Kush))
(ADDRESS (VALUE Ayodhya))
(STATE (VALUE Uttar Pradesh))

Advantages of frame representation:

1. By grouping the related data makes the programming easier and hence the frame knowledge representation is very easy.
2. Many applications in AI used the flexible frame representation.
3. Adding slots for new attribute and relations are very easy.
4. Including default data and missing value search can have done very easily.
5. Visualization and understanding frame representation is easy.

Disadvantages of frame representation:

1. inference mechanism is not supported by frame representation.
2. In frame representation Inference mechanism cannot be proceeded smoothly.
3. generalized approach is the main disadvantage of frame representation.

5.6.4. Production Rules

Condition & action together can make Production rules system; it can be follow as:

"If condition then action". It is divided into 3 parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

In production rules agent will check for the situation or condition and if the solution is available for the condition he will do the task as per the condition. The agent with his prior knowledge of the situation will decide which rule may be applied to solve the problem. The steps of action take to solve the situation or problem is known as problem solving steps. And this complete process of problem solving is known as act cycle.

the description of the current state of problems stored in the working memory of problem-solving system, and rule can write as steps perform to solve the problem.

If there is an unknown situation (state) generates, then the agent will use multiple production rules to find the solution for the problem, this situation is called as conflict set. In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

Example:

- IF (at theatre AND got ticket) THEN action (watch movie)
- IF (in the train AND paid AND empty seat) THEN action (sit down).
- IF (pressure is high AND air is locked) THEN action (volume is small).
- IF (the road is slippery) THEN action (driving is dangerous).

Advantages of Production rule:

1. Natural language is used to expressed production rules.
2. Individual rule can easily remove, add or modify therefore the production rules are highly modular.

Disadvantages of Production rule:

1. Production rule system does not store the result of the problem for the future uses hence it does not support any learning capabilities.
2. rule-based production systems are inefficient as during the execution of the program, many rules may be active.

5.6.5. Ontology

Ontology is study about what kind of things or entities exist in the universe. In AI, ontology is the specification of conceptualization, used to help programs and humans to share knowledge about a particular domain.

In turn ontology is a set of concepts, like entity, relationships among the entities, events that are expressed in a uniform way in order to create a vocabulary for information exchange.

For example: consider a map showing hotels, railway station, buildings, schools, hospitals in a particular locality. In this map the symbols used to indicate these entities are enough to describe them.

5.7 PROPERTIES OF GOOD KNOWLEDGE REPRESENTATION SYSTEM

- **Representational Accuracy:** all kinds of required knowledge it should present in the prescribed format.
- **Inferential Adequacy**: manipulation should be done as and when required to represent the new structure which corresponds to the producing new structure with the using existing structure.
- **Inferential Efficiency**: appropriate guides can be used to direct the inferential knowledge mechanism into the most productive directions.

- **Acquisitional efficiency**: using automatic methods it should acquire new knowledge easily.

5.8 ISSUES IN KNOWLEDGE REPRESENTATION

The main aim of knowledge Representation is to give immense feature of inference (conclusions) from knowledge.

Many issues are faced by people during the use of KR techniques. Some are listed below

1. Important Attributes:

There are two main attributes present in the Knowledge representation "instance" and "isa", they represent the inheritance property of knowledge.

2. Relationship among attributes:

The attributes are used to describe the objects and they are themselves entities that we can represent in knowledge.

The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:

i. **Inverse** — in this we will check consistency among the attributes, whenever needed the value is added to one attribute. other in many different ways the entities are related to each.

ii. **Existence in an isa hierarchy** — This tells about specification, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. For example, the height attribute is a specialize attribute of general attribute physical-size. This shows the relationship between generalization-specialization which support inheritance property.

iii. **Technique for reasoning about values** — the reasoning value about the attributes which cannot given explicitly. Different type of reasoning, like, weight: must be in number, Age: child age must not be greater than the age of parents. The values are based on knowledge.

iv. **Single valued attributes** — the unique value about a specific attribute. For example, a football player can at time have only a member of one team. KR systems provide different approaches for single valued attributes.

3. Choosing Granularity:

- The level of knowledge and the primitives required for the KR can done with choosing granularity.
- It should have a small number or large number of High-level facts or low-level primitives.
- In this Low-level primitives may require a lot of storage while High-level facts may not be adequate for inference while.

4. Set of objects:

To present the appropriate properties of objects which are the member of a set that are true;

Example: Consider the assertion made in the sentences:

"there are more ladies than gents in Kerala", and
"beautiful girls can be found all over the world."

To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English.

The sets of objects can be representing in: if a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.

- universal quantifier can be used for the representation of logical sentences
- in this hierarchical structure, node represent as a set and inheritance propagate set level assertion down to individual.

5. Finding Right structure:

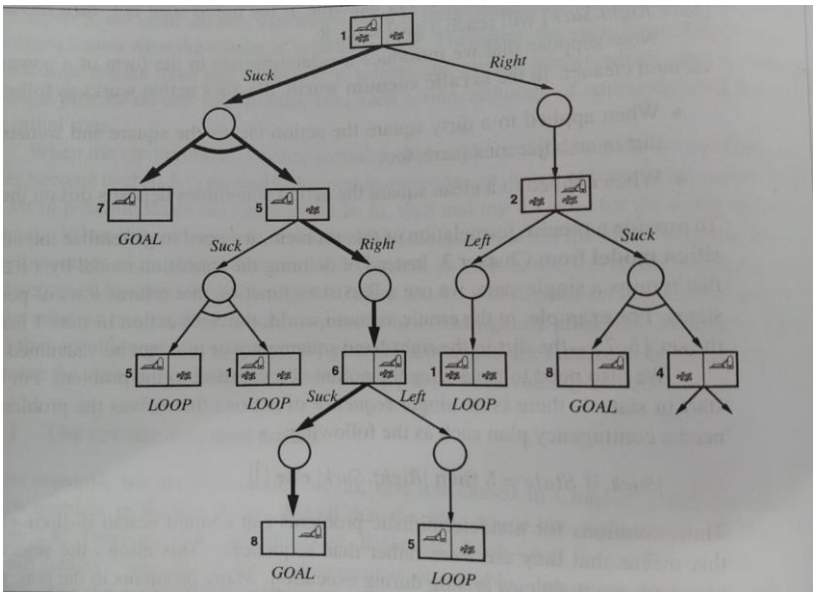
In this phase one need to find the right structure to represent the data or a particular situation.
This can be done through the selecting an initial structure and then revising the choice.
While doing it, one need to consider the following problem
How does an initial selection of the most appropriate structure?

- How we can fill the appropriate details of current situations and how we can relate with each other.
- if the variable chosen initially turns out not to be appropriate what should be the corrective step.
- What should be the step if none of the available structures is correct.
- When to create and remember a new structure.

5.9 AND- OR SEARCH TREE

Now the question arises how to find solutions to nondeterministic problems. In other hand in in a deterministic environment, branching can be introduced by the agent's own choices in every state. And these all node called as **OR nodes**. In the example of vacuum world, in OR condition agent has a liberty to choose Left or Right or Suck. In the other hand of nondeterministic environment, branching can be introduced by the environment's choice of output for each and every action. We call these nodes **AND nodes**. For example, the Suck action in state 1 leads to a state in the set {5,7}, so the agent would need to find the plan for state 5 and for state 7. These two kinds of nodes alternate, leading to an AND-OR **tree** as illustrated in following figure.

A best solution for AND-OR search problem is a construction of subtree that has goal node at each leaf node, every node specifies the action to meet the goal node of its OR node, and (3) consist every extension node at each branch of its AND nodes.in the diagram solution of this darken and; it corresponds to the process given in Equation. (it uses if-then-else conditional statements to handle the AND branches, but when there are present more branches of nodes, it easy to use case constructor.



Here we have to Modify the basic problem-solving agent as seen in above Figure. The contingent solutions can be find straightforward when executed tree. One can also travel with different path where we get different agent design, in which the agent can travel the entire tree and fin some paths like in some situation it has to deal with contingencies and act *before* it has found a guaranteed plan This situation is called as **interleaving** of search and this type of execution can be useful for exploration of problems and for game tree.

AND OR graph search algorithm is a recursive, depth-first algorithm. Important feature of that is how the algorithm deals with cycles, which often arise in nondeterministic problems. In current state If the identical path is there from the root, then it returns with failure. it shows the solution is not from the current state; it simply turns out with a noncyclic solution, it must be reachable from the earlier node of the current state, so the new path can be discarded. With this we can say that the termination of algorithm is at every finite state space, because every node of a tree makes path must reach a goal, a dead end, or a repeated state. Here one thing we should notice that the algorithm sometimes goes with the current state repetition of a state on some *other* path from the root, which is important for efficiency.

AND-OR graphs follow the properties of breadth-first or best-first methods. In AND-OR graph The heuristic function must be modified to calculate the cost of a contingent solution rather than a sequence, but the notion of admissibility carries over and there is an analog of the A* algorithm for finding optimal solution.

5.10 SUMMARY

As we have learnt that knowledge base is a setoff representations of facts/information about surroundings. Every single representation in the set is called a sentence and sentence are expressed with the help of formal representation language. In artificial intelligence Knowledge representation and reasoning are the most important feature with which we can identify how the machine can contribute in thinking and acting like human. This specify the intelligent bbehaviour of agents. Types of knowledge are declarative knowledge, meta knowledge, procedural knowledge, structural knowledge. There are many ways to represent the knowledge like simple representation, semantic network, frame representation, logical representation, ontology and many more. There are many properties of knowledge representation such as representation accuracy, Inferential Adequacy, Inferential Efficiency, Acquisitional efficiency. AND-OR graphs follow the properties of breadth-first or best-first methods. In AND-OR graph The heuristic function must be modified to calculate the cost of a contingent solution. AND OR graph search algorithm is a recursive, depth-first algorithm. Important feature of that is how the algorithm deals with cycles, which often arise in nondeterministic problems.

5.11 UNIT AND EXCERCISE

1. What is Knowledge? Explain how it can have mapped with intelligence?
2. Explain types of knowledge.
3. Discuss about AND-OR search graph.
4. Explain different approaches of knowledge representation.
5. Discuss various types of knowledge representation ways.
6. Explain ways of knowledge representation.
7. What are the properties of good knowledge representation techniques?
8. Explain AI life cycle.
9. What type of problem are facing during knowledge representation?

5.12 REFERENCES

- Artificial Intelligence a Modern Approach 4th Edition by Peter Norvig and Stuart Russell published by Pearson
- Artificial Intelligence a Modern Approach 3rd Edition by Peter Norvig and Stuart Russell published by Pearson
- Understanding machine learning from theory to algorithms 1st Edition Shai Shalev and Shai Ben David published by Cambridge University Press

❄❄❄❄❄❄❄

6

CONCEPTS OF SOFT COMPUTING

Unit Structure

6.1 SOFT COMPUTING VS HARD COMPUTING

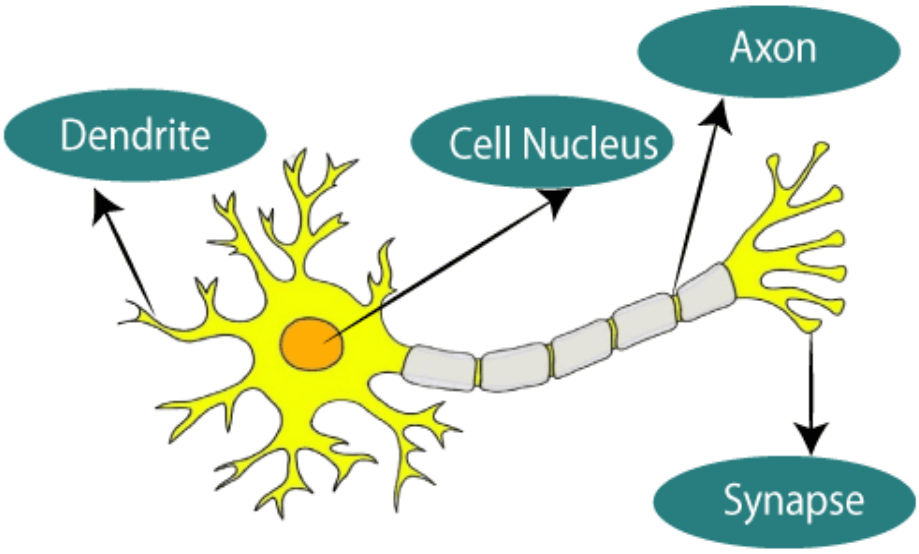
Soft computing	Hard computing
1. Soft Computing is liberal of inexactness, uncertainty, partial truth and approximation.	**1.** Hard computing needs an exactly state analytic model.
2. Soft computing relies on formal logic and probabilistic reasoning.	**2.** Hard computing relies on binary and crisp logic system.
3. Soft computing is stochastic in nature.	**3.** Hard computing is deterministic in nature.
4. Soft computing can perform parallel computations.	**4.** Hard computing uses sequential computations.
5. Soft computing works on ambiguous and noisy data	**5.** Hard computing works on exact data.

6.1.1 Soft Computing Constituents

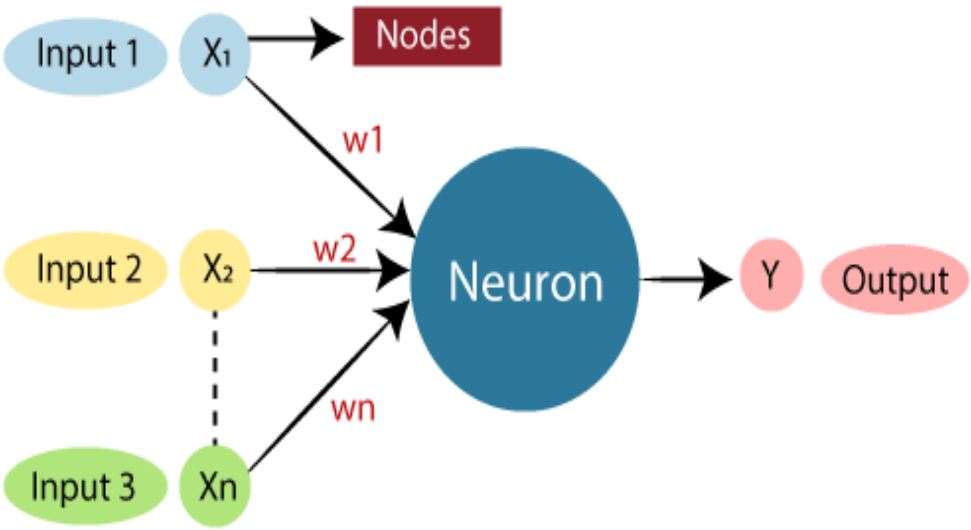
6.1.2 Artificial Neural Network (ANN)

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the

human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.



The typical Artificial Neural Network looks something like the given figure.

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

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic

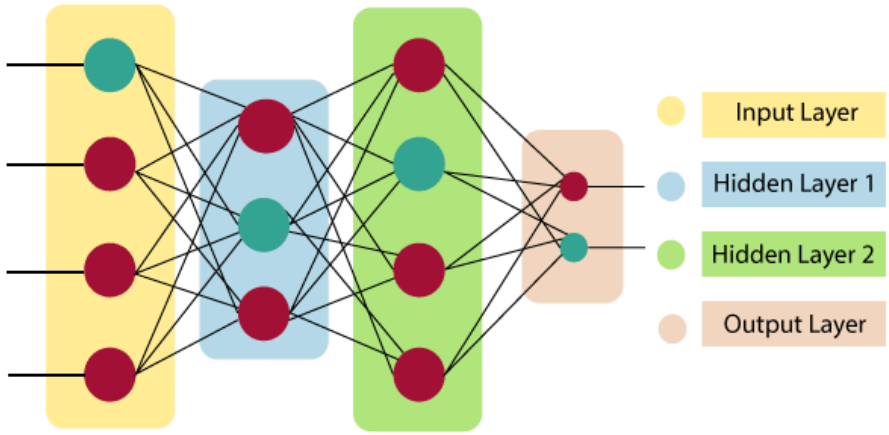
Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

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

6.1.3 The architecture of an artificial neural network

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



Input Layer:

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

Hidden Layer:

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

Output Layer:

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

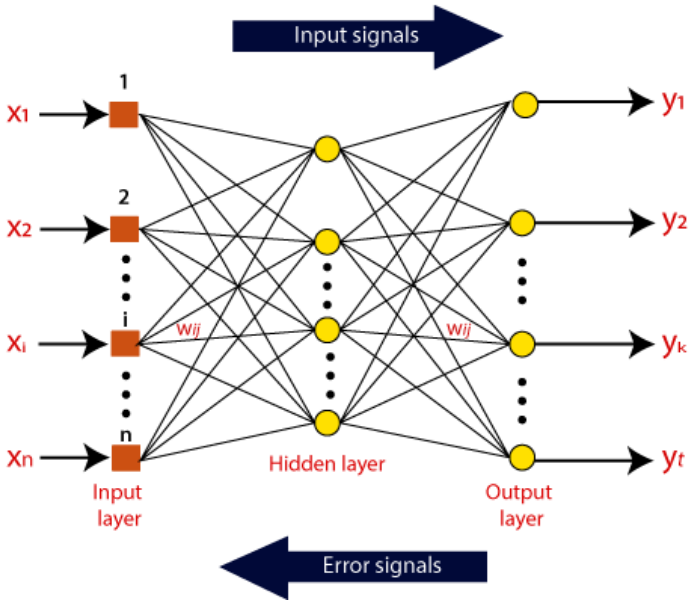
The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

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

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

How do artificial neural networks work?

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



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the

artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:

Binary:

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

Sigmoidal Hyperbolic:

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

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

Where ???? is considered the Steepness parameter.

6.1.4 Types of Artificial Neural Network:

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

Feedback ANN:

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

Feed-Forward ANN:

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

Prerequisite

No specific expertise is needed as a prerequisite before starting this tutorial.

Audience

Our Artificial Neural Network Tutorial is developed for beginners as well as professionals, to help them understand the basic concept of ANNs.

Problems

We assure you that you will not find any problem in this Artificial Neural Network tutorial. But if there is any problem or mistake, please post the problem in the contact form so that we can further improve it.

6.2 FUZZY LOGIC

What is Fuzzy Logic?

The term Fuzzy means something that is a bit vague. When a situation is vague, the computer may not be able to produce a result that is True or False. As per Boolean Logic, the value 1 refers to True and 0 means False. But a Fuzzy Logic algorithm considers all the uncertainties of a problem, where there may be possible values besides True or False.

The term Fuzzy Logic was first described by **Lotfi Zadeh in 1965**. He thought that traditional computer logic is not capable of handling unclear or vague data. Similar to humans, there are many possible values between True and False that a computer can incorporate. These can be:

- Certainly yes
- Possibly yes
- Can’t say
- Possibly no
- Certainly no

Check out this simple example of Fuzzy Logic:

Problem – *Is it hot outside?*
Boolean Logic
Solution:

- Yes (1.0)
- No (0)

According to conventional Boolean Logic, the algorithm will take a definite input and produce a precise result Yes or No. This is represented by 0 and 1, respectively.

Fuzzy Logic

Solution:

- Very hot (0.9)
- Little hot (0.20)
- Moderately hot (0.35)
- Not hot (1.0)

As per the above example, Fuzzy Logic has a wider range of outputs, such as very hot, moderately hot and not hot. These values between 0 and 1 display the range of possibilities.

So, in cases where accurate reasoning cannot be provided, Fuzzy Logic provides an acceptable method of reasoning. An algorithm based on Fuzzy Logic takes all available data while solving a problem. It then takes the best possible decision according to the given input.

Charles Elkan, Assistant Professor of the Computer Science and Engineering department at the University of California at San Diego, shed some light upon Fuzzy Logic. He said that Fuzzy Logic in artificial intelligence is a generalized form of standard logic, where any concept might have a truth degree ranging between 0.0 and 1.0. Fuzzy Logic can be used for vague concepts, such as the characteristic of tallness. For example, we can say that President Clinton is tall, and the concept can have a degree of truth of 0.9.

He further said that Fuzzy Logic is very useful in low-level machine control especially in consumer appliances. Some special-purpose microprocessors built on Fuzzy Logic perform fuzzy operations on their hardware.

6.2.1 Fuzzy Logic Architecture:

The architecture of Fuzzy Logic consists of the following components:

Rule base

This is the set of rules along with the If-Then conditions that are used for making decisions. But, modern developments in Fuzzy Logic have reduced the number of rules in the rule base. These set of rules are also called a knowledge base.

Fuzzification

This is the step where crisp numbers are converted into fuzzy sets. A crisp set is a set of elements that have identical properties. Based on certain logic, an element can either belong to the set or not. Crisp sets are based on binary logic – Yes or No answers.

Here, the error signals and physical values are converted into a normalized fuzzy subset. In any Fuzzy Logic system, the fuzzifier separates the input signals into five states that are:

- Large positive
- Medium positive
- Small
- Medium negative
- Large negative

The fuzzification process converts crisp inputs, such as room temperature, fetched by sensors and passes them to the control system for further processing. A Fuzzy Logic control system is based on Fuzzy Logic. Common household appliances, such as air-conditioners and washing machines have Fuzzy Control systems within them.

Inference Engine

The inference engine determines how much the input values and the rules match. The rules are applied based on the input values received. Then, the rules are used to develop control actions. The inference engine and the knowledge base together are called a controller in a Fuzzy Logic system.

Defuzzification

This is the inverse process of fuzzification. Here, the fuzzy values are converted into crisp values by mapping. There will be several defuzzification methods for doing this, but the best one is selected as per the input. This is a complicated process where methods, such as the maximum membership principle, weighted average method and centroid method, are used.

6.2.2 Advantages of Fuzzy Logic in Artificial Intelligence

1. It is a robust system where no precise inputs are required
2. These systems are able to accommodate several types of inputs including vague, distorted or imprecise data
3. In case the feedback sensor stops working, you can reprogram it according to the situation
4. The Fuzzy Logic algorithms can be coded using less data, so they do not occupy a huge memory space
5. As it resembles human reasoning, these systems are able to solve complex problems where ambiguous inputs are available and take decisions accordingly
6. These systems are flexible and the rules can be modified
7. The systems have a simple structure and can be constructed easily
8. You can save system costs as inexpensive sensors can be accommodated by these systems

Disadvantages of Fuzzy Logic in Artificial Intelligence

1. The accuracy of these systems is compromised as the system mostly works on inaccurate data and inputs
2. There is no single systematic approach to solve a problem using Fuzzy Logic. As a result, many solutions arise for a particular problem, leading to confusion
3. Due to inaccuracy in results, they are not always widely accepted
4. A major drawback of Fuzzy Logic control systems is that they are completely dependent on human knowledge and expertise
5. You have to regularly update the rules of a Fuzzy Logic control system
6. These systems cannot recognize machine learning or neural networks
7. The systems require a lot of testing for validation and verification

6.2.3 Applications of Fuzzy Logic

The applications of Fuzzy Logic are spread across several fields. They are as follows:

1. Medicine

- Controlling arterial pressure when providing anaesthesia to patients
- Used in diagnostic radiology and diagnostic support systems
- Diagnosis of prostate cancer and diabetes

2. Transportation systems

- Handling underground train operations
- Controlling train schedules
- Braking and stopping vehicles based on parameters, such as car speed, acceleration and wheel speed.

3. Defence

- Locating and recognizing targets underwater
- Supports naval decision making
- Using thermal infrared images for target recognition
- Used for controlling hypervelocity interceptor.

4. Industry

- Controlling water purification plants
- Handling problems in constraint satisfaction in structural design
- Pattern analysis for quality assurance
- Fuzzy Logic is used for tackling sludge wastewater treatment.

5. Naval control

- Steer ships properly
- Selecting the optimal or best possible routes for reaching a destination
- Autopilot is based on Fuzzy Logic
- Autonomous underwater vehicles are controlled using Fuzzy Logic.

6.3 GENETIC ALGORITHM APPLICATIONS OF SOFT COMPUTING

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of **Genetics and Natural Selection**. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve optimization problems, in research, and in machine learning.

6.3.1 Introduction to Optimization

Optimization is the process of **making something better**. In any process, we have a set of inputs and a set of outputs as shown in the following figure.

What are Genetic Algorithms?

Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs is a subset of a much larger branch of computation known as **Evolutionary Computation**.

GAs was developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success.

In GAs, we have a **pool or a population of possible solutions** to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals. This is in line with the Darwinian Theory of "Survival of the Fittest".

Advantages of Genetic Algorithms

1. Does not require any derivative information (which may not be available for many real-world problems).
2. Is faster and more efficient as compared to the traditional methods.
3. Has very good parallel capabilities.
4. Optimizes both continuous and discrete functions and also multi-objective problems.
5. Provides a list of "good" solutions and not just a single solution.
6. Always gets an answer to the problem, which gets better over the time.
7. Useful when the search space is very large and there are a large number of parameters involved.

Limitations of Genetic Algorithms

1. GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
2. Fitness value is calculated repeatedly which might be computationally expensive for some problems.
3. Being stochastic, there are no guarantees on the optimality or the quality of the solution.
4. If not implemented properly, the GA may not converge to the optimal solution.

6.3.2 GA – Motivation

Genetic Algorithms have the ability to deliver a "good-enough" solution "fast-enough". This makes genetic algorithms attractive for use in solving optimization problems. The reasons why GAs are needed are as follows –

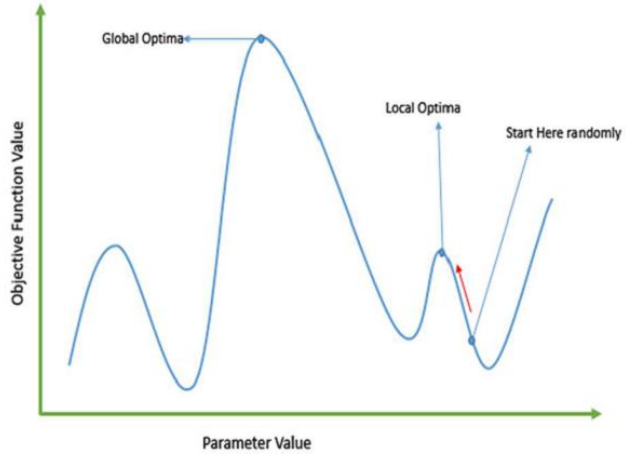
Solving Difficult Problems

In computer science, there is a large set of problems, which are **NP-Hard**. What this essentially means is that, even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs prove to be an efficient tool to provide **usable near-optimal solutions** in a short amount of time.

Failure of Gradient Based Methods

Traditional calculus-based methods work by starting at a random point and by moving in the direction of the gradient, till we reach the top of the hill. This technique is efficient and works very well for single-peaked objective functions like the cost function in linear regression. But, in most real-world situations, we have a very complex problem called as landscapes, which are made of many peaks and many valleys, which causes such

methods to fail, as they suffer from an inherent tendency of getting stuck at the local optima as shown in the following figure.



Getting a Good Solution Fast

Some difficult problems like the Travelling Salesperson Problem (TSP), have real-world applications like path finding and VLSI Design. Now imagine that you are using your GPS Navigation system, and it takes a few minutes (or even a few hours) to compute the "optimal" path from the source to destination. Delay in such real-world applications is not acceptable and therefore a "good-enough" solution, which is delivered "fast" is what is required.

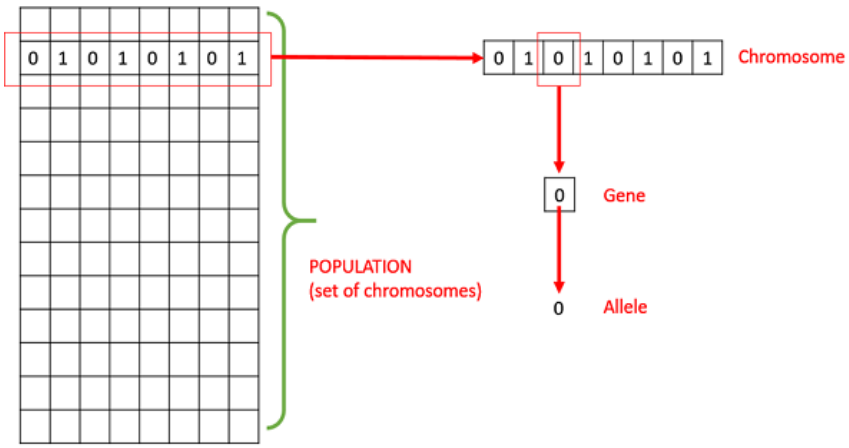
6.3.3 Genetic Algorithm

This section introduces the basic terminology required to understand GAs. Also, a generic structure of GAs is presented in both **pseudo-code and graphical forms**. The reader is advised to properly understand all the concepts introduced in this section and keep them in mind when reading other sections of this tutorial as well.

Basic Terminology

Before beginning a discussion on Genetic Algorithms, it is essential to be familiar with some basic terminology which will be used throughout this tutorial.

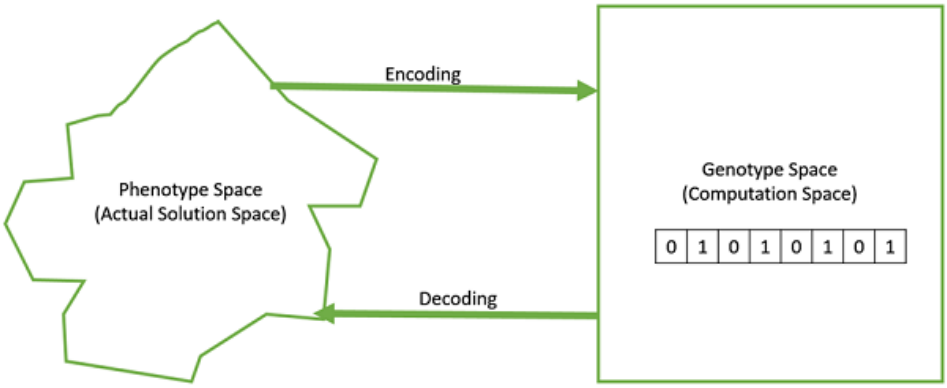
- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** − A chromosome is one such solution to the given problem.
- **Gene** − A gene is one element position of a chromosome.
- **Allele** − It is the value a gene takes for a particular chromosome.



- **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** − Phenotype is the population in the actual real-world solution space in which solutions are represented in a way they are represented in real world situations.
- **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

For example, consider the 0/1 Knapsack Problem. The Phenotype space consists of solutions which just contain the item numbers of the items to be picked.

However, in the genotype space it can be represented as a binary string of length n (where n is the number of items). A **0 at position x** represents that $\mathbf{x^{th}}$ item is picked while a 1 represents the reverse. This is a case where genotype and phenotype spaces are different.



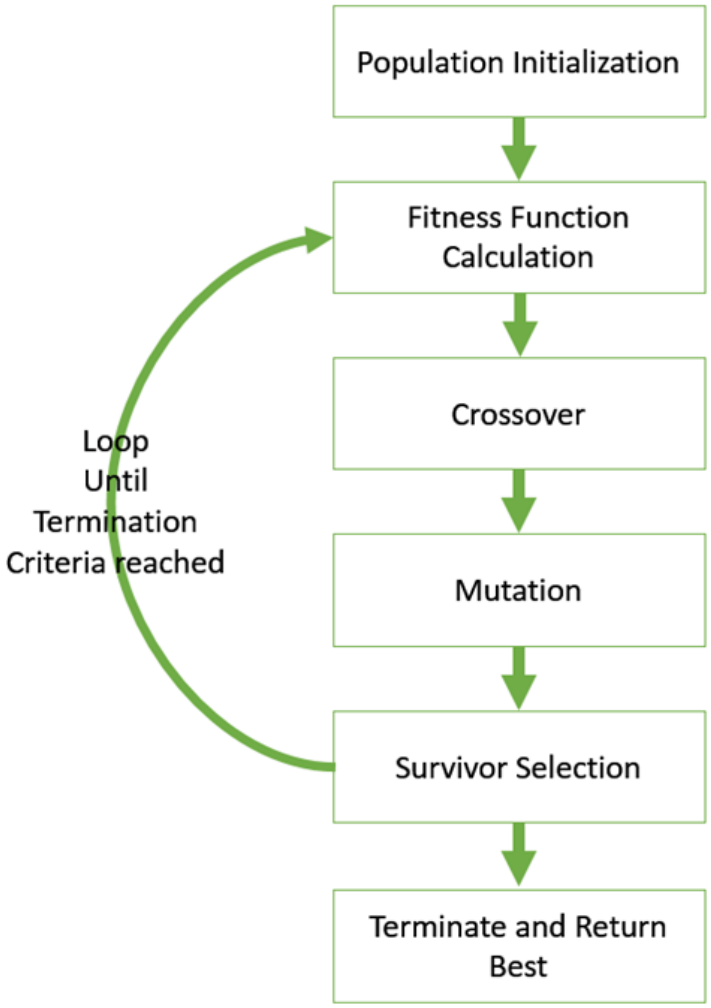
- **Fitness Function** − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
- **Genetic Operators** − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

6.3.4 Basic Structure

The basic structure of a GA is as follows −

We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating. Apply crossover and mutation operators on the parents to generate new off-springs. And finally, these off-springs replace the existing individuals in the population and the process repeats. In this way genetic algorithms actually try to mimic the human evolution to some extent.

Each of the following steps are covered as a separate chapter later in this tutorial.



A generalized pseudo-code for a GA is explained in the following program −

```
GA()
  initialize population
  find fitness of population

  while (termination criteria is reached) do
    parent selection
    crossover with probability pc
    mutation with probability pm
    decode and fitness calculation
    survivor selection
    find best
  return best
```

Genetic Algorithms are primarily used in optimization problems of various kinds, but they are frequently used in other application areas as well.

In this section, we list some of the areas in which Genetic Algorithms are frequently used. These are −

- **Optimization** − Genetic Algorithms are most commonly used in optimization problems wherein we have to maximize or minimize a given objective function value under a given set of constraints. The approach to solve Optimization problems has been highlighted throughout the tutorial.
- **Economics** − GAs are also used to characterize various economic models like the cobweb model, game theory equilibrium resolution, asset pricing, etc.
- **Neural Networks** − GAs are also used to train neural networks, particularly recurrent neural networks.
- **Parallelization** − GAs also have very good parallel capabilities, and prove to be very effective means in solving certain problems, and also provide a good area for research.
- **Image Processing** − GAs are used for various digital image processing (DIP) tasks as well like dense pixel matching.
- **Vehicle routing problems** − With multiple soft time windows, multiple depots and a heterogeneous fleet.
- **Scheduling applications** − GAs are used to solve various scheduling problems as well, particularly the time tabling problem.
- **Machine Learning** − as already discussed, genetics-based machine learning (GBML) is a niche area in machine learning.
- **Robot Trajectory Generation** − GAs have been used to plan the path which a robot arm takes by moving from one point to another.

- **Parametric Design of Aircraft** − GAs have been used to design aircrafts by varying the parameters and evolving better solutions.
- **DNA Analysis** − GAs have been used to determine the structure of DNA using spectrometric data about the sample.
- **Multimodal Optimization** − GAs are obviously very good approaches for multimodal optimization in which we have to find multiple optimum solutions.
- **Traveling salesman problem and its applications** − GAs have been used to solve the TSP, which is a well-known combinatorial problem using novel crossover and packing strategies

Multiple Choice Questions

1) Which of the following is associated with fuzzy logic?

   a) Crisp set logic
   b) Many-valued logic
   c) Two-valued logic
   d) Binary set logic

2) The truth values of traditional set theory can be defined as _____ and that of fuzzy logic is termed as _____.

   a) Either 0 or 1, either 0 or 1.
   b) Between 0 & 1, either 0 or 1.
   c) Either 0 or 1, between 0 & 1.
   d) Between 0 & 1, between 0 & 1.

3) How many types of random variables are there in Fuzzy logic?

   a) 2
   b) 4
   c) 1
   d) 3

4) Which of the following fuzzy operators are utilized in fuzzy set theory?

a) AND
b) OR
c) NOT
d) EX-OR

5) What happens if chain-termination mutation is in the S gene?

   a) Cell lysis is blocked
   b) Growth of cells containing low levels of packaging proteins is notallowed
   c) The lysis of cells is not carried artificially
   d) Packaging is not carried out efficiently

6) A perceptron can be defined as _____

a) A double layer auto-associative neural network
b) A neural network with feedback
c) An auto-associative neural network
d) A single layer feed-forward neural network with pre-processing

7) What is meant by an auto-associative neural network?

a) A neural network including feedback
b) A neural network containing no loops
c) A neural network having a single loop
d) A single layer feed-forward neural network containing feedback

8) Each connection link in ANN is linked with _____ that contains statics about the input signal.

a) Neurons
b) Activation function
c) Weights
d) Bias

9) What is the name of the process that represents modified elements of the DNA?

a) Selection
b) Mutation
c) Recombination
d) None of the above

10) What is the name of the operator that is functioned on the population?

a) Recombination
b) Reproduction
c) Mutation
d) None of the above

❄❄❄❄❄❄❄

7

ARTIFICIAL NEURAL NETWORK

Unit Structure

7.0 INTRODUCTION

When we fail to get satisfactory predictive models use neural networks and hence neural networks have emerged as advanced data mining tools. The term neural network is inspired from biological concept. The basic unit by which the brain works is a neuron and it is used for transmitting electrical signals from one end to another. Electrical signals are transmitted from dendrites to the axon terminals through the axon body. Further the electrical signals continue to be transmitted across the synapse from one neuron to another neuron. The human brain consists of approximately 100 billion neurons. It is very difficult for us to imitate this level of complexity with existing computers.

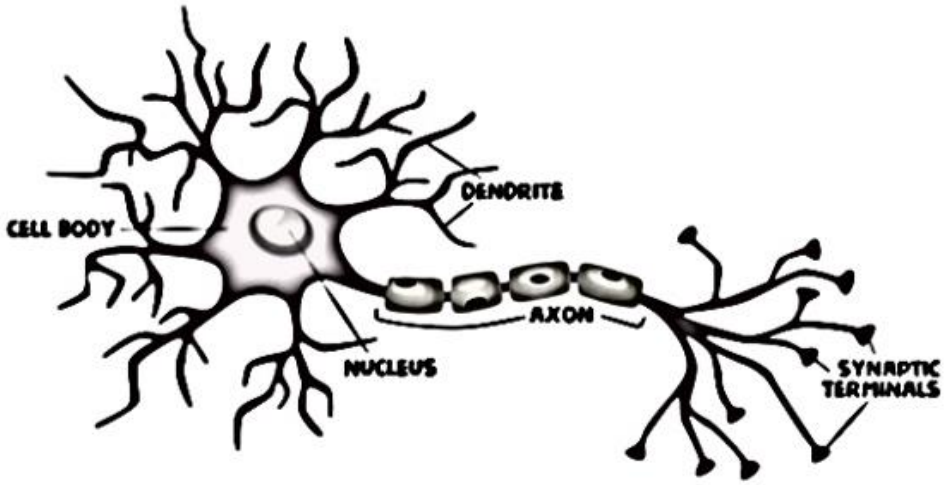


fig. 7.1 Biological neuron

Working of a Biological Neuron

As proven within the above diagram, a typical neuron includes the subsequent 4 components with the help of which we can explain its running −

- Dendrites − they are tree-like branches, accountable for receiving the records from other neurons which are linked to it. In other words, we can say that they're like the ears of neuron.
- Cell Body − it is the cellular frame of the neuron and is answerable for processing of records, they've received from dendrites.
- Axon − it is similar to a cable via which neurons send the facts.
- Synapses − it's the relationship among the axon and different neuron dendrites It is used for sending information from other neurons, in the form of electrical impulses, enters the dendrites at connection points

ANN had been investigated by using researchers for greater than half of a century the formal has a look at of ANN started out with the pioneering paintings of McCulloch & Pitts in 1943. Inspired via the results of biological experiments and observations McCulloch & Pitts added a simple version of a binary artificial neuron that captured some of the capabilities of biological neurons. using facts processing machines the next day within the brain McCulloch & Pitts built their neural community model the use of a huge number of interconnected synthetic binary neurons. From this starting neural network community research became quite popular in the past due 1950s and early 1960s after a thorough evaluation open early neural community model name does the perceptron which use no hidden layer in addition to a pessimistic evaluation of studies capacity by way of Minsky and Papert in 1969, interest neural networks diminished.

7.1 FUNDAMENTAL CONCEPT

The human brain has capabilities for information processing and problem solving those modern computers cannot compete within many aspects. It has been postulated that a model or a system that is enlightened and supported by the results from the brain research is the structure similar to the biological neurons could exhibit similar intelligent functionality. Based on this bottom up approach a name also known as connectionist model parallel distributed processing models neuromorphic systems or simply neural networks has been developed as a biologically inspired model for doing various tasks.

Neural computing refers to a pattern recognition methodology for a machine learning. The resulting model from neural computing is often called as artificial neural network or a neural network. Neural networks have been used in many business applications for forecasting predictions, pattern recognition and classification. It is a key computing component of any data mining toolkit. Applications of neural networks can be found in

finance, marketing, manufacturing, operations, information systems and so on.

7.2 ARTIFICIAL NEURAL NETWORK

An artificial neural network is a series of algorithms which aim at finding underlying relationships in a set of data by using a process which is inspired by the working of human brain. An artificial neural network is based on Feed-forward strategy.

A neural network is an oriented graph such as nodes, which within the biological analogy constitute neurons, linked by way of arcs, which correspond to dendrites and synapses. every arc is related to a weight, even as at each node an activation characteristic is described which is applied to the values obtained as enter by means of the node alongside the incoming arcs, adjusted by using the weights of the arcs. The training stage is achieved via analysing in series the observations contained in the training set one after the opposite and through editing at each iteration the weights associated with the arcs.

Neural networks are organised in various layers:

- Input layer

  The training observations are fed through these neurons.
- Hidden layer

  The intermediate layer is called as hidden layer. It is said to be hidden because there is no contact with the outside world directly.

  The outputs of each hidden layer are given as input to the next layer.
- Output layer

  This is the last layer of neural network. It is producing results and predictions.

7.3 BRAIN VS. COMPUTER - COMPARISON BETWEEN BIOLOGICAL NEURON AND ARTIFICIAL NEURON

1. Size

Our brain consists of about an average of 86 billion neurons and more than a 100 trillion synapses.

The artificial neural network consists of a smaller number of neurons as compared to human brain. Perceptrons just take inputs on their “dendrites” and generate output on their “axon branches”. A single layer perceptron network includes numerous perceptrons that aren’t interconnected: all of them just carry out this very same task at once. Deep Neural Networks normally encompass input neurons, output neurons and neurons in the hidden layers, in-between.

2. Storage

While allocating storage for a new process artificial neural network is strictly irreplaceable as the old location is used for saving the previous process whereas allocation for storage of new process is very easy by adjusting the interconnection strength in biological neural networks.

3. Mode of operation & speed

in artificial neural network processes are operated in sequential mode whereas in biological neural network processes can operate in massive parallel operations.

Processing speed of artificial neural network is faster as compared to biological neural networks.

4. Fault-tolerance

Biological neurons are fault tolerant because the information is stored redundantly so any minor issues will not result in memory loss. The brain can recover or heal to an extent whereas artificial neural networks are not modelled for fault tolerance or self-generation though the recovery is possible by returning back to the currently saved state.

5. Consumption of power

Artificial neural networks our way less efficient than the biological neural networks with respect to the power consumption.

6. Learning process

Artificial neural network cannot learn by recalling information. The biological neural network can learn with the help of recalling and the learning process deepens even during the sleep. The artificial neural network cannot do self-learning, the model can be trained to learn but has limitations as we cannot add more neurons in the trained network.

7.4 BASIC MODELS OF ARTIFICIAL NEURAL NETWORK

Basic models of artificial neural network are classified into 3 different categories:

7.4.1 Interconnections
7.4.2 Learning rules
7.4.3 Activation function

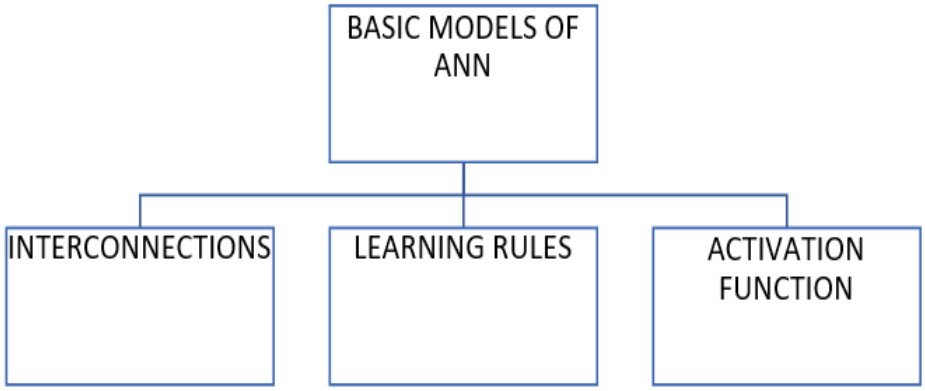


fig 7.2 Basic models of ANN

7.4.1 Interconnections

Based on interconnections ANN can be further classified as:

- Feedforward network
- Feedback network
- Recurrent network

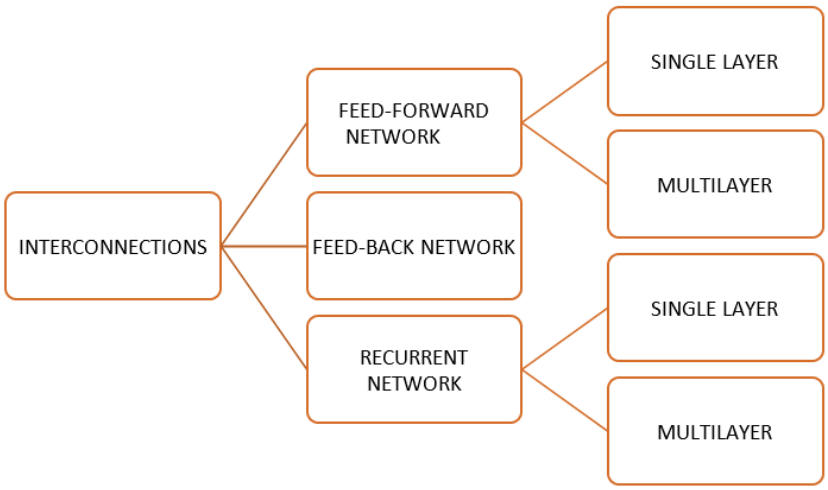


fig 7.3 Classification based on Interconnections

- Single layer feed-forward network

  In this type of network, we have only two layers, i.e. input layer and output layer however the input layer does not count due to the fact that no computation is performed in this layer.

Output Layer is formed when specific weights are applied on input nodes and the cumulative effect according to the per node is taken.

After this, the neurons together deliver the output layer to compute the output signals.

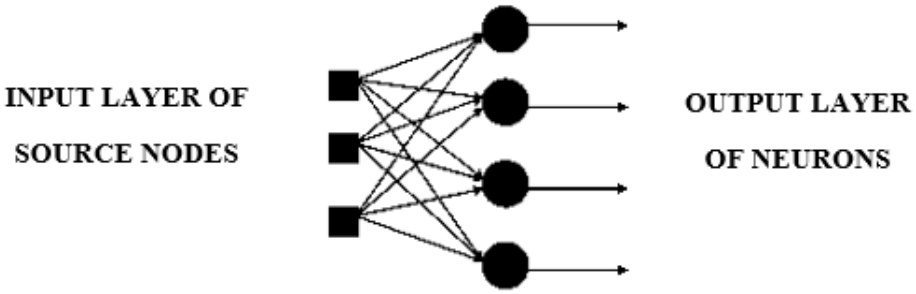


fig 7.4 Single layer feed-forward network

Input and outputs are linked to each other. The simplest kind of neural network is a single-layer perceptron network and corresponds to a single neuron that receives as input the values (x_1, x_2,...,x_n) along the incoming connections, and returns an output value f (x). The input values coincide with the values of the explanatory attributes, while the output value determines the prediction of the response variable y. Each of the n input connections is associated with a weight wj . An activation function g and a constant ϑ, called the distortion, are also assigned.

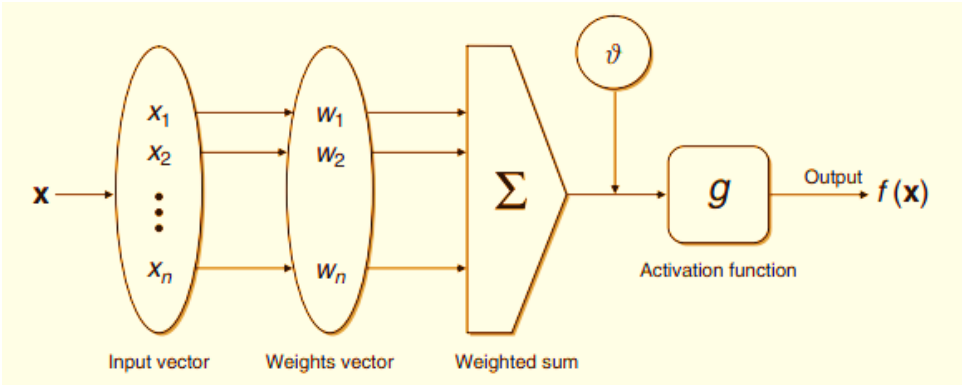


fig 7.5 Operation of a single unit in a neural network

Suppose that the values of the weights and the distortion have already been determined during the training phase. The prediction for a new observation x is then derived by performing the following steps.

First, the weighted linear combination of the values of the explanatory variables for the new observation is calculated and the distortion is subtracted from it

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta = \mathbf{w}'\mathbf{x} - \vartheta.$$

(7.1)

The prediction f (x) is then obtained by applying the activation function g to the linear combination of the predictors

$$f(\mathbf{x}) = g(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta) = g(\mathbf{w}'\mathbf{x} - \vartheta).$$

(7.2)

The purpose of the function g is to map the linear combination into the set $H = \{v_1, v_2,...,v_H\}$ of the values assumed by the target variable, usually by means of a sigmoid profile. For binary classification problems we have $H = \{-1, 1\}$, so that one may select $g(\cdot) = \text{sgn}(\cdot)$, making the prediction coincide with the sign of the weighted sum in (7.1)

$$f(\mathbf{x}) = \text{sgn}(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta) = g(\mathbf{w}'\mathbf{x} - \vartheta).$$

(7.3)

An iterative algorithm is then used to determine the values of the weights wj and the distortion ϑ, examining the examples in sequence, one after the other. For each example xi the prediction f (xi) is calculated, and the value of the parameters is then updated using recursive formulas that take into account the error yi − f (xi).

classification problems it is possible to give a geometrical interpretation of the prediction obtained using a Rosenblatt perceptron. Indeed, if we place the m observations of the training dataset in the space Rn, the weighted linear combination in (7.1) calculated for xi expresses the slack between the observation and the hyperplane

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta = \mathbf{w}'\mathbf{x} - \vartheta.$$

(7.4)

The purpose of the activation function $g(\cdot) = \text{sgn}(\cdot)$ is therefore to establish if the point associated with the example xi is placed in the lower or upper halfspace with respect to the separating hyperplane.

- Multi-layer feed-forward network

A multi-layer feed-forward network is having a more complex structure than a simple perceptron. In this network there are no feedback connections in which outputs of the model are fed back into itself.

It consists of following component:

Input nodes:

The use of input nodes is to get input values of explanatory attributes for every observation. The number of input nodes present in input layer is equal to the number of explanatory variables. These nodes are giving inputs to network but does not perform any computation.

Hidden nodes:

These nodes are used to apply transformations to the input values inside the given network.

All hidden nodes are connected to the incoming arcs that arrives from other hidden nodes or from input nodes depending on the level of hidden layer.

Also, hidden nodes are connected with the outgoing arcs to output nodes or other hidden nodes depending on the level of hidden layer.

Output nodes:

Hidden nodes give input to output nodes by using the connections of outgoing arcs.

Output nodes returns an output value that corresponds the prediction of the response variable.

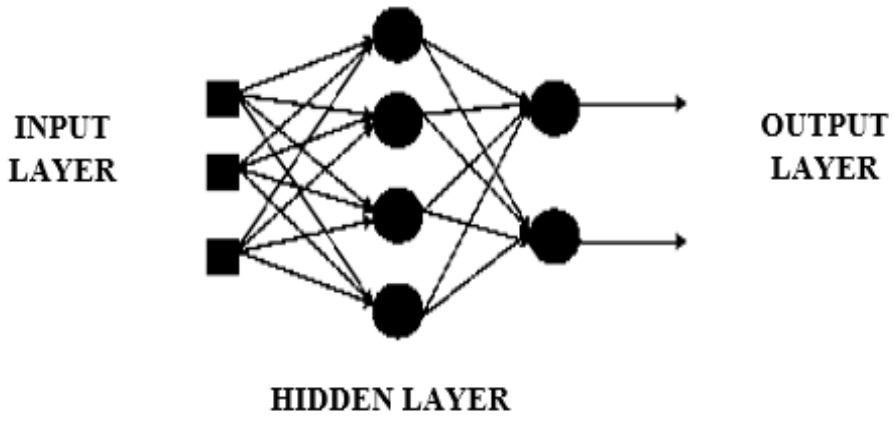


fig 7.6 Multi-layer feed-forward network

- Recurrent Neural Network

  The networks which are with closed loop are called as recurrent network. When output is directed back as inputs to the same layer or preceding layer nodes, then it results in feedback networks.

  This type of neural network is used in solving problems where the solution depends not just on current input but also on past inputs.

  An auto-associative network is equivalent to a neural network that contains feedback.

  Recurrent neural networks can be trained by using backpropagation algorithm.

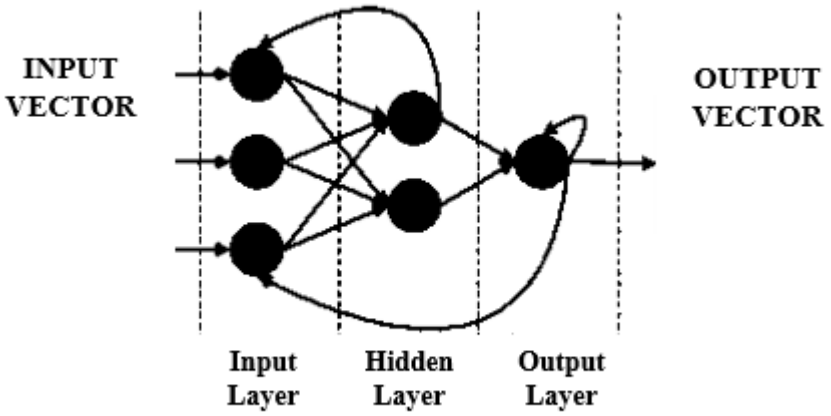


fig 7.7 Recurrent neural network

7.4.2 Learning Rules

It is a methodology by which neural network learns from existing conditions and increases the performance. It is a continuous process.

There are various learning rules like:

7.4.2.1 Hebbian learning rule

It is used to identify, how to modify the weights of nodes of a network.

Hebbian rule was the first law of learning. In 1949 Donald Hebb developed it as a learning algorithm for the unsupervised neural network. We can use it to discover a way to enhance the weight of network nodes.

The Hebb learning rule assumes - When two neighbouring neurons are activated and deactivated at the same time. Then the weight connecting these nodes should increase. For neurons working inside the opposite section, the weight among them have to decrease. If there is no signal correlation, the weight should not change.

If the input of both nodes is positive or negative, then there is strong positive weight present between the nodes. When a node input is positive and negative in another, a strong negative weight is present between the nodes.

Initially, the values of all weights are set to zero. This learning rule can be applied to both soft and hard work. Since the desired responses of neurons can be used in the learning process, this is an unsupervised learning law. Total weight values are generally equal to the duration of the study, which is not required.

Mathematical formula of Hebb Learning Rule in ANN

$$W_{ij} = x_i * x_j$$

7.4.2.2 Perceptron learning rule

In this network begins its learning process by assigning a random value to each weight.

In a neural network each node is associated to weight that changes in the course of learning. According to it, an instance of supervised learning, the network begins its learning through assigning a random value to every weight.
Calculate the output value on the premise of a set of information for which we are able to understand the anticipated output value. that is the learning pattern that suggests the complete definition. As a result, it is referred to as a learning pattern. The network then compares the calculated output value with the anticipated value. next calculates an error function ∈, which may be the sum of squares of the errors happening for every character inside the learning pattern.

Mathematical formula of Perceptron Learning Rule in ANN

$$\sum_i \sum_j \left(E_{ij} - O_{ij} \right)^2$$

Carry out the first summation on the people of the learning sample set, and carry out the second summation at the output units. Eij and Oij are the anticipated and obtained values of the jth unit for the ith individual.

The network then adjusts the weights of the distinctive units, checking every time to peer if the mistake function has expanded or reduced. As in a traditional regression, that is a matter of solving a problem of least squares.

Considering that assigning the weights of nodes consistent with users, it's an instance of supervised learning.

7.4.2.3 Delta learning rule

In this modification in sympatric weight of a node is equal to the multiplication of error and the input.

It is the most common learning algorithm which was developed by Widrow and Hoff. It relies upon supervised learning. This rule states that the change in sympatric weight of a node is identical to the multiplication of error and the input.

Mathematical formula of Delta Learning Rule in ANN

$$\Delta w = \eta \, (t - y) \, x_i$$

For a given input vector, examine the output vector is the precise solution. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from ui to uj is: dwij = r* ai * ej.

wherein r is the learning rate, ai represents the activation of ui and ej is the difference between the anticipated output and the actual output of uj. If the set of input patterns form an impartial set, then arbitrary associations can be learnt using the delta rule.

It has seen that for networks with linear activation functions and without hidden units. The error squared vs. the weight graph is a paraboloid in n-space. As the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point wherein it reduces the error. The weight vector corresponding to this point is then the ideal weight vector.

We can use the delta learning rule with both single output unit and several output units. While applying the delta rule assume that the error can be directly measured. The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

7.4.2.4 Correlation learning rule

The correlation rule is majorly used in supervised learning, it is based on the principle of Hebbian rule.

The correlation learning rule assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative.

Contrary to the Hebbian rule, the correlation rule is the supervised learning.

The response, oj, the desired response, dj, uses for the weight-change calculation.

Mathematical formula of Correlation Learning Rule in ANN

$$\Delta w_{ij} = \eta x_i d_j$$

Here dj is the anticipated value of response signal. This training algorithm generally starts with weights value initialized to zero.

7.4.2.5 Out Star learning rule

The out-star learning rule is used when we assume that nodes or neurons in a network arranged in a layer. It is a supervised learning process because desired outputs must be known.

Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights. The out-star rule produces the desired response t for the layer of n nodes.

Apply this type of learning for all nodes in a particular layer. Update the weights for nodes are as in Kohonen neural networks. The Kohonen neural network is an unsupervised network used for clustering.

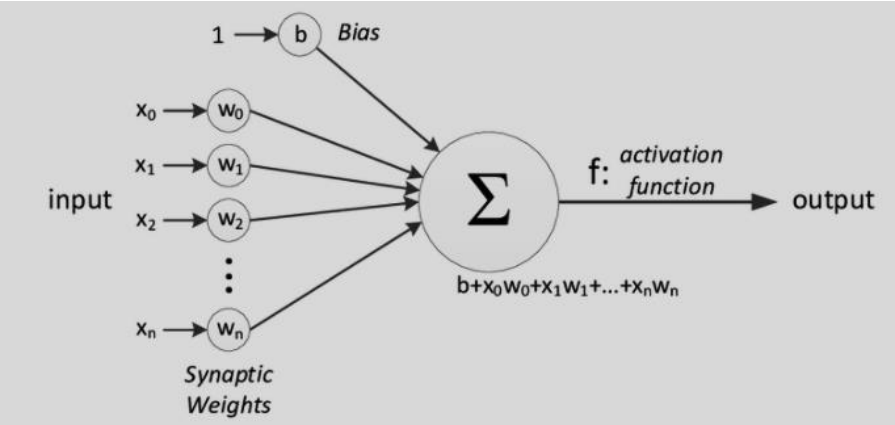
Mathematical formula of Out Star Learning Rule in ANN

$$w_{jk} = \begin{cases} \eta(y_k - w_{jk}) & \text{if node j wins the competition} \\ 0 & \text{if node j losses the competition} \end{cases}$$

7.4.3 Activation function

The general model of ANN is inspired by biological neuron. The single layer neuron is also called as Perceptron. It is meant to give single output.

Diagrammatic representation on Rosenblatt Perceptron



In the above figure, for one single observation, x_0, x_1, x_2, x_3...$x_{(n)}$ represents multuiple inputs of independent variables to the network. Each of these inputs is multiplied by a connection weight or synapse. The weights are represented as w_0, w_1, w_2, w_3….$w_{(n)}$. Weight shows the strength of a particular node, b is a bias value. A bias value allows you to shift the activation function up or down.

In the simplest case, these products are summed, fed to a transfer function called as activation function to generate a result, and this result is sent as output.

Mathematically, $x_1.w_1 + x_2.w_2 + x_3.w_3$ ..... $x_n.w_n = \sum x_i.w_i$

Now activation function is applied $\phi(\sum x_i.w_i)$

Activation function

The main purpose of an activation function is to convert an input signal of a node in an Artificial neural network to output signal. This output signal is used as input for the next layers in the network.

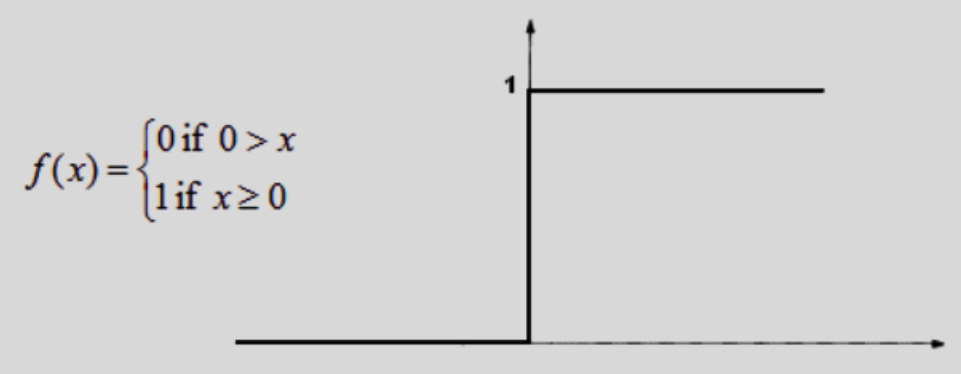
Activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The motive is to introduce non-linearity into the output of a neuron.

If we do not apply activation function then the output signal would be definitely linear function. Now, a linear function is easy to solve but they are constrained in their complexity, have less power. Without activation function, our model cannot learn and model complex data such as images, videos, audio, speech, etc.

Types of Activation Functions:

7.4.3.1 Threshold Activation Function — (Binary step function)

A Binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and sends exactly the same signal to the next layer.



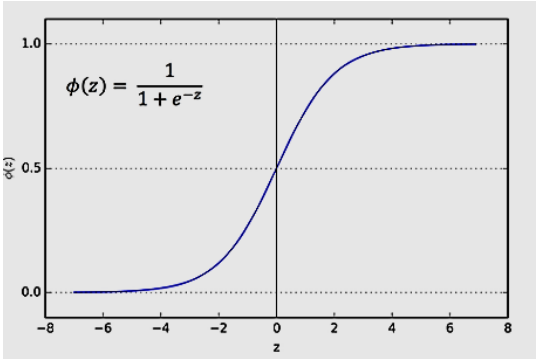
Activation function A = “activated” if Y > threshold

else not or A=1 if y > threshold, 0 otherwise.

The problem with this function is for creating a binary classifier (1 or 0), but if you want multiple such neurons to be connected to bring in more classes, Class1, Class2, Class3, etc. In this case, all neurons will give 1, so we cannot decide.

7.4.3.2 Sigmoid Activation Function — (Logistic function)

A Sigmoid function is a mathematical function having a characteristic “S”-shaped curve or sigmoid curve which ranges between 0 and 1, therefore it is used for models where we need to predict the probability as an output.



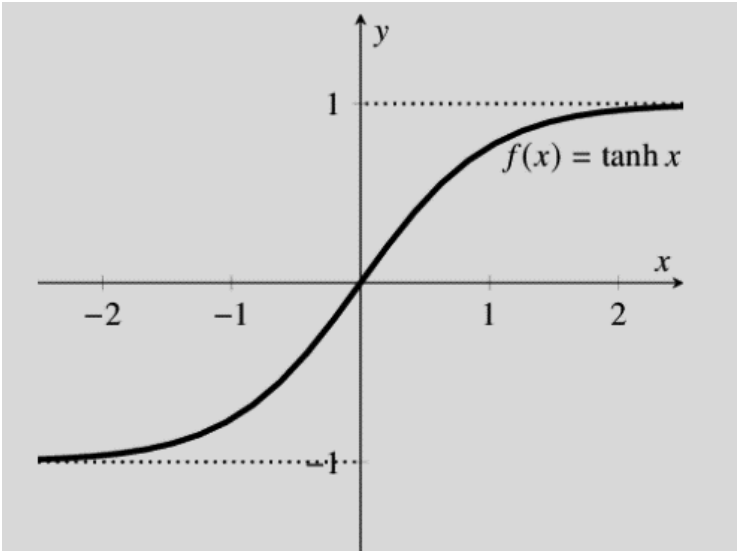
The Sigmoid function is differentiable, means we can find the slope of the curve at any two points.

The drawback of the Sigmoid activation function is that it can cause the neural network to get stuck at training time if strong negative input is provided.

7.4.3.3 Hyperbolic Tangent Function — (tanh)

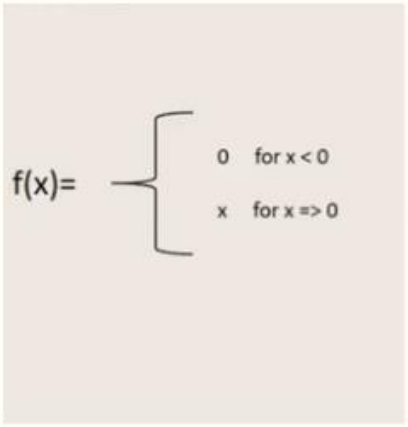
It is similar to Sigmoid but better in performance. It is nonlinear in nature, so great we can stack layers. The function ranges between (-1,1).

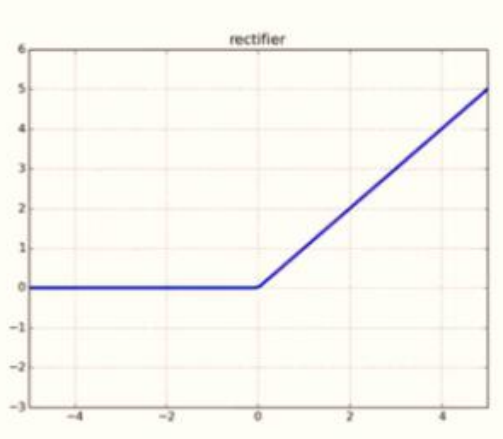
The main advantage of this function is that strong negative inputs will be mapped to negative output and only zero-valued inputs are mapped to near-zero outputs. So less likely to get stuck during training.



7.4.3.4 Rectified Linear Units — (ReLu)

ReLu is the most used activation function in CNN and ANN which ranges from zero to infinity. (0, ∞)





It gives an output ‘x’ if x is positive and 0 otherwise. It looks like having the same problem of linear function as it is linear in the positive axis.

Relu is non-linear in nature and a combination of ReLu is also non-linear.

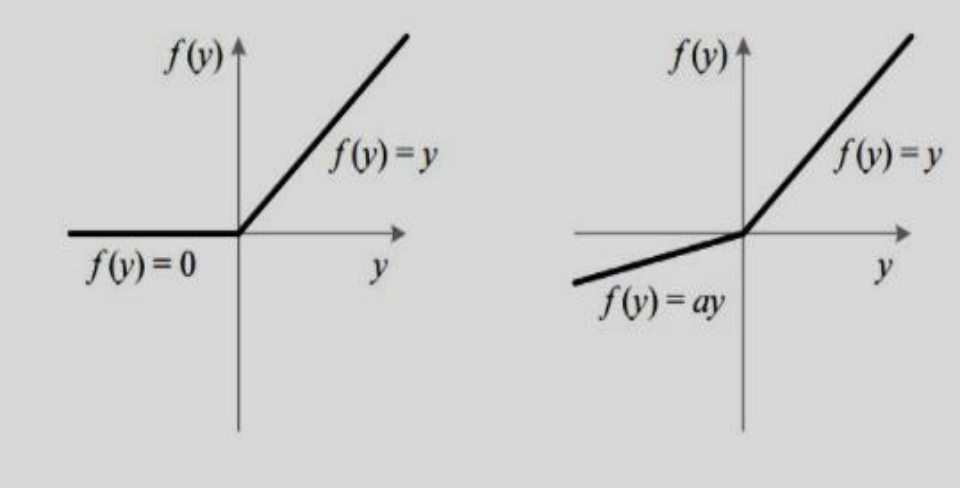
In fact, it is a good approximator and any function can be approximated with a combination of Relu.

ReLu is 6 times improved over hyperbolic tangent function. It should only be applied to hidden layers of a neural network. So, for the output layer use softmax function for classification problem and for regression problem use a Linear function.

Here one problem is some gradients are fragile during training and can die. It causes a weight update which will make it never activate on any data point again.

Basically, ReLu could result in dead neurons.

To fix the problem of dying neurons, Leaky ReLu was introduced. So, Leaky ReLu introduces a small slope to keep the updates alive. Leaky ReLu ranges from -∞ to +∞.



Leak helps to increase the range of the ReLu function. Usually, the value of a = 0.01 or so.

When a is not 0.01, then it is called Randomized ReLu

7.5 LIST OF REFERENCES

- Kumar Satish ,Neural Networks, Second edition Tata McGraw Hill-, 2013, ISBN1259006166, 9781259006166
- Neural Networks and Deep Learning: A Textbook 1st ed. 2018 Edition ISBN-13: 978-3319944623 ISBN-10: 3319944622
- Artificial Neural Networks, Robert J. Schalkoff, Tata McGraw Hil , ISBN: 9780071155540, 0071155546

7.6 QUIZ

1. What is a perceptron?
   a) a neural network that contains no loops
   b) a neural network that has only one loop
   c) a single layer feed-forward neural network with pre-processing
   d) a neural network that contains feedback
2. What are the advantages of neural networks over conventional computers?
   i. They have ability to learn by example
   ii. They are fault tolerant
   iii. They are more suited for real time operation due to their high computational rates
   a) i and ii are true
   b) i and iii are true
   c) only i is true
   d) i, ii and iii are true
3. Which network involves backward links from output to the input and hidden layers ?
   a) Perceptron
   b) Multi layered perceptron
   c) Self-organizing maps
   d) Recurrent neural networks
4. Which of the following is not a component of learning systems?
   a) Goal
   b) Model
   c) Experience
   d) **Trees**
5. In artificial Neural Network interconnected processing elements are called
   a) nodes or neurons
   b) weights
   c) axons
   d) cell body
6. Each connection link in ANN is associated with _____ which has information about the input signal.
   a) nodes or neurons
   b) weights
   c) bias
   d) activation function

7. Information from other neurons, in the form of electrical impulses, enters the dendrites at connection points called_____

   a) Axons
   b) Synapse
   c) Dendrites
   d) Weights

8. In FeedForward ANN, information flow is _____

   a) unidirectional
   b) Bidirectional
   c) Multidirectional
   d) polydirectional

9. What was the name of the first model which can perform weighted sum of inputs?

   a) Papert neuron model
   b) **McCulloch-pitts neuron model**
   c) Marvin Minsky neuron model
   d) Hopfield model of neuron

10. Who proposed the first perceptron model in 1958?

    a) McCulloch-pitts
    b) Marvin Minsky
    c) Hopfield
    d) Rosenblatt

11. What is R in RNN?

    a) Recurrent
    b) Recovered
    c) Repetitive
    d) Reversed

12. _____takes a real-valued input and squashes it to the range.

    a) Sigmoid
    b) tanh
    c) ReLU
    d) Network

13. Correlation learning law is special case of?

    a) Hebb learning rule
    b) Perceptron learning rule
    c) Delta learning rule
    d) LMS learning rule

14. Which learning rule was developed by Widrow and Hoff?

    a) Hebb learning rule
    b) Perceptron learning rule
    c) Delta learning rule
    d) Out star learning rule

15. What is an auto-associative network?

    a) a neural network that contains no loops
    b) a neural network that has only one loop
    c) a single layer feed-forward neural network with pre-processing
    d) **a neural network that contains feedback**

7.7 EXERCISE

1. What is Artificial neural network?
2. Compare similarities and differences between BNN & ANN.
3. What are the basic models of ANN?
4. Explain models classified based on interconnections with diagram
5. Explain learning models
6. Explain activation function
7. What is Rosenblatt perceptron?
8. Short note on Recurrent neural network

7.8 VIDEO LINKS

1. (1857) Neural Network Full Course | Neural Network Tutorial For Beginners | Neural Networks | Simplilearn - YouTube
2. (1857) Biological and Artificial Neural Network | Basic Concepts | Neural Networks - YouTube
3. (1857) Types of Artificial Neural Network(ANN) Architectures - YouTube
4. (1863) Perceptron learning algorithm with solved example (in Hindi) - YouTube
5. (1863) Neural Network Learning Rules - Part 2 - Perceptron, Delta, Widrow-Hoff - YouTube
6. (1863) Lecture 14: Widrow-Hoff Learning Rule | ANN - YouTube
7. (1863) Delta Learning Rule | Application of Soft Computing | Lecture Series - YouTube

❄❄❄❄❄❄❄

8

SUPERVISED & UNSUPERVISED LEARNING NETWORKS

Unit Structure

8.0 INTRODUCTION

Artificial neural network is a complex adaptive system which is changing itself based on data it is processing. Machines can learn how to perform tasks better with experience. We can define learning as art of being able to perform at a given task, or a range of tasks with experience in a better way.

Neural networks help to develop this adaptation skill by training with the help of past evaluations and solutions over multiple problems.

8.0.1 What is Learning?

Learning is the ability to adapt with the changes in the environment whenever any new data is processed or event is evaluated.

ANN can model learning process by adjusting the weighted connections or arcs found between neurons or nodes in the network.

Learning algorithms are very useful when it comes to few problems that cannot be written practically.

Learning Algorithms

Define an architecture-dependent procedure to encode pattern information into weights. Learning proceeds by modifying connection strengths.

Learning is data driven:

A set of input–output patterns derived from a (possibly unknown) probability distribution.

- Output pattern might specify a desired system response for a given input pattern
- Learning involves approximating the unknown function as described by the given data

  Alternatively, the data might comprise patterns that naturally cluster into some number of unknown classes
- Learning problem involves generating a suitable classification of the samples.

8.0.2 Types of Learning Process

8.0.2.1 Supervised Learning

Throughout the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is compared with the preferred output vector. An error signal is generated, if there is a difference between the actual output and the preferred output vector. On the basis of this error signal, the weights are adjusted till the actual output is matched with the desired output.

Supervised learning happens in the presence of a supervisor. An output or target attribute of supervised learning is already known. Machine is fed with multiple input data and then the model is built to predict the output.

It is used as fast learning mechanisms with better accuracy. It includes regression & classification.

8.0.2.2 Unsupervised Learning

Unsupervised learning happens without the help of supervisor. It is having an independent learning process. No output is mapped with the input as we are not aware of about the output variable that we are going to achieve after evaluation.

In unsupervised learning the neural network is responsible to find the patterns from the input provided to the model use for predicting the preferences or similarities. It is using unlabelled dataset for finding hidden patterns. It includes Clustering and Association rule. It is descriptive in nature.

Unsupervised learning provides the system with an input X_k, and allow it to self-organize its weights to generate internal prototypes of sample vectors.

There is no teaching input involved here. The system attempts to represent the entire data set by employing a small number of prototypical vectors which are enough to allow the system to retain a desired level of discrimination between samples. As new samples continuously buffer the system, the prototypes will be in a state of constant flux. This kind of learning is often called adaptive vector quantization.

8.0.2.3 Reinforcement Learning

Reinforcement learning, learns from feedback on past experiences. It is a long-term iterative process. If more feedback is obtained then more learning has been performed.

Reinforcement learning is similar to supervised learning. The aim of reinforcement learning is to maximize the reward of the system through a trial and error. This is strongly with how learning works in nature. Remembering the actions performed in the past and taking decisions based on those actions will lead to gain more rewards. It is also called as Markov Decision Process.

It is more accurate system.

8.0.3 Supervised Learning Network

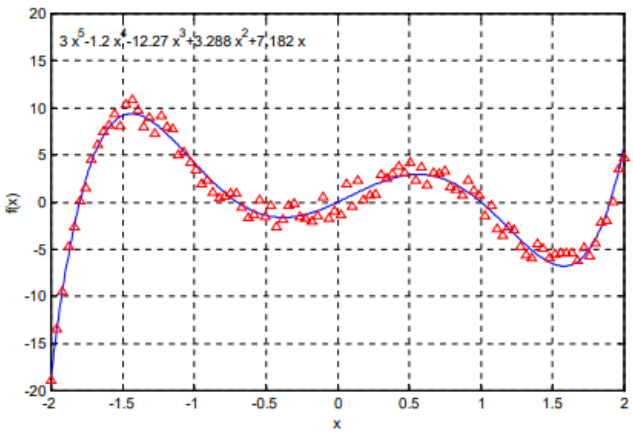
In supervised learning network we get labelled input data. Data comprises a set of discrete samples drawn from the pattern space where each sample relates an input vector $X_k \in R^n$ to an output vector $D_k \in R^p$.

$$T = \{(X_k, D_k)\}_{k=1}^{Q}$$

- The set of samples describe the behaviour of an unknown function

  $f : R^n \rightarrow R^p$ which is to be characterized.

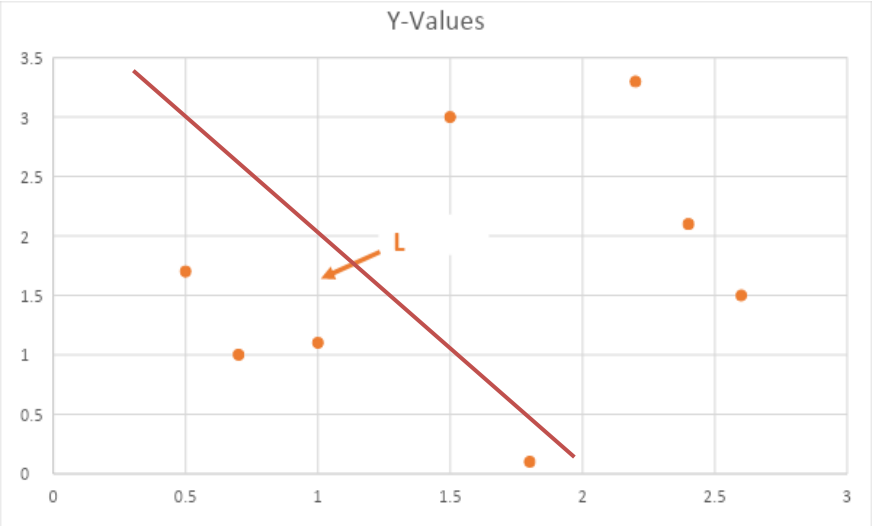
An example function described by a set of noisy data points



8.1 Linear Separability

Linear Separability refers to the fact that classes of patterns with 'n' dimensional vector $X=(x_1,x_2,\ldots..,x_n)$ can be separated with single decision surface. When two classes of patterns that can be separated by decision boundary represented by linear equation. then they are referred to as Linearly separable.

Below graph shows Linearly separable pattern which separates data or classes with the help of single line 'L'



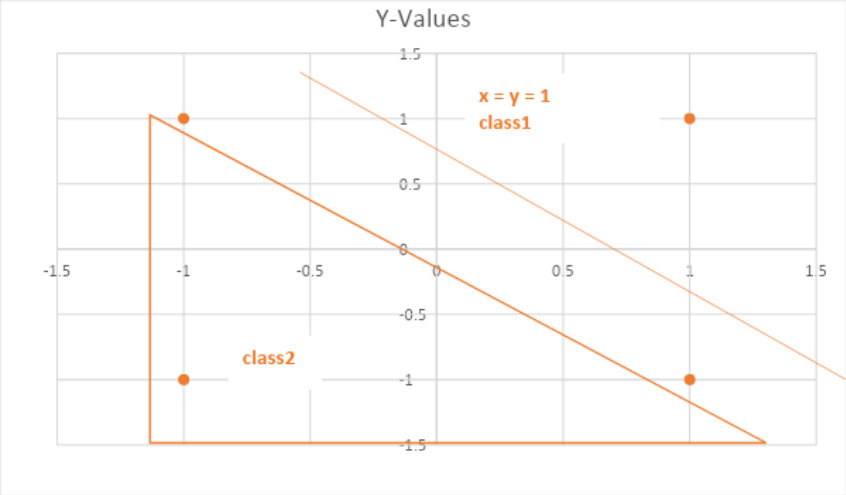
- AND Problem

  As per truth table of AND we can have two classes; class1 will hold values when output is 1 i.e y=1 and class2 will hold values when output is -1 i.e y=-1

i/p	i/p	o/p
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

Graph to evaluate whether the values are linearly separable or not.

If we consider a class1 it lies in positive quadrant and class2 consist of remaining three quadrants that very clearly can be separated by a single line.



Hence, we can say that AND Problem is linearly separable.

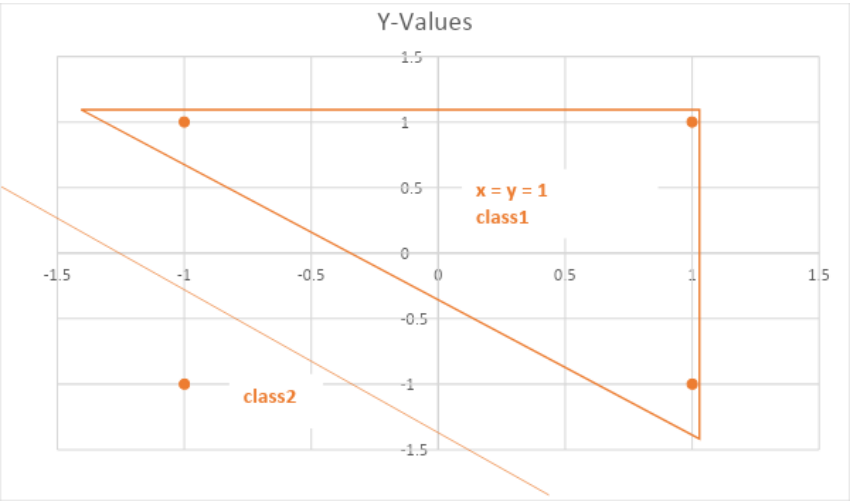
- OR Problem

  As per truth table of OR we can have two classes; class1 will hold values when output is 1 i.e y=1 and class2 will hold values when output is -1 i.e y=-1

i/p	i/p	o/p
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

Graph to evaluate whether the values are linearly separable or not.

If we consider a class1 it lies in quadrant 1,2,4 and class2 consist of quadrant 3 hence we can separate data using single line



Hence, we can say that OR Problem is linearly separable

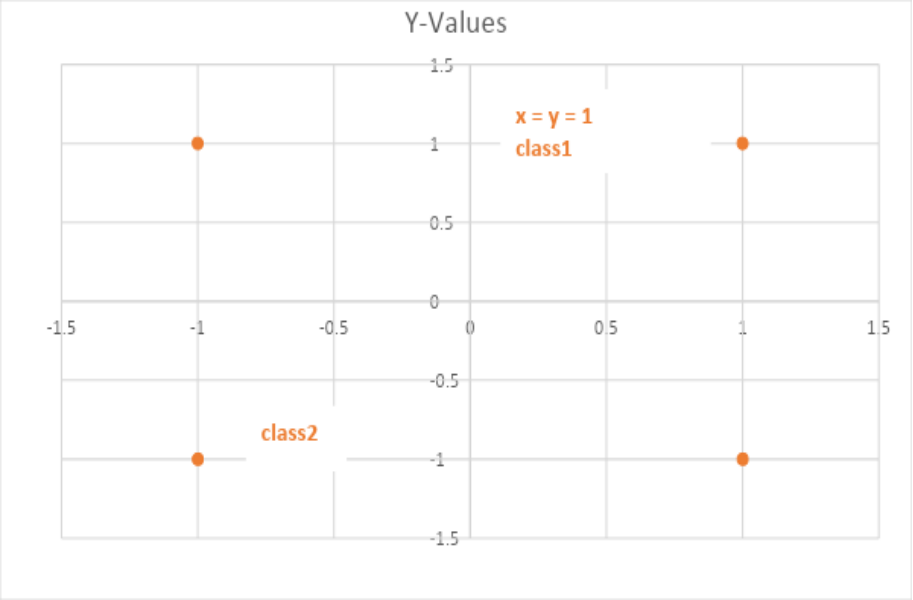
- XOR Problem

  As per truth table of XOR we can have two classes; class1 will hold values when output is 1 i.e y=1 and class2 will hold values when output is -1 i.e y=-1

i/p	i/p	o/p
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Graph to evaluate whether the values are linearly separable or not.

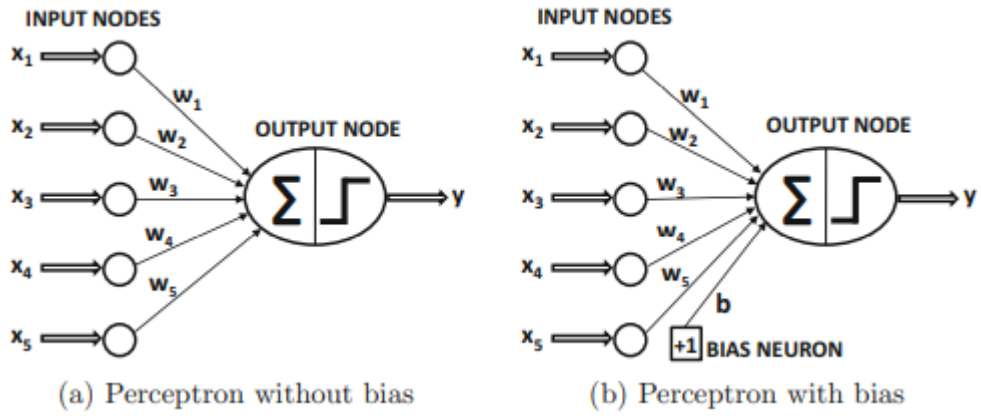
If we consider a class1 it lies in quadrant 1,3 and class2 consist of quadrant 2,4 hence we cannot separate data using single line



Hence, we can say that XOR Problem is not linearly separable

8.2 PERCEPTRON NETWORKS

In the single layer network, a set of inputs is directly mapped to an output by using a generalized variation of a linear function. This simple instantiation of a neural network is also referred to as the perceptron. This neural network contains a single input layer and an output node. The basic architecture of the perceptron is shown below:



Architecture of the perceptron

Components of Perceptron

- Input: Features are taken as inputs in the perceptron algorithm. Inputs are denoted as x_1, x_2, x_3, x_4, .x_n – 'x' in these inputs suggests the characteristic value and 'n' the entire occurrences of those features. There is also a unique input type, which is referred to as bias.
- Weights: These are values that are calculated during the training of the model. The weights are given a preliminary value in the beginning. With every occurrence of a training error, the values of weights are modified. Weights are represented as w_1, w_2, w_3, w_4, ..w_n.
- Bias: A bias is a special input type which allows the classifier to move the decision boundary around from its original position to the right, left, up, or down. The objective of the bias is to shift every point in a specific path for a specified distance. Bias allows for higher quality and quicker model training. Perceptron algorithms can be categorized into single-layer and multi-layer perceptron. The single-layer type arranges neurons in a single layer while the multi-layer type arranges neurons in multiple layers. In case of multi-layer, each neuron of the first layer picks inputs and provides a response to the group of neurons present in the second layer. This process continues until the final layer.
- Activation/step function: Activation or step functions are used to create non-linear neural networks. These functions can change the value of neural networks to 0 or 1. The conversion of value is done to make a data set easy to classify. We can use the step function depending on the value required. Sigmoid function and sign functions can be used for values between 0 and 1 and 1 and -1, respectively. The sign function is a hyperbolic tangent function which is ideal for multi-layer neural networks. Rectified linear unit (ReLu) is another step function which can be used for values approaching zero – value more less than or more than zero. However, linear classification requires the perceptron to be linear.

- Weighted summation: The multiplication of each feature or input value (x_n) associated with related weight values (w_n) gives us a sum of values that are called weighted summation. Weighted summation is represented as $\sum w_i x_i$ for all i -> [1 to n].

The input layer contains d nodes that transmit the d features $\overline{X} = [x_1 \ldots x_d]$ with edges of weight $\overline{W} = [w_1 \ldots w_d]$ to an output node. The input layer does not perform any computation in its own right. The linear function $\overline{W} \cdot \overline{X} = \sum_{i=1}^{d} w_i x_i$ is computed at the output node. Subsequently, the sign of this real value is used in order to predict the dependent variable of $\overline{X}$. Therefore, the prediction $\hat{y}$ is computed as follows:

$$\hat{y} = \text{sign}\{\overline{W} \cdot \overline{X}\} = \text{sign}\{\sum_{j-1}^{d} w_j x_j\}$$

The sign function maps a real value to either +1 or −1, which is appropriate for binary classification.

The error of the prediction is therefore $E(\overline{X}) = y - \hat{y}$, which is one of the values drawn from the set {−2, 0, +2}. In cases where the error value $E(\overline{X})$ is nonzero, the weights in the neural network need to be updated in the (negative) direction of the error gradient. As we will see later, this process is similar to that used in various types of linear models in machine learning.

In many settings, there is an invariant part of the prediction, which is referred to as the bias. For example, consider a setting in which the feature variables are mean centered, but the mean of the binary class prediction from {−1, +1} is not 0. This will tend to occur in situations in which the binary class distribution is highly imbalanced. We need to incorporate an additional bias variable b that captures this invariant part of the prediction. The bias can be incorporated as the weight of an edge by using a bias neuron. This is achieved by adding a neuron that always transmits a value of 1 to the output node. The weight of the edge connecting the bias neuron to the output node provides the bias variable.

$$\hat{y} = \text{sign}\{\overline{W} \cdot \overline{X} + b\} = \text{sign}\{\sum_{j=1}^{d} w_j x_j + b\}$$

Algorithm:

Step 1 - Initialize weight and bias to 0. Set learning rate α between 0 to 1.

Step 2 - Feed the features of the model that is required to be trained as input in the first layer.

Step 3 - Product of all weights and inputs will be added up.

Step 4 - The Bias value will be added to shift the output function

Step 5 - This value will be presented to the activation function

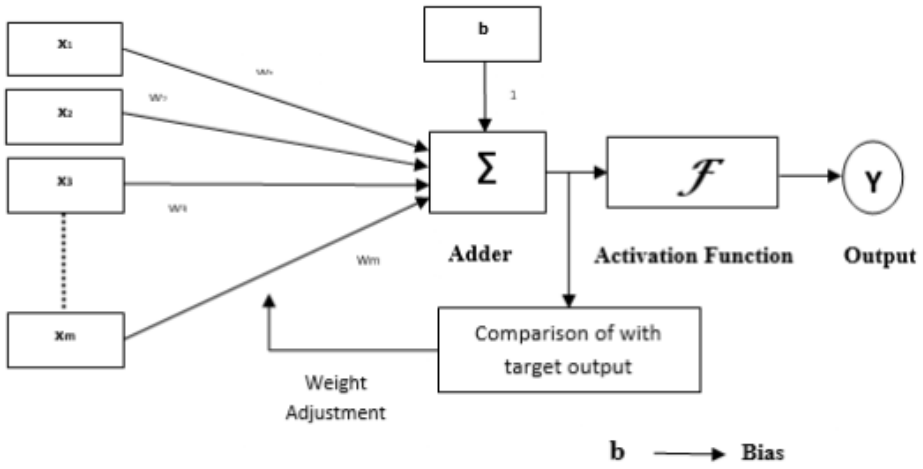
Step 6 - The value received after the last step is the output value.

8.3 ADAPTIVE LINEAR NEURON (ADALINE)

Adaline is also referred to as the Widrow-Hoff learning rule. The units with linear activation function are called as linear unit and a network with single linear unit is called as Adaline. The relationship between input and output is linear. Adaline makes a use of bipolar activation function for its input and output signals. It has only one output. The delta rule is used for weight adjustment and it is given as

$$\Delta W_{ij} = \alpha (t_j - y_{in}) x_i$$

We can train the network to have minimum error by using the Least Mean Squares method.



The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired target output. After comparison on the basis of training algorithm, the weights and bias will be updated.

Algorithm

Step 1 - Initialize weight and bias to some random value but not 0. Set learning rate α between 0 to 1.

Step 2 - Calculate the net input to the output unit

$$y_{in} = b + \sum_{i=1}^{n} wixi$$

here,
b = bias
w = weights
x = inputs
y = output

Step 3 - Keep updating weights and bias for i = 1 till n, until least mean square is achieved, $t-y_{in}$

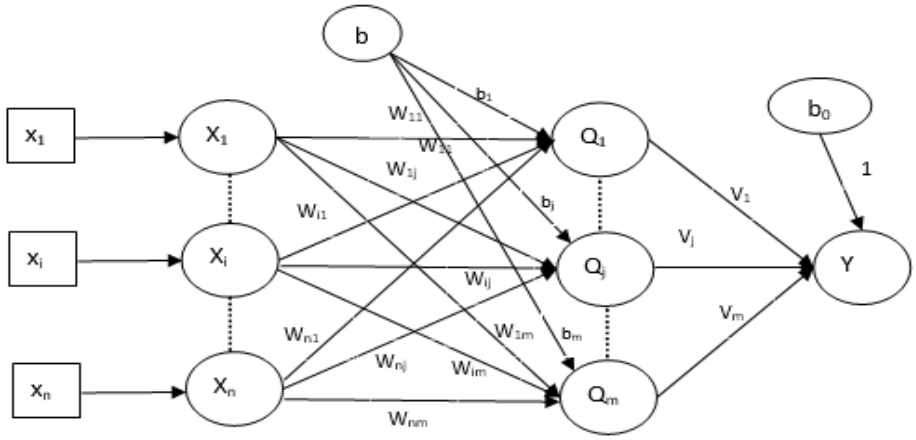
$$w_{i(new)} = w_{i(old)} + \alpha(t-y_{in})\, x_i$$

$$b_{(new)} = b_{(old)} + \alpha(t-y_{in})$$

here,
w is weight
b is bias
x_i is input
α is learning rate
t is desired output
y_{in} is net input to output neuron

8.4 MULTIPLE ADAPTIVE LINEAR NEURONS (MADALINE)

It is a network which consists of many Adaline networks in parallel. It will be having single output unit. It acts like multilinear perceptron in which hidden unit between input and Madaline layer will be Adaline. The weights and the bias between the input and the Adaline layer is adjustable. Both Adaline & Madaline layers have a fixed bias as 1.



The architecture of Madaline consists of "n" neurons of the input layer, "m" neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.

Algorithm

Step 1 - Initialize weight and bias to 0. Set learning rate α between 0 to 1 preferably 1

Step 2 - When stopping criteria is false go to step 3 to 8

Step 3 - For each bipolar training pair s.t do step 4 to 7

Step 4 - Set activation of input unit xi = si (i =1 to n)

Step 5 - Calculate the net input to the output unit

$$Q_{inj} = b_j + \sum_{i=1}^{n} wixi \quad (j = 1 \text{ to } m)$$

here,

b = bias

w = weights

x = inputs

y = output

n = total number of input neurons

Step 6 - Apply the activation function to obtain the final output at the Adaline and Madaline layer

$$f(x) = \begin{cases} 1 & if\ x \geqslant 0 \\ -1 & if\ x < 0 \end{cases}$$

Output at the hidden unit

$$Q_j = f(Q_{inj})$$

Final output of the network

$$y = f(y_{in})$$

$$\text{i.e. } y_{inj} = b + \sum_{j=1}^{m} Qjvj$$

$$\sum_{j=1}^{m} Qjvj$$

Step 7 - Calculate the error and adjust the weights as follows –

Case 1 − if **y ≠ t** and **t = 1** then,

$$\mathbf{w_{ij(new)} = w_{ij(old)} + \alpha(1 - Q_{inj})\, x_i}$$

$$\mathbf{b_{j(new)} = b_{j(old)} + \alpha(1 - Q_{inj})}$$

here,

w is weight

b is bias

x_i is input

α is learning rate

y_{inj} is net input to output neuron

In this case, the weights would be updated on **Q_j** where the net input is close to 0 because **t = 1**.

Case 2 − if **y ≠ t** and **t = -1** then,

$$\mathbf{w_{ik(new)} = w_{ik(old)} + \alpha(-1 - Q_{ink})\, x_i}$$

$$\mathbf{b_{k(new)} = b_{k(old)} + \alpha(-1 - Q_{ink})}$$

In this case, the weights would be updated on **Q_k** where the net input is positive because **t = -1**.

Here **'y'** is the actual output and **'t'** is the desired/target output.

Case 3 − if **y = t** then

There would be no change in weights.

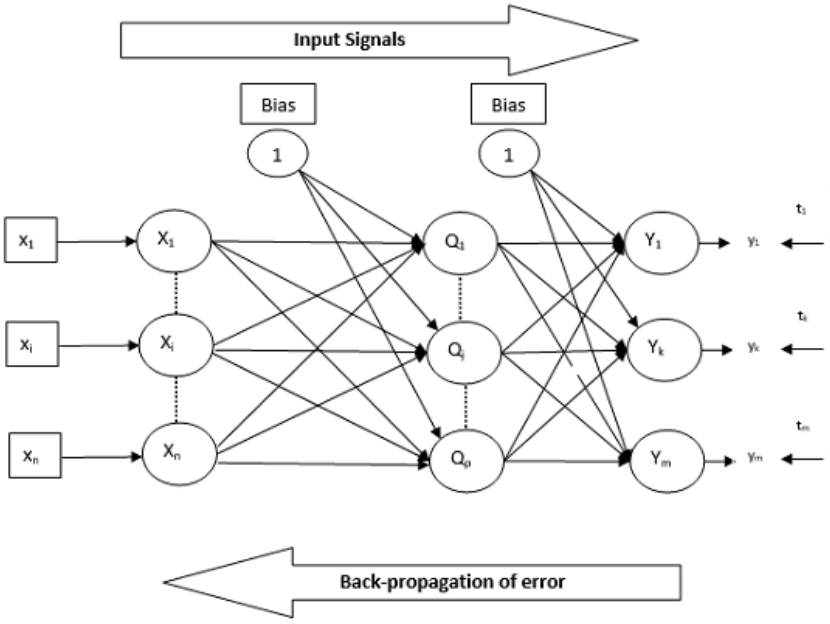
Step 8 - Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

8.5 BACK-PROPAGATION NETWORK

The backpropagation algorithm is a supervised learning algorithm for training neural networks. The backpropagation algorithm is one of the algorithms responsible for updating network weights with the objective of reducing the network error.

The principle behind the back propagation algorithm is to reduce errors of randomly allocated weights and biases. We need to update the weights such that we get the global loss minimum.

The most important features of backpropagation algorithm are that it is recursive, iterative and efficient method.



Gradient Descent

We can update weights by using gradient descent. It is used for finding the minimum of a function.

Types of Back propagation network

- Static back propagation

It deals with static input and static output. The mapping is faster in static back propagation.

- Recurrent back propagation

In it network is fed until a fixed value is obtained. Later the error is calculated and propagated backward. The mapping is slower in recurrent back propagation.

As we know in artificial neural networks, training occurs in various steps:

Step 1 - Initialization.
Step 2 - Forward propagation.
Step 3 - Error Function.
Step 4 - Backpropagation.
Step 5 - Weight Update.
Step 6 - Iteration.

It is the fourth step of the process, a backpropagation algorithm that calculates the gradient of a loss function of the weights in the neural network to ensure the error function is minimum.

The backpropagation algorithm accomplishes this through a set of Back Propagation Algorithm Steps, which involves:

Step 1 - **Selecting Input & Output:** To choose an input for the process and to set the desired output.

Step 2 - **Setting Random Weights:** Once the input and output are set, random weights are allocated, as it will be needed to manipulate the input and output values. Later, the output of each neuron is calculated through the forward propagation, which goes through:

- Input Layer
- Hidden Layer
- Output Layer

Step 3 - **Error Calculation:** This is an important step that calculates the total error by determining how far and suitable the actual output is from the required output. This is done by calculating the errors at the output neuron.

Step 4 - **Error Minimization:** Based on the observations made in the earlier step, here the focus is on minimizing the error rate to ensure accurate output is delivered.

Step 5 - **Updating Weights & other Parameters:** If the error rate is high, then parameters (weights and biases) are changed and updated to reduce the rate of error using the delta rule or gradient descent. This is accomplished by assuming a suitable learning rate and propagating backward from the output layer to the previous layer. Acting as an example of dynamic programming, this helps avoid redundant calculations of repeated errors, neurons, and layers.

Step 6 - **Modelling Prediction Readiness:** Finally, once the error is optimized, the output is tested with some testing inputs to get the desired result.
This process is repeated until the error reduces to a minimum and the desired output is obtained.

Advantages of the backpropagation algorithm:

- It's memory-efficient in calculating the derivatives, as it uses less memory compared to other optimization algorithms.
- The backpropagation algorithm is fast, especially for small and medium-sized networks. As more layers and neurons are added, it starts to get slower as more derivatives are calculated.
- This algorithm is generic so it can be used to work with different network architectures, like convolutional neural networks, generative adversarial networks, fully-connected networks, and more.
- There are no parameters to tune the backpropagation algorithm, so there's less overhead. The only parameters in the process are related to the gradient descent algorithm, like learning rate.

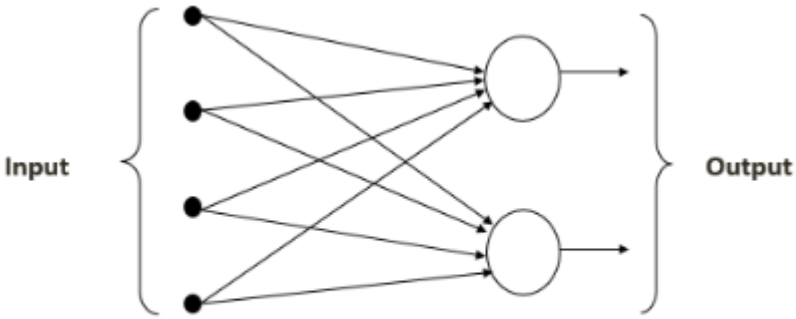
8.6 UNSUPERVISED LEARNING NETWORKS

Unsupervised learning provides the system with an input X_k, and allow it to self-organize its weights to generate internal prototypes of sample vectors. There is no supervisor involved here. The system attempts to represent the entire data set by employing a small number of prototypical vectors- enough to allow the system to retain a desired level of discrimination between samples. As new samples continuously buffer the system, the prototypes will be in a state of constant flux. This kind of learning is often called adaptive vector quantization.

Local information is used for learning i.e unlabelled data and this allows synapses to learn in real time.

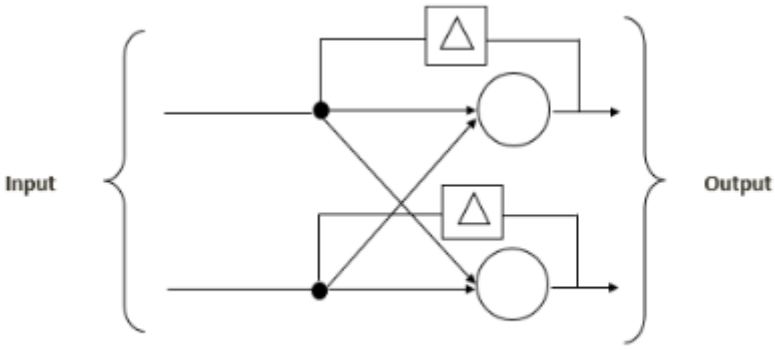
8.6.1 MaxNet

MaxNet is a type of competitive learning network where one of the competing neurons has a non-zero output. Before understanding MaxNet we need to know about Hamming neural network. The output of hamming neural network will be given as input to MaxNet. It is a feed backward neural network with single layer. It has fixed weights in it. It takes inputs and cluster them.



Hamming Neural Network

It is a feed backward neural network with single layer. It has fixed weights in it. It depends on Winner-Take-All WTA policy.



MAXNET Neural Network

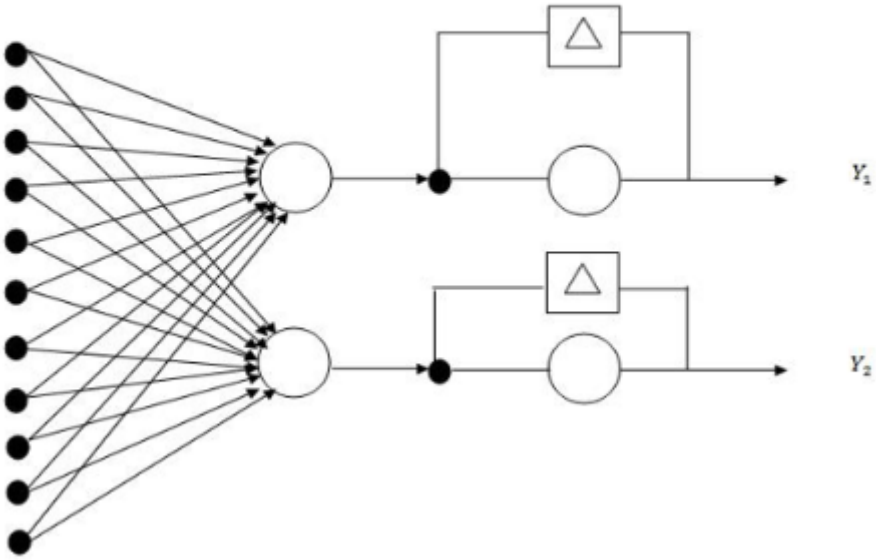
An example of TB test:

A simple test for Tuberculosis disease is carried out. TP has 11 symptoms from which 6 come together for an infected person. Each input or a case of the test is a vector of 11 values. Two examples in the test, one of for the infected person and the other one for the normal person. Set the value of 1 for the existing symptoms in the input vector, otherwise -1. Anybody has 6 known symptoms like cough, fever, chest pain, coughing up blood, fatigue, night sweating or more will be classified as an infected person. Any case with less than six known symptoms will be classified as normal person, but he or she may suffer from the other disease.

The TB test

The case is the input of the first layer which is hamming network. The output of Hamming network will be the input of MaxNet network. Each case is classified to the closest example. One non zero output at the end will be the winner.

The Hamming & MAXNET network for TB test :



Hamming Algorithm

Step 1 - Specify the examples

Step 2 - Fix the weights matrix

$$W_h = {}^1/_2 * \begin{bmatrix} S_1^1 & \cdots & S_n^1 \\ \vdots & \ddots & \vdots \\ S_1^m & \cdots & S_n^m \end{bmatrix}$$

m*n where n is no of vector values and m is no of examples.

Step 3 - Find ϴ as ϴ= n/2

Step 4 - Specify the input vector

$$X = [X_1 .. X_n]$$

Step 5 - Find the output as

$$Y_k = X * W_h + \Theta$$

MaxNet Algorithm

Step 1 - Fix the weights matrix :

$$W_{ij} = \begin{cases} 1 & if\ i = j \\ -\varepsilon & if\ i \neq j \end{cases} \quad where\ 0 < \varepsilon < 1/m$$

$$W_m = \begin{bmatrix} 1 & \cdots & -\varepsilon \\ -\varepsilon & \ddots & -\varepsilon \\ -\varepsilon & \cdots & 1 \end{bmatrix} \quad m * m$$

Step 2 - Find Y_{k+1}

$$Y_{k+1} = f\,(W_m * Y_k) \quad f(x) = \begin{cases} x & if\ x > 0 \\ 0 & if\ x \leq 0 \end{cases}$$

Step 3 - Repeat step 2 until convergence

Hamming – MaxNet Network is having high speed in classification of random inputs.

Only one node which has the short Hamming distance from the example will be the non-zero output at the end that is, it will be considered as the winner. Hamming-MaxNet network always give a closed output to the examples.

8.7 LIST OF REFERENCES

- Kumar Satish, Neural Networks, Second edition Tata McGraw Hill-, 2013, ISBN1259006166, 9781259006166
- Neural Networks and Deep Learning: A Textbook 1st ed. 2018 Edition ISBN-13: 978-3319944623 ISBN-10: 3319944622
- Artificial Neural Networks, Robert J. Schalkoff, Tata McGraw Hill, ISBN: 9780071155540, 0071155546
- Widrow, "Generalization and information storage in networks of adaline 'neurons'," in Self Organizing Systems 1962, (M. C. Yovitz, G. T. Jacobi, and G. D. Goldstein, eds.), pp. 435-461, Washington, DC: Spartan Books, 1962
- A Comprehensive Guide to the Backpropagation Algorithm in Neural Networks - neptune.ai
- Back Propagation Algorithm (professional-ai.com)
- K Ming Leung (2007) Fixed eight Competitive Nets : Hamming Net. Polytechnic University

8.8 QUIZ

1. What is the objective of backpropagation algorithm?

   a) To develop learning algorithm for multilayer feedforward neural network, so that network can be trained to capture the mapping implicitly

   b) To develop learning algorithm for multilayer feedforward neural network

c) To develop learning algorithm for multilayer feedforward neural network
d) All of the above

2. The general limitations of back propagation rule is/are

a) Scaling
b) Slow convergence
c) Local minima problem
d) All of the above

3. Identify the type of learning in which labelled training data is used.

a) Unsupervised
b) Supervised
c) Reinforcement
d) Semi supervised

4. What is unsupervised learning?

a) Number of groups may be known
b) Features of group explicitly stated
c) Neither feature nor number of groups is known
d) None of the above

5. Real-Time decisions, Game AI, Learning Tasks, Skill acquisition, and Robot Navigation are applications of _____

a) Unsupervised Learning: Regression
b) Supervised Learning: Classification
c) Reinforcement Learning
d) Semi supervised

6. _____ are used to create non-linear neural networks.

a) nodes or neurons
b) weights
c) bias
d) activation function

7. The aim of _____ is to maximize the reward of the system through a trial and error.

a) supervised learning
b) **reinforcement learning**
c) unsupervised learning
d) semi supervised learning

8. Which problem is not linearly separable?

   a) **XOR**
   b) AND
   c) OR
   d) both XOR & OR

9. Which methos can train the network to have minimum error ?

   a) generalized least square
   b) Least Mean Squares method
   c) ordinary least squares
   d) Minima

10. Sigmoid function can be used for values between _____.

    a) 1 and 2
    b) 0 and -1
    c) 1 and -1
    d) 0 and 1

11. What is M in MADALINE?

    a) Multiple
    b) Many
    c) Minimum
    d) Maximum

12. _____takes a real-valued input and squashes it to the range.

    a) Sigmoid
    b) tanh
    c) ReLU
    d) Network

13. Adaline is also referred to as the _____

    a) **Widrow-Hoff learning rule**
    b) Perceptron learning rule
    c) Delta learning rule
    d) LMS learning rule

14. The output of _____ will be given as input to MaxNet.

    a) Adaline neural network
    b) backpropagation neural network
    c) **hamming neural network**
    d) Madaline neural network

15. Which of the following depends on Winner-Take-All WTA policy?

    a) Linear separability

    b) Madaline

    c) Adaline

    d) **MaxNet**

8.9 EXERCISE

1. What is learning? Explain the types of learning process.
2. Explain

a) Supervised learning
b) Unsupervised learning
c) Reinforcement learning

3. Explain the implementation of linear separability
4. Show that XOR is not linear separable.
5. Explain perceptron learning algorithm
6. What is adaptive linear neuron?
7. Describe Madaline in detail.
8. Give the steps for back propagation algorithm.
9. Explain MaxNet in detail.

8.10 VIDEO LINKS

1. (1897) Supervised, Unsupervised, Reinforcement learning in Machine Learning | Talib Iqbal - YouTube
2. (1863) Perceptron learning algorithm with solved example (in Hindi) - YouTube
3. (1863) Neural Network Learning Rules - Part 2 - Perceptron, Delta, Widrow-Hoff - YouTube
4. (1897) Lecture 8: Linear Separability | ANN - YouTube
5. (1863) Lecture 14: Widrow-Hoff Learning Rule | ANN - YouTube
6. (1897) Adaptive Linear Neuron | Adaline | Basic Concepts | Neural Networks - YouTube
7. (1897) Multiple Adaptive Linear Neuron | Madaline | Basic Concepts | Neural Networks - YouTube
8. (1897) Maxnet Neural Networks - YouTube

❄❄❄❄❄❄❄

9

FUZZY LOGIC

Unit Structure

9.1 OBJECTIVE

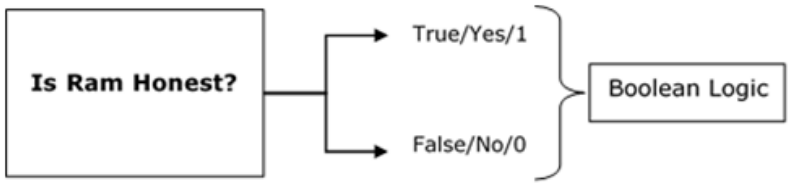
The word **fuzzy** refers to things which are not clear or are vague. Any event, process, or function that is changing continuously cannot always be defined as either true or false, which means that we need to define such activities in a Fuzzy manner.

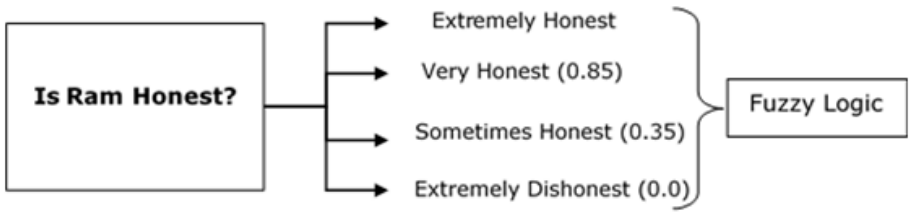
9.1.1 Introduction

What is Fuzzy Logic?

Fuzzy Logic resembles the human decision-making methodology. It deals with vague and imprecise information. This is gross oversimplification of the real-world problems and based on degrees of truth rather than usual true/false or 1/0 like Boolean logic.

Take a look at the following diagram. It shows that in fuzzy systems, the values are indicated by a number in the range from 0 to 1. Here 1.0 represents **absolute truth** and 0.0 represents **absolute falseness**. The number which indicates the value in fuzzy systems is called the **truth value**.





In other words, we can say that fuzzy logic is not logic that is fuzzy, but logic that is used to describe fuzziness. There can be numerous other examples like this with the help of which we can understand the concept of fuzzy logic.

Fuzzy Logic was introduced in 1965 by Lofti A. Zadeh in his research paper “Fuzzy Sets”. He is considered as the father of Fuzzy Logic.

9.1.2 Architecture:

Its Architecture contains four parts :

Rule base: It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision-making system, on the basis of linguistic information. Recent developments in fuzzy theory offer several effective methods for the design and tuning of fuzzy controllers. Most of these developments reduce the number of fuzzy rules.

Fuzzification: It is used to convert inputs i.e. crisp numbers into fuzzy sets. Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm’s, etc.

Inference engine: It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field. Next, the fired rules are combined to form the control actions.

Defuzzification: It is used to convert the fuzzy sets obtained by the inference engine into a crisp value. There are several defuzzification methods available and the best-suited one is used with a specific expert system to reduce the error.

9.1.3 Membership function:

Definition: A graph that defines how each point in the input space is mapped to membership value between 0 and 1. Input space is often referred to as the universe of discourse or universal set (u), which contains all the possible elements of concern in each particular application.

There are largely three types of fuzzifiers:

1. Singleton fuzzifier
2. Gaussian fuzzifier
3. Trapezoidal or triangular fuzzifier

9.1.4 What is Fuzzy Control?

It is a technique to embody human-like thinking's into a control system.

It may not be designed to give accurate reasoning but it is designed to give acceptable reasoning. It can emulate human deductive thinking, that is, the process people use to infer conclusions from what they know. Any uncertainties can be easily dealt with the help of fuzzy logic.

Advantages of Fuzzy Logic System:

1. This system can work with any type of inputs whether it is imprecise, distorted or noisy input information.
2. The construction of Fuzzy Logic Systems is easy and understandable.
3. Fuzzy logic comes with mathematical concepts of set theory and the reasoning of that is quite simple.
4. It provides a very efficient solution to complex problems in all fields of life as it resembles human reasoning and decision-making.
5. The algorithms can be described with little data, so little memory is required.

Disadvantages of Fuzzy Logic Systems:

1. Many researchers proposed different ways to solve a given problem through fuzzy logic which leads to ambiguity. There is no systematic approach to solve a given problem through fuzzy logic.
2. Proof of

3. 100its characteristics is difficult or impossible in most cases because every time we do not get a mathematical description of our approach.
4. As fuzzy logic works on precise as well as imprecise data so most of the time accuracy is compromised.

9.1.5 Application :

1. It is used in the aerospace field for altitude control of spacecraft and satellites.
2. It has been used in the automotive system for speed control, traffic control.
3. It is used for decision-making support systems and personal evaluation in the large company business.
4. It has application in the chemical industry for controlling the pH, drying, chemical distillation process.
5. Fuzzy logic is used in Natural language processing and various intensive applications in Artificial Intelligence.
6. Fuzzy logic is extensively used in modern control systems such as expert systems.
7. Fuzzy Logic is used with Neural Networks as it mimics how a person would make decisions, only much faster. It is done by Aggregation of data and changing it into more meaningful data by forming partial truths as Fuzzy sets.

9.2 CLASSICAL SETS

Classical set is a collection of **distinct** objects. For example, a set of students passing grades.

Each individual entity in a set is called a **member** or an **element** of the set.

The classical set is defined in such a way that the universe of discourse is splitted into two groups **members** and **non-members**. Hence, In case classical sets, **no partial membership exists**.

Let A is a given set. The membership function can be used to define a set A is given by:

$$\mu A(x)=\{ 1 \text{ if } x \in A$$

$$0 \text{ if } x \notin A \}$$

Operations on classical sets: For two sets A and B and Universe X:
Union:

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

This operation is also called **logical OR**.

Intersection:

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

This operation is also called **logical AND**.

Complement:

$$A' = \{x \mid x \notin A,\ x \in X\}$$

Difference:

$$A \mid B = \{x \mid x \in A \text{ and } x \notin B\}$$

Properties of classical sets: For two sets A and B and Universe X:

Commutativity:

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Associativity:

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

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

Distributivity:

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

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

Idempotency:

$$A \cup A = A$$

$$A \cap A = A$$

Identity:

$$A \cup \emptyset = A$$

$$A \cap X = A$$

$$A \cap \emptyset = \emptyset$$

$$A \cup X = X$$

Transitivity:

$$\text{If } A \subseteq B \text{ and } B \subseteq C, \text{ then } A \subseteq C$$

9.3 FUZZY SET

Fuzzy set is a set having **degrees of membership** between 1 and 0. Fuzzy sets are represented with tilde character(~). For example, Number of cars following traffic signals at a particular time out of all cars present will have membership value between [0,1].

Partial membership exists when member of one fuzzy set can also be a part of other fuzzy sets in the same universe. The degree of membership or truth is not same as probability, fuzzy truth represents membership in vaguely defined sets.

A fuzzy set A~ in the universe of discourse, U, can be defined as a set of ordered pairs and it is given by,

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

When the universe of discourse, U, is **discrete and finite**, fuzzy set A~ is given by

$$\tilde{A} = \sum_{i=1}^{n} \frac{\mu_{\tilde{A}}(x_i)}{x_i} = \frac{\mu_{\tilde{A}}(x_1)}{x_1} + \frac{\mu_{\tilde{A}}(x_2)}{x_2} + \ldots + \frac{\mu_{\tilde{A}}(x_n)}{x_n}$$

$$\tilde{A} = \int \frac{\mu_{\tilde{A}}(x)}{x}$$

Fuzzy sets also satisfy every property of classical sets.

Common Operations on fuzzy sets: Given two Fuzzy sets A~ and B~

Union : Fuzzy set C~ is union of Fuzzy sets A~ and B~ :

$$\tilde{C} = \tilde{A} \cup \tilde{B},$$

$$\mu_{\tilde{C}}(x) = \max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

Intersection: Fuzzy set D~ is intersection of Fuzzy sets A~ and B~ :

$$\tilde{D} = \tilde{A} \cap \tilde{B}$$

$$\mu_{\tilde{D}}(x) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$$

Complement: Fuzzy set E~ is complement of Fuzzy set A~ :

$$\tilde{E} = \mathbb{C}_{\tilde{A}} X$$

$$\mu_{\tilde{E}}(x) = 1 - \mu_{\tilde{A}}(x)$$

Some other useful operations on Fuzzy set:

Algebraic sum:

$$\mu A+B(x)= \mu A(x) + \mu B(x) - \mu A(x).\ \mu B(x)$$

Algebraic product:

$$\mu A.B(x)= \mu A(x).\ \mu B(x)$$

Bounded sum:

$$\mu A \oplus B(x)= \min\{1,\ \mu A(x) + \mu B(x)\}$$

Bounded difference:

$$\mu A \odot B(c)= \max\{0,\ \mu A(x) - \mu B(x)\}$$

9.4 CLASSICAL RELATIONS AND FUZZY RELATIONS

A relation is of fundamental importance in all-engineering, science, and mathematically based fields. It is associated with graph theory, a subject of wide impact in design and data manipulation. Relations are intimately involved in logic, approximate reasoning, classification, rule-based systems, pattern recognition, and control. Relations represent the mapping of the sets. In the case of crisp relation there are only two degrees of relationship between the elements of sets in a crisp relation, i.e., "completely related" and "not related". But fuzzy relations have infinite number of relationships between the extremes of completely related and not related between the elements of two or more sets considered. A crisp relation represents the presence or absence of association, interaction, or interconnectedness between the elements of two or more sets. Degrees of association can be represented by membership grades in a fuzzy relation by membership grades in a fuzzy relation in the same way as degrees of set membership are represented in the fuzzy set. Crisp set can be viewed as a restricted case of the more general fuzzy set concept.

9.5 CARTESIAN PRODUCT OF RELATIONS

Cartesian product is most commonly implemented in set theory. In addition to this, many real-life objects can be represented by using **cartesian products** such as a deck of cards, chess boards, computer images, etc. Most of the digital images displayed by computers are represented as pixels which are graphical representations of cartesian products.

9.5.1 What Is a Cartesian Product?

Cartesian product is the product of any two sets, but this product is actually ordered i.e., the resultant set contains all possible and ordered pairs such that the first element of the pair belongs to the first set and the second element belongs to the second set. Since their order of appearance is important, we call them first and second elements respectively. We use ordered pairs to obtain a new set from two given sets A and B.

An ordered pair (p, q) consists of two values p and q. Example: (1, 3) and (- 4, 10) is an ordered pair where these pair of numbers are in a specific order.

Consequently, $(p, q) \neq (q,p)$ unless p = q. In general, (p, q) = (s, t) if and only if p = s and q = t. Example: (1, 3) is not equivalent to (3, 1) i.e., $(1, 3) \neq (3, 1)$.

(1, 2) and (- 4, 12) An ordered pair is a pair of numbers in a specific order. For example, (1, 2) and (- 4, 12) are ordered pairs. The order of the two numbers is important: (1, 2) is not equivalent to (2, 1) -- $(1, 2) \neq (2, 1)$.

9.5.2 Cartesian Product Definition

If C and D are two non-empty sets, then the cartesian product, C × D is the set of all ordered pairs (a, b) with the first element from C and the second element from D. Similar to the other product operations, we use the same multiplication sign × to represent the cartesian product between two sets. Here, we use the notation C × D for the Cartesian product of C and D.

By using the set-builder notations, we can write the cartesian product as:

$C \times D = \{(a,b): a \in C, b \in D\}$. Here a belongs to set C and b belongs to set D.

If both the sets are the same i.e, if C = D then C × D is called the cartesian square of the set C and it is denoted by C^2

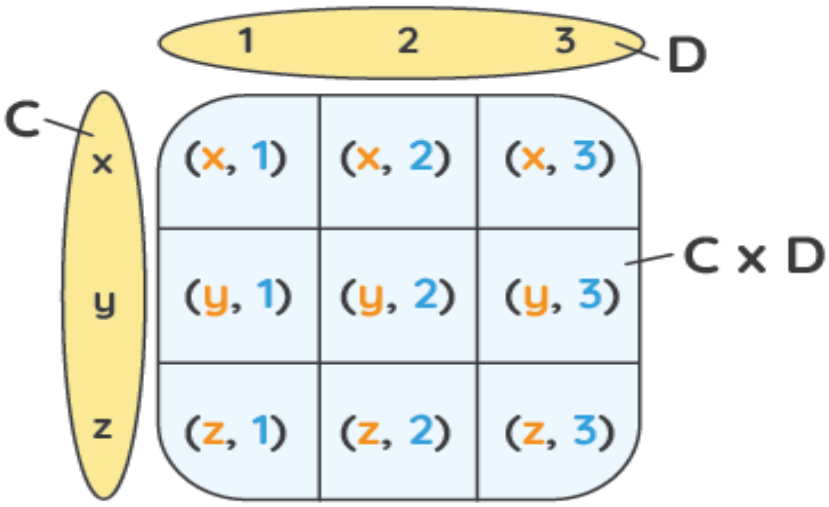
$C^2 = C \times C = \{(a,b): a \in C, b \in C\}$

Cartesian Product of Sets:

The cartesian products of sets can be considered as the product of two non-empty sets in an ordered way. The final product of the sets will be a collection of all ordered pairs obtained by the product of the two non-empty sets. In an ordered pair, two elements are taken from each of the two sets.

9.5.3 How to Find Cartesian Product?

Consider two non-empty sets C = {x, y, z} and D = {1, 2, 3} as shown in the below image:



Cartesian Product

The cartesian product, also known as the cross-product or the product set of C and D is obtained by following the below-mentioned steps:

The first element x is taken from the set C {x, y, z} and the second element 1 is taken from the second set D {1, 2, 3}

Both these elements are multiplied to form the first ordered pair (x,1)

The same step is repeated for all the other pairs too until all the possible combinations are chosen

The entire collection of all such ordered pairs gives us a cartesian product C x D = {(x,1), (x,2), (x,3), (y,1), (y,2), (y,3),(z,1), (z,2), (z,3)}.

Similarly, we can find the cartesian product of D x C.

Let us find the cartesian product of the two sets C and D, where C = {11,12,13} and D = {7, 8}.

After following the steps mentioned above:

The resultant product C x D will be: {(11,7), (11,8), (12,7), (12,8), (13,7), (13,8)}.

Similarly, we can find the cartesian product of D and C as D × C = { (7,11),(7,12),(7,13),(8,11),(8,12),(8,13)}.

The cartesian products C × D and D × C do not contain exactly the same ordered pairs. Hence, in general, C × D ≠ D × C.

9.5.4 Cartesian Product of Several Sets:

We can extend or define the cartesian product to more than two sets. The cartesian product of several input sets is a larger set that contains every ordered combinations of all the input set elements. The cartesian product of three sets P, Q, and R can be written as:

P × Q × R = { (a,b,c): a ∈ P, b ∈ Q, c ∈ R }

Let us consider the example of three sets A, B and C, where A = {2,3} , B = {x,y}, and C = {5,6}. In order to find the cartesian product of A × B × C, let us find the cartesian product of A × B first.

A × B = {(2,x), (2,y),(3,x),(3,y)}.

A × B × C = {(2,x,5), (2,x,6), (2,y,5), (2,y,6), (3,x,5), (3,x,6), (3,y,5), (3,y,6)}

9.5.5 Cartesian Product of Empty Set:

The empty set is a unique set with no elements. Both its size or cardinality i.e, the total count of elements in a set will remain as zero. An empty set is also referred to as a void set. The Cartesian product of C and the empty set ∅ is the empty set ∅.

Let C × ∅ = {(a,b)| a ∈ C, b∈ ∅}. There is no element in ∅. C × D =∅ if and only if C = ∅ or D = ∅. Here, the cartesian product of two sets will result in an empty set if and only if, either of the sets is an empty set.

Consider the example: If C = {1, 2} and D = ϕ. Then, C × D = ϕ and D × C = ϕ.

These are the properties of the empty set:

Empty set's subset is the empty set itself: ∀C:C⊆∅⇒C=∅∀C:C⊆∅⇒C=∅

The empty set's power set is the set containing only the empty set: $2^n = 2^0 = 1$.

The cardinality of the empty set i.e., the number of elements of the set is zero: n(∅) = 0

9.5.6 Cartesian Product of Countable Sets:

The cartesian product of two countable sets is countable. Let us take these two cases to understand this:

Consider an integer b in such a way that b > 1. Then the cartesian product of b countable sets is countable.

Consider the two countable sets A = { a0a0, a1a1,a2a2.. } and B = {b0b0, b1b1,b2b2.. }. If both the sets A and B are countable, then the resulting set will also be countable.

9.5.7 Cartesian Product of Relations:

The cartesian product of relations is the same as the relation across two sets. Generally, the cartesian product is represented for a set and not for a relation. Further the universal relation relates every element of one set to an element of another set, and hence it can be represented as the cartesian product of relations.

Properties	Representation
Cartesian product is non-commutative i.e., the result depends on the order of the sets	Consider the two sets C and D: C × D ≠ D × C C × D = D × C, if and only if C = D.

Properties	Representation
	C × D = ∅, if either C = ∅ or D =∅
Cartesian product is non-associative i.e., it does not follow the associative property. rearranging the parentheses in this expression will change the result.	(C × D) × E ≠ C × (D × E)
Distributive property over the intersection of sets.	C × (D∩E) = (C × D) ∩ (C × E)
Distributive property over the union of sets.	C × (D∪E) = (C × D) ∪ (C × E)

9.5.8 Cardinality of a Cartesian Product

The cardinality of a set is the total number of elements present in the set. The cardinal number of A is n(A) = number of all the elements in set A. Example: The cardinal number of a set of English alphabets A = (a, b, c .....x. y. z) is n(A) = 26.

The cardinality of a cartesian product of two sets C and D is equal to the product of the cardinalities of these two sets: n(C × D) = n(D × C) = n(C) × n(D). Similarly, n(C1C1 ×…× CnCn) = n(C1C1) ×…× n(CnCn).

Consider two sets C and D, where C = {2,3} and n(C) = 2, D = {5,4,7} and n(D) = 3. So, n(C × D) = n(C) × n(D) = 2 × 3 = 6. Here, we can see that the cardinality of the output set C × D is equal to the product of the cardinalities of all the input sets C and D. That is, 6.

9.6 CLASSICAL RELATIONS

Crisp relation basically is a collection of ordered pairs. So, if A and B are the two sets then all its ordered pairs are denoted by the Cartesian product A ×B i.e {(a , b)∨ a∈ A ∧ b∈B} and a subset of this collection will be Binary relation Let us make it clear with the help of an example

Example: Consider two crisp sets A and B as given below

A={1,2,3,4} and B={3,5,7}

Then cartesian product A X B = {(1,3),(1,5),(1,7), (2,3), (2,5), (2,7), (3,3), (3,5), (3,7), (4,3), (4,5), (4,7)}

Now let us define a relation R as a collection of all those ordered pairs where b=a+1 i.e if i write it mathematically

R= {(a,b)|b= a+1, (a,b) Ɛ A X B

Then binary relation R= {(2,3), (4,5)} in this case

Note since we have two sets A and B, the relation R is a binary relation.

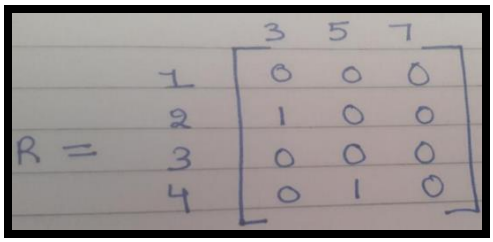
So, a relation is basically a collection of order pairs which satisfy a particular mapping or a particular definition.

We can also say that A ×B essentially provides a mapping from an element a that belongs to set A i.e a ∈ A to another element b that belongs to set B i.e b ∈ B . So, it is basically a mapping and this mapping is expressed by means of an order pair and this particular mapping is called a relation.

This crisp relation can be represented in a more visual and compact way in the form of a matrix

If we take the previous example R= {(2,3), (4,5)}

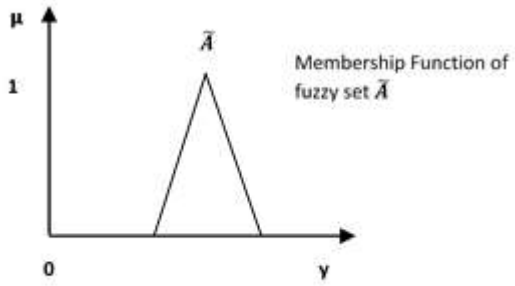
We can also represent the relation in given example in matrix form as shown below:



- So, you can see 1 and 3 does not belong to the set therefore it is 0 and the elements 2 and 3 belong to the set R, so it is 1. So, here 0 and 1 are the entries in the relation matrix. 0 indicates that the ordered pair does not belong to relation R and 1 indicates that the ordered pair belongs to the relation R.
- So, this way a relation can be represented by means of a Relation Matrix.
- You can also easily observe from the relation matrix R given above that there are only two degrees of relationship between elements of the sets in a crisp relation: the relationships ‘‘completely related’’ (1) and ‘‘not related,’’ (0) in a binary sense.

9.7 FUZZY RELATIONS MEMBERSHIP FUNCTIONS

We already know that fuzzy logic is not logic that is fuzzy but logic that is used to describe fuzziness. This fuzziness is best characterized by its membership function. In other words, we can say that membership function represents the degree of truth in fuzzy logic.



Membership Function

Following are a few important points relating to the membership function −

1. Membership functions were first introduced in 1965 by Lofti A. Zadeh in his first research paper "fuzzy sets".
2. Membership functions characterize fuzziness (i.e., all the information in fuzzy set), whether the elements in fuzzy sets are discrete or continuous.
3. Membership functions can be defined as a technique to solve practical problems by experience rather than knowledge.
4. Membership functions are represented by graphical forms.
5. Rules for defining fuzziness are fuzzy too.

9.7.1 Mathematical Notation

We have already studied that a fuzzy set $\tilde{A}$ in the universe of information U can be defined as a set of ordered pairs and it can be represented mathematically as −

A˜={(y,μA˜(y))|y∈U}A~={(y,μA~(y))|y∈U}

Here μA˜(·)μA~(·) = membership function of A˜A~; this assumes values in the range from 0 to 1, i.e., μA˜(·)∈[0,1]μA~(·)∈[0,1]. The membership function μA˜(·)μA~(·) maps UU to the membership spaceMM.

The dot (·)(·) in the membership function described above, represents the element in a fuzzy set; whether it is discrete or continuous.

9.8 FEATURES OF MEMBERSHIP FUNCTIONS

We will now discuss the different features of Membership Functions.

1. Core

For any fuzzy set A˜A~, the core of a membership function is that region of universe that is characterize by full membership in the set. Hence, core consists of all those elements yy of the universe of information such that,

μA˜(y)=1μA~(y)=1

2. Support

For any fuzzy set A˜A~, the support of a membership function is the region of universe that is characterize by a nonzero membership in the set.

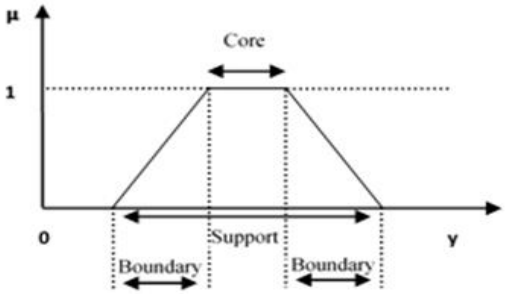
Hence core consists of all those elements yy of the universe of information such that,

μA˜(y)>0μA~(y)>0

3. Boundary

For any fuzzy set A˜A~, the boundary of a membership function is the region of universe that is characterized by a nonzero but incomplete membership in the set. Hence, core consists of all those elements yy of the universe of information such that,

1>μA˜(y)>0



Features of Membership Function

9.9 FUZZIFICATION

It may be defined as the process of transforming a crisp set to a fuzzy set or a fuzzy set to fuzzier set. Basically, this operation translates accurate crisp input values into linguistic variables.

Following are the two important methods of fuzzification −

Support Fuzzification(s-fuzzification) Method

In this method, the fuzzified set can be expressed with the help of the following relation −

A˜=μ1Q(x1)+μ2Q(x2)+...+μnQ(xn)A~=μ1Q(x1)+μ2Q(x2)+...+μnQ(xn)

Here the fuzzy set Q(xi)Q(xi) is called as kernel of fuzzification. This method is implemented by keeping μiμi constant and xixi being transformed to a fuzzy set Q(xi)Q(xi).

Grade Fuzzification (g-fuzzification) Method

It is quite similar to the above method but the main difference is that it kept xixi constant and μiμi is expressed as a fuzzy set.

9.10 METHODS OF MEMBERSHIP VALUE ASSIGNMENT

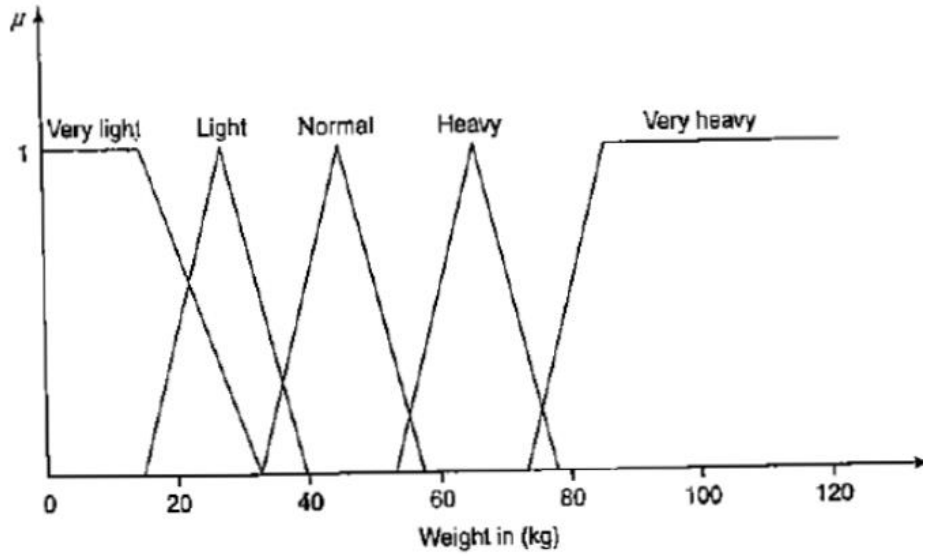
The method of assigning membership values are as follows:

1. Intuition
2. Inference
3. Rank Ordering
4. Angular Fuzzy Sets

5. Neural Networks
6. Genetic Algorithm
7. Inductive Reasoning.

1) Intuition

Intuition method is based upon the common intelligence of human. It is the capacity of the human to develop membership functions on the basis of their own intelligence and understanding capacity. There should be an in-depth knowledge of the application to which membership value assignment has to be made. The Figure below shows various shapes of weights of people measured in kilogram in the universe. Each curve is a membership function corresponding to various fuzzy (linguistic) variables; such as very light, light, normal, heavy and very heavy. The curves are based on context functions and the human developing them. For example, if the weights are referred to range of thin persons we get one set of curves, and if they are referred to range of normal weighing persons we get another set and so on. The main characteristics of these curves for their usage in fuzzy operations are based on their overlapping capacity.



2) Inference

The inference method uses knowledge to perform deductive reasoning. Deduction achieves conclusion by means of forward inference. There are various methods for performing deductive reasoning. Here the knowledge of geometrical shapes and geometry is used for defining membership values. The membership functions may be defined by various shapes: triangular, trapezoidal, bell-shaped, Gaussian and so on. The inference method here is discussed via triangular shape.

Consider a triangle, where X, Y and Z are the angles such that

$X \geq Y \geq Z \geq 0$

and let u be the universe of triangles, i.e., $U = \{(X, Y, Z) | X > Y \geq Z \geq 0; X + Y + Z = 180\}$

There are various types of triangles available. Here a few are considered to explain inference methodology:

I = isosceles triangle (approximate)
E = equilateral triangle (approximate)
R = right-angle triangle (approximate)
IR = isosceles and right-angle triangle (approximate)
T = other triangles

The membership values of approximate isosceles triangle are obtained using the

following definition, where: $X \geq Y \geq Z \geq 0$ and $X + Y + Z = 180°$:

$$\mu_I(X, Y, Z) = 1 - \frac{1}{60°} \min(X - Y, Y - Z)$$

If X=Y or Y=Z and if X=120°,Y=60° and Z=0° , we get

$$\mu_I(X, Y, Z) = 1 - \frac{1}{60°} \min(120° - 60°, 60° - 0°)$$
$$= 1 - \frac{1}{60°} \min(60°, 60°)$$
$$= 1 - \frac{1}{60°} \times 60°$$
$$= 1 - 1 = 0$$

The membership value of approximate right-angle triangle is given by:

$$\mu_R(X, Y, Z) = 1 - \frac{1}{90°} |X - 90°|$$

3) Rank Ordering

The formation of government is based on the polling concept; to identify a best student, ranking may be performed; to buy a car, one can ask for several opinions and so on. All the above-mentioned activities are carried out on the basis of the preferences made by an individual, a committee, a poll and other opinion methods. This methodology can be adapted to assign membership values to a fuzzy variable. Pairwise comparisons enable us to determine preferences and this results in determining the order of the membership.

9.11 DEFUZZIFICATION

It may be defined as the process of reducing a fuzzy set into a crisp set or to convert a fuzzy member into a crisp member.

We have already studied that the fuzzification process involves conversion from crisp quantities to fuzzy quantities. In a number of engineering applications, it is necessary to defuzzify the result or rather "fuzzy result" so that it must be converted to crisp result. Mathematically, the process of Defuzzification is also called "rounding it off".

9.12 LAMBDA –CUTS FOR FUZZY SETS (ALPHA-CUTS)

Consider a fuzzy set Aλ,. The set Aλ(0< λ<1) ,called the lambda (λ)- cut or (alpha cut [α]-cut)set, is a crisp set of the fuzzy set and is defined as follows:

$A\lambda = \{x|\mu A(x) \geq \lambda\}$

The setAλ.is called a weak lambda-cut set if it consists of all the elements of a fuzzy set whose membership functions have values greater than or equal to a specified value. On the other hand, the set Aλ, is called a Strong lambda-cut set if it consists of all the elements of a fuzzy set whose membership functions have values strictly greater than specified value. A strong λ-cut set is given by: **$A\lambda = \{x|\mu A(x) \geq \lambda\}$**

Properties of lambda cut fuzzy sets:

If A and B are two fuzzy sets, defined with the same universe of discourse, then

1. 1 $(A \cup B)\lambda = A\lambda \cup B\lambda$
2. 2 $(A \cap B)\lambda = A\lambda \cap B\lambda$
3. 3 $(A)\lambda$ 6= Aλ except for value of $\lambda = 0.5$
4. 4 For any $\lambda \leq \alpha$, where α varies between 0 and 1, it is true that $A\alpha \subseteq A\lambda$, where the value of A0 will be the universe of discourse.

9.13 LAMBDA CUT FOR FUZZY RELATIONS

The lambda cut procedure for relations is similar to that for the lambda cut sets. Considering a fuzzy relation R ∼, in which some of the relational matrix represents a fuzzy set.

A fuzzy relation can be converted into a crisp relation by depending the lambda cut relation of the fuzzy relation as:

$R\lambda = \{x,y/\mu R\ (x,y) \geq \lambda\}$

Properties of Lambda Cut Relations:

If R and S are two fuzzy relations, defined with the same fuzzy sets over the same universe of discourses, then

(1) $(R \cup S)\lambda = R\lambda \cup S\lambda$

(2) $(R \cap S)\lambda = R\lambda \cap S\lambda$

(3) $(R)\lambda\ 6= R\lambda$

(4) For $\lambda \leq \alpha$, where α between 0 and 1 , then $R\alpha \subseteq R\lambda$

9.14 DEFUZZIFICATION METHODS

The different methods of Defuzzification are described below :

1. Max-Membership Method

This method is limited to peak output functions and also known as height method. Mathematically it can be represented as follows −

$\mu\tilde{A}(x*)>\mu\tilde{A}(x)forallx\in X\mu A\sim(x*)>\mu A\sim(x)forallx\in X$

Here, $x*x*$ is the defuzzified output.

2. Centroid Method

This method is also known as the center of area or the center of gravity method. Mathematically, the defuzzified output $x*x*$ will be represented as −

$x*=\int\mu\tilde{A}(x).xdx\int\mu\tilde{A}(x).dxx*=\int\mu A\sim(x).xdx\int\mu A\sim(x).dx$

3. Weighted Average Method

In this method, each membership function is weighted by its maximum membership value. Mathematically, the defuzzified output $x*x*$ will be represented as −

$x*=\sum\mu\tilde{A}(\overline{xi})\overline{.xi}\sum\mu\tilde{A}(\overline{xi})x*=\sum\mu A\sim(\bar{xi}).\bar{xi}\sum\mu A\sim(\bar{xi})$

4. Mean-Max Membership

This method is also known as the middle of the maxima. Mathematically, the defuzzified output $x*x*$ will be represented as −

$x*=\sum i=1nxi\overline{n}$

Multiple Choice Questions

1) Which of the following is associated with fuzzy logic?

   a) Crisp set logic

   b) Many-valued logic

   c) Two-valued logic

   d) Binary set logic

2) The truth values of traditional set theory can be defined as _____ and that of fuzzy logic is termed as _____.

a) Either 0 or 1, either 0 or 1.
b) Between 0 & 1, either 0 or 1.
c) Either 0 or 1, between 0 & 1.
d) Between 0 & 1, between 0 & 1.

3) How many types of random variables are there in Fuzzy logic?

a) 2
b) 4
c) 1
d) 3

4) Which of the following represents the values of set membership?

a) Degree of truth
b) Probabilities
c) Discrete set
d) Both a & b

5) The probability density function is represented by

a) Continuous variable
b) Discrete variable
c) Probability distributions for Continuous variables
d) Probability distributions

6) _____is used for probability theory sentences.

a) Logic
b) Extension of propositional logic
c) Conditional logic
d) None of the above

7) Which of the following fuzzy operators are utilized in fuzzy set theory?

a) AND
b) OR
c) NOT
d) EX-OR

8) What is the name of the operator in fuzzy set theory, which is found to be linguistic in nature?

a) Lingual Variable
b) Fuzz Variable
c) Hedges
d) None of the above

9) _____ represents the fuzzy logic

a) IF-THEN rules
b) IF-THEN-ELSE rules
c) Both a & b
d) None of the above

10) Uncertainty can be represented by _____

a) Entropy
b) Fuzzy logic
c) Probability
d) All of the above

❄❄❄❄❄❄❄

10

FUZZY INFERENCE SYSTEM - I

Unit Structure

10.1 OBJECTIVE

In a fuzzy inference system, an inference rule is a mapping from a set of premise facts to a conclusion fact. There are several approaches to fuzzy inference system design. For example, one approach is based on a set of rules whose premises are all combinations of the input fuzzy sets, while the conclusion is determined by the output fuzzy set. Another is based on a set of rules whose premises are all combinations of the input fuzzy sets, while the conclusion is determined by the complement (negation) of the output fuzzy set. Yet another approach is based on a set of rules whose premises are the input fuzzy sets, and whose conclusions are the complement of the output fuzzy set.

10.1.1 Introduction

Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

Characteristics of Fuzzy Inference System:

- The output from FIS is always a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have fuzzy output when it is used as a controller.

- A defuzzification unit would be there with FIS to convert fuzzy variables into crisp variables.

10.2 TRUTH VALUES AND TABLES IN FUZZY LOGIC

Fuzzy logic uses linguistic variables. The values of a linguistic variable are words or sentences in a natural or artificial language. For example, height is a linguistic variable if it rakes values such as tall, medium, short and so on. The linguistic variable provides approximate characterization of a complex problem. The name of the variable, the universe of discourse and a fuzzy subset of universe of discourse characterize a fuzzy variable. A linguistic variable is a variable of a higher order than a fuzzy variable and its values are taken to be fuzzy variables.

A linguistic variable is characterized by:

- name of the variable (x);
- term set of the variable t(x);
- syntactic rule for generating the values of x;

semantic rule for associating each value of x with its meaning.

Apart from the linguistic variables, there exists what are called as linguistic hedges (linguistic modifiers).

Examples of linguistic hedges In the fuzzy set "very tall", the word "very" is a linguistic hedge. A few popular linguistic hedges include: very, highly, slightly, moderately, plus, minus, fairly, rather.

Reasoning has logic as its basis, whereas propositions are text sentences expressed in any language and are generally expressed in a canonical form as **zisP**

Where the symbol of the subject and P the predicate designing the characteristics of the subject.

Example:

"London is in United Kingdom"

"London" the subject and "in United Kingdom" the predicate, which specifies a property of "London," i.e., its geographical location in United Kingdom.

Negation:

Every proposition has its opposite, called negation.

Logic Functions:

Truth cables define logic functions of two propositions.

Let X and Y be two propositions, either of which can be true or false.

The basic logic operations performed over the propositions are the following:

1. Conjunction (Λ) :X AND Y.
2. Disjunction (∨) :XOR Y.
3. Implication or conditional (=>): IF X THEN Y.
4. Bidirectional or equivalence **(<->):** X IF AND ONLY IF Y.

Inference Rules:

On the basis of these operations on propositions, inference rules can be formulated. Few inference rules are as follows:

$$[X \wedge (X \Rightarrow Y)] \Rightarrow Y$$
$$[\bar{Y} \wedge (X \Rightarrow Y)] \Rightarrow \bar{X}$$
$$[(X \Rightarrow Y) \wedge (Y \Rightarrow Z)] \Rightarrow (X \Rightarrow Z)$$

The above rules produce certain propositions that are always true irrespective of the truth values of propositions X and Y. Such propositions are called **tautologies.**

An extension of set-theoretic bivalence logic is the fuzzy logic where the truth values are terms of the linguistic variable "truth." The truth values of propositions in fuzzy logic are allowed to range over the unit interval [0, 1]. A truth value in fuzzy logic "very true" may be interpreted as a fuzzy set in [0, I]. The truth value of the proposition" Z is A," or simply the truth value of A, denoted by tv(A) is defined by a point in [0, 1] (called the numerical truth value) or a fuzzy set in [0, 1] (called the linguistic truth value). The truth value of a proposition can be obtained from the logic operations of other propositions whose truth values are known. If tv(X) and tv(Y) are numerical truth values of propositions X and Y, respectively, then

$$tv(X \text{ AND } Y) = tv(X) \wedge tv(Y) = \min[tv(X), tv(Y)] \quad \text{(Intersection)}$$
$$tv(X \text{ OR } Y) = tv(X) \vee tv(Y) = \max[tv(X), tv(Y)] \quad \text{(Union)}$$
$$tv(\text{NOT } X) = 1 - tv(X) \quad \text{(Complement)}$$
$$tv(X \Rightarrow Y) = tv(X) \Rightarrow tv(B) = \max[1 - tv(X), \min[tv(X), tv(Y)]]$$

10.3 FUZZY PROPOSITIONS

The fuzzy propositions are as follows:

1. **Fuzzy predicates:** In fuzzy logic the predicates can be fuzzy, for example, tall, short, quick. Hence, we have proposition like "Peter is tall."

2. **Fuzzy-predicate modifiers:** In fuzzy logic, there exists a wide range of predicate modifiers that act as hedges. For example, very, fairly, moderately, rather, slightly. These predicate modifiers are necessary for generating the values of a linguistic variable. An example can be the proposition "Climate is moderately cool," where" moderately" is the fuzzy predicate modifier.

3. **Fuzzy quantifiers:** The fuzzy quantifiers such as most, several, many, frequently are used in fuzzy logic. Employing these, we can have proposition like "Many people are educated." A fuzzy quantifier can be interpreted as a fuzzy number or a fuzzy proposition, which provides an imprecise characterization of the cardinality of one or more fuzzy or non-fuzzy sets. Fuzzy quantifiers can be used to represent the meaning of propositions containing probabilities; as a result, they can be used to manipulate probabilities within fuzzy logic.

4. **Fuzzy qualifiers:** There are four modes of qualification in fuzzy logic, which are as follows:

   (i) Fuzzy truth qualification: It is expressed as "x is τ ," in which τ is a fuzzy truth value. A fuzzy truth value claims the degree of truth of a fuzzy proposition.

   Example

   (Paul is Young) is NOT VERY True.
   Here the qualified proposition is (Paul is Young) and the qualifying fuzzy truth value is "NOT Very True."

   (ii) Fuzzy probability qualification: It is denoted as "x is λ "where, λ is fuzzy probability. In conventional logic, probability is either numerical or an interval. In fuzzy logic, fuzzy probability is expressed by terms such as likely, very likely, unlikely, around and so on.

   Example:
   (Paul is Young) is Likely.
   Here the qualifying fuzzy probability is "Likely." These probabilities may be interpreted as fuzzy numbers, which may be manipulated using fuzzy arithmetic.

   (iii) Fuzzy possibility qualification:

   It is expressed as "x is π",where πis a fuzzy possibility and can be of the following forms: possible, quite possible, almost impossible. These values can be interpreted as labels of fuzzy subsets of the real line.

   Example:
   (Paul is Young) is Almost Impossible.
   Here the qualifying fuzzy possibility is "Almost Impossible."

(iv) Fuzzy usuality qualification: It is expressed as "usually (X) =usually (X is F),"in which the subject X is a variable taking values in a universe of discourse U and the predicate F is a fuzzy subset of U and interpreted as a usual value of X denoted by U(X) = F. The propositions that are usually true or the events that have high probability of occurrence are related by the concept of usuality qualification.

10.4 FORMATION OF RULES

The general way of representing human knowledge is by forming natural language expressions given by

IF antecedent THEN consequent.

The above expression is referred to as the IF - THEN rule based form. There are three general forms that exist for any linguistic variable. They are:

(a) Assignment statements;
(b) Conditional statements;
(c) Unconditional statements.

1. Assignment statements:

They are of the form

y =small

Orange colour = orange

a=s

Paul is not tall and not very short Climate = autumn

Outside temperature = normal

These statements utilize "=" for assignment.

2. Conditional statements:

The following are some examples.

- IF y is very cool THEN stop.
- IF A is high THEN B is low ELSE B is not low.
- IF temperature is high THEN climate is hot.

The conditional statements use the "IF - THEN" rule-based form.

3. Unconditional statements:

They can be of the form

- Go to sum.
- Stop.
- Divide by a.
- Turn the pressure low

10.5 DECOMPOSITION OF RULES

A compound rule is a collection of many simple rules combined together. Any compound rule structure may be decomposed and reduced to a number of simple canonical rule forms. The rules are generally based on natural language representations.

The following are the methods used for decomposition of compound linguistic rules into simple canonical rules.

- Multiple conjunctive antecedents
- Multiple disjunctive antecedents
- Conditional statements (ELSE and UNLESS)
- Nested IF-THEN rules

10.6 AGGREGATION OF FUZZY RULES

The rule-based system involves more than one rule. Aggregation of rules is the process of obtaining the overall consequents from the individual consequents provided by each rule.

The following two methods are used for aggregation of fuzzy rules:

1. Conjunctive system of rules:

For a system of rules to be jointly satisfied, the rules are connected by "and" connectives. Here, the aggregated output, y, is determined by the fuzzy intersection of all individual rule consequents, yi, where i=1ton, as

Y= y1, y2……and yn

This aggregated output can be defined by the membership function

$$\mu_y(y) = \min[\mu_{y_1}(y), \mu_{y_2}(y), \ldots, \mu_{y_n}(y)] \quad \text{for} \quad y \in Y$$

2. Disjunctive system of rules:

In this case, the satisfaction of at least one rule is required. The rules are connected by "or" connectives. Here, the fuzzy union of all individual rule contributions determines the aggregated output, as

Y= y1 or y2 or…. Or yn

Again, it can be defined by the membership function

$$\mu_y(y) = \max[\mu_{y_1}(y), \mu_{y_2}(y), \ldots \mu_{y_n}(y)] \text{ for } y \in Y$$

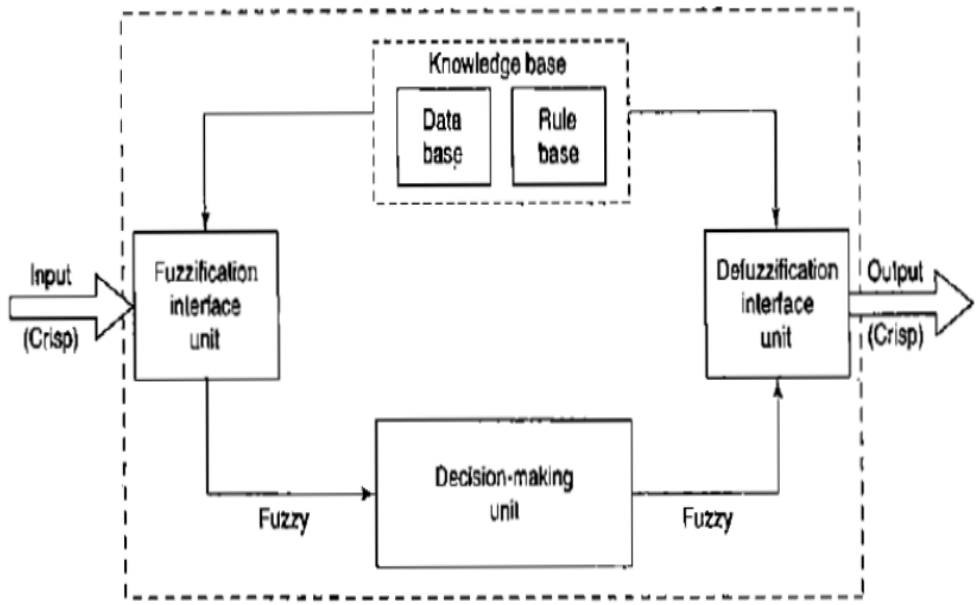
10.7 FUZZY INFERENCE SYSTEM (FIS)

Fuzzy rule-based systems, fuzzy models, and fuzzy expert systems are generally known as inference systems. The key unit of a fuzzy logic system is FIS. The primary work of this system is decision making.

FIS uses "IF ... THEN" rules along with connectors "OR" or "AND" for making necessary decision rules. The input to FlS may be fuzzy or crisp, but the output from FIS is always a fuzzy set. When FlS is used as a controller, it is necessary to have crisp output. Hence, there should be a defuzzification unit for converting fuzzy variables into crisp variables along FIS.

10.7.1 Construction and Working Principle of FIS:

1. A rule base that contains numerous fuzzy IF-THEN rules.
2. A database that defines the membership functions of fuzzy sets used in fuzzy rules.
1. 3.Decision making unit that performs operations on the rules.
2. 4.Fuzzification interface unit that converts the crisp quantities into fuzzy quantities.
3. 5.Defuzzification interface unit that converts the fuzzy quantities into crisp quantities.



Working methodology of FIS:

Initially, in the fuzzif1cation unit, the crisp input is converted into a fuzzy input. Various fuzzification methods are employed for this. After this process, rule base is formed. Database and rule base are collectively called the knowledge base. Finally, defuzzification process is carried out to produce crisp output. Mainly, the fuzzy rules are formed in the rule base and suitable decisions are made in the decision-making unit.

10.7.2 Methods of FI:

Let us now discuss the different methods of FIS. Following are the two important methods of FIS, having different consequent of fuzzy rules −

- Mamdani Fuzzy Inference System
- Takagi-Sugeno Fuzzy Model (TS Method)

Mamdani Fuzzy Inference System

This system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

Steps for Computing the Output

Following steps need to be followed to compute the output from this FIS −

Step 1 Set of fuzzy rules need to be determined in this step.

Step 2 In this step, by using input membership function, the input would be made fuzzy.

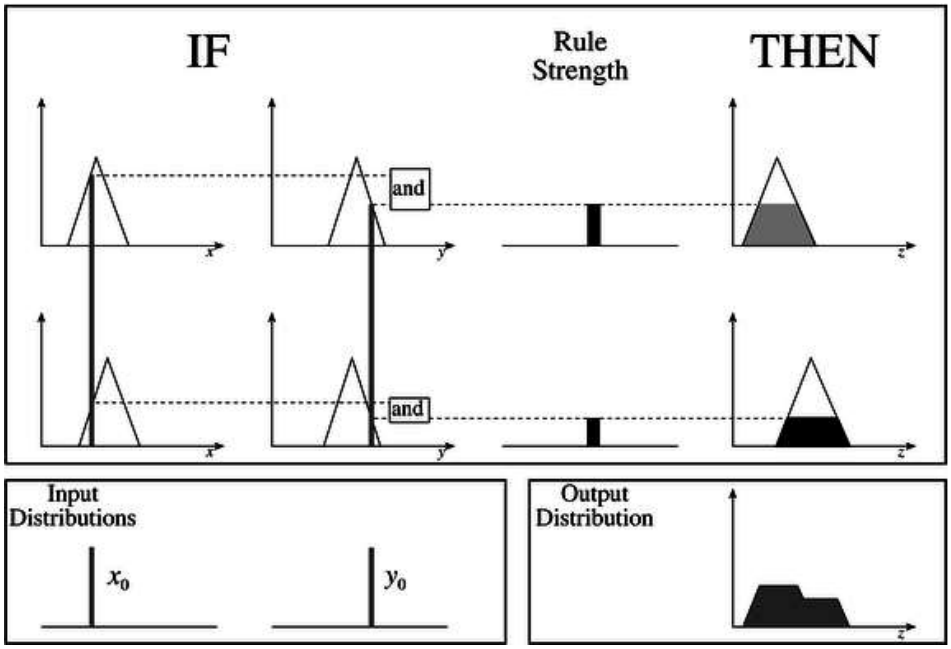
Step 3 Now establish the rule strength by combining the fuzzified inputs according to fuzzy rules.

Step 4 In this step, determine the consequent of rule by combining the rule strength and the output membership function.

Step 5 For getting output distribution combine all the consequents.

Step 6 Finally, a defuzzified output distribution is obtained.

Following is a block diagram of Mamdani Fuzzy Interface System.



Takagi-Sugeno Fuzzy Model (TS Method)

This model was proposed by Takagi, Sugeno and Kang in 1985. Format of this rule is given as −

IF x is A and y is B THEN Z = f(x,y)

Here, *AB* are fuzzy sets in antecedents and *z = f(x,y)* is a crisp function in the consequent.

Fuzzy Inference Process

The fuzzy inference process under Takagi-Sugeno Fuzzy Model (TS Method) works in the following way −

- **Step 1: Fuzzifying the inputs** − Here, the inputs of the system are made fuzzy.
- **Step 2: Applying the fuzzy operator** − In this step, the fuzzy operators must be applied to get the output.

Rule Format of the Sugeno Form

The rule format of Sugeno form is given by −

if 7 = x and 9 = y then output is z = ax+by+c

10.7.3 Comparison between the two methods

Let us now understand the comparison between the Mamdani System and the Sugeno Model.

- **Output Membership Function** − The main difference between them is on the basis of output membership function. The Sugeno output membership functions are either linear or constant.
- **Aggregation and Defuzzification Procedure** − The difference between them also lies in the consequence of fuzzy rules and due to the same their aggregation and defuzzification procedure also differs.
- **Mathematical Rules** − More mathematical rules exist for the Sugeno rule than the Mamdani rule.
- **Adjustable Parameters** − The Sugeno controller has more adjustable parameters than the Mamdani controller.

Multiple Choice Questions

1) Which of the following is considered while determining the nature of the learning problem?

   a) Problem
   b) Feedback
   c) Environment
   d) All of the above

2) The probability density function is represented by
   a) Continuous variable
   b) Discrete variable
   c) Probability distributions for Continuous variables
   d) Probability distributions

3) What is the name of the operator in fuzzy set theory, which is found to be linguistic in nature?
   a) Lingual Variable
   b) Fuzz Variable
   c) Hedges
   d) None of the above

4) _____ represents the fuzzy logic
   a) IF-THEN rules
   b) IF-THEN-ELSE rules
   c) Both a & b
   d) None of the above

5) Uncertainty can be represented by _____
   a) Entropy
   b) Fuzzy logic
   c) Probability
   d) All of the above

6) Which of the following condition can directly influence a variable by all the others?
   a) Fully connected
   b) Local connected
   c) Partially connected
   d) None of the above

7) Based on _____ membership function can be used to solve empirical problems.
   a) Knowledge
   b) Learning
   c) Examples
   d) Experience

8) Which of the following takes input as an object described by a set of attributes?
   a) Graph
   b) Decision graph
   c) Tree
   d) Decision tree

9) FSI stands for_____

a) Fuzzy Inference System
b) Fuzzy Inferred System
c) Fuzzy Inter System
d) Fuzzy Interconnected system

10) A perceptron can be defined as _____

a) A double layer auto-associative neural network
b) A neural network with feedback
c) An auto-associative neural network
d) A single layer feed-forward neural network with pre-processing

❄❄❄❄❄❄❄

11

FUZZY INFERENCE SYSTEM - II

Unit Structure

11.1 OBJECTIVE

In a fuzzy inference system, an inference rule is a mapping from a set of premise facts to a conclusion fact. There are several approaches to fuzzy inference system design. For example, one approach is based on a set of rules whose premises are all combinations of the input fuzzy sets, while the conclusion is determined by the output fuzzy set. Another is based on a set of rules whose premises are all combinations of the input fuzzy sets, while the conclusion is determined by the complement (negation) of the output fuzzy set. Yet another approach is based on a set of rules whose premises are the input fuzzy sets, and whose conclusions are the complement of the output fuzzy set.

11.2 INTRODUCTION

Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the “IF…THEN” rules along with connectors “OR” or “AND” for drawing essential decision rules.

Characteristics of Fuzzy Inference System:

- The output from FIS is always a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have fuzzy output when it is used as a controller.
- A defuzzification unit would be there with FIS to convert fuzzy variables into crisp variables.

Combining Outputs into an Output Distribution:

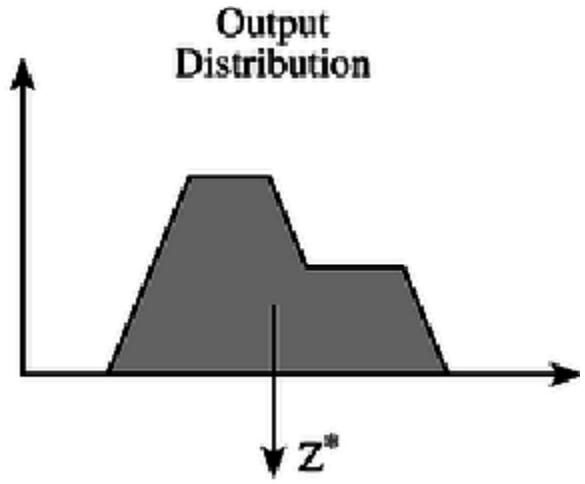
The outputs of all of the fuzzy rules must now be combined to obtain one fuzzy output distribution. This is usually, but not always, done by using the fuzzy. The output membership functions on the right hand side of the figure are combined using the fuzzy "or" to obtain the output distribution shown on the lower right corner of the figure.

Defuzzification of Output Distribution

In many instances, it is desired to come up with a single crisp output from a FIS. For example, if one was trying to classify a letter drawn by hand on a drawing tablet, ultimately the FIS would have to come up with a crisp number to tell the computer which letter was drawn. This crisp number is obtained in a process known as defuzzification.

$$z = \frac{\sum_{j=1}^{q} Z_j u_c(Z_j)}{\sum_{j=1}^{q} u_c(Z_j)}$$

where z is the center of mass and u_c is the membership in class c at value z_j. An example outcome of this computation is shown in below figure.



Fuzzy Query System:

A fuzzy query system is an interface to users to get information from the database using (quasi) natural language sentences. Many fuzzy query implementations have been proposed, resulting in slightly different languages. Although there are some variations according to the particularities of different implementations, the answer to a fuzzy query sentence is generally a list of records, ranked by the degree of matching.

Fuzzy Logic : Application

Aerospace

In aerospace, fuzzy logic is used in the following areas −

- Altitude control of spacecraft
- Satellite altitude control
- Flow and mixture regulation in aircraft deicing vehicles

Automotive

In automotive, fuzzy logic is used in the following areas −

- Trainable fuzzy systems for idle speed control
- Shift scheduling method for automatic transmission
- Intelligent highway systems
- Traffic control
- Improving efficiency of automatic transmissions

Business

In business, fuzzy logic is used in the following areas −

- Decision-making support systems
- Personnel evaluation in a large company

Defense

In defense, fuzzy logic is used in the following areas −

- Underwater target recognition
- Automatic target recognition of thermal infrared images
- Naval decision support aids
- Control of a hypervelocity interceptor
- Fuzzy set modeling of NATO decision making

Electronics

In electronics, fuzzy logic is used in the following areas −

- Control of automatic exposure in video cameras
- Humidity in a clean room
- Air conditioning systems
- Washing machine timing
- Microwave ovens
- Vacuum cleaners

Finance

In the finance field, fuzzy logic is used in the following areas −

- Banknote transfer control
- Fund management
- Stock market predictions

Industrial Sector

In industrial, fuzzy logic is used in following areas −

- Cement kiln controls heat exchanger control

- Activated sludge wastewater treatment process control
- Water purification plant control
- Quantitative pattern analysis for industrial quality assurance
- Control of constraint satisfaction problems in structural design
- Control of water purification plants

Manufacturing

In the manufacturing industry, fuzzy logic is used in following areas −

- Optimization of cheese production
- Optimization of milk production

Marine

In the marine field, fuzzy logic is used in the following areas −

- Autopilot for ships
- Optimal route selection
- Control of autonomous underwater vehicles
- Ship steering

Medical

In the medical field, fuzzy logic is used in the following areas −

- Medical diagnostic support system
- Control of arterial pressure during anesthesia
- Multivariable control of anesthesia
- Modeling of neuropathological findings in Alzheimer's patients
- Radiology diagnoses
- Fuzzy inference diagnosis of diabetes and prostate cancer

Securities

In securities, fuzzy logic is used in following areas −

- Decision systems for securities trading
- Various security appliances

Transportation

In transportation, fuzzy logic is used in the following areas −

- Automatic underground train operation
- Train schedule control
- Railway acceleration
- Braking and stopping

Pattern Recognition and Classification

In Pattern Recognition and Classification, fuzzy logic is used in the following areas −

- Fuzzy logic based speech recognition
- Fuzzy logic based
- Handwriting recognition
- Fuzzy logic based facial characteristic analysis
- Command analysis
- Fuzzy image search

Psychology

In Psychology, fuzzy logic is used in following areas −

- Fuzzy logic based analysis of human behavior
- Criminal investigation and prevention based on fuzzy logic reasoning

Fuzzy Query System

A fuzzy query system is an interface to users to get information from the database using (quasi) natural language sentences. Many fuzzy query implementations have been proposed, resulting in slightly different languages. Although there are some variations according to the particularities of different implementations, the answer to a fuzzy query sentence is generally a list of records, ranked by the degree of matching.

11.2.1 Truth Values and Tables in Fuzzy Logic

Fuzzy logic uses linguistic variables. The values of a linguistic variable are words or sentences in a natural or artificial language. For example, height is a linguistic variable if it rakes values such as tall, medium, short and so on. The linguistic variable provides approximate characterization of a complex problem. The name of the variable, the universe of discourse and a fuzzy subset of universe of discourse characterize a fuzzy variable. A linguistic variable is a variable of a higher order than a fuzzy variable and its values are taken to be fuzzy variables.

A linguistic variable is characterized by:

- name of the variable (x);
- term set of the variable t(x);
- syntactic rule for generating the values of x;

semantic rule for associating each value of x with its meaning.

Apart from the linguistic variables, there exists what are called as linguistic hedges (linguistic modifiers).

Examples of linguistic hedges In the fuzzy set "very tall", the word "very" is a linguistic hedge. A few popular linguistic hedges include: very, highly, slightly, moderately, plus, minus, fairly, rather.

Reasoning has logic as its basis, whereas propositions are text sentences expressed in any language and are generally expressed in a canonical form as **zisP**

Where the symbol of the subject and P the predicate designing the characteristics of the subject.

Example:

"London is in United Kingdom"

"London" the subject and "in United Kingdom" the predicate, which specifies a property of "London," i.e., its geographical location in United Kingdom.

Negation:

Every proposition has its opposite, called negation.

Logic Functions:

Truth cables define logic functions of two propositions.

Let X and Y be two propositions, either of which can be true or false.

The basic logic operations performed over the propositions are the following:

1. Conjunction (Λ) :X AND Y.
2. Disjunction (∨) :XOR Y.
3. Implication or conditional (=>): IF X THEN Y.
4. Bidirectional or equivalence **(<->):** X IF AND ONLY IF Y.

Inference Rules:

On the basis of these operations on propositions, inference rules can be formulated. Few inference rules are as follows:

$$[X \wedge (X \Rightarrow Y)] \Rightarrow Y$$

$$[\bar{Y} \wedge (X \Rightarrow Y)] \Rightarrow \bar{X}$$

$$[(X \Rightarrow Y) \wedge (Y \Rightarrow Z)] \Rightarrow (X \Rightarrow Z)$$

The above rules produce certain propositions that are always true irrespective of the truth values of propositions X and Y. Such propositions are called **tautologies.**

An extension of set-theoretic bivalence logic is the fuzzy logic where the truth values are terms of the linguistic variable "truth." The truth values of propositions in fuzzy logic are allowed to range over the unit interval [0, 1].

A truth value in fuzzy logic "very true" may be interpreted as a fuzzy set in [0, I]. The truth value of the proposition" Z is A," or simply the truth value of A, denoted by tv(A) is defined by a point in [0, 1] (called the numerical truth value) or a fuzzy set in [0, 1] (called the linguistic truth value).

The truth value of a proposition can be obtained from the logic operations of other propositions whose truth values are known. If tv(X) and tv(Y) are numerical truth values of propositions X and Y, respectively, then

$$tv(X \text{ AND } Y) = tv(X) \wedge tv(Y) = \min\{tv(X), tv(Y)\} \quad \text{(Intersection)}$$
$$tv(X \text{ OR } Y) = tv(X) \vee tv(Y) = \max\{tv(X), tv(Y)\} \quad \text{(Union)}$$
$$tv(\text{NOT } X) = 1 - tv(X) \quad \text{(Complement)}$$
$$tv(X \Rightarrow Y) = tv(X) \Rightarrow tv(B) = \max\{1 - tv(X), \min[tv(X), tv(Y)]\}$$

11.3 FUZZY PROPOSITIONS

Fuzzy propositions are assigned to fuzzy sets. Suppose a fuzzy proposition ‘P’ is assigned to a fuzzy set ‘A’, then the truth value of the proposition is proposed by T (P) = $\mu_A(x)$ where $0 \leq \mu_A(x) \leq 1$

Therefore, truthness of a proposition P is membership value of x in fuzzy set A.

The logical connectives like disjunction, conjunction, negation and implication are also defined on fuzzy propositions.

Let, a fuzzy proposition ‘P’ is defined on a fuzzy set A Q is defined on fuzzy set B

Conjunction

P / Q : x is A and B
T(P / Q) = Min [T(P), T(Q)]

Negation

$T(P^c) = 1 - T(P)$

Disjunction

P V Q : x in A or B
T (P V Q) = Max [T(P), T(Q)]

Implication

$P \rightarrow Q$: x is A then x is B
$T(P \rightarrow Q) = T (P^c \vee Q) = Max [T(P^c, T(Q)]$

If P is a proposition defined on set A on universe of discourse X and Q is another proposition defined on set B on universe of discourse Y, then the implication $P \rightarrow Q$ can be represented by the relation R

$R = (A \times B) \cup (A^c \times Y)$ = If A then B

If $x \in A$, where $x \in X$ and $A \subset X$ then $y \in B$, where $y \in Y$ and $B \subset Y$

Implication of Classical Logic:-

Properties P and Q are given by P : $x \in A$, where A is defined on x. Q : $y \in B$, where B is defined on y.

Then the implication $P \rightarrow Q$ is represented in set theoretic form by a relation R as
$R = (A \times B) \cup (A^c \times Y)$

The implication is equivalent to linguistic rule form, if $x \in A$ then $y \in B$.

For the classical predicate logical rule, $(P \rightarrow Q) \vee (P^c \rightarrow S)$ the linguistic rule form is, if x is A then y is B, else y is C. Where C is defined as S : y is C , $C \subset Y$.

The above linguistic rule form is decomposed as if (x is A) then (y is B) or if (x is A) then (y is not B)

In set theoretic form it can be represented by the relation $R = (A \times B) \cup (A^c \times C)$

The characteristic function for above compound proposition is given by

$$\chi_R(x, y) = Max\left[\min\left(\chi_A(x), \chi_B(y)\right), \min\left(1 - \chi_A(x), \chi_C(y)\right)\right]$$

For example, suppose we have two universe of discourse X and Y X = {1, 2, 3, 4} Y = {1, 2, 3, 4, 5, 6}

X is the universe of normalized temperatures. Y is the universe of normalized pressures.

Two crisp sets A and B defined on universe of discourses X and Y are A={2, 3} ; B={3,4} for the deductive inference if A and B, find the relational matrix R.

sets A and B in Zedah's notations are given by A = {0/1 + 1/2 + 1/3 + 0/4}
B = {0/1 + 0/2 + 1/3 + 1/4 + 0/5 + 0/6}

for the deductive inference if A then B, the set theoretic form is given by the relation $R = (A \times B) \cup (A^c \times Y)$

The fuzzy propositions are as follows:

1. **Fuzzy predicates:** In fuzzy logic the predicates can be fuzzy, for example, tall, short, quick. Hence, we have proposition like "Peter is tall."
2. **Fuzzy-predicate modifiers:** In fuzzy logic, there exists a wide range of predicate modifiers that act as hedges. For example, very, fairly, moderately, rather, slightly. These predicate modifiers are necessary for generating the values of a linguistic variable. An example can be the proposition "Climate is moderately cool," where" moderately" is the fuzzy predicate modifier.
3. **Fuzzy quantifiers:** The fuzzy quantifiers such as most, several, many, frequently are used in fuzzy logic. Employing these, we can have proposition like "Many people are educated." A fuzzy quantifier can be interpreted as a fuzzy number or a fuzzy proposition, which provides an imprecise characterization of the cardinality of one or more fuzzy or non-fuzzy sets. Fuzzy quantifiers can be used to represent the meaning of propositions containing probabilities; as a result, they can be used to manipulate probabilities within fuzzy logic.
4. **Fuzzy qualifiers:** There are four modes of qualification in fuzzy logic, which are as follows:

   (i) **Fuzzy truth qualification:** It is expressed as "x is τ ," in which τ is a fuzzy truth value. A fuzzy truth value claims the degree of truth of a fuzzy proposition.

   Example

   (Paul is Young) is NOT VERY True.

   Here the qualified proposition is (Paul is Young) and the qualifying fuzzy truth value is "NOT Very True."

   (ii) Fuzzy probability qualification: It is denoted as "x is λ "where, λ is fuzzy probability. In conventional logic, probability is either numerical or an interval. In fuzzy logic, fuzzy probability is expressed by terms such as likely, very likely, unlikely, around and so on.

   Example:

   (Paul is Young) is Likely.

   Here the qualifying fuzzy probability is "Likely." These probabilities may be interpreted as fuzzy numbers, which may be manipulated using fuzzy arithmetic.

   (iii) Fuzzy possibility qualification:

   It is expressed as "x is π",where πis a fuzzy possibility and can be of the following forms: possible, quite possible, almost impossible. These values can be interpreted as labels of fuzzy subsets of the real line.

Example:

(Paul is Young) is Almost Impossible.

Here the qualifying fuzzy possibility is "Almost Impossible."

(iv) Fuzzy usuality qualification: It is expressed as "usually (X) =usually (X is F),"in which the subject X is a variable taking values in a universe of discourse U and the predicate F is a fuzzy subset of U and interpreted as a usual value of X denoted by U(X) = F. The propositions that are usually true or the events that have high probability of occurrence are related by the concept of usuality qualification.

11.4 FORMATION OF RULES

The formation of rules is in general the canonical rule formation. For any linguistic variable, there are three general forms in which the canonical rules can be formed. They are:

(1) Assignment statements
(2) Conditional statements
(3) Unconditional statements

(1) Assignment statements These statements are those in which the variable is assignment with the value. The variable and the value assigned are combined by the assignment operator "=." The assignment statements are necessary in forming fuzzy rules. The value to be assigned may be a linguistic term.

Fuzzy Rule-Based System

The examples of this type of statements are:

y = low,

Sky color = blue,

Climate = hot a = 5 p = q + r

Temperature = high

The assignment statement is found to restrict the value of a variable to a specific equality.

(2) Conditional statements

In this statements, some specific conditions are mentioned, if the conditions are satisfied then it enters the following statements, called as restrictions.

If x = y Then both are equal,

If Mark > 50 Then pass,

If Speed > 1, 500 Then stop.

These statements can be said as fuzzy conditional statements, such as If condition C Then restriction F

(3) Unconditional statements

There is no specific condition that has to be satisfied in this form of statements. Some of the unconditional statements are:

Go to F/o

Push the value

Stop

The control may be transferred without any appropriate conditions. The unconditional restrictions in the fuzzy form can be:

R1 : Output is B1 AND

R2 : Output is B2 AND ..., etc.

where B1 and B2 are Fuzzy consequents. Both conditional and unconditional statements place restrictions on the consequent of the rule-based process because of certain conditions

The general way of representing human knowledge is by forming natural language expressions given by

IF antecedent THEN consequent.

The above expression is referred to as the IF - THEN rule based form. There are three general forms that exist for any linguistic variable. They are:

(a) Assignment statements;

(b) Conditional statements;

(c) Unconditional statements.

1. Assignment statements:

They are of the form

y =small

Orange colour = orange

a=s

Paul is not tall and not very short Climate = autumn

Outside temperature = normal

These statements utilize "=" for assignment.

2. Conditional statements:

The following are some examples.

- IF y is very cool THEN stop.
- IF A is high THEN B is low ELSE B is not low.
- IF temperature is high THEN climate is hot.

The conditional statements use the "IF - THEN" rule-based form.

3. Unconditional statements:

They can be of the form

- Go to sum.
- Stop.
- Divide by a.
- Turn the pressure low

11.5 DECOMPOSITION OF RULES (COMPOUND RULE)

A compound rule is a collection of many simple rules combined together. Any compound rule structure may be decomposed and reduced to a number of simple canonical rule forms. The rules are generally based on natural language representations.

The following are the methods used for decomposition of compound linguistic rules into simple canonical rules.

- Multiple conjunctive antecedents
- Multiple disjunctive antecedents
- Conditional statements (ELSE and UNLESS)
- Nested IF-THEN rules

11.6 AGGREGATION OF FUZZY RULES

The rule-based system involves more than one rule. Aggregation of rules is the process of obtaining the overall consequents from the individual consequents provided by each rule.

The following two methods are used for aggregation of fuzzy rules:

1. Conjunctive system of rules:

For a system of rules to be jointly satisfied, the rules are connected by "and" connectives. Here, the aggregated output, y, is determined by the fuzzy intersection of all individual rule consequents, yi, where i=1ton, as

Y= y1, y2……and yn

This aggregated output can be defined by the membership function

$$\mu_y(y) = \min[\mu_{y_1}(y), \mu_{y_2}(y), \ldots, \mu_{y_n}(y)] \quad \text{for} \quad y \in Y$$

2. Disjunctive system of rules:

In this case, the satisfaction of at least one rule is required. The rules are connected by "or" connectives. Here, the fuzzy union of all individual rule contributions determines the aggregated output, as

Y= y1 or y2 or…. Or yn

Again, it can be defined by the membership function

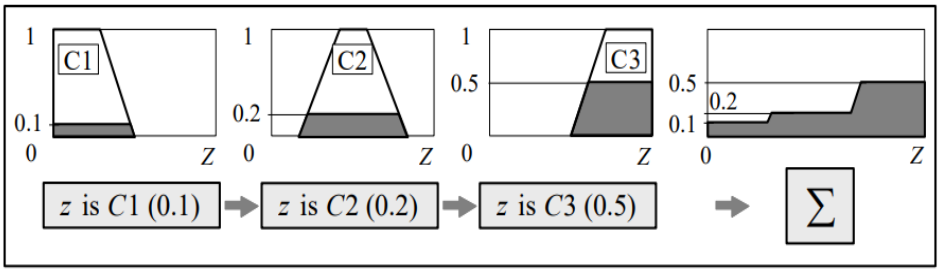
$$\mu_y(y) = \max[\mu_{y_1}(y), \mu_{y_2}(y), \ldots \mu_{y_n}(y)] \text{ for } y \in Y$$

Aggregation of The Rule Outputs

1. Aggregation is the process of unification of the outputs of all rules.
2. We take the membership functions of all rule consequents previously clipped or scaled and combine them into a single fuzzy set.
3. There are several defuzzification methods, but probably the most popular one is the centroid technique.
4. It finds the point where a vertical line would slice the aggregate set into two equal masses. Mathematically this centre of gravity

$$COG = \frac{\int_a^b \mu_A(x)\, x\, dx}{\int_a^b \mu_A(x)\, dx}$$

Aggregation of the rule outputs



❄❄❄❄❄❄❄

12

FUZZY INFERENCE SYSTEM - III

Unit Structure

12.1 FUZZY INFERENCE SYSTEM (FIS)

Fuzzy rule-based systems, fuzzy models, and fuzzy expert systems are generally known as inference systems. The key unit of a fuzzy logic system is FIS. The primary work of this system is decision making.

FIS uses "IF ... THEN" rules along with connectors "OR" or "AND" for making necessary decision rules. The input to FlS may be fuzzy or crisp, but the output from FIS is always a fuzzy set. When FlS is used as a controller, it is necessary to have crisp output. Hence, there should be a defuzzification unit for converting fuzzy variables into crisp variables along FIS.

Applications of FIS

A fuzzy inference system is used in different fields, for example, information order, choice examination, master system, time arrangement forecasts, advanced mechanics, and example acknowledgment. It is otherwise called a fuzzy rule-based system, fuzzy model, fuzzy logic controller, fuzzy expert system, and fuzzy associative memory.

It is the vital unit of a fuzzy logic system that deals with decision-making and choosing essential tasks. It utilizes the “IF… . At that point” leads alongside the connectors “AND” “OR” to draw fundamental choice standards.

Characteristics of Fuzzy Inference system

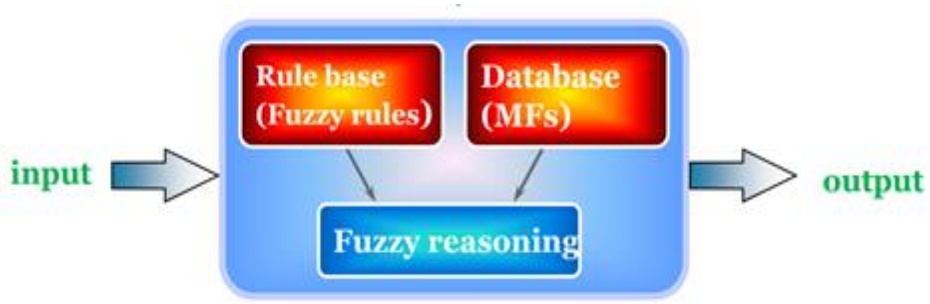
- The yield from FIS is consistently a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have a fuzzy output when it is used as a controller.
- A defuzzification unit would accompany the FIS to convert the fuzzy variable into a crisp variable.

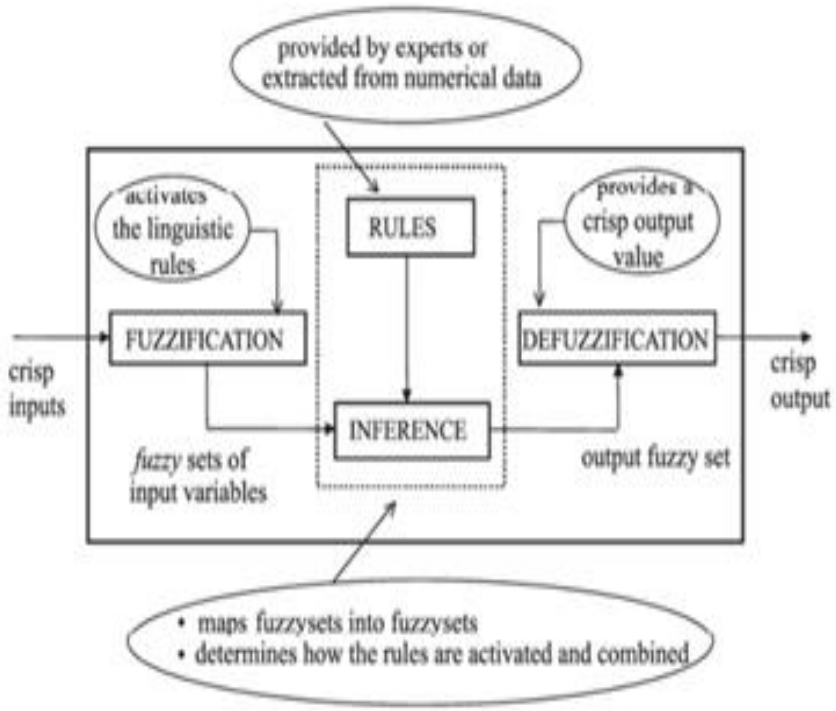
Structure of Fuzzy Inference System

The essential structure of a fuzzy inference system comprises three entities:

- A rule base containing fuzzy rules

- A database (or dictionary), containing the participation functions utilized in the fuzzy rules.
- A reasoning mechanism performing the induction made upon the guidelines and the facts given to infer a reasonable output or conclusion.





What is Defuzzification?

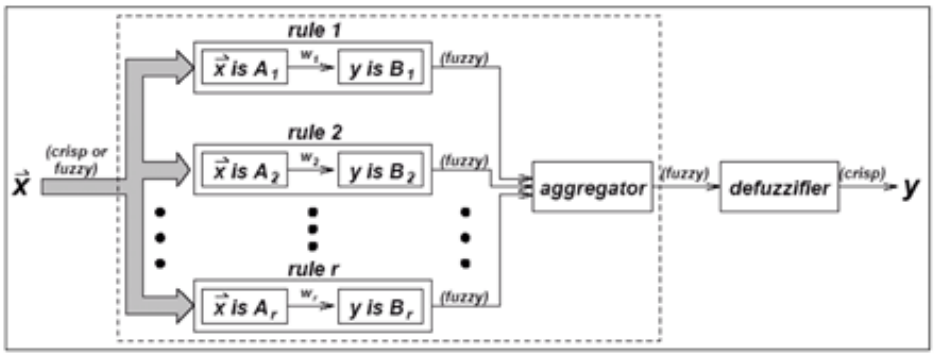
Defuzzification is the extraction of a value representing a fuzzy set.
Defuzzification methods:

1. Centroid of area
2. Bisector of area
3. Mean of max
4. Smallest of max
5. Largest of max

It is mandatory to have a crisp output in some instances where we use an interference system as a controller.

Fuzzy Inference System Inputs and Outputs

- The fundamental **fuzzy inference system** can take either fuzzy inputs or crisp inputs, yet the yield it produces is quite often fuzzy sets.
- Sometimes it is important to have a crisp output, particularly in a situation where a **fuzzy inference system** is utilized as a controller.
- Therefore, we need a technique of defuzzification to extricate a crisp value to represent a fuzzy set.



Block diagram for a Fuzzy Inference System with Crisp Output

Popular Fuzzy Inference Systems (Fuzzy Models)

1. Mamdani Fuzzy Models
2. Sugeno Fuzzy Models

The core difference between these fuzzy inference systems is in the consequents of their fuzzy rules, and their distinguishing conglomeration and defuzzification procedures.

1. Ebrahim Mamdani Fuzzy Model

This is the most used fuzzy inference system.

Professor Mamdani fabricated one of the primary fuzzy systems to control a steam motor and kettle mix. He applied fuzzy rules put forth by experienced human operators.

Steps for Computing the Output

Following advances should be followed to compute the output from this FIS

Step 1: Deciding a bunch of fuzzy principles

Step 2: Fuzzifying the inputs with the elements of info participation

Step 3: Amalgamating the fuzzified inputs according to the fuzzy guidelines to discover a standard strength

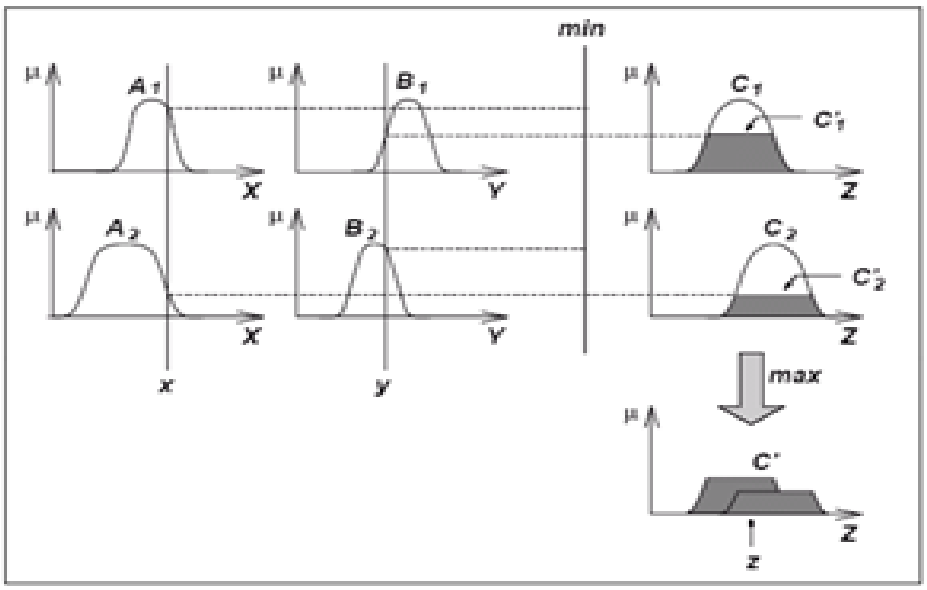
Step 4: Finding the aftereffect of the standard by summarizing the standard strength with the yield participation work

Step 5: Combining the outcomes to get the yield conveyance

Step 6: Performing defuzzification of the output dispersion

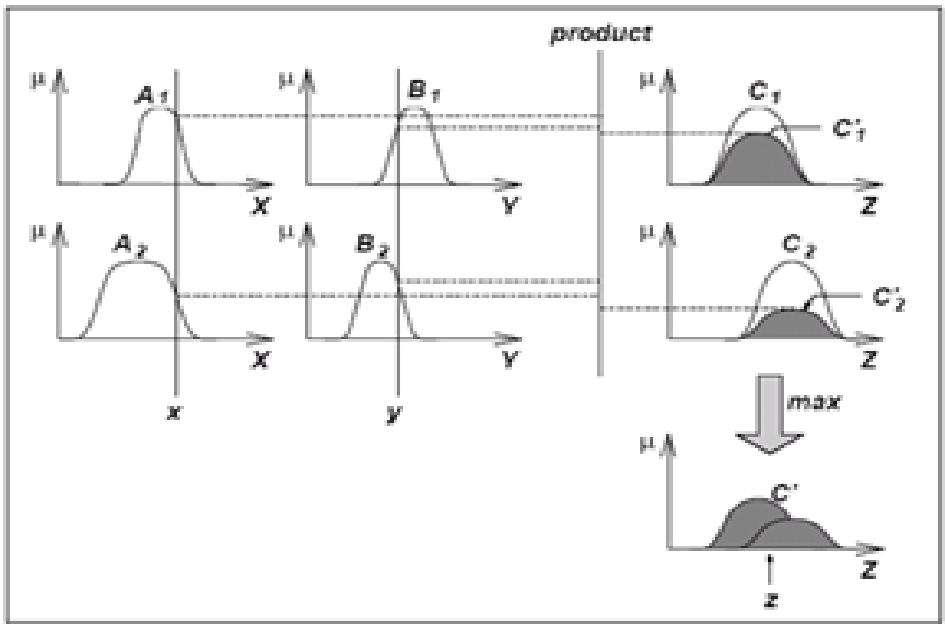
Two Rules Mamdani with Min and Max Operators

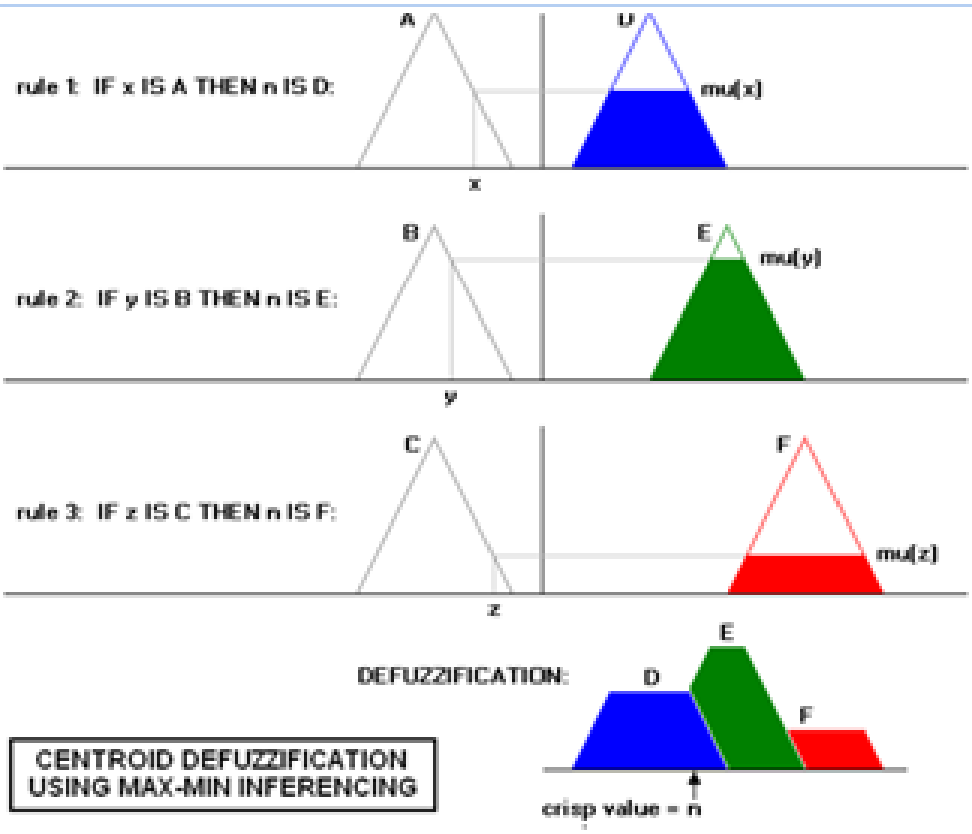
The Mamdani FIS using **min** and **max** for T-norms and S-norms, subject to two crisp inputs x and y.



Two Rules Mamdani FIS with Max and Product Operators

The Mamdani FIS using **product** and **max** for T-norms and S-norms, subject to two crisp inputs x and y.





Mamdani composition of three SISO fuzzy outputs

2. Sugeno Fuzzy Model

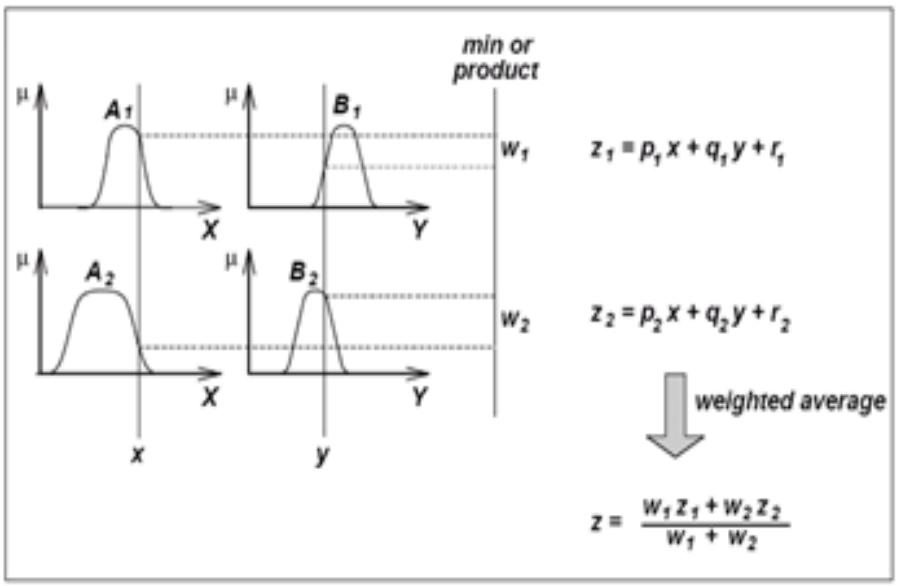
This model was proposed by Takagi, Sugeno, and Kang.

For developing a scientific approach to generate fuzzy rules from a given set of input-output data.
The format of this rule is given as:
IF x is A and y is B; Z= f(x,y)
Here, AB is fuzzy sets in antecedents, and z= f(x, y) is a crisp function within the consequent.
The most commonly used zero-order Sugeno fuzzy model applies fuzzy rules within the following form:
IF x is A AND y is B; z is k
Where k is a constant

In this case, the output of every fuzzy rule is constant, and every consequent membership function is represented by singleton spikes.

So,

- First-order Sugeno fuzzy model: f(x, y) – first-order polynomial
- Zero-order Sugeno fuzzy model: f – constant



Fuzzy reasoning procedure for a first-order Sugeno Fuzzy Model

The **fuzzy inference system** under Sugeno Fuzzy method works in the following way-

Step 1: Fuzzifying the inputs- the inputs of the system are made fuzzy.

Step 2: Applying the fuzzy operator- the fuzzy operators must be applied to get the output.

Rule Format

The rule format of Sugeno form-

If 7 = x and 9 = y; output is z = ax+by+c

The Sugeno **fuzzy inference system** is very similar to the Mamdani method.

Only change a rule consequent: instead of a fuzzy set, used a mathematical function of the input variable.

How to Decide Whether to Apply- Mamdani or Sugeno Fuzzy Inference System?

- Mamdani technique is broadly acknowledged for capturing expert knowledge and information. It allows us to depict the skill in a more instinctive, more human-like way.

However, Mamdani type fuzzy inference entails a considerable computational burden.

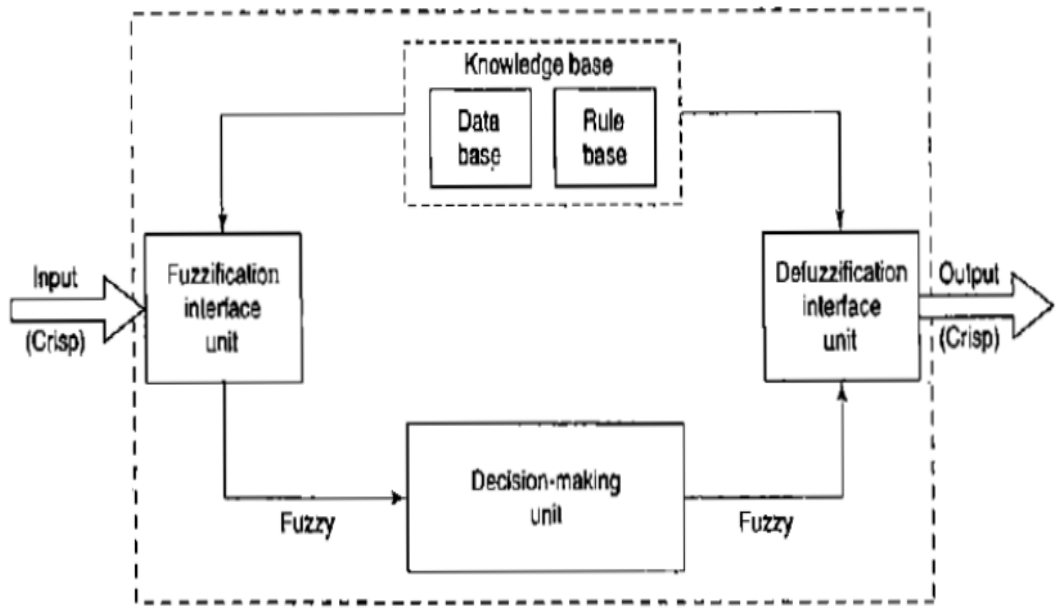
- On the other hand, the Sugeno method is computationally feasible. It functions effectively with advancement and versatile procedures making it exceptionally alluring in versatile issues, particularly for dynamic nonlinear frameworks.

Fuzzy Inference Systems Advantages

Fuzzy Inference System	Advantages
Mamdani	• Intuitive • Well-suited to human inputs • More interpretable and rule-based • Has widespread acceptance
Sugeno	• Computationally efficient • Functions well with linear techniques, like PID control • Functions with optimization and adaptive techniques • Guarantees output surface continuity • Well-suited to mathematical analysis

12.2 CONSTRUCTION AND WORKING PRINCIPLE OF FIS

1. A rule base that contains numerous fuzzy IF-THEN rules.
2. A database that defines the membership functions of fuzzy sets used in fuzzy rules.
3. Decision making unit that performs operations on the rules.
4. Fuzzification interface unit that converts the crisp quantities into fuzzy quantities.
5. Defuzzification interface unit that converts the fuzzy quantities into crisp quantities.



Working methodology of FIS:

Initially, in the fuzzif1cation unit, the crisp input is converted into a fuzzy input. Various fuzzification methods are employed for this. After this process, rule base is formed. Database and rule base are collectively called the knowledge base. Finally, defuzzification process is carried out to produce crisp output. Mainly, the fuzzy rules are formed in the rule base and suitable decisions are made in the decision-making unit.

12.3 METHODS OF FI:

Let us now discuss the different methods of FIS. Following are the two important methods of FIS, having different consequent of fuzzy rules −

- Mamdani Fuzzy Inference System
- Takagi-Sugeno Fuzzy Model (TS Method)

Mamdani Fuzzy Inference System

This system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

Steps for Computing the Output

Following steps need to be followed to compute the output from this FIS −

Step 1 Set of fuzzy rules need to be determined in this step.

Step 2 In this step, by using input membership function, the input would be made fuzzy.

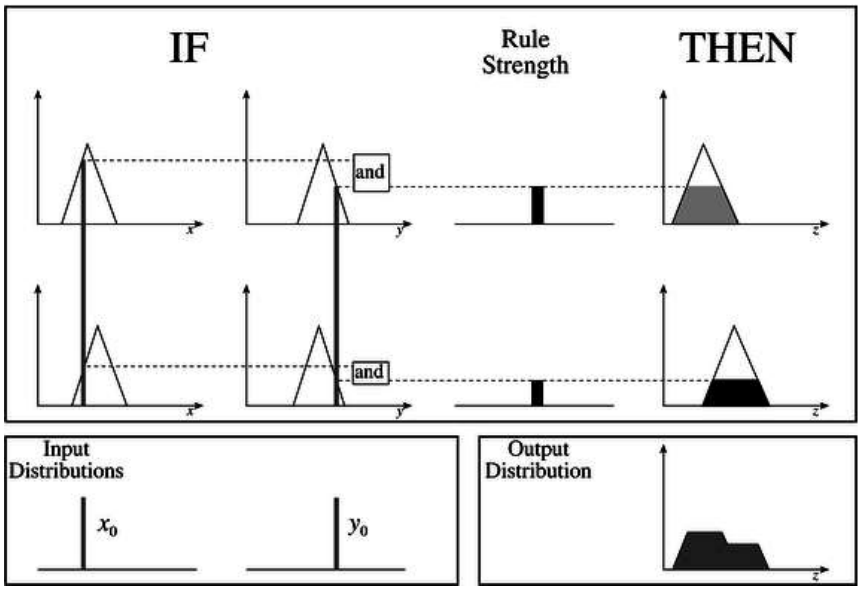
Step 3 Now establish the rule strength by combining the fuzzified inputs according to fuzzy rules.

Step 4 In this step, determine the consequent of rule by combining the rule strength and the output membership function.

Step 5 For getting output distribution combine all the consequents.

Step 6 Finally, a defuzzified output distribution is obtained.

Following is a block diagram of Mamdani Fuzzy Interface System.



Takagi-Sugeno Fuzzy Model (TS Method)

This model was proposed by Takagi, Sugeno and Kang in 1985. Format of this rule is given as −

IF x is A and y is B THEN Z = f(x,y)

Here, *AB* are fuzzy sets in antecedents and $z = f(x,y)$ is a crisp function in the consequent.

Fuzzy Inference Process

The fuzzy inference process under Takagi-Sugeno Fuzzy Model (TS Method) works in the following way −

Step 1: **Fuzzifying the inputs** − Here, the inputs of the system are made fuzzy.

Step 2: **Applying the fuzzy operator** − In this step, the fuzzy operators must be applied to get the output.

Rule Format of the Sugeno Form

The rule format of Sugeno form is given by −
if 7 = x and 9 = y then output is z = ax+by+c

12.2.1 Comparison between the two methods

Let us now understand the comparison between the Mamdani System and the Sugeno Model.

- **Output Membership Function** − The main difference between them is on the basis of output membership function. The Sugeno output membership functions are either linear or constant.
- **Aggregation and Defuzzification Procedure** − The difference between them also lies in the consequence of fuzzy rules and due to the same their aggregation and defuzzification procedure also differs.

- **Mathematical Rules** − More mathematical rules exist for the Sugeno rule than the Mamdani rule.
- **Adjustable Parameters** − The Sugeno controller has more adjustable parameters than the Mamdani controller.

12.4 OVERVIEW OF FUZZY EXPERT SYSTEM

What is a Fuzzy Expert System?

Put as simply as possible, a fuzzy expert system is an expert system that uses fuzzy logic instead of Boolean logic. In other words, a fuzzy expert system is a collection of membership functions and rules that are used to reason about data. Unlike conventional expert systems, which are mainly symbolic reasoning engines, fuzzy expert systems are oriented toward numerical processing.

The rules in a fuzzy expert system are usually of a form similar to the following:

if x is low and y is high then z = medium

where x and y are input variables (names for-know data values), z is an output variable (a name for a data value to be computed), low is a membership function (fuzzy subset) defined on x, high is a membership function defined on y, and medium is a membership function defined on z. The part of the rule between the "if" and "then" is the rule's _premise_ or _antecedent_. This is a fuzzy logic expression that describes to what degree the rule is applicable. The part of the rule following the "then" is the rule's _conclusion_ or _consequent_. This part of the rule assigns a membership function to each of one or more output variables. Most tools for working with fuzzy expert systems allow more than one conclusion per rule.

A typical fuzzy expert system has more than one rule. The entire group of rules is collectively known as a _rulebase_ or _knowledge base_.

The Inference Process

With the definition of the rules and membership functions in hand, we now need to know how to apply this knowledge to specific values of the input variables to compute the values of the output variables. This process is referred to as _inferencing_. In a fuzzy expert system, the inference process is a combination of four subprocesses: _fuzzification_, _inference_, _composition_, and _defuzzification_. The defuzzification subprocess is optional.

For the sake of example in the following discussion, assume that the variables x, y, and z all take on values in the interval [0, 10], and that we have the following membership functions and rules defined.

low(t) = 1 - t / 10
high(t) = t / 10
rule 1: if x is low and y is low then z is high

rule 2: if x is low and y is high then z is low
rule 3: if x is high and y is low then z is low
rule 4: if x is high and y is high then z is high

Notice that instead of assigning a single value to the output variable z, each rule assigns an entire fuzzy subset (low or high).

Note:

1. In this example, low(t)+high(t)=1.0 for all t. This is not required, but it is fairly common.
2. The value of t at which low(t) is maximum is the same as the value of t at which high(t) is minimum, and vice-versa. This is also not required, but fairly common.
3. The same membership functions are used for all variables. This isn't required, and is also *not* common.

Fuzzification

In the fuzzification subprocess, the membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule premise. The degree of truth for a rule's premise is sometimes referred to as its _alpha_. If a rule's premise has a nonzero degree of truth (if the rule applies at all...) then the rule is said to _fire_.

For example:

x y low(x) high(x) low(y) high(y) alpha1 alpha2 alpha3 alpha4

0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
0.0	3.2	1.0	0.0	0.68	0.32	0.68	0.32	0.0	0.0
0.0	6.1	1.0	0.0	0.39	0.61	0.39	0.61	0.0	0.0
0.0	10.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
3.2	0.0	0.68	0.32	1.0	0.0	0.68	0.0	0.32	0.0
6.1	0.0	0.39	0.61	1.0	0.0	0.39	0.0	0.61	0.0
10.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
3.2	3.1	0.68	0.32	0.69	0.31	0.68	0.31	0.32	0.32
3.2	3.3	0.68	0.32	0.67	0.33	0.67	0.33	0.32	0.32
10.0	10.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0

Inference

In the inference subprocess, the truth value for the premise of each rule is computed, and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule.

I've only seen two _inference methods_ or _inference rules_: _MIN_ and _PRODUCT_. In MIN inferencing, the output membership function is clipped off at a height corresponding to the rule premise's computed degree of truth. This corresponds to the traditional interpretation of the fuzzy logic

AND operation. In PRODUCT inferencing, the output membership function is scaled by the rule premise's computed degree of truth.

Due to the limitations of posting this as raw ASCII, I can't draw you a decent diagram of the results of these methods. Therefore, I'll give the example results in the same functional notation I used for the membership functions above.

For example, let's look at rule 1 for x = 0.0 and y = 3.2. As shown in the table above, the premise degree of truth works out to 0.68. For this rule, MIN inferencing will assign z the fuzzy subset defined by the membership function:

rule1(z) = { z / 10, if z <= 6.8

0.68, if z >= 6.8 }

For the same conditions, PRODUCT inferencing will assign z the fuzzy subset defined by the membership function:

rule1(z) = 0.68 * high(z)

= 0.068 * z

Note: I'm using slightly nonstandard terminology here. In most texts, the term "inference method" is used to mean the combination of the things I'm referring to separately here as "inference" and "composition." Therefore, you'll see terms such as "MAX-MIN inference" and "SUM-PRODUCT inference" in the literature. They mean the combination of MAX composition and MIN inference, or SUM composition and PRODUCT inference respectively, to use my terminology. You'll also see the reverse terms "MIN-MAX" and "PRODUCT-SUM" - these mean the same things as the reverse order. I think it's clearer to describe the two processes separately.

Composition

In the composition subprocess, all of the fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable.

I'm familiar with two _composition rules_: _MAX composition_ and _SUM composition_. In MAX composition, the combined output fuzzy subset is constructed by taking the pointwise maximum over all of the fuzzy subsets assigned to the output variable by the inference rule. In SUM composition the combined output fuzzy subset is constructed by taking the pointwise sum over all of the fuzzy subsets assigned to the output variable by the inference rule. Note that this can result in truth values greater than one! For this reason, SUM composition is only used when it will be followed by a defuzzification method, such as the CENTROID method, that doesn't have a problem with this odd case.

For example, assume x = 0.0 and y = 3.2. MIN inferencing would assign the following four fuzzy subsets to z:

rule1(z) = { z / 10, if z <= 6.8
0.68, if z >= 6.8 }
rule2(z) = { 0.32, if z <= 6.8
1 - z / 10, if z >= 6.8 }
rule3(z) = 0.0
rule4(z) = 0.0

MAX composition would result in the fuzzy subset:

fuzzy(z) = { 0.32, if z <= 3.2
z / 10, if 3.2 <= z <= 6.8
0.68, if z >= 6.8 }

PRODUCT inferencing would assign the following four fuzzy subsets to z:

rule1(z) = 0.068 * z
rule2(z) = 0.32 - 0.032 * z
rule3(z) = 0.0
rule4(z) = 0.0

SUM composition would result in the fuzzy subset:

fuzzy(z) = 0.32 + 0.036 * z

Where are Fuzzy Expert Systems Used?

To date, fuzzy expert systems are the most common use of fuzzy logic. They are used in several wide-ranging fields, including:

- Linear and nonlinear control.
- Pattern recognition.
- Financial systems.

Multiple Choice Questions

1) Which of the following is considered while determining the nature of the learning problem?
   a) Problem
   b) Feedback
   c) Environment
   d) All of the above
2) The probability density function is represented by
   a) Continuous variable
   b) Discrete variable
   c) Probability distributions for Continuous variables
   d) Probability distributions
3) What is the name of the operator in fuzzy set theory, which is found to be linguistic in nature?
   a) Lingual Variable
   b) Fuzz Variable
   c) Hedges
   d) None of the above

4) _____ represents the fuzzy logic
   a) IF-THEN rules
   b) IF-THEN-ELSE rules
   c) Both a & b
   d) None of the above
5) Uncertainty can be represented by _____
   a) Entropy
   b) Fuzzy logic
   c) Probability
   d) All of the above
6) Which of the following condition can directly influence a variable by all the others?
   a) Fully connected
   b) Local connected
   c) Partially connected
   d) None of the above
7) Based on _____ membership function can be used to solve empirical problems.
   a) Knowledge
   b) Learning
   c) Examples
   d) Experience
8) Which of the following takes input as an object described by a set of attributes?
   a) Graph
   b) Decision graph
   c) Tree
   d) Decision tree
9) FSI stands for_____
   a) Fuzzy Inference System
   b) Fuzzy Inferred System
   c) Fuzzy Inter System
   d) Fuzzy Interconnected system
10) A perceptron can be defined as _____
   a) A double layer auto-associative neural network
   b) A neural network with feedback
   c) An auto-associative neural network
   d) A single layer feed-forward neural network with pre-processing

❄❄❄❄❄❄❄

13

GENETIC ALGORITHM

Unit Structure

13.0 OBJECTIVE

This chapter will able you to understand the following concept

- What is Genetic algorithm
- Approaches of genetic algorithm
- Pseudocode of genetic algorithm
- Types of Operators of genetic algorithm
- Different characteristics of genetic algorithm
- Limitations of genetic algorithm
- Difference between genetic algorithm and traditional algorithm

13.1 INTRODUCTION

A Genetic algorithm is dealing with reproduction, for this one need to combine feature of two parent to reproduce the new offspring's. it is not modifying the feature of one parent it is combining the feature of both parent to generate a new offspring. Hence the new offspring is much better than the old one as they have inherited properties of both the parents, this approach is scholastic beam search and this is the variant of Genetic algorithm. Can be done with some steps like population, selection, crossover, mutation. It has different approaches as well as different operators with which we will combine the feature of parents.

In scholastic beam search a set of k randomly generated states, called population. In population the words, numbers are used to represent the population. Most of the time binary number is used to represent the population i.e. 0 and 1. Each individual number represent chromosome of the selected population. Production of the next generation depends on the fitness function of the selected parents, the properties of the both parents are inherited by the new offspring. In this scholastic method selection will be depend on the fitness function of the chromosome. Crossover point should be select for the mutation purpose

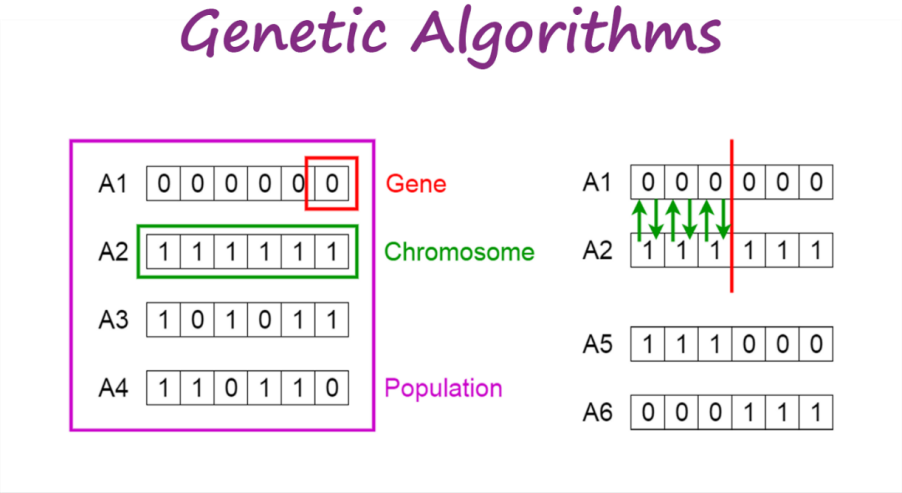
13.2 GENETIC ALGORITHM

A theory of natural evolution coined by Charles Darwin's and named as genetic algorithm in which search heuristic is used. Natural selection process is used in which it selected fittest individuals for reproduction in order to produce offspring of the next generation.

To find a best solution of some problem we will check the space of all feasible solutions which are all possible to solve the problem is called search space. Every point in search space represents one possible solution. Every possible solution can be checked by its value which is also called as fitness for the problem. Using GA one will try to find the best solution among a number of possible solutions - represented by one point in the search space.

When we will be finding solution of the problem at the same time we are checking its Looking for a minimum or maximum value in the search space. Sometime the search space may be well defined, or sometime we know only a few points in the search space. To ease of use in GA, the process of finding solutions generates other possible points for further evaluation.

In some cases, the search can be very complicated sometimes. Hence one may not know where to start and how to proceed for solution. Many methods are available to finding a suitable solution, but this will not surely have said that the outcome of any solution will be best solution. In some methods hill climbing, tabu search, simulated annealing and the genetic algorithm this technique I used. These methods salutations are often considered as good solutions.



13.3 APPROACHES OF GENETIC ALGORITHM

Natural selection process is used in which it selected fittest individuals for reproduction in order to produce offspring of the next generation. From the process new offspring inherits the properties of the parents and will be added to the next generation. If the properties of parents better in fitness, then their offspring properties will be additional as well as better than parents hence they have a better chance at surviving. This is iterative process until it reaches to end, and in result a new generation with the fittest individuals will be found. This process is used for a search problem.

Five approaches of genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

13.3.1 Initial Population

In this process we will starts with a set of individuals which is called a Population. Each individual can be a solution of the problem.

An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).

Using a string the set of genes are used to represent an individual in genetic algorithm, in terms of an alphabet. In binary values 1 and 0 the strings are defined. A chromosome can be made from encoding of genes.

13.3.2 Fitness Function

How fit an individual is (the ability of an individual to compete with other individuals) determines the fitness function. With fitness score we can determine each individual. While reproducing the probability of an individual will be selected and based on its fitness score the reproduction quality will judge.

13.3.3 Selection

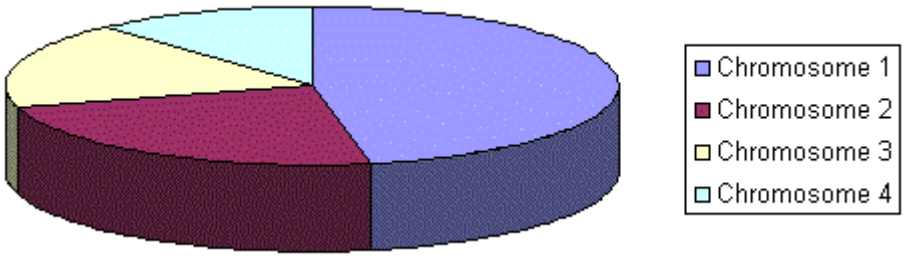
For producing next generation, the selection is very important as it selects the individual with its fitness for reproduction. For making new generation two pairs (parent) are required from the initial population according to their fitness score. The good fitness score ca has higher chance of selection for reproduction.

As we already know in Genetic Algorithm, selection of chromosomes is done form the population and make them as parents for crossover. According to Darwin's theory selection of chromosomes are based on fitness function of the selected chromosomes to reproduce the new offspring's with better properties so that it can survive in the environment.

There are many ways to select the best chromosomes. They are such as roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.

Roulette Wheel Selection

The fitness function of the parents is deciding factor in this selection method. If they have better fitness function the chances of selecting these chromosomes are higher. In this selection method e all the chromosomes in the population are placed. According to the value of fitness function proportionate the section in the roulete wheel indicate. Hence the high value of fitness function is proportionate to the bigger section of the Roulette Wheel shows in following picture.



As we can see the bigger section contains the higher value of chromosomes hence they have higher chance to get selected for crossover. So the selection can be more times from the particular section.

Following are the steps of algorithm.

1. **[Sum]** sum of the all chromosome fitnesses in population - sum ***S***.
2. **[Select]** Generate random number from the interval ***(0,S)*** - ***r***.

3. **[Loop]** Go through the population and sum the fitnesses from ***0*** - sum ***s***. When the sum ***s*** is greater then ***r***, stop and return the chromosome where you are.

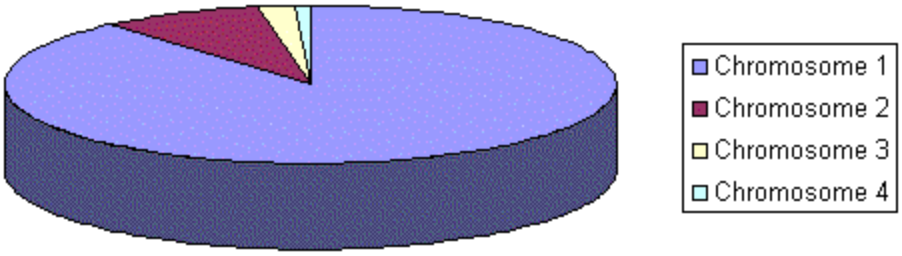
Of course, the step **1** is performed only once for each population.

Rank Selection

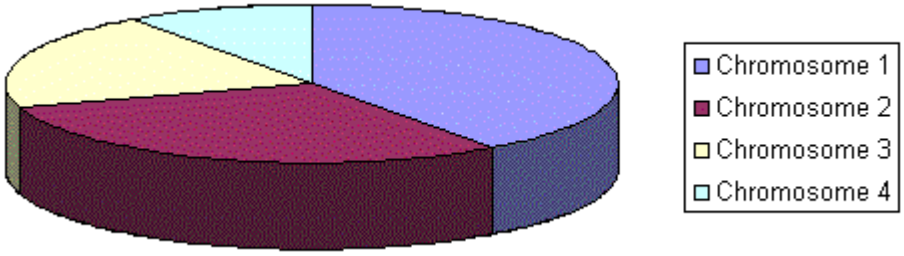
In the roulette wheel selection method, the main issue is that selection will be complicated as there have big difference between fitness value. For example, if 90% fitness is there in best chromosome sum of all finesses then the other rest chromosomes are having very low chances to select for crossover and mutation.

In Rank selection method it ranks the population first then for every chromosome it gives fitness value calculated by this ranking. The worst will have the fitness ***1***, the second worst ***2*** etc. and the best will have fitness ***N*** (number of chromosomes in population).

The following figure shows that how the situation gets changed after changing fitness to the numbers determined by the ranking.



Situation before ranking (graph of fitnesses)



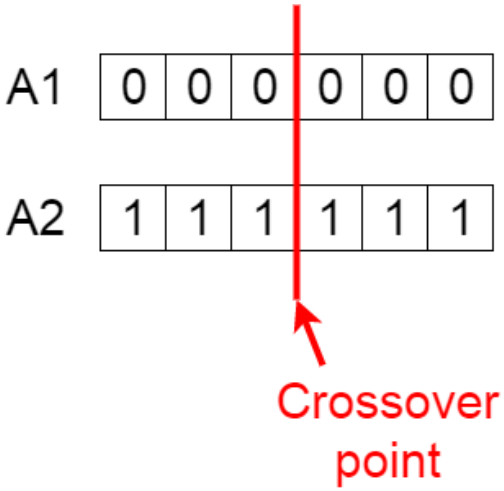
Situation after ranking (graph of order numbers)

We can see that all the chromosomes will get the chance to be selected. Due to this the selection lead to slower convergence.

13.3.4 Crossover

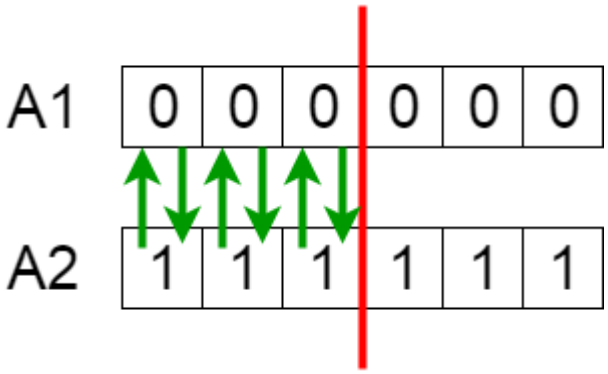
Crossover is the most important step in a genetic algorithm. Here mutation of two selected parents are done with a crossover point **is** chosen at random from within the genes.

In the below figure it is shown (random crossover point).



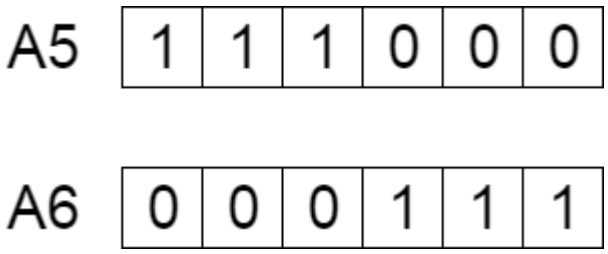
Crossover point

By exchanging the parent's genes Offspring are created among themselves until the crossover point is reached.



Exchanging genes among parents

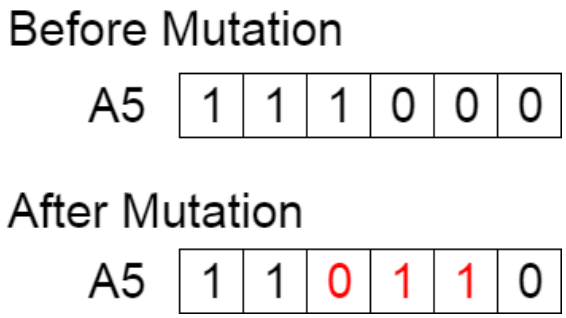
The new offspring are added to the population.



New offspring

13.3.5 Mutation

From the population to create new offspring genes can be subjected to mutation with some probability. With the implementation of this some pf the random bits are changed.



Mutation: Before and After

Mutation occurs to maintain diversity within the population and prevent premature convergence.

13.4 OPERATORS OF GENETIC ALGORITHM

There are some basic operators of GA like Encoding, Crossover and mutation. They are the most important factors on which Performance of GA depends. The operator implementation is totally depending on encoding and also on the problem.

We can see the ways of crossover and mutation in following example.

13.4.1 Encoding

Encoding is changing of chromosomes as per the problem solution in GA. Encoding depends on the problem.

1. Binary Encoding

It is the most used and common property in GA, as it is very simple and relative in nature. Every chromosome is represented by **0 and 1** In binary encoding. As shown in below figure

Chromosome A	10110010110010101110 0101
Chromosome B	11111110000011000001 1111

Binary encoding has many possible chromosomes. Sometimes in this concept encoding is not natural or for some cases it will change or have correction after crossover or mutation.

2. Permutation Encoding

In ordering problem Permutation encoding can be used, such as task ordering problem or travelling salesman problem.

In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

3. Value Encoding

In some problem direct value encoding can be used where some are more complicated values such as real numbers are used. It is difficult if we used binary encoding for this type. every chromosome is a sequence of some values in this concept. Values can be anything like numbers, chars or any object as shown in below table.

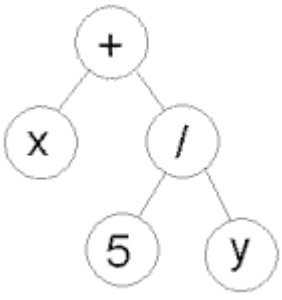
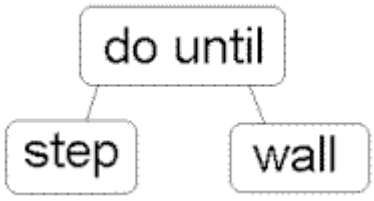
Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

In some mutation and crossover problem this concept is a good choice for the specific problem.

4. Tree Encoding

To evolve programs and expression Tree encoding is mainly used, i.e. for genetic programming.

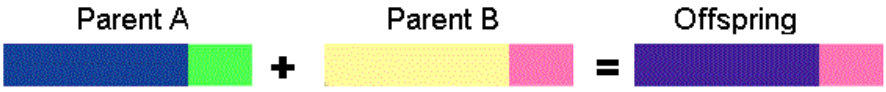
In this concept every chromosome is a tree of some objects, such as commands or functions in programming language.

Chromosome A	Chromosome B
	
(+ x (/ 5 y))	(do_until step wall)

Tree encoded structured problem solution can be find with the help of Tree encoding. LISP programming language is used for this purpose, since programs in LISP are represented directly in the form of tree and can be easily parsed as a tree.

13.4.2 Crossover

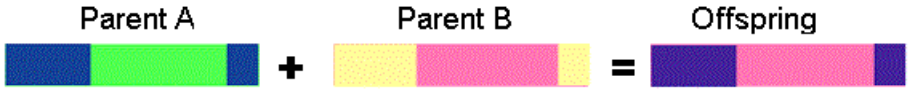
1. **Single point crossover** – in this concept one random crossover point is selected, till reach the crossover point the chromosome of the first parent will be copied as it is once it reached to the crossover point it will change that means it will have copied from second parent binary digit and again it is copied from the first parent, the rest is copied from the other parent



11001011+11011**111** = **11001111**

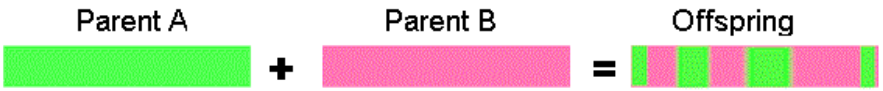
(**1 2 3 4 5** 6 7 8 9) + (**4 5 3 6 8** 9 7 2 1) = (**1 2 3 4 5** 6 8 9 7)

2. **Two-point crossover** - in this concept one random crossover point is selected, till reach the crossover point the chromosome of the first parent will be copied as it is once it reached to the crossover point it will change that means it will have copied from second parent binary digit and again it is copied from the second parent, the rest is copied from the other parent characteristic



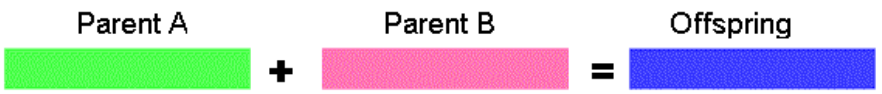
11001**011** + 11**011**111 = **11011111**

3. **Uniform crossover** -in this concept bits are randomly copied from the both parents.



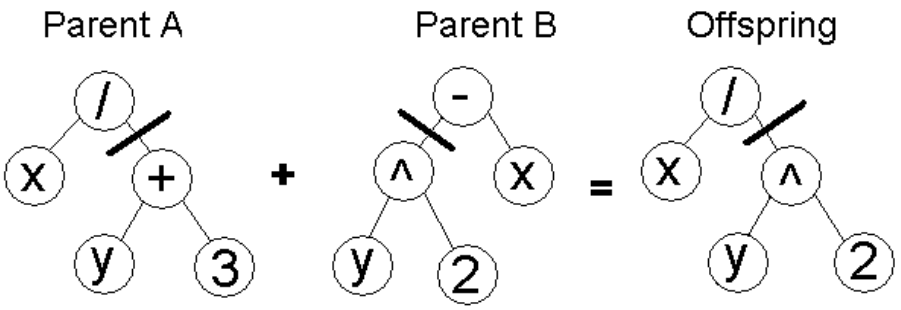
1**10**0101**1** + **1**10**111**01 = 11011111

4. **Arithmetic crossover** – in some cases arithmetic operation is performed to make a new offspring



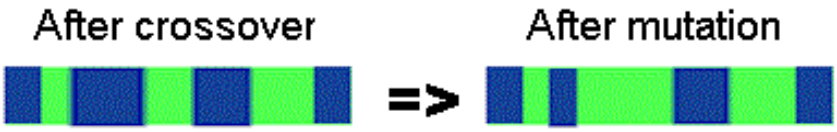
11001011 + 11011111 = 11001001 (AND)

5. **Tree crossover** – from both the parents one crossover point is selected, parents are divided in other two point and the parts below crossover points are exchanged to produce new offspring



13.4.3 Mutation

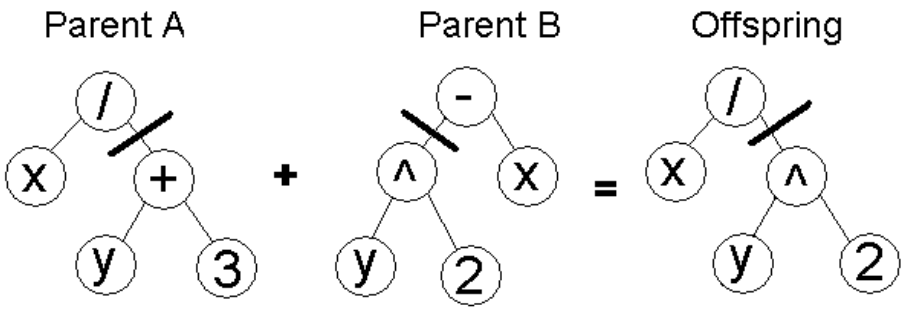
1. **Bit inversion** – in this some bits are randomly selected and inverted



1**1**001001 => 10001001

2. **Order changing** - in this some bits are randomly selected and exchanged

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)



13.4.4 Termination

At this stage he algorithm terminates if the selected population has converged i.e. it does not produce any new offspring which are different from the previous generation offspring. The difference should be there in parent and child offspring. If it gives differentiation between them then it is said to be that the genetic algorithm has provided a solution to our problem.

Pseudocode

```
START
Generate the initial population
Compute fitness
REPEAT
   Selection
   Crossover
```

```
    Mutation
    Compute fitness
UNTIL population has converged
STOP
```

13.5 PARAMETERS OF GENETIC ALGORITHM

Crossover probability and Mutation probability are two basic parameter of GA.

13.5.1 Crossover probability:

crossover probability is property of exchanging data between both parents. If the crossover will not happen, new offspring are inheriting all properties of parents as it is hence it is carbon copies of parents. If the crossover is done, new offspring are generating from the properties of both parent's chromosomes. If **100%** crossover probability is there then all new offspring are made by crossover and mutation. If it is **0%**, whole new generation is exact copies of chromosomes from old population (but this does not mean that the new generation is the same!). Crossover is useful in making the new offspring will be better than the old chromosomes as it is having good properties in it. However, it is good to leave some part of old population survive to next generation.

13.5.2 Mutation probability:

If the mutation is not done, new offspring are generated proceeding immediately after crossover (or directly copied) without any change. If mutation is done in any case, at least one or more parts of a chromosome are changed for mutation. If 100% mutation probability is **there**, then whole chromosomes are changed, if **0%mutation is there**, then nothing is changed.
GA generally prevents by mutation from falling into local extremes. Mutation may not be occurring very often, as GA will change to **random search**.

Other Parameters
population size ca be the other parameters of GA which is mainly used.

13.5.3 Population size:

the number of chromosomes are present in population is called population size. The problem solvation is totally depending on the population size, if the chromosomes present are too few, GA have less possibilities to perform crossover hence only small part of search space is travel through. On the other side, if the chromosomes are too many in numbers the search space is too large and finding solution is too difficult as it required more time to travel all the nodes hence the result is not accurate and it will not solve the problem faster than moderate sized populations.

13.6 LIMITATIONS OF GENETIC ALGORITHM

- For complex problem or big problem continually repeated fitness function of population evaluation is the most prohibitive and limiting segment of genetic algorithms. Hence one cannot Find the optimal solution of complex, multimodal problems, for finding the solution one need to use very expensive fitness function. In some cases of real world problems like structural optimization problems, a single fitness function valuation may require much hours, its time consuming process.
- In complex problem Genetic algorithms cannot use. Because in mutation if the number of elements are higher than the search space size is also increase and it ultimately time taking process. To find out the solution of such problems the bigger problem has to decomposed into several small simplest representations possible. The another problem of complexity is having the issue protection of the parts that have used in representation as a good solution from destructive mutation, accordingly when the fitness assessment requires combination of parts can do the same.
- The other limitation of GA, is it having the feature of converge towards or even arbitrary points instead of global optimum of the problem. Hence it does not "know how" one can use short-term fitness to produce longer-term fitness.
- In GA the operation performs on dynamic data values which is difficult, as it genomes starts converge early towards solutions which may no longer be valid for later data. many methods are present to solve this by adding genetic diversity and preventing early convergence, by extending the probability of mutation or occasionally introducing new elements generated to improve the quality of the solution.
- when the solution quality drops (called *triggered hypermutation*), or by occasionally introducing entirely new, randomly generated elements into the gene pool (called *random immigrants*). Again, evolution strategies and evolutionary programming can be implemented with a so-called "comma strategy" in which parents are not maintained and new parents are selected only from offspring. This can be more effective on dynamic problems.
- When single measure function is considered GAs cannot effectively work for solve problems. In such cases, GA search a random solution as quickly. If that random solution allows the success/failure trial giving probability of different results, then the ratio of successes to failures can be measured as suitable fitness function of the problem.
- To find optimise solution of problem in some cases use of optimization algorithms are more efficient than the genetic algorithm as it finds the solution in minimum speed.
- Some Alternative and complementary algorithms are like hill climbing swarm intelligence, simulated annealing, Gaussian

adaption, etc. the working of the of genetic algorithms is dependent on the amount of knowledge of the problem; well-known problems often have better, more specialized approaches.

13.7 GENETIC ALGORITHM V/s TRADITIONAL ALGORITHM

What is Genetic Algorithm

This algorithm is based on Genetics and Natural Selection. It is similar to the process of inheritance where the species which can adapt to changes of the environment and are ready to survive in the environment. In other words, it is based on biological evolution.

To move forward, in this algorithm continuous changes can be happening in the population of individual solutions at every step, it varies from parent to child in which it chooses individuals randomly from the current population as parents and generates children for the next generation.

- **Selection rules** – It chooses individuals from population (parents) that can help to create new child.
- **Crossover rules** –it merges two parent's properties and it will help in creating new child.
- **Mutation rules** – to create a chid required changes are made in parent selected from population. This algorithm is helpful for finding the optimal solution or near-optimal solution to a problem. With this great feature one can reduce time require to find the solution of the problem.

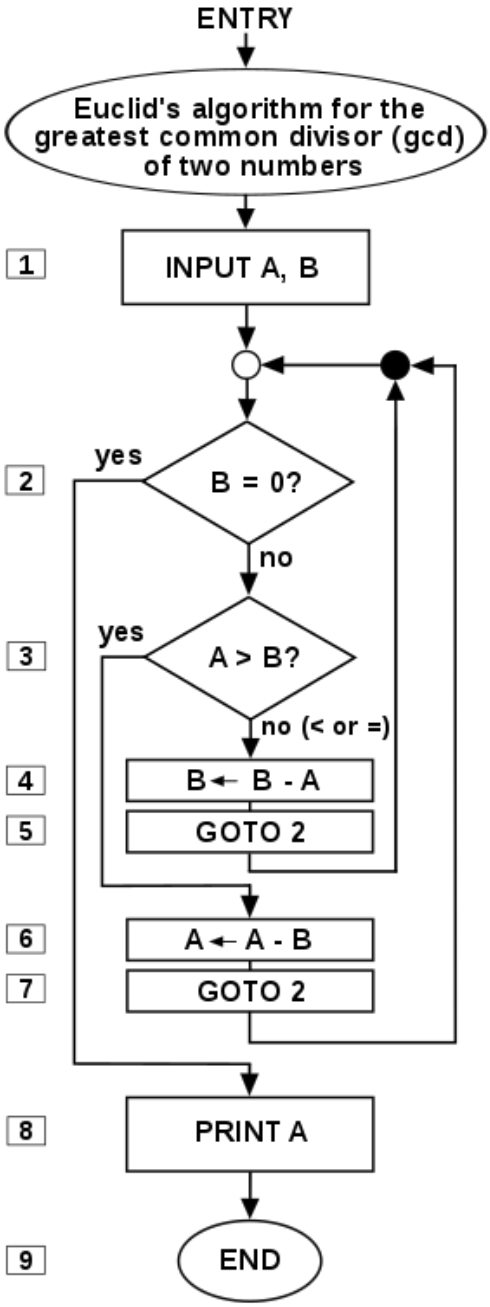
What is Traditional Algorithm

General algorithms which we use to solve problems is called as Traditional Algorithms. It is a method in which step by step procedure are made to solve a given problem. Algorithm itself is called as steps of process. There are many algorithms are available to solve any problem. The best algorithm is which algorithm in that required time for computation, speed and result accuracy is perfect.

This type of algorithms is traditional algorithm which must be clear and have defines 0 for finite number of well-defined inputs and 1 or more well-defined outputs. With the help of available resources, it can be finding solutions.it should also complete the process after some finite number of steps. Some common types of traditional algorithms.

- **Search** – this type of algorithms help to search a particular item in a particular data structure. (Linear search, Binary search, Tree search)
- **Sort** – this type of algorithms helps to sort data in either ascending or descending order. (Bubble sort, Selection sort, Insertion sort, radix sort)

- **Divide and Conquer** – it is helpful in large problems like dividing a large problem into small sub problems and solving them individually to find the solution of the original problem. (merge sort)



Difference Between Genetic Algorithm and Traditional Algorithm

1. Definition

Genetic algorithm is an algorithm which solve both constrained and unconstrained problem solution which will be based on Genetics and Natural Selection whereas, traditional algorithm is a process to find out unambiguous specification that shows how to solve a problem by its rules.

2. Usage

Genetic algorithm is helps in finding optimal or near optimal solution of the problem whereas traditional algorithm provides step by step process to find the solution of the problem which may not be optimal.

3. Complexity

Genetic algorithm is simpler whereas traditional algorithm is more complex hence it cannot have used for big problem.

4. Applications

Genetic Algorithm is applied in the fields of Artificial Intelligence, research, Machine Learning whereas, Traditional algorithm is applied in the fields of Programming, Mathematics, etc.

Travelling Salesman Problem

In Travelling salesman problem (TSP) many cities are listed with their appropriate distance from one another. The task is to visit all the cities at least once, no city should be repeated for finding the path and also it should take smallest path to reach the destination. The traveller should record the distance travelled by agent. In minimum time he has to travel maximum cities.

13.8 SUMMARY

A theory of natural evolution coined by Charles Darwin's and named as genetic algorithm in which search heuristic is used. Natural selection process is used in which it selected fittest individuals for reproduction in order to produce offspring of the next generation. There are five approaches of genetic algorithm firstly finding the population for mutation process, from population one need to select the chromosomes with great fitness function, after getting the selection of chromosomes crossover activity need to be perform in which the parents are going to share their features between them to produce new offspring's. in crossover the activity crossover point is to be selected to do the crossover. Last but not the least in mutation the new offspring were created with the inheritance property of parents as we select the chromosomes with great fitness function the new offspring is better than the parents. There are some basic operators of GA like Encoding, Crossover and mutation. They are the most important factors on which Performance of GA depends. The operator implementation is totally depending on encoding and also on the problem. General algorithms which we use to solve problems is called as Traditional Algorithms Genetic algorithm is an algorithm which solve both constrained and unconstrained problem solution which will be based on Genetics and Natural Selection whereas, traditional algorithm is a process to find out unambiguous specification that shows how to solve a problem by its rules.

13.9 UNIT AND EXCERCISE

1. What is Genetic algorithm? Explain in detail.
2. Explain 5 approaches of genetic algorithm.
3. List and explain all the parameters of genetic algorithm.
4. Explain various operators of genetic algorithm.
5. Discuss various limitations of genetic algorithm.
6. Write down the difference between traditional algorithm and genetic algorithm.
7. What is traditional algorithm and how it works?
8. Explain various selection process methods.

13.10 REFERENCES

- Artificial Intelligence a Modern Approach 4th Edition by Peter Norvig and Stuart Russell published by Pearson
- Artificial Intelligence a Modern Approach 3rd Edition by Peter Norvig and Stuart Russell published by Pearson
- Understanding machine learning from theory to algorithms 1st Edition Shai Shalev and Shai Ben David published by Cambridge University Press

❄❄❄❄❄❄❄