POKHARA UNIVERSITY

Full Note

ARTIFICIAL INTELEGENCE

CMP 421

Bachelor of Computer Engineering

By: Mr. Deepak Bhatta (Lecturer)

National Academy of Science and Technology
DHANGADHI ENGINEERING COLLEGE
Uttar Behadi, Dhangadhi-4, Kailali
(NEPAL)

Unit 1 6 hrs.

Introduction to AI

1.1 Definition of Artificial Intelligence
1.2 Brief history of Artificial Intelligence
1.3 Importance and Application of Artificial Intelligence
1.4 AI and related fields
1.5 Definition of Knowledge, and Learning
1.6 Intelligent Agents & it's types and performance measure

1.1 Definition of Artificial Intelligence

DEEPAK BHATTA

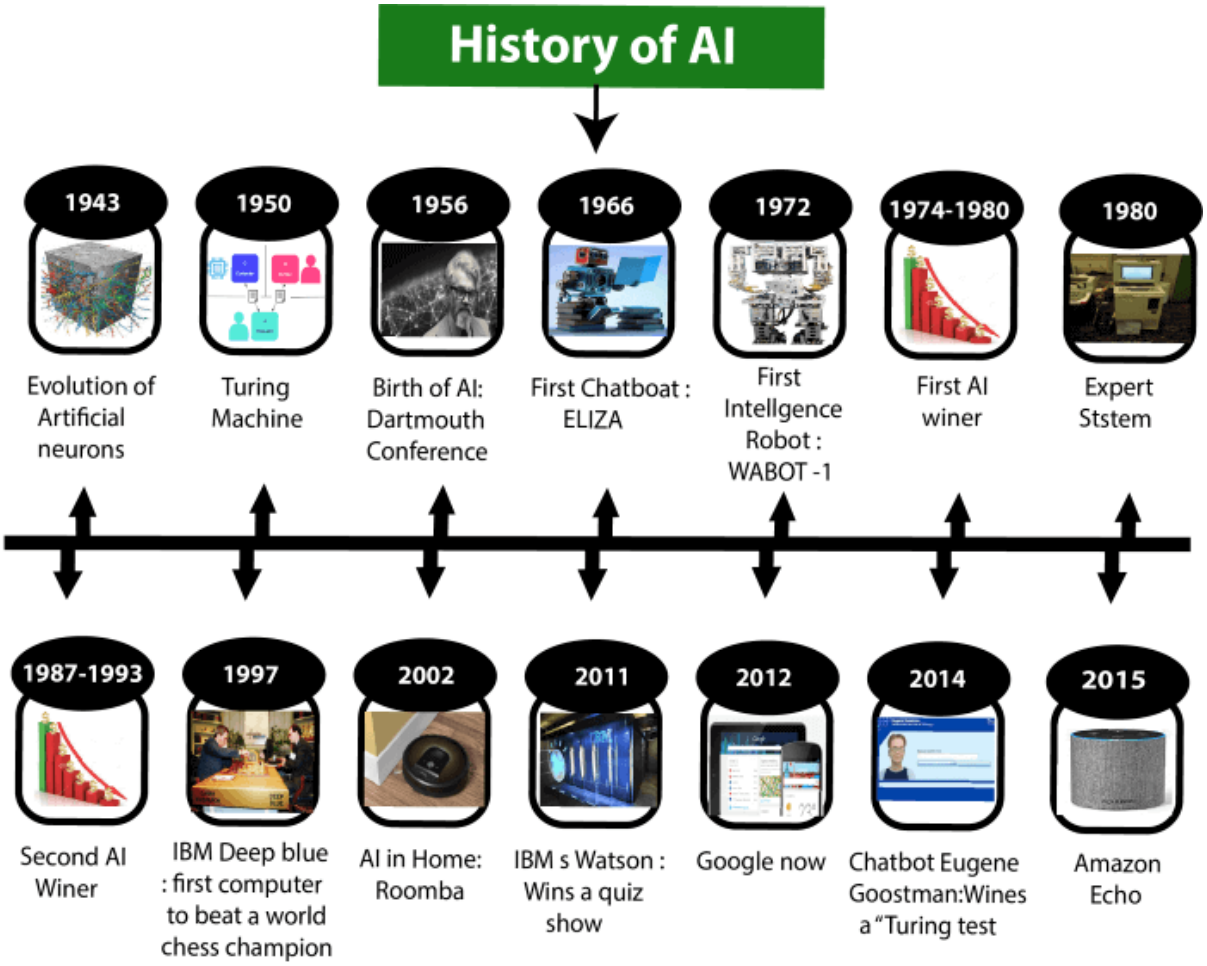
- **Intelligence:** Ability to learn, understand and think.
- **Artificial Intelligence:** Artificial intelligence is the ability to let machines learn, understand and think.
- It includes reasoning, knowledge, planning, learning, communication.
- Artificial intelligence (AI) is the intelligence exhibited by machines or software.
- Artificial Intelligence refers to intelligence displayed by machines that simulates human and animal intelligence.
- Artificial Intelligence sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to natural intelligence displayed by human and other animals.
- AI is an area of computer science that emphasize the creation of intelligent machine that work and react like human.
- AI textbooks define the field as "the study and design of intelligent agents" where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success.
- **AI in practice:**
  - Self-Driving Cars.
  - Applications like Siri that understands and respond to human speech.
  - Google's AlphaGo AI has defeated many GO champions such as Ke Jie.
  - Implementing AI in chess (IBM Deep Blue vs. Garry Kasparov, 1996).
    - **6 games: K, D, draw, draw, draw, D**
  - Amazon ECHO product (home controlled chatbot device).

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

1.2 Brief history of Artificial Intelligence

***John McCarthy**, who coined the term in 1956, defines it as "it is the science and engineering of making intelligent machines, especially intelligent computer program." i.e. it is the branch of computer science dedicated to develop program that enable the computer system to reproduce the human activities to perform some task or try to imitate the activities.*

A Brief History of Artificial Intelligence



Maturation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

The birth of Artificial Intelligence (1952-1956)

- **Year 1955:** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden years-Early enthusiasm (1956-1974)

DEEPAK BHATTA

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

The first AI winter (1974-1980)

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

A boom of AI (1980-1987)

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

The second AI winter (1987-1993)

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

The emergence of intelligent agents (1993-2011)

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

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

DEEPAK BHATTA KAJI

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

Now, AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

1.3 Importance and Applications of AI

Artificial Intelligence is the machines which are designed and programmed in such a manner that they can think and act like a human.

Artificial Intelligence becomes the important part of our daily life. Our life is changed by AI because this technology is used in a wide area of day to day services.

These technologies reduce human effort. Now in many industries, people are using this technology to develop machine slaves to perform the different activity. Using the machine for the work speed up your process of doing work and give you an accurate result.

The introduction of AI brings the idea of error free world. This technology will slowly introduce in all the sector to reduce human effort and give accurate and faster result.

AI is important because it can help solve immensely difficult issues in various industries, such as entertainment, education, health, commerce, transport, and utilities. AI applications can be grouped into five categories:

- **Reasoning:** The ability to solve problems through logical deduction. e.g. financial asset management, legal assessment, financial application processing, autonomous weapons systems, games
- **Knowledge:** The ability to present knowledge about the world. e.g. financial market trading, purchase prediction, fraud prevention, drug creation, medical diagnosis, media recommendation
- **Planning:** The ability to set and achieve goals. e.g. inventory management, demand forecasting, predictive maintenance, physical and digital network optimization, navigation, scheduling, logistics
- **Communication:** The ability to understand spoken and written language. e.g. real-time translation of spoken and written languages, real-time transcription, intelligent assistants, voice control
- **Perception:** The ability to infer things about the world via sounds, images, and other sensory inputs. e.g. medical diagnosis, autonomous vehicles, surveillance

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

AI Used in Bank and Financial System

Banks are using AI technology to handle numerous activities in the bank. They handle work like financial operations, Money investing in stocks, Managing different properties and much more. Using AI to handle this activity beat a human in trading challenges. Using AI in the bank helps the bank to handle their customer and give them a quick solution.

Use of AI in Medical Science

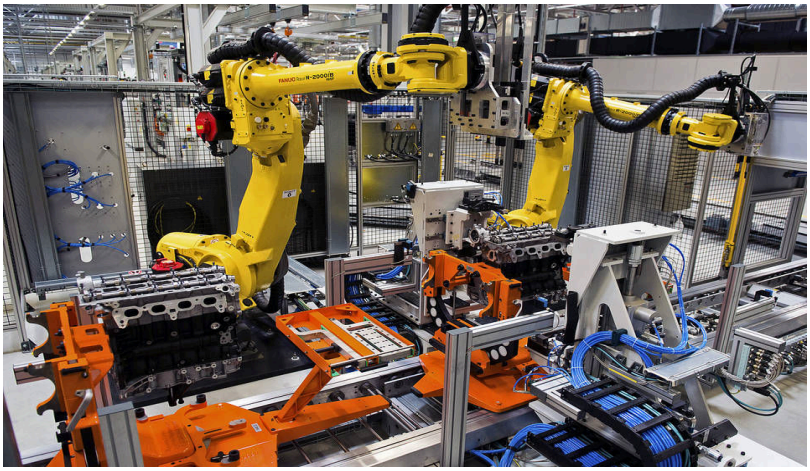
AI technology changes the face of medical science. There is a number of applications in which AI are used and give incredible value. In medical science AI is used to create virtual personal health care assistant. They are used for research and analytics. Even Healthcare bots are also introduced to give customer support. Bots are used for scheduling an appointment in hospitals. And the most important thing they give 24/7 assistance.

Heavy Industries

Today in most of the big manufacturing company AI are mostly used in the production unit. They are used as a robot who give a different shape to an object, who displace object from one place to another, they are used as a convey belt and much more.

If they are used in management system also. They are used to keep the records of the employee. They are used to extract correct data for decision making of the company. Using AI in the big industry help them to complete their task in time and helps business to get proper leads generation.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Role of AI in Air Transport

One of the most systematic transport is air transport. And without AI air transport can't survive. A machine which is used in the plans for performing different functions is run on the basis of AI. All most all the activity which performed to control air transportation is based on AI technologies. There are different softwares designed on the AI platform to give better flight to passengers and feel free from the danger.

AI role in Gaming Zone

Computer and TV games got more development and updates in their fields. There was a time when "Super Mario" was considered as the best game. But nowadays there are different gaming bots are introduced and you don't have to wait for other to play with yours. Bots are developed who will play with you.

You can buy machines that can play master level chess for a few hundred dollars. There is some AI in them, but they play well against people mainly through brute force computation--looking at hundreds of thousands of positions. To beat a world champion by brute force and known reliable heuristics requires being able to look at 200 million positions per second.

Deep Blue versus Garry Kasparov was a pair of six-game chess matches between world chess champion Garry Kasparov and an IBM supercomputer called Deep Blue.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Conclusion

AI technology becomes the part of everyone's life. From small to big work everyone is using AI to generate leads and remove the work pressure. These technologies are so advanced that you don't have to write code for every activity. They understand the motion of work automatically.

1.4 AI and related fields

DEEPAK BHATTA KAJI

1. Healthcare

AI and ML technology has been particularly useful in the healthcare industry because it generates massive amounts of data to train with and enables algorithms to spot patterns faster than human analysts.

- Medecision developed an algorithm that detects 8 variables in diabetes patients to determine if hospitalization is required.
- An app called BiliScreen utilizes a smartphone camera, ML tools, and computer vision algorithms to detect increased levels of bilirubin in the sclera (white portion) of a person's eye, which is used to screen people for pancreatic cancer. This cancer has no telltale symptoms; hence it has one of the worst prognoses of all cancers.
- NuMedii, a biopharma company, has developed a platform called Artificial Intelligence for Drug Discovery (AIDD), which uses big data and AI to detect the link between diseases and drugs at the systems level.
- GNS Healthcare uses ML algorithms to match patients with the most effective treatments for them.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2. Entertainment

A familiar application of AI in everyday life is seen with services like Netflix or Amazon, wherein ML algorithms analyze the user's activity and compare it with that of other users to determine which shows or products to recommend. The algorithms are becoming intelligent with time—to the extent of understanding that a user may want to buy a product as a gift and not for himself/herself, or that different family members have different watching preferences.

3. Finance

- Financial services companies use AI-based natural language processing tools to analyze brand sentiment from social media platforms and provide actionable advice.
- Investment companies like Aidya and Nomura Securities use AI algorithms to conduct trading autonomously and robo-traders to conduct high-frequency trading for greater profits, respectively.
- Fintech firms like Kensho and Forward Lane use AI-powered B2C robo-advisors to augment rebalancing decisions and portfolio management performed by human analysts. Wealthfront uses AI algorithms to track account activity and help financial advisors customize their advice.
- Chatbots, powered by natural language processing, can serve banking customers quickly and efficiently by answering common queries and providing information promptly.
- Fraud detection is an important application of AI in financial services. For example, Mastercard uses Decision Intelligence technology to analyze various data points to detect fraudulent transactions, improve real-time approval accuracy, and reduce false declines.

4. Data security

Cyber-attacks are becoming a growing reality with the move to a digital world. There are also concerns about AI programs themselves turning against systems.

- Automatic exploit generation (AEG) is a bot that can determine whether a software bug, which may cause security issues, is exploitable. If a vulnerability is found, the bot automatically secures it. AEG systems help develop automated signature generation algorithms that can predict the likelihood of cyberattacks.
- PatternEx and MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) have developed an AI platform called AI2 which claims to predict cyber-attacks better than existing systems. The platform uses Active Contextual Modeling, a continuous feedback loop between a human analyst and the AI system, to provide an attack detection rate that is better than ML-only solutions by a factor of 10.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- Deep Instinct, an institutional intelligence company, says that malware code varies between 2% - 10% in every iteration and that its AI model is able to handle the variations and accurately predict which files are malware.

5. Manufacturing

- Landing.ai claims to have created machine-vision tools to **find microscopic defects** in objects like circuit boards using an ML algorithm trained using tiny volumes of sample images. In the future, self-driving robots may be created which can move finished goods around without endangering anyone or anything around.
- Robots in factories are often stationary but are still in danger of crashing into objects around it. A new concept called collaborative robots or "cobots, enabled by AI, can take instructions from humans, including instructions that the robot has not been previously exposed to, and work productively with them.
- AI algorithms can influence the **manufacturing supply chain** by detecting the patterns of demand for products across geographies, socioeconomic segments, and time, and **predicting market demand**. This, in turn, will affect inventory, raw material sourcing, financing decisions, human staffing, energy consumption, and maintenance of equipment.
- AI tools help in **predicting malfunctions and breakdown of equipment** and taking or recommending preemptive actions as well as tracking operating conditions and performance of factory tooling.

6. Automotive industry

- Tesla introduced **TeslaBot**, an intelligent virtual assistant integrated with Tesla models S and X, allows users to interact with their car from their phone or desktop.
- Uber AI Labs is working on developing **self-driven cars** with the help of the best engineers and scientists. Uber has already tested a batch of self-driving cars in 2016.
- Nvidia has partnered with Volkswagen to develop **"intelligent co-pilot systems"** in cars that will enable safety warnings, gesture control, and voice and facial recognition.
- Ericsson predicts that 5G technology will improve vehicle-to-vehicle communication wherein sensors will be implanted in airport runways, railways, and roads.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

1.5 Definition of Knowledge and Learning

What is Knowledge?

Knowledge is the information about a domain that can be used to solve problems in that domain. To solve many problems requires much knowledge, and this knowledge must be represented in the computer. As part of designing a program to solve problems, we must define how the knowledge will be represented. A **representation scheme** is the form of the knowledge that is used in an agent. A **representation** of some piece of knowledge is the internal representation of the knowledge. A representation scheme specifies the form of the knowledge. A **knowledge base** is the representation of all of the knowledge that is stored by an agent.

A good representation scheme is a compromise among many competing objectives. A representation should be

- rich enough to express the knowledge needed to solve the problem.
- as close to the problem as possible; it should be compact, natural, and maintainable. It should be easy to see the relationship between the representation and the domain being represented, so that it is easy to determine whether the knowledge represented is correct. A small change in the problem should result in a small change in the representation of the problem.
- able to be acquired from people, data and past experiences.

What is learning?

- According to **Herbert Simon**, learning denotes changes in a system that enable a system to do the same task more efficiently the next time.
- **Arthur Samuel stated that**, "Machine learning is the subfield of computer science, that gives computers the ability to learn without being explicitly programmed ".
- In 1997, **Mitchell** proposed that, " A computer program is said to learn from experience **'E'** with respect to some class of tasks **'T'** and performance measure **'P'**, if its performance at tasks in **'T'**, as measured by **'P'**, improves with experience **E** ".
- The main purpose of machine learning is to study and design the algorithms that can be used to produce the predicates from the given dataset.
- Besides these, the machine learning includes the agent's percepts for acting as well as to improve their future performance.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

The following tasks must be learned by an agent.

- To predict or decide the result state for an action.
- To know the values for each state (understand which state has high or low vale).
- To keep record of relevant percepts.
-

Why do we require machine learning?

- Machine learning plays an important role in improving and understanding the efficiency of human learning.
- Machine learning is used to discover a new thing not known to many human beings.

Various forms of learnings are explained below:

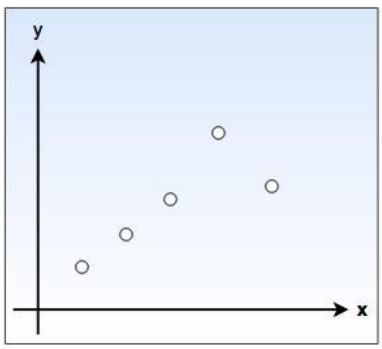
1. Rote learning

- Rote learning is possible on the basis of memorization.
- This technique mainly focuses on memorization by avoiding the inner complexities. So, it becomes possible for the learner to recall the stored knowledge. **For example:** When a learner learns a poem or song by reciting or repeating it, without knowing the actual meaning of the poem or song.

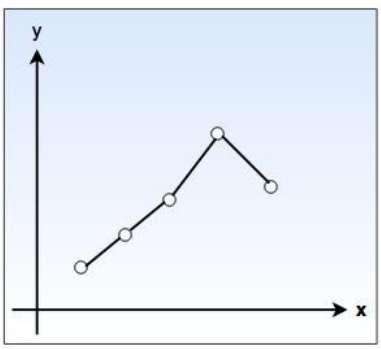
2. Induction learning (Learning by example).

- Induction learning is carried out on the basis of supervised learning.
- In this learning process, a general rule is induced by the system from a set of observed instances.
- However, class definitions can be constructed with the help of a classification method.

For Example:
Consider that **'*f*'** is the target function and example is a pair (x f(x)), where 'x' is input and f(x) is the output function applied to 'x'.



Fig(a)



Fig(b)

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3. Learning by taking advice

- This type is the easiest and simple way of learning.
- In this type of learning, a programmer writes a program to give some instructions to perform a task to the computer. Once it is learned (i.e. programmed), the system will be able to do new things.
- Also, there can be several sources for taking advice such as humans(experts), internet etc.
- However, this type of learning has a more necessity of inference than rote learning.
- As the stored knowledge in knowledge base gets transformed into an operational form, the reliability of the knowledge source is always taken into consideration.

Explanation based learning

- Explanation-based learning (EBL) deals with an idea of single-example learning.
- This type of learning usually requires a substantial number of training instances but there are two difficulties in this:
  i. it is difficult to have such a number of training instances
  ii. Sometimes, it may help us to learn certain things effectively, especially when we have enough knowledge.
  Hence, it is clear that instance-based learning is more data-intensive, data-driven while EBL is more knowledge-intensive, knowledge-driven.
- Initially, an EBL system accepts a training example.
- On the basis of the given goal concept, an operationality criteria and domain theory, it "generalizes" the training example to describe the goal concept and to satisfy the operationality criteria (which are usually a set of rules that describe relationships between objects and actions in a domain).
- Thus, several applications are possible for the knowledge acquisition and engineering aspects.

Learning in Problem Solving

- Humans have a tendency to learn by solving various real-world problems.
- The forms or representation, or the exact entity, problem solving principle is based on reinforcement learning.
- Therefore, repeating certain action results in desirable outcome while the action is avoided if it results into undesirable outcomes.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- As the outcomes have to be evaluated, this type of learning also involves the definition of a utility function. This function shows how much is a particular outcome worth?
- There are several research issues which include the identification of the learning rate, time and algorithm complexity, convergence, representation (frame and qualification problems), handling of uncertainty (ramification problem), adaptivity and "unlearning" etc.
- In reinforcement learning, the system (and thus the developer) know the desirable outcomes but does not know which actions result into desirable outcomes.
- In such a problem or domain, the effects of performing the actions are usually compounded with side-effects. Thus, it becomes impossible to specify the actions to be performed in accordance to the given parameters.
- Q-Learning is the most widely used reinforcement learning algorithm.
- The main part of an algorithm is a simple value iteration update. For each state 'S', from the state set S, and for each action, a, from the action set 'A', it is possible to calculate an update to its expected reduction reward value, with the following expression:

1.6 Intelligent Agents & its type and performance measures

An AI system can be defined as the study of the rational agent and its environment. The agents sense the environment through sensors and act on their environment through actuators. An AI agent can have mental properties such as knowledge, belief, intention, etc. An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of perceiving, thinking, and acting.

An agent can be:

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

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

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

Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

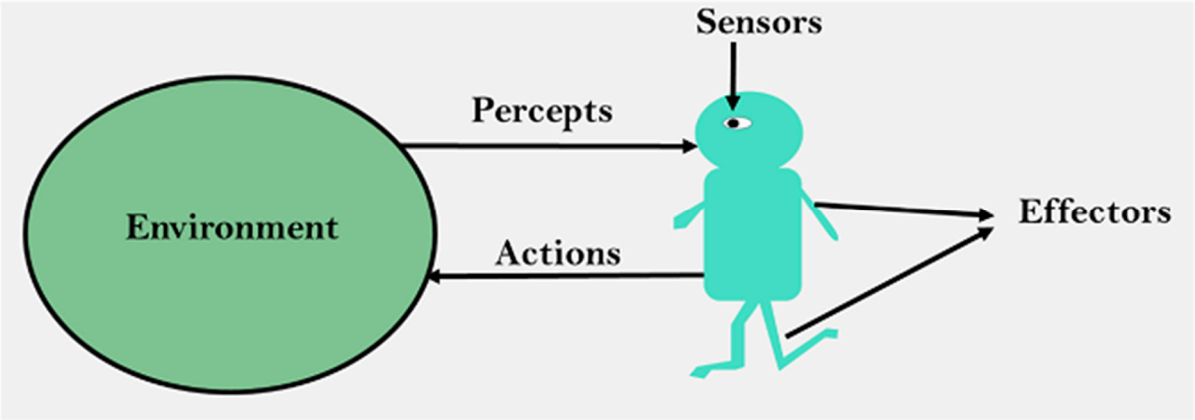
Before moving forward, we should first know about sensors, effectors, and actuators.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

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

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

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



Intelligent Agents:

An intelligent agent is an autonomous entity which act upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

Rule 1: An AI agent must have the ability to perceive the environment.

Rule 2: The observation must be used to make decisions.

Rule 3: Decision should result in an action.

Rule 4: The action taken by an AI agent must be a rational action.

Rational Agent:

A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.

A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward.

Note: Rational agents in AI are very similar to intelligent agents.

Rationality:

The rationality of an agent is measured by its performance measure. Rationality can be judged on the basis of following points:

- Performance measure which defines the success criterion.
- Agent prior knowledge of its environment.
- Best possible actions that an agent can perform.
- The sequence of percepts.

Note: Rationality differs from Omniscience because an Omniscient agent knows the actual outcome of its action and act accordingly, which is not possible in reality.

Structure of an AI Agent

The task of AI is to design an agent program which implements the agent function. The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

1. Agent = Architecture + Agent program

Following are the main three terms involved in the structure of an AI agent:

Architecture: Architecture is machinery that an AI agent executes on.

Agent Function: Agent function is used to map a percept to an action.

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

Agent program: Agent program is an implementation of agent function. An agent program executes on the physical architecture to produce function f.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

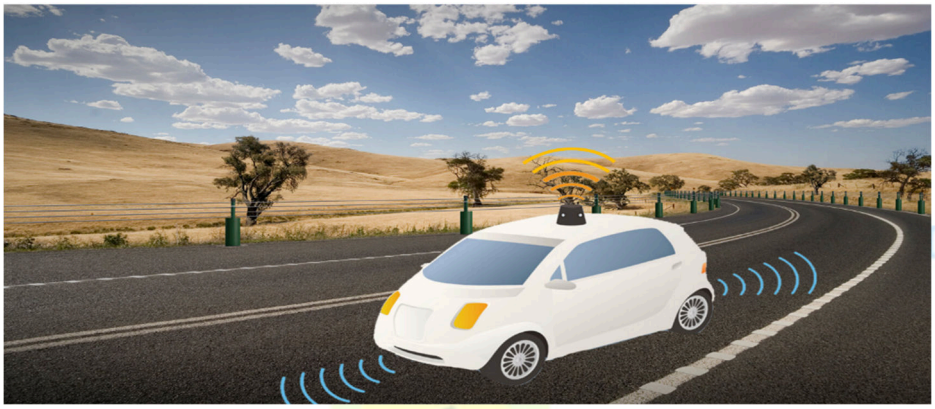
PEAS Representation

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

- **P:** Performance measure
- **E:** Environment
- **A:** Actuators
- **S:** Sensors

Here performance measure is the objective for the success of an agent's behavior.

PEAS for self-driving cars/ Automated Taxi Driver Agent:



Let's suppose a self-driving car or Automated Taxi Driver Agent.

Then PEAS representation will be:

Performance: Safety, Time, Legal drive, Comfort, Getting the correct destination, Less Cost

Environment: Roads, Traffic Other Vehicles, Road Signs, Pedestrian, Different types of Passenger

Actuators: Steering, Accelerator, Brake, Signal, Horn

Sensors: Camera, GPS, Speedometer, Odometer, Accelerometer, Sonar, IR sensor.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example of Agents with their PEAS representation

Agent	Performance measure	Environment	Actuators	Sensors
1. Medical Diagnose	○ Healthy patient ○ Minimized cost	○ Patient ○ Hospital ○ Staff	○ Tests ○ Treatments	Keyboard (Entry of symptoms)
2. Vacuum Cleaner	○ Cleanness ○ Efficiency ○ Battery life ○ Security	○ Room ○ Table ○ Wood floor ○ Carpet ○ Various obstacles	○ Wheels ○ Brushes ○ Vacuum Extractor	○ Camera ○ Dirt detection sensor ○ Cliff sensor ○ Bump Sensor ○ Infrared Wall Sensor
3. Part - picking Robot	○ Percentage of parts in correct bins.	○ Conveyor belt with parts, ○ Bins	○ Jointed Arms ○ Hand	○ Camera ○ Joint angle sensors.

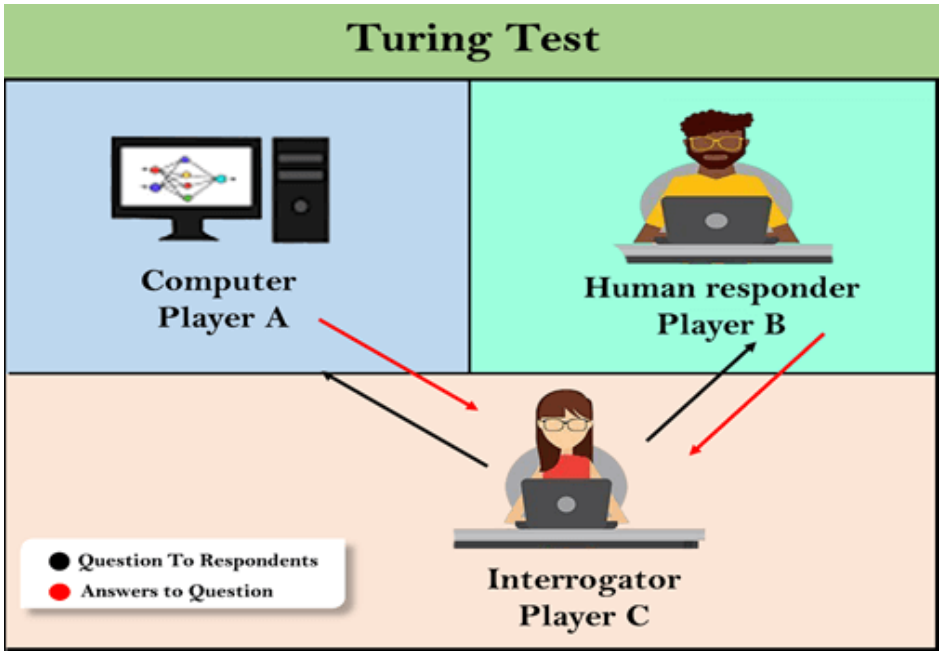
THE END

Turing Test in AI

In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. In this test, Turing proposed that the computer can be said to be an intelligent if it can mimic ***(imitating or copying something)*** human response under specific conditions.

Turing Test was introduced by Turing in his 1950 paper, "Computing Machinery and Intelligence," which considered the question, "Can Machine think?"

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



The Turing test is based on a party game "Imitation game," with some modifications. This game involves three players in which one player is Computer, another player is human responder, and the third player is a human Interrogator, who is isolated from other two players and his jobs are to find that which player is machine among two of them.

Consider, Player A is a computer, Player B is human, and Player C is an interrogator. Interrogator is aware that one of them is machine, but he needs to identify this on the basis of questions and their responses.

The conversation between all players is via keyboard and screen so the result would not depend on the machine's ability to convert words as speech.

The test result does not depend on each correct answer, but only how closely its responses like a human answer. The computer is permitted to do everything possible to force a wrong identification by the interrogator.

The questions and answers can be like:

Interrogator: Are you a computer?

Player A (Computer): No

Interrogator: Multiply two large numbers such as (256896489*456725896)

Player A: Long pause and give the wrong answer.

In this game, if an interrogator would not be able to identify which is a machine and which is human, then the computer passes the test successfully, and the machine is said to be intelligent and can think like a human.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

"In 1991, the New York businessman Hugh Loebner announces the prize competition, offering a $100,000 prize for the first computer to pass the Turing test. However, no AI program to till date, come close to passing an undiluted Turing test".

Chatbots to attempt the Turing test:

ELIZA: ELIZA was a Natural language processing computer program created by Joseph Weizenbaum. It was created to demonstrate the ability of communication between machine and humans. It was one of the first chatterbots, which has attempted the Turing Test.

Parry: Parry was a chatterbot created by Kenneth Colby in 1972. Parry was designed to simulate a person with **Paranoid schizophrenia**(most common chronic mental disorder). Parry was described as "ELIZA with attitude." Parry was tested using a variation of the Turing Test in the early 1970s.

Eugene Goostman: Eugene Goostman was a chatbot developed in Saint Petersburg in 2001. This bot has competed in the various number of Turing Test. In June 2012, at an event, Goostman won the competition promoted as largest-ever Turing test content, in which it has convinced 29% of judges that it was a human. Goostman resembled as a 13-year old virtual boy.

The Chinese Room Argument:

There were many philosophers who really disagreed with the complete concept of Artificial Intelligence. The most famous argument in this list was "**Chinese Room**."

In the year **1980, John Searle** presented "**Chinese Room**" thought experiment, in his paper "**Mind, Brains, and Program**," which was against the validity of Turing's Test. According to his argument, "**Programming a computer may make it to understand a language, but it will not produce a real understanding of language or consciousness in a computer**."

He argued that Machine such as ELIZA and Parry could easily pass the Turing test by manipulating keywords and symbol, but they had no real understanding of language. So it cannot be described as "thinking" capability of a machine such as a human.

Features required for a machine to pass the Turing test:

- **Natural language processing:** NLP is required to communicate with Interrogator in general human language like English.
- **Knowledge representation:** To store and retrieve information during the test.
- **Automated reasoning:** To use the previously stored information for answering the questions.
- **Machine learning:** To adapt new changes and can detect generalized patterns.
- **Vision (For total Turing test):** To recognize the interrogator actions and other objects during a test.
- **Motor Control (For total Turing test):** To act upon objects if requested.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Unit 2 4 hrs.

Problem Solving

2.1 Defining problems as a State Space Search
2.2 Problem formulation and Problem types
2.3 Well defined problems
2.4 Constraint Satisfaction Problem
2.5 Game Playing
2.6 Production System

2.1 Defining problems as a State Space Search

"It is complete set of states including start and goal states, where the answer of the problem is to be searched".

Problem:

"It is the question which is to be solved. For solving the problem, it needs to be precisely defined. The definition means, defining the start state, goal state, other valid states and transitions".

A state space representation allows for the formal definition of a problem which makes the movement from initial state to the goal state quite easily. So, we can say that various problems like planning, learning, theorem proving etc. are all essentially search problems only.

State space search:

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 the intention of finding a ***goal state*** with a desired property.

Problems are often modelled as a state space, a set of ***states*** that a problem can be in. The set of states forms a graph where two states are connected if there is an ***operation*** that can be performed to transform the first state into the second.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

State space search often differs from traditional computer science search methods because the state space is *implicit*: the typical state space graph is much too large to generate and store in memory. Instead, nodes are generated as they are explored, and typically discarded thereafter. A solution to a combinatorial search instance may consist of the goal state itself, or of a path from some *initial state* to the goal state.

Basic Search Problem:

In state space search a state space is formally represented as a tuple.

Given, [S, s, O, G] // 4Tuples

Where,
S is the (implicitly specified) set of states.
s is the start state.
O is the set of state transition operators.
G is the set of goal state.

2.2 Problem formulation and Problem types

Problem Formulation:

It is all about deciding what action and states to be consider.

- Agent sensor give it enough information to tell exactly which states it is in currently.
- It knows exactly what each of its action does.
- Then it can calculate exactly which state it will be in after any sequence of action.

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

The Problem: 8- Puzzle Problem

The eight-tile or 8-puzzle consist of a 3 by 3 (3*3) square frame board which holds 8 movable tiles numbered 1 to 8. One square is empty, allowing the adjacent tiles to be shifted. The objective of the puzzle is to find a sequence of tile movements that leads from a starting configuration to a goal configuration.

The states of 8 tile puzzle are the different permutations of the tiles within frame.

Let's do a standard formulation of this problem now.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

States: It specifies the location of each of the 8 tiles and the blank in one of the nice squares.

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

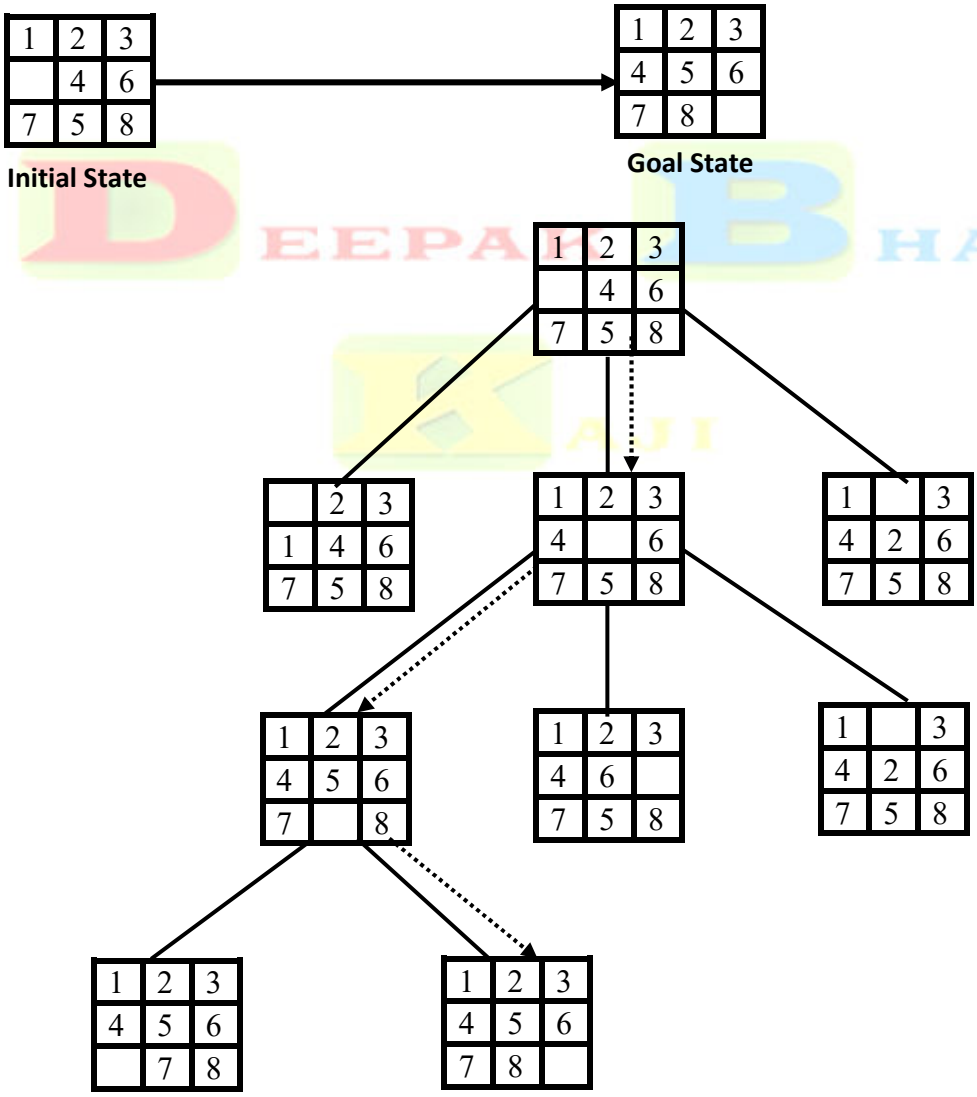
Goal : Many goal configurations are possible one such is shown in the figure

Legal moves (or state) : They generate legal states that result from trying the four actions-

- Blank moves left
- Blank moves right
- Blank moves up
- Blank moves down

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

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



Here, we Choose only those states which has minimum path cost.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Problem Types

There are mainly two types of problems:

1. Single state problem.
2. Multiple state problem.

Single State Problem:

When the environment is completely accessible and the agent can calculate its state after any sequence of action, we call it a single-state problem.

Multiple State problem:

When the environment is not fully accessible, the agent must reason about sets of states that it might get to, rather than single states. We call this a multiple-state problem.

The Problem: 8-Queen Puzzle

The eight queen (8-queen) puzzle is the problem of placing eight chess queens on an 8x8 chessboard so that no two queens attack each other; thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general N- queen problem of placing N non-attacking queens on an NxN chessboard, for which solutions exist for all-natural numbers with the exception of n=2 and n=3.

Solutions:

The eight queens puzzle has 94 distinct solutions. If solutions that differ only by the symmetry operations of rotation and reflection of the board are counted as one, the puzzle has 12 solutions. These are called Fundamental solutions.

A fundamental solution usually has eight variants obtained by rotating 90º, 180º, 270º and then reflecting each of the four rotational variants in a mirror in a fixed position. Thus, the number of distinct solutions is $11\times8 + 1\times4 = 92$.

		Q					
				Q			
	Q						
							Q
Q							
						Q	
			Q				
					Q		

Note: For more detail N-Queen Problem is discussed in CSP problem.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

The Problem: Missionaries and Cannibals

The Missionaries and Cannibals problem is a classic AI-Puzzle that can be defined as follows:

On One bank of river are three Missionaries and three cannibals. There is one boat available that can hold up to two people and that would like to use boat to cross the river. If the Cannibals ever outnumbers the Missionaries on either of the river's banks, the Missionaries will get eaten or Missionaries may die.

How can the boat be used to safely carry all the Missionaries and Cannibals across the river?

There are 3 missionaries, 3 cannibals, and 1 boat that can carry up to two people on one side of a river.

Goal: Move all the missionaries and cannibals across the river.

Constraint: Missionaries can never be outnumbered by cannibals on either side of river, or else the missionaries are killed.

State: Configuration of missionaries and cannibals and boat on each side of river.

Operators: Move boat containing some set of occupants across the river (in either direction) to the other side.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Rules for solving Missionaries and Cannibal Problem:

#Rules	Action	Meaning
1.	(0, M)	One Missionary Sailing the boat from Bank 1 to Bank 2.
2.	(M, 0)	One Missionary Sailing the boat from Bank 2 to Bank 1.
3.	(M, M)	Two Missionary Sailing the boat from Bank 1 to Bank 2.
4.	(M, M)	Two Missionary Sailing the boat from Bank 2 to Bank 1.
5.	(M, C)	One Missionary & One Cannibal Sailing from Bank 1 to Bank 2.
6.	(C, M)	One Missionary & One Cannibal Sailing from Bank 2 to Bank 1.
7.	(C, C)	Two Cannibal Sailing the boat from Bank 1 to Bank 2.
8.	(C, C)	Two Cannibal Sailing the boat from Bank 2 to Bank 1.
9.	(0, C)	One Cannibal Sailing the boat from Bank 1 to Bank 2.
10.	(C, 0)	One Cannibal Sailing the boat from Bank 2 to Bank 1.

Table: Production Rules for the Missionaries and Cannibal Problem

DEEPAK BHATTA KAJI

Applying the rules: Formulation I

Rule Applied	Person in the River Bank 1	Person in the River Bank 2	Boat Position
Initial State	***M, M, M, C, C, C***	***0***	***Bank 1***
5	M, M, C, C	M, C	Bank 2
2	M, M, C, C, M	C	Bank 1
7	M, M, M	C, C, C	Bank 2
10	M, M, M, C	C, C	Bank 1
3	M, C	C, C, M, M	Bank 2
6	M, C, C, M	C, M	Bank 1
3	C, C	C, M, M, M	Bank 2
10	C, C, C	M, M, M	Bank 1
7	C	M, M, M, C, C	Bank 2
10	C, C	M, M, M, C	Bank 1
7	0	M, M, M, C, C, C	Bank 2
Goal State	***0***	***M, M, M, C, C, C***	***Bank 2***

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Applying the rules: Formulation II

Rule Applied	Person in the River Bank 1	Person in the River Bank 2	Boat Position
Initial State	***M, M, M, C, C, C***	***0***	***Bank 1***
7	M, M, M, C	C, C	Bank 2
9	M, M, M, C, C	C	Bank 1
7	M, M, M	C, C, C	Bank 2
10	M, M, M, C	C, C	Bank 1
3	M, C	C, C, M, M	Bank 2
6	M, C, C, M	C, M	Bank 1
3	C, C	C, M, M, M	Bank 2
10	C, C, C	M, M, M	Bank 1
7	C	M, M, M, C, C	Bank 2
2	C, M	M, M, C, C	Bank 1
5	0	M, M, M, C, C, C	Bank 2
Goal State	***0***	***M, M, M, C, C, C***	***Bank 2***

DEEPAK BHATTA KAJI

Try Yourself

Try to solve this problem same like Missionaries and Cannibal Problem.

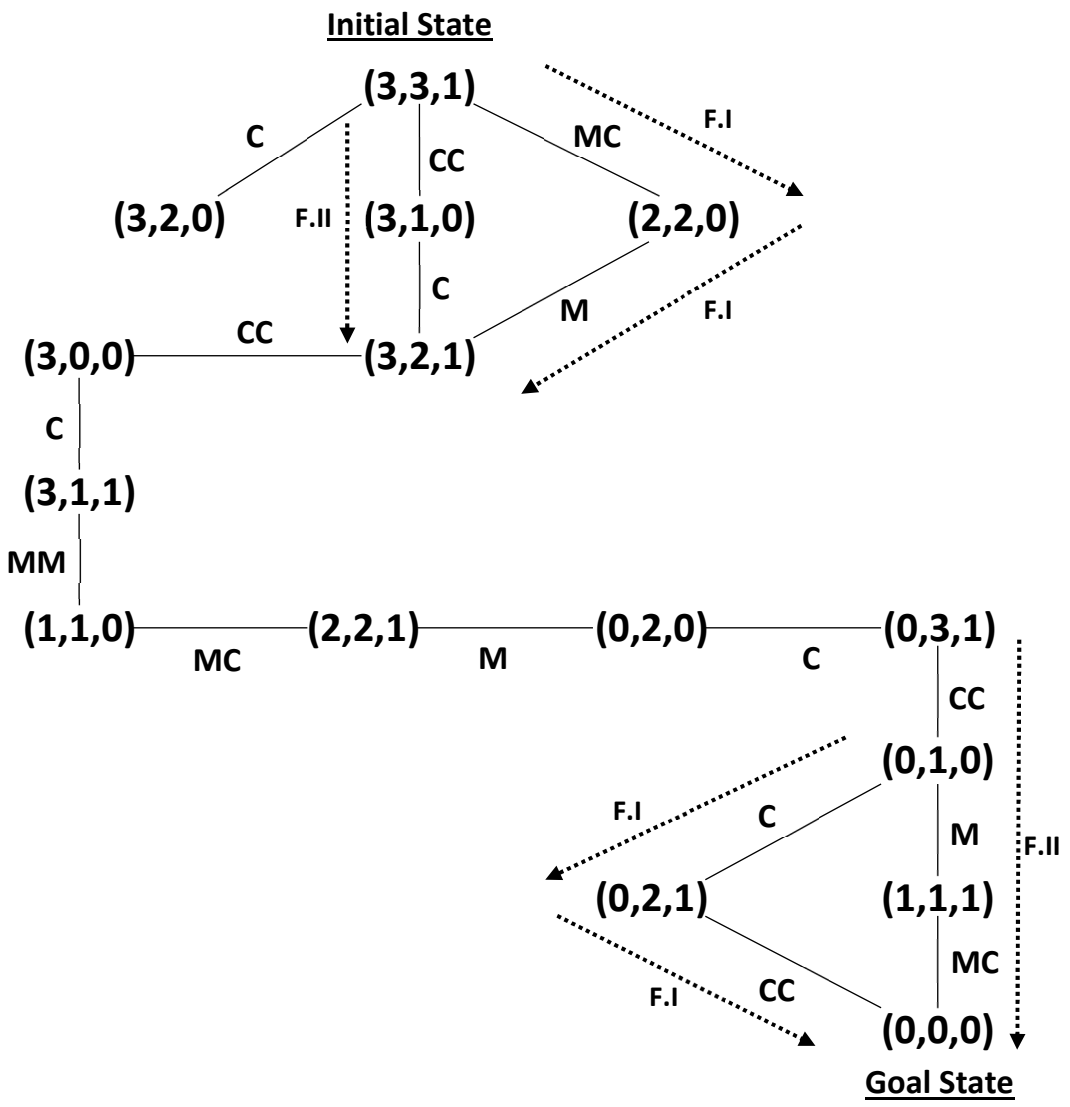
Read the problem carefully to solve gently.

A farmer with wolf, goat & cabbage come to the edge of river. They want to cross the river. There is a boat at river edge but only the farmer can row, the boat can carry two things at a time. If the wolf is ever left alone with the goat, the wolf will eat the goat, similarly if the goat is left alone with cabbage, the goat will eat cabbage. Schedule the things so that all four characters arrive safely on the other side of the river.

Man → M **Wolf → W** **Goat → G** **Cabbage → C**

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

State Space Diagram for Missionaries and Cannibals Problem Using Formulation I and Formulation II :



Here,

Flow of:

F.I = Formulation I

F.II = Formulation II

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

The Problem: Water Jug Problem

A Water Jug Problem: You are given two jugs, a 4-Liter one and a 3-Liter one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it.
How can you get exactly 2-Liter of water in the 4-Liter jug?

Solution:

State Representation and Initial State = we will represent a state of the problem as a tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-Liter jug. Note $0 \leq x \leq 4$ and $0 \leq y \leq 3$.

Our initial state: (0,0)

Goal Predicate: state = $(2, y)$ where $0 \leq y \leq 3$.

Operators: we must define a set of operators that will take us from one state to another:

#Rule	Meaning	Check Condition	Calculation
1.	Fill 4-Liter jug	$(x, y) \rightarrow x < 4$	$(4, y)$
2.	Fill 3-Liter jug	$(x, y) \rightarrow y < 3$	$(x, 3)$
3.	Empty 4-Liter jug on the ground	$(x, y) \rightarrow x > 0$	$(0, y)$
4.	Empty 3-Liter jug on the ground	$(x, y) \rightarrow y > 0$	$(x, 0)$
5.	Pour water from 3-Liter jug to fill 4-Liter jug	$(x, y) \rightarrow$ $0 < x + y \geq 4$ and $(y > 0)$	$(4, y - (4 - x))$
6.	Pour water from 4-Liter jug to fill 3-Liter jug	$(x, y) \rightarrow$ $0 < x + y \geq 3$ and $(x > 0)$	$(x - (3 - y), 3)$
7.	Pour all of water from 3-Liter jug into 4-Liter jug	$(x, y) \rightarrow$ $0 < x + y \leq 4$ and $(y > 0)$	$(x + y, 0)$
8.	Pour all of water from 4-Liter jug into 3-Liter jug	$(x, y) \rightarrow$ $0 < x + y \leq 3$ and $(x > 0)$	$(0, x + y)$

Table: Production Rules for the Water Jug Problem

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Formulation I.

Liters in 4-Liter jug	Liters in 3-Liter jug	Rule Applied	Meaning
0	0	-	Initial State
0	3	2	Fill 3-Liter jug.
3	0	7	Pour water from 3-Liter jug to 4-Liter jug.
3	3	2	Fill 3-Liter jug.
4	2	5	Pour water from 3-Liter jug to 4-Liter jug until 4-Liter jug is full.
0	2	3	Empty 4-Liter jug on ground.
2	0	8	Pour water from 3-Liter jug to 4-Liter jug.
2	0	-	Goal State Reached.

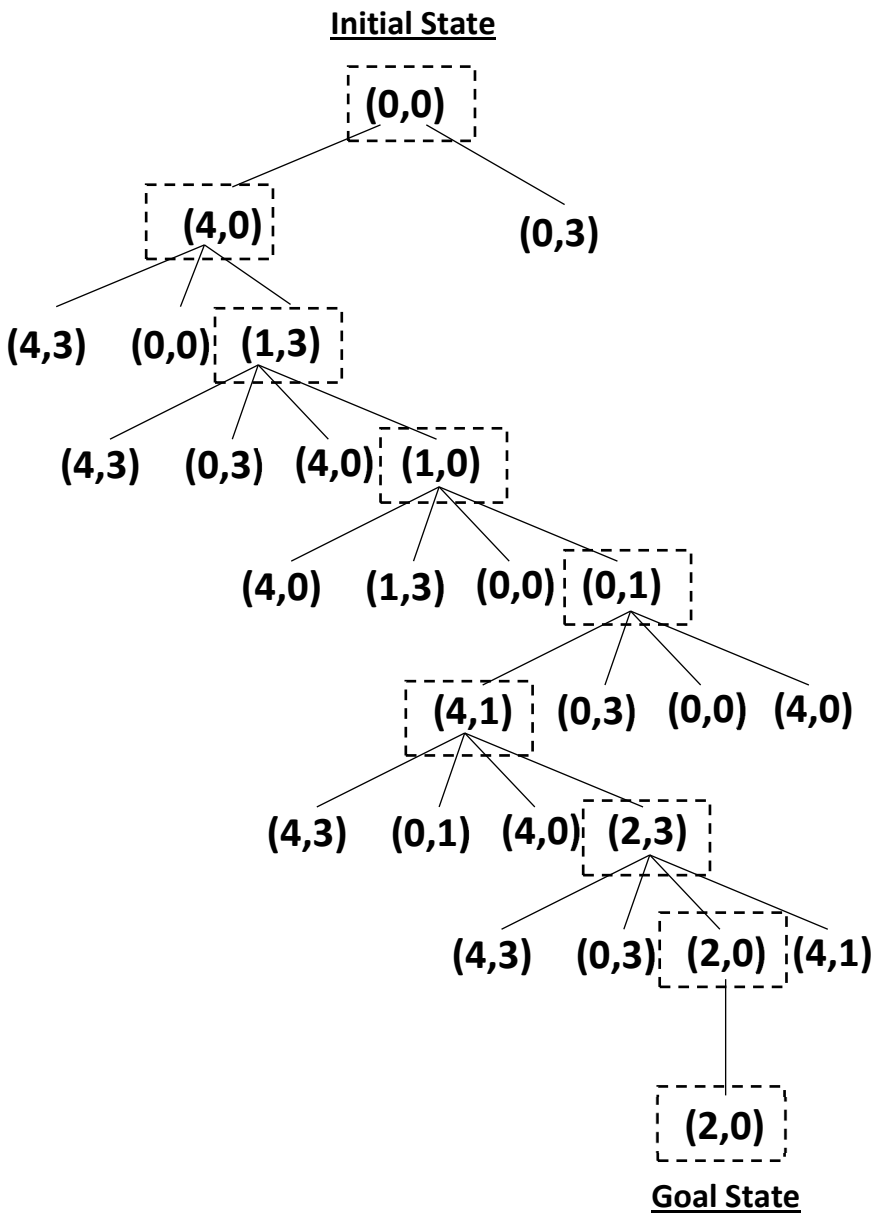
DEEPAK BHATTA

Formulation II.

Liters in 4-Liter jug	Liters in 3-Liter jug	Rule Applied	Meaning
0	0	-	Initial State
4	0	1	Fill 4-Liter jug.
1	3	6	Pour water from 4-Liter jug to 3-Liter jug.
1	0	4	Empty 3-Liter jug on ground.
0	1	8	Pour all the water from 4-Liter jug to 3-Liter jug.
4	1	1	Fill 4-Liter jug.
2	3	6	Pour water from 4-Liter jug to 3-Liter jug until 3-Liter jug is full.
2	0	4	Empty 3-Liter jug on ground.
2	0	-	Goal State Reached.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

State Space Tree Diagram for formulation II:



Try Yourself

Try to draw state space tree diagram for formulation I:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2.3 Well defined problems

Well-Defined Problem: - Clear, definite and well-formed problem.

Problem Solving by Search

An important aspect of intelligence is *goal-based* problem solving.

The **well-defined problems** have specific goals, **clearly defined** solution paths, and clear expected solutions. For example, 8-Puzzle Problem, Water Jug Problem, Tower of Hanoi etc.

The solution of many problems (e.g. Tic Tac Toe (noughts 0 and crosses x), timetabling, chess) can be described by finding a sequence of actions that lead to a desirable goal. Each action changes the *state* and the aim is to find the sequence of actions and states that lead from the initial (start) state to a final (goal) state.

A well-defined problem can be described by:

- **Initial state**
- **Operator or successor function** - for any state x returns s(x), the set of states reachable from x with one action
- **State space** - all states reachable from initial by any sequence of actions
- **Path** - sequence through state space
- **Path cost** - function that assigns a cost to a path. Cost of a path is the sum of costs of individual actions along the path
- **Goal test** - test to determine if at goal state

Extra:

Ill-defined Problem: - Not Clear, indefinite and ill-formed or not well-formed problem.

A problem that lacks one or more of these specified properties is an ill-defined problem, and most problems that are encountered in everyday life fall into this category.

- **Well-defined vs. ill-defined problems**: Problems where the goal or solution is recognizable--where there is a right answer (ex. a math or physics problem) vs. problems where there is no "right" answer but a range of more or less acceptable answers.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2.4 Constraint Satisfaction Problem

What is CSP ?

- Finite set of variables V_1, V_2, V_3, …….. V_n
- Finite set of Constraints C_1, C_2, C_3, …….. C_n
- Non Empty Domain of possible values for each variable D_1, D_2, D_3, …….. D_n
- Each Constraint C_1 limits the values that variables can take, e.g.$V_1 \neq V_2$

A state is defined as an assignment of values to some or all variables.

Solution to a CSP is an assignment to each variable such that each constraint is satisfied.

Some Examples of CSP problems are as follows:

- **8-Puzzle Problem**
- **N-Queen Problem (4x4, 8x8)**
- **Graph Coloring/ Map Coloring/ 3-Color Problem**
- **Crypt-Arithmetic Problem**
- **Crossword**
- **Sudoku (9x9)**

N Queen Problem:

N-Queens problem is a well-known Constraint Satisfactory Problem of Artificial Intelligence. In this problem, we have an **NxN** square grid board and we have **N queens** which need to be placed on them. The queens should be placed on the board in such a way so that it satisfies the below-mentioned constraints:

2 No row should contain more than one queen placed in it
3 No column should contain more than one queen placed in it.
4 Not more than one queen should be placed in the single diagonal.
5 No row or column should be left without any queen placed in it.

On summing up all the constraints, we can conclude that each row and each column should contain exactly one queen in them, neither more nor less than that.

In this series of problems, mostly there are grids whose size is even in number, like 4, 6, 8 and so on. It should be noted that the minimum number of the grid that we can have in this problem is 4, not less than that.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Here the **4-Queen** problem and the 8-Queen problem are the most popular in the N-Queen problem series.

There can exist many solutions for solving this problem, which mean that the solution to these problems is not unique. Yet, one of those solutions to both these types are given below:

4-Queens problem

In the **4-Queens problem**, we have a **4x4 grid** and we have **4 queens** to place on it. The layout for the **4-Queens problem** while satisfying all the constraints is as follows:

		Q	
Q			
			Q
	Q		

	Q		
			Q
Q			
		Q	

8-Queens problem

In the **8-Queens problem**, we have an **8x8 grid** and we have **8-queens** to place on it. The layout for the **8-Queens problem** while satisfying all the constraints is as follows:

					Q		
Q							
				Q			
	Q						
							Q
		Q					
						Q	
			Q				

Original

		Q					
							Q
			Q				
						Q	
Q							
					Q		
	Q						
				Q			

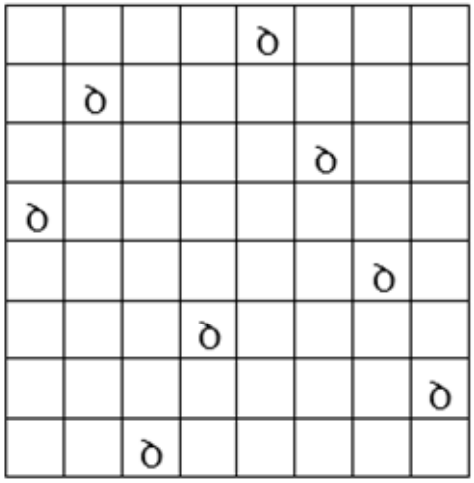
Mirroring(L=>R or R=>L)

			Q				
						Q	
		Q					
							Q
	Q						
				Q			
Q							
					Q		

						Q	
				Q			
		Q					
Q							
					Q		
							Q
	Q						
			Q				

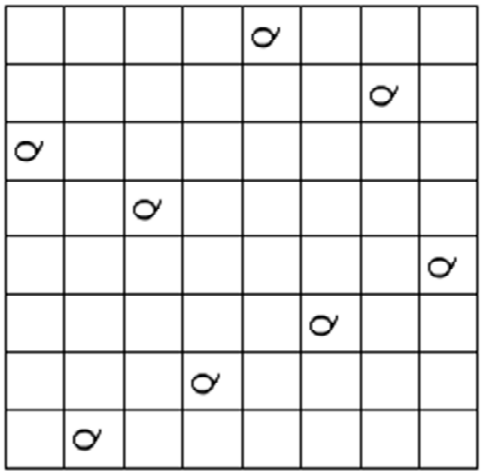
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Mirroring(U=>D or D=>U)



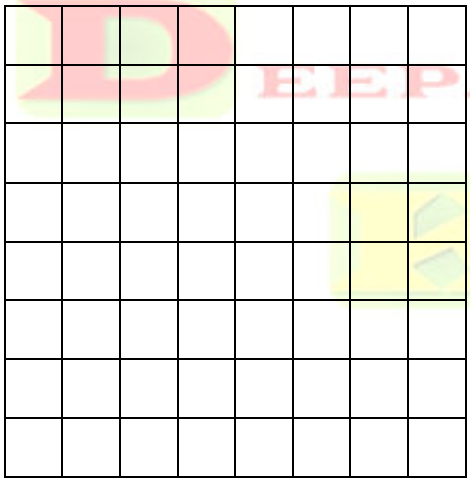
180° Rotation

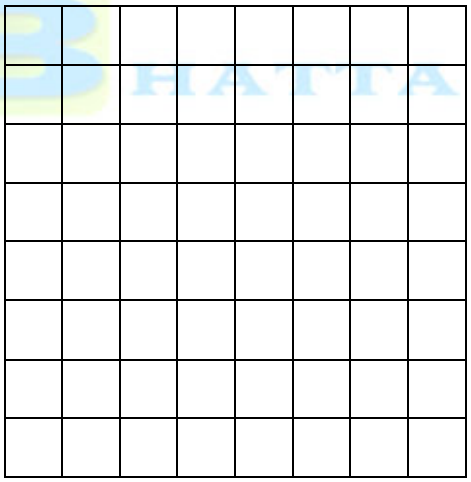
90° Rotation

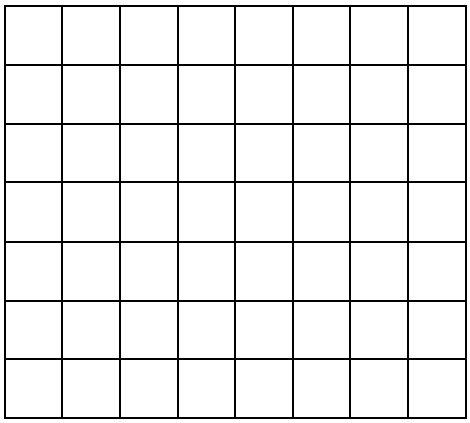


270° Rotation

Please try to solve other ways: *(Use Pencil and Eraser)*







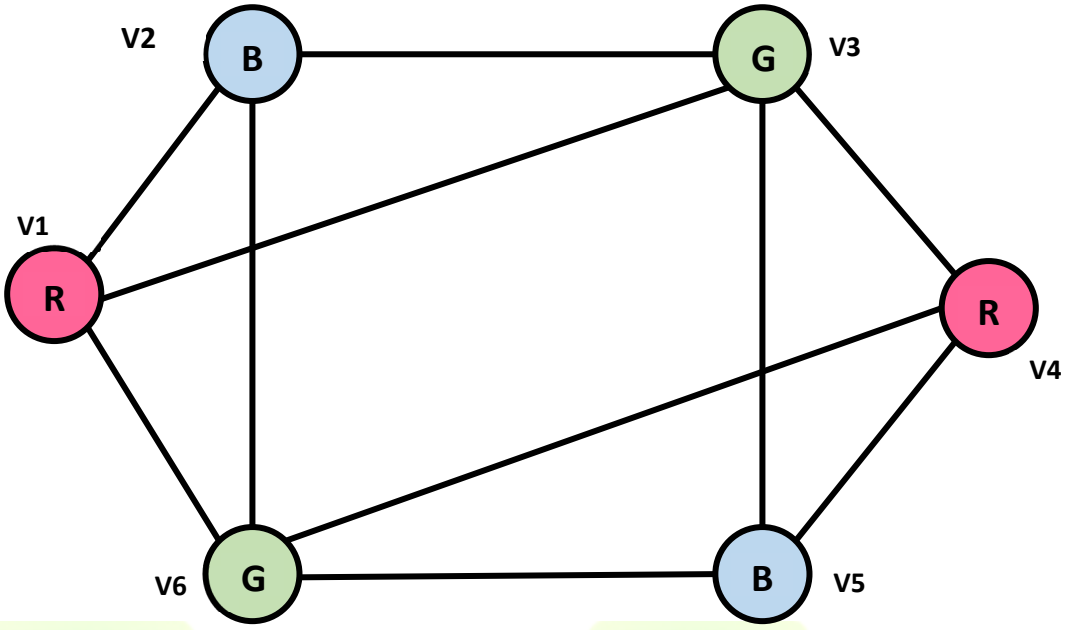
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

DEEPAK BHATTA KAJI

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Graph Coloring Problem/ Map Coloring Problem:

A classic CSP is the problem of coloring a map so that no adjacent regions have the same color.



Variables = V_1, V_2, V_3, V_4, V_6

Domain = R, G, B

Constraints = adjacent regions must have different colors

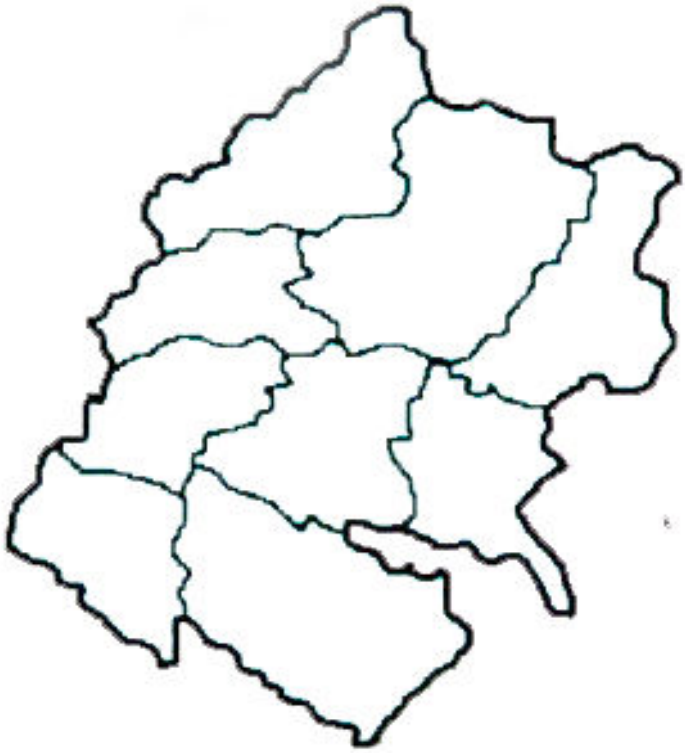
E.g: $V_1 \neq V_2$, $V_1 \neq V_6$, $V_1 \neq V3$

Means, $V_1 \neq V_2$, or ($V_1 \neq V_2$) in {(R,G), (R,B), (G,R), (G,B), (B,R), (B,G)}

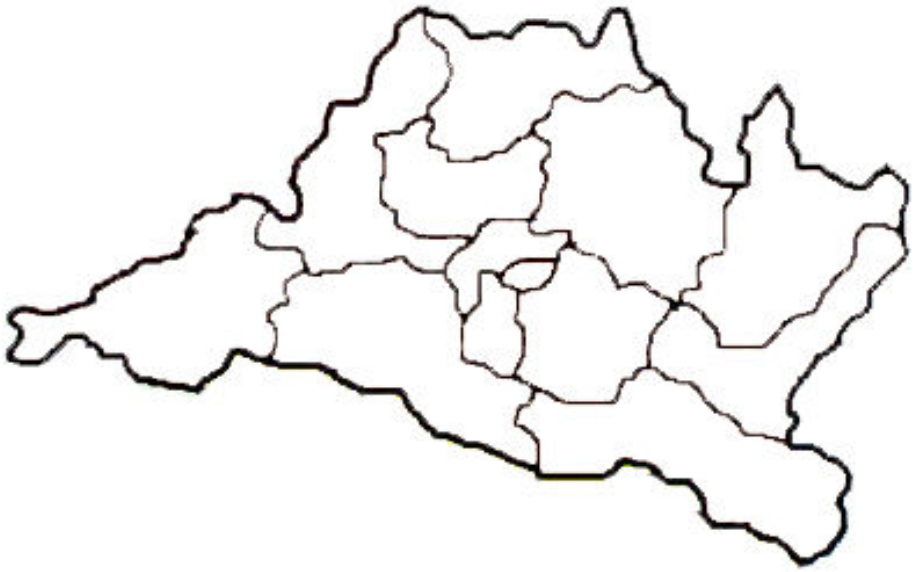
The Map of Nepal can be colored with 3 colors (R, G, B).

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Q1. Can the map of Sudur Paschim Pradesh (Province 7) be colored with 3 Color? If possible, try it out.



Q2. Can the map of (Province 3) be colored with 3 Color?.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Crypt-Arithmetic Problem:

The **Crypt-Arithmetic problem in** Artificial Intelligence is a type of encryption problem in which the written message in an alphabetical form which is easily readable and understandable is converted into a numeric form which is neither easily readable nor understandable. In simpler words, the crypt-arithmetic problem deals with the converting of the message from the readable plain text to the non-readable ciphertext. The constraints which this problem follows during the conversion is as follows:

1. A number 0-9 is assigned to a particular alphabet.
2. Each different alphabet has a unique number.
3. All the same, alphabets have the same numbers.
4. The numbers should satisfy all the operations that any normal number does.

Let us take an example of the message: SEND + MORE = MONEY.

Here, to convert it into numeric form, we first split each word separately and represent it as follows:

```
    S   E   N   D
    M   O   R   E
  ----------------
M   O   N   E   Y
```

These alphabets then are replaced by numbers such that all the constraints are satisfied. So initially we have all blank spaces.

We first look for the MSB in the last word which is **'M'** in the word **'MONEY'** here. It is the letter which is generated by carrying. So, carry generated can be only one. SO, we have **M=1**.

Now, we have **S+M=O** in the second column from the left side. Here **M=1**. Therefore, we have, **S+1=O**. So, we need a number for **S** such that it generates a carry when added with **1**. And such a number is **9**. Therefore, we have **S=9** and **O=0**.

Now, in the next column from the same side we have **E+O=N**. Here we have **O=0**. Which means **E+0=N** which is not possible. This means a carry was generated by the lower place digits. So, we have:

1+E=N ----------(i)

Next alphabets that we have are **N+R=E -------(ii)**

So, for satisfying both equations **(i) and (ii)**, we get **E=5 and N=6**.

Now, **R** should be **9**, but **9** is already assigned to **S**, So, **R=8** and we have **1** as a carry which is generated from the lower place digits.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Now, we have **D+5=Y** and this should generate a carry. Therefore, **D** should be greater than **4**. As **5, 6, 8** and **9** are already assigned, we have **D=7** and therefore **Y=2**.

Therefore, the **solution to the given Crypt-Arithmetic problem is**:

S=9; E=5; N=6; D=7; M=1; O=0; R=8; Y=2

Which can be shown in layout form as:

```
    9   5   6   7
    1   0   8   5
---------------------------
1   0   6   5   2
```

Some Examples of Cryptarithmetic Problem with Solution

(ADDITION)

1. LET + LEE = ALL

```
    L   E   T          1   5   6
    L   E   E          1   5   5
----------------   ----------------
    A   L   L          3   1   1
```

Hint: If Two characters are same then the result will be Even Number.

Here, L + L = any even number, so 1 + 1 = 2, *(if it takes carry then add it)*
E + E = any even number, so 5 + 5 = 10 *(0 is the even number)*

2. KANSAS + OHIO = OREGON

```
K   A   N   S   A   S          4   9   7   2   9   2
        O   H   I   O                  7   8   6   5
-------------------------------------  -------------------------------
O   R   E   G   O   N          5   0   3   1   5   7
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3. HERE + SHE = COMES

```
        H   E   R   E          9   4   5   4
            S   H   E          8   9   4
  -----------------------------------------
    C   O   M   E   S      1   0   3   4   8
```

E + E = 8
then,,E = 4 AND S = 8,
E + S = M
4 + 8 = 12
M = 2
H + 1 = 10 so, O = 0 C = 1,
now R + H = E = 4
H = 9
so R has to be 5 as 5 + 9 = 14 which leaves carry 1..
so M = 3
so R + H + O = 5 + 9 + 0 = 14

4. POINT + ZERO = ENERGY

```
        P   O   I   N   T          9   8   5   0   4
            Z   E   R   O          3   1   6   8
  -----------------------------------------
  E     N   E   R   G   Y      1   0   1   6   7   2
```

5. GO + TO = OUT

```
                G   O              8   1
                T   O              2   1
  -----------------------------------------
            O   U   T              1   0   2
```

Clearly, O = 1., as it is the carry generated by G + T. a number cannot start from 0 in cryptarithmetic addition.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Since O = 1, O + O = 1 + 1 =2. So, T = 2.
G + 2 = 10 + U.
If G = 9, U = 1. Which is not valid since O = 1.
So, G = 8 and U = 0.
Hence, O + U + T = 1 + 0 + 2 = 3

6. USA + USSR = PEACE

```
        U   S   A
        U   S   R
---------------------
    P   E   A   C   E
```

```
        9   3   2
    9   3   3   8
---------------------
1   0   2   7   0
```

USA + USSR = PEACE
Here P is carry , P = 1
when P = 1, E = 0 with carry 1 AND U = 9
A + R = E = 0 with carry 1.
so, A = 2 and R = 8
U + S = A = 2 with carry 1, S = 3
S + S + 1 = C, 3 + 3 + 1 = C = 7

7. EVER + SINCE = DARWIN

```
        E   V   E   R
    S   I   N   C   E
-----------------------------
D   A   R   W   I   N
```

```
        5   6   5   3
    9   7   8   2   5
-----------------------------
1   0   3   4   7   8
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

8. EAT + EAT + EAT = BEET

```
        E   A   T
        E   A   T
        E   A   T
  ----------------------------
    B   E   E   T
```

```
        4   8   0
        4   8   0
        4   8   0
  ----------------------------
    1   4   4   0
```

8 + 8 + 8 = 24
2 carry over
4 + 4 + 4 = 12 + 2 = 14
So, BEET = 1440
Note, In 3 word addition if there is carry over then count only 2

9. E A T + T H A T = A P P L E

```
        E   A   T
    T   H   A   T
  ----------------------------
  A   P   P   L   E
```

```
        8   1   9
    9   2   1   9
  ----------------------------
  1   0   0   3   8
```

digit number (EAT) + 4 digit number(THAT) = 5 digit number(APPLE)
If so, then A can be 1 and P can be 0.
Again here T is yielding a two digit number(10).
So there must be a carry 1 and T = 9.
Now the expression becomes
E 1 9
9 H 1 9
————————-
1 0 0 L E
Now it is clear that E = 8 and L = 3

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

10. NINE + FINE = WIVES

```
    N  I  N  E
    F  I  N  E
  ---------------
W   I  V  E  S
```

```
    N  I  N  E
    F  I  N  E
  ---------------
W   I  V  E  S
```

11. WAIT + ALL = GIFTS

```
    W  A  I  T
       A  L  L
  ---------------
G   I  F  T  S
```

```
    9  6  0  8
       6  7  7
  ---------------
1   0  2  8  5
```

12. FORTY + TEN + TEN = SIXTY

```
F   O  R  T  Y
       T  E  N
       T  E  N
  ---------------
S   I  X  T  Y
```

```
2   9  7  8  6
       8  5  0
       8  5  0
  ---------------
3   1  4  8  6
```

13. SCOOBY + DOOO = BUSTED

```
S   C  O  O  B  Y
       D  O  O  O
  ---------------
B   U  S  T  E  D
```

```
1   9  4  4  2  3
       7  4  4  4
  ---------------
2   0  1  8  6  7
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

14. S C O O B Y + D O O = B L I N K S

S	C	O	O	B	Y
			D	O	O
B	L	I	N	K	S

3	6	O	O		1
			D	2	2
3	6	2	9	5	3

Y + O = S, B + O = K, O + D = N , O = I , C = L ,S = B
so, we can take values S&B = 3, C&L = 6, O&I = 2 Y = 1, D = 7,N = 9,K = 5 .

15. BANANA + GUAVA = ORANGE

B	A	N	A	N	A
	G	U	A	V	A
O	R	A	N	G	E

2	4	9	4	9	4
	6	5	4	7	4
3	1	4	9	6	8

DEEPAK BHATT

16. HOW + MUCH = POWER

		H	O	W
	M	U	C	H
P	O	W	E	R

		7	0	5
	9	8	3	7
1	0	5	4	2

17. DAYS + TOO = SHORT

	D	A	Y	S
		T	O	O
S	H	O	R	T

	9	7	4	3
		5	2	2
1	0	2	6	5

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

18. TWO + DAYS = MORE

```
    T W O
  D A Y S
 -----------
  M O R E
```

```
    8 9 3
  6 4 1 2
 -----------
  7 3 0 5
```

19. CROSS + ROADS = DANGER

```
    C R O S S
    R O A D S
  -------------
  D A N G E R
```

```
    9 6 2 3 3
    6 2 5 1 3
  -------------
  1 5 8 7 4 6
```

20. YOUR + YOU = HEART

```
    Y O U R
      Y O U
  -------------
  H E A R T
```

```
    9 4 2 6
      9 4 2
  -------------
  1 0 3 6 8
```

21. TWO + TWO = FOUR

```
    T W O
    T W O
  -------------
  F O U R
```

```
    9 3 8
    9 3 8
  -------------
  1 8 7 6
```

Here are other possibilities:

938+938=1876
928+928=1856
867+867=1734
846+846=1692
836+836=1672
765+765=1530
734+734=1468

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

22. ABC + DEF = GHI

```
    A   B   C
    D   E   F
----------------
    G   H   I
```

```
    2   6   9
    1   3   8
----------------
    4   0   7
```

This question may have thousand solutions because the characters are distinct. Make sure A + D must not generate a carry.

23. AB + CD = EF

```
    A   B
    C   D
----------------
    E   F
```

```
    4   9
    1   8
----------------
    6   7
```

This question may have more than hundreds of solutions because the characters are distinct. Make sure A + C must not generate a carry.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2.5 Game Playing

A Game can be defined by the following:

- ✓ initial state (How the board is setup).
- ✓ the operation (which define the legal moves.
- ✓ a terminal test (which says when the game is over). and
- ✓ a utility or payoff function (which says who won and by how much).

Perfect Information: Information is *shown* or clear. See the exact state to the game. Chess is an example of a game with perfect information as each player can see all the pieces on the board at all times. Other examples of games with perfect information include tic-tac-toe, checkers, infinite chess, and Go

Imperfect Information: Information is Hidden. Card games where each player's cards are *hidden* from other players such as poker and bridge are examples of games with imperfect information.

Game Playing Problem:

- ✓ States where the game has ended are called terminal states.
- ✓ In two-player game, assume one is called MAX (tries maximize utility) and one is called MIN.
- ✓ In the search tree, first layer is more by MAX, next layer by MIN, and alternate to terminal state.
- ✓ Each layer in the search is called a ply.
  *a **ply** is one turn taken by one of the players.*

Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game.

Both players try to win the game. So, both of them try to make the best move possible at each turn. Searching techniques like BFS (Breadth First Search) are not accurate for this as the branching factor is very high, so searching will take a lot of time.

So, we need another search procedure that improve –

Generate procedure so that only good moves are generated.

Test procedure so that the best move can be explored first.

Sometimes Generate and Test procedure is known as **Hit and Trial**.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

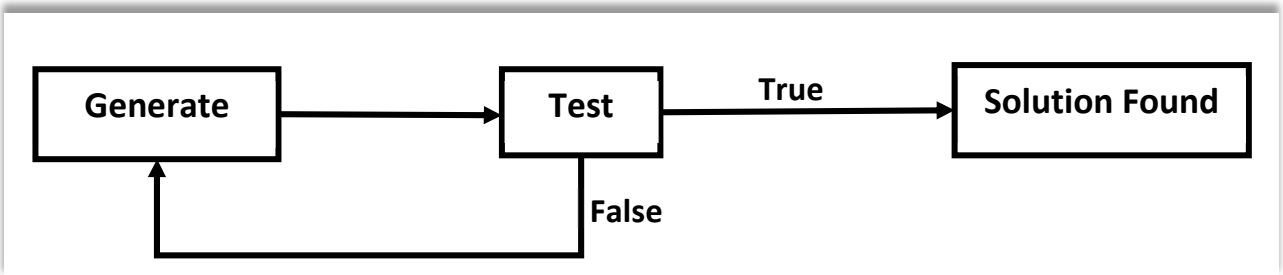


Fig.: Flow of Generate and Test Procedure

Algorithm:

1. Generate a possible solution.
2. Test to see if this is actually a solution.
3. Quit if a solution has been found otherwise - return to step 1.

The most common search technique in game playing is Minimax search procedure. It is depth-first depth-limited search procedure. It is used for games like chess and tic-tac-toe.

Examples: MiniMax algorithm, Alpha Beta Pruning.

These examples will be discussed in the next chapter

2.6 Production System

Rule

Situation/Condition ➔ Action

LHS (Left Hand Side) ***RHS (Right Hand Side)***

A production system is nothing but a set of rules, each rule consists of LHS and RHS where, LHS patterns determines the applicability of rules and a RHS describes the operations to be performed if the rule is applied.

Simple, Production system are the rules of the form C ➔ A, where LHS is known Condition and RHS is known as Action.

LHS described as Applicability of Rule.

RHS described Operation to be Performed.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

For Example. Water Jug Problem, Missionaries and Cannibal ***(These problem production rule systems are already done. See above)***

More Generalize Definition:

If one adopts a system with production rule and rule Interpreter then that system is known as Production system.

Production System is a model of computation that provides pattern directed search control using set of:

1. **Production Rules**
   - ✓ $C_i \rightarrow A_i$
     - i. C_i = Conditional Part (If…. Else)
     - ii. A_i = Action Part
       - ✓ Primitive to Action
       - ✓ A call to another production rule
       - ✓ Set of instructions/actions/programs
2. **Working Memory (Knowledge Database)**
   - ✓ Information appropriate for particular task.
3. **Control Strategy**
   - ✓ Specifies the order in which rule will be applied and it resolves conflicts if any
   - ✓
4. **Rule Applier**
   - ✓ Checks current state with LHS of rule in Knowledge Database and finds appropriate rule to apply

Types of Production System or Characteristics:

- ✓ Monotonic Production System
- ✓ Non-Monotonic Production System
- ✓ Partially Commutative Production System
- ✓ Commutative Production System

1. **Monotonic Production System:** In monotonic production system all conclusions are still valid after adding information to the existing information. Example: Theorem Proving (Pythagoras Theorem)

2. **Non-Monotonic Production System:** In non-monotonic production system some conclusion can be invalidated by adding more information to the existing information. Example: Robot Navigation (GPS). Like we human beings even change our conclusion if new information is added.

3. **Partially Commutative Production System:** A system in which application of a particular sequence of rules transforms state x to state y then any permutation of these

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

rules that is allowable also transform state x to state y. Example: ABCD is applied then it takes to x=> y If I use Permutation of ABCD or Combination of ABCD like BCDA then also it takes stare x to state y. || AxB = BxA

4. **Commutative Production System:** Combination of Monotonic + Partially

---

Q. Define and describe the difference between knowledge, belief, hypothesis and data.

Ans:

Knowledge: can be defined as the body of facts and principles accumulated by humankind or the act, fact, or state of knowing.

Belief: It is defined as essentially any meaningful and coherent expression that can be represented. Thus, a belief can be true or false.

Hypothesis: It is defined as justified belief that is not known to be true. Thus, a hypothesis is a belief that is backed up with some supporting evidence, but it may still be false. In other words, it is a preliminary assumption or tentative explanation that accounts for a set of facts, taken to be true for the purpose of investigation and testing.

Data: Data in computer terminology mean raw facts and figures. For example, 'DBK', 11716713, 'A' are data. Data are processed to form information.

Information: Data arranged in useful and meaningful form is known as information. For example, 'DBK, whose roll number is 11716713, has got grade A' is an information as it is conveying some meaning.

THE END

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Unit 3 6 hrs.

Search Techniques

3.1 Uninformed Search Techniques:
Depth First Search (DFS), Breadth First Search (BFS), Depth Limit Search (DLS), and Search Strategy Comparison

3.2 Informed Search Techniques:
Hill Climbing, Best First Search (BFS), Greedy Search (GS), A* Search, Simulated Annealing, Genetic Algorithms

3.3 Adversarial Search Techniques:
Min-Max Procedure, Alpha-Beta Procedure

3.1 Uninformed Search Techniques:

DEEPAK BHATTA

Uninformed Search:

An uninformed search can also be called as **blind search**. This search technique has no information about its domain. The only thing that a blind search can do is distinguish a non-goal state from a goal state. *Depth First search* and *Bread First search* are good examples of Blind (or Uninformed) search strategies.

Uninformed search does not have additional information about states beyond problem definition. Uninformed search is a class of general-purpose search algorithms which operates in brute force-way.

Following are the different Uninformed Search Techniques:

a. Breadth-first search

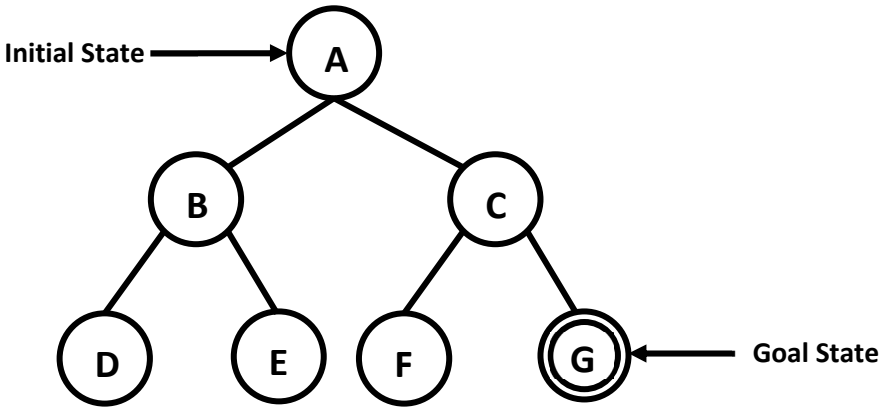
b. Depth-first search

c. Depth-limited search

d. Iterative deepening depth-first search

e. Bidirectional search

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



- ⇨ We don't know the cost of start node to goal node.
- ⇨ We don't know the cost path of any node.
- ⇨ We don't know the steps to complete or reach to the goal state.
- ⇨ We just search blindly for reaching our goal state.

Depth First Search (DFS)

Depth-first search is a recursive algorithm for traversing a tree or graph data structure.

It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.

DFS uses a stack data structure for its implementation.

The process of the DFS algorithm is similar to the BFS algorithm.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

Disadvantage:

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Algorithm:

1. Start
2. Push root node to the stack
   - Mark root node as visited
   - Print root node in output sequence
3. Check Top stack
   - **If** (stack == Empty) **Goto Step 6**
4. Check adjacent of Top stack

   If (adjacent == not_visited)
   - Push node to the top of the stack
   - Mark node as visited
   - Print node in output sequence

   Else *i.e. (Adjacent == Visited)*
   - Pop- Top stack
5. Goto Step 3
6. Stop

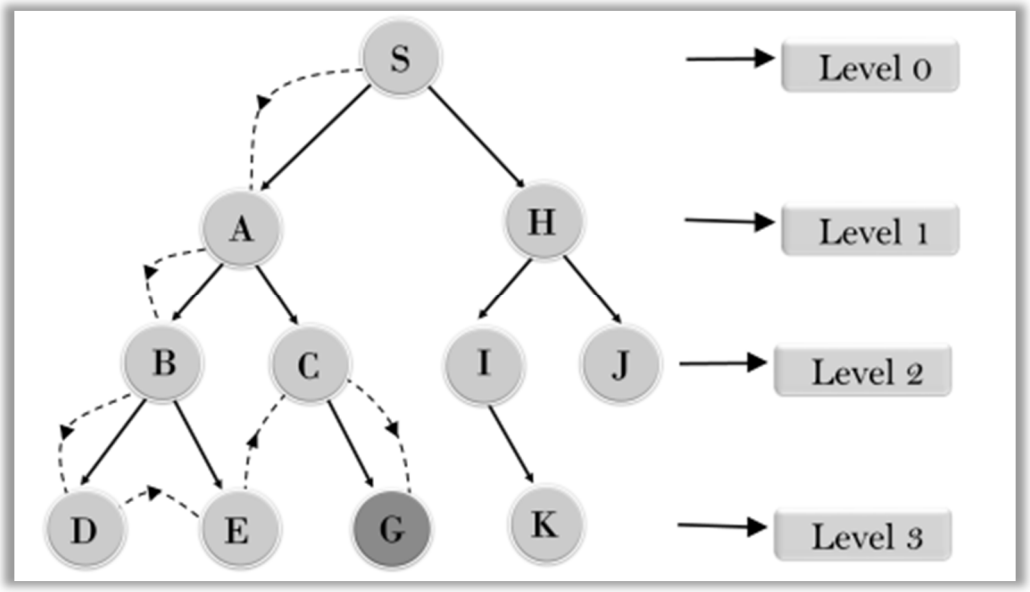
Example:

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

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

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

S---> A---> B----> D---> E----> C---> G



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

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

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

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

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

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

Breadth First Search (BFS)

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

BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.

The breadth-first search algorithm is an example of a general-graph search algorithm.

Breadth-first search implemented using FIFO queue data structure.

Advantages:

- BFS will provide a solution if any solution exists.
- If there are more than one solution for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Algorithm:

1. Start
2. Point out the pointer to root node
   - Mark root node as visited
   - Print root node in output sequence
3. Search Pointer and Check for its adjacent nodes
   - **If** (Adjacent == Not_Visited)
     - Enqueue node into the Queue
     - Mark it as visited node
     - Goto Step 3

     Else *i.e. (Adjacent == Visited)*
     - Update the pointer to Top of the queue
     - Dequeue Top of the queue
     - Print node in output sequence
4. Check for the empty queue and pointer node has visited adjacent node

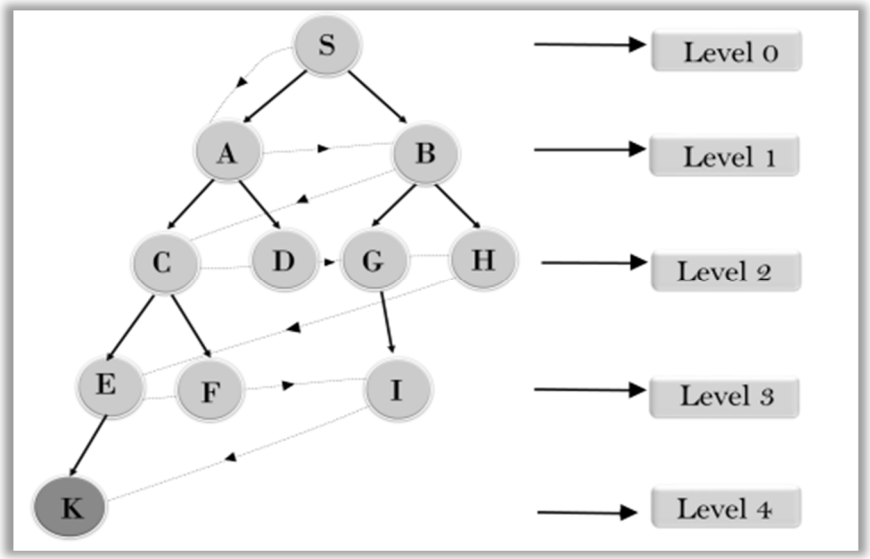
   If ((Queue == Empty) && (Pointer_Node_Adjacent == Visited))
   - Goto Step 5

   Else
   - Goto Step 3
5. Stop

Example:

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

S---> A---> B----> C---> D----> G---> H---> E----> F----> I----> K



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Depth Limit Search (DLS)

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

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

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

Advantages:

- Depth-limited search is Memory efficient.

Disadvantages:

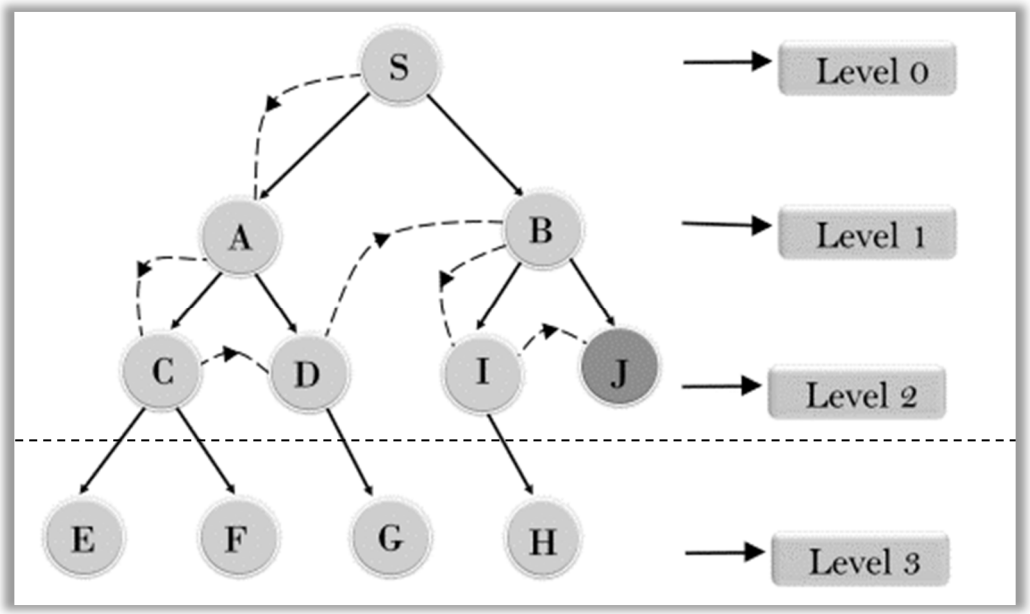
- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

Example:

In the below tree structure, we have shown the traversing of the tree using DLS algorithm from the root node **S** to goal node **J**. DLS search algorithm traverse in depth and has some restriction i.e. limit here **limit=2** so, do not cross the limit beyond the dotted line, and the traversed path will be:

S---> A---> C----> D---> B----> I---> J

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Completeness: DLS search algorithm is complete if the solution is above the depth-limit.

Time Complexity: Time complexity of DLS algorithm is **O(b^{ℓ})**.

Space Complexity: Space complexity of DLS algorithm is O**($b \times \ell$)**.

Optimal: Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if ℓ>d.

Iterative Deepening Depth First Search (IDDFS)

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

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

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

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Advantages:

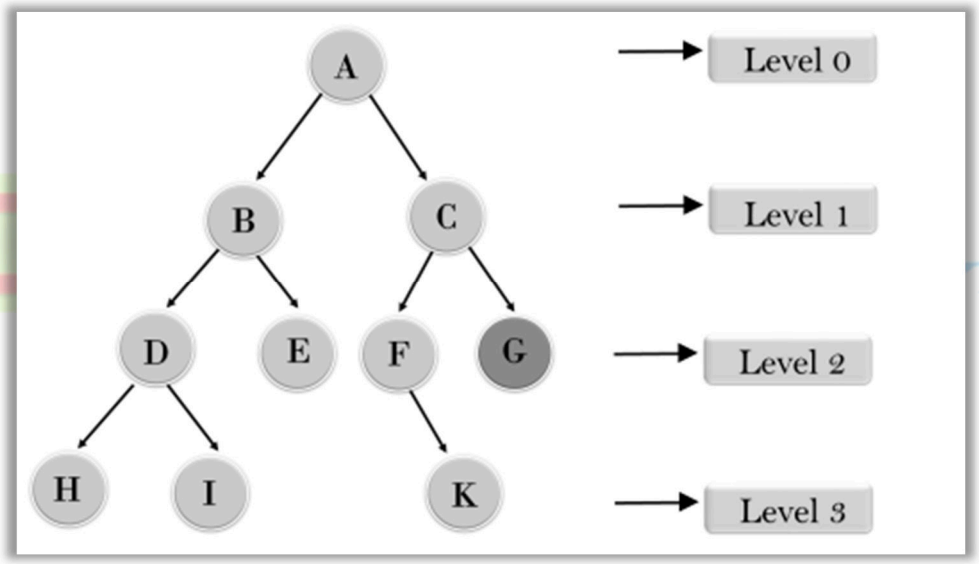
- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

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

Example:

Following tree structure is showing the iterative deepening depth-first search. IDDFS algorithm performs various iterations until it does not find the goal node. The iteration performed by the algorithm is given as:



1'st Iteration------> A

2'nd Iteration-----> A, B, C

3'rd Iteration------> A, B, D, E, C, F, G

4'th Iteration------> A, B, D, H, I, E, C, F, K, G

In the fourth iteration, the algorithm will find the goal node.

Completeness:

This algorithm is complete is if the branching factor is finite.

Time Complexity:

Let's suppose b is the branching factor and depth is d then the worst-case time complexity is **$O(b^d)$**.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Space Complexity:

The space complexity of IDDFS will be **O(bd)**.

Optimal:

IDDFS algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

Search Strategy Comparison

Criteria	BFS	DFS	Depth-Limit	IDS
Complete?	Yes	No	No	Yes
Time	$O(b^d)$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^d)$	O(bm)	O(bl)	O(bd)
Optimal?	Yes	No	No	Yes

Where,

b=Branching factor

d=Depth of the shallowest goal

m=Maximum depth of search tree

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3.2 Informed Search Techniques:

Informed Search:

Informed Search is also known as Heuristic Search. Heuristic search is an AI search technique that employs heuristic for its moves. Heuristic is a rule of thumb that probably leads to a solution. Heuristics play a major role in search strategies because of exponential nature of the most problems. Heuristics helps to reduce the numbers of alternatives from an exponential number to a polynomial number.

In Artificial Intelligence, heuristic search a general meaning, and a more specialized technique meaning. In a general sense the term heuristic is used for any advice that is often effective, but is not guaranteed to work in every case. Within the heuristic search architecture, however, the term heuristic usually refers to the special case of a heuristic function.

Heuristic Algorithm:

The term heuristic is used for algorithms which find solutions among all possible ones, but they do not guarantee that the best will be found, therefore they may be considered as approximately and not accurate algorithms.

These algorithms, usually find a solution close to the best one and they find it fast and easily.

Some times these algorithms can be accurate, that is they actually find the best solution, but the algorithm is still called heuristic until this best solution is proven to be the best solution.

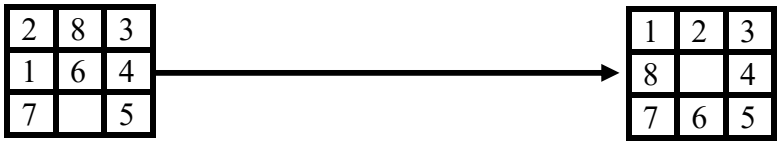
Summary of Informed / Heuristic Search:

- ⇨ Heuristic is derived from **heuriskein** in Greek, meaning ***"to find"*** or ***"to discover"***
- ⇨ Heuristic value will be associated to each node as an additional information to traverse the tree.
- ⇨ A solution cost estimation is used to guide the search.
- ⇨ The optimal solution, or even a solution is not guaranteed.
- ⇨ Based on h(n)- Estimated cost of path to goal ("Remaining path cost").
- ⇨ You cannot reach the node if you don't have additional information.
- ⇨ Heuristic Search which can give an optimized solution.
- ⇨ The purpose of heuristic function is to guide the search process in the most profitable path among all the available.
- ⇨ Artificial intelligence heuristic search AKA guided search. Heuristic in AI is a method that might not always find best solution but it guarantees to find a good solution in a reasonable time.
- ⇨ Heuristic in AI sacrifices completeness but increases efficiency.
- ⇨ Heuristic have heuristic function which estimates cost of path.
    - $\boldsymbol{f(n) = g(n) + h(n)}$

Where, $\boldsymbol{f(n) = Evaluation\ function\ \ or\ \ Total\ Path\ Cost}$

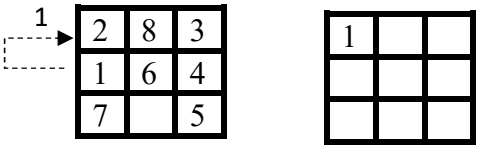
$\boldsymbol{g(n) = Local\ Path\ Cost}$ $\boldsymbol{h(n) = Heuristic\ Cost}$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



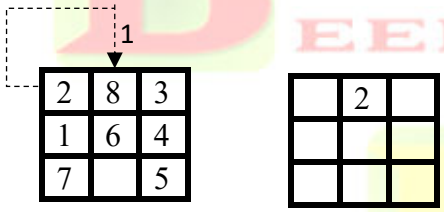
Start from number 1 :

Observe Initial State where is 1 located and try to match it with goal state, do a calculation at how many shift will I get the position of 1 like in the goal state *(for this you need to count the block either from the left, right or from the top, bottom)*. To shift 1 use always shortest path. At 1 shift it will reach to the goal state position. So, count it as 1 and write it in a sum of sequence.



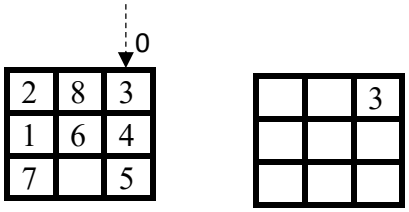
For number 2 :

Observe Initial State where is 2 located and try to match it with goal state, do a calculation at how many shift will I get the position of 2 like in the goal state *(for this you need to count the block either from the left, right or from the top, bottom)*. To shift 2 use always shortest path. At 1 shift it will reach to the goal state position. So, count it as 1 and write it in a sum of sequence.



For number 3 :

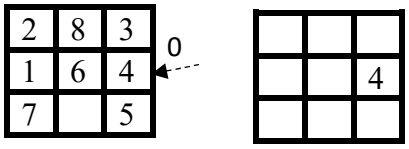
Observe Initial State where is 3 located and try to match it with goal state, do a calculation at how many shift will I get the position of 3 like in the goal state *(for this you need to count the block either from the left, right or from the top, bottom)*. To shift 3, use always shortest path. At 0 shift it will reach to the goal state position. No. 3 is already in the goal state. So, count it as 0 and write it in a sum of sequence.



For number 4 :

Observe Initial State where is 4 located and try to match it with goal state, do a calculation at how many shift will I get the position of 4 like in the goal state *(for this you need to count the block either from the left, right or from the top, bottom)*. To shift 4, use always shortest path. At 0 shift it will reach to the goal state position. No. 4 is already in the goal state. So, count it as 0 and write it in a sum of sequence.

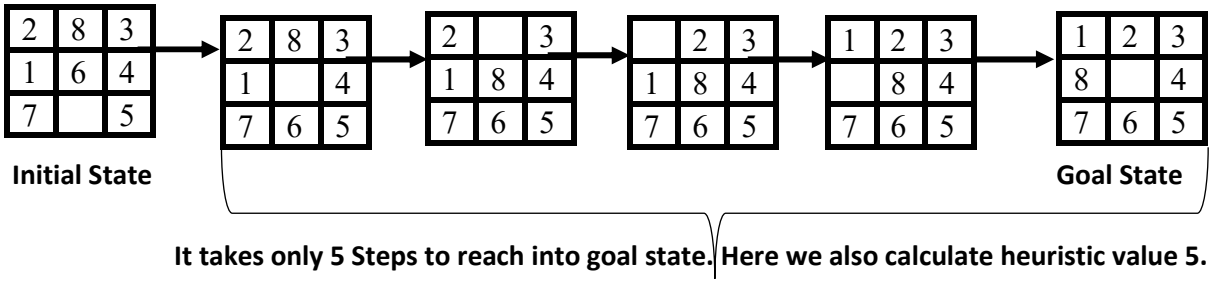
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Like wise we do a calculation for remaining 5, 6, 7, 8 numbers

1 + 1 + 0 + 0 + 0 + 1 + 0 + 2 = 5

So, at 5 steps we'll reach at goal state. Here, h is the heuristic value.



Remember this will not always give optimal result, but here we got exact cost to reach into goal state.

Hill Climbing (HC)

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.
- Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.
- It is also called greedy local search as it only looks to its good immediate neighbor state and not beyond that.
- A node of hill climbing algorithm has two components which are state and value.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Features of Hill Climbing:

Following are some main features of Hill Climbing Algorithm:

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

Types of Hill Climbing Algorithm:

- Simple hill Climbing:
- Steepest-Ascent hill-climbing:
- Stochastic hill Climbing:

1. Simple Hill Climbing:

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

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

2. Steepest-Ascent hill climbing:

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

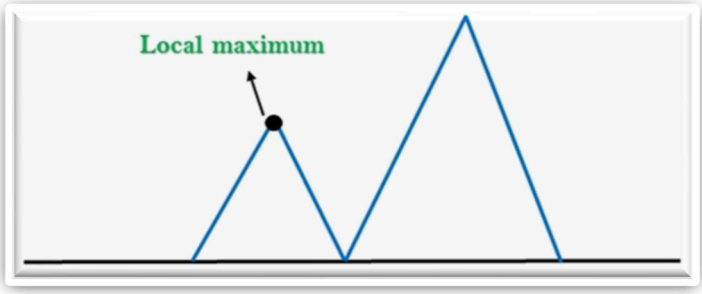
3. Stochastic hill climbing:

Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm selects one neighbor node at random and decides whether to choose it as a current state or examine another state.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

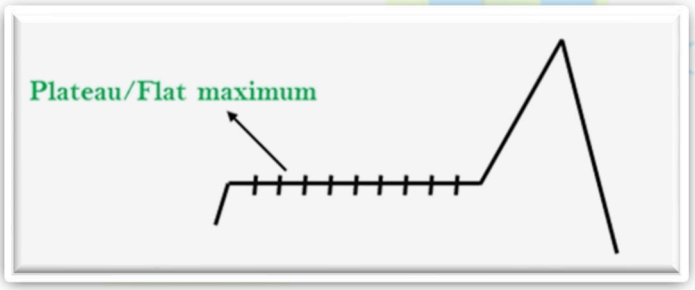
Limitations/Problems in Hill Climbing Algorithm:

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



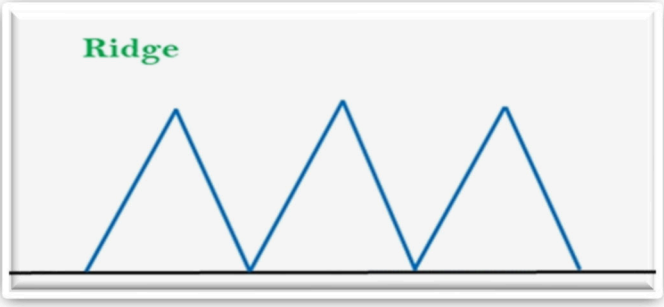
Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.

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



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

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



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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Algorithm:

1. Start
2. Evaluate the initial state
    a. If it is Goal state then Goto Step 6.
3. Loop until a solution is found or there are no new operators left.
4. Select and apply new operator.
5. Evaluate new state:
    a. If it is goal state, then Goto Step 6.
    b. Else If it is better than current state then
        i. Make it as New Current State.
    c. Else If it is not a better than current state then Goto Step 3.
6. Stop

Example:

Check out in previous chapter.

Hill Climbing example we already did in previous chapter **(8 Puzzle Problem).** To find a goal state at optimal (minimum) time.

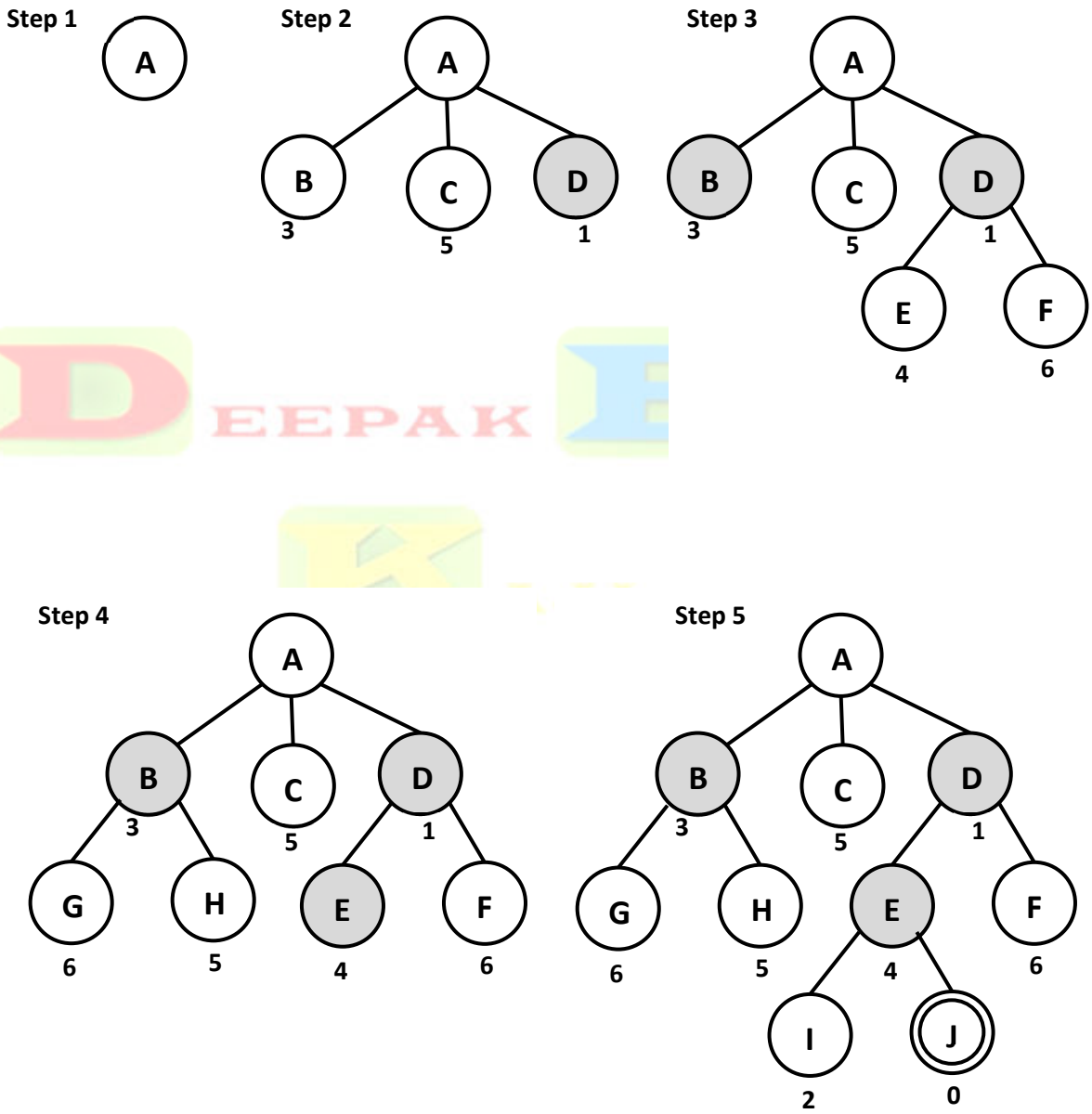
Best First Search (BFS)

- ✓ A combination of **DFS** *(Higher Time Complexity)* and **BFS** *(Higher Space Complexity)*.
- ✓ At each step using an appropriate heuristic function, the most promising node is selected.
- ✓ If the node is a solution quit.
- ✓ Otherwise, add it to the set of nodes generated so far and carry on with generating and selecting the next node.
- ✓ Expand only those nodes which has least cost or less Evaluation function f(n).
- ✓ Best-First Search is a search algorithm which explores a graph by expanding the most promising node chosen according to a specified rule.
- ✓ Best-First Search maintain Priority Queue.
- ✓ It has two list:
    - **Open List:** (Visited, Not Expanded/Explored)
        - Nodes that have been generated, but have not examined or expanded
    - **Close List:** (Visited, Expanded/Explored)
        - Nodes that have already been examined, whenever a new node is generated, check whether it has been generated before.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- ✓ Every node in the search space has an evaluation function (heuristic function) associated with it.
- ✓ **Depth First Search:** Not all competing branches having to be expanded or **Backtrack.**
- ✓ **Breadth First Search:** Not getting trapped on dead-end paths or **No dead-end trap.**
- ✓ Combining the two is to follow a single path at a time, but switch paths whenever some competing path look more promising than the current one.

Simple Example:



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Algorithm:

1. OPEN= {Initial State}
2. Loop until a goal is found or there are no node left in OPEN:
    a. Pick the best node in OPEN.
    b. Generate its Successors.
    c. For each Successor:
        i. New → Evaluate it, add it into OPEN, record its parent.
        ii. Generate Before → Change parent, Update Successors.

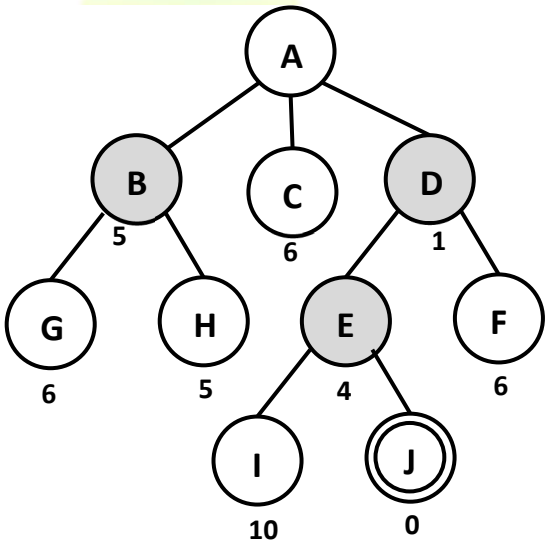
Advantages:

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

Disadvantages:

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

Example:



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

Expand the nodes of A and put in the CLOSED list

Initialization:

Open List: [D, B, C] **Closed List**: [A]

Iteration 1:

Open List: [E, B, C, F] **Closed List**: [A, D]

Iteration 2:

Open List: [J, B, C, F, I] **Closed List**: [A, D, E]

Iteration 3:

Open List: [B, C, F, I] **Closed List**: [A, D, E, **J**] *//Reach to the Goal State*

Hence the final solution path will be: **A----> D-----> E----> J**

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

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

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

Optimal: Best first search algorithm is not optimal.

Greedy Search (GS)

- A search method of selecting the best local choice at each step-in hope of finding an optimal solution.
- Does not consider how optimal the current solution is.
- At each step, uses a heuristic to estimate the distance (cost) of each local choice from the goal.
- A greedy algorithm always makes the choice that looks best at the moment.
- Greed makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
- Greedy algorithms do not always yield/produce optimal solutions, but for some problems they do.
- You have to be greedy, so that best solution can be taken up.
- The decision is taken on the basis of current available information without worrying about the effect of current decision in future.
- **Feasible solution**, that may or may not **optimal solution**.
- **Feasible solution→** Any subset that satisfy given criteria (condition)
- **Optimal solution→** Best or most favorable solution (i.e. Maximum or Minimum Value).

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- Select the best local choice at each step. NO matter what will happen in the upcoming nodes/future value.
- Once a decision has been made, it is never reconsidered.
- Obtain best solution than Best-First Search.
- But No guaranteed the optimum solution.

Features / Characteristics:

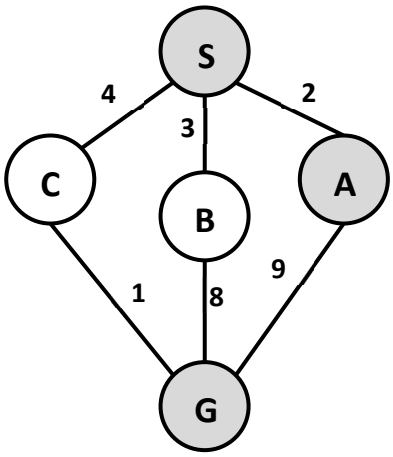
- ✓ To construct the solution in an optimal way. Algorithm maintains two sets:
    - One contains chosen items.
    - Another contains rejected items.
- ✓ Greedy algorithm makes good local choices
    - An optimal solution
    - Feasible solution

Areas of Applications:

- ✓ Finding Shortest Path.
- ✓ Finding Minimum Spanning Tree (Prim's or Kruskal Algorithms).
- ✓ Job Sequencing with deadline.
- ✓ Fractional Knapsack Problem.

DEEPAK BHATTA

Example:



S ----> A ----> G 2 + 9 = 11

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

A* Search

It is also known as A star. A* is the algorithm that is widely used in path finding and graph traversal, which is the process of finding a path between multiple points, called "nodes".

Worst case performance O(|E|) = O(b^d)
Worst case space complexity O(|V|) = O(b^d)

The A* algorithm combines features of uninformed cost search and pure heuristic search to efficiently compute optimal solutions. The A* algorithm is a Best First search algorithm in which the cost associated with a node is $\boldsymbol{f(n) = g(n) + h(n)}$

Where, $\boldsymbol{f(n) = Evaluation\ function\ \ or\ \ Total\ Path\ Cost}$

$\boldsymbol{g(n) = Local\ Path\ Cost}$ $\boldsymbol{h(n) = Heuristic\ Cost}$

A* is the best-known form of Best First Search. It avoids expanding paths that are already expensive, but expands most promising path first. A* algorithm guides an optimal path to a goal if the heuristic function $\boldsymbol{h(n)}$ is admissible, meaning it never overestimates actual costs.

DEEPAK BHATTA

Algorithm:

Step1: Place the starting node in the OPEN list.

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

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

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

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

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

Advantages:

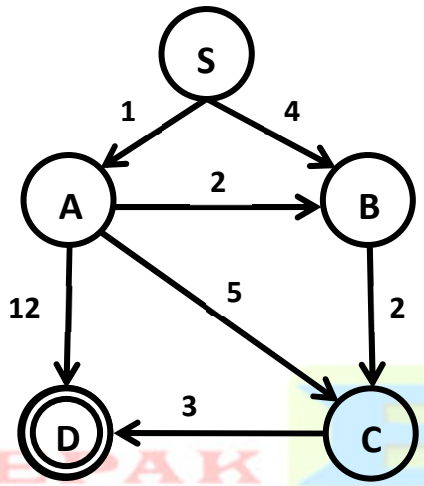
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Disadvantages:

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

Example:



Heuristic Values

S	7
A	6
B	2
C	1
D	0

Note:

D is a goal node so heuristic value is always 0 for every goal node.

Sol^n:

$$f(n) = g(n) + h(n)$$

S →
= 0+7 = 7

S → A
= 1+6 = 7

S → B
= 4+2 = 6

S → A → B
= 1+2+2 = 5

S → A → C
= 1+5+1 = 7

S → A → D
= 1+12+0 = 13

S → A → B → C
= 1+2+2+1 = 6

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

S → A → B → C → D
= 1+2+2+3+0 = 8

S → B → C → D
= 4+2+3+0 = 9

S → A → C → D
= 1+5+3+0 = 9

Therefore, **S → A → B → C → D = 8**

We will follow the path to reach the goal state because it has low cost path, Hence, it is optimized solution.

Points to remember:

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

Complete: A* algorithm is complete as long as:

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

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

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

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

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

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Simulated Annealing

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

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

Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate **"survival of the fittest"** among individual of consecutive generation for solving a problem. Each generation consist of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

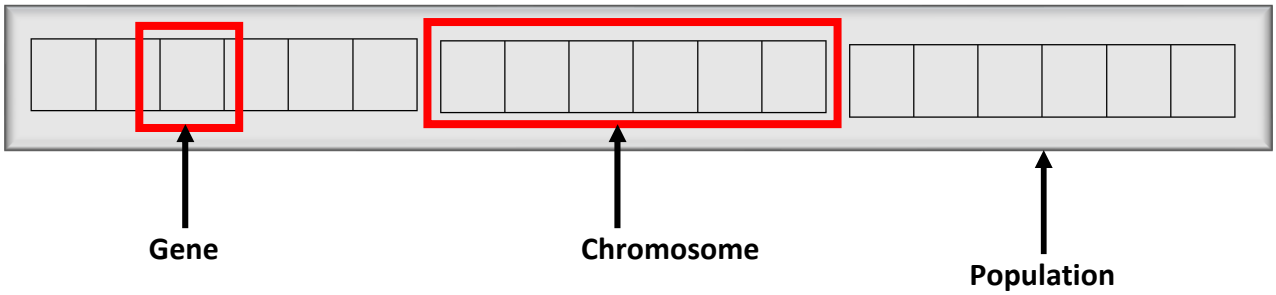
Foundation of Genetic Algorithms

Genetic algorithms are based on an analogy with genetic structure and behavior of chromosome of the population. Following is the foundation of GAs based on this analogy –

- Individual in population compete for resources and mate.
- Those individuals who are successful (fittest) then mate to create more offspring than others.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- Genes from "fittest" parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.
- Thus, each successive generation is more suited for their environment.



Fitness Score

A Fitness Score is given to each individual which **shows the ability of an individual to "compete"**. The individual having optimal fitness score (or near optimal) are sought.

The GAs maintains the population of n individuals (chromosome/solutions) along with their fitness scores. The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce **better offspring** by combining chromosomes of parents. The population size is static so the room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals eventually creating new generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solutions will arrive while least fit die.

Each new generation has on average more **"better genes"** than the individual (solution) of previous generations. Thus, each new generation have better **"partial solutions"** than previous generations. Once the offspring produced having no significant difference than offspring produced by previous populations, the population is converged. The algorithm is said to be converted to a set of solutions for the problem.

Operators of Genetic Algorithms

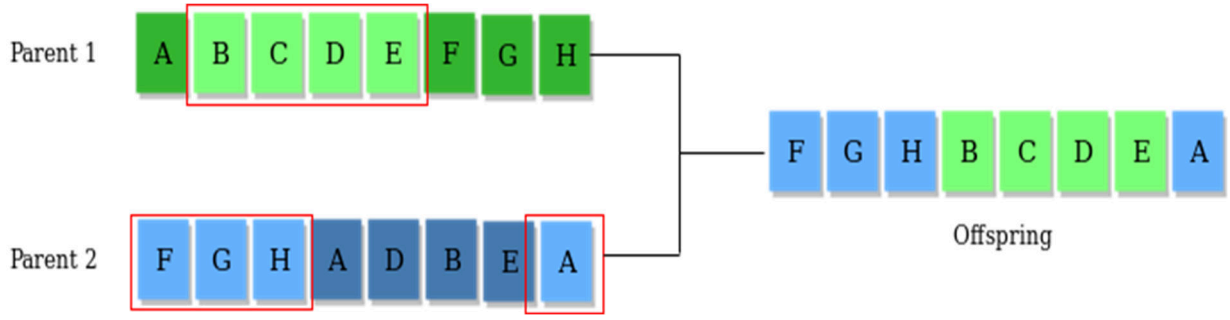
Once the initial generation is created, the algorithm evolves the generation using following operators –

1) Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to the successive generations.

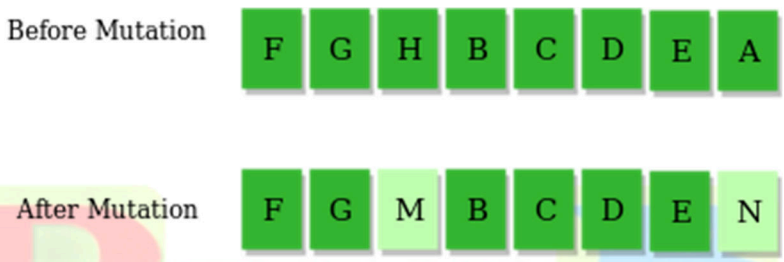
2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).

For example –

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



3) Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence. For example –



The whole algorithm can be summarized as –

1) Randomly initialize populations p

2) Determine fitness of population

3) Until convergence repeat:

   a) Select parents from population

   b) Crossover and generate new population

   c) Perform mutation on new population

   d) Calculate fitness for new population

Example problem and solution using Genetic Algorithms

Given a target string, the goal is to produce target string starting from a random string of the same length. In the following implementation, following analogies are made –

- Characters A-Z, a-z, 0-9 and other special symbols are considered as genes
- A string generated by these characters is considered as chromosome/solution/Individual

Fitness score is the number of characters which differ from characters in target string at a particular index. So individual having lower fitness value is given more preference.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Note: Every time algorithm starts with random strings, so output may differ

As we can see from the output, our algorithm sometimes stuck at a local optimum solution, this can be further improved by updating fitness score calculation algorithm or by tweaking mutation and crossover operators.

Why use Genetic Algorithms

- They are Robust
- Provide optimization over large space state.
- Unlike traditional AI, they do not break on slight change in input or presence of noise

Application of Genetic Algorithms

Genetic algorithms have many applications, some of them are –

- Recurrent Neural Network (RNN)
- Mutation testing
- Code breaking
- Filtering and signal processing
- Learning fuzzy rule base etc

3.3 Adversarial Search Techniques:

- In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.
- But there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.
- The environment with more than one agent is termed as multi-agent environment, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.
- So, Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.
- Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Just for Review *(We already discussed in Game Playing Topic)*

Types of Games in AI:

	Deterministic	Chance Moves
Perfect information	Chess, Checkers, go, Othello	Backgammon, monopoly
Imperfect information	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble, nuclear war

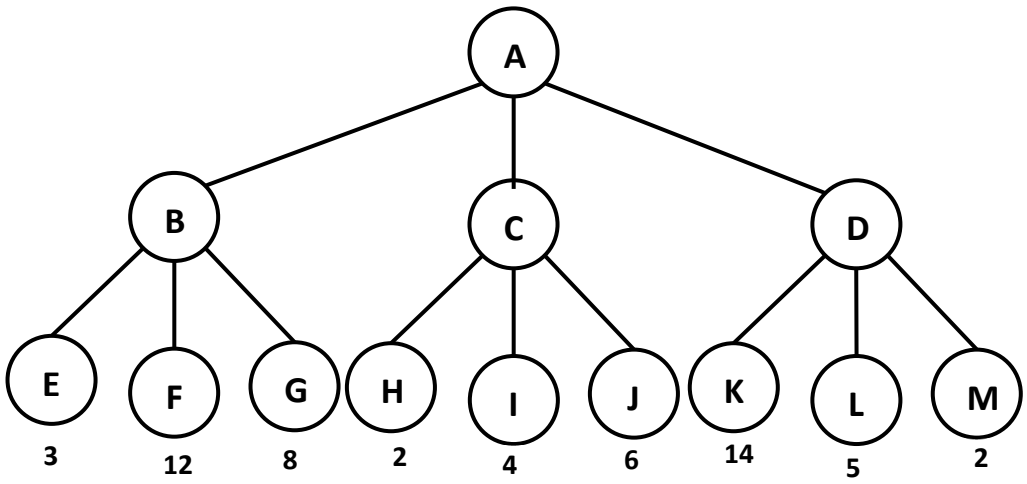
- **Perfect information:** A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.
- **Imperfect information:** If in a game agent do not have all information about the game and not aware with what's going on, such type of games is called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.
- **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

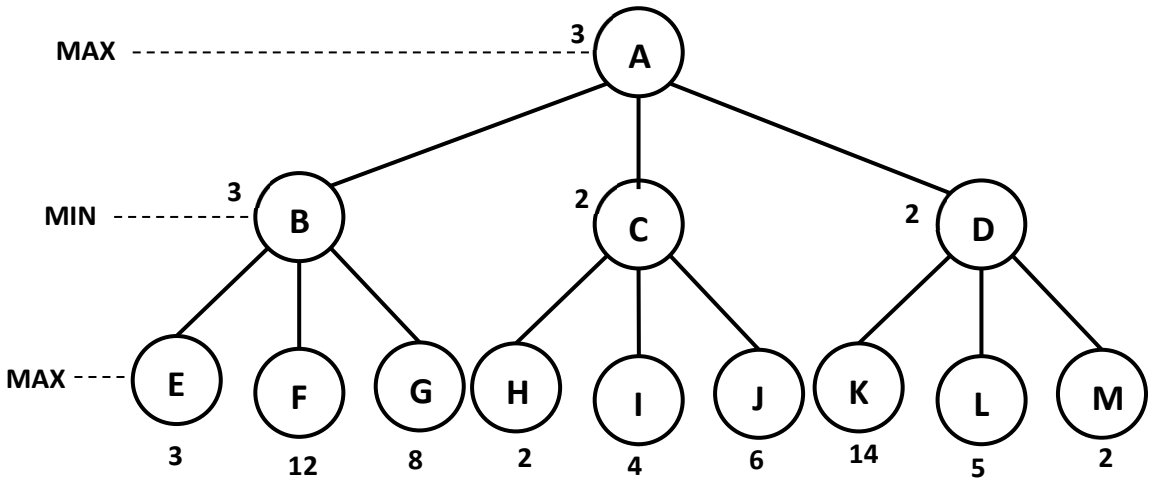
Min-Max Procedure

- Backtracking algorithm.
- Best move strategy used.
- MAX will try to maximize its utility (Best Move).
- MIN will try to minimize utility (Worst Move).
- Start with the current position as a MAX node.
- Expand the game tree a fixed number of ply (half-Moves).
- Apply the evolution function to the leaf position.
- Calculate back-up up values bottom-up.
- Pick the move which was chosen to give the MAX value at the root node.
- $\boldsymbol{MAX} = -\infty$ $\boldsymbol{MIN} = \infty$

Example:



Sol^n:



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$\boldsymbol{MIN} = \infty$

(∞, 3) = (3, 12) = (3, 8) = 3

(∞, 2) = (2, 4) = (2, 6) = 2

(∞, 14) = (14,5) = (5, 2) = 2

$\boldsymbol{MAX} = -\infty$

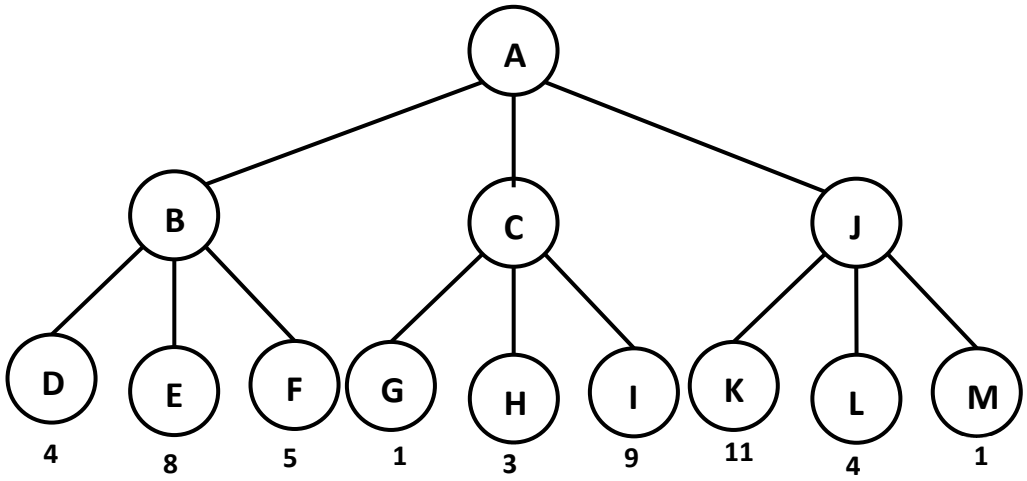
(−∞, 3) = (3, 2) = (3, 2) = 3

Alpha-Beta Procedure

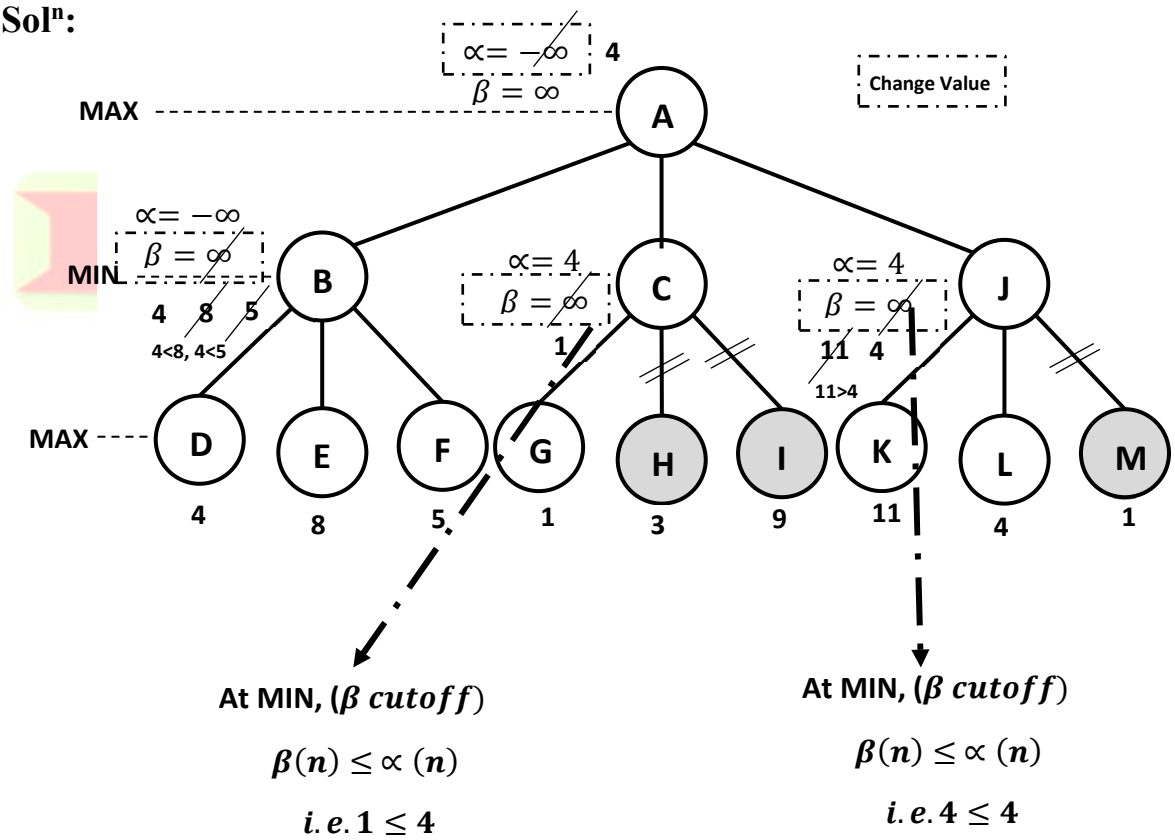
- A method that can often cutoff a half the game tree.
- Based on the idea that if a move is clearly bad, there is no need to follow the consequence of it.
- Alpha $(\propto)$ − Highest value we have found so far.
- Beta (β) − Lowest value we have found so far.
- Traverse the search tree in Depth-First order.
- **MAX cut off rule:** At a MAX node n, cut off search if $\propto(\boldsymbol{n}) \geq \boldsymbol{\beta}(\boldsymbol{n})$
- **MIN cut off rule:** At a MIN node n, cut off search if $\boldsymbol{\beta}(\boldsymbol{n}) \leq \propto(\boldsymbol{n})$
- Pruning does not affect final results.
- Entire subtree can be pruning.
- **Pruning:** Way of degerming that certain branch which will do not be useful.
- **Alpha Cut** $(\propto)$ − If current MAX value is greater than the successor MIN value, don't explore that MIN sub tree any more.
- **Beta Cut** (β) − If current MAX value is greater than the successor MIN value, don't explore that MIN sub tree any more.
- Alpha Beta Pruning is a method that reduces the number of nodes explored in Mini-Max strategy. It reduces the time required for the search.
- It is a depth-first search strategy. As root node is visited first then the sub tree.
- For the minimum nodes, the score computer start with **+ INFINITY** $+\infty$ and decreases with time.
- For the maximum nodes, the score computer start with **- INFINITY** $-\infty$ and increases with time.
-

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example:



Sol^n:



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Difference Between Uninformed and Informed Search

Un-Informed Search	Informed Search
1. Information is not available.	1. More Information is available.
2. Blind grouping is done.	2. Based on heuristic methods.
3. Search efficiency is low.	3. Searching is fast.
4. Impractical to solve very large problems.	4. Can handle large search problems
5. Best solution can be achieved.	5. Mostly a good enough solution is accepted as optimal solution.
6. Eg: DFS, BFS, IDS	6. Eg: Best First Search, A^*, Hill Climbing

Difference Between BFS and DFS Search

BFS	DFS
1. BFS stands for Breadth First Search	1. DFS stands for Depth First Search.
2. It uses the data structure queue.	2. It uses the data structure stack.
3. BFS traverse the tree level wise.	3. DFS traverse the tree depth wise.
4. BFS uses FIFO list.	4. DFS uses LIFO list.
5. BFS requires more memory.	5. DFS requires less memory
6. BFS is used to find shortest path.	6. DFS is used in cycle detection.
7. Always provide shallowest path solution.	7. Does not guarantee shallowest path Sol^n.
8. No backtracking is required in BFS.	8. Backtracking is required in DFS.
9. Never get trapped into finite loops.	9. Gets trapped into infinite loops.
10. Complete	10. Not Complete

THE END

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Unit 4 8 hrs.

Knowledge Representation, Inference and Reasoning

4.1 Approaches to Knowledge Representation
4.2 Issues in Knowledge Representation
4.3 Propositional Logic, Predicate Logic, FOPL
4.4 Rules of Inference, Resolution Refutation System (RRS), Answer Extraction from RRS
4.5 Statistical Reasoning:
Probability and Bayes Theorem and Causal Networks, Reasoning in Belief Network
4.6 Semantic Nets and Frames

Knowledge Representation:

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning. Hence, we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real-world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Basic Terminology

Object: All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.

Events: Events are the actions which occur in our world.

Performance: It describe behavior which involves knowledge about how to do things.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Meta-knowledge: It is knowledge about what we know.

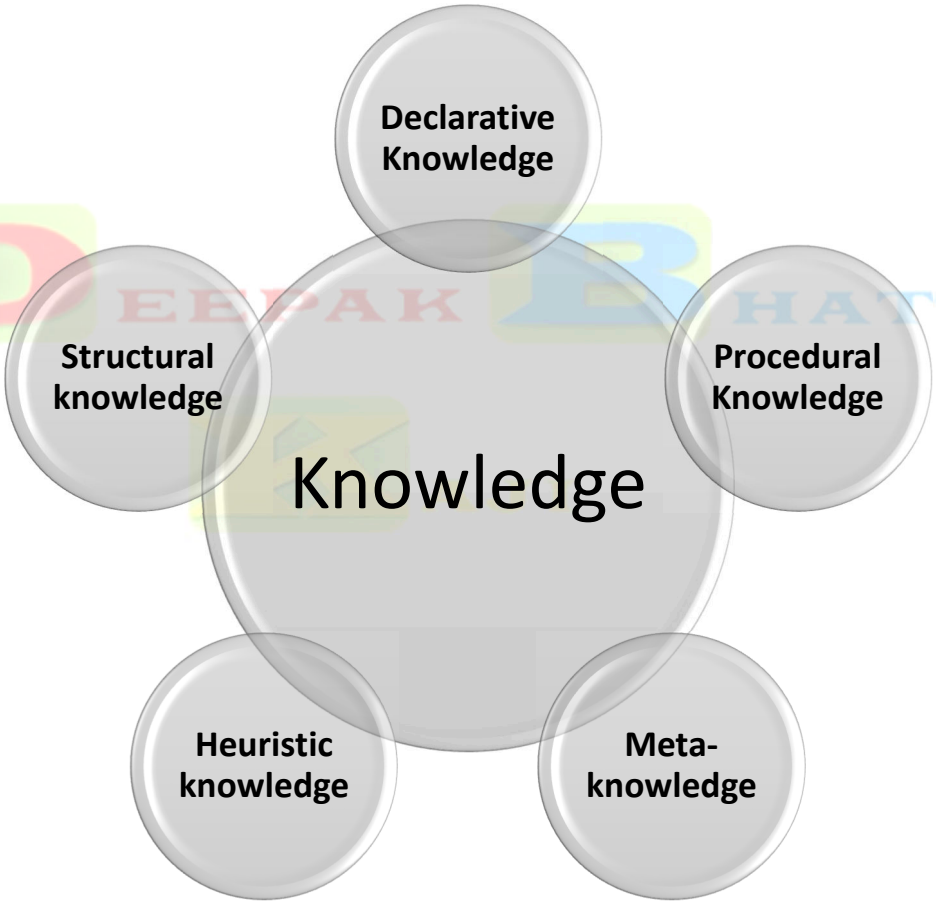
Facts: Facts are the truths about the real world and what we represent.

Knowledge-Base: The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Knowledge: Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

Types of knowledge

Following are the various types of knowledge:



1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarative sentences.
- It is simpler than procedural language.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

3. Meta-knowledge:

- Knowledge about the other types of knowledge is called Meta-knowledge.

4. Heuristic knowledge:

- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

5. Structural knowledge:

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4.1 Approaches to Knowledge Representation

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

1. Simple relational knowledge:

- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

Example: The following is the simple relational knowledge representation.

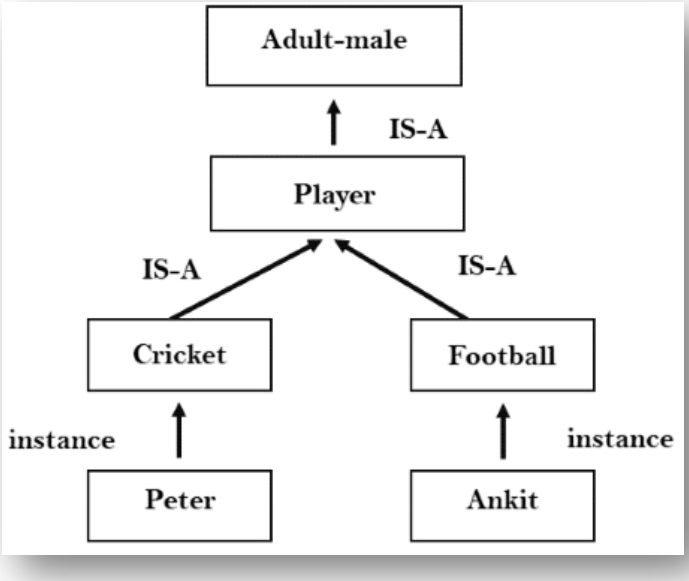
Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- **Example:**



3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
  a. **Marcus is a man**
  b. **All men are mortal**

  Then it can represent as;

  a. **man (Marcus)**
  b. **∀x = man (x) ----------> mortal (x)**

4. Procedural knowledge:

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Requirements for knowledge Representation system:

A good knowledge representation system must possess the following properties.

1. **Representational Accuracy:**

   KR system should have the ability to represent all kind of required knowledge.

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

4. **Acquisitional efficiency:**

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

DEEPAK BHATTA

4.2 Issues in Knowledge Representation

Issues that arises while Knowledge Representation techniques:

1. Important attributes.
2. Relationships among attributes.
3. Choosing the granularity of representation.
4. Representing sets of objects.
5. Finding the right structure as needed

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4.3 Propositional Logic, Predicate Logic, FOPL

In order to solve complex problems in AI, we need

- Large amount of knowledge.
- Mechanism to manipulate knowledge to create solutions to new problem.

Methods for knowledge representation:

1. Logical Representation
   a. Propositional Logic
   b. Predicate Logic or FOPL (First Order Predicate Logic)
2. Production Rule
3. Semantic Networks
4. Frame Representation

DEEPAK BHATTA KAJI

Propositional Logic

- ➔ Propositional is a statement of a fact.
- ➔ Propositional is a declarative sentence whose value is either **true** (denoted either T or 1) or **false** (denoted either F or 0).
- ➔ Variables are used to represent propositions.
- ➔ Propositional Symbols/Variables a, b, and c, or
- ➔ The most common variables used are p, q, and r.
- ➔ Examples:
  - *"Today is Sunday".* **[Atomic Proposition]**
  - *"Today is Sunday and it's sunny day.* **[Molecular/Complex Proposition]**
  - *"10+10=20"* **[Atomic Proposition]**
- ➔ Ram is an honest boy. ---- **Fact {Propositional** — T or F**}**
- ➔ Is Ram an honest boy? ---- {It fails to be a propositional}

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Some Terminologies in Propositional Algebra

- **Statement:** Sentence that can be true / false
- **Properties of Statements:**
  - **Satisfiability:** a sentence is satisfiable if there is an interpretation for which it is true. A sentence is true if it is true in some model
    - **E.g. *"Today is cold climate"***
  - **Contradiction:** if there is no interpretation for which sentence is true.
    - **E.g. *"Japan is capital of India"***
  - **Validity:** a sentence is valid if it is true for every interpretation. Valid sentences are also known as **Tautology.**
    - **E.g. *"Kathmandu is the capital of Nepal"***

Logical Operators

- **Unary Operator**
  - Negation: *"NOT p"*, $\neg p$, $\sim p$
- **Binary Operators**
  - Conjunction: "p AND q", $p \wedge q$
  - Disjunction: "p OR q", $p \vee q$
  - Implication: "IF p THEN q", $p \rightarrow q$
  - Biconditional: "p IF AND ONLY IF q", $p \leftrightarrow q$

Unary Operator: Negation

- Negation Operator, "NOT", has symbol $\neg$ ***or*** $\sim$
- **Example:**
  - p: Today is sunny day.
  - $\neg$**p** can be read as:
    - Today is not sunny day
    - Today is cold day.
    - Today is cloudy day.
    - Today is rainy day.
  - **Truth Table:**

p	$\neg p$
T	F
F	T

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Binary Operator: Conjunction

- Conjunction Operator, "AND", has symbol ∧
- **Example:**
  - p: Today is sunny day.
  - q: I am eating an Ice-cream.
  - $\boldsymbol{p} \wedge \boldsymbol{q}$ can be read as:
    - Today is sunny day, and I am eating Ice-cream.
  - **Truth Table:**

p	q	$\boldsymbol{p} \wedge \boldsymbol{q}$
T	T	T
T	F	F
F	T	F
F	F	F

Binary Operator: Disjunction

- Conjunction Operator, "OR", has symbol ∨
- **Example:**
  - p: Today is sunny day.
  - q: I am eating an Ice-cream.
  - $\boldsymbol{p} \vee \boldsymbol{q}$ can be read as:
    - Today is sunny day, or I am eating Ice-cream.
  - **Truth Table:**

p	q	$\boldsymbol{p} \vee \boldsymbol{q}$
T	T	T
T	F	T
F	T	T
F	F	F

Binary Operator: Implication

- Implication Operator, "IF…THEN…", has symbol →
- **Example:**
  - p: It is raining outside.
  - q: The road is wet.
  - $\boldsymbol{p} \rightarrow \boldsymbol{q}$ can be read as:
    - If it is raining outside, then the road is wet.
- For the compound statement $\boldsymbol{p} \rightarrow \boldsymbol{q}$
  - $\boldsymbol{p}$ is called premise, hypothesis or the antecedent.
  - $\boldsymbol{q}$ is called the conclusion or consequent.
  - $\boldsymbol{q} \rightarrow \boldsymbol{p}$ is the converse of $p \rightarrow q$
  - $\neg\boldsymbol{q} \rightarrow \neg\boldsymbol{p}$ is the contrapositive of $\boldsymbol{p} \rightarrow \boldsymbol{q}$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- **Truth Table:**

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Binary Operator: Biconditional

- Biconditional Operator, "IF and only IF", has symbol $\boldsymbol{p \leftrightarrow q}$
- **Example:**
  - p: It is raining outside.
  - q: The road is wet.
  - $\boldsymbol{p \leftrightarrow q}$ can be read as:
    - If it is raining outside if and only if the road is wet.
  - The biconditional statement is equivalent to $(\boldsymbol{p} \rightarrow \boldsymbol{q}) \wedge (\boldsymbol{q} \rightarrow \boldsymbol{p})$.
  - In other word, for $\boldsymbol{p \leftrightarrow q}$ to be true we must have both p and q true or both false.

DEEPAK BHATTA

- **Truth Table:**

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Using truth table, show that $p \leftrightarrow q$ is logically equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$

p	q	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Predicate Logic or FOPL

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

- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in an easier way and can also express the relationship between those objects.
- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
    - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ...
    - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
    - **Function:** Father of, best friend, third inning of, end of, ...
- As a natural language, first-order logic also has two main parts:
    - **Syntax**
    - **Semantics**

Syntax of First-Order logic:

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

Constant	1, 2, a, john, mumbai, cat, dog ....
Variables	X, Y, Z, A, B, X1, X2 ....
Predicates	brother, father ….
Function	sqrt, leftLegOf, ...
Connectives	∧, ∨, ¬, ⇒, ⇔
Equality	==
Quantifier	∀ *(for All)*, ∃ *(there Exist)*

Atomic sentences:

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

Example: Ram and Lakshman are brothers: => brothers(ram, lakshman).
munna is a cat: => cat (munna).

Complex Sentences:

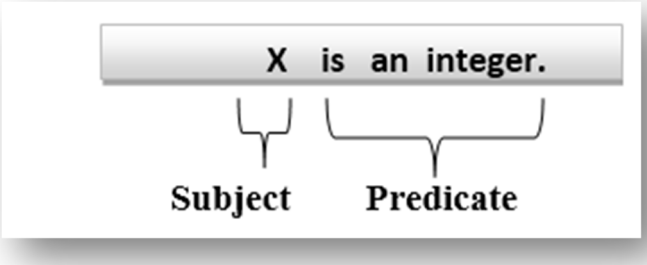
- Complex sentences are made by combining atomic sentences using connectives.

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

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

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
    1. (∀) **Universal Quantifier, (for all, everyone, everything)**
    2. (∃) **Existential quantifier, (for some, at least one).**

Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.

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

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

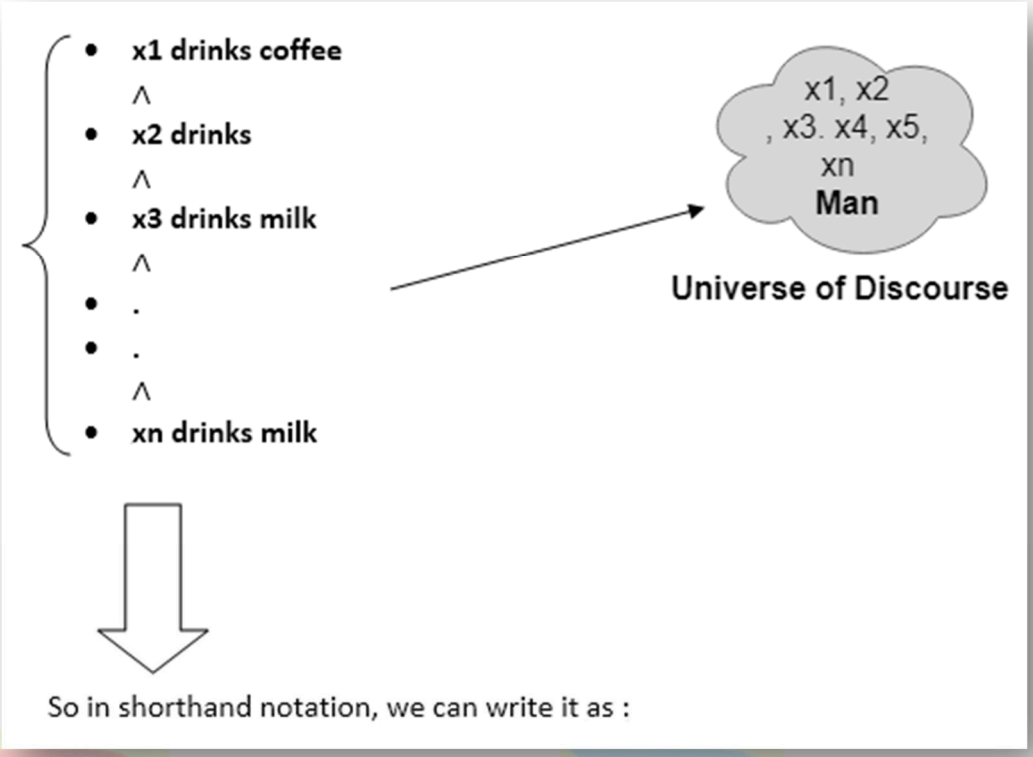
- **For all x**
- **For each x**
- **For every x.**

Example:

All man drink coffee.

Let a variable x which refers to a cat so all x can be represented in UOD as below:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



DEEPAK BHATTA

∀x man(x) → drink (x, coffee).

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

Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

Note: In Existential quantifier we always use AND or Conjunction symbol (∧).

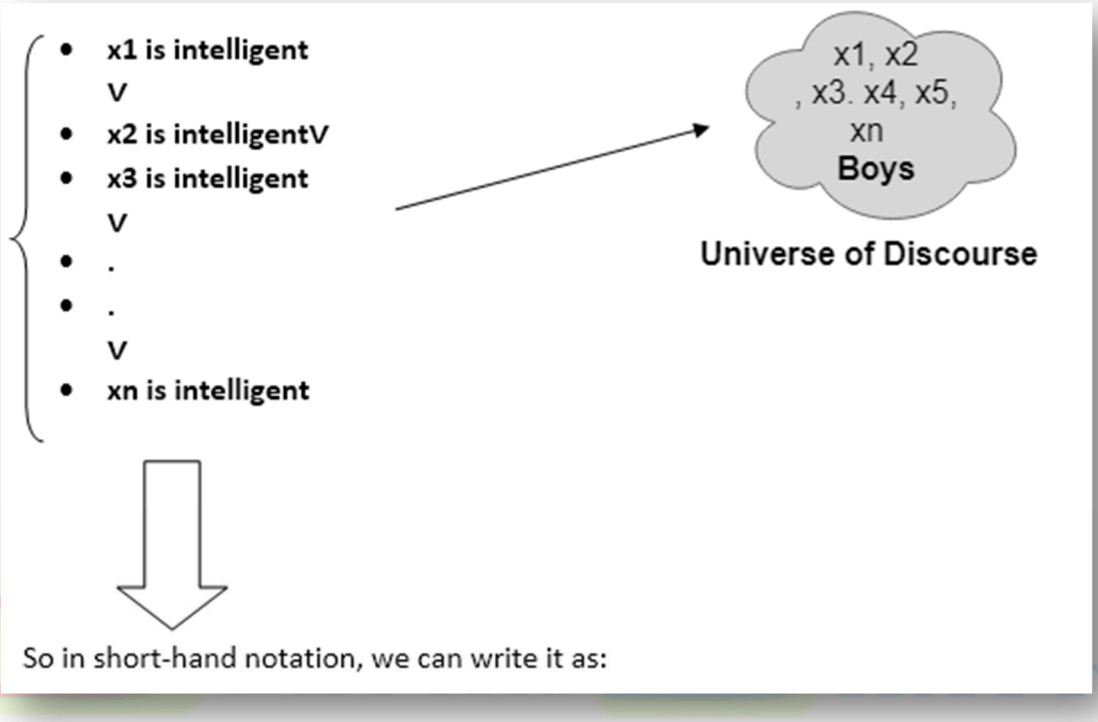
If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example:

Some boys are intelligent.



∃x: boys(x) ∧ intelligent(x)

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

Points to remember:

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

Properties of Quantifiers:

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

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Some Examples of FOL/FOPL using quantifier:

1. All birds fly.

In this question the predicate is "**fly(bird)**."

And since there are all birds who fly so it will be represented as follows.

∀x: bird(x) →fly(x).

2. Every man respects his parent.

In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.

Since there is every man so will use ∀, and it will be represented as follows:

∀x: man(x) → respects (x, parent).

3. Some boys play cricket.

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

∃x: boys(x) → play(x, cricket).

4. Not all students like both Mathematics and Science.

In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.

Since there are not all students, so we will use **∀ with negation, so** following representation for this:

¬∀ (x): [student(x) → like(x, Mathematics) ∧ like(x, Science)].

5. Only one student failed in Mathematics.

In this question, the predicate is "**failed(x, y)," where x= student, and y= subject**.

Since there is only one student who failed in Mathematics, so we will use following representation for this:

∃(x): [student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Rules of Inference

Idempotent rule: $p \wedge p \equiv p$ $p \vee p \equiv p$	**Commutative rule:** $p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$
Associative rule: $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$ $p \vee (p \vee r) \equiv (p \vee q) \vee r$	**Distributive rule:** $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (p \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
De-Morgan's rule: $\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	**Contrapositive rule:** $p \rightarrow q \equiv \neg q \rightarrow \neg p$
Implication Elimination: $p \rightarrow q \equiv \neg p \vee q$	**Bidirectional Implication Elimination:** $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
Double Negation rule: $\neg(\neg p) \equiv p$	**Absorption rule:** $p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$
Fundamental Identities: $p \wedge \neg p \equiv F\ [Contradiction]$ $p \vee \neg p \equiv T\ [Tautology]$ $p \vee T \equiv p$ $p \vee F \equiv p$ $p \wedge T \equiv p$ $p \wedge F \equiv F$	**Modus Ponens:** -If p is **true** and $p \rightarrow q$ then we infer q is also **true** p $p \rightarrow q$ Hence, q *E.g: "If Tweety is a bird then Tweety filies."* *"Tweety is a bird."* *Therefore, Tweety flies.* **Modus Ponens:** -If ¬p is **true** and $p \rightarrow q$ then we infer ¬q is also **true** $\neg p$ $p \rightarrow q$ Hence, ¬q *E.g: "If Tweety is a bird then Tweety filies."* *"Tweety doesn't fly."* *Therefore, Tweety is not a bird.*

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Resolution Refutation System (RRS)

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

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

Clause: Disjunction of literals (an atomic sentence) is called a clause. It is also known as a unit clause.

Conjunctive Normal Form: A sentence represented as a conjunction of clauses is said to be conjunctive normal form or CNF.

The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

DEEPAK BHATTA

Where $\mathbf{l_i}$ and $\mathbf{m_j}$ are complementary literals.

This rule is also called the **binary resolution rule** because it only resolves exactly two literals.

Example:

KAJI

We can resolve two clauses which are given below:

[Animal (g(x) V Loves (f(x), x)] **and** **[¬ Loves(a, b) V ¬Kills(a, b)]**

Where two complimentary literals are: **Loves (f(x), x) and ¬ Loves (a, b)**

These literals can be unified with unifier θ= [a/f(x), and b/x], and it will generate a resolvent clause:

[Animal (g(x) V ¬ Kills(f(x), x)].

Steps for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4. Draw resolution graph (unification).

To better understand all the above steps, we will take an example in which we will apply resolution.

Example:

a. John likes all kind of food.

b. Apple and vegetable are food

c. Anything anyone eats and not killed is food.

d. Anil eats peanuts and still alive

e. Harry eats everything that Anil eats.

Prove by resolution that: **John likes peanuts.**

Solution:

Step-1: Conversion of Facts into FOL

Step-2: Convert FOL statements into CNF

In the first step we will convert all the given statements into its first order logic and then convert it into CNF.

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

- **Eliminate all implication (→) and rewrite.**
- **Move negation (¬) inwards and rewrite.**
- **Rename variables or standardize variables.**
- **Eliminate existential instantiation quantifier by elimination.**
- **Drop Universal quantifiers.**

a. John likes all kind of food.

FOL: $\forall X: \boldsymbol{food}(X) \rightarrow \boldsymbol{likes}(\boldsymbol{john}, X)$

CNF: $\neg \boldsymbol{food}(X) \vee \boldsymbol{likes}(\boldsymbol{john}, X)$

b. Apple and vegetable are food

FOL: $\boldsymbol{food}(\boldsymbol{apple}) \wedge \boldsymbol{food}(\boldsymbol{vegetables})$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

CNF: $\boldsymbol{i)\ food(apple)}$ $\boldsymbol{ii)\ food(vegetables)}$

c. Anything anyone eats and not killed is food.

FOL: $\forall \boldsymbol{X}, \forall \boldsymbol{Y}: \boldsymbol{eats(X,Y)} \wedge \neg \boldsymbol{killed(X)} \rightarrow \boldsymbol{food(Y)}$

CNF: $\neg \boldsymbol{eats(X1,Y1)} \vee \boldsymbol{killed(X1)} \vee \boldsymbol{food(Y1)}$

d. Anil eats peanuts and still alive

FOL: $\boldsymbol{eats(anil,peanuts)} \wedge \boldsymbol{alive(anil)}$

CNF: $\boldsymbol{eats(anil,peanuts)} \wedge \boldsymbol{alive(anil)}$ **or**

$\boldsymbol{eats(anil,peanuts)} \wedge \neg \boldsymbol{killed(anil)}$

$\boldsymbol{i)\ eats(anil,peanuts)}$ $\boldsymbol{ii)\ \neg killed(anil)}$

e. Harry eats everything that Anil eats.

FOL: $\forall \boldsymbol{X}: \boldsymbol{eats(anil,X)} \rightarrow \boldsymbol{eats(harry,X)}$

CNF: $\neg \boldsymbol{eats(anil,X2)} \vee \boldsymbol{eats(harry,X2)}$

f. John likes Peanuts *(proven statement)*

FOL: $\boldsymbol{likes(john,peanuts)}$

CNF: $\boldsymbol{likes(john,peanuts)}$

Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

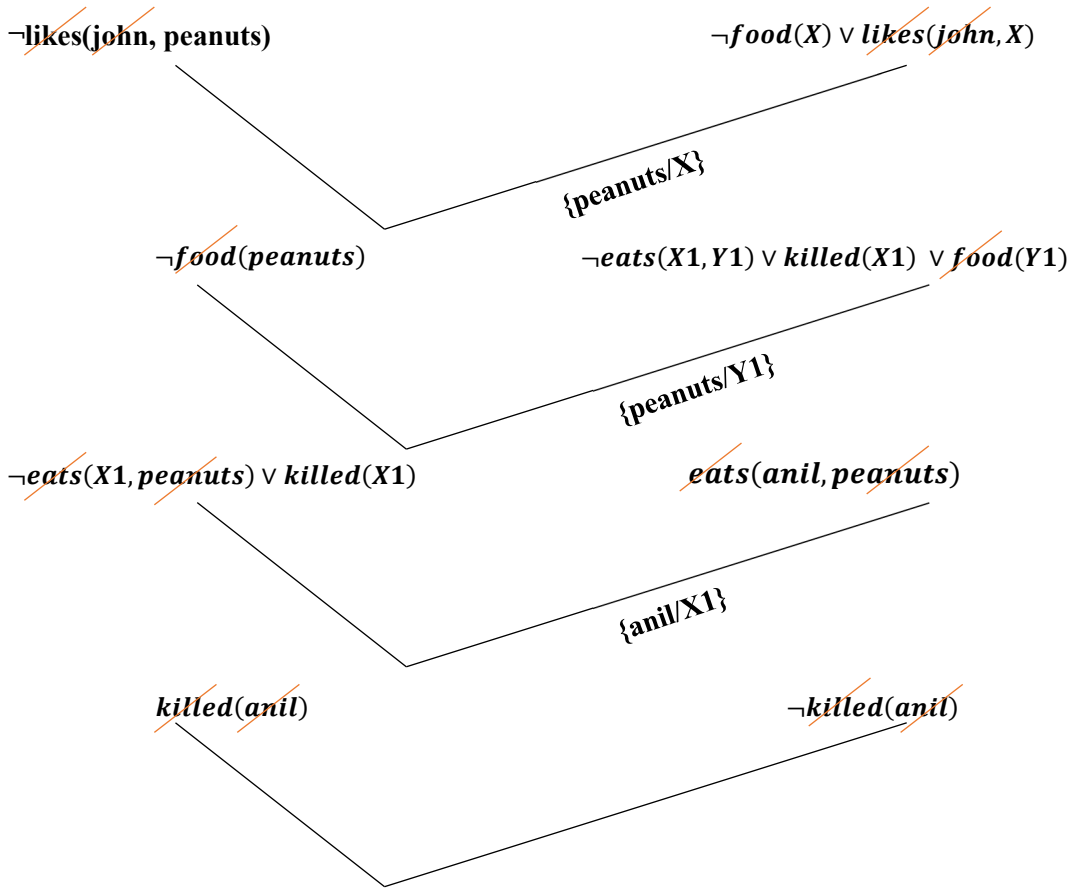
Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as: **¬likes(john, peanuts)**

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Conclusion: - $\neg likes(john, peanuts)$ is a false.

Therefore, $likes(john, peanuts)$ must be True.

Hence, John likes Peanuts. Proved.

Answer Extraction from RRS

Problem: How to extract answers from the given facts.

Assume the following facts:

i. Kabita like easy courses.
ii. Science Courses are hard.
iii. All the course in the Computer Engineering Department are easy.
iv. Com101 is a Computer Engineering Course.

Use resolution to answer the question "What course would Kabita like?"

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Sol^n:

i. Kabita like easy courses.

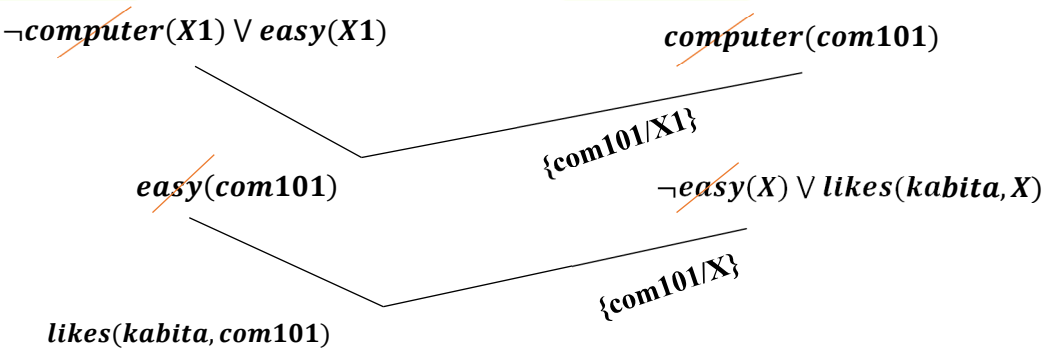
FOL: $\forall X: easy(X) \rightarrow likes(kabita, X)$
CNF: $\neg easy(X) \vee likes(kabita, X)$

ii. Science Courses are hard.
FOL: $hard(science) \equiv \neg easy(science)$
CNF: $\neg easy(science)$

iii. All the course in the Computer Engineering Department are easy.
FOL: $\forall X: computer(X) \rightarrow easy(X)$
CNF: $\neg computer(X1) \vee easy(X1)$ *//use different variable names*

iv. Com101 is a Computer Engineering Course.
FOL: $computer(com101)$
CNF: $computer(com101)$

Draw Resolution graph:



Likes(kabita,com101)

Thus, Kabita likes Com101

Using the resolution we can extract the answer of question.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4.5 Statistical Reasoning:

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So, to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- **Bayes' rule**
- **Bayesian Statistics**

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Probability and Bays Theorem

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

- ✓ $0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
- ✓ $P(A) = 0$, indicates total uncertainty in an event A.
- ✓ $P(A) = 1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

- $P(\neg A)$ = probability of a not happening event.
- $P(\neg A) + P(A) = 1$.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional probability:

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

Where $P(A \wedge B)$= Joint probability of a and B

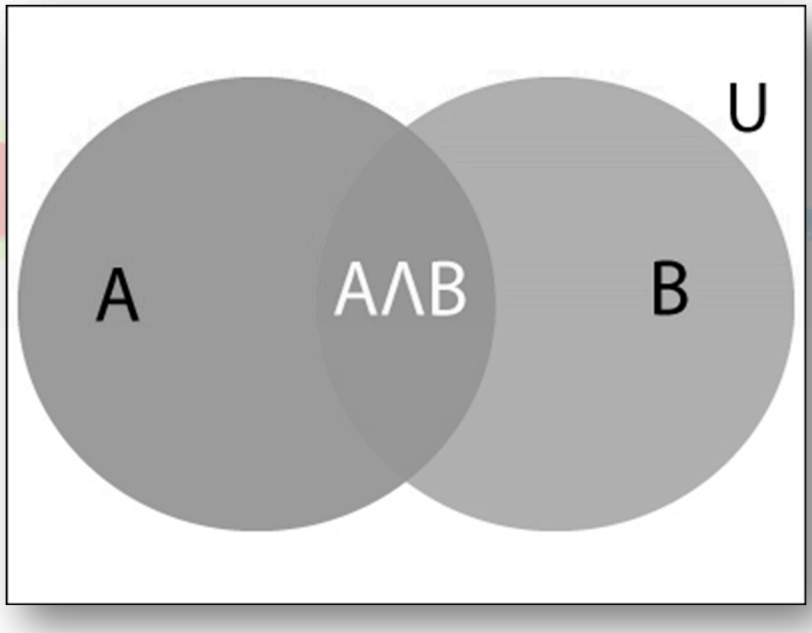
P(B)= Marginal probability of B.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A∧*B*) by P(B)**.



Example:

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

Solution:

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

Bayes' theorem:

Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

Bayes' theorem was named after the British mathematician **Thomas Bayes**. The **Bayesian inference** is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

Example: If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:

As from product rule we can write:

$P(A \wedge B) = P(A|B)\, P(B)$ or

Similarly, the probability of event B with known event A:

$P(A \wedge B) = P(B|A)\, P(A)$

Equating right hand side of both the equations, we will get:

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \quad \text{....(a)}$$

The above equation (a) is called as **Bayes' rule** or **Bayes' theorem**. This equation is basic of most modern AI systems for **probabilistic inference**.

It shows the simple relationship between joint and conditional probabilities. Here,

P(A|B) is known as **posterior**, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

P(B|A) is called the **likelihood**, in which we consider that hypothesis is true, then we calculate the probability of evidence.

P(A) is called the **prior probability**, probability of hypothesis before considering the evidence

P(B) is called **marginal probability**, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i)*P(B|A_i)}{\sum_{i=1}^{k} P(A_i)*P(B|A_i)}$$

Where A_1, A_2, A_3,....., A_n is a set of mutually exclusive and exhaustive events.

Applying Bayes' rule:

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(*B*), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect}|\text{cause})\,P(\text{cause})}{P(\text{effect})}$$

Application of Bayes' theorem in Artificial intelligence:

Following are some applications of Bayes' theorem:

- It is used to calculate the next step of the robot when the already executed step is given.
- Bayes' theorem is helpful in weather forecasting.
- It can solve the Monty Hall problem.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example of Bayes' theorem

Q) An insurance company insured 2000 scooter driver, 4000 car driver, and 6000 truck drivers. The probability of an accident involving a scooter, a car and a truck are 0.01, 0.03, and 0.15 respectively. One of the insured person meets with an accident. What is the probability that he is a scooter driver?

Sol^n:

Let, E1, E2, and E3 be the events of a driver being a scooter driver, car driver, and truck driver respectively.

Let A be the event that the person meets with an accident.

$$P(E1) = \frac{2000}{1200} \quad = \frac{1}{6} \quad = 0.16$$

$$P(E2) = \frac{4000}{1200} \quad = \frac{1}{3} \quad = 0.3$$

$$P(E3) = \frac{6000}{1200} \quad = \frac{1}{2} \quad = 0.5$$

Also, we have,

$P(A|E1) = 0.01$

$P(A|E2) = 0.03$

$P(A|E3) = 0.15$

Now, the probability that the insured person who meets with an accident is a scooter driver is $P(E1|A)$

Using Baye's Theorem;

$$\boldsymbol{P(E1|A) = \frac{P(E1) * P(A|E1)}{P(E1) * P(A|E1) + P(E2) * P(A|E2) + P(E3) * P(A|E3)}}$$

$$\boldsymbol{= \frac{0.16 * 0.01}{0.16 * 0.01 + 0.3 * 0.03 + 0.5 * 0.15}}$$

$$\boldsymbol{= \frac{0.0016}{0.0016 + 0.009 + 0.075}}$$

$$\boldsymbol{= \frac{0.0016}{0.0856}}$$

$$\boldsymbol{= 0.018}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Causal Network

Belief networks have often been called **causal networks** and have been claimed to be a good representation of causality. Recall that a causal model predicts the result of interventions.

A **belief network**, also called a **Bayesian network**, is an acyclic directed graph (DAG), where the nodes are random variables. There is an arc from each element of *parents*(X_i) into X_i. Associated with the belief network is a set of conditional probability distributions - the conditional probability of each variable given its parents (which includes the prior probabilities of those variables with no parents).

Thus, a belief network consists of

- a DAG, where each node is labeled by a random variable;
- a domain for each random variable; and
- a set of conditional probability distributions giving $P(X|parents(X))$ for each variable X.
- set of interconnected nodes, where each node represents a variable in the dependency model and the connecting arcs represent the causal relationship between these variables.

A belief network is acyclic by construction. The way the chain rule decomposes the conjunction gives the ordering. A variable can have only predecessors as parents. Different decompositions can result in different belief networks.

Reasoning in Belief Network

- A Belief Network is also called a Bayesian Network.
- A Bayesian Belief Network is a graphical representation of a probabilistic dependency model.
- Bayesian networks have been the most important contribution to the field of AI in the last 10 years.
- Provide a way to represent knowledge in an uncertain domain and a way to reason about this knowledge.
- Many applications: medicine, factories, help desks, spam filtering, etc.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- A Bayes net (also called a belief network) is an augmented directed acyclic graph, represented by the pair V , E

where:

– **V** is a set of vertices.

– **E** is a set of directed edges joining vertices. No loops of any length are allowed.

- Each vertex in V contains the following information:

– The name of a random variable

– A probability distribution table indicating how the probability of this variable's values depends on all possible combinations of parental values.

Example:

You have a new burglar alarm installed at home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and sometimes misses the alarm altogether. Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

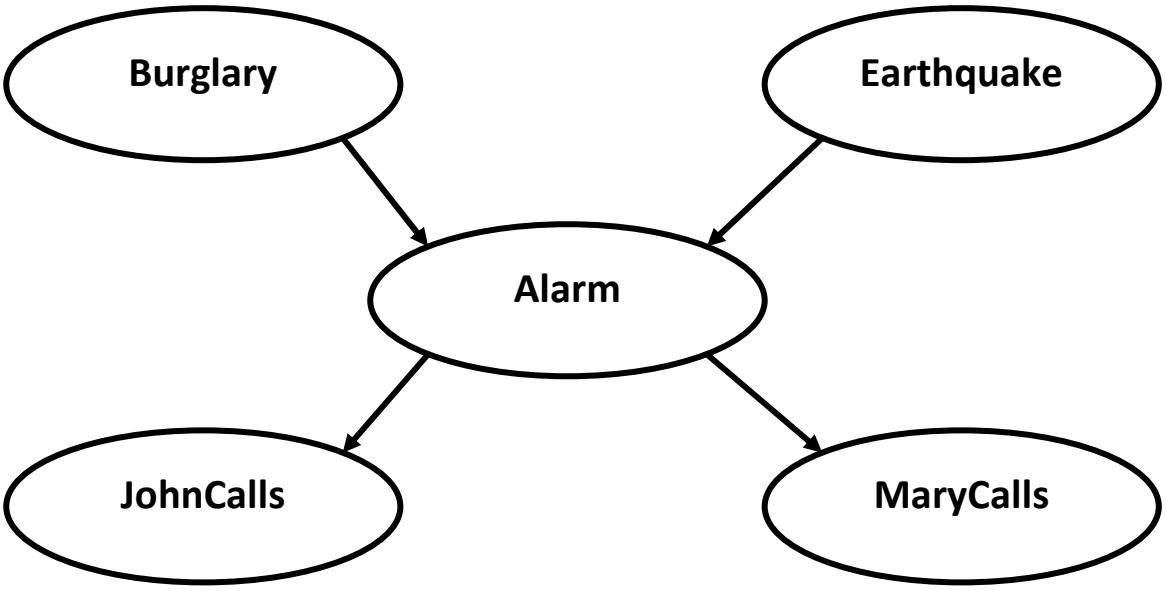


Fig.: A Typical Belief Network

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

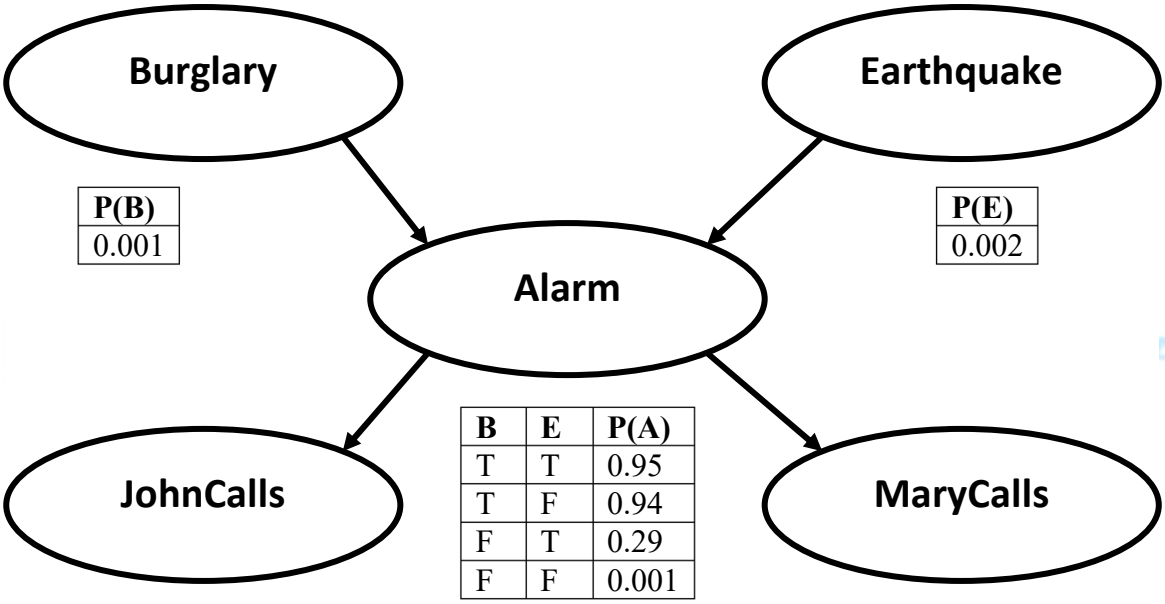
A typical belief network with conditional probabilities. The letters **B**, **E**, **A**, **J**, and **M** stands for **Burglary**, **Earthquake**, **Alarm**, **JohnCalls**, and **MarryCalls**, respectively. All variables (nodes) are Boolean, so the probability of, say **~P(A)** in any row of its table is **1 - P(A)**

P(JohnCalls, MaryCalls, Alarm, Burglary, Earthquake)

= P(JohnCalls | Alarm) P(MaryCalls | Alarm) P(Alarm | Burglary,Earthquake) P(Burglary) P(Earthquake)

OR

P(J, M, A, B, E) = P(J | A) P(M | A) P(A | B, E) P(B) P(E)



P(B)
0.001

P(E)
0.002

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

A	P(J)
T	0.90
F	0.05

A	P(M)
T	0.70
F	0.01

Q) Calculate the probability of the event that the alarm has sounded but neither a burglary nor an earthquake has occurred and John and Mary call.

Sol^n: $P(J \wedge M \wedge A \neg B \wedge \neg E) = ?$

$$= P(J|A) * P(M|A) * P(A|\neg B, \neg E) * P(\neg B) * P(\neg E)$$

$$= 0.90 * 0.70 * 0.001 * (1 - 0.001) * (1 - 0.002)$$

$$= 0.90 * 0.70 * 0.001 * 0.999 * 0.998$$

$$= \mathbf{0.00062}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4.6 Semantic Nets and Frames

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

This representation consists of mainly two types of relations:

i. IS-A relation (Inheritance)
ii. Kind-of-relation

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

Statements:

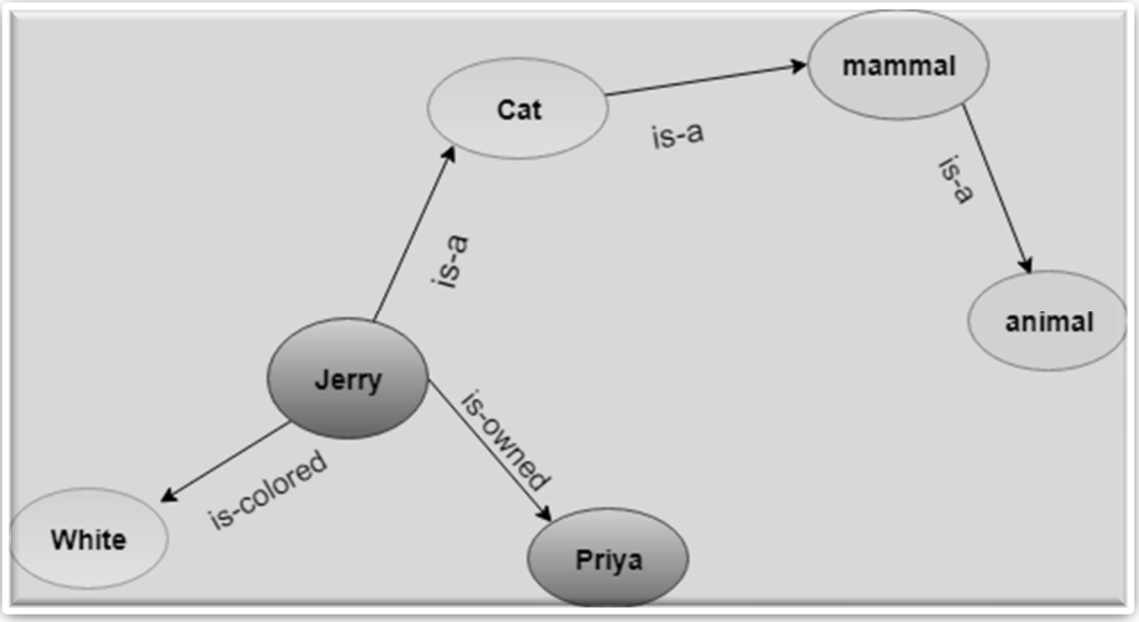
a. Jerry is a cat.

b. Jerry is a mammal

c. Jerry is owned by Priya.

d. Jerry is brown colored.

e. All Mammals are animal.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst-case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, **e.g., for all, for some, none, etc.**
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

DEEPAK BHATTA

3. Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

Facets: The various aspects of a slot are known as **Facets**. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of 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.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example: 1

Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Example 2:

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So, following is the frame representation for this:

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Advantages of frame representation:

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

Disadvantages of frame representation:

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much-generalized approach.

DEEPAK BHATTA KAJI

THE END

Unit 5 — 12 hrs.

Machine Learning

5.1 Concept of Learning

5.2 Learning by Analogy, Inductive Learning, Explanation based Learning

5.3 Supervised Learning (Classification/Regression):
K-Nearest Neighbors (KNN), Naïve Bays, Logistic Regression, Support Vector Machine (SVM), Neural Networks

5.4 Unsupervised Learning:

5.4.1 Clustering, K-Means

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.4.2 Dimensionality Reduction [Principal Component Analysis (PCA), Linear Discriminant Analysis]

5.5 Reinforcement Learning

5.6 Fuzzy Learning

5.7 Boltzmann Machines

5.8 Deep Learning

5.1 Concept of Learning

- The *art* and *science* of:
  - Giving computers the ability to learn to make decisions from data
  - … without being explicitly programmed!
- Examples:
  - Learning to predict whether an email is spam or not
  - Clustering Wikipedia entries into different categories
- **Supervised learning:** Uses labeled data
- **Unsupervised learning:** Uses unlabeled data

Simple Definition I - Branch of Artificial Intelligence that gives computers to learn without being explicitly programmed.

Simple Definition II - Branch of Artificial Intelligence, about to construct a system that learn from data.

Actual Definition - A computer program is said to learn from **Experience** E with respect to some **Task** T and some **Performance** measure P, if its performance on T, as measured by P, improves with experience E.

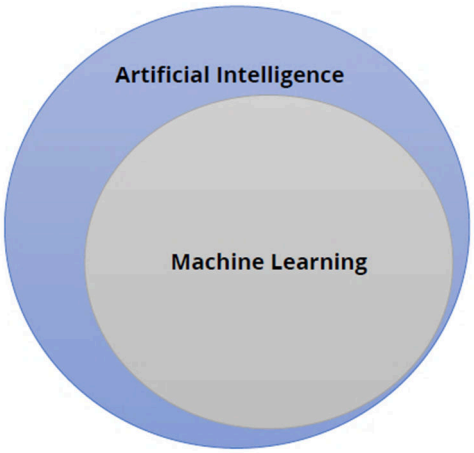
Some more examples

1. Classification of e-mail into **SPAM** or **NON-SPAM**
2. Classification of transaction as **FRAUD** or **GENUINE**
3. Patients diagnosed as **DIABETIC** or **NON-DIABETIC**
4. Gene Classification into **CODING** or **NON-CODING**.

…… Many more.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Machine Learning is an approach or subset of Artificial Intelligence that is based on the idea that machines can be given access to data along with the ability to learn from it.



The capability of Artificial Intelligence systems to learn by extracting patterns from data is known as Machine Learning.

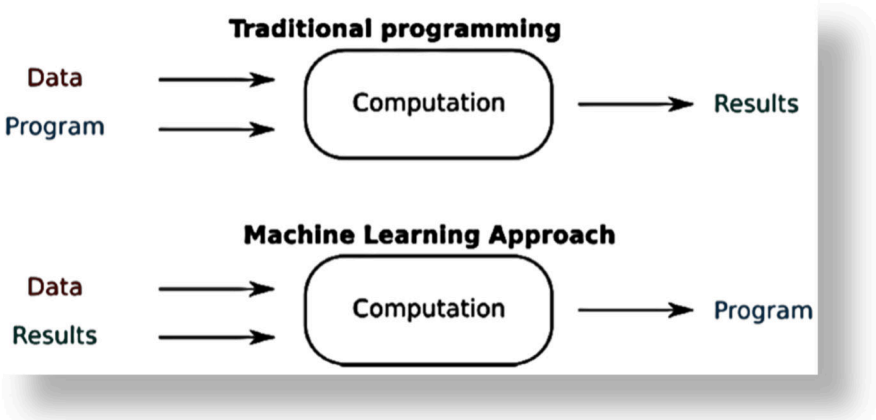
Machine learning is the science of getting computers to act without being explicitly programmed. Many researchers also think it is the best way to make progress towards human-level AI.

Machine learning is the category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed.

Machine learning is a sub set of Artificial Intelligence in the field of computer science that often uses statistical technique to give computers the ability to "learn" with data, without being explicitly programmed.

The name Machine learning was coined in 1959 by **-Arthur Samuel.**

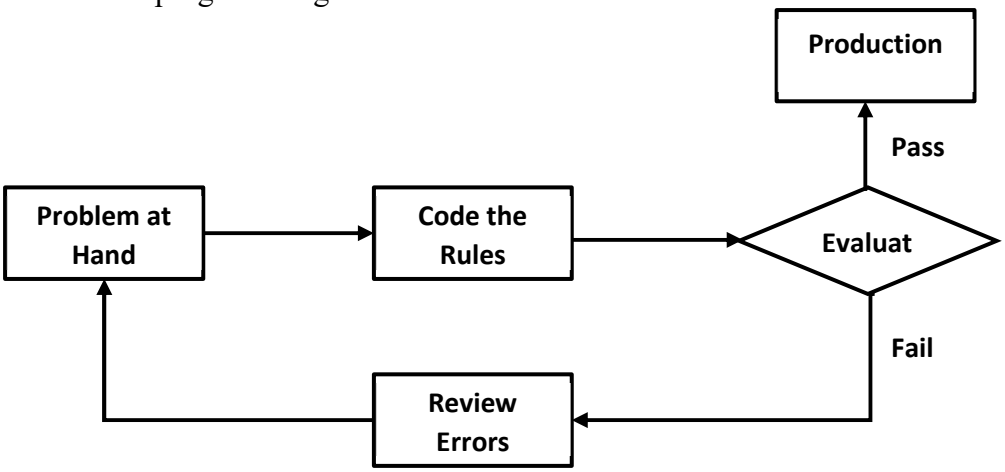
Traditional Programming vs. Machine Learning Approach



Traditional Approach

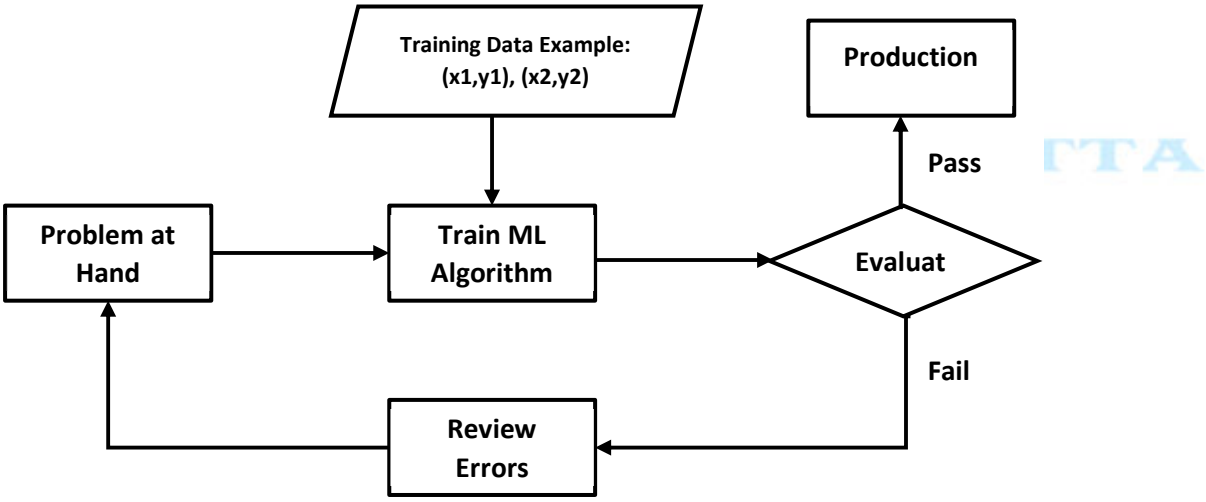
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Traditional programming relies on hard-coded rules.



Machine Learning Approach

Machine Learning relies on learning patterns based on sample data.



Relationship between Data Science and Machine Learning

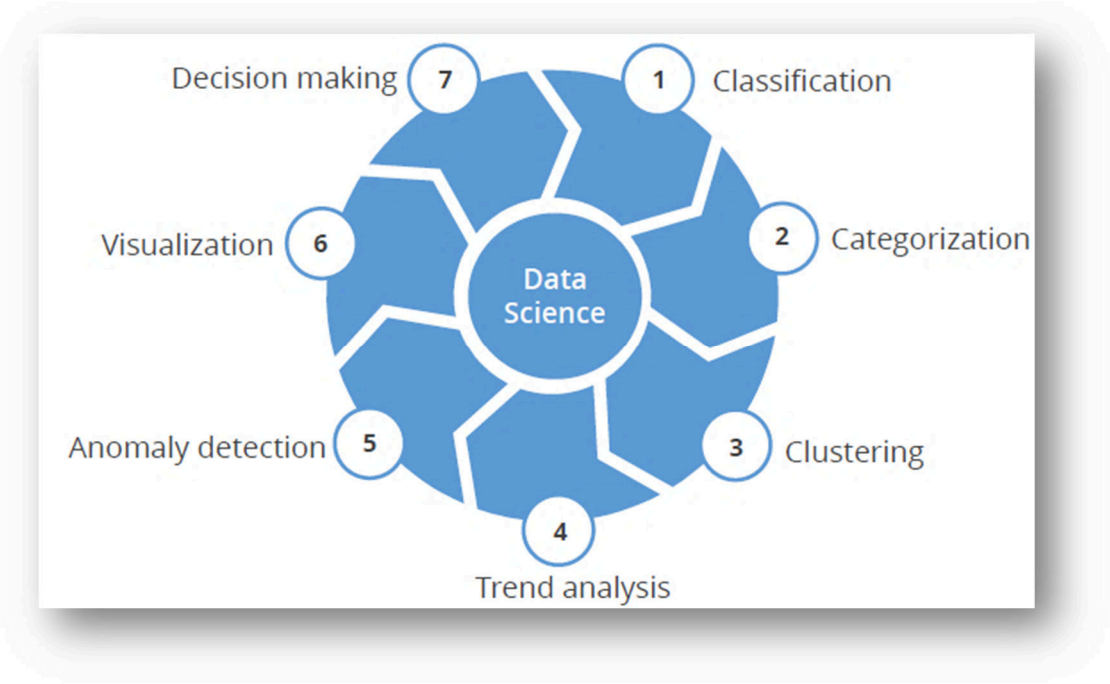
Data Science and Machine Learning go hand in hand. Data Science helps evaluate data for Machine Learning algorithms.

- Data science is the use of statistical methods to find patterns in the data.
- Statistical machine learning uses the same math and techniques as data science.
- These techniques are integrated into algorithms that learn and improve on their own.
- Machine Learning facilitates Artificial Intelligence as it enables machines to learn from the patterns in data.

Machine Learning Techniques

Machine Learning uses a number of theories and techniques from Data Science:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Machine Learning Algorithms

Machine Learning can learn from **labelled data** (known as supervised learning) or unlabeled data (known as unsupervised learning).

Machine Learning algorithms involving **unlabeled data**, or unsupervised learning, are more complicated than those with the labelled data or supervised learning.

Machine Learning algorithms can be used to make decisions in subjective areas as well.

Examples:

- Logistic Regression can be used to predict which party will win at the ballots.
- Naïve Bayes algorithm can separate valid emails from spam.

Applications of Machine Learning

Applications of Machine Learning

Artificial intelligence and Machine learning are being increasingly used in various functions such as:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

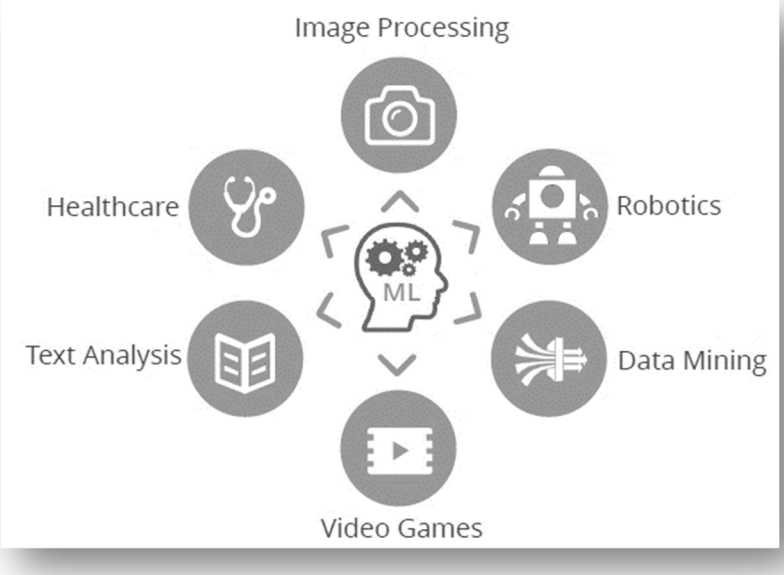
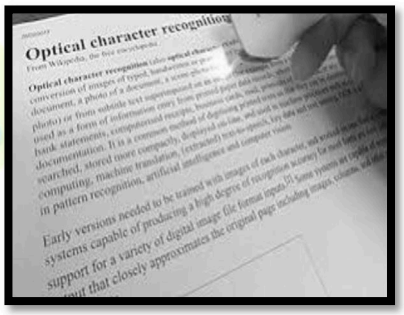
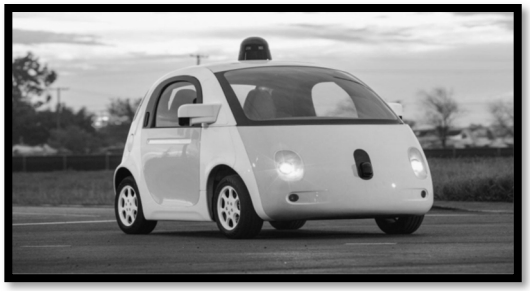


Image tagging and recognition

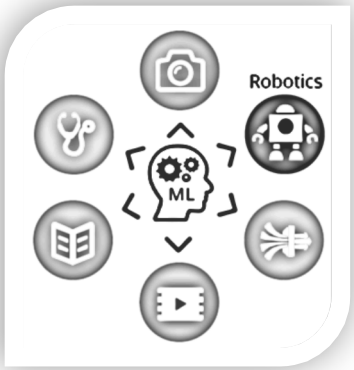


Optical Character Recognition (OCR)



Self-driving cars

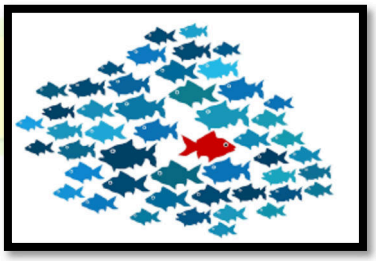
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



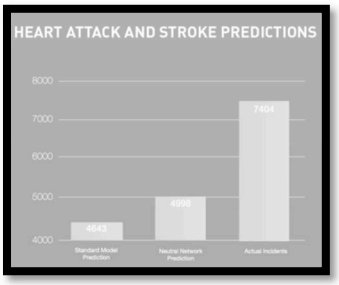
Human simulation Robert

Industrial robotics

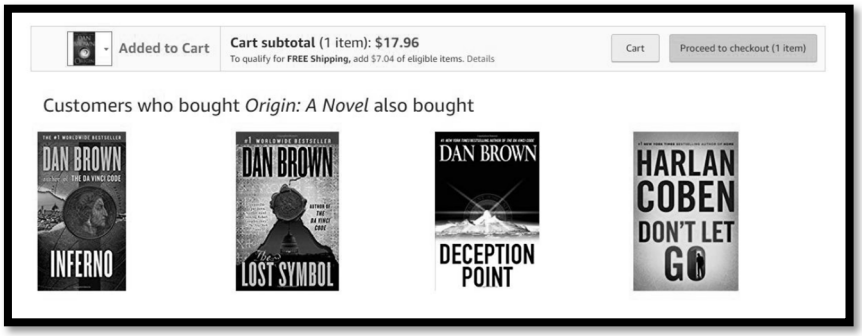
DEEPAK BHATTA



Anomaly detection



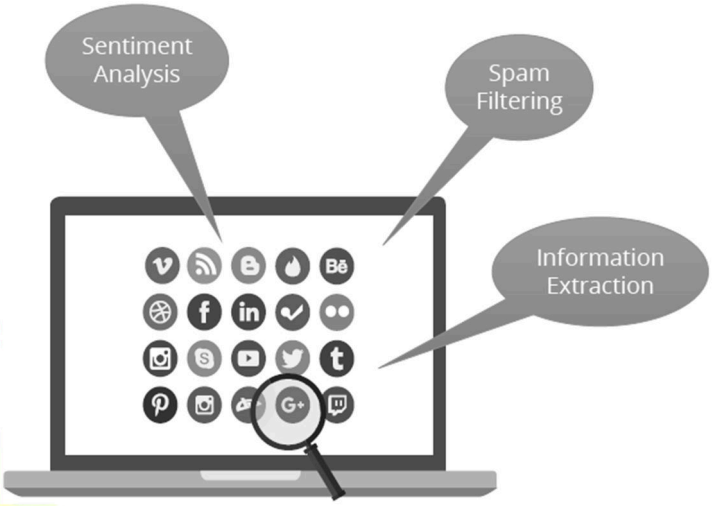
Grouping and Predictions

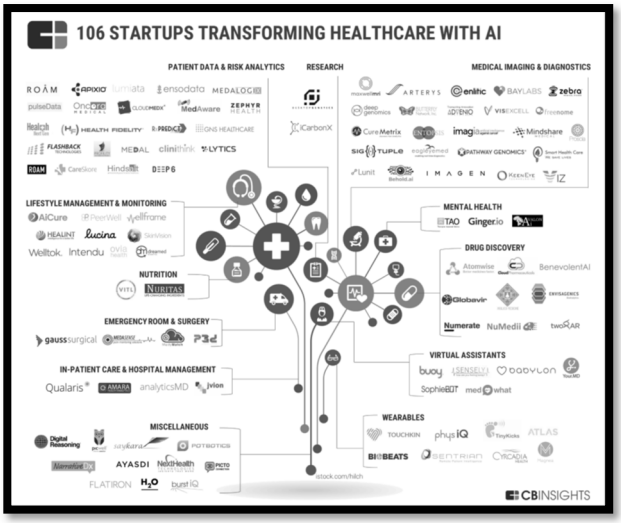


Association rules

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Some games implement reinforcement learning





© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.2 Learning by Analogy, Inductive Learning, Explanation based Learning

Learning:

Learning is one of those everyday terms which is broadly and vaguely used in English language.

- Learning is making useful changes in our minds.
- Learning is constructing or modifying representations of what is being experienced.
- Learning is the phenomenon of knowledge acquisition in the absence of explicit programming.

Herbert Simon, 1983

Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the system to do the same task or tasks drawn from the same population more efficiently and more effectively next time.

Learning involves the recognition of patterns in data or experience and then using that information to improve performance on another task.

Learning involves 3 factors:

Changes: Learning changes the learner:

For machine learning the problem is determining the nature of these changes and how to best represent them.

Generalization: Learning leads to generalization:

Performance must improve not only on the same task but on similar tasks.

Improvement: Learning leads to improvement:

Machine learning must address the possibility that changes may degrade performance and find ways to prevent it.

Learning Methods:

There are two different kinds of information processing which must be considered in a machine learning system.

a. Inductive Learning

b. Deductive Learning

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Inductive Learning:

Inductive learning is concerned with determining general patterns, organizational schemes, rules, and laws from raw data, experience or examples.

Deductive Learning:

Deductive learning is concerned with determining specific facts using general rules or that determination of new general rules from old general rules.

Learning by Analogy:

- Analogical reasoning is a powerful mechanism for exploiting past experience in planning and problem solving.
- An analogical transformation process is developed to extract knowledge from past successful problem-solving situations that bear a strong similarity to the current problem.
- Learning by analogy generally involves abstracting details from a particular set of problems and resolving structural similarities between previously distinct problems.
- Analogical learning refers to this process of recognition and then applying the solution from the known problem to the new problem.
- Such a technique is often identified as case-based reasoning (CBR).
- Analogical learning generally involves developing a set of mappings between features of two instances.

Inductive Learning:

Inductive Learning Algorithm (ILA) is an iterative and inductive machine learning algorithm which is used for generating a set of a classification rule, which produces rules of the form "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

This involves the process of learning by example -- where a system tries to induce a general rule from a set of observed instances.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Explanation Based Learning:

Explanation-Based Learning (EBL) is a principled method for exploiting available domain knowledge to improve supervised learning. Improvement can be in speed of learning, confidence of learning, accuracy of the learned concept, or a combination of these. In modern EBL the domain theory represents an expert's approximate knowledge of complex systematic world behavior. It may be imperfect and incomplete. Inference over the domain knowledge provides analytic evidence that compliments the empirical evidence of the training data. By contrast, in original EBL the domain theory is required to be much stronger; inferred properties are guaranteed. Another important aspect of modern EBL is the interaction between domain knowledge and labeled training examples afforded by explanations.

An EBL accepts 4 kinds of input:

A training examples
-- what the learning sees in the world.

A goal concepts
-- a high level description of what the program is supposed to learn.

An operational criterion
-- a description of which concepts are usable.

A domain theory
-- a set of rules that describe relationships between objects and actions in a domain.

From this EBL computes a generalization of the training example that is sufficient not only to describe the goal concept but also satisfies the operational criterion.

This has two steps:

Explanation
-- the domain theory is used to prune away all unimportant aspects of the training example with respect to the goal concept.

Generalization
-- the explanation is generalized as far possible while still describing the goal concept.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.3 Supervised Learning (Classification/Regression)

What is Supervised Learning?

Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It allows us to predict output for future or unseen data.

In Supervised learning, you train the machine using data which is well "labeled." It means some data is already tagged with the correct answer. It can be compared to learning which takes place in the presence of a supervisor or a teacher.

A supervised learning algorithm learns from labeled training data, helps you to predict outcomes for unforeseen data. Successfully building, scaling, and deploying accurate supervised machine learning Data science model takes time and technical expertise from a team of highly skilled data scientists. Moreover, Data scientist must rebuild models to make sure the insights given remains true until its data changes.

Why Supervised Learning?

- Supervised learning allows you to collect data or produce a data output from the previous experience.
- Helps you to optimize performance criteria using experience.
- Supervised machine learning helps you to solve various types of real-world computation problems.

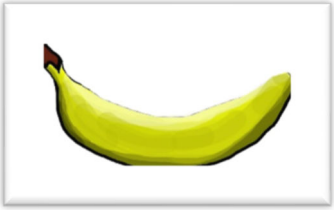
Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically, supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data.

For instance, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- If shape of object is rounded and depression at top having color Red then it will be labelled as –**Apple**.
- If shape of object is long curving cylinder having color Green-Yellow then it will be labelled as –**Banana**.

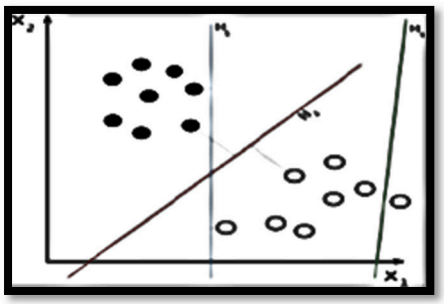
Now suppose after training the data, you have given a new separate fruit say Banana from basket and asked to identify it.



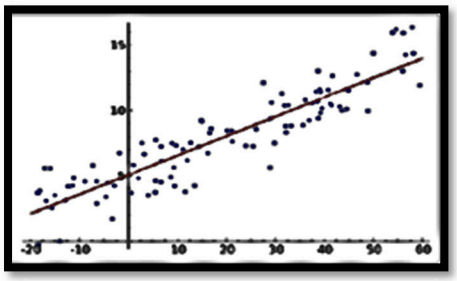
Since the machine has already learned the things from previous data and this time have to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in Banana category. Thus, the machine learns the things from training data (basket containing fruits) and then apply the knowledge to test data (new fruit).

Supervised learning classified into two categories of algorithms:

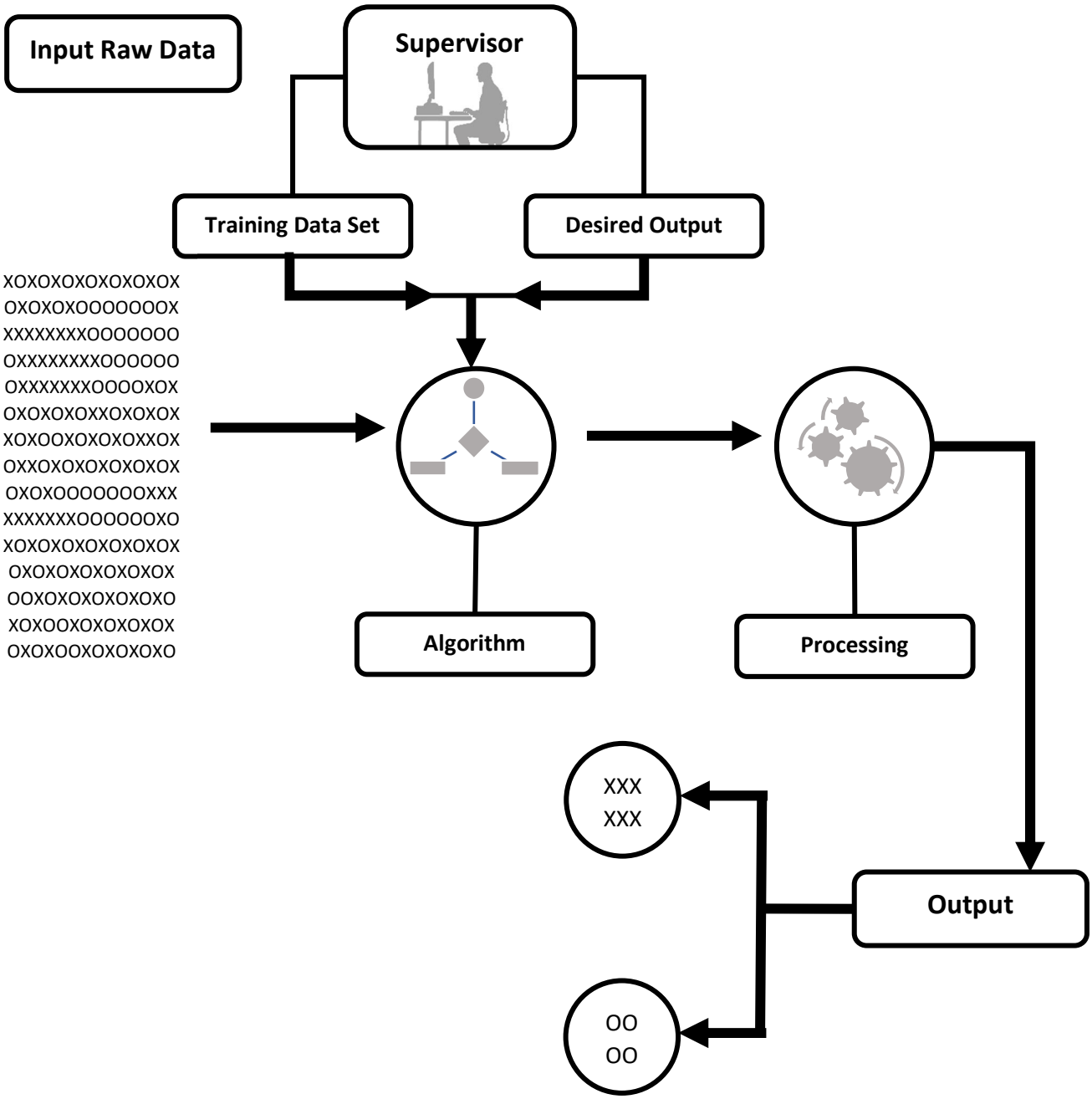
- **Classification**: A classification problem is when the output variable is a category, such as "Red" or "Blue" or "Disease" and "No disease" or "Pass" and "Fail", "Spam" and "Not spam".
    - Answers **"What class?"**
    - Applied when the output has finite and discreet values



- **Regression**: A regression problem is when the output variable is a real value, such as "dollars" or "weight".
    - Answers **"How much?"**
    - Applied when the output is a continuous number
    - A simple regression algorithm: y = wx + b.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Amazon uses supervised learning algorithms to predict what items the user may like based on the purchase history of similar classes of users.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

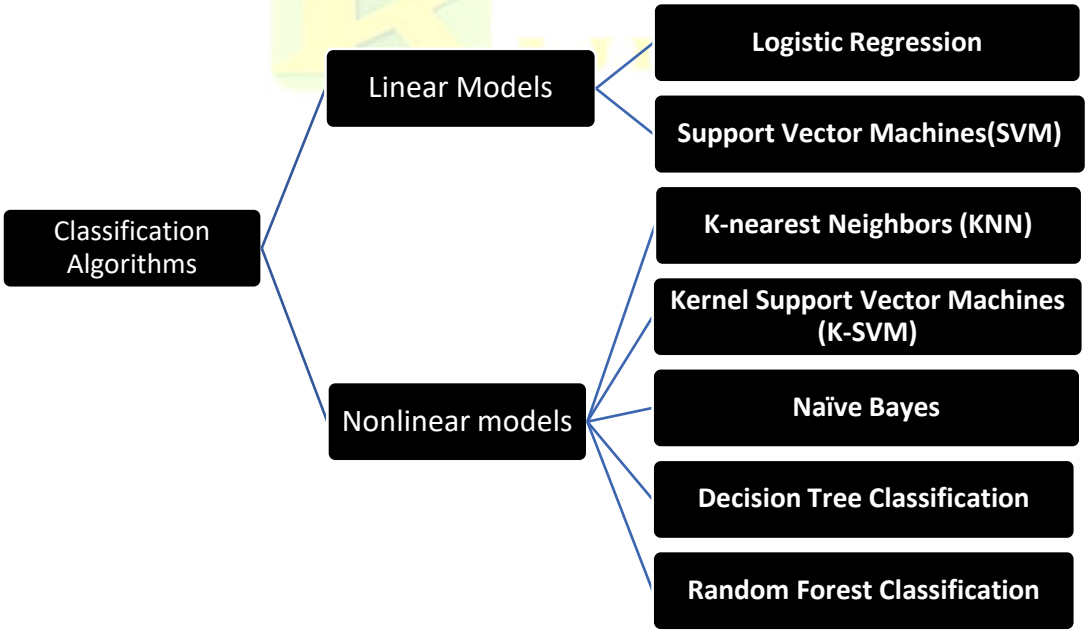
Classification

Classification problems categorize all the variables that form the output. Examples of these categories formed through classification would include demographic data such as marital status, sex, or age. The most common model used for this type of service status is the support vector machine. The support vector machines set forth to define the linear decision boundaries.

- Classification is a type of supervised learning.
- It specifies the class to which data elements belong to.
- It is best used when the output has finite and discreet values.
- It predicts a class for an input variable.

There are 2 types of classification, binomial and multi-class.

Types of Classification Algorithms

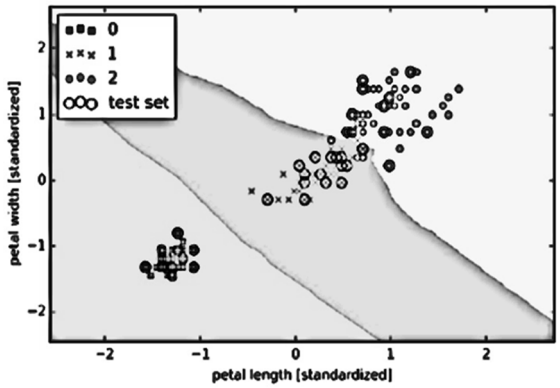


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

1. K-Nearest Neighbors (KNN):

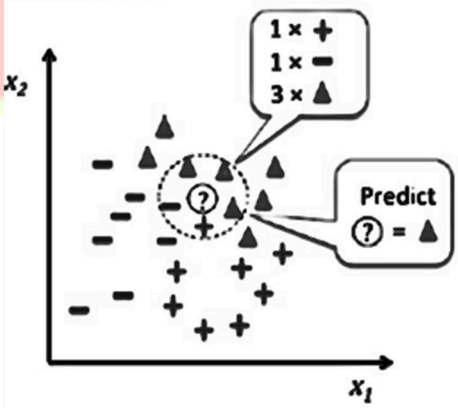
- K-nearest Neighbors algorithm is used to assign a data point to clusters based on similarity measurement.
- It uses supervised method for classification.

The steps to write a k-means algorithm are as given below:



- Choose the number of k and a distance metric. (k = 5 is common)
- Find k-nearest neighbors of the sample that you want to classify.
- Assign the class label by majority vote.

KNN Classification



A new input point is classified in the category such that it has the greatest number of neighbors from that category.

For example:

- Classify a patient as high risk or low risk.
- Mark email as spam or ham.

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points.
4. Calculate the distance between test data and each row of training data. Here we will use **Euclidean distance** as our distance metric since it's the most popular method.
5. Sort the calculated distances in ascending order based on distance values.
6. Get top k rows from the sorted array.
7. Get the most frequent class of these rows.
8. Return the predicted class

Python:

```
import pandas as pd          // pandas to read the datasets

from sklearn.cross_validation import train_test_split    // Older Version {Split into train | test}

from sklearn.model_selection import train_test_split    // Newer Version

df=pd.read_csv("iris_dataset.csv")    // Load the Dataset

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)     {Data is splited into 80:20}

from sklearn.neighbors import KNeighborsClassifier       //KNN package

knn = KNeighborsClassifier(n_neighbors = 5)          //K=5

knn.fit(X_train, Y_train)                            //We always fit train data to predicate the new data

train_score= knn.score(X_train, Y_train)  //Score is the value to describe the algorithm for the given
test_score= knn.score(X_test, Y_test)        split the data sets data set ranging score from
                                            (>0 and <=1). This will generate the result on the basis
                                          of datasets and how you

print("The accuracy of the Knn classifier on training data is {:f}".format(train_score))

print("The accuracy of the Knn classifier on test data is {:f}".format(test_score))
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2. Naïve Bays:

- According to Bayes model, the conditional probability P(Y|X) can be calculated as: P(Y|X) = P(X|Y)*P(Y) / P(X)
- This means you have to estimate a very large number of P(X|Y) probabilities for a relatively small vector space X.
- For example, for a Boolean Y and 30 possible Boolean attributes in the X vector, you will have to estimate 3 billion probabilities P(X|Y).
- To make it practical, a Naïve Bayes classifier is used, which assumes conditional independence of P(X) to each other, with a given value of Y.
- This reduces the number of probability estimates to 2*30=60 in the above example.

How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	**No**	**Yes**
Overcast	-	4
Rainy	3	2
Sunny	2	3
Grand Total	**5**	**9**

Likelihood Table				
Weather	**No**	**Yes**		
Overcast	-	4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
Grand Total	**5**	**9**		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have,

P (Sunny |Yes) = 3/9 = 0.33,
P(Sunny) = 5/14 = 0.36,
P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Python:

```
import pandas as pd          // pandas to read the datasets
from sklearn.cross_validation import train_test_split    // Older Version {Split into train | test}
from sklearn.model_selection import train_test_split    // Newer Version
df=pd.read_csv("iris_dataset.csv")    // Load the Dataset
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)     {Data is splited into 80:20}
from sklearn.naive_bayes import GaussianNB   //package for naïve bays
nb = GaussianNB()
nb.fit(X_train, Y_train)
train_score= nb.score(X_train, Y_train)
test_score= nb.score(X_test, Y_test)
print("The accuracy of the Gaussian NB on training data is {:f}".format(train_score))
print("The accuracy of the Gaussian NB on test data is {:f}".format(test_score))
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also performs well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naive Bayes Algorithms

- Real Time Prediction
- Multi Class Prediction
- Text Classification/ Spam Filtering/ Sentiment Analysis
- Recommendation System

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3. Logistic Regression:

- This refers to a regression model that is used for classification.
- Logistic regression is a supervised learning algorithm because it uses true labels for training. Supervised learning algorithm should have **input variables (X)** and a **target variable (Y)** when you train the model.
- Logistic regression is a classification algorithm, don't confuse with the name regression.
- This method is widely used for binary classification problems. It can also be extended to multi-class classification problems.
- Here, the dependent variable is categorical: **y ϵ {0, 1}**
- A binary dependent variable can have only two values, like 0 or 1, win or lose, pass or fail, healthy or sick, etc.
- In this case, you model the probability distribution of output y as 1 or 0. This is called as sigmoid probability (σ).
- The probability in the logistic regression is often represented by the Sigmoid function (also called the logistic function or the S-curve):

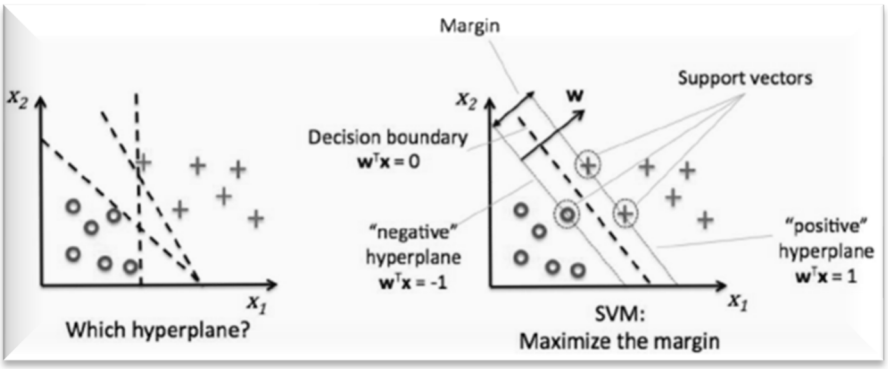
$$S(t) = \frac{1}{1 + e^{-t}}$$

Python:

```
import pandas as pd          // pandas to read the datasets
from sklearn.cross_validation import train_test_split    // Older Version {Split into train | test}
from sklearn.model_selection import train_test_split    // Newer Version
df=pd.read_csv("iris_dataset.csv")    // Load the Dataset
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)     {Data is splited into 80:20}

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
print("The accuracy of the Logistic Regression on training data is {:f}".format(lr.score(X_train, Y_train)))
print("The accuracy of the Logistic Regression on test data is {:f}".format(lr.score(X_test, Y_test)))
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4. Support Vector Machine (SVM):



- SVMs are classification algorithms used to assign data to various classes. They involve detecting hyperplanes which segregate data into classes.
- SVMs are very versatile and are also capable of performing linear or nonlinear classification, regression, and outlier detection.
- Once ideal hyperplanes are discovered, new data points can be easily classified.
- The optimization objective is to find "maximum margin hyperplane" that is farthest from the closest points in the two classes (these points are called support vectors).
- In the given figure, the middle line represents the hyperplane.

Python:

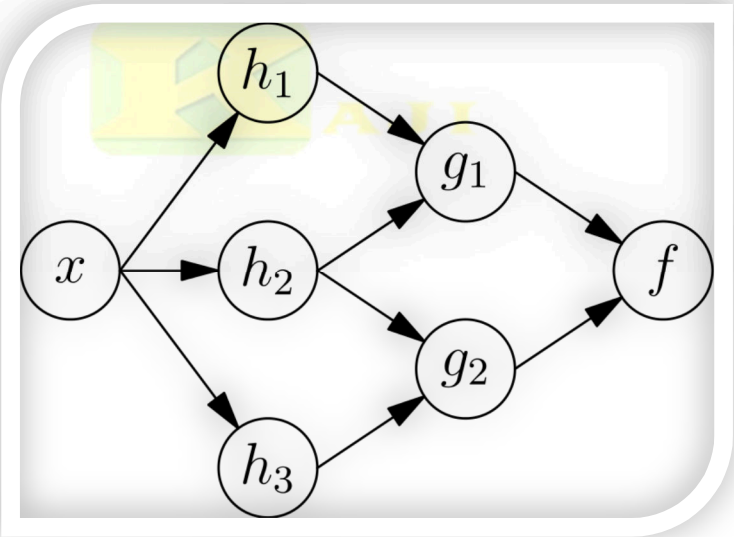
```
import pandas as pd          // pandas to read the datasets
from sklearn.cross_validation import train_test_split    // Older Version {Split into train | test}
from sklearn.model_selection import train_test_split    // Newer Version
df=pd.read_csv("iris_dataset.csv")    // Load the Dataset
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20)     {Data is splited into 80:20}
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, Y_train)
print("The accuracy of the SVM classifier on training data is {:f}".format(svm.score(X_train, Y_train)))
print("The accuracy of the SVM classifier on test data is {:f}".format(svm.score(X_test, Y_test)))
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5. Neural Networks:

- Neural networks are one of those cool words that are often used to lend credence to research.
- Neural nets take inspiration from the learning process occurring in human brains.
- They consist of an artificial network of functions, called parameters, which allows the computer to learn, and to fine tune itself, by analyzing new data.
- Each parameter, sometimes also referred to as neurons, is a function which produces an output, after receiving one or multiple inputs.
- Those outputs are then passed to the next layer of neurons, which use them as inputs of their own function, and produce further outputs.
- Those outputs are then passed on to the next layer of neurons, and so it continues until every layer of neurons have been considered, and the terminal neurons have received their input.
- Those terminal neurons then output the final result for the model.

Below figure shows a visual representation of such a network. The initial input is x, which is then passed to the first layer of neurons (the h bubbles in Figure), where three functions consider the input that they receive, and generate an output. That output is then passed to the second layer (the g bubbles in Figure). There further output is calculated, based on the output from the first layer. That secondary output is then combined to yield a final output of the model.

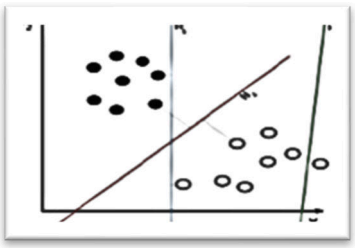


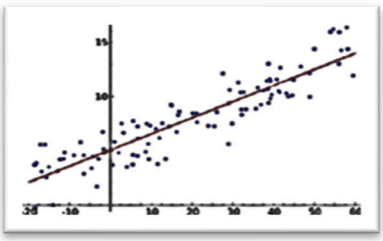
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Regression

- A regression problem is when the output variable is a real value, such as "dollars" or "weight".
- Regression technique predicts a single output value using training data.
- Example: You can use regression to predict the house price from training data. The input variables will be locality, size of a house, etc.

Classification	Regression
Classification means to group the output into a class.	Regression means to predict the output value using training data.
Discrete/Categorial values.	Real/Continuous values.
Method of calculation by measuring accuracy.	Method of calculation by measurement of root mean square error.
It predicts a label.	It predicts a quantity.
Find Decision boundary.	Find Best fit line.
Predicting student pass or fail	Predicting student Marks percentage.
Classification problem: Predicting the type of tumor i.e. harmful or not harmful.	Regression problem: Predicting the house rent price.





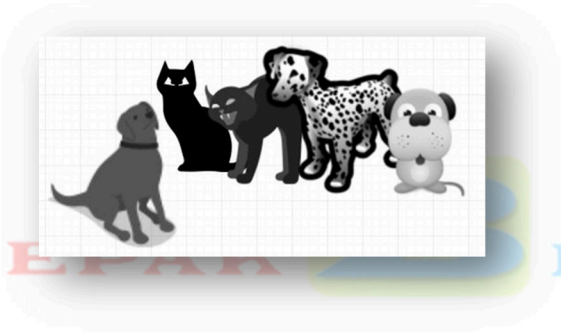
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.4 Unsupervised Learning:

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore, machine is restricted to find the hidden structure in unlabeled data by our-self.

For instance, suppose it is given an image having both dogs and cats which have not seen ever.



Thus, the machine has no idea about the features of dogs and cat so we can't categorize it in dogs and cats. But it can categorize them according to their similarities, patterns, and differences i.e., we can easily categorize the above picture into two parts. First may contain all pics having dogs in it and second part may contain all pics having cats in it. Here you didn't learn anything before, means no training data or examples.

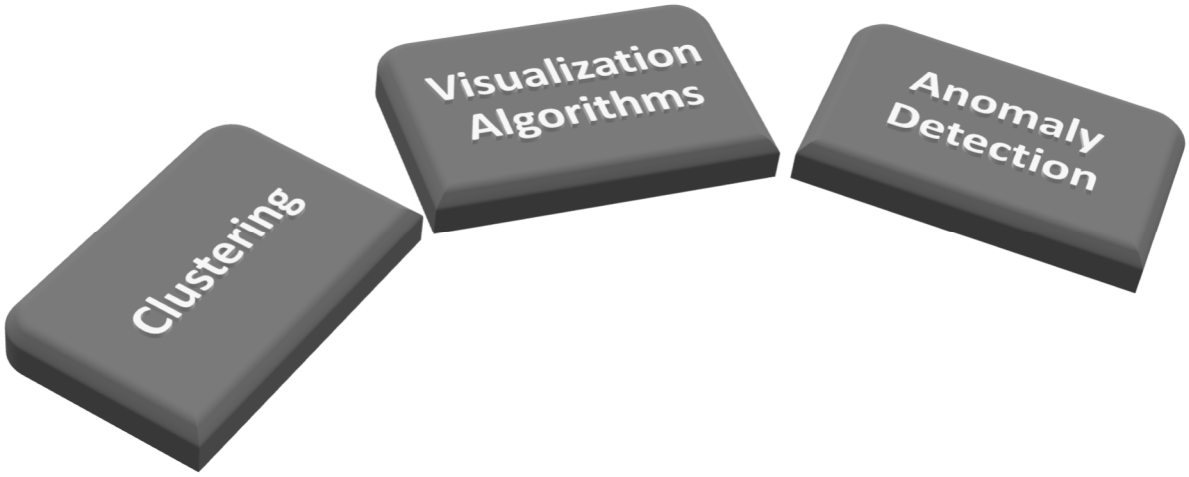
Unsupervised learning classified into two categories of algorithms:

Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Types of Unsupervised Learning

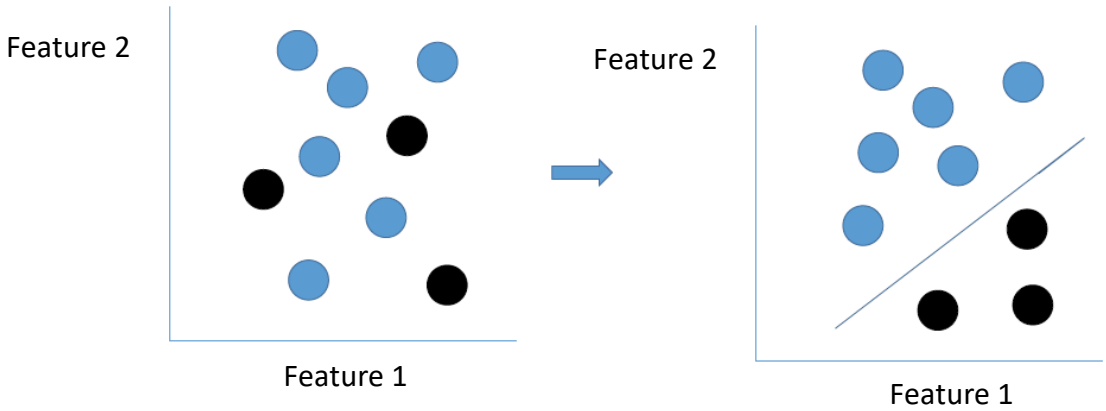


NASA uses unsupervised learning to create clusters of heavenly bodies, with each cluster containing objects of a similar nature.

DEEPAK BHATTA

1. CLUSTERING

The most common unsupervised learning method is cluster analysis. It is used to find data clusters so that each cluster has the most closely matched data.



Example:

An online news portal segments articles into various categories like Business, Technology, Sports, etc.

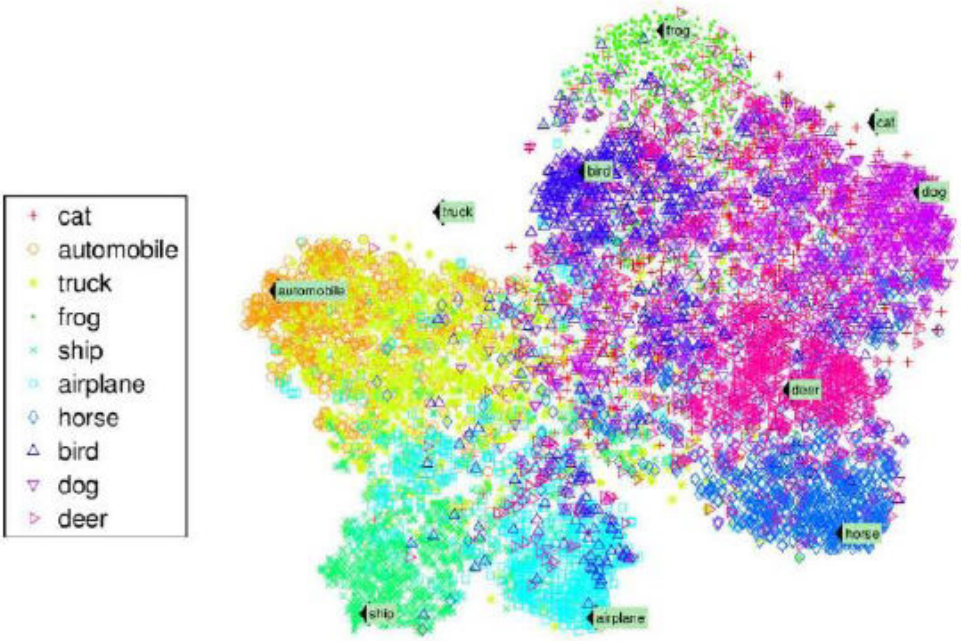
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2. VISUALIZATION ALGORITHMS

Visualization algorithms are unsupervised learning algorithms that accept unlabeled data and display this data in an intuitive 2D or 3D format. The data is separated into somewhat clear clusters to aid understanding.

Example:

In the figure, the animals are rather well separated from vehicles. Horses are close to deer but far from birds, and so on.



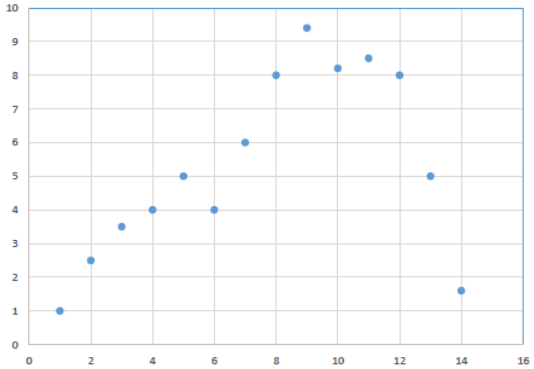
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3. ANOMALY DETECTION

This algorithm detects anomalies in data without any prior training.

Example:

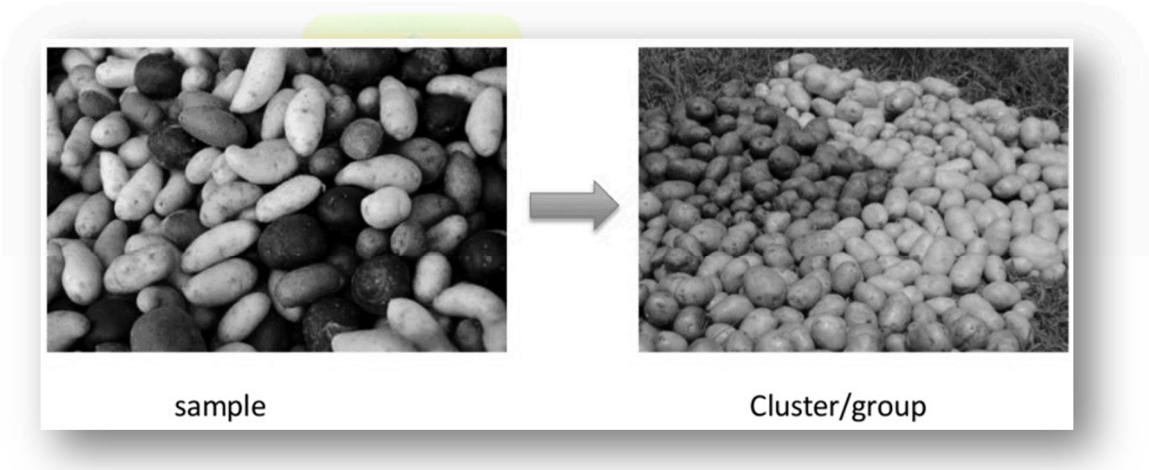
It can detect suspicious credit card transactions and differentiate a criminal from a set of people.



5.4.1 Clustering, K-Means

DEEPAK BHATTA

Clustering:



Clustering is an important concept when it comes to unsupervised learning. It mainly deals with finding a structure or pattern in a collection of uncategorized data. Clustering algorithms will process your data and find natural clusters(groups) if they exist in the data. You can also modify how many clusters your algorithms should identify. It allows you to adjust the granularity of these groups.
Cluster analysis or clustering is the most commonly used technique of unsupervised learning. It is used to find data clusters such that each cluster has the most closely matched data.

In simple, A group of objects having similar property is called clustering.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Why use Clustering?

Grouping similar entities together help profile the attributes of different groups. In other words, this will give us insight into underlying patterns of different groups. There are many applications of grouping unlabeled data, for example, you can identify different groups segments of customers and market each group in a different way to maximize the revenue.

Another example is grouping documents together which belongs to the similar topics etc.

Clustering is also used to reduces the dimensionality of the data when you are dealing with copious *[abundant in quantity (प्रचुर)]* number of variables.

Euclidian Distance: The Euclidean distance or Euclidean metric is the ordinary straight-line distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space.

The Euclidean distance between points p and q is the length of the lie segment connecting them.

$$\boldsymbol{P(x_1 y_1), \qquad Q(x_2 y_2)}$$

$$\boldsymbol{d = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}}$$

K-Means

K-means Clustering: Example

PROBLEM STATEMENT

Let's say, in Sudur-Pacchim Pardesh (Province No.7) the local government tries to identify high density clusters to build hospitals (no other ground truth or features are provided apart from the population data). How can the clusters be identified?

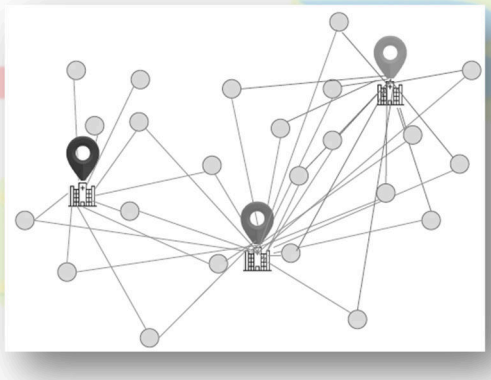
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

STEP 1: RANDOMLY PICK K CENTROIDS Start by picking k centroids. Assume, k = 3

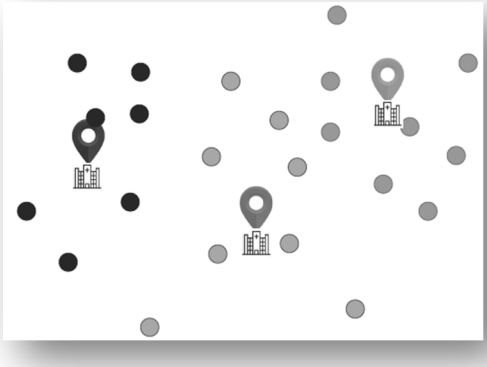
Finding the number of clusters: Use Elbow Method (to be reviewed later)

STEP 2: ASSIGN EACH POINT TO THE NEAREST CENTROID μ(j), j ∈ {1,…,k}.

The points are assigned such that the Euclidean distance of each point from the respective centroid is minimized.



STEP 3: MOVE EACH CENTROID TO THE CENTRE OF THE RESPECTIVE CLUSTER

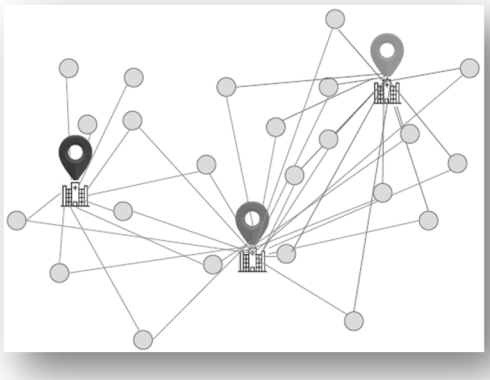


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

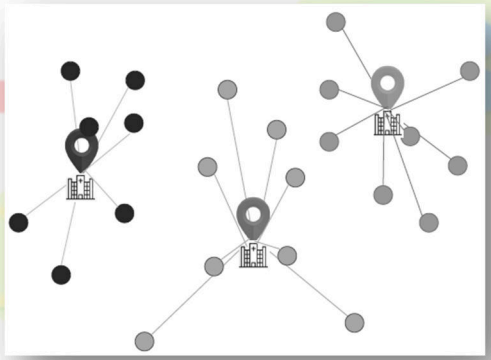
STEP 4: CALCULATE DISTANCE OF THE CENTROIDS FROM EACH POINT AGAIN

Calculate the Euclidean distance between each point and its centroid.

$$d = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$$

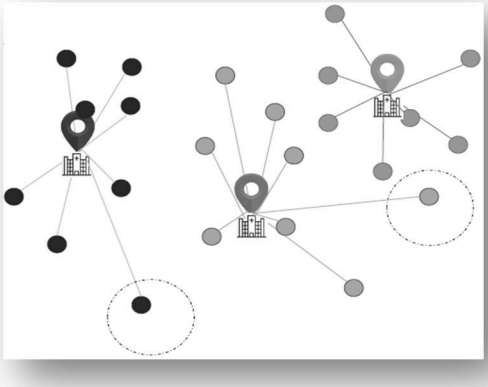


STEP 5: MOVE POINTS ACROSS CLUSTERS AND RE-CALCULATE THE DISTANCE FROM THE CENTROID



STEP 6: KEEP MOVING THE POINTS ACROSS CLUSTERS UNTIL THE EUCLIDEAN DISTANCE IS MINIMIZED

Repeat the steps until within-cluster Euclidean distance is minimized for each cluster (or a user-defined limit on number of iterations is reached)



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Python:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [2]:
```
df=pd.read_csv("Mall_Customers.csv")
df.head()
```

Out[2]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [3]:
```
df.shape
```

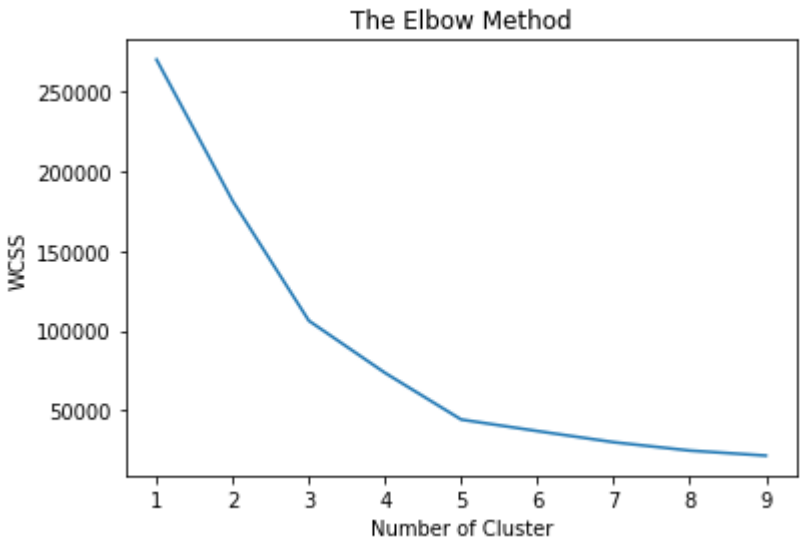
Out[3]: (200, 5)

In [4]:
```
X=df.iloc[:,[3,4]].values
```

Using the Elbow method to find the optimal number of Clusters

In [7]:
```
#WCSS=> Within Clusters Sum of Squares
wcss=[]
for i in range(1,10):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,10),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of Cluster')
plt.ylabel('WCSS')
plt.show()
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Fitting K-Means to the Dataset

```
In [9]: kmeans=KMeans(n_clusters=5,init='k-means++', random_state=42)
        Y_kmeans=kmeans.fit_predict(X)

In [11]: Y_kmeans

Out[11]: array([2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
                2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 0,
                2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 1, 4, 0, 4, 1, 4, 1, 4,
                0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
                1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
                1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4,
                1, 4])

In [14]: len=list(Y_kmeans)
         len.count(4)

Out[14]: 39
```


© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Visualizing the Clusters

```
In [16]: plt.scatter(X[Y_kmeans==0,0], X[Y_kmeans==0,1],s=20,c='red',label='cluster 1')
         plt.scatter(X[Y_kmeans==1,0], X[Y_kmeans==1,1],s=20,c='blue',label='cluster 2')
         plt.scatter(X[Y_kmeans==2,0], X[Y_kmeans==2,1],s=20,c='green',label='cluster 3')
         plt.scatter(X[Y_kmeans==3,0], X[Y_kmeans==3,1],s=20,c='cyan',label='cluster 4')
         plt.scatter(X[Y_kmeans==4,0], X[Y_kmeans==4,1],s=20,c='black',label='cluster 5')

         plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=100,c='yellow',label='Centroid')

         plt.title('CLuster of Cusstomers')
         plt.xlabel('Annual Income (k$)')
         plt.ylabel('Sspending Score (1-100)')
         plt.legend()
         plt.show()
```

Other Examples of K-means Clustering

- Grouping articles (example: Google news).
- Grouping products (example: Amazon, Flipkart etc.)
- Grouping customers who share similar interests, example: analyzing customers who like contemporary fashion vs. those who prefer traditional clothing.
- Classifying high risk and low risk patients from a patient pool.
- Segregating criminals from normal crowd in a security control process.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Supervised Vs Un-Supervised Learning

Parameters	Supervised Learning	Unsupervised Learning
Process	In a supervised learning model, input and output variables will be given.	In unsupervised learning model, only input data will be given
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data which is not labeled
Algorithms Used	Support vector machine, Neural network, Linear and logistics regression, random forest, and Classification trees.	Unsupervised algorithms can be divided into different categories: like Cluster algorithms, K-means, Hierarchical clustering, etc.
Computational Complexity	Supervised learning is a simpler method.	Unsupervised learning is computationally complex
Use of Data	Supervised learning model uses training data to learn a link between the input and the outputs.	Unsupervised learning does not use output data.
Accuracy of Results	Highly accurate and trustworthy method.	Less accurate and trustworthy method.
Real Time Learning	Learning method takes place offline.	Learning method takes place in real time.
Number of Classes	Number of classes is known.	Number of classes is not known.
Main Drawback	Classifying big data can be a real challenge in Supervised Learning.	You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.4.2 Dimensionality Reduction [Principal Component Analysis (PCA), Linear Discriminant Analysis]

Dimensionality Reduction:

- Dimensionality reduction involves transformation of data to new dimensions in a way that facilitates discarding of some dimensions without losing any key information.
- Large scale problems bring about several dimensions that can become very difficult to visualize.
- Some of such dimensions can be easily dropped for a better visualization.
- Example: Car attributes might contain maximum speed in both units, kilometer per hour, and miles per hour. One of these can be safely discarded in order to reduce the dimensions and simplify the data.

Dimensionality Reduction with Principal Component Analysis (PCA)

- Principal component analysis (PCA) is a technique for dimensionality reduction that helps in arriving at better visualization models.

DEEPAK BHATTA KAJI

Applications of PCA

Noise reduction

PCA can eliminate noise or non-critical aspects of the data set to reduce complexity. Also, during image processing or comparison, image compression can be done with PCA, eliminating the noise such as lighting variations in face images.

Compression

It is used to map high dimensional data to lower dimensions. For example, instead of having to deal with multiple car types (dimensions), we can cluster them into fewer types.

Preprocess

It reduces data dimensions before running a supervised learning program and saves on computations as well as reduces overfitting.

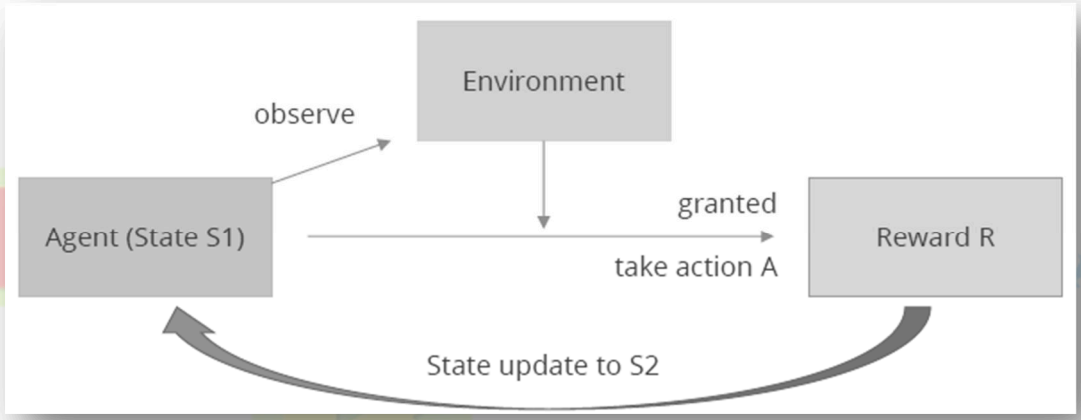
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.1 Reinforcement Learning

Reinforcement Learning is a type of Machine Learning that allows the learning system to observe the environment and learn the ideal behavior based on trying to maximize some notion of cumulative reward.

Features of Reinforcement Learning

- The learning system (agent) observes the environment, selects and takes certain actions, and gets rewards in return (or penalties in certain cases).
- The agent learns the strategy or policy (choice of actions) that maximizes its rewards over time.



Example of Reinforcement Learning

In a manufacturing unit, a robot uses deep reinforcement learning to identify a device from one box and put it in a container.

The robot learns this by means of a rewards-based learning system, which incentivizes it for the right action.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.6 Fuzzy Learning

What is Fuzzy Logic?

Fuzzy logic is an approach to computing based on "degree of truth" rather than the usual "True or False" (1 or 0) Boolean logic on which the modern computer is based.

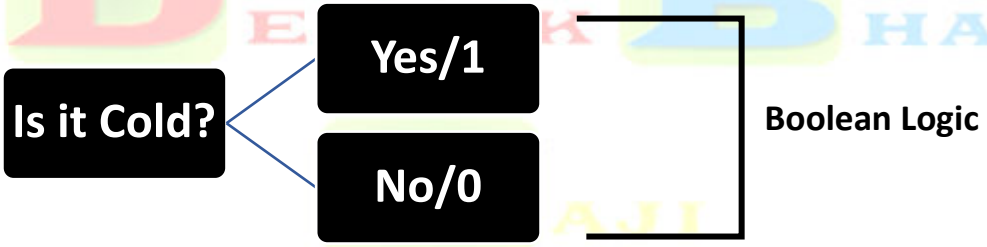
The idea of fuzzy logic was first announced by Dr. Lotfi Zadeh of the university of California at Berkeley in 1960s.

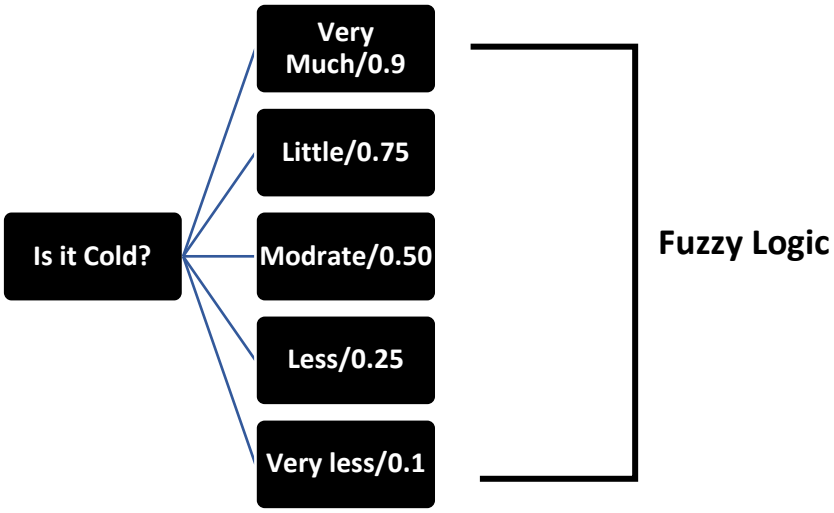
Fuzzy Logic (FL) is a method of reasoning that resembles human reasoning. This approach is similar to how humans perform decision making. And it involves all intermediate possibilities between YES and NO.

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

Fuzzy logic can be implemented in systems with various sizes and capabilities ranging from small micro-controllers to large, networked, workstation-based control systems.

Fuzzy logic can be implemented in hardware, software and or a combination of both.





© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

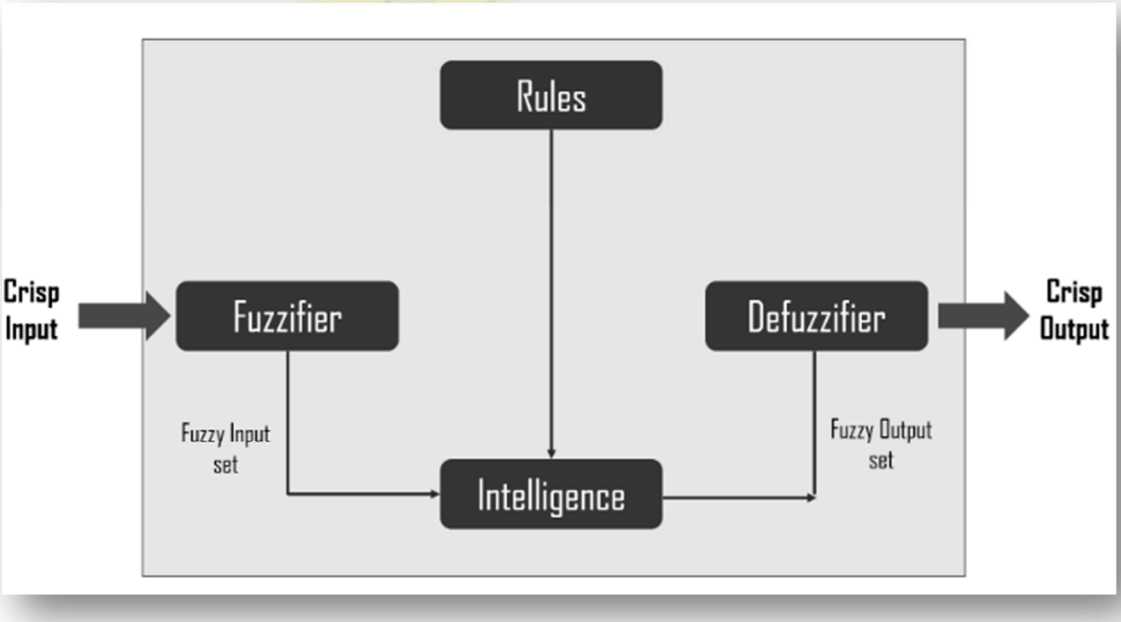
Applications of Fuzzy Logic

- The Fuzzy logic is used in various fields such as automotive systems, domestic goods, environment control, etc. Some of the common applications are:
- It is used in the aerospace field for altitude control of spacecraft and satellite.
- This controls the speed and traffic in the automotive systems.
- It is used for decision making support systems and personal evaluation in the large company business.
- It also controls the pH, drying, chemical distillation process in the chemical industry.
- Fuzzy logic is used in Natural language processing and various intensive applications in Artificial Intelligence.
- It is extensively used in modern control systems such as expert systems.
- Fuzzy Logic mimics how a person would make decisions, only much faster. Thus, you can use it with Neural Networks.

DEEPAK BHATTA

Fuzzy Logic Architecture

The fuzzy logic architecture consists of four main parts:



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- **Rules** – It contains all the rules and the if-then conditions offered by the experts to control the decision-making system. The recent update in the fuzzy theory provides different effective methods for the design and tuning of **fuzzy controllers**. Usually, these developments reduce the number of fuzzy rules.
- **Fuzzification** – This step converts inputs or the crisp numbers into fuzzy sets. You can measure the crisp inputs by sensors and pass them into the **control system** for further processing. It splits the input signal into five steps such as-

LP	X is Large Positive
MP	X is Medium Positive
S	Small
MN	X is Medium Negative
LN	X is Large Negative

- **Inference Engine** – It determines the degree of match between fuzzy input and the rules. According to the input field, it will decide the rules that are to be fired. Combining the fired rules, form the control actions.
- **Defuzzification** – The Defuzzification process converts the fuzzy sets into a crisp value. There are different types of techniques available, and you need to select the best-suited one with an expert system.

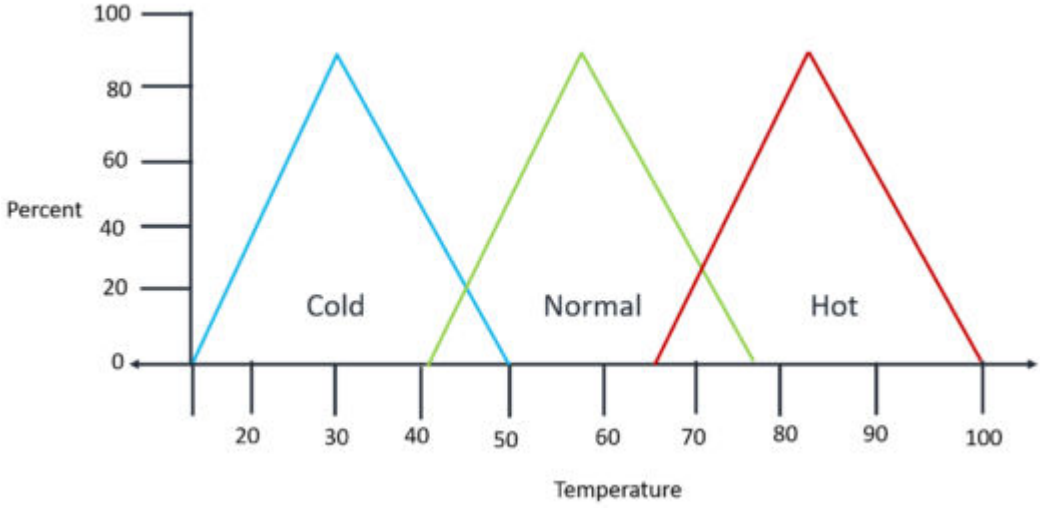
Fuzzy Logic in AI: Example

The design of a fuzzy logic system starts with a set of membership functions for each input and a set for each output. A set of rules is then applied to the membership functions to yield a crisp output value. Let's take an example of process control and understand fuzzy logic.

Step 1

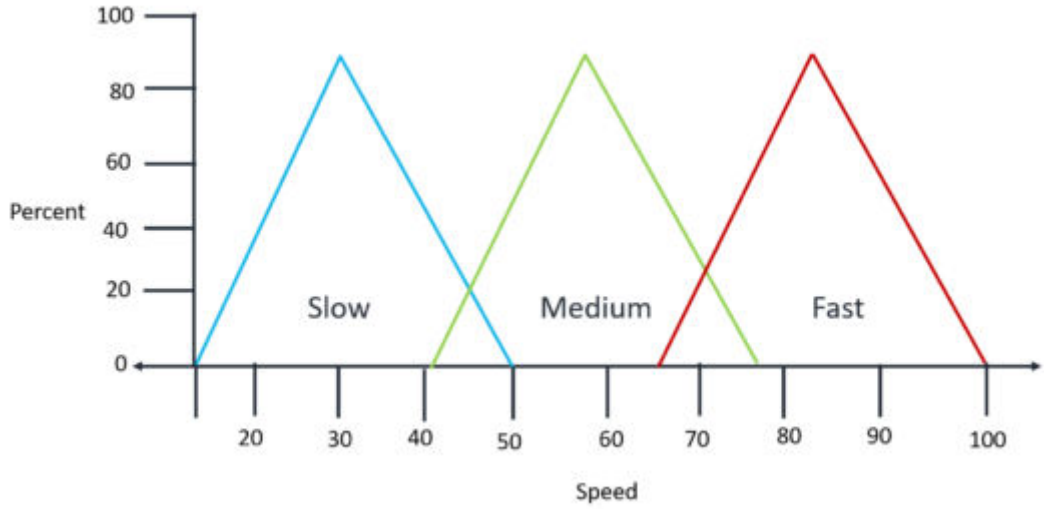
Here, **Temperature** is the input and **Fan Speed** is the output. You have to create a set of membership functions for each input. A membership function is simply a graphical representation of the fuzzy variable sets. For this example, we will use three fuzzy sets, **Cold, Warm** and **Hot**. We will then create a membership function for each of three sets of temperature:

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Step 2

In the next step, we will use three fuzzy sets for the output, **Slow, Medium** and **Fast**. A set of functions is created for each output set just as for the input sets.

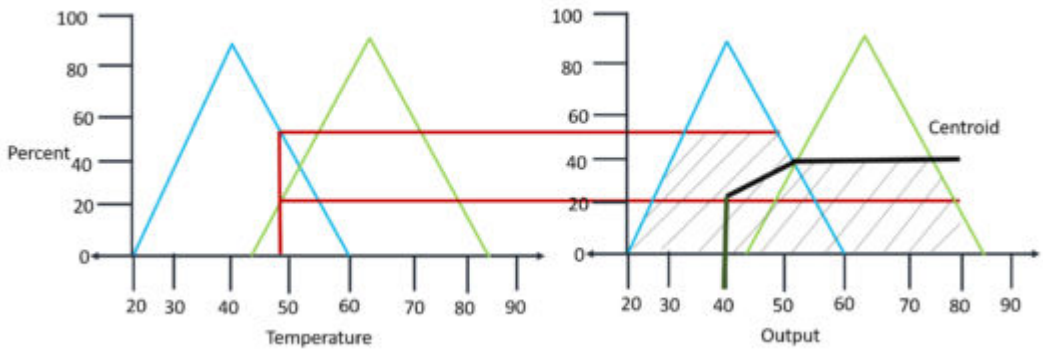


Step 3

Now that we have our membership functions defined, we can create the rules that will define how the membership functions will be applied to the final system. We will create three rules for this system.

- **If Hot then Fast**
- **If Warm then Medium**
- **And, if Cold then Slow**

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



These rules apply to the membership functions to produce the crisp output value to drive the system. Thus, for an input value of **52 degrees**, we intersect the membership functions. Here, we are applying two rules as the intersection occurs on both functions. You can extend the intersection points to the output functions to produce an intersecting point. You can then truncate the output functions at the height of the intersecting points.

Advantages of Fuzzy Logic

Fuzzy logic provides simple reasoning similar to human reasoning. There are more such **advantages** of using this logic, such as:

- The structure of Fuzzy Logic Systems is **easy and understandable.**
- Fuzzy logic is widely used for **commercial** and **practical purposes.**
- It helps you to **control machines** and consumer products.
- It helps you to deal with the **uncertainty in engineering.**
- Mostly **robust** as no precise inputs required.
- If the feedback sensor stops working, you can **program it** into the situation.
- You can **easily modify** it to improve or alter system performance.
- **Inexpensive sensors** can be used which helps you to keep the overall system cost and complexity low.

Disadvantages of Fuzzy Logic

Fuzzy logic is **not always accurate**. So, the results are perceived based on assumptions and may not be widely accepted

- It **cannot recognize** machine learning as-well-as neural network type patterns.
- **Validation and Verification** of a fuzzy knowledge-based system needs **extensive testing** with hardware.
- Setting exact, fuzzy rules and, membership functions is a **difficult task.**
- At times, the fuzzy logic is **confused** with **probability theory.**

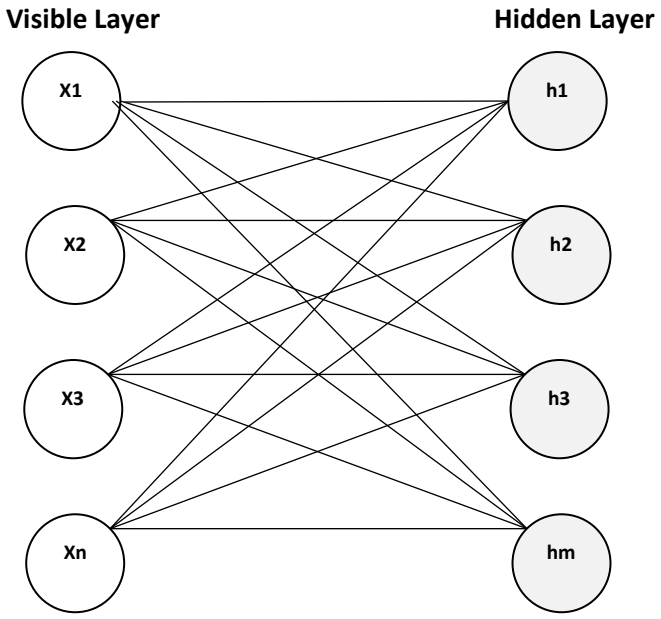
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.7 Boltzmann Machines

A Hopfield Net consisting of Binary Stochastic Neuron with hidden units is called Boltzmann Machine.

A Boltzmann Machine is a network of symmetrically connected, neuron like units that make stochastic decisions about whether to be ON or OFF.

Structure of Boltzmann Machine



The stochastic neurons of Boltzmann machine are in two groups: Visible and Hidden.

Visible neurons provide an interface between the net and its environment.

During the training phase, the visible neurons are clamped; the hidden neurons always operate freely, they are used to explain underlying constraints in the environmental input vectors.

Boltzmann machines are stochastic and generative neural networks capable of learning internal representations and are able to represent and (given sufficient time) solve difficult combinatoric problems.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

They are named after the Boltzmann distribution (**also known as Gibbs Distribution**) which is an integral part of Statistical Mechanics and helps us to understand the impact of parameters like Entropy and Temperature on the Quantum States in Thermodynamics.

That's why they are called Energy-Based Models (EBM). They were invented in 1985 by Geoffrey Hinton, then a Professor at Carnegie Mellon University, and Terry Sejnowski, then a Professor at Johns Hopkins University.

Some key features Boltzmann Machine:

- Each node is connected to all other nodes.
- There is no single output in the machine, only hidden units and visible units, where the visible units represent the data. This model tries to understand the distribution of the data and recreate the data based on that distribution.

Application of Boltzmann Machine:

- Stock market trend prediction
- Character recognition
- Face recognition
- Internet application
- Cancer detection
- Loan application
- Decision making

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

5.8 Deep Learning

What is Deep Learning?

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

Deep Learning is a specialized form of Machine Learning that uses supervised, unsupervised, or semi-supervised learning to learn data representations. It is similar to the structure and function of the human nervous system.

Why Deep Learning?

- Experts have discovered multi-layered learning networks that can be leveraged for deep learning as they learn in layers.
- The vast availability of Big Data enables machines to be trained.
- Scientists have figured out that high-performing graphics processing units (GPU) can be used for deep learning.
- GPUs allow parallel computing, unlike conventional Central Processing Units (CPU) that use sequential processing.

Difference: Machine Learning and Deep Learning

Machine Learning allows one to learn algorithms that represent the relationship between input and output.	Deep Learning is a more specialized form of Machine Learning that allows one to learn the representations of data itself, which leads to more complex decision-making, such as object detection, speech processing, or text recognition.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Applications of Deep Learning:

1. Automated Driving:

Automated driving is becoming one of the most emerging topics nowadays. Various companies are applying deep learning technique to create an automated vehicle which doesn't requires human supervision to function. They are automatically detecting objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

2. Medical Research:

Various researchers are working on deep learning so that they can create a machine which are automatically detected the defect or disease in living body by just observing its patterns. Cancer researchers are using deep learning to automatically detect cancer cells.

3. Aerospace and Defense:

Deep learning is used in various defense sectors as to identify an unauthorized or can be used to locate areas of interest by using given environmental aspects. Using this information, we can give commands or order to troops that whether it is safe or unsafe to work in environment.

THE END

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

PU Examination Questions with Solution

K-Means Algorithm

We know the K-Means Clustering Algorithm

Step 1: Select initial centroids at random of each cluster (from datasets of n objects).

Step 2: Assign each object to the cluster with the nearest centroid.

Step 3: Compute each centroid as the mean of the objects assigned to it. (Get the mean value of each cluster) & make them as a new centroid.

Step 4: Repeat **Step 2** and **Step 3** until no change in mean value.

Step 5: Stop when no more new assignment.

Euclidean Distance:

$$d = \sqrt{(x-a)^2 + (y-b)^2}$$

x & y => Observed value from the given dataset.

a & b => Centroid value or Mean value

DEEPAK BHATTA KAJI

[2016 Fall 4.b]

When is it useful to use clustering? The following data set contains the scores of two variables on each of seven individuals. Taking 1 and 4 as initial cluster center, applying K-means clustering to group the given data:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Ans: Clustering is used to find structure in unlabeled data. It's the most common form of unsupervised learning. Given a dataset we don't know anything about, a clustering algorithm can discover groups of objects where the average distances between the members of each cluster are closer than to members in other clusters.

Clustering → is grouping similar data → together → and this can be used for prediction or just clustering.

Or even dimension reduction → depending on how we use this.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Why do we cluster?

- **Summarizing data**
  - Look at large amount of data
  - Represent a large continuous vector with the cluster number
- **Counting**
  - Computing feature histograms
- **Prediction**
  - Data points in the same cluster may have the same labels

Sol^n: Given, 1 and 4 as initial cluster center.

Subject	A	B
1	**1.0**	**1.0**
2	1.5	2.0
3	3.0	4.0
4	**5.0**	**7.0**
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Cluster	Individual	Mean Value / Centroid
K1 (Group 1)	1	(1.0, 1.0)
K2 (Group 2)	4	(5.0, 7.0)

The remaining individual are now examined in sequence and allocated to the cluster to which they are closest in terms of Euclidean distance to the cluster mean. The mean value (Centroid) is re-calculated each time a new member is added.

We know, Euclidean distance

$$d = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$$

A	B	K1 = (1.0, 1.0) $\sqrt{(A-1.0)^2 + (B-1.0)^2}$	K2 = (5.0, 7.0) $\sqrt{(A-5.0)^2 + (B-7.0)^2}$	Cluster Assignment
1.0	1.0	**0**	7.211	K1
1.5	2.0	**1.118**	6.10	K1
3.0	4.0	**3.60**	3.60	K1
5.0	7.0	7.211	**0**	K2
3.5	5.0	4.716	**2.5**	K2
4.5	5.0	5.315	**2.061**	K2
3.5	4.5	4.301	**2.91**	K2

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Here how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(1.0, 1.0), (1.5, 2.0), (3.0, 4.0)}

K2 = {(5.0, 7.0), (3.5, 5.0), (4.5, 5.0), (3.5, 4.5)}

Now, Calculate Mean value (Centroid)

$$K1 = (A, B) = \left(\frac{A_1+A_2+A_3}{3}\right), \left(\frac{B_1+B_2+B_3}{3}\right)$$

$$= \left(\frac{1.0+1.5+3.0}{3}\right), \left(\frac{1.0+2.0+4.0}{3}\right)$$

$$= \left(\frac{5.5}{3}\right), \left(\frac{7}{3}\right)$$

K1 = 1.83, 2.33

$$K2 = (A, B) = \left(\frac{A_1+A_2+A_3+A_4}{4}\right), \left(\frac{B_1+B_2+B_3+B_4}{4}\right)$$

$$= \left(\frac{5.0+3.5+4.5+3.5}{4}\right), \left(\frac{7.0+5.0+5.0+4.5}{4}\right)$$

$$= \left(\frac{16.5}{4}\right), \left(\frac{21.5}{4}\right)$$

K2 = 4.125, 5.375

Now, Update the cluster centroid and calculate Euclidean distance:

A	B	K1 = (1.8, 2.3) $\sqrt{(A-1.8)^2+(B-2.3)^2}$	K2 = (4.1, 5.4) $\sqrt{(A-4.1)^2+(B-5.4)^2}$	Cluster Assignment
1.0	1.0	**1.526**	5.38	K1
1.5	2.0	**0.424**	4.28	K1
3.0	4.0	2.080	**1.78**	K2
5.0	7.0	5.685	**1.83**	K2
3.5	5.0	3.190	**0.72**	K2
4.5	5.0	3.818	**0.56**	K2
3.5	4.5	2.780	**1.08**	K2

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Here, how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(1.0, 1.0), (1.5, 2.0)}

K2 = {(3.0, 4.0), (5.0, 7.0), (3.5, 5.0), (4.5, 5.0), (3.5, 4.5)}

Now, Calculate Mean value (Centroid)

$$K1 = (A, B) = \left(\frac{A_1+A_2}{2}\right), \left(\frac{B_1+B_2}{2}\right)$$

$$= \left(\frac{1.0+1.5}{2}\right), \left(\frac{1.0+2.0}{2}\right)$$

$$= \left(\frac{2.5}{2}\right), \left(\frac{3}{2}\right)$$

K1 = 1.25=1.3, 1.5

$$K2 = (A, B) = \left(\frac{A_1+A_2+A_3+A_4+A_5}{5}\right), \left(\frac{B_1+B_2+B_3+B_4+B_5}{5}\right)$$

$$= \left(\frac{3.0+5.0+3.5+4.5+3.5}{5}\right), \left(\frac{4.0+7.0+5.0+5.0+4.5}{5}\right)$$

$$= \left(\frac{19.5}{5}\right), \left(\frac{25.5}{5}\right)$$

K2 = 3.9, 5.1

Now, Update the cluster centroid and calculate Euclidean distance:

A	**B**	**K1 = (1.3, 1.5)** $\sqrt{(A-1.3)^2+(B-1.5)^2}$	**K2 = (3.9, 5.1)** $\sqrt{(A-3.9)^2+(B-5.1)^2}$	**Cluster Assignment**
1.0	1.0	**0.583**	5.021	K1
1.5	2.0	**0.538**	3.92	K1
3.0	4.0	3.023	**1.421**	K2
5.0	7.0	6.628	**2.19**	K2
3.5	5.0	4.134	**0.41**	K2
4.5	5.0	4.74	**0.60**	K2
3.5	4.5	4.72	**0.72**	K2

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Here, how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(1.0, 1.0), (1.5, 2.0)}

K2 = {(3.0, 4.0), (5.0, 7.0), (3.5, 5.0), (4.5, 5.0), (3.5, 4.5)}

Now, Calculate Mean value (Centroid)

$$K1 = (A, B) = \left(\frac{A_1+A_2}{2}\right), \left(\frac{B_1+B_2}{2}\right)$$

$$= \left(\frac{1.0+1.5}{2}\right), \left(\frac{1.0+2.0}{2}\right)$$

$$= \left(\frac{2.5}{2}\right), \left(\frac{3}{2}\right)$$

K1 = 1.25=1.3, 1.5

$$K2 = (A, B) = \left(\frac{A_1+A_2+A_3+A_4+A_5}{5}\right), \left(\frac{B_1+B_2+B_3+B_4+B_5}{5}\right)$$

$$= \left(\frac{3.0+5.0+3.5+4.5+3.5}{5}\right), \left(\frac{4.0+7.0+5.0+5.0+4.5}{5}\right)$$

$$= \left(\frac{19.5}{5}\right), \left(\frac{25.5}{5}\right)$$

K2 = 3.9, 5.1

Since, mean value (Centroid) of K1 and K2 has no any change so, Stop no more new assignment.

So final partition table would be,

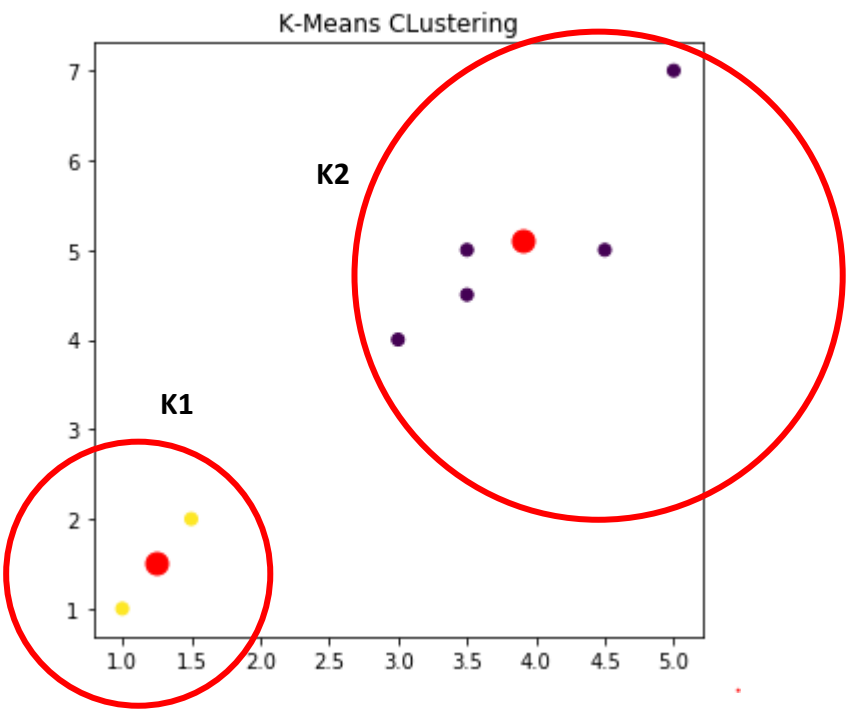
Cluster	**Individual**	**Mean Value / Centroid**
K1 (Group 1)	1, 2	(1.3, 1.5)
K2 (Group 2)	3, 4, 5, 6, 7	(3.9, 5.1)

----------X----------X----------

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Python:

```
In [1]: import numpy as np
        from sklearn.cluster import KMeans

In [2]: X2 = np.array([[1.0, 1.0], [1.5, 2.0], [3.0, 4.0],[5.0, 7.0], [3.5, 5.0], [4.5, 5.0], [3.5, 4.5]])

In [3]: X2_kmeans = KMeans(n_clusters=2).fit(X2)

In [4]: X2_kmeans.labels_

Out[4]: array([1, 1, 0, 0, 0, 0, 0])

In [5]: len=list(X2_kmeans.labels_)

In [6]: print("Total No. of Cluster K1= ",len.count(1))
        print("Total No. of Cluster K2= ",len.count(0))

        Total No. of Cluster K1=  2
        Total No. of Cluster K2=  5

In [7]: X2_kmeans.cluster_centers_

Out[7]: array([[3.9 , 5.1 ],
               [1.25, 1.5 ]])

In [9]: import matplotlib.pyplot as plt

        plt.figure(figsize=(12, 12))

        X2_pred = KMeans(n_clusters=2).fit_predict(X2)

        plt.subplot(221)
        plt.scatter(X2[:, 0], X2[:, 1], c=X2_pred)
        plt.scatter(X2_kmeans.cluster_centers_[:,0],X2_kmeans.cluster_centers_[:,1],s=120,c='red')
        plt.title("K-Means CLustering")
        plt.show()
```



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

[2018 Fall 3.b]

Define Bayes rule for probabilistic problem. If probability of symptoms on patients is 1/20 and probability of disease known by doctor is 1/45000, Also the probability of symptoms condition to disease is 50%. Now, calculate the probability of disease condition to symptoms using Bayes theorem. **[7]**

Ans: Bayes' Theorem is a way of finding a probability when we know certain other probabilities. Bayes' theorem is a way to figure out conditional probability. Conditional probability is the probability of an event happening, given that it has some relationship to one or more other events.

The formula is:

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

Which tells us: how often A happens given that B happens, written P(A|B),

When we know: how often B happens given that A happens, written P(B|A)

and how likely A is on its own, written P(A)

and how likely B is on its own, written P(B)

Sol^n: Given,

Probability of **Symptoms** on patients P(S) = 1/20 = 0.50

Probability of **Diseases** known by doctor P(D) = 1/45000 = 0.0000222

Probability of **Symptoms** condition to **Diseases** P(S|D) = 50% = 0.50

Here we have to, calculate the probability of **Disease** condition to **Symptoms** P(D|S) using Bayes theorem.

According to Bayes Theorem, $P(D|S) = \frac{P(D) * P(S|D)}{P(S)}$

$$= \frac{0.50 * 1/45000}{1/20}$$

$$= 0.000222 \quad Ans.$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

[2019 Fall 4.a]

What is Deep Learning? Explain the types of stochastic neurons of Boltzmann Machine?

Ans: Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

The field of artificial intelligence is essentially when machines can do tasks that typically require human intelligence. It encompasses machine learning, where machines can learn by experience and acquire skills without human involvement.

Examples of Deep Learning

1. Virtual assistants

Whether it's Alexa or Siri or Cortana, the virtual assistants of online service providers use deep learning to help understand your speech and the language humans use when they interact with them.

2. Translations

In a similar way, deep learning algorithms can automatically translate between languages. This can be powerful for travelers, business people and those in government.

3. Vision for driverless delivery trucks, drones and autonomous cars

The way an autonomous vehicle understands the realities of the road and how to respond to them whether it's a stop sign, a ball in the street or another vehicle is through deep learning algorithms. The more data the algorithms receive, the better they are able to act human-like in their information processing—knowing a stop sign covered with snow is still a stop sign.

4. Chatbots and service bots

Chatbots and service bots that provide customer service for a lot of companies are able to respond in an intelligent and helpful way to an increasing amount of auditory and text questions thanks to deep learning.

5. Image colorization

Transforming black-and-white images into color was formerly a task done meticulously by human hand. Today, deep learning algorithms are able to use the context and objects in the images to color them to basically recreate the black-and-white image in color. The results are impressive and accurate.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6. Facial recognition

Deep learning is being used for facial recognition not only for security purposes but for tagging people on Facebook posts and we might be able to pay for items in a store just by using our faces in the near future. The challenges for deep-learning algorithms for facial recognition is knowing it's the same person even when they have changed hairstyles, grown or shaved off a beard or if the image taken is poor due to bad lighting or an obstruction.

7. Medicine and pharmaceuticals

From disease and tumor diagnoses to personalized medicines created specifically for an individual's genome, deep learning in the medical field has the attention of many of the largest pharmaceutical and medical companies.

8. Personalized shopping and entertainment

Ever wonder how Netflix comes up with suggestions for what you should watch next? Or where Amazon comes up with ideas for what you should buy next and those suggestions are exactly what you need but just never knew it before? Yep, it's deep-learning algorithms at work.

DEEPAK BHATTA

The types of stochastic neurons of Boltzmann Machine are:

1. Restricted Boltzmann machine
2. Deep Boltzmann machine
3. Spike-and-slab RBMs

1. Restricted Boltzmann machine

Although learning is impractical in general Boltzmann machines, it can be made quite efficient in a restricted Boltzmann machine (RBM) which does not allow intralayer connections between hidden units. After training one RBM, the activities of its hidden units can be treated as data for training a higher-level RBM. This method of stacking RBMs makes it possible to train many layers of hidden units efficiently and is one of the most common deep learning strategies. As each new layer is added the generative model improves.

An extension to the restricted Boltzmann machine allows using real valued data rather than binary data.

One example of a practical RBM application is in speech recognition.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

2. Deep Boltzmann machine

Deep Boltzmann machine (DBM) is a type of binary pairwise Markov random field (undirected probabilistic graphical model) with multiple layers of hidden random variables. It is a network of symmetrically coupled stochastic binary units.

DBMs can learn complex and abstract internal representations of the input in tasks such as object or speech recognition, using limited, labeled data to fine-tune the representations built using a large set of unlabeled sensory input data. However, unlike DBNs and deep convolutional neural networks, they pursue the inference and training procedure in both directions, bottom-up and top-down, which allow the DBM to better unveil the representations of the input structures.

3. Spike-and-slab RBMs

The need for deep learning with real-valued inputs, as in Gaussian RBMs, led to the spike-and-slab RBM (ssRBM), which models continuous-valued inputs with binary latent variables. Similar to basic RBMs and its variants, a spike-and-slab RBM is a bipartite graph, while like GRBMs, the visible units (input) are real-valued. The difference is in the hidden layer, where each hidden unit has a binary spike variable and a real-valued slab variable. A spike is a discrete probability mass at zero, while a slab is a density over continuous domain; their mixture forms a prior.

[2019 Fall 4.b]

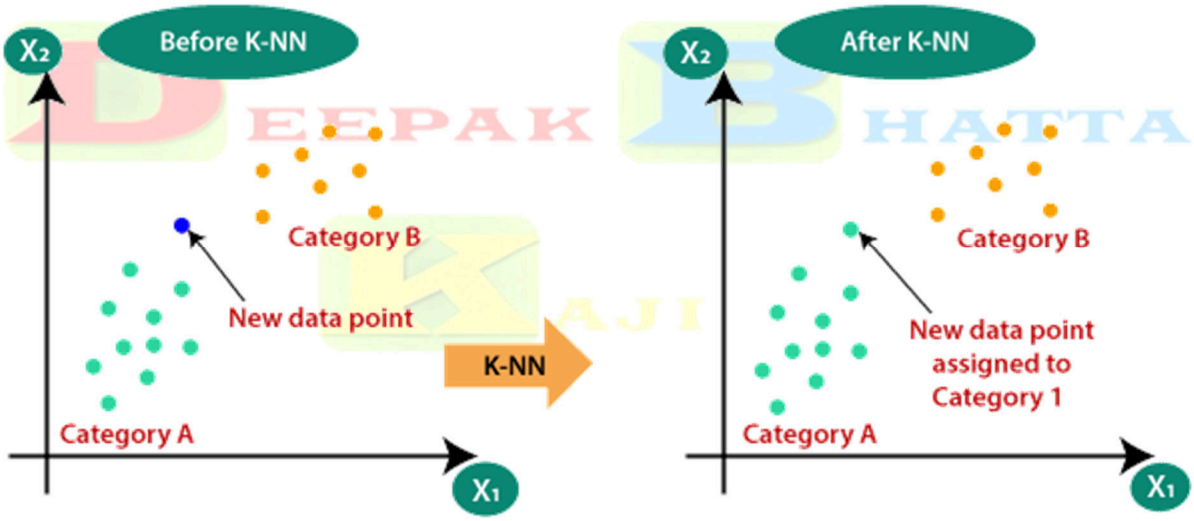
Why K-Nearest Neighbors Algorithm is known as Non-Parametric Lazy algorithm? Explain with example.

Ans:

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- **K-NN is a non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So, for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

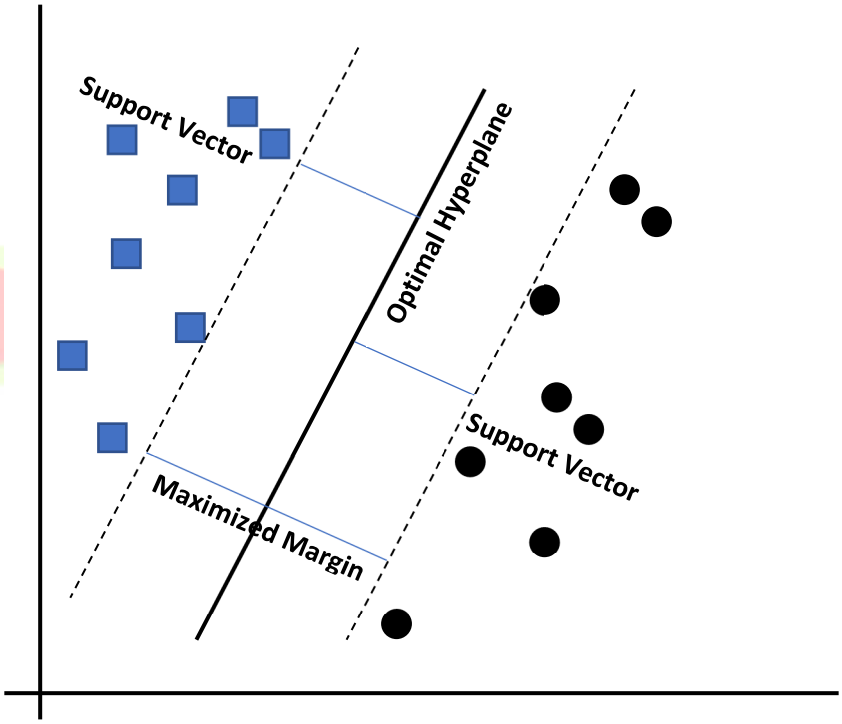
[2019 Fall 5.a]

How can we find the right hyperplane in SVM? Using K-Means clustering, cluster the following data into two clusters and show each step.

{2, 4, 10, 12, 3, 20, 30, 11, 25}

Ans: A hyperplane in an n-dimensional Euclidean space is a flat, n-1 Dimensional subset of that space that divides the space into two disconnected parts.

According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.



Sol^n: Given data set K = {2, 4, 10, 12, 3, 20, 30, 11, 25}, where k = 2

First sort the given dataset for easier distribution between two cluster.

K = {2, 3, 4, 10, 11, 12, 20, 25, 30, 25, 30}

How can you calculate Euclidean distance for one vector, this is?

$$\cancel{\sqrt{(x-a)^2}}$$

Let us assume Mean value (centroid)

M1 = 4 K1 = {2, 3, 4}	**M2 = 12** K2 = {10, 11, 12, 20, 25, 30}

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

So,

K	M1 = 4	M2 = 12	Cluster Assignment
2	**|4 – 2| = 2**	|12 – 2| = 10	K1
3	**|4 – 3| = 1**	|12 – 3| = 9	K1
4	**|4 – 4| = 0**	|12 – 4| = 8	K1
10	|4 – 10| = 6	**|12 – 10| = 2**	K2
11	|4 – 11| = 7	**|12 – 11| = 1**	K2
12	|4 – 12| = 8	**|12 – 12| = 0**	K2
20	|4 – 20| = 16	**|12 – 20| = 6**	K2
25	|4 – 25| = 21	**|12 – 25| = 8**	K2
26	|4 – 30| = 26	**|12 – 30| = 18**	K2

Now, Calculate Mean value (Centroid)

M1 = $\frac{2+3+4}{3} = \frac{9}{3} = \mathbf{3}$	**M2** = $\frac{10+11+12+20+25+30}{6} = \frac{108}{6} = \mathbf{18}$

Now, Update the cluster centroid and calculate Euclidean distance:

K	M1 = 3	M2 = 18	Cluster Assignment
2	**|3 – 2| = 1**	|18 – 2| = 16	K1
3	**|3 – 3| = 0**	|18 – 3| = 15	K1
4	**|3 – 4| = 1**	|18 – 4| = 14	K1
10	**|3 – 10| = 7**	|18 – 10| = 8	K1
11	|3 – 11| = 8	**|18 – 11| = 7**	K2
12	|3 – 12| = 9	**|18 – 12| = 6**	K2
20	|3 – 20| = 17	**|18 – 20| = 2**	K2
25	|3 – 25| = 22	**|18 – 25| = 7**	K2
26	|3 – 30| = 27	**|18 – 30| = 12**	K2

M1 = 3 K1 = {2, 3, 4, 10}	**M2 = 18** K2 = {11, 12, 20, 25, 30}

Now, Calculate Mean value (Centroid)

M1 = $\frac{2+3+4+10}{4} = \frac{19}{4} = \mathbf{4.75}$	**M2** = $\frac{11+12+20+25+30}{5} = \frac{98}{5} = \mathbf{19.6}$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Now, Update the cluster centroid and calculate Euclidean distance:

K	M1 = 4.75 = 5	M2 = 19.6 = 20	Cluster Assignment
2	**|5 – 2| = 3**	|20 – 2| = 18	K1
3	**|5 – 3| = 2**	|20 – 3| = 17	K1
4	**|5 – 4| = 1**	|20 – 4| = 16	K1
10	**|5 – 10| = 5**	|20 – 10| = 10	K1
11	**|5 – 11| = 6**	|20 – 11| = 9	K1
12	**|5 – 12| = 7**	|20 – 12| = 8	K1
20	|5 – 20| = 15	**|20 – 20| = 0**	K2
25	|5 – 25| = 20	**|20 – 25| = 5**	K2
26	|5 – 30| = 25	**|20 – 30| = 10**	K2

M1 = 5 K1 = {2, 3, 4, 10, 11, 12}	**M2 = 20** K2 = {20, 25, 30}

Now, Calculate Mean value (Centroid)

M1 = $\frac{2+3+4+10+11+12}{6} = \frac{42}{6} = \mathbf{7}$	**M2** = $\frac{20+25+30}{3} = \frac{75}{3} = \mathbf{25}$

Now, Update the cluster centroid and calculate Euclidean distance:

K	M1 = 7	M2 = 25	Cluster Assignment
2	**|7 – 2| = 5**	|25 – 2| = 23	K1
3	**|7 – 3| = 4**	|25 – 3| = 22	K1
4	**|7 – 4| = 3**	|25 – 4| = 21	K1
10	**|7 – 10| = 3**	|25 – 10| = 15	K1
11	**|7 – 11| = 4**	|25 – 11| = 14	K1
12	**|7 – 12| = 5**	|25 – 12| = 13	K1
20	|7 – 20| = 13	**|25 – 20| = 5**	K2
25	|7 – 25| = 18	**|25 – 25| = 0**	K2
26	|7 – 30| = 23	**|25 – 30| = 5**	K2

M1 = 5 K1 = {2, 3, 4, 10, 11, 12}	**M2 = 20** K2 = {20, 25, 30}

Now, Calculate Mean value (Centroid)

M1 = $\frac{2+3+4+10+11+12}{6} = \frac{42}{6} = \mathbf{7}$	**M2** = $\frac{20+25+30}{3} = \frac{75}{3} = \mathbf{25}$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Thus, we are getting the same mean value (Centroid). So we have to stop.

Therefore, new cluster is:

K1 = {2, 3, 4, 10, 11, 12}

K2 = {20, 25, 30}

----------X----------X----------

[2019 Spring 5.a]

What is Machine Learning and Why is Machine Learning important? Discuss K-Means Clustering for following dataset with K=2.

Staff ID	Age	No. of Leaves
1	21	3
2	39	7
3	43	6
4	58	12
5	32	15

DEEPAK BHATTA KAJI

Ans: Machine Learning is a field of computer science which gives the computer the ability to learn without being explicitly programmed. Machine Learnings main focus is to provide algorithms which can be trained to perform a task.

It is closely related to the field of computational statistics as well as mathematical optimization. It contains multiple methods like Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning which each has their own use cases and algorithms.

Machine learning (ML) is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Machine Algorithm types:

Machine learning algorithms can be organized based on the desired outcome of the algorithm or the type of input available during training the machine.

1. Supervised learning algorithms are trained on labeled examples, i.e., input where the desired output is known.

2. Unsupervised learning algorithms operate on unlabeled examples, i.e., input where the desired output is unknown.

3. Semi-supervised learning combines both labeled and unlabeled examples to generate an appropriate function or classifier.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

4. Reinforcement learning is concerned with how intelligent agents ought to act in an environment to maximize some notion of reward from sequence of actions.

Machine Learning is important because of following points:

- Some tasks cannot be defined well, except by examples (e.g. recognizing people)
- Relationship and correlations can be hidden within large amounts of data. Machine learning may be able to find these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used.
- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans (e.g. Medical diagnostic).
- Environment changes over time.
- New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design system "by hand".

Solⁿ: Given that, K=2

Let, Age=X and Nos of Leaves = Y

Assume that: Initial mean value (centroid) from the given data sets be,

Staff ID	Age (X)	No. of Leaves (Y)
1	21	3
2	39	7
3	43	6
4	58	12
5	32	15

Cluster	Individual	Mean Value / Centroid
K1 (Group 1)	1	(21, 3)
K2 (Group 2)	2	(39, 7)

The remaining individual are now examined in sequence and allocated to the cluster to which they are closest in terms of Euclidean distance to the cluster mean. The mean value (Centroid) is re-calculated each time a new member is added.

We know, Euclidean distance

$$d = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

X	Y	K1 = (21, 3) $\sqrt{(X-21)^2+(Y-3)^2}$	K2 = (39, 7) $\sqrt{(X-39)^2+(Y-7)^2}$	Cluster Assignment
21	3	**0**	18.4	K1
39	7	18.4	**0**	K2
43	6	22.2	**4.1**	K2
58	12	38.1	**19.6**	K2
32	15	16.3	**10.6**	K2

Here how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(21, 3)}

K2 = {(39, 7), (43, 6), (58, 12), (32, 15)}

Now, Calculate Mean value (Centroid)

K1 = (X, Y) = $\left(\frac{X_1}{1}\right), \left(\frac{Y_1}{1}\right)$

$= \left(\frac{21}{1}\right), \left(\frac{3}{1}\right)$

K1 = 21, 3

K2 = (X, Y) = $\left(\frac{X_1+X_2+X_3+X_4}{4}\right), \left(\frac{Y_1+Y_2+Y_3+Y_4}{4}\right)$

$= \left(\frac{39+43+58+32}{4}\right), \left(\frac{7+6+12+15}{4}\right)$

$= \left(\frac{172}{4}\right), \left(\frac{40}{4}\right)$

K2 = 43, 10

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Now, Update the cluster centroid and calculate Euclidean distance:

X	Y	K1 = (21, 3) $\sqrt{(X-21)^2+(Y-3)^2}$	K2 = (43, 10) $\sqrt{(X-43)^2+(Y-10)^2}$	Cluster Assignment
21	3	**0**	23.1	K1
39	7	18.4	**5**	K2
43	6	22.2	**4**	K2
58	12	38.1	**15.1**	K2
32	15	16.3	**12.1**	K2

Here how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(21, 3)}

K2 = {(39, 7), (43, 6), (58, 12), (32, 15)}

Now, Calculate Mean value (Centroid)

K1 = (X, Y) = $\left(\frac{X_1}{1}\right), \left(\frac{Y_1}{1}\right)$

$= \left(\frac{21}{1}\right), \left(\frac{3}{1}\right)$

K1 = 21, 3

K2 = (A, B) = $\left(\frac{X_1+X_2+X_3+X_4}{4}\right), \left(\frac{Y_1+Y_2+Y_3+Y_4}{4}\right)$

$= \left(\frac{39+43+58+32}{4}\right), \left(\frac{7+6+12+15}{4}\right)$

$= \left(\frac{172}{4}\right), \left(\frac{40}{4}\right)$

K2 = 43, 10

Thus, The Final Cluster is:

K1 = {(21, 3)}

K2 = {(39, 7), (43, 6), (58, 12), (32, 15)}

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Model Question 1:

Apply K-means clustering for the following data sets for two clusters. Tabulate all the arguments.

Sample No.	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

Sol^n: Given, 1 and 4 as initial cluster center.

Sample No.	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

Cluster	Individual	Mean Value / Centroid
K1 (Group 1)	1	(185, 72)
K2 (Group 2)	2	(170, 56)

The remaining individual are now examined in sequence and allocated to the cluster to which they are closest in terms of Euclidean distance to the cluster mean. The mean value (Centroid) is re-calculated each time a new member is added.

We know, Euclidean distance

$$d = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

X	Y	K1 = (185, 72) $\sqrt{(X-185)^2+(Y-72)^2}$	K2 = (170, 56) $\sqrt{(X-170)^2+(Y-56)^2}$	Cluster Assignment
185	72	**0**	21.93	K1
170	56	21.93	**0**	K2
168	60	20.80	**4.47**	K2
179	68	**7.21**	15	K1
182	72	**3**	20	K1
188	77	**5.83**	27.65	K1

Here how we are picking cluster is that look the minimum value by comparing K1 and K2,

if (K1<K2) then
    assign K1
else
    assign K2

K1 = {(185, 72), (179, 68), (182, 72), (188, 77)}

K2 = {(170, 56), (168, 60)}

Now, Calculate Mean value (Centroid)

$$K1 = (X, Y) = \left(\frac{X_1+X_2+X_3+X_4}{4}\right), \left(\frac{Y_1+Y_2+Y_3+Y_4}{4}\right)$$

$$= \left(\frac{185+179182+188}{4}\right), \left(\frac{72+68+72+77}{4}\right)$$

$$= \left(\frac{734}{4}\right), \left(\frac{289}{4}\right)$$

K1 = 183.5, 72.25

$$K2 = (X, Y) = \left(\frac{X_1+X_2}{2}\right), \left(\frac{Y_1+Y_2}{2}\right)$$

$$= \left(\frac{170+168}{2}\right), \left(\frac{56+60}{2}\right)$$

$$= \left(\frac{338}{2}\right), \left(\frac{116}{2}\right)$$

K2 = 169, 58

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Now, Update the cluster centroid and calculate Euclidean distance:

X	Y	K1 = (183.5, 72.25) $\sqrt{(X-183.5)^2+(Y-72.25)^2}$	K2 = (169, 58) $\sqrt{(X-169)^2+(Y-58)^2}$	Cluster Assignment
185	72	**1.52**	21.26	K1
170	56	21.12	**2.23**	K2
168	60	19.75	**2.23**	K2
179	68	**6.189**	14.14	K1
182	72	**1.52**	19.10	K1
188	77	**6.54**	26.87	K1

K1 = {(185, 72), (179, 68), (182, 72), (188, 77)}

K2 = {(170, 56), (168, 60)}

Now, Calculate Mean value (Centroid)

K1 = (X, Y) = $\left(\frac{X_1+X_2+X_3+X_4}{4}\right), \left(\frac{Y_1+Y_2+Y_3+Y_4}{4}\right)$

$= \left(\frac{185+179182+188}{4}\right), \left(\frac{72+68+72+77}{4}\right)$

$= \left(\frac{734}{4}\right), \left(\frac{289}{4}\right)$

K1 = 183.5, 72.25

K2 = (X, Y) = $\left(\frac{X_1+X_2}{2}\right), \left(\frac{Y_1+Y_2}{2}\right)$

$= \left(\frac{170+168}{2}\right), \left(\frac{56+60}{2}\right)$

$= \left(\frac{338}{2}\right), \left(\frac{116}{2}\right)$

K2 = 169, 58

Thus, we are getting the same mean value (Centroid). So, we have to stop.

Therefore, new cluster is:

K1 = {(185, 72), (179, 68), (182, 72), (188, 77)}

K2 = {(170, 56), (168, 60)}

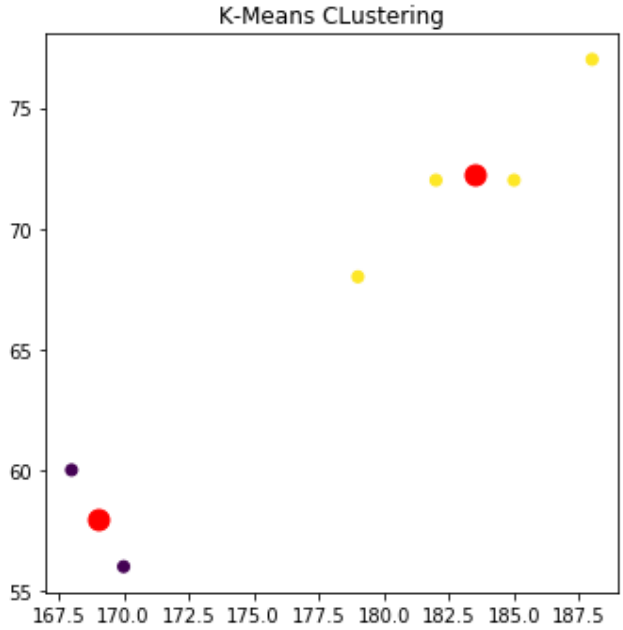
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

----------X----------X----------

Python:

```
In [1]: import numpy as np
        from sklearn.cluster import KMeans

In [2]: X = np.array([[185, 72], [170, 56], [168, 60],[179, 68], [182, 72], [188, 77]])

In [3]: X_kmeans = KMeans(n_clusters=2).fit(X)

In [4]: X_kmeans.labels_

Out[4]: array([0, 1, 1, 0, 0, 0])

In [52]: #X_kmeans.predict([[150, 120], [12, 3]])

In [5]: X_kmeans.cluster_centers_

Out[5]: array([[183.5 ,  72.25],
               [169.  ,  58.  ]])
```

```
In [7]: import matplotlib.pyplot as plt

        plt.figure(figsize=(12, 12))

        X_pred = KMeans(n_clusters=2).fit_predict(X)

        plt.subplot(221)
        plt.scatter(X[:, 0], X[:, 1], c=X_pred)
        plt.scatter(X_kmeans.cluster_centers_[:,0],X_kmeans.cluster_centers_[:,1],s=120,c='red')
        plt.title("K-Means CLustering")
        plt.show()
```



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Try Yourself

(2, 3), (5, 6), (8, 7), (1, 4), (2, 2), (6, 7), (3, 4), (8, 6) where k=2

Sol^n:

Hint:

Assume the mean value (centroid) from the given data sets,

K1 = (2, 3)

K2 = (5, 6)

DEEPAK BHATTA KAJI

THE END

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Unit 6 9 hrs.

Applications of AI

6.1 Neural Network
- **6.1.1 Network Structure**
- **6.1.2 Perceptron**
- **6.1.3 Adaline Network**
- **6.1.4 Multilayer Perceptron, Back Propagation**
- **6.1.5 Hopefield Network**
- **6.1.6 Kohonen Network**

6.2 Expert System
- **6.2.1 Architecture of an Expert System**
- **6.2.2 Development of Expert System**

6.3 Natural Language Processing
- **6.3.1 Levels of Analysis: Phonetic, Syntactic, Semantic, Pragmatic**

6.4 Introduction to Machine Vision

6.5 Current trends and the future

DEEPAK BHATTA KAJI

6.1 Neural Network

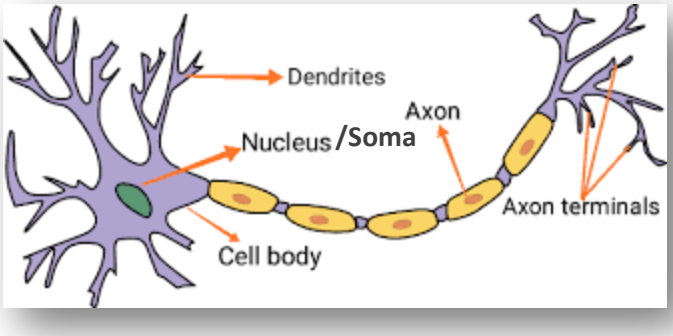
Biological Neural Network (BNN):

A neural network can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called neurons.

The human brain incorporates nearly 10 billion neurons and 60 trillion connections, synapses, between them. By using multiple neurons simultaneously, the brain can perform its functions much faster than the faster computers in existence today.

Each neuron has a very simple structure, but an army of such elements constitutes a tremendous processing power.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Neuron: Fundamental functional unit of all nervous system tissue.

Soma: Cell body, contain nucleus.

Dendrites: A number of fibers, input.

Axon: Single long fiber with many branches, output.

Synapse: Junction of dendrites and axon, each neuron form synapse with 10 to 1,00,000 other neurons.

Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

How does brain work

Signals are propagated from neuron to neuron by electrochemical reaction:

1. Chemical substances are released from the synapses and enter the dendrite, raising or lowering the electrical potential of the cell body.
2. When a potential reaches a threshold, an electrical pulse or action potential is sent down the axon.
3. The pulse spreads out along the branches of axon, eventually reaching synapses and releasing transmitters into the bodies of other cells.
    a. **Excitatory Synapses:** Excitatory synapses increase its likelihood.
    b. **Inhibitory Synapses:** Inhibitory synapses decrease the likelihood of the firing action potential of a cell.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Artificial Neural Network (ANN):

The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as –

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

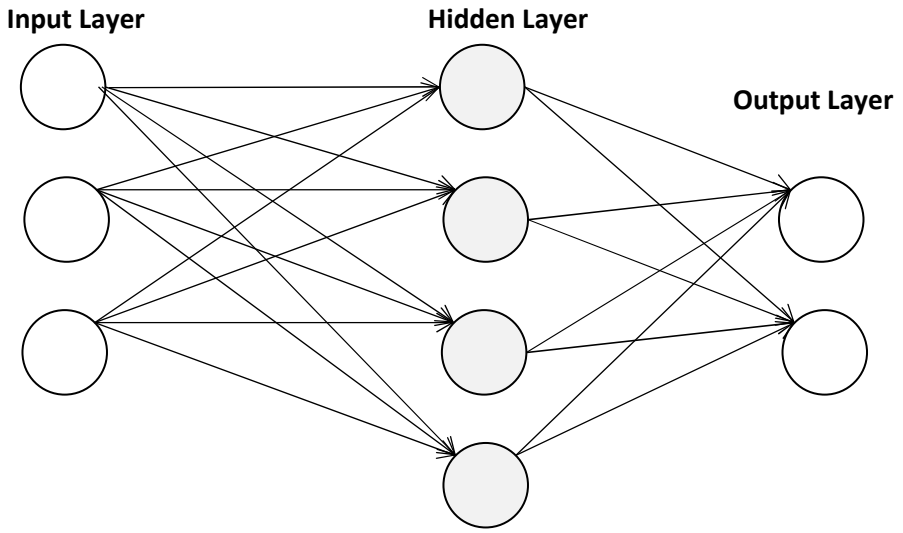
Basic Structure of ANNs

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

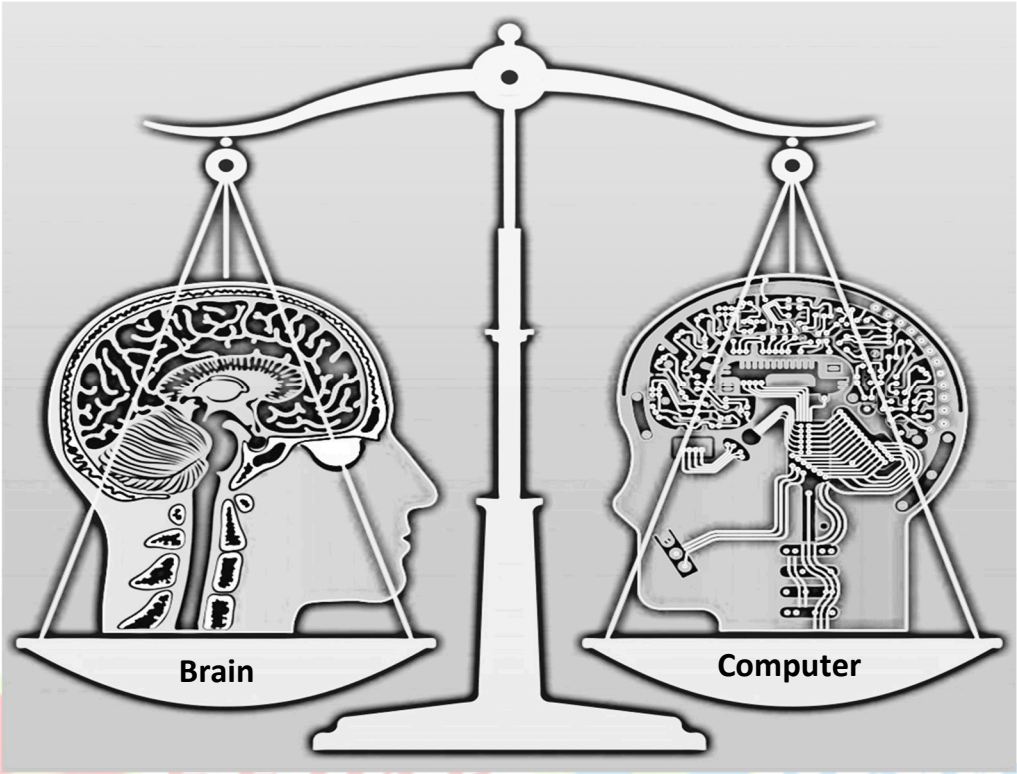
ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN –



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Brain Vs Computer



Parameters	Brain	Computer
Number of Processor	10^{11}, Neurons	10^{8}, Transistors
Processor complexity	Simple, Inaccurate	Complex, Accurate
Processor Speed	Slow (millisec) 10^{2} Hz	Fast (nanosec) 10^{12} Hz
Style of Computation	Parallel, Distributed	Serial, Centralized
Energy Use	30 W	30 W (CPU)
Learning mode	Learn from experience	Manual programming
Fault Tolerance	Has fault tolerance	No fault tolerance, it gets disrupted when interconnections are disconnected
Storage Capacity	Stores information in its interconnection or in synapse. No loss of memory	Contiguous memory locations. Loss of memory may happen sometimes.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Evolution of ANN:

Year	Neural Network	Designer	Description
1943	McCulloch and Pits Neuron	McCulloch Pitts	Logic Gates
1949	Hebb	Donald Hebb	Strength increases if neurons are active
1958-1988	Perceptron	Rosenblatt	Weights of pat can be adjusted
1960	Adaline	Widrow and Hoff	Mean Squared Error (MSE)
1972	SOM	Kohenen	Clustering
1982	Hopefield	John Hopefield	Associative Memory Nets
1986	Back - Propagation	Rumelhard	Multilayer
1987-1990	ART	Carpenter	Used for both binary and analog

ANN Introduction

An Artificial Neural Network (ANN) is a computational model. It is based on the structure and functions of bilogical neural networks. It works like the way human brain process information.

It includes a large number of connected processing units that work together to process information. They also generate meaningful results from it.

A neural network is an oriented graph. It consists of nodes which in the biological analogy represent neurons, connected by arcs. It corresponds to dendrites and synapses. Each arc associated with a weight at each node.

A neural network is a machine learning algorithm based on the model of a human neuron. The human brain consists of millions of neurons. It spends and process signals in the form of electrical and chemical signals. These neurons are connected with a special structure known as synapses. Synapses allow neurons to pass signals.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

An Artificial Neural Network is an information processing technique. It works like the way human brain processes information. ANN includes a large number of connected processing units that work together to process information. They also generate meaningful result from it.

We can apply Neural Network not only for classification, it can also apply for regression of continuous target attributes.

A neural network may contain the following 3 layers:

1. Input layer
2. Hidden layer
3. Output layer

1. **Input layer:** *The activity of the input units represents the raw information that can feed the network.* The purpose of the input layer is to receive as input the value of the explanatory attributes for each observation. Usually, the number of input nodes in an input layer is equal to the number of explanatory variables. Input layer presents the pattern to the network which communicates to one or more hidden layers.
   The nodes of input layer are passive, meaning they do not change the data. They receive a single value on their input and duplicate the value to their many outputs. From the input layer, it duplicates each value and send to all the hidden nodes.

2. **Hidden layer:** *To determine the activity of each hidden unit. The activities of the input units and the weights on the connections between the input and the hidden units. There may be one or more hidden layers.* The hidden layers apply given transformation to he input values inside the network. In this coming arc that go from other hidden nodes or from inputs nodes connected to each node. It connects with going arcs to output nodes or to other hidden nodes.
   In hidden layer, the actual processing is done via a system of weighted connections. There may be one or more hidden layers. The values entering a hidden node multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

3. **Output layer:** *The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.* The hidden layer then link to an output layer. Output layer receives connections from hidden layers or from input layers. It returns an output value that corresponds to the prediction of the response variable. In classification problems, there is usually only one output node. The active nodes of the output layer combine and change the data to produce the output values.

   The ability of the neural network to provide useful data manipulation lies in the proper selection of the weights. This is different from conventional information processing.

6.1.1 Network Structure

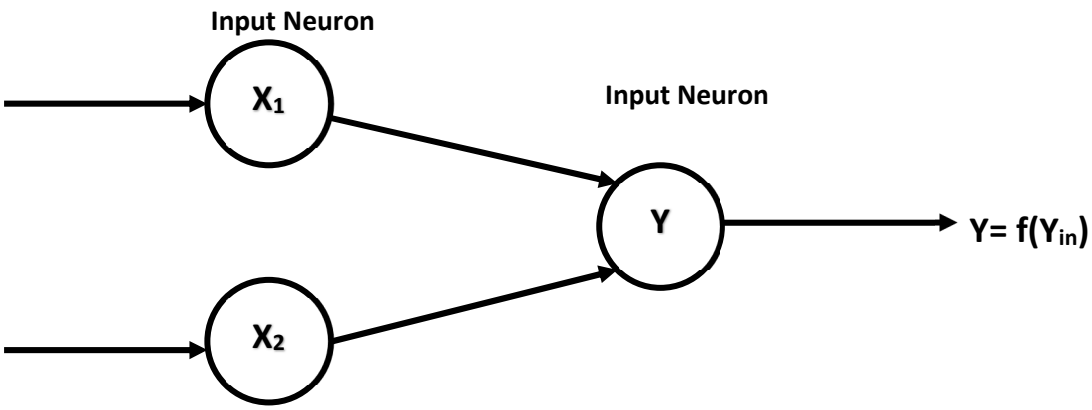
Architecture of Artificial Neural – ANN:

- Information-processing system.
- Neurons process the information.
- The signals are transmitted by means of connection links.
- The links possess an associated weight.
- The output signal is obtained by applying activations to the net input.

DEEPAK BHATTA KAJI

- ✓ ANN causes large number of highly interconnected elements called nods or neurons.
- ✓ Which usually operates in parallel and are configured in regular architecture.
- ✓ Each neuron is connected by other, by a link connection.
- ✓ Each connection is associated with weights which contains information about the input signal.
- ✓ This information is used by a neural network to solve particular problems.
- ✓ Each neuron will have their own internal state.
- ✓ The internal state is called activation or activity level of neuron.
- ✓ The activation can be transferred to one neuron to another neuron.
- ✓ One neuron can send one signal at a time.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

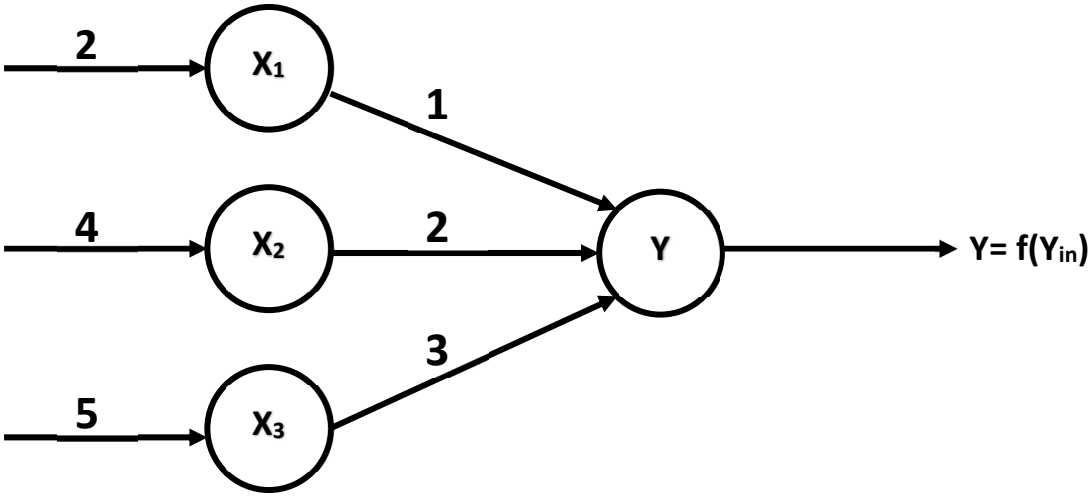


- ✓ The function which is applied over the neural net is called "Activation Function".
- ✓ **$Y_{in} = X_1W_1+X_2W_2$**
- ✓ General Equation of **Y_{in}:**
- ✓ **$Y_{in} = X_1W_1+X_2W_2+\ldots\ldots\ldots\ldots\ldots\ldots+X_nW_n =$**

$$\sum_{i=1}^{n} X_i * W_i$$

DEEPAK BHATTA

Example:



$Y_{in} = X_1W_1+X_2W_2+X_3W_3$

$Y_{in} = 2*1 + 4*2 + 5*3$

$Y_{in} = 25$ Ans.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Why develop artificial intelligence?

The purpose of building artificial intelligence model of the brain can be neuro-science, they study of the brain and the nervous system in general. It is tempting to think that by mapping the human brain in enough details, we can discover the secrets of human and animal cognition and consciousness.

Basic Model of ANN:

(i) The model's synaptic interconnection
(ii) The training or learning rules adopting for updating and adjusting the connection weights.
(iii) Their activation functions.

The arrangement of neurons to/from layer and the connection patterns formed within and between the layer is called network architecture.

A layer is formed by taking processing element and combining it within other processing element.

An ANN consists of a set of highly interconnected processing elements(neurons) such that each processing elements output is found to be connected through weights to other neurons or itself.

Basically, Neural Nets are classified into:

i. Single-layer
ii. Multi-layer

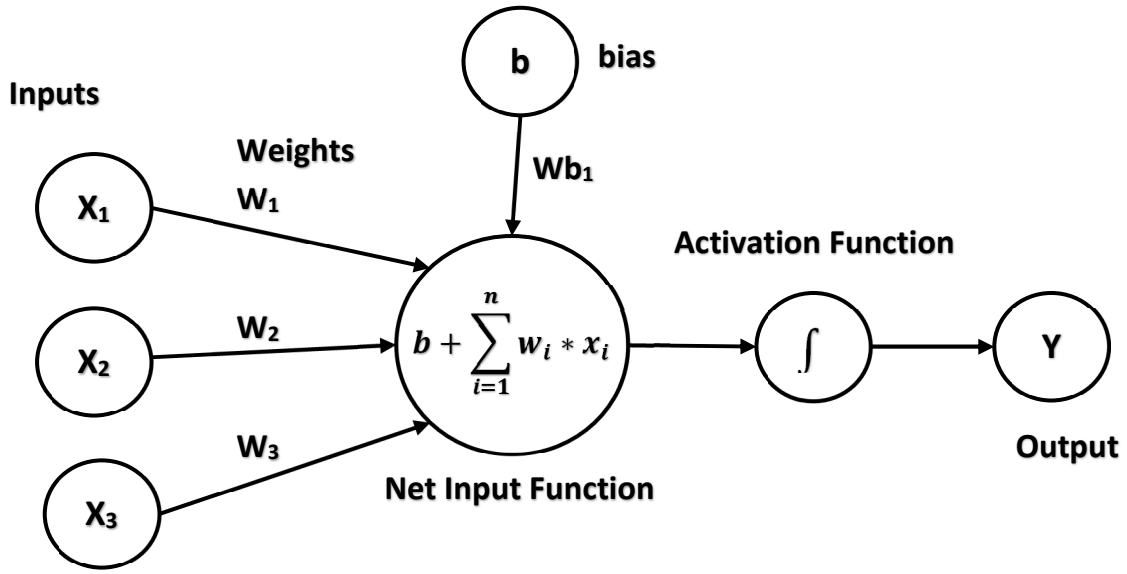
Different types of connections in ANN:

i. Single layer feed-forward network
ii. Multi-layer feed-forward network
iii. Single node with its own feedback
iv. Single layer recurrent network
v. Multi-layer recurrent network

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

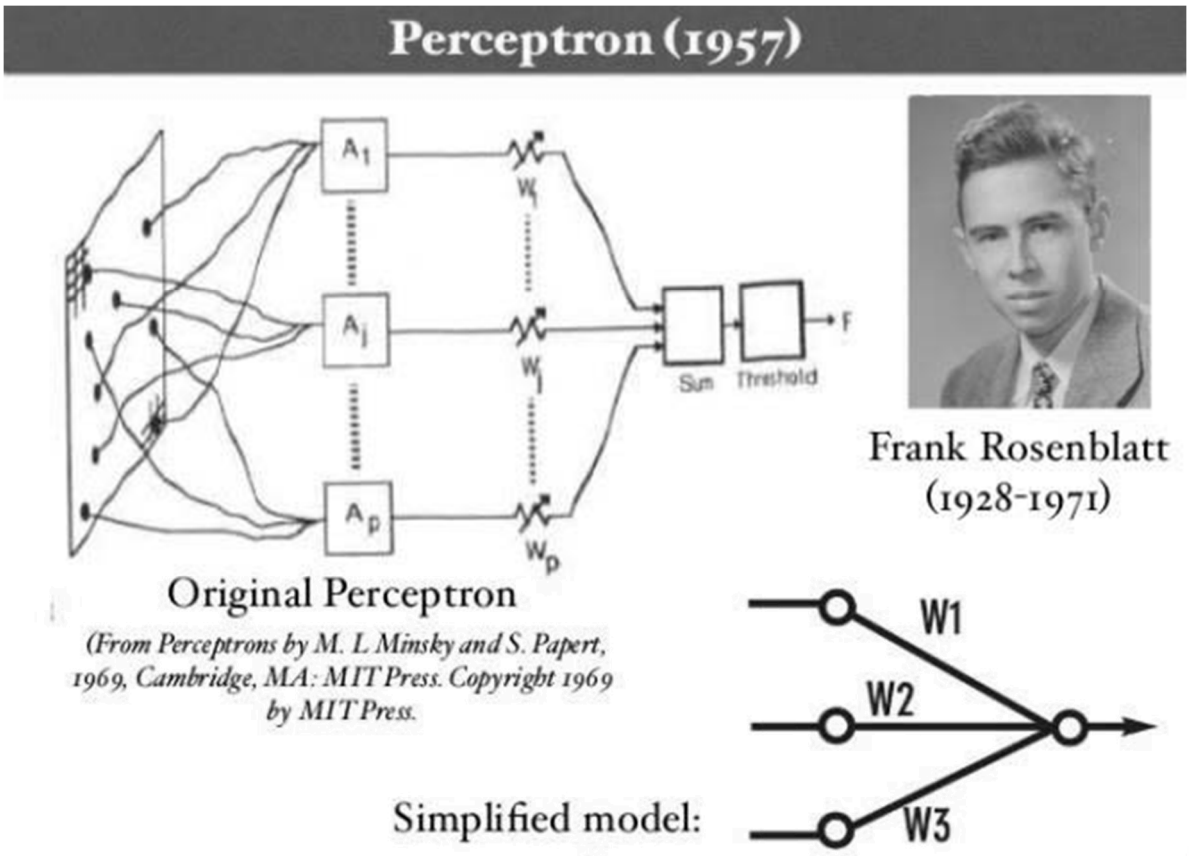
6.1.2 Perceptron

- The perceptron is the simplest form of neural network is able to classify data into two classes (Input and Output).
- It consists of single neuron with a number of adjustable weights.
- A single artificial neuron that compute its weighted input and uses a threshold activation function.
- It is also called a TLU (Threshold Logic Unit).

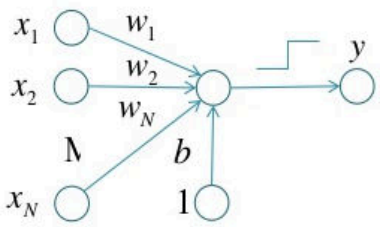


- Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.
- A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.
- There are two types of Perceptrons: Single layer and Multilayer.
- **Single layer Perceptrons** can learn only linearly separable patterns.
- **Multilayer Perceptrons** or **feedforward neural networks** with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Perceptron Original Research Paper By : Frank Rosenblatt



decision function

$\hat{f}(\bar{x}) = \text{sign}(\bar{w} \cdot \bar{x} + b)$

$\bar{x} = (x_1, x_2, \ldots, x_N)$ features of the customer $\bar{x}$

approve credit if $\sum_{i=1}^{N} w_i x_i > \text{threshold}$

deny credit if $\sum_{i=1}^{N} w_i x_i < \text{threshold}$

Frank Rosenblatt

input: $T = \{(\bar{x}_i, y_i) \mid i = \overline{1,p}\} \subset \mathbf{R}^N \times \{+1,-1\}$

$\bar{w} \leftarrow \bar{0}, b \leftarrow 0$

repeat

  for i=1 **to** p

    if $\text{sign}(\bar{w} \cdot \bar{x}_i + b) \neq y_i$ **then**

      $\bar{w} \leftarrow \bar{w} + y_i \bar{x}_i$

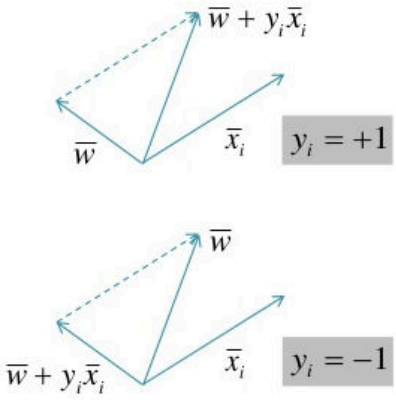
      $b \leftarrow b + y_i$

    end if

  end for

until $\text{sign}(\bar{w} \cdot \bar{x}_j + b) = y_j, \forall j = \overline{1,p}$

return $\bar{w}, b$



CORNELL AERONAUTICAL LABORATORY, INC.
BUFFALO, N. Y.

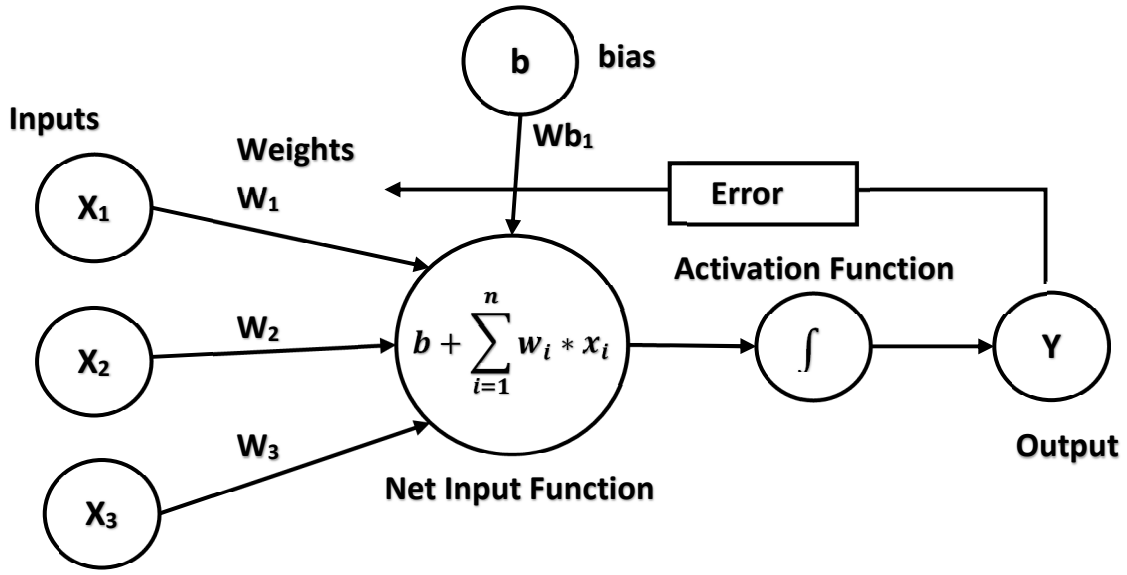
REPORT NO. 85-460-1

THE PERCEPTRON
A PERCEIVING AND RECOGNIZING AUTOMATON
(PROJECT PARA)
January, 1957

Prepared by: Frank Rosenblatt
Frank Rosenblatt,
Project Engineer

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Perceptron Learning Rule:



Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fire or not.

The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

Perceptron Function:

Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1, & if\ w.x + b > 0 \\ 0, & if\ w.x + b \leq 0 \end{cases}$$

In the equation given above:

"w" = vector of real-valued weights

"b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

"x" = vector of input x values

$$\sum_{i=1}^{n} w_i * x_i$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

"n" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

What is Logic Gate?

Logic gates are the building blocks of a digital system, especially neural network. In short, they are the electronic circuits that help in addition, choice, negation, and combination to form complex circuits.

Using the logic gates, Neural Networks can learn on their own without you having to manually code the logic. Most logic gates have two inputs and one output.

Each terminal has one of the two binary conditions, low (0) or high (1), represented by different voltage levels. The logic state of a terminal changes based on how the circuit processes data.

Based on this logic, logic gates can be categorized into seven types:

1. AND
2. NAND
3. OR
4. NOR
5. NOT
6. XOR
7. XNOR

First, we need to know that the Perceptron algorithm states that:

Prediction (y`) = 1 if Wx+b > 0 and 0 if Wx+b ≤ 0

Also, the steps in this method are very similar to how Neural Networks learn, which is as follows;

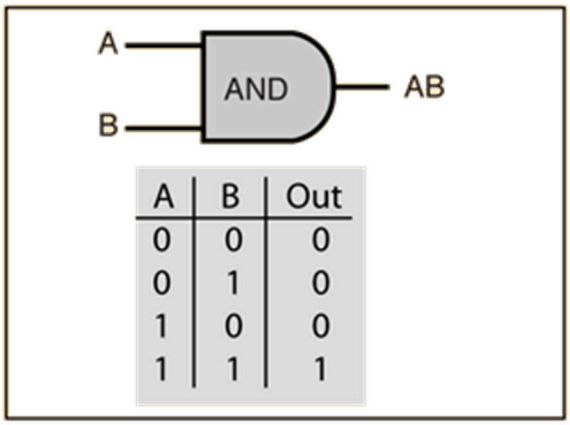
- Initialize weight values and bias
- Forward Propagate
- Check the error
- Backpropagate and Adjust weights and bias
- Repeat for all training examples

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Implement Basic Logic Gates with Perceptron:

AND Gate:

From our knowledge of logic gates, we know that an AND logic table is given by the diagram below:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

The question is, what are the weights and bias for the AND perceptron?

First, we need to understand that the output of an AND gate is 1 only if both inputs (in this case, x1 and x2) are 1.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the AND logic table (**X_1=0, X_2=0**), we get;

$$= \mathbf{0(1) + 0(1) + (-1)}$$

$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- Passing the first row of the AND logic table (**X_1=0, X_2=1**), we get;

$$= \mathbf{0(1) + 1(1) + (-1)}$$

$$= \mathbf{0 + 1 - 1 = 0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 3

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the AND logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1(1) + 0(1) + (-1)}$$

$$= \mathbf{1 + 0 - 1 = 0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 4

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the AND logic table (**X_1=1, X_2=1**), we get;

$$= \mathbf{1(1) + 1(1) + (-1)}$$

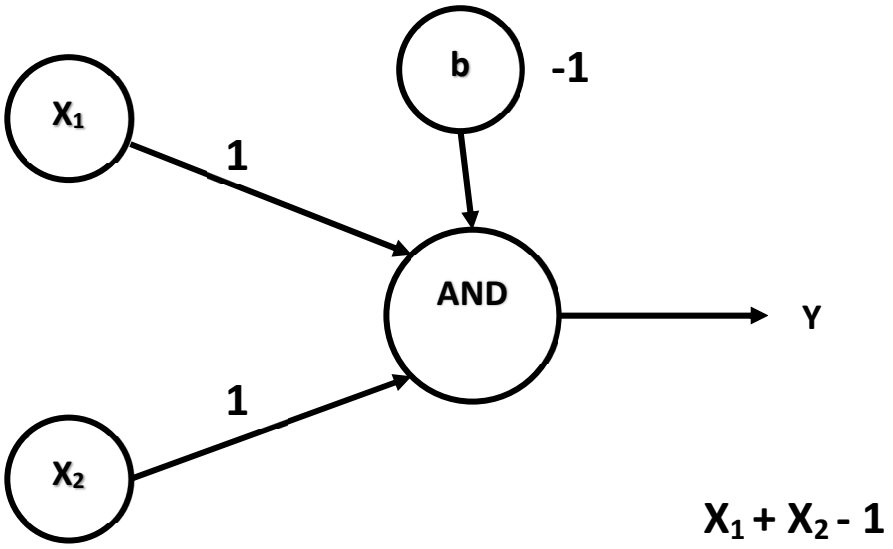
$$= \mathbf{1 + 1 - 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Therefore, we can conclude that the model to achieve an AND gate, using the Perceptron algorithm is;

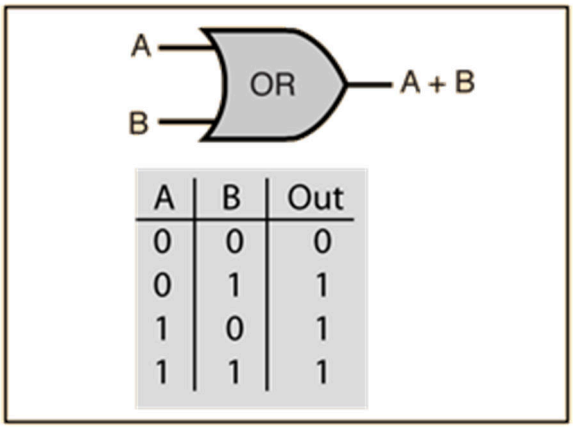
$$= \boldsymbol{X_1 + X_2 - 1}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



OR Gate:

From our knowledge of logic gates, we know that an OR logic table is given by the diagram below:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

From the diagram, the OR gate is 0 only if both inputs are 0.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the OR logic table (**X_1=0, X_2=0**), we get;

$$= \mathbf{0(1) + 0(1) + (-1)}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the OR logic table (**X_1=0, X_2=1**), we get;

$$= \mathbf{0(1) + 1(1) + (-1)}$$

$$= \mathbf{0 + 1 - 1 = 0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, and need for Backpropagation.
- So, we want values that will make inputsX_1=0 and X_2=1 give y` a value of 1. If we change W_2 to 2, we have;

$$= \boldsymbol{X_1(1) + X_2(2) + (-1)}$$

$$= \mathbf{0(1) + 1(2) + (-1)}$$

$$= \mathbf{0 + 2 - 1 = 1}$$

- From the Perceptron rule, this is correct for both row1 and row2 no any change in row1.

Row 3

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the OR logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1(1) + 0(1) + (-1)}$$

$$= \mathbf{1 + 0 - 1 = 0}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, and need for Backpropagation.
- Since it is similar to that of row 2. We can just change W_1 to 2, we have;

$$= \boldsymbol{X_1(2) + X_2(1) + (-1)}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$$= \mathbf{1(2) + 0(1) + (-1)}$$

$$= \mathbf{2 + 0 - 1 = 1}$$

- From the Perceptron rule, this is correct for both row1, row2 and row3.

Row 4

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the OR logic table (**X_1=1, X_2=1**), we get;

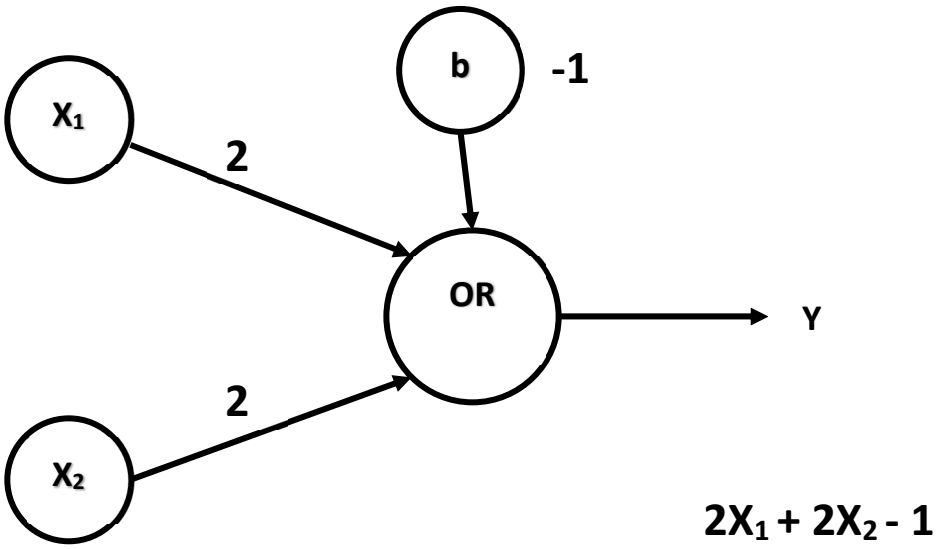
$$= \mathbf{1(1) + 1(1) + (-1)}$$

$$= \mathbf{1 + 1 - 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

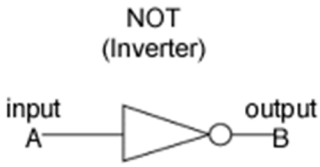
Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;

$$= \mathbf{2X_1 + 2X_2 - 1}$$



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

NOT Gate:



A	B
0	1
1	0

From the diagram, the output of a NOT gate is the inverse of a single input.

Row 1

- From W_1*X_1+b, initializing W_1 as 1 (since single input), and b as –1, we get;

$$= \boldsymbol{X_1(1) + (-1)}$$

- Passing the first row of the NOT logic table (x1=0), we get;

$$= \mathbf{0(1) + (-1)}$$
$$= \mathbf{0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`=0. This row is incorrect, as the output is 1 for the NOT gate.
- So, we want values that will make input x1=0 to give y` a value of 1. If we change b to +1, we have;

$$= \mathbf{0(1) + (+1)}$$
$$= \mathbf{0 + 1 = 1}$$

- From the Perceptron rule, this works.

Row 2

- From W_1*X_1+b, initializing W_1 as 1 (since single input), and b as +1, we get;

$$= \boldsymbol{X_1(1) + (+1)}$$

- Passing the second row of the NOT logic table (x1=1), we get;

$$= \mathbf{1(1) + (+1)}$$
$$= \mathbf{1 + 1 = 2}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

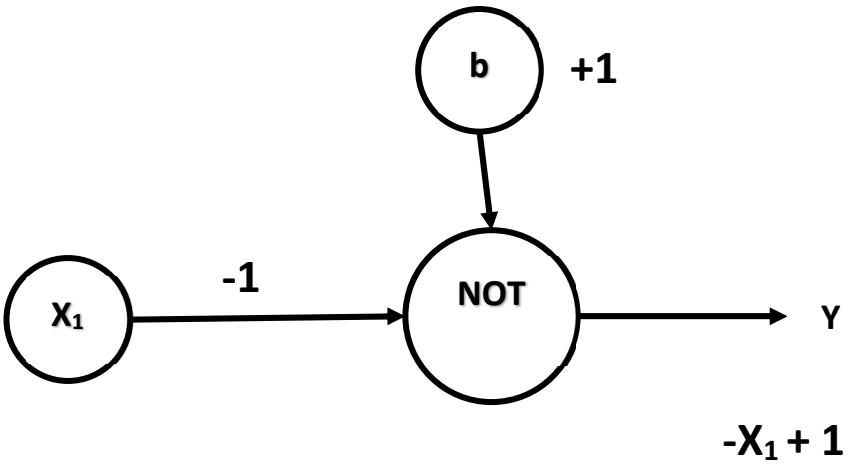
- From the Perceptron rule, if $W * X + b > 0$, then y`=1. This row is so incorrect, as the output is 0 for the NOT gate.
- So, we want values that will make input x1=1 to give y` a value of 0. If we change W_1 to –1, we have;

$$= \mathbf{-1(1) + (+1)}$$
$$= \mathbf{-1 + 1 = 0}$$

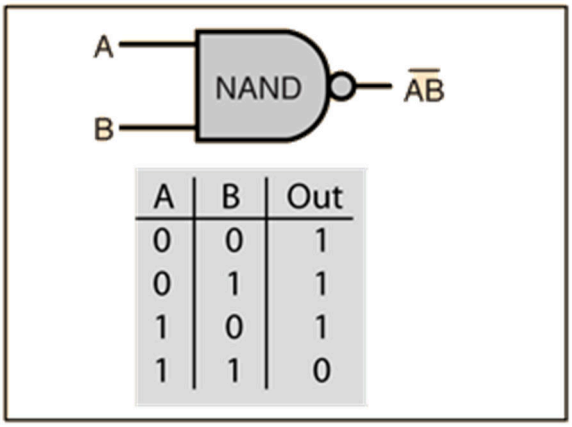
- From the Perceptron rule, if $W * X + b \leq 0$, then y`=0. Therefore, this works (for both row 1 and row 2).

Therefore, we can conclude that the model to achieve a NOT gate, using the Perceptron algorithm is;

$$= -\boldsymbol{X_1} + \mathbf{1}$$



NAND Gate:



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

From the diagram, the NAND gate is 0 only if both inputs are 1.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the NAND logic table (**X_1=0, X_2=0**), we get;
$$= \mathbf{0(1) + 0(1) + (-1)}$$

$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, as the output is 1 for the NAND gate and need for Backpropagation.

- So, we want values that will make input x1=0 and x2 = 0 to give y` a value of 1. If we change b to 1, we have;
$$= \mathbf{0(1) + 0(1) + (+1)}$$
$$= \mathbf{0 + 0 + 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct

DEEPAK BHATTA

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table (**X_1=0, X_2=1**), we get;
$$= \mathbf{0(1) + 1(1) + (+1)}$$

$$= \mathbf{0 + 1 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Row 3

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table (**X_1=1, X_2=0**), we get;
$$= \mathbf{1(1) + 0(1) + (+1)}$$

$$= \mathbf{1 + 0 + 1 = 2}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct, and no need for Backpropagation.

Row 4

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NAND logic table (**X_1=1, X_2=1**), we get;

$$= \mathbf{1(1) + 1(1) + (+1)}$$

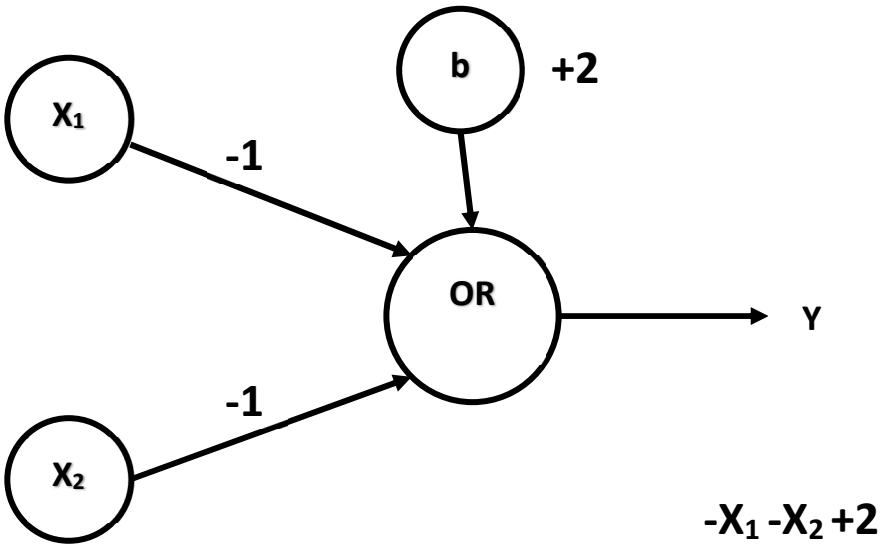
$$= \mathbf{1 + 1 + 1 = 3}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. It is incorrect.
- This is not the expected output, as the output is 0 for a NAND combination of X_1=1 and X_2=1.
- Changing values of W_1 and W_2 to -1, and value of b to 2, we get;

$$= \mathbf{1(-1) + 1(-1) + (+2)}$$
$$= \mathbf{-1 - 1 + 2 = 0}$$

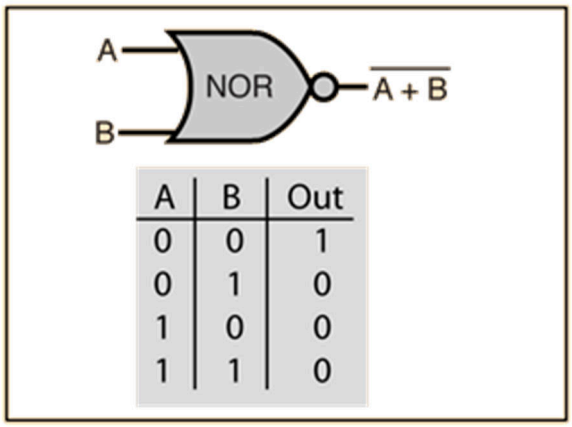
- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct.
- It works for all rows.
- Therefore, we can conclude that the model to achieve a NAND gate, using the Perceptron algorithm is;

$$= \boldsymbol{-X_1 - X_2 + 2}$$



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

NOR Gate:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

From the diagram, the NOR gate is 1 only if both inputs are 0.

Row 1

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as –1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (-1)}$$

- Passing the first row of the NOR logic table (**X_1=0, X_2=0**), we get;
$$= \mathbf{0(1) + 0(1) + (-1)}$$
$$= \mathbf{0 + 0 - 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is incorrect, as the output is 1 for the NOR gate and need for Backpropagation.

- So, we want values that will make input x1=0 and x2 = 0 to give y` a value of 1. If we change b to 1, we have;
$$= \mathbf{0(1) + 0(1) + (+1)}$$
$$= \mathbf{0 + 0 + 1 = 1}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is correct.

Row 2

- From $W_1*X_1+W_2*X_2+b$, initializing W_1, W_2, as 1 and b as +1, we get;
$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=0, X_2=1**), we get;
$$= \mathbf{0(1) + 1(1) + (+1)}$$
$$= \mathbf{0 + 1 + 1 = 2}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is incorrect, and need for Backpropagation.

- So, we want values that will make input X_1=0 and X_2 = 1 to give y` a value of 0. If we change W_2 to –1, we have;

$$\mathbf{0(1) + 1(-1) + (+1)}$$
$$\mathbf{= 0 - 1 + 1 = 0}$$

- From the Perceptron rule, this is valid for both row 1 and row 2.

Row 3

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=1, X_2=0**), we get;

$$= \mathbf{1(1) + 0(1) + (+1)}$$

$$= \mathbf{1 + 0 + 1 = 2}$$

- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. Therefore, this row is incorrect, and need for Backpropagation.

- So, we want values that will make input X_1=0 and X_2 = 1 to give y` a value of 0. If we change W_1 to –1, we have;

$$\mathbf{1(-1) + 0(1) + (+1)}$$
$$\mathbf{= 1 - 0 + 1 = 0}$$

- From the Perceptron rule, this is valid for both row 1, row 2 and row 3.

Row 4

- From W_1*X_1+W_2*X_2+b, initializing W_1, W_2, as 1 and b as +1, we get;

$$= \boldsymbol{X_1(1) + X_2(1) + (+1)}$$

- Passing the first row of the NOR logic table (**X_1=1, X_2=1**), we get;

$$= \mathbf{1(1) + 1(1) + (+1)}$$

$$= \mathbf{1 + 1 + 1 = 3}$$

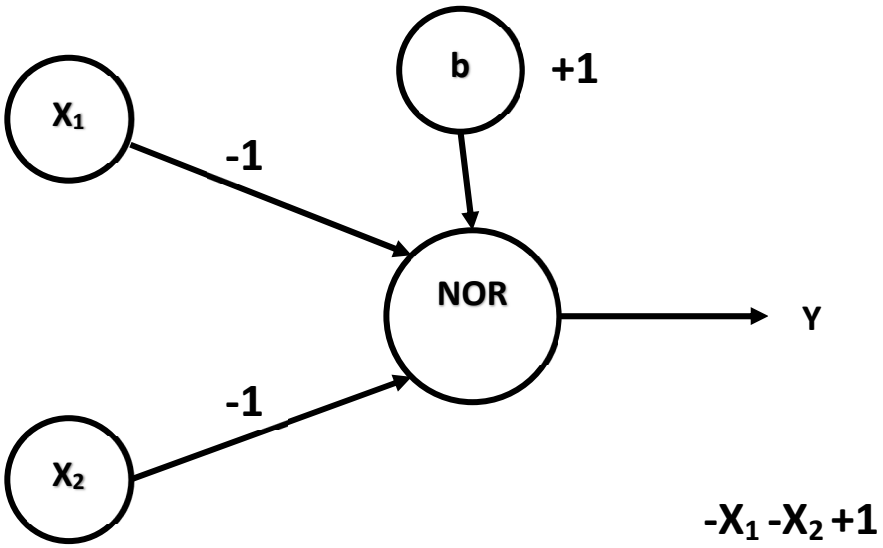
- From the Perceptron rule, if $W * X + b > 0$, then y`= 1. It is incorrect.
- This is not the expected output, as the output is 0 for a NOR combination of X_1=1 and X_2=1.
- Changing values of W_1 and W_2 to -1, we get;

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

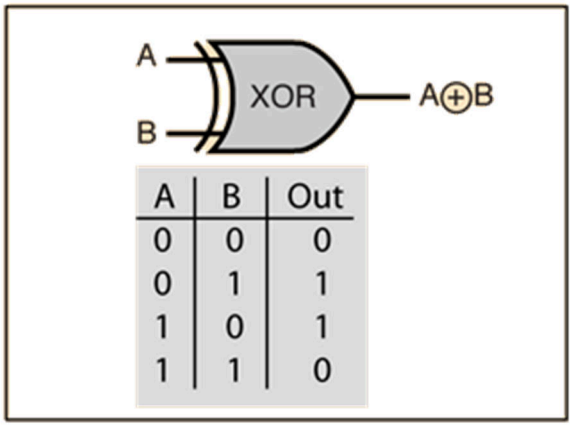
$$= \mathbf{1(-1) + 1(-1) + (+1)}$$
$$= \mathbf{-1 - 1 + 1 = -1}$$

- From the Perceptron rule, if $W * X + b \leq 0$, then y`= 0. Therefore, this row is correct.
- It works for all rows.
- Therefore, we can conclude that the model to achieve a NOR gate, using the Perceptron algorithm is;

$$= -\boldsymbol{X_1} - \boldsymbol{X_2} + \mathbf{1}$$



XOR Gate:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

If a specific type of gate is not available, a circuit that implements the same function can be constructed from other available gates.

A circuit implementing an XOR function can be trivially constructed from an XNOR gate followed by a NOT gate.

If we consider the expression, $(\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B})$ we can construct an XOR gate circuit directly using AND, OR and NOT gates. However, this approach requires five gates of three different kinds.

As alternative, if different gates are available we can apply Boolean algebra to transform $(\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B}) \equiv (\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}})$ as stated above, and apply de Morgan's Law to the last term to get $(\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}.\ \boldsymbol{B}})$ which can be implemented using only three gates

How this expression $(\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}})$ is transformed?

The Boolean representation of an XOR gate is;

$$= (\boldsymbol{A}.\overline{\boldsymbol{B}}) + (\overline{\boldsymbol{A}}.\boldsymbol{B})$$

We first simplify the Boolean expression

$= \boldsymbol{X_1}\overline{\boldsymbol{X_2}} + \overline{\boldsymbol{X_1}}\boldsymbol{X_2}$ **(Apply De-Morgan's Law)**

$= \overline{\boldsymbol{A}.\overline{\boldsymbol{B}}}.\ \overline{\overline{\boldsymbol{A}}.\boldsymbol{B}}$ **(Apply De-Morgan's Law)**

$= \overline{\boldsymbol{A}} + \overline{\overline{\boldsymbol{B}}}.\ \overline{\overline{\boldsymbol{A}}} + \overline{\boldsymbol{B}}$

$= (\overline{\boldsymbol{A}} + \boldsymbol{B}).\ (\boldsymbol{A} + \overline{\boldsymbol{B}})$ **(Apply Distributive Law)**

$= \overline{\boldsymbol{A}}(\boldsymbol{A} + \overline{\boldsymbol{B}}) + \boldsymbol{B}(\boldsymbol{A} + \overline{\boldsymbol{B}})$ **(Apply Distributive Law)**

$= \overline{\boldsymbol{A}}.\boldsymbol{A} + \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A} + \boldsymbol{B}.\overline{\boldsymbol{B}}$ **(Apply Complement Law)** $\overline{\boldsymbol{A}}.\boldsymbol{A} = \boldsymbol{0}$

$= \boldsymbol{0} + \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A} + \boldsymbol{0}$

$= \overline{\boldsymbol{A}}.\overline{\boldsymbol{B}} + \boldsymbol{B}.\boldsymbol{A}$ **(Apply De-Morgan's Law)**

$= \overline{(\overline{\boldsymbol{A}}.\overline{\boldsymbol{B}}).(\boldsymbol{B}.\boldsymbol{A})}$

$= (\overline{\overline{\boldsymbol{A}}} + \overline{\overline{\boldsymbol{B}}}).(\overline{\boldsymbol{B}.\boldsymbol{A}})$ **(Apply Commutative Law) B.A = A.B**

$= (\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}.\boldsymbol{B}})$

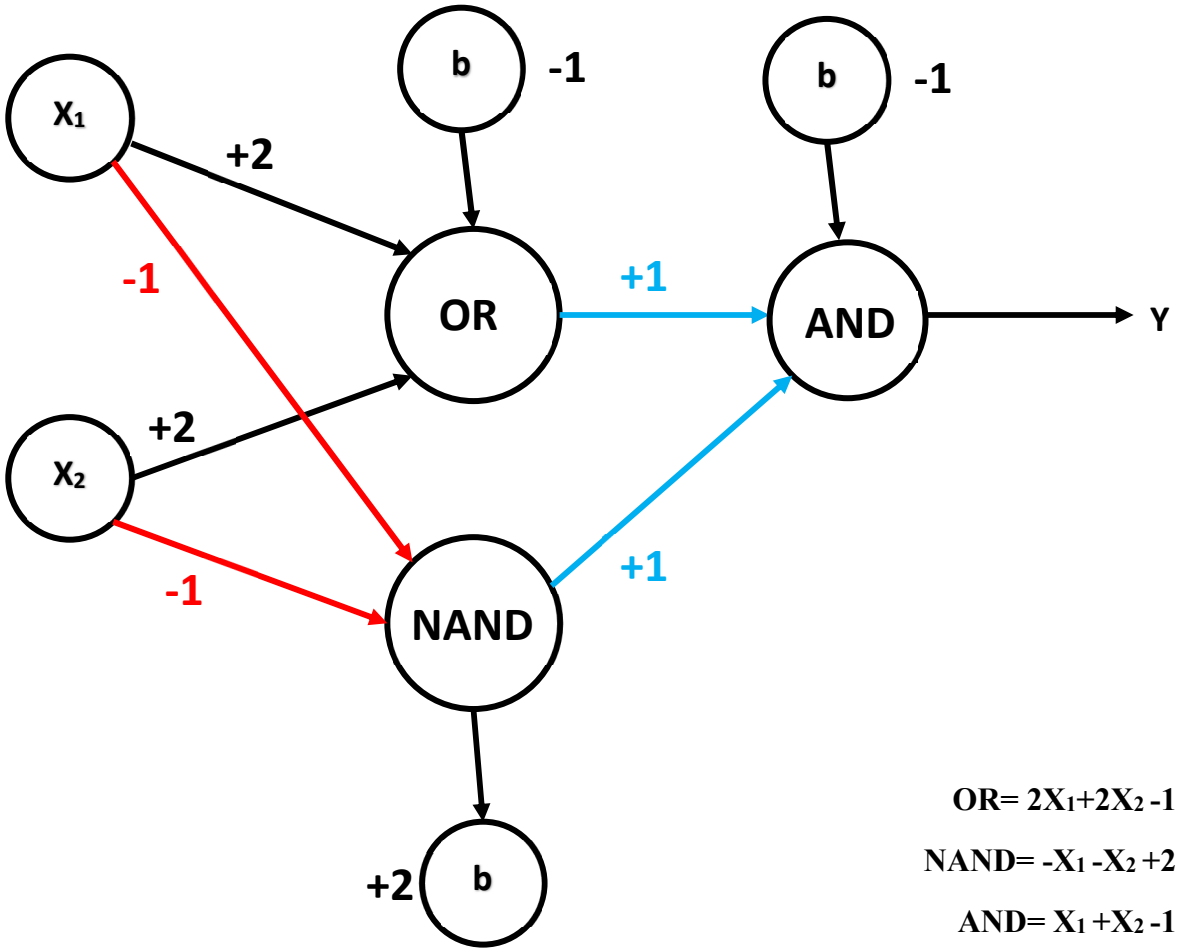
$= X_1\overline{X_2} + \overline{X_1}X_2$ **So, this can be equivalent as,**

$= (X_1 + X_2) \,.\, (\overline{X_1 \,.\, X_2})$

From the simplified expression, we can say that the XOR gate consists of an OR gate $(X_1 + X_2)$, a NAND gate $(\overline{X_1 \,.\, X_2})$ and an AND gate $((X_1 + X_2) \,.\, (\overline{X_1 \,.\, X_2}))$

This means we will have to combine 3 perceptrons:

- OR ($2X_1 + 2X_2 - 1$)
- NAND ($-X_1 - X_2 + 1$)
- AND ($X_1 + X_2 - 1$)



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Row 1 (X_1= 0, X_2= 0)		**OR(X_1)= 0, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*0+2*0-1= -1	**-0-0+2= +2**	**0+1-1= 0**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XOR gate (0 & 0 is 0)		

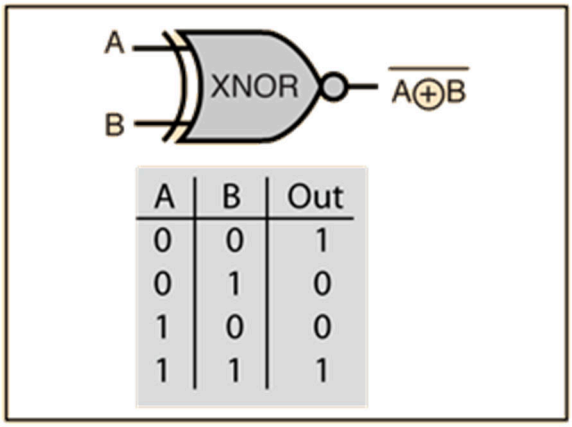
Row 2 (X_1= 0, X_2= 1)		**OR(X_1)= 1, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*0+2*1-1= +1	**-0-1+2= +1**	**1+1-1= 1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XOR gate (0 & 1 is 1)		

Row 3 (X_1= 1, X_2= 0)		**OR(X_1)= 1, NAND(X_2)= 1**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*1+2*0-1= +1	**-1-0+2= +1**	**1+1-1= 1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XOR gate (1 & 0 is 1)		

Row 4 (X_1= 1, X_2= 1)		**OR(X_1)= 1, NAND(X_2)= 0**
OR ($2X_1+2X_2-1$)	**NAND ($-X_1-X_2+2$)**	**AND (X_1+X_2-1)**
2*1+2*1-1= +3	**-1-1+2= 0**	**1+0-1= 0**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XOR gate (1 & 1 is 0)		

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

XNOR Gate:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Now that we are done with the necessary basic logic gates, we can combine them to give an XNOR gate. This XNOR gate is the complement or inverse of XOR gate.

Means, XOR= $(\boldsymbol{A}+\boldsymbol{B}).(\overline{\boldsymbol{A}.\boldsymbol{B}})$ **i.e.** $\boldsymbol{XNOR} = \overline{\boldsymbol{XOR}}$

Then, XNOR= $\overline{(A+B).(\overline{A.B})}$

$= (\overline{\boldsymbol{A}+\boldsymbol{B}}) + (\overline{\overline{\boldsymbol{A}.\boldsymbol{B}}})$

The Boolean representation of an XNOR gate is;

$= (\overline{\boldsymbol{A}}.\ \overline{\boldsymbol{B}}) + (\boldsymbol{A}.\ \boldsymbol{B})$

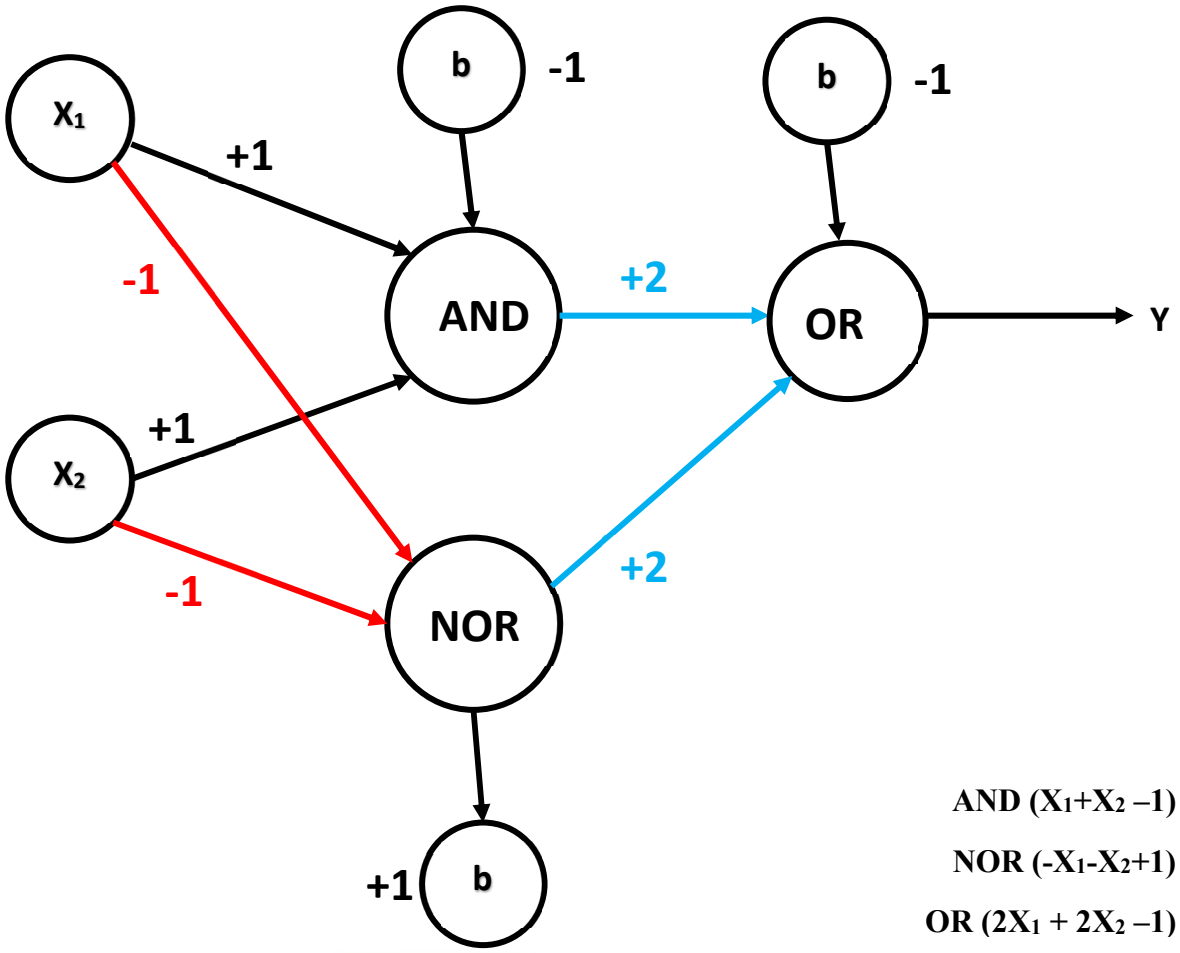
$= (\overline{\boldsymbol{X_1}}.\ \overline{\boldsymbol{X_2}}) + (\boldsymbol{X_1}.\boldsymbol{X_2})$

From the expression, we can say that the XNOR gate consists of an AND gate $(\boldsymbol{X_1}.\boldsymbol{X_2})$, a NOR gate $(\overline{\boldsymbol{X_1}}.\ \overline{\boldsymbol{X_2}})$, and an OR gate $((\overline{\boldsymbol{X_1}}.\ \overline{\boldsymbol{X_2}}) + (\boldsymbol{X_1}.\boldsymbol{X_2})$).

This means we will have to combine 3 perceptrons:

- **AND (X_1+X_2 –1)**
- **NOR (-X_1-X_2+1)**
- **OR (2X_1 + 2X_2 –1)**

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Row 1 (X_1= 0, X_2= 0)		AND(X_1)= 0, NOR(X_2)= 1
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
0+0-1= -1	**-0-0+1= +1**	**2*0+2*1-1= 1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XNOR gate (0 & 0 is 1)		

Row 2 (X_1= 0, X_2= 1)		AND(X_1)= 0, NOR(X_2)= 0
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
0+1-1= 0	**-0-1+1= 0**	**2*0+2*0-1= -1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XNOR gate (0 & 1 is 0)		

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Row 3 (X_1= 1, X_2= 0)		**AND(X_1)= 0, NOR(X_2)= 0**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
1+0-1= 0	**-1-0+1= 0**	**2*0+2*0-1= -1**
From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**
This row is correct as the output is 0 for the XNOR gate (1 & 0 is 0)		

Row 4 (X_1= 1, X_2= 1)		**AND(X_1)= 1, NOR(X_2)= 0**
AND (X_1+X_2 –1)	**NOR (-X_1-X_2+1)**	**OR (2X_1 + 2X_2 –1)**
1+1-1= +1	**-1-1+1= -1**	**2*1+2*0-1= +1**
From the perceptron rule, if $W * X + b > 0$, then **y`= 1**	From the perceptron rule, if $W * X + b \leq 0$, then **y`= 0**	From the perceptron rule, if $W * X + b > 0$, then **y`= 1**
This row is correct as the output is 0 for the XNOR gate (1 & 1 is 1)		

DEEPAK BHATTA KAJI

CONCLUSION

In conclusion, this is just a custom method of achieving this, there are many other ways and values you could use in order to achieve logic gates using perceptrons. For example;

AND (20X_1+20X_2–30)

OR (20X_1+20X_2–10)

NOT (-20X_1+10)

This will still work.

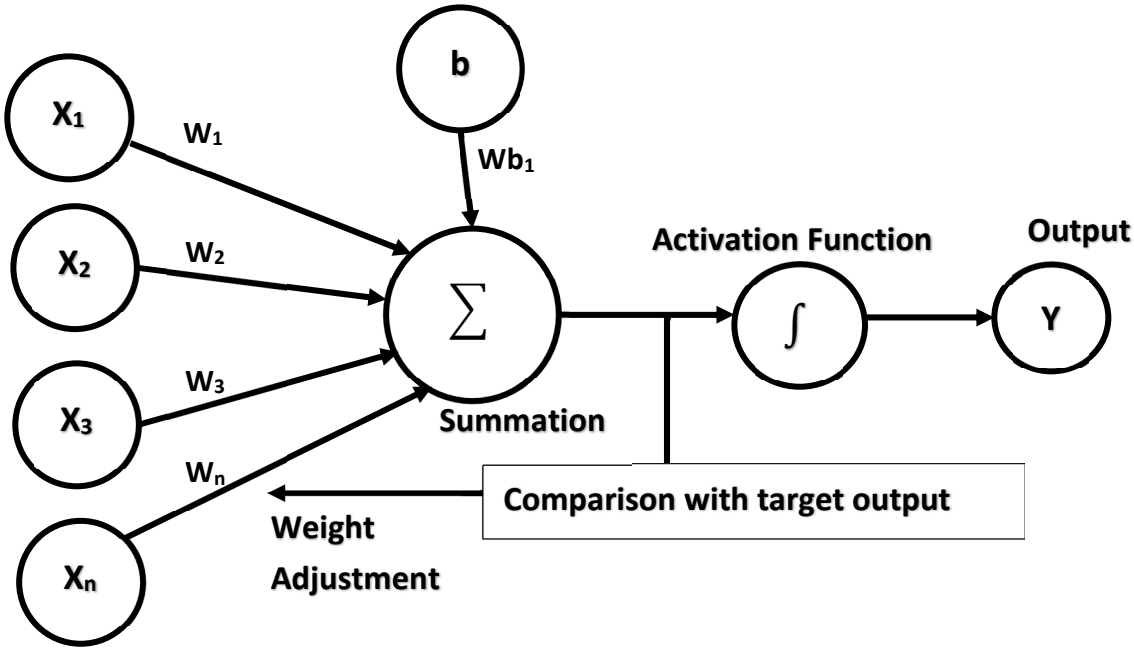
© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.1.3 Adaline Network

- ADALINE stands for **ADA**ptive **LI**near **NE**uron or **ADA**ptive **LIN**ear **E**lement, is a network having a single linear unit.
- It was developed by Widrow and Hoff in 1960.
- Some important points about Adaline are as follows: -
    - It uses bipolar(+1 or -1) activation function.
    - It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/ target output.
    - The weights and the bias are adjustable.

Architecture:

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.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Adaline Algorithm:

Step 1: Initialize the weights to small random values and select a learning rate, α

Step 2: For each input vector x, with target output t, set the inputs to S=(x-t)

Step 3: Compute the neuron inputs: $\boldsymbol{Y_{in} = b + \sum X_i W_i}$

Step 4: Use the delta rule to update the bias and weights:

$$\boldsymbol{b(new) = b(old) + \propto (t - Y_{in})}$$

$$\boldsymbol{W_i(new) = W_i(old) + \propto (t - Y_{in}) X_i}$$

Step 5: Stop if the largest weight change across all the training samples is less than a specified tolerance, otherwise cycle through the training set again.

The Learning Rate, $\propto$

- The performance of an ADALINE neuron depends heavily on the choice of the learning rate.
  - If it is too large the system will not converge
  - If it is too small the convergence will take too long
- Typically, $\propto$ is selected by trial and error
  - Typical range: $0.01 < \propto < 1.0$
  - Often start at 0.1
  - Sometimes it is suggested that:
    - $0.01 < n \propto < 1.0$
      Where n is the number of inputs

Running Adaline

- One unique feature of ADALINE is that its activation function is different for training and running
- When running ADALINE use the following:
  - Initialize the weights to those found during training
  - Compute the Net input
  - Apply the activation function

Neuron Input

$$\boldsymbol{Y_{in} = b + \sum X_i W_i}$$

Activation Function

$$\boldsymbol{Y = \begin{cases} 1, & if\ Y_{in} \geq 0 \\ -1, & if\ Y_{in} < 0 \end{cases}}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.1.4 Multilayer Perceptron, Back Propagation

Multiple Layer Perceptron:

In simple a multilayer perceptron is a feedforward neural network with one or more hidden layers.

In the Multilayer perceptron, there can more than one linear layer (combinations of neurons). If we take the simple example the three-layer network, first layer will be the **input layer** and last will be **output layer** and middle layer will be called **hidden layer**. We feed our input data into the input layer and take the output from the output layer. We can increase the number of the hidden layer as much as we want, to make the model more complex according to our task.

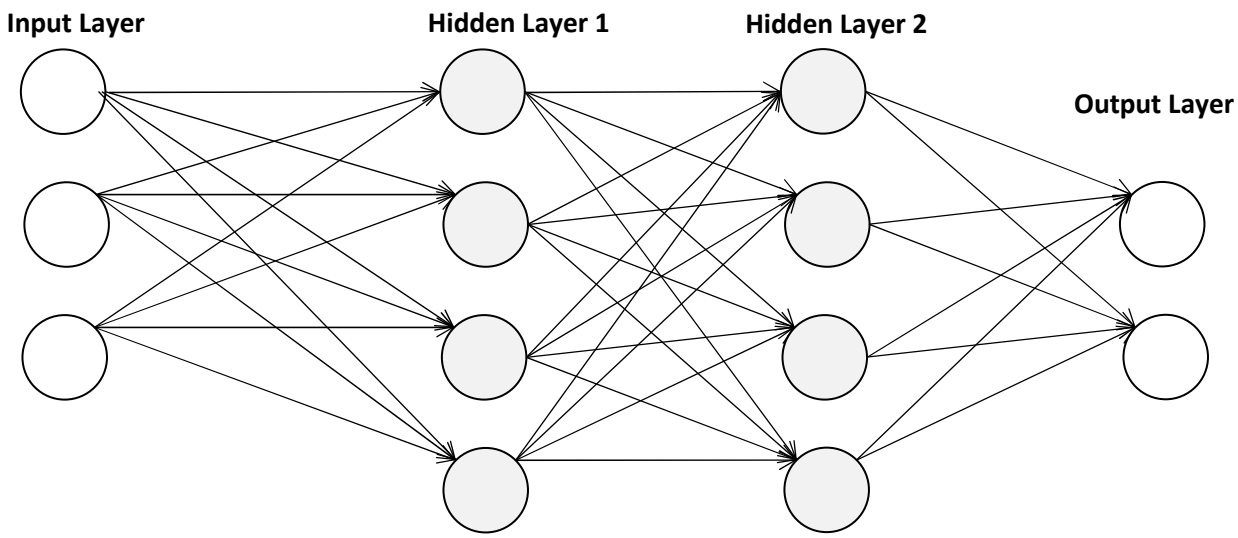
So, in MLP network consists of an **input layer** of source neurons, at least one middle or **hidden layer** of computational neurons, and an **output layer** of computational neurons.

The input signals are propagated in a forward direction on a layer-by-layer basis.

A hidden layer "hides" its desired output. Neurons in the hidden layer cannot be observed through the input/output behavior of the network. There is no obvious way to know what the desired output of the hidden layer should be.

DEEPAK BHATTA

Perceptrons can be improved if it is placed in a multilayered network.



In logic gate XOR and XNOR are the example of multilayered perceptron network.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Back Propagation:

The term back propagation and its general use in neural networks was announced by Rumelhart, Hinton & Williams in 1986.

Backpropagation, short for "**backward propagation of errors,"** is an algorithm for supervised learning of artificial neural networks using gradient descent.

In machine learning, back propagation is widely used algorithm in training multi-layer perceptrons (artificial Neural Networks).

The back-propagation algorithm is used to update the neural network weights when they are not able to make the correct predictions. Hence, we should train the neural network before applying backpropagation.

Learning in a multilayer network proceeds the same way as for a perceptron.

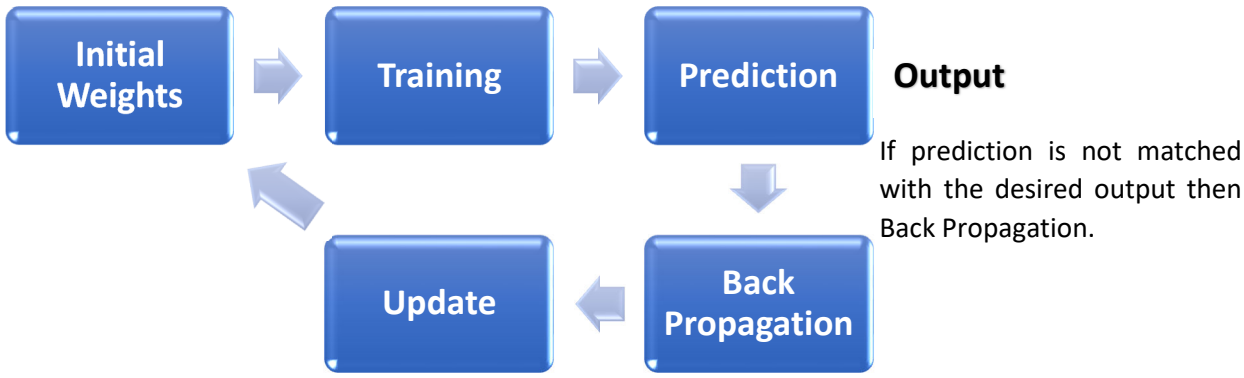
A training set of input patterns is presented to the network.

The network computes its output pattern, and if there is error or in other words a difference between actual output and desired output patterns the weights are adjusted to reduce the error.

In a back-propagation neural network, the training algorithm has two phases.

First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer.

If the pattern is different from the desired output, an error is calculated and then propagated backward through the network from output layer to the input layer. The weights are modified as the error is propagated.



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Back Propagation Training Algorithm:

Step 1: Initialization of weight and bias

Step 2: Feed Forward

- Caluclate sum of Products

$$s = b + \sum_{i=1}^{n} X_i * W_i$$

- Activation Function

We can use Sigmoid Activation Function

$$f(s) = \frac{1}{1 + e^{-s}}$$

Step 3: Back Propagation of Error

- Prediction Error

$$E = \frac{1}{2}(desired\ output - predicated\ output)^2$$

Step 4: Updation of Weight and Bias

We can use weight update equation:

$$w_{new} = w_{old} + \propto [d - y] . X$$

w_{new} = New Updated Weights.

w_{old} = Current Weights.

$\propto$ = Network Learning Rate or η (Eta)

$Assume\ any\ learning\ rate\ between\ 0\ to\ 1\ any\ positive\ value, default\ vlue\ is\ 0.1\ or\ 0.01$

d = Desired output

y = Predicted output

X = Current input at which the network made false prediction

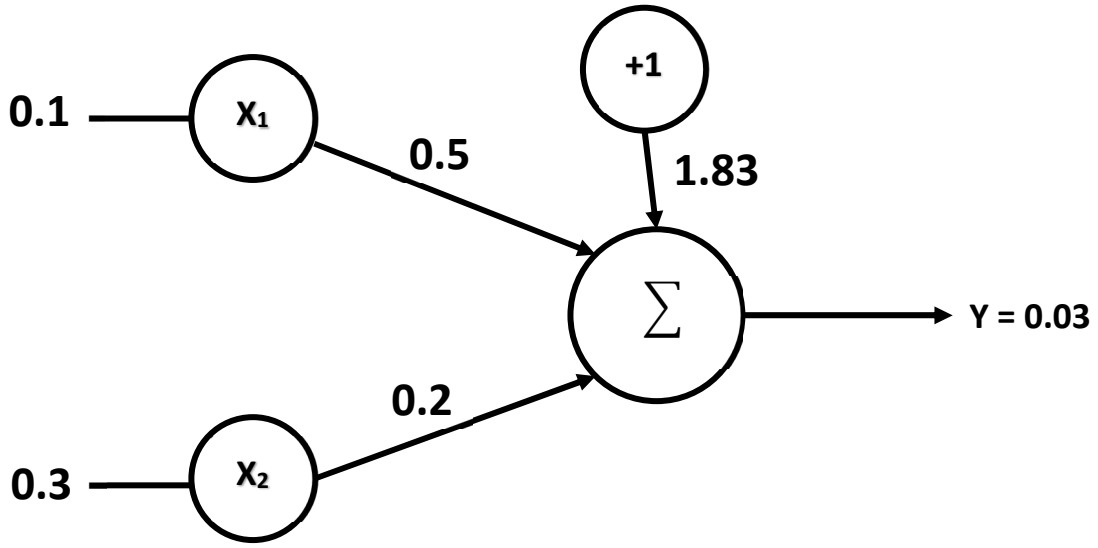
 # How to Minimize Prediction Error?

There is a prediction error and it should be minimized until reaching an acceptable error.

There must be something to change in order to minimize the error. The only parameter to change is the weight. So, we can use the weight update equation.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Example:



Sol^n:

Trainning Data

X_1	X_2	Output
0.1	0.3	0.03

Initial Weights

W_1	W_2	b
0.5	0.2	1.83

Network Training

Steps to train our network:

1. Prepare activation function input (Sum of products between inputs and weights).
2. Activation function output.

 # Sum of Products:

$$s = b + \sum_{i=1}^{n} X_i * W_i$$

$$s = X_1 * W_1 + X_2 * W_2 + b$$

$$s = 0.1 * 0.5 + 0.3 * 0.2 + 1.83$$

$$s = 1.94$$

 # Activation Function:

Here, sigmoid activation function is used,

$$f(s) = \frac{1}{1+e^{-s}}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$$f(s) = \frac{1}{1+e^{-1.94}}$$

$$f(s) = \frac{1}{1+0.144}$$

$$f(s) = \frac{1}{1.144}$$

$$f(s) = 0.874 \quad \textbf{(Predicted Output)}$$

 # Prediction Error:

- After getting the predicated outputs, next is to measure the prediction error of the network.
- We can use the Square Error function as follows:

$$E = \frac{1}{2}(desired\ output - predicted\ output)^2$$

- Based on the predicted output, the prediction error is:

$$E = \frac{1}{2}(0.03 - 0.874)^2$$

$$E = \frac{1}{2}(-0.844)^2$$

$$E = \frac{1}{2}(0.713)$$

$E = 0.357$ *(The training process is updated till the sum of squared error is less than 0.001)*

 # Weight Training :

- We can use the weight update equation:

$$w_{new} = w_{old} + \propto [d - y] . X$$

Here,

w_{new} = New Updated Weights

w_{old} = Current Weights = **[1.83, 0.5, 0.2]** *[b, W_1, W_2]*

$\propto$ = Network Learning Rate or η (Eta) = 0.01

$Assume\ any\ learning\ rate\ between\ 0\ to\ 1\ any\ positive\ value, default\ vlue\ is\ 0.1\ or\ 0.01$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$\boldsymbol{d}$ = Desired output = **0.03**

$\boldsymbol{y}$ = Predicted output = **0.874**

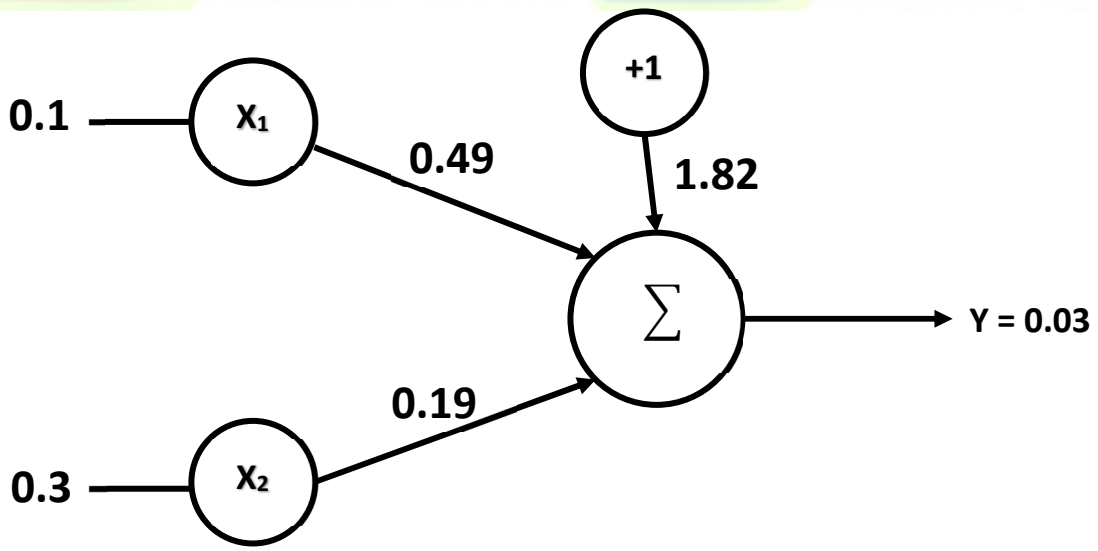
$\boldsymbol{X}$ = Current input at which the network made false prediction = **[+1, 0.1, 0.3]** *[b, X_1, X_2]*

$$\boldsymbol{w_{new} = w_{old} + \propto [d - y] . X}$$
$$= [1.83, 0.5, 0.2] + 0.01[0.03 - 0.874].[+1, 0.1, 0.3]$$
$$= [1.83, 0.5, 0.2] + [-0.00844].[+1, 0.1, 0.3]$$
$$= [1.83, 0.5, 0.2] + [-0.0084, -0.0084, -0.0025]$$
$$= [1.83, 0.5, 0.2] - [0.0084, 0.0084, 0.0025]$$
$$= [(1.83 - 0.0084),\ (0.5 - 0.0084),\ (0.2 - 0.0025)]$$
$$= [1.82, 0.49, 0.19]$$ *[b, W_1, W_2]*

- **The new weights are:**

W_1	W_2	b
0.49	0.19	1.82

DEEPAK BHATTA



- Based on the new weights, the network will be re-trained.
- Continue these operations until prediction error reaches an acceptable value.
  i. Updating Weights
  ii. Re-training Networks
  iii. Calculating Prediction Error

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.1.5 Hopefield Network

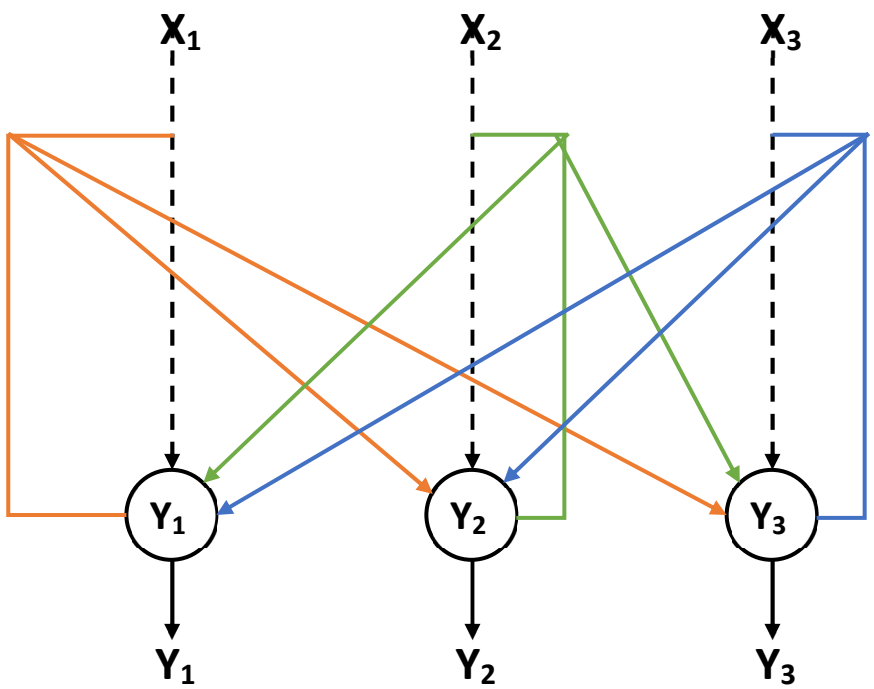
Hopfield neural network was invented by Dr. John J. Hopfield in 1982. It consists of a single layer which contains one or more fully connected recurrent neurons. The Hopfield network is commonly used for auto-association and optimization tasks.

Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary 0, 1 or bipolar +1, −1 in nature. The network has symmetrical weights with no self-connections i.e., $W_{ij} = W_{ji}$ and $W_{ii} = 0$.

Architecture

Following are some important points to keep in mind about discrete Hopfield network −

- This model consists of neurons with one inverting and one non-inverting output.
- The output of each neuron should be the input of other neurons but not the input of self.
- Weight/connection strength is represented by W_{ij}.
- Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
- Weights should be symmetrical, i.e. $W_{ij} = W_{ji}$



The output from Y_1 going to Y_2 and Y_3 have the weights W_{12} and W_{13} respectively. Similarly, other arcs have the weights on them.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.1.6 Kohonen Network

A Kohonen Network sometimes called **self-organizing map (SOM)** or **self-organizing feature map (SOFM)**.

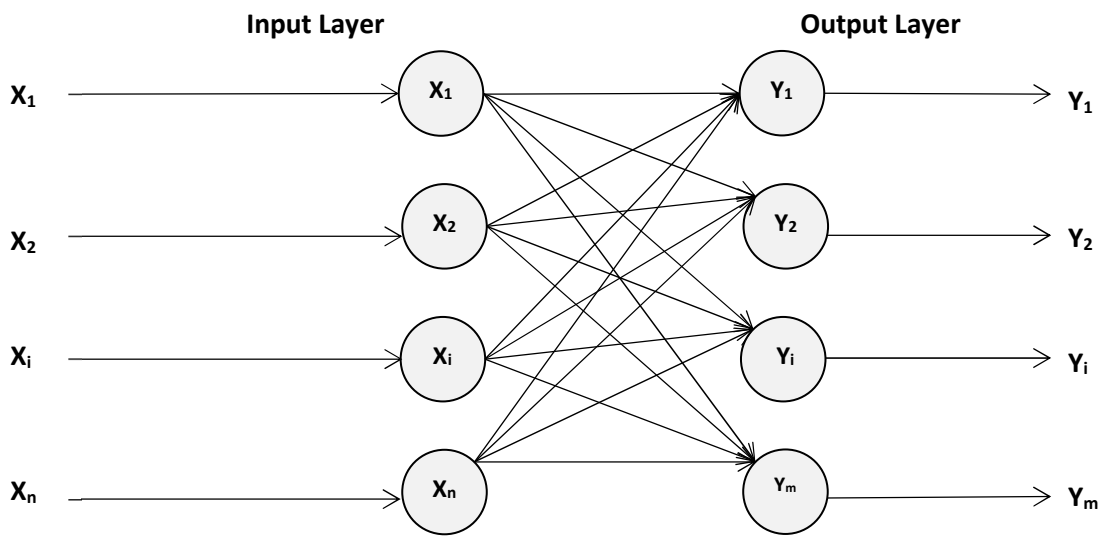
This network was introduced by the Finnish professor Teuvo Kohonen in the 1980s.

It is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction.

Kohonen's networks are one of basic types of self-organizing neural networks. The ability to self-organize provides new possibilities - adaptation to formerly unknown input data. It seems to be the most natural way of learning, which is used in our brains, where no patterns are defined. Those patterns take shape during the learning process, which is combined with normal work. Kohonen's networks are a synonym of whole group of nets which make use of self-organizing, competitive type learning method.

- The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into higher dimensional output or Kohonen layer.
- Training in the Kohonen network begins with the winner's neighborhood of a fairly large size. Then, as training proceeds, the neighborhood size gradually decreases.

Architecture of Kohonen Network or SOM



© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Algorithm of Kohonen Network

Step 0: Initialize the weights W_{ij}. Random values may be assumed. Initialize the learning rate $(\boldsymbol{\eta}\ \boldsymbol{or}\ \boldsymbol{\alpha})$.

Step 1: While stopping condition false, do steps 2 to 8.

Step 2: For each input vector **X**, do steps 3 to 5.

Step 3: Calculate Square of the Euclidean Distance i.e. for each i=1 to m.

$$\boldsymbol{D(j)} = \sqrt{\sum_{i=1}^{n}\left(\boldsymbol{W_{ij}} - \boldsymbol{X_i}\right)^2}$$

You'll get same Winning Neuron from both the formula.

Or in some text book or in research papers you'll see

$$\boldsymbol{D(j)} = \sum_{i=1}^{n}\left(\boldsymbol{W_{ij}} - \boldsymbol{X_i}\right)^2$$

Don't be panic values are different but this will still yield the same output at last.

Step 4: The minimum $\boldsymbol{D(i,j)}$ is selected to be the winning unit.

Step 5: For all neurons j within a specific neighborhood of j and for all i, calculate new weights.

$$\boldsymbol{W_{ij}(new)} = \boldsymbol{W_{ij}(old)} + \boldsymbol{\alpha}[\boldsymbol{X_i} - \boldsymbol{W_{ij}(old)}]$$

Step 6: Update learning rate $\boldsymbol{\alpha}$. (Reduce the $\boldsymbol{\alpha}$ at each Iteration) $\boldsymbol{\alpha(t+1)} = \frac{\boldsymbol{\alpha(t)}}{\mathbf{2}}$

Step 7: Reduce radius of topological neighborhood at specified times.

Step 8: Test stopping condition. Typically, this is a small value of the learning rate with which the weight updates are insignificant.

Simple Example:

A Kohonen Self-Organizing Map (SOM) to cluster four vectors

Let the vector to be cluster be

(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1).

The maximum number of clusters to be formed is m=2.

Suppose the learning rate (geometric decrease) is $\boldsymbol{\alpha(0) = 0.6}$,

$$\boldsymbol{\alpha(t+1)} = \frac{\boldsymbol{\alpha(t)}}{\mathbf{2}}$$

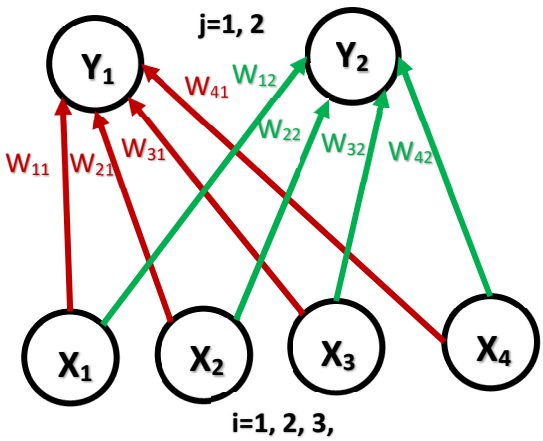
$$\boldsymbol{or}\quad \mathbf{0.5} * \boldsymbol{\alpha(t)}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Step 0: Let, Initial Weight Matrix between 0 to 1

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

Let, Initial learning rate: $\alpha(0) = 0.6$



Step 1: Begin Training.

Step 2: For the first vector, $(X_1, X_2, X_3, X_4) = (1, 1, 0, 0)$, do step 3 to 5.

Step 3:

$$D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}$$

$$= \sqrt{(1 - 0.2)^2 + (1 - 0.6)^2 + (0 - 0.5)^2 + (0 - 0.9)^2}$$

$$= \sqrt{0.64 + 0.16 + 0.25 + 0.81}$$

$$= \sqrt{1.86}$$

$$= 1.36$$

$$D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}$$

$$= \sqrt{(1 - 0.8)^2 + (1 - 0.4)^2 + (0 - 0.7)^2 + (0 - 0.3)^2}$$

$$= \sqrt{0.04 + 0.36 + 0.49 + 0.09}$$

$$= \sqrt{0.98}$$

$= 0.98$ **(Minimum Euclidean Distance)**

Step 4: The input vector is closest to output node 2, so j=2.

(As $D(2)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]$$

Where, learning rate $\alpha = 0.6$

$$W_{12}(new) = W_{12}(old) + 0.6[X_1 - W_{12}(old)]$$
$$W_{12}(new) = 0.8 + 0.6[1 - 0.8] = 0.92$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$$W_{22}(new) = W_{22}(old) + 0.6[X_2 - W_{22}(old)]$$
$$W_{22}(new) = 0.4 + 0.6[1 - 0.4] = 0.76$$

$$W_{32}(new) = W_{32}(old) + 0.6[X_3 - W_{32}(old)]$$
$$W_{32}(new) = 0.7 + 0.6[0 - 0.7] = 0.28$$

$$W_{42}(new) = W_{42}(old) + 0.6[X_4 - W_{42}(old)]$$
$$W_{42}(new) = 0.3 + 0.6[0 - 0.3] = 0.12$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

Step 2: For the second vector, $(X_1, X_2, X_3, X_4) = (0, 0, 0, 1)$, do step 3 to 5.

Step 3:

$$D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}$$

$$= \sqrt{(0 - 0.2)^2 + (0 - 0.6)^2 + (0 - 0.5)^2 + (1 - 0.9)^2}$$

$$= \sqrt{0.04 + 0.36 + 0.25 + 0.01}$$

$$= \sqrt{0.66}$$

$= 0.81$ **(Minimum Euclidean Distance)**

$$D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}$$

$$= \sqrt{(0 - 0.92)^2 + (0 - 0.76)^2 + (0 - 0.28)^2 + (1 - 0.12)^2}$$

$$= \sqrt{0.85 + 0.58 + 0.08 + 0.77}$$

$$= \sqrt{2.28}$$

$$= 1.51$$

Step 4: The input vector is closest to output node 1, so j=1.

(As $D(1)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

$Where, learning\ rate\ \boldsymbol{\alpha} = \mathbf{0.6}$

$$\boldsymbol{W_{11}(new) = W_{11}(old) + 0.6[X_1 - W_{11}(old)]}$$
$$\boldsymbol{W_{11}(new) = 0.2 + 0.6[0 - 0.2] = 0.08}$$

$$\boldsymbol{W_{21}(new) = W_{21}(old) + 0.6[X_2 - W_{21}(old)]}$$
$$\boldsymbol{W_{21}(new) = 0.6 + 0.6[0 - 0.6] = 0.06}$$

$$\boldsymbol{W_{31}(new) = W_{31}(old) + 0.6[X_3 - W_{31}(old)]}$$
$$\boldsymbol{W_{31}(new) = 0.5 + 0.6[0 - 0.5] = 0.20}$$

$$\boldsymbol{W_{41}(new) = W_{41}(old) + 0.6[X_4 - W_{41}(old)]}$$
$$\boldsymbol{W_{41}(new) = 0.9 + 0.6[1 - 0.9] = 0.96}$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.06 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

Step 2: For the third vector, $(\boldsymbol{X_1, X_2, X_3, X_4}) = (\mathbf{1, 0, 0, 0})$, do step 3 to 5.

Step 3:

$$\boldsymbol{D(1)} = \sqrt{(\boldsymbol{X_1 - W_{11}})^2 + (\boldsymbol{X_2 - W_{21}})^2 + (\boldsymbol{X_3 - W_{31}})^2 + (\boldsymbol{X_4 - W_{41}})^2}$$

$$= \sqrt{(\mathbf{1 - 0.08})^2 + (\mathbf{0 - 0.06})^2 + (\mathbf{0 - 0.20})^2 + (\mathbf{0 - 0.96})^2}$$

$$= \sqrt{\mathbf{0.8464 + 0.0036 + 0.04 + 0.9216}}$$

$$= \sqrt{\mathbf{1.8116}}$$

$$= \mathbf{0.95}$$

$$\boldsymbol{D(2)} = \sqrt{(\boldsymbol{X_1 - W_{12}})^2 + (\boldsymbol{X_2 - W_{22}})^2 + (\boldsymbol{X_3 - W_{32}})^2 + (\boldsymbol{X_4 - W_{43}})^2}$$

$$= \sqrt{(\mathbf{1 - 0.92})^2 + (\mathbf{0 - 0.76})^2 + (\mathbf{0 - 0.28})^2 + (\mathbf{0 - 0.12})^2}$$

$$= \sqrt{\mathbf{0.0064 + 0.58 + 0.08 + 0.0144}}$$

$$= \sqrt{\mathbf{0.6808}}$$

$= \mathbf{0.28}$ **(Minimum Euclidean Distance)**

Step 4: The input vector is closest to output node 2, so j=2.

(As $\boldsymbol{D(2)}$ has minimum Euclidean Distance)

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Step 5: The weights on the winning unit/neuron are updated:

$$\boldsymbol{W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]}$$

Where, learning rate $\boldsymbol{\alpha = 0.6}$

$\boldsymbol{W_{12}(new) = W_{12}(old) + 0.6[X_1 - W_{12}(old)]}$
$\boldsymbol{W_{12}(new) = 0.92 + 0.6[1 - 0.92] = 0.968}$

$\boldsymbol{W_{22}(new) = W_{22}(old) + 0.6[X_2 - W_{22}(old)]}$
$\boldsymbol{W_{22}(new) = 0.76 + 0.6[0 - 0.76] = 0.304}$

$\boldsymbol{W_{32}(new) = W_{32}(old) + 0.6[X_3 - W_{32}(old)]}$
$\boldsymbol{W_{32}(new) = 0.28 + 0.6[0 - 0.28] = 0.112}$

$\boldsymbol{W_{42}(new) = W_{42}(old) + 0.6[X_4 - W_{42}(old)]}$
$\boldsymbol{W_{42}(new) = 0.12 + 0.6[0 - 0.12] = 0.048}$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.06 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

Step 2: For the fourth vector, $(\boldsymbol{X_1, X_2, X_3, X_4}) = (\boldsymbol{0, 0, 1, 1})$, do step 3 to 5.

Step 3:

$\boldsymbol{D(1) = \sqrt{(X_1 - W_{11})^2 + (X_2 - W_{21})^2 + (X_3 - W_{31})^2 + (X_4 - W_{41})^2}}$

$\boldsymbol{= \sqrt{(0 - 0.08)^2 + (0 - 0.06)^2 + (1 - 0.20)^2 + (1 - 0.96)^2}}$

$\boldsymbol{= \sqrt{0.0064 + 0.0036 + 0.64 + 0.0016}}$

$\boldsymbol{= \sqrt{0.6516}}$

$\boldsymbol{= 0.2}$ **(Minimum Euclidean Distance)**

$\boldsymbol{D(2) = \sqrt{(X_1 - W_{12})^2 + (X_2 - W_{22})^2 + (X_3 - W_{32})^2 + (X_4 - W_{43})^2}}$

$\boldsymbol{= \sqrt{(0 - 0.968)^2 + (0 - 0.304)^2 + (1 - 0.112)^2 + (1 - 0.048)^2}}$

$\boldsymbol{= \sqrt{0.937024 + 0.092416 + 0.788544 + 0.906304}}$

$\boldsymbol{= \sqrt{2.726032}}$

$\boldsymbol{= 0.9757}$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Step 4: The input vector is closest to output node 1, so j=1.

(As $D(1)$ has minimum Euclidean Distance)

Step 5: The weights on the winning unit/neuron are updated:

$$W_{ij}(new) = W_{ij}(old) + \alpha[X_i - W_{ij}(old)]$$

$Where, learning\ rate\ \alpha = 0.6$

$$W_{11}(new) = W_{11}(old) + 0.6[X_1 - W_{11}(old)]$$
$$W_{11}(new) = 0.08 + 0.6[0 - 0.08] = 0.032$$

$$W_{21}(new) = W_{21}(old) + 0.6[X_2 - W_{21}(old)]$$
$$W_{21}(new) = 0.24 + 0.6[0 - 0.24] = 0.096$$

$$W_{31}(new) = W_{31}(old) + 0.6[X_3 - W_{31}(old)]$$
$$W_{31}(new) = 0.20 + 0.6[1 - 0.20] = 0.680$$

$$W_{41}(new) = W_{41}(old) + 0.6[X_4 - W_{41}(old)]$$
$$W_{41}(new) = 0.96 + 0.6[1 - 0.96] = 0.984$$

The new weighted matrix is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

As all the input vectors are done then, Iteration 1 is completed.

Step 6: Reduce the learning rate:

$$\alpha(t+1) = \frac{\alpha(t)}{2}\ \ or\ \ 0.5 * \alpha(t)$$

So, $\frac{0.6}{2} = 0.3$

Step1: While stopping condition false, do steps 2 to 8.

Now, having a new learning rate 0.3

The weight matrix after the second epoch/iteration of training is:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.015 & 0.9776 \\ 0.047 & 0.5128 \\ 0.633 & 0.0776 \\ 0.992 & 0.0336 \end{bmatrix}$$

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Modifying the adjustment procedure for the learning rate $\boldsymbol{\alpha}$ so that it decreases geometrically from 0.6 to 0.01 over 100 epochs/iterations gives the more precise output. Although the Neural network Kohonen SOM algorithm automatically stops when it gets the more accurate result.

But for the exam point of view you can just do up to 1st Iteration. Not necessary to do all the iteration to get result. It'll be time consuming.

At Iteration 100: Weight matrix:

$$\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix} = \begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$

Cluster 1 ← → Cluster 2

The first column of which is the average of the two vectors placed in cluster 1 and the second column of which is the average of the two vectors placed in cluster 2.

<u>TEST NETWORK</u>

Suppose the input pattern is 1100.

Then,

$$D(j) = \sqrt{(X_1 - W_{1j})^2 + (X_2 - W_{2j})^2 + (X_3 - W_{3j})^2 + (X_4 - W_{4j})^2}$$

$$D(1) = \sqrt{(1-0)^2 + (1-0)^2 + (0-0.5)^2 + (0-1)^2} = 3.25$$

$$D(2) = \sqrt{(1-1)^2 + (1-0.5)^2 + (0-0)^2 + (0-0)^2} = 0.25$$

Thus **Neuron 2** is the **"Winner"**, and is the localized active region of the SOM. Notice that we may label this input pattern to belong to **cluster 2**.

For all the other patterns, we find the cluster are as listed below.

X_1	X_2	X_3	X_4	Cluster
1	1	0	0	**2**
0	0	0	1	**1**
1	0	0	0	**2**
0	0	1	1	**1**

Why Use Kohonen Network or SOM (Self-Organizing Maps)?

- Find clusters in large amounts of data (dimensionality reduction).
- Combine diverse datasets to find patterns.
- Powerful Visualization.
- Application in any field.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.2 Expert System

Expert System are computer programs built for commercial application using the programming techniques of AI which are developed for problem solving.

Built for varieties of purposes including **Medical Diagnosis, Electronic Fault Finding, Mineral Prospecting, Computer System Configuration,** etc.

Definition of Expert System:

An Expert System is a collection of programs or Computer Software that solves problems in the domain of interest. *It is called system because it consists of both problem-solving component and a support component.*

The process of building Expert System is called Knowledge Engineering and is done by Knowledge Engineer.

What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first Expert System was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

The performance of an expert system is based on the expert's knowledge stored in its Knowledge Base (KB). The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an Expert System is a **suggestion of spelling errors** while typing in the Google search box.

Knowledge Engineer:

- It is a human with a background of computer and AI who knows how to build an Expert System.
- Decides how to represent the knowledge from human expert or other sources.
- Creates a relation with human expert to elicit knowledge from him/her.

Knowledge Engineering:

- It is the acquisition of knowledge and gives report to knowledge engineer.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Expert:

- Evaluates the expert system and gives report to knowledge engineer.
- The process continues until the performance is found satisfactory by the experts.

Expert system provides the following important features:

- Facility for non-expert personnel to solve problems that require some expertise
- Speedy solution.
- Reliable solution.
- Cost reduction.
- Power to manage without human expert.
- Wider areas of knowledge.

Expert system provides the following important features:

- Human experts are difficult to find.
- Human experts are expensive.
- Knowledge improvement is important.
- The available information is poor, partial, incomplete.
- Problems are incompletely defined.
- There is lack of knowledge among all those who need it.
- The problem is rapidly changing legal rules and codes.

Some popular examples of the Expert System are as follows:

DENDRAL: It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of their mass spectra and knowledge base of chemistry.

MYCIN: It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteremia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.

PXDES: It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like the shadow. This shadow identifies the type and degree of harm.

CaDeT: The CaDeT expert system is a diagnostic support system that can detect cancer at early stages.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Characteristics of Expert System:

- **High Performance:**
  - The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.

- **The Highest Level of Expertise:**
  - The expert system offers the highest level of expertise. It provides efficiency, accuracy and imaginative problem-solving.

- **Adequate response time:**
  - Should respond within reasonable amount of time comparable to or better than time taken by human experts to reach the decision.
  - An Expert System interacts in a very reasonable period of time with the user.
  - The total time must be less than the time taken by an expert to get the most accurate solution for the same problem.

- **Good reliability:**
  - The Expert System needs to be reliable, and it must not make any mistake.

- **Flexibility:**
  - Huge Knowledge → Must have efficient mechanism to add, change and delete the knowledge.

- **Highly responsive:**
  - Expert System provides the result for any complex query within a very short period of time.

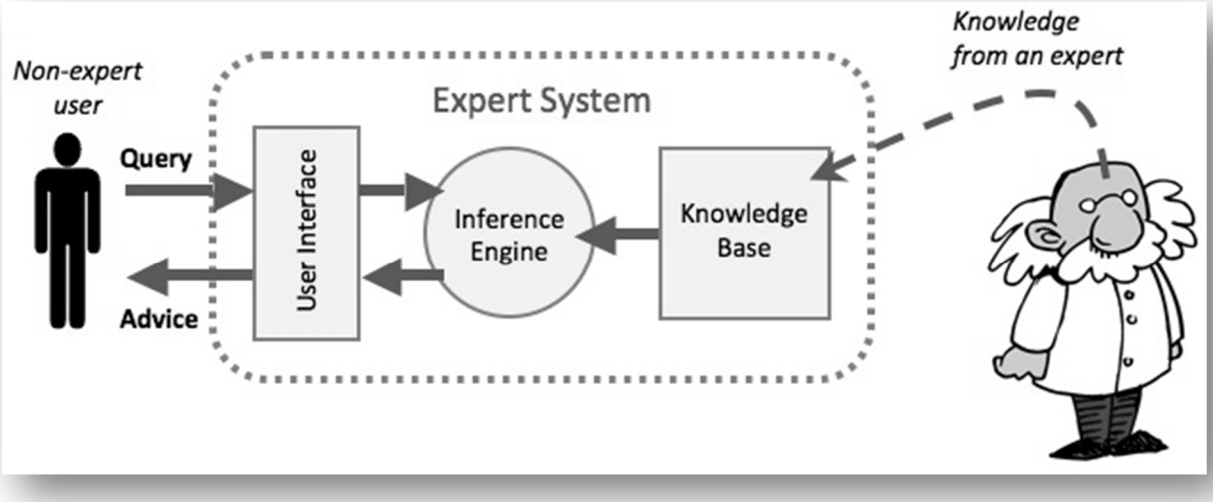
- **Capable of handling challenging decision & problems:**
  - An expert system is capable of handling challenging decision problems and delivering solutions.

Components of Expert System:

An expert system mainly consists of three components:

1. User Interface
2. Inference Engine
3. Knowledge Base

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



1. **User Interface:**

- User interface provides interaction between user of the Expert System and Expert System itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the Expert System need not be necessarily an expert in Artificial Intelligence.
- It explains how Expert System has arrived at a particular recommendation
- With the help of a user interface, the expert system interacts with the user, takes queries as an input in a readable format, and passes it to the inference engine. After getting the response from the inference engine, it displays the output to the user. In other words, it is an interface that helps a non-expert user to communicate with the expert system to find a solution.

2. **Inference Engine (Rules of Engine):**

- The inference engine is known as the brain of the expert system as it is the main processing unit of the system. It applies inference rules to the knowledge base to derive a conclusion or deduce new information. It helps in deriving an error-free solution of queries asked by the user.
- With the help of an inference engine, the system extracts the knowledge from the knowledge base.
- **There are two types of inference engine:**

  i. **Deterministic Inference Engine:** The conclusions drawn from this type of inference engine are assumed to be true. It is based on facts and rules.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

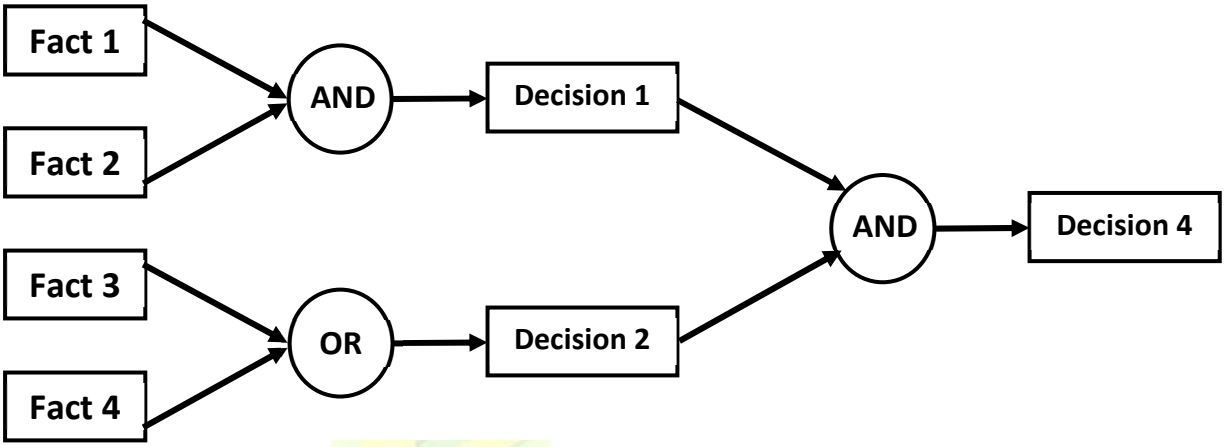
ii. **Probabilistic Inference engine:** This type of inference engine contains uncertainty in conclusions, and based on the probability.

Inference engine uses the below modes to derive the solutions:

Forward Chaining: It starts from the known facts and rules, and applies the inference rules to add their conclusion to the known facts.

Backward Chaining: It is a backward reasoning method that starts from the goal and works backward to prove the known facts.

Forward Chaining:

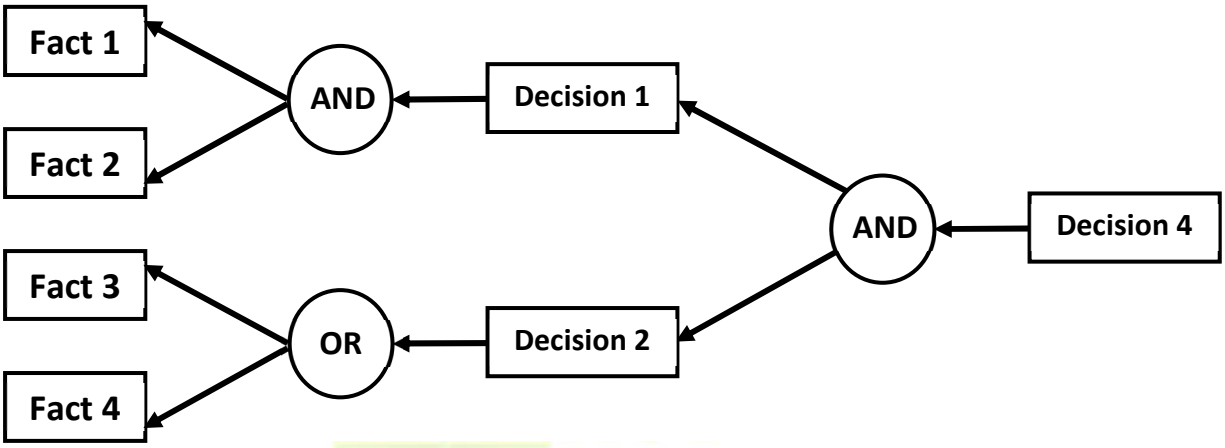


- The forward chaining (or forward reasoning) is one of the two main methods of reasoning when using an inference engine.
- It is a strategy of an expert system to answer the question, **"What can happen next?"**
- It is also known as **data driven** inference technique.
- Forward chaining matches the set of conditions and inform results from these conditions. Basically, forward chaining starts from a new data and aims for any conclusion.
- It is **bottom-up** reasoning.
- It uses a **breadth first search** technique.
- It continues until no more rules can be applied or some cycle limit is met.
- Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- This strategy is followed for working on conclusion, result, or effect. For example, **prediction of share market status** as an effect of changes in interest rates.
- It is mostly used in commercial applications i.e. event driven systems are common example of forward chaining.
- It can create an infinite number of possible conclusions.
- **For example:** Let say, **if it is cold then I will wear a sweater**. Here **"it is cold"** is the ***data*** and **"I will wear a sweater"** is a ***decision***. It was already known that it is cold that is why it was decided to wear a sweater, this process is called **forward chaining**.

Backward Chaining:



- The forward chaining (or backward reasoning) is an interface method described colloquially *(बोलचाल) (Not formal or literary)* as working backward from goal.
- It is used in automated theorem provers, interface engines, proof assistants, and other artificial intelligence applications.
- With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**
- It is also called as **goal driven** information technique.
- It is a backward search from goal to the conditions used to get the goal. Basically, it starts from the possible conclusion or goal and aims for necessary data.
- It is **top-down** reasoning.
- It uses **depth first search** technique.
- It processes operations in a backward direction from end to start, it will stop when matching initial condition is met.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

- On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result.
- This strategy is followed for finding out cause or reason. For example, **diagnosis of blood cancer in humans**.
- It is used in interrogative commercial applications i.e. finding items that fulfill possible goals.
- Number of possible final answers is reasonable.
- **For example:** Let say, **if it is cold then I will wear a sweater**. Here, we have our possible conclusion **"I will wear a sweater".** If I am wearing a sweater then it can be started that "**it is cold**" that is why I am wearing a sweater. Hence it was derived in backward direction, this process is called **forward chaining**.

3. Knowledge Base:

- The knowledgebase is a type of storage that stores knowledge acquired from the different experts of the particular domain.
- It is considered as big storage of knowledge. The more the knowledge base, the more precise will be the Expert System.
- It is similar to a database that contains information and rules of a particular domain or subject.
- One can also view the knowledge base as collections of objects and their attributes. Such as a Lion is an object and its attributes are it is a mammal, it is not a domestic animal, etc.

Components of Knowledge Base

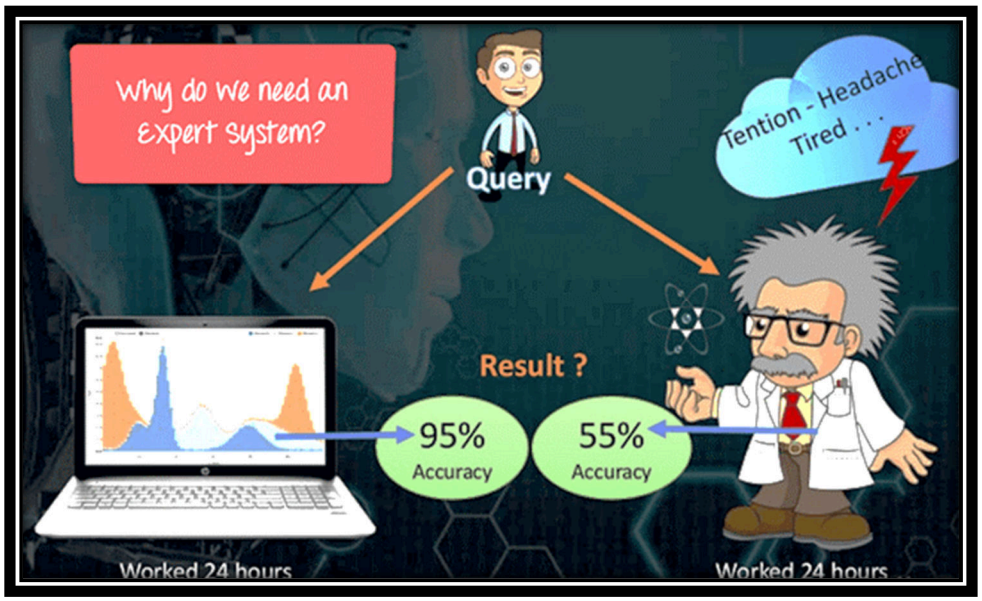
Factual Knowledge: The knowledge which is based on facts and accepted by knowledge engineers comes under factual knowledge.

Heuristic Knowledge: This knowledge is based on practice, the ability to guess, evaluation, and experiences.

Knowledge Representation: It is used to formalize the knowledge stored in the knowledge base using the If-else rules.

Knowledge Acquisitions: It is the process of extracting, organizing, and structuring the domain knowledge, specifying the rules to acquire the knowledge from various experts, and store that knowledge into the knowledge base.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]



Advantages of Expert System

- Increased output and productivity.
- Decreased decision-making time.
- Increased process and product quality.
- Reduced downtime.
- Capture scarce expertise.
- Flexibility.
- Easier equipment operation.
- Elimination of expensive equipment.
- Operation in hazardous environment.
- Accessibility to knowledge and help desks.
- Integration of several experts' opinions.
- Can work with incomplete or uncertain information.
- Provide training.
- Enhancement of problem solving and decision making.
- Improved decision-making processes.
- Improved decision quality.
- Ability to solve complex problems.
- Knowledge transfer to remote locations.
- Enhancement of other MIS.

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

Limitations of Expert System

- Knowledge is not always readily available.
- Expertise can be hard to extract from humans.
- Each expert's approach may be different yet correct.
- Hard, even for a highly skilled expert, to work under time pressure.
- Expert System users have natural cognitive limits.
- Expert System works well only in narrow domain of knowledge.
- Most experts have no independent means to validate their conclusions.
- Experts' vocabulary often limited and highly technical.
- Knowledge engineers are rare and expensive.
- Lack of trust by end-users.
- Knowledge transfer subject to a host of perceptual and judgmental biases.
- Expert System may not be able to arrive at valid conclusions.
- Expert System sometimes produce incorrect recommendations.

DEEPAK BHATTA KAJI

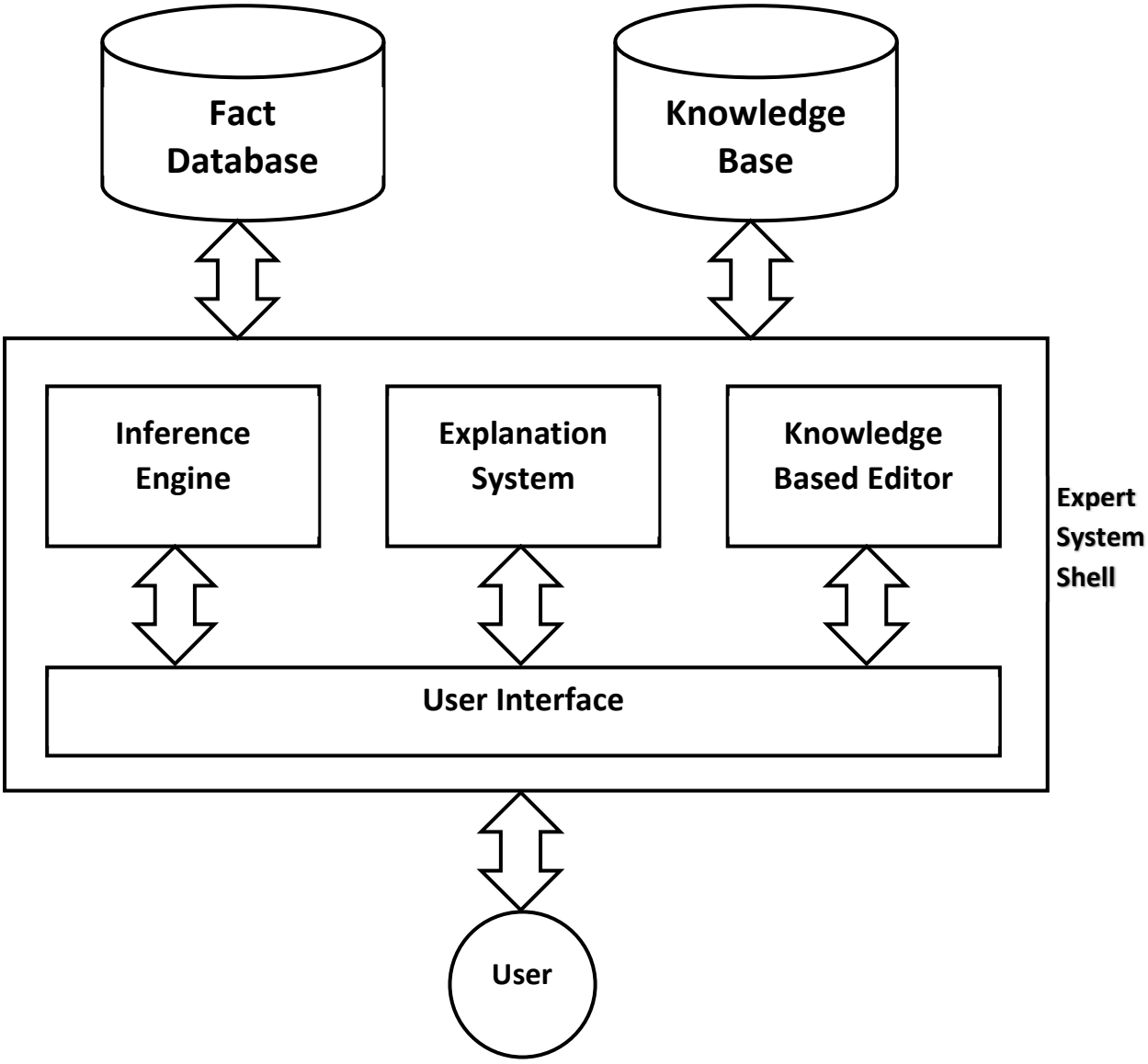
Applications of Expert Systems and Artificial Intelligence:

- Credit granting
- Information management and retrieval
- AI and expert systems embedded in products
- Hospitals and medical facilities
- Help desks and assistance
- Employee performance evaluation
- Loan analysis
- Virus detection
- Repair and maintenance
- Shipping
- Marketing
- Warehouse optimization
- Plant layout

© Deepak Bhatta Kaji (MCA)
Department of Computer Science – DEC [NAST]

6.2.1 Architecture of an Expert System

The Architecture of Expert System



- Expert knowledge derived from human experts.
- **Purpose:**
  - Diagnose Illnesses
  - Provide Recommendations
  - Solve other complex problems
- **Knowledge Base:** Database of rules (Domain Knowledge).

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

- **Explanation System:** Explains the decisions the system makes.
- **User Interface:** The means by which the user interacts with the expert system.
- **Knowledge Base Editor:** Allows the user to edit the information in the knowledge base.
- **Expert System Shell:** The part of an expert system that does not contain any domain specific or case specific knowledge is the expert system shell. A single expert system shell can be used to build a number of different expert systems. An example of an expert system shell is CLIPS. CLIPS is a C language Integrated Production System – an expert system shell. CLIPS use a LIPS-like notation to enter rules.

(Refer from component of Expert System which is discussed above)

A Simple Example of Medical Expert System:

- **Rules**
  1. If headache then prescribe pain killer.
  2. If headache and sore throat and coughing than diagnose flu.
  3. If tired and headache then diagnose glandular fever.
  4. If tired and sore throat then diagnose tonsillitis.
  5. If tired than diagnose stress.

- **User Interaction**
  - **Patient:** I have a headache
  - **System:** I suggest you take a pain killer. Do you have a sore throat?
  - **Patient:** Yes, I do.
  - **System:** Have you been coughing?
  - **Patient:** No, why do you ask?
  - **System:** If you had a head ache and a sore throat and had been coughing, I would have to conclude you might have the flu. It doesn't sound as though you have the flu? Have you been feeling tired recently?
  - **Patient:** Yes, I have.
  - **System:** It sounds as though you may have glandular fever or tonsillitis.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

6.2.2 Development of Expert System

Development of Expert Systems: General Steps

The process of Expert System development is iterative. Steps in developing the ES include −

Identify Problem Domain

- The problem must be suitable for an Expert System to solve it.
- Find the experts in task domain for the Expert System project.
- Establish cost-effectiveness of the system.

Design the System

- Identify the Expert System Technology
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Develop the Prototype

From Knowledge Base: The knowledge engineer works to −

- Acquire domain knowledge from the expert.
- Represent it in the form of If-THEN-ELSE rules.

Test and Refine the Prototype

- The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance.
- End users test the prototypes of the Expert System.

Develop and Complete the Expert System

- Test and ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems.
- Document the Expert System project well.
- Train the user to use Expert System.

Maintain the System

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Roles of Individuals who interact with the system: or Participants in Expert System Development and Use:

- **Domain Expert:** The individual who currently are expert in solving the problems; here the system is intended to solve.

- **Knowledge Engineer:** The individual who encodes the expert's knowledge in a declarative form that can be used by the expert system.

- **End-User:** The individual who will be consulting with the system to get advice which would have been provided by the expert.

- **System Engineer:** Builds the user interface, designs the declarative format of the knowledge base, and implements the inference engine.

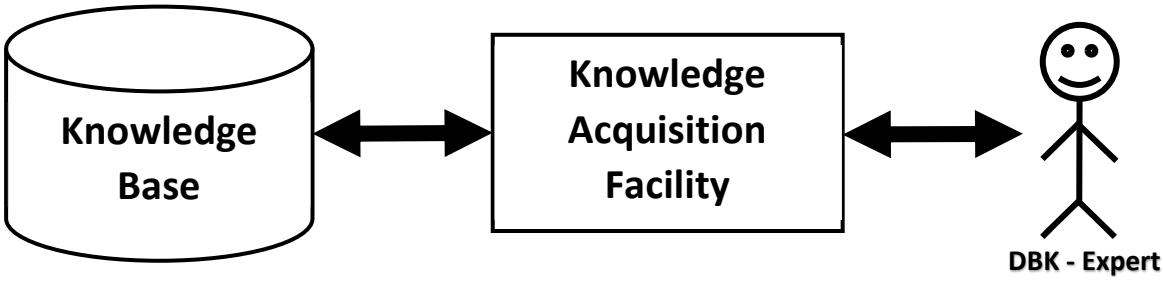
Other Key terms used in Expert systems

 # Facts and Rules:

- A fact is a small portion of important information. Facts on their own are of very limited use.
- The rules are essential to select and apply facts to a user problem.

 # Knowledge Acquisition:

- The term knowledge acquisition means how to get required domain knowledge by the expert system.
- The entire process starts by extracting knowledge from a human expert, converting the acquired knowledge into rules and injecting the developed rules into the knowledge base.
- Knowledge acquisition covers all forms of knowledge and any methods by which they may be obtained.
- It provides a convenient and efficient means of capturing and storing all components of the knowledge base.



© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

6.3 Natural Language Processing

Introduction of NLP:

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English.

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be −

- Speech
- Written Text

Components of NLP:

There are two components of NLP as given

1. **Natural Language Understanding (NLU):**

- Understanding involves the following tasks.
- Mapping the given input in natural language into useful representations.
- Analyzing different aspects of the language.

2. **Natural Language Generation (NLG):**

- It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

It involves:

- Text planning − It includes retrieving the relevant content from knowledge base.
- Sentence planning − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- Text Realization − It is mapping sentence plan into sentence structure.

The NLU is harder than NLG.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Difficulties in NLU / Ambiguities in Natural Language Processing

NL has an extremely rich form and structure. It is very ambiguous. There can be different levels of ambiguity:

- **Lexical ambiguity:** It is at very primitive level such as word-level. This type of ambiguity represents words that can have multiple assertions.

  For example, treating the word **"board"** as noun or verb? Another example the word "back" can be a noun (back stage), an adjective (back door) or an adverb (back away).

- **Syntax Level ambiguity:** A sentence can be parsed in different ways. Syntactic ambiguity is not usually something one strives for in clear communication, however, it does have its uses. One of the most entertaining is when double meanings are applied for comedic purposes. Ignoring the accepted context of a phrase and embracing an alternative meaning often ends in a laugh. Each of these general examples of ambiguity can carry double meanings:

  For example, **"He lifted the beetle with red cap."** *(उसले रातो टोपीले गोब्रे कीरा उठायो।)*

  – **Did he use red cap to lift the beetle or he lifted a beetle that had red cap?**

  – *के उसले गोब्रे कीरा उठाउन रातो टोपी प्रयोग गर्यो कि उसले रातो टोपीमा भएको गोब्रे कीरा उठायो?*

  "**I saw someone on the hill with a telescope."** *(मैले टेलिस्कोपमा कसैलाई देखें।)*

  – **Did you use a telescope to see someone on the hill or did you see someone on the hill holding a telescope?**

  – *ताईंले पहाडमा कसैलाई हेर्न टेलिस्कोप प्रयोग गर्नुभयो वा पहाडमा कसैलाई टेलिस्कोप समातेको ताईंले देख्नुभयो?*

- **Referential ambiguity:** Referring to something using pronouns. For example, Rima went to Gauri. She said, **"I am tired."** – Exactly who is tired?

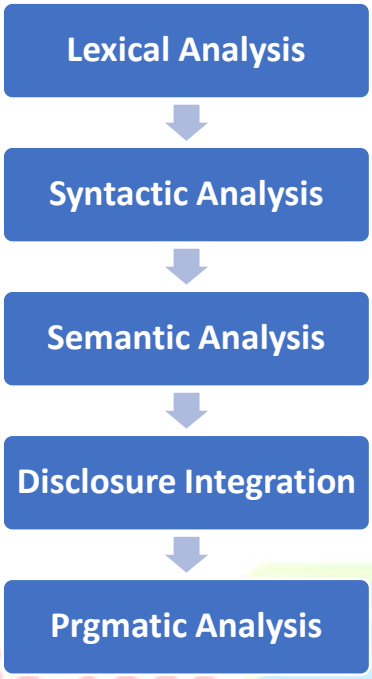
- One input can mean different meanings.

- Many inputs can mean the same thing.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Steps in NLP

There are generally five steps in Natural Language Processing:



1. **Lexical Analysis**: We have to analyze the structure of words. The collection of words and phrases in a language is a lexicon of a language. It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.

2. **Syntactic Analysis (Parsing)**: It involves analysis of words in the sentence for grammar and arranging words in a manner. That shows the relationship among the words. The sentence such as **"The school goes to boy" or "I eats rice"** is rejected by English syntactic analyzer.

3. **Semantic Analysis:** It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as **"hot ice-cream" or "I eat house"**.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

4. **Discourse Integration:** In this step, the meaning of any sentence depends upon the meaning of the previous sentence. In addition, it also brings about the meaning of immediately succeeding sentence. It is a sense of the context. The meaning of any single sentence depends upon the sentences that precedes it and also invokes the meaning of the sentences that follow it. Example: the word **"He"** and **"it"** in the sentence **"He likes it"** depends upon the prior discourse context. Means, who is he and what he likes is in the previous sentence. *("Kaji bought an AI book. He likes it.")* – Full Sentence

5. **Pragmatic Analysis:** In this step, data is interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge. It is a study of language use. Relation between context and meaning. Pragmatics are the study of words and their meaning in a language with concern to their context. Additionally, focuses on the meaning of words according to the context and their inferred meanings as well. Studies the intended or the inferred meaning as well. Example: **"Close the window?", "Do you know what time it is?"**

   Conversations between A and B:
   A: I have a 14 years old son.
   B: Well that's all right.
   A: I also have a dog.
   B: Oh, I'm Sorry!

Syntactic Processing

- Syntactic Processing is the step in which a flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning in the sentence.
- This process is called parsing.
- It plays an important role in natural language understanding systems for two reasons:
    - Semantic processing must operate on sentence constituent. If there is no syntactic parsing step, then the semantics system must decide on its own constituents. If parsing is done, on the other hand, it constrains the number of constituents that semantics can consider.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

- Syntactic parsing is computationally less expensive than is semantic processing. Thus, it can play a significant role in reducing overall system complexity.

- Although it is often possible to extract the meaning of a sentence without using grammatical facts, it is not always possible to do so. Consider the examples:
  - The satellite orbited Mars.
  - Mars orbited the satellite.

- Almost all the systems that are actually used have two main components:
  - **A declarative representation, called a grammar**, of the syntactic facts about the language.
  - **A procedure, called parser**, that compares the grammar against input sentences to produce parsed structures.

Grammars and Parsers

DEEPAK BHATTA

✓ In Context of NLP, Parsing implies analyzing a sentence syntactically to assign syntactic tags (subject, verb, object etc.) to provide constituent structure (noun phrase, verb phrase etc.) or to characterize the syntactic relations between two words.

✓ Parsing technique is further divided into **rule-based parsing** and **statistical parsing**.

Rule-based parsing:

- In rule-based parsing, syntactic structure of language is provided in form of linguistics rules which can be coded as production rules that are similar to context free rules.
- Production rules are defined using non-terminal and terminal symbols.

Statistical parsing:

- Require large corpora and linguistic knowledge is represented as statistical parameters or probability.

✓ Once the grammar rules are defined, a sentence is parsed using the grammar and tree kind of structure is built, if sentence is syntactically correct. This tree is called parse tree.

✓ Parsing can be done in two methods: **top-down parsing** and **bottom up parsing**.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

i. **Bottom-up parsing:** we start with the words in the sentence and apply grammar rules in the backward direction until a single tree is produced whose root matches with start symbol.

ii. **Top-down parsing:** we start with start symbol and apply grammar rules in forward direction until the terminal symbol of the parse tree corresponds to the word in the sentence.

- ✓ The choice between these two approaches is similar to the choice between forward and backward reasoning in other problem-solving tasks.
- ✓ The most important consideration is the branching factor. Is it greater going backward or forward?
- ✓ Sometimes these two approaches are combined to a single method called "bottom-up parsing with top-down filtering".

Consider the simple context free grammar for English language.

Rules	Dictionary Words
$< S > \; -> \; < NP >< VP >$	$< Det > \; -> \; a \mid an \mid the \mid this$
$< NP > \; -> \; < Adj >< N >$	$< N > \; -> \; girl \mid apple \mid john$
$< NP > \; -> \; < Det >< N >$	$< Adj > \; -> \; cute \mid smart$
$< NP > \; -> \; < Det >< Adj >< N >$	$< V > \; -> \; sings \mid eats \mid ate$
$< NP > \; -> \; < Det >< N >< PP >$	$< P > \; -> \; in \mid on \mid to \mid into$
$< VP > \; -> \; < V >$	$< Aux > \; -> \; is \mid will \mid do$
$< VP > \; -> < V >< NP >$	
$< VP > \; -> < V >< NP >< PP >$	
$< PP > \; -> < P>< NP >$	

- ✓ The symbol $->$ is used for 'defined as'.
- ✓ Vertical bar | for alternative definitions (OR)
- ✓ **S** for Sentence
- ✓ **N** for Noun
- ✓ **V** for Verb
- ✓ **Adj** for Adjective
- ✓ **NP** for Noun Phrase
- ✓ **VP** for Verb Phrase
- ✓ **P** for Preposition
- ✓ **PP** for Preposition Phrase

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Hits:

- ✓ Each Noun belongs to Noun Phrase.
- ✓ Verb cannot be Noun Phrase.
- ✓ Before the Verb Phrase all are Noun Phrase.
- ✓ After the Verb Phrase all are Verb Phrase.

The simple sentence recognized by this grammar is:

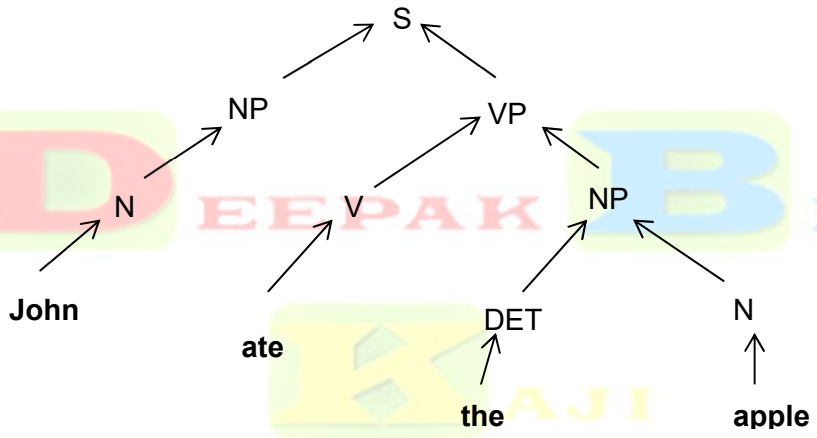
a) John ate the apple.

b) The cute girl ate an apple.

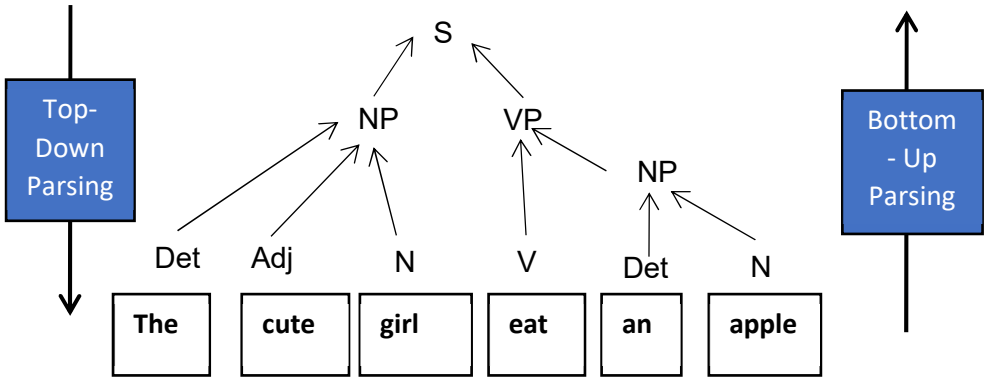
c) The cute girl sings a song.

A Parse Tree

i. John ate the apple.



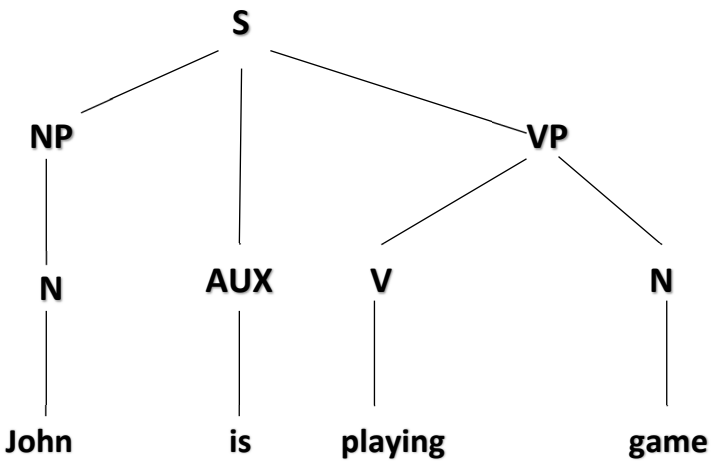
ii. The cute girl ate an apple



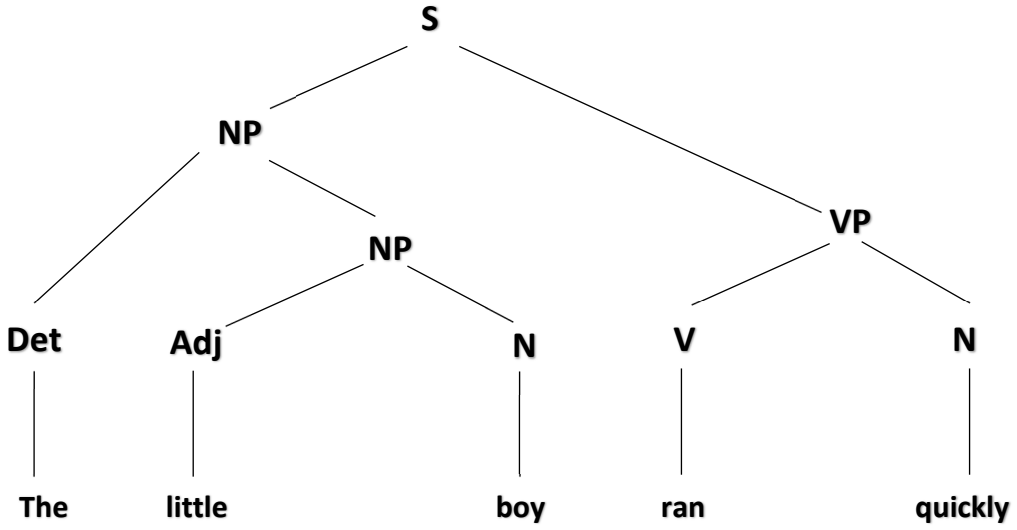
© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

iii. **The cute girl sings a song.** *(Do it Yourself)*

iv. **John is playing game.**



v. **The little boy ran quickly.**



[Just change the arrow for bottom-up or top down parsing, if it is asked in exam]

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

6.3.1 Levels of Analysis: Phonetic, Syntactic, Semantic, Pragmatic

Natural Language Processing — Terminologies

- **Phonetics/Phonology:** It's a study of organizing sound. How words are pronounced in terms of sequences of sound.
- **Morphology:** Basically, it's a study of the construction of words from primitive meaningful units. Individual words are analyzed into their components and non-word tokens such as punctation are separated from words.
- **Morpheme:** As we can say that it's a primitive unit of meaning in a language.
- **Syntax:** In this, we have to arrange words to make a sentence. Also, involves determining the structural role of words. That is in the sentence and in phrases.
- **Semantics:** It defines the meaning of words. Moreover, how to combine words into meaningful phrases and sentences.
- **Pragmatics:** It deals with use and understanding sentences in different situations. Also, defines how the interpretation of the sentence is affected.
- **World Knowledge:** It includes general knowledge about the world.
- **Discourse:** It deals with how the immediately preceding sentences can affect the interpretation of the next sentence.

6.4 Introduction to Machine Vision

Artificial Intelligence (AI) is one of the most coveted technologies across the world. Since **John McCarthy** coined the term **'Artificial Intelligence'** for the first time, this technology has been unlocking possibilities for the mankind almost every single day. In the years to come, AI is sure to change the way we look at things; especially machines or devices.

In short, AI is enabling machines to analyze and think just like we humans do. Unlike before, now machines can think on their own, and respond accordingly. But this is not the end!

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

As a technology, AI is making advancements and its presence felt in almost all the verticals. Machine Vision is one among the newest trends AI that would surely make us rethink on the way we look at machines today.

What is Machine Vision?

Machine vision (MV) is an integrated mechanical-optical-electronic-software technology which makes use of optical instrumentation, digital video, electromagnetic sensing, mechanics and image processing technology. The technology's goal is optical and non-contact sensing to receive and analyze a real image in order to provide more information. Machine vision technology is widely used in monitoring and controlling a wide range of applications.

In technical terms, Machine Vision (MV) can be defined as a technology that enables a machine to see; to inspect, identify, and evaluate objects at rest or in motion. Though this may sound quite similar to the surveillance cameras, it should be mistaken as a similar one.

While surveillance cameras mostly work on image processing, Machine Visions enables cameras to detect the slightest of motions; may be by a micro millimeter, to recognize facial differences even at almost zero-light exposure, identify faces accurately, and evaluate still and motion images within a fraction of time.

DEEPAK BHATTA

But Why Do We Need Machines to See?

Human vision is limited to certain conditions and parameters. Usually, human vision is limited within a wavelength range of 380-740 nanometers. Anything below or above this range cannot be picked up by the human eye. But with machines, the range of visibility wavelength can be extended much further than the human range. In addition to increasing vision limit, this would also ensure better visibility, precision, and faster processing.

Now, combining this vision with Artificial Intelligence, one can match numerous patterns to identify objects in real-time. For example, packaging errors in packaging industry; even the minutest one that might not be visible with naked human eyes, can be identified by means of Machine Vision (MV). Thus, process efficiency can be enhanced with the correct implementation of MV in the production line.

Another great example would be integrating the surveillance cameras with Machine Vision to detect any unwanted person around one's premises. Though surveillance cameras can do this job, integrating them with MV would add a newer edge to it by enabling the cameras to detect trespassers, compare the images received with millions of faces in the data base, and send the information thus received to the property owner. With Machine Vision (MV), surveillance cameras can even detect the closest-matching figures even from silhouettes. Thus, a Machine Vision system can redefine and bolster the traditional security or surveillance methods.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Some Applications of Machine Vision (MV) Systems:

In Defense:

Drones are one of the most relied devices in defense sector. Equipping these drones with Machine Vision would provide information about the slightest of motion to the forces. Furthermore, the MV systems can also detect enemy movements, help to detect landmine fields, or any kind of ambush from the other side. The same can also be used to conduct real-time surveillance across locations that could not be accessed or require much effort.

In Robotics:

At present, almost all the rovers used for space research are laced with Machine Vision Systems; especially the ones used for Mars exploration. Unlike before, these modern-day rovers can find their own ways, detect obstacles on the way, recourse their way if required, and most importantly, they relay real-time images and videos to the space station. MV systems used in these rovers can even help the scientists detect water, keep a close eye on the temperature level, identify the available gases across those atmospheres, and lot more even while sitting millions of miles away from the rovers.

DEEPAK BHATTA

In Real Estate:

Security is one of the most commonly sought-after parameters among the real estate buyers. Developers these days, are using MV systems to ensure better and advanced security around their properties. In addition to detecting the visitors' faces, these MV Systems can predict the probable actions or intentions of the visitors by matching their course of action across millions of patterns, and notify the authorities in case there are any threats detected.

In Banking:

Face detection is already one of the most effective technology used in banking sector. You can read our blog on 'Face Scanning' here. Taking face scanning a step further is the Machine Vision technology. With this advanced AI-technology, banks can now not only scan the faces, but also match or compare the respective faces with millions of others and detect any of the slightest fraudulent activities if any.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

6.5 Current trends and the future

ANI: Today's Artificial Intelligence

AI that is available right now: it's called **Artificial Narrow Intelligence** (ANI). ANI, also known as applied AI, is able to perform domain-specific reasoning tasks and problem solving, such as:

Speech and Text Recognition: Thanks to Natural Language Processing (NLP), a subfield of AI, program computers can process and analyze human (natural) languages, making possible the interactions between computers and humans in natural language. One of the most common applications of NLP is in voice assistants**: Amazon's Alexa, Apple's Siri, Microsoft's Cortana, etc.**

Computer Vision: The ability of a computer to process, analyze and understand digital images through the transformation of the visual input. The applications of computer vision are various: object or facial recognition, image restoration, motion analysis, etc.

Autonomous Vehicles: Autonomous vehicles, like these being developed by Tesla. Self-driving cars, although yet not so common on our roads, are an excellent example of the impact of AI on everyday life. Multiple AI subfields must be put together in order to make a vehicle that will replace human-driven cars: deep learning, speech recognition, voice search, image recognition and processing, motion detection, etc.

Product Recommendation Systems: Product recommendation systems, widely used by ecommerce shops. AI-powered recommendation engines analyze consumer behavior and preferences to deliver accurate, personalized product recommendations.

Ecommerce giants have successfully implemented AI recommendations in their marketing strategies: for instance, 35% percent of Amazon's revenue is generated by its recommendation's engine, and Zalando uses content-based and collaborative filtering algorithms to provide users with tailor-made product recommendations.

These problems require processing huge amount of data, something impossible for a human brain, but easily done for a well-trained AI algorithm. Nevertheless, it doesn't mean that a computer program performing these tasks is fully intelligent, as an algorithm built and trained for one purpose won't be able to perform well in a completely different task. This limitation

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

makes it hard to recognize current AI applications as a true AI. Actually, in 2017, intelligence tests conducted with some popular and publicly available ANI such as Google AI and Siri revealed that these AI achieved a score of about 47 IQ points, which is a level of a six-year-old child.

However, the ANI is universally acknowledged to be only one, least developed of the possible types of AI. What are the other types and how intelligent can they become?

AGI: The Future of Artificial Intelligence

The increasing computing power of modern computers and vast amount of data available to feed the algorithms allow assuming that we're on the way to reach the next levels of AI and machine learning development.

The next-level AI that is believed to appear in the future is known as Artificial General Intelligence (AGI) Right now, there are no AGI solutions available on the market, which means that at the moment ANI is the only type of AI that the humanity is able to build, therefore any other AI types are purely theoretical. On the other hand, lots of resources are spent every year on research, and there are many theories on how the advanced types of AI can be built, and although with no success yet, it gives hope that someday it will be achieved.

In short, Artificial General Intelligence (also known **as Deep AI or Strong AI**) is believed to be able to mimic human intelligence and behavior so well that it will become indistinguishable from the human mind in terms of learning and understanding any task, therefore successfully surpassing Turing test. AGI wouldn't be limited to performing specific tasks, like ANI is, and could be used for multiple purposes. The concept of human-like intelligent machines can feel disturbing, and indeed, it has caused concerns: in 2015, an open letter on artificial intelligence was signed by a number of scientists and researchers, among them Stephen Hawking and Elon Musk. The letter emphasized the importance of using the increasingly advanced AI systems for the societal benefit, and researching "how to reap its benefits while avoiding potential pitfalls", in other words: avoiding potential dangers for the humanity.

However, even that there is a lot of effort and resources put into AGI research, it's not expected to be built in the next couple of years. In the most optimistic scenario, there are decades more research required to develop a true AI.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

XAI: eXplainable Artificial Intelligence

Another future AI trend that is being talked about is the XAI – eXplainable Artificial Intelligence. The term hints that this type of AI would be able to explain itself. Nowadays, there are many industries and use cases where machine learning methods give results that aren't easy to interpret. For instance, data scientist sometimes can struggle to understand what has led AI to make a specific decision, let alone explain it to another person, for example the customer. The concept of XAI, an AI that would explain its own decision-making process, was born from the need of being able to interpret the algorithms' way of reasoning, as it also could allow to improve it and avoid its mistakes.

The main obstacle in the development of XAI is that neural networks mostly are not so easy to explain, as the layers return weights matrices. There are already some ideas being implemented, like heat maps for the weight matrices that improve neural networks interpretability. For sure more research will be done in this area in the upcoming years.

The evolution of deep learning

Deep learning surely will be still a hot topic in the next decades and one of the most significant machine learning trends, as there is still a lot of work that can be done in this field. We see new methods introduced each year, that perform better than other methods or allow to use deep learning in brand new areas. You can stay up to date with the recent developments and breakthrough machine learning trends and methods thanks to events such as the annual NeurIPS conference.

What are some of the most interesting trends in deep learning?

Capsule neural networks, or CapsNet: it's a type of neural network that can improve hierarchical relationships models to better mimic human neurons.

FractalNet: it's an attempt to build ultra-deep neural network as an alternative to deep residual learning.

Deep reinforcement learning: this ML trend mixes deep learning (i.e. deep neural networks) with reinforcement learning (Q-learning, actor critic, etc.) in order to create powerful algorithms that might be able to solve complex problems.

THE END

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

PU Questions Answers

MYCIN:

MYCIN was based on backward chaining and could identify various bacteria that could cause acute infections. It could also recommend drugs based on the patient's weight.

Here, we will explain the working of an expert system by taking an example of MYCIN ES. Below are some steps to build an MYCIN:

- Firstly, ES should be fed with expert knowledge. In the case of MYCIN, human experts specialized in the medical field of bacterial infection, provide information about the causes, symptoms, and other knowledge in that domain.
- The KB of the MYCIN is updated successfully. In order to test it, the doctor provides a new problem to it. The problem is to identify the presence of the bacteria by inputting the details of a patient, including the symptoms, current condition, and medical history.
- The ES will need a questionnaire to be filled by the patient to know the general information about the patient, such as gender, age, etc.
- Now the system has collected all the information, so it will find the solution for the problem by applying if-then rules using the inference engine and using the facts stored within the KB.
- In the end, it will provide a response to the patient by using the user interface.

SOM (Self-Organizing Map) Explanation in Short:

- SOM is algorithm that projects high-dimensional data onto a two-dimensional map.
- The projection preserves the topology of the data so that similar data items will be mapped to nearby locations on the map.
- SOM still have many practical applications in **pattern recognition, speech analysis, industrial and medical diagnostics, data mining**.
- Large quantity of goods quality representative training data required.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

[Fall 2015]

Q.6 b) What is neural network? Define Perceptron. How is winning node chosen in Kohonen network, illustrate with following dataset: **[8]**

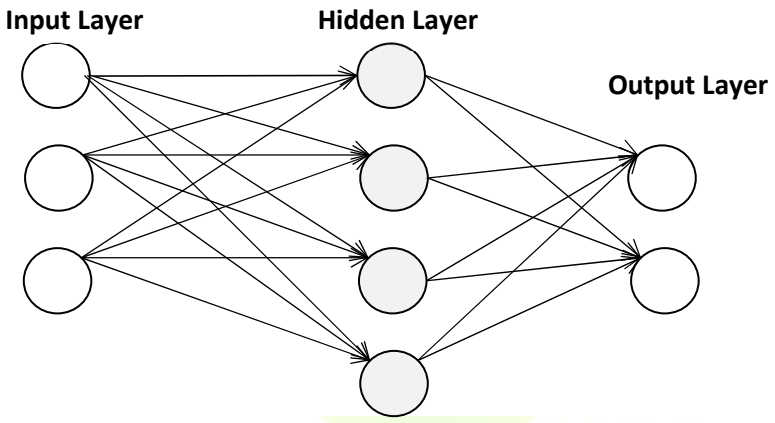
$$X = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix} \quad W_1 = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix} \quad W_2 = \begin{bmatrix} 0.27 \\ 0.81 \end{bmatrix} \quad W_3 = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix}$$

Where,

X two-dimensional input vector presented in 3-Dimensional Kohonen Network.

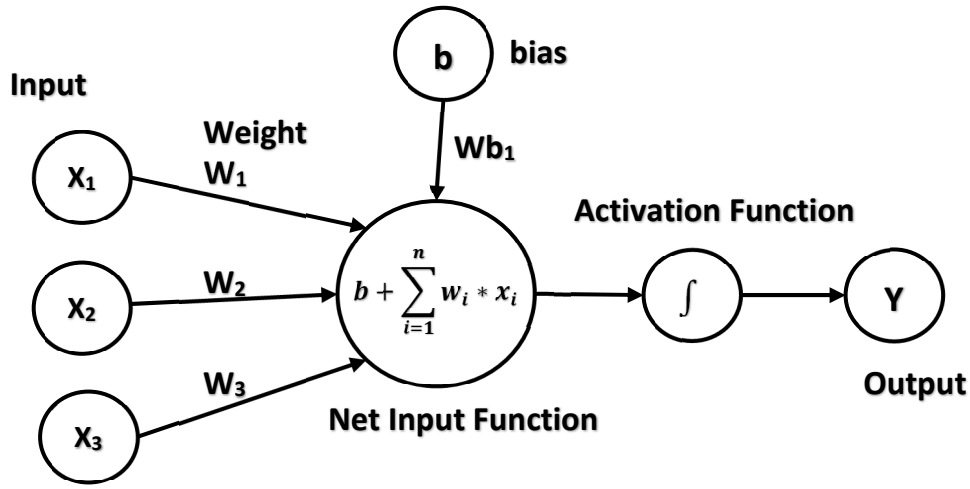
$$W_i = Weight\ Vectors, i = 1, 2, 3$$

Ans: A neural network is either a system software or hardware that works similar to the tasks performed by neurons of human brain. Neural networks include various technologies like deep learning, and machine learning as a part of Artificial Intelligence (AI). Artificial neural networks (ANN) is the key tool of machine learning. These are systems developed by the inspiration of neuron functionality in the brain, which will replicate the way we humans learn.



A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data. Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.

A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Sol^n:

Given, the 2-Dimensional input vector X is represented to the neuron Kohonen network,

$$\boldsymbol{X} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix}$$

The intial weights, W_j are given by,

$$\boldsymbol{W_1} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \boldsymbol{W_2} = \begin{bmatrix} \mathbf{0.27} \\ \mathbf{0.81} \end{bmatrix} \quad \boldsymbol{W_3} = \begin{bmatrix} \mathbf{0.43} \\ \mathbf{0.21} \end{bmatrix}$$

Now, find the wining (Best-Matching Neuron) j_x using the Minimum Euclidean Distance:

$$\boldsymbol{D(1)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{11}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{21}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.27})^2 + (\mathbf{0.12} - \mathbf{0.81})^2} = \mathbf{0.73}$$

$$\boldsymbol{D(2)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{12}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{22}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.42})^2 + (\mathbf{0.12} - \mathbf{0.70})^2} = \mathbf{0.59}$$

$$\boldsymbol{D(3)} = \sqrt{(\boldsymbol{X_1} - \boldsymbol{W_{13}})^2 + (\boldsymbol{X_2} - \boldsymbol{W_{23}})^2}$$

$$= \sqrt{(\mathbf{0.52} - \mathbf{0.43})^2 + (\mathbf{0.12} - \mathbf{0.21})^2} = \mathbf{0.13}$$ **(Minimum Euclidean Distance)**

$\therefore$ *Neuron 3 is the winner and its weight vector W_3 is updated according to the competitive learning rule.*

Assume, Learning rate $\alpha = 0.1$

The weights on the winning unit/neuron are updated:

$$\boldsymbol{W_{ij}(new)} = \boldsymbol{W_{ij}(old)} + \boldsymbol{\alpha}[\boldsymbol{X_i} - \boldsymbol{W_{ij}(old)}]$$

$$\begin{bmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.52} \\ \mathbf{0.12} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{X_{13}} \\ \boldsymbol{X_{23}} \end{bmatrix} = \begin{bmatrix} \mathbf{0.43} \\ \mathbf{0.21} \end{bmatrix}$$

$$\boldsymbol{W_{13}(new)} = \boldsymbol{W_{13}(old)} + \mathbf{0.1}[\boldsymbol{X_1} - \boldsymbol{W_{13}(old)}]$$
$$\boldsymbol{W_{13}(new)} = \mathbf{0.43} + \mathbf{0.1}[\mathbf{0.52} - \mathbf{0.43}] = \mathbf{0.44}$$

$$\boldsymbol{W_{23}(new)} = \boldsymbol{W_{23}(old)} + \mathbf{0.1}[\boldsymbol{X_2} - \boldsymbol{W_{23}(old)}]$$
$$\boldsymbol{W_{23}(new)} = \mathbf{0.21} + \mathbf{0.1}[\mathbf{0.12} - \mathbf{0.21}] = \mathbf{0.20}$$

The weight vector W_3, of the winning neuron 3 becomes closer to the input vector X with each iteration/epoch.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

[Fall 2016]

Q.6.a) Why is Natural Language understanding is difficult? Explain about Morphological Analysis with relevant example.

Ans: The primary problem with natural language processing is the ambiguity of language. There are a number of levels at which ambiguity may occur in natural language (of course a single sentence may include several of these levels). First a sentence or phrase may be ambiguous at a syntactic level. Syntax relates to the structure of the language, the way the words are put together. Some word sequences make valid sentences in a given language, some do not.

However, some sentence structures have more than one correct interpretation. In the first place, these are syntactically ambiguous. Secondly, a sentence may be ambiguous at a lexical level. The lexical level is the word and ambiguity here occur when a word can have more than one meaning. Thirdly, a sentence may be ambiguous at a referential level.

This is concerned with what the sentence (or a part of it) refers to. Ambiguity occurs when it is not clear what the sentence is referring to or it may legally refer to more than one thing. Fourthly, a sentence can be ambiguous at a semantic level, that is, at the point of the meaning of the sentence.

Morphological Analysis:

The study of word formulation – how words are built up from smaller pieces. Identification, analysis, and description of the structure of a given language's MORPHEMES and other linguistic units, such as root words, affixes, part of speech, intonations and stresses or implied context. Examples:

- Washing = Wash + ing
- Browser = Browse + er
- Rats = Rat + s

Individual words are analyzed into their components and non-word tokens such as punctuation are separated from the words.

Tries to extract root word from decline or inflectional form of word after removing suffices and prefixes. Ex: getting the root "push" from declined from pushes, pushed, pushing, etc.

Assign appropriate syntactic categories such as noun, verb, adjective etc. to all words in the sentence.

Types of Morphology:

a. **Inflectional Morphology:** Modification of a word to express different grammatical categories. Examples: Cats, Men etc.
b. **Derivational Morphology:** Creation of a new word from existing word by changing grammatical category. Examples: Happiness, Brotherhood etc.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

[Fall 2018]

Q.5.b) Solve the AND function using Hebb network. **[8]**

Ans: Hebb Network:

- This rule, one of the oldest and simplest, was introduced by **Donald Hebb** in his book The Organization of Behavior in **1949**. It is a kind of feed-forward, unsupervised learning.
- Basic Concept − This rule is based on a proposal given by Hebb, who wrote − "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."
- From the above postulate, we can conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.
- Mathematical Formulation − According to Hebbian learning rule, following is the formula to increase the weight of connection at every time step.

$$\Delta W_{ji}(t) = \alpha X_i(t).Y_j(t)$$

Here,

$\Delta W_{ji}(t) =$ increment by which the weight of connection increases at time step **t**

$\alpha =$ the positive and constant learning rate

$X_i(t) =$ the input value from pre-synaptic neuron at time step **t**

$Y_j(t) =$ the output of pre-synaptic neuron at same time step **t**

In short:

- Hebb learning rule is the simplest one.
- Hebb rule is suited for bipolar data (+1, -1).
- The learning in the brain is performed by the change in the synaptic gap.
- When an axon of cell A is near enough to excite cell B and repeatedly keep firing it, some growth process takes place in one or both cells.
- According to Hebb rule, weight vector is found to increase proportionately to the product of the input and learning signal.

$$W_i(new) = W_i(old) + X_i Y$$

- Here learning signal is equals to neuron output.

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Hebb Network Algorithm:

Step 0: Initialize all weight and bias to zero.

$$W_i = 0, \quad b = 0, \quad for\ i = 1\ to\ n$$

Step 1: For each of the training sample **s:t** *[i.e. pair (s, t)]*, do step 2 to 4

s is the input pattern, **t** is the target output of the sample

Step 2: Set **s** to input units

$$X_i = S_i \qquad for\ i = 1\ to\ n$$

Step 3: Set **Y** to the target **t**

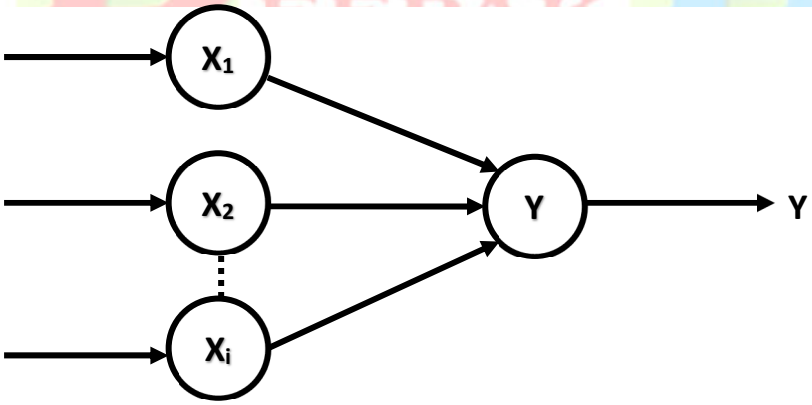
$$Y = t$$

Step 4: Update the weight and bias by applying Hebb Rule:

$$W_i(new) = W_i(old) + X_i * Y \qquad for\ i = 1\ to\ n$$

$$b(new) = b(old) + Y$$

The algorithm requires only one pass through this training set



Solve the **AND function** using Hebb network.

Hebb rule is suited for bipolar unit (+1, -1). So, we do +1 for +1 and 0 for -1

[METHOD – 1] First draw the table like below:

Input Unit			Output	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t			
1	1	1	1			
1	-1	1	-1			
-1	1	1	-1			
-1	-1	1	-1			

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Initially, we have **W_i=0 and b=0**

Now, calculate $\boldsymbol{W_i(new)\ for\ i = 1\ to\ n}$

$\boldsymbol{W_i(new) = W_i(old) + X_i * Y}$

$\boldsymbol{W_1(new) = W_1(old) + X_1 * Y}$

Initially $\boldsymbol{W_i = 0}$

$W_1(old)$	+	X_1	*	Y	=	$W_1(new)$	$W_1 = 0$
0	+	1	*	1	=	**1**	$W_1 = 1$
1	+	1	*	-1	=	**0**	$W_1 = 0$
0	+	-1	*	-1	=	**1**	$W_1 = 1$
1	+	-1	*	-1	=	**2**	

Value of W_1 *is updated from* 0 *to* 1

Value of W_1 *is updated from* 1 *to* 0

Value of W_1 *is updated from* 0 *to* 1

DEEPAK BHATTA

Now, calculate $\boldsymbol{W_i(new)\ for\ i = 1\ to\ n}$

$\boldsymbol{W_i(new) = W_i(old) + X_i * Y}$

$\boldsymbol{W_2(new) = W_2(old) + X_2 * Y}$

Initially $\boldsymbol{W_i = 0}$

$W_2(old)$	+	X_2	*	Y	=	$W_2(new)$	$W_2 = 0$
0	+	1	*	1	=	**1**	$W_2 = 1$
1	+	-1	*	-1	=	**2**	$W_2 = 2$
2	+	1	*	-1	=	**1**	$W_2 = 1$
1	+	-1	*	-1	=	**2**	

Value of W_2 *is updated from* 0 *to* 1

Value of W_2 *is updated from* 1 *to* 2

Value of W_2 *is updated from* 2 *to* 1

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Now, calculate $\boldsymbol{b(new)}$

$\boldsymbol{b(new) = b(old) + Y}$

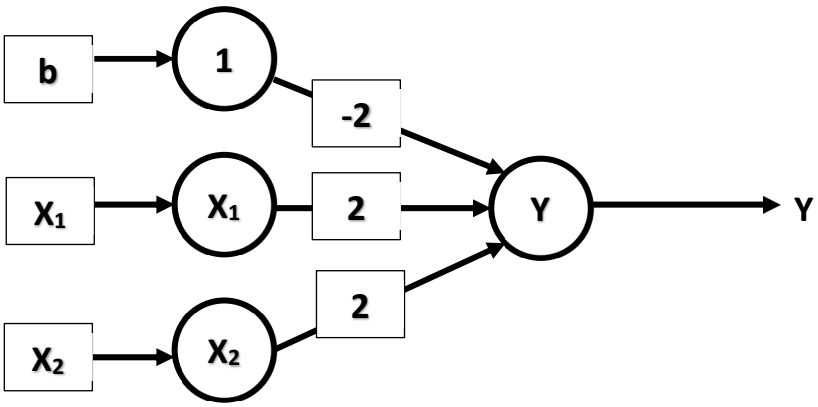
Initially $b = 0$

$b(old)$	+	Y	=	$b(new)$	$b = 0$	
0	+	1	=	**1**	$b = 1$	**Value of b is updated from 0 to 1**
1	+	-1	=	**0**	$b = 0$	**Value of b is updated from 1 to 0**
0	+	-1	=	**-1**	$b = -1$	**Value of b is updated from 0 to -1**
-1	+	-1	=	**-2**		

Now adjust the newly updated weight and bias into the table

Input Unit			Output	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t			
1	1	1	1	1	1	1
1	-1	1	-1	0	2	0
-1	1	1	-1	1	1	-1
-1	-1	1	-1	2	2	-2

Hebb Network for AND Function



© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

[METHOD – 2] First draw the table like below:

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1						
1	-1	1	-1						
-1	1	1	-1						
-1	-1	1	-1						

Initially, the weights are set to zero and bias is also set as zero.

$\mathbf{W_1 = W_2 = b = 0}$

Calculate,

$\Delta W_i = X_i Y$ **and**

$\Delta b = Y$ and fill the table

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1	1	1	1			
1	-1	1	-1	-1	1	-1			
-1	1	1	-1	1	-1	-1			
-1	-1	1	-1	1	1	-1			

For ΔW_1

$\Delta W_1 = X_1 * Y$

1	*	1	=	1
1	*	-1	=	-1
-1	*	-1	=	1
-1	*	-1	=	1

For ΔW_2

$\Delta W_2 = X_2 * Y$

1	*	1	=	1
-1	*	-1	=	1
1	*	-1	=	-1
-1	*	-1	=	1

For Δb

$\Delta b = Y$

1	=	1
-1	=	-1
-1	=	-1
-1	=	-1

Now, calculate new weights and new bias

$$W_i(new) = W_i(old) + \Delta W_i$$

$$b(new) = b(old) + \Delta b$$

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_2	New b
X_1	X_2	b=1	Y=t						
1	1	1	1	1	1	1			
1	-1	1	-1	-1	1	-1			
-1	1	1	-1	1	-1	-1			
-1	-1	1	-1	1	1	-1			

For $W_1(new)$

$W_1(new) = W_1(old) + \Delta W_1$

0	+	1	=	1
1	+	-1	=	0
0	+	1	=	1
1	+	1	=	2

Initial $W_1 = 0$

For $W_2(new)$

$W_2(new) = W_2(old) + \Delta W_2$

0	+	1	=	1
1	+	1	=	2
2	+	-1	=	1
1	+	1	=	2

Initial $W_2 = 0$

For $b(new)$

$b(new) = b(old) + \Delta b$

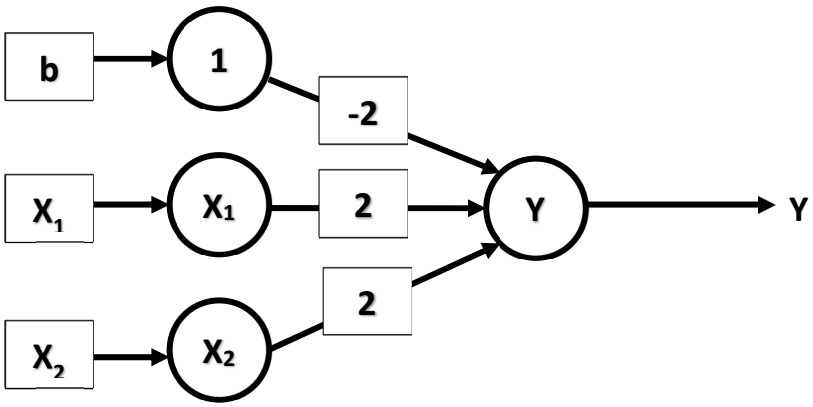
0	+	1	=	1
1	+	-1	=	0
0	+	-1	=	-1
1	+	-1	=	2

Initial b = 0

Final Table is:

Input Unit			Output	ΔW_1	ΔW_2	Δb	New W_1	New W_1	New b
X_1	X_2	b=1	Y=t						
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	2

Hebb Network for AND Function



© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

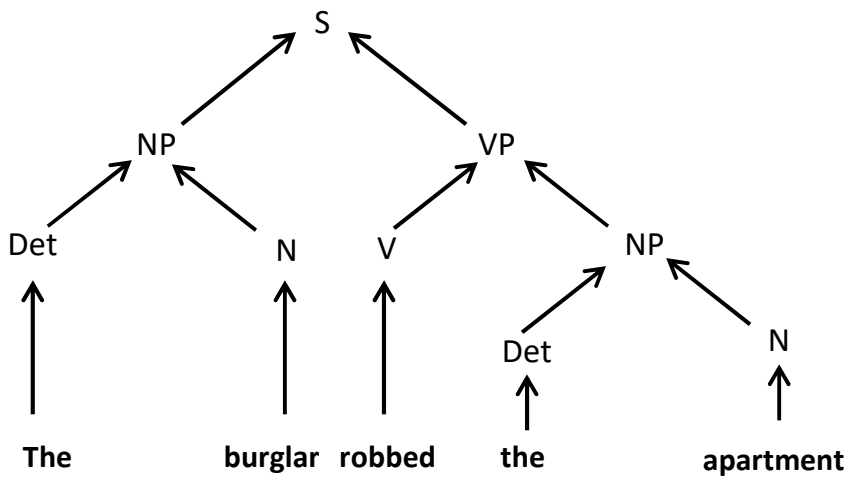
[Fall 2019]

Q.6.b) What do you mean by NLP? Construct a parse tree for the sentence **"The burglar robbed the apartment"**.

Ans: About NLP ***[Already discussed see above]***

Parse Tree for **"The burglar robbed the apartment".**

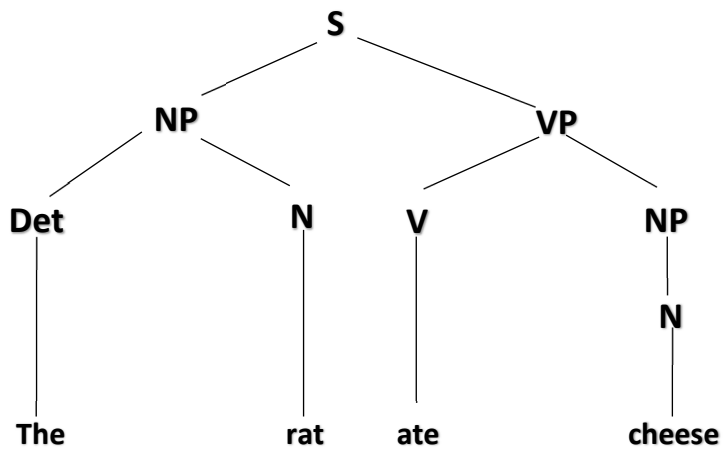
Rules	Dictionary Words
$< S > \;\; -> \;\; < NP >< VP >$	$< Det > \;\; -> \; a \,\|an\,\|the\,\|\,this$
$< NP > \;\; -> \; < Adj >< N >$	$< N > \;\; -> \; girl \,\|\, apple \,\|\, john$
$< NP > \;\; -> \; < Det >< N >$	$< Adj > \;\; -> \; cute \,\|\, smart$
$< NP > \;\; -> \; < Det >< Adj >< N >$	$< V > \;\; -> \; sings \,\|\, eats \,\|\, ate$
$< NP > \;\; -> \; < Det >< N >< PP >$	$< P > \;\; -> \; in \,\|\, on \,\|\, to\|\, into$
$< VP > \;\; -> \; < V >$	$< Aux > \;\; -> \; is \,\|\, will \,\|\, do$
$< VP > \;\; -> < V >< NP >$	
$< VP > \;\; -> < V >< NP >< PP >$	
$< PP > \;\; -> < P>< NP >$	



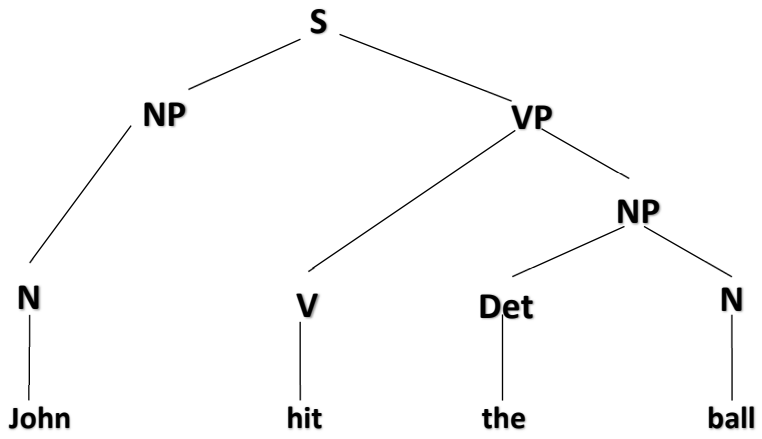
© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

Some Practice on Parse Tree:

i. The rat ate cheese.

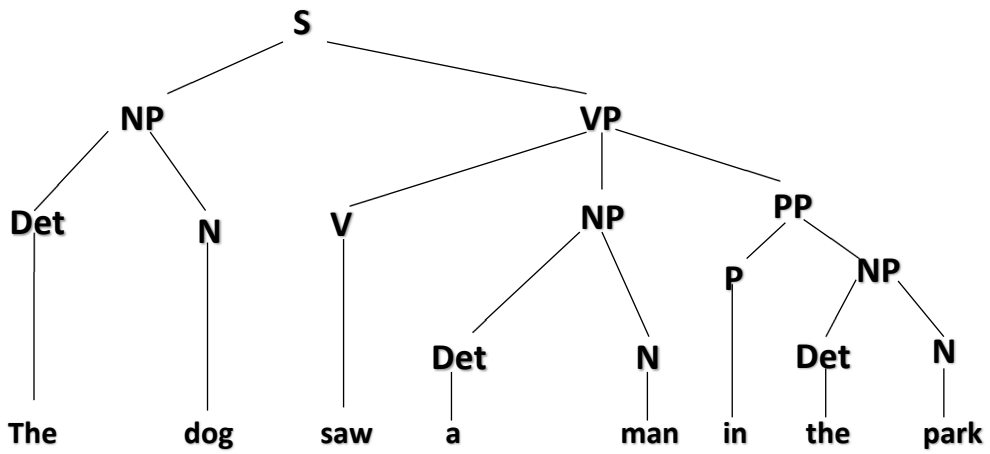


ii. John hit the ball.



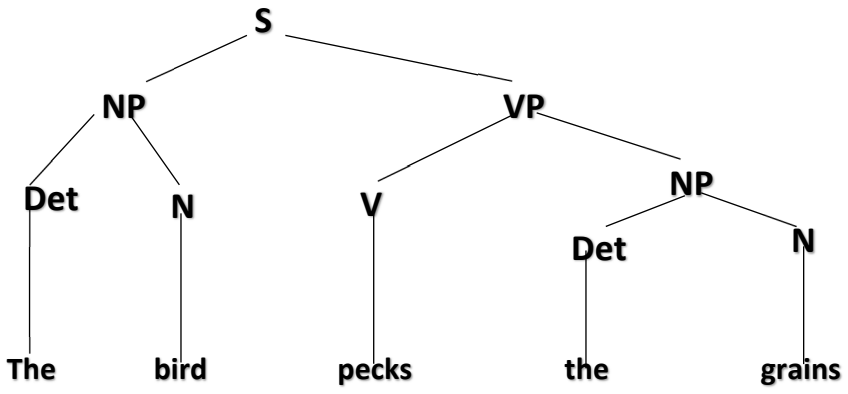
HATTA

iii. The dog saw a man in the park.



© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

iv. The bird pecks the grains.



 # What is an Activation Function? Why to use them?

Ans: Neural network activation functions are a crucial component of deep learning. Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network.

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

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

We know neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weight and bias of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases

---

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

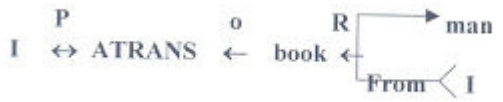
[Spring 2014]

Q.7) Write Short Notes on:

b) Conceptual dependencies:

- A Model of Natural Language understanding in Artificial Intelligence -Represent knowledge acquire from NL input.
- Conceptual Dependency (CD) theory was developed by Schank in 1973 to 1975 to represent the meaning of NL sentences.
- It helps in drawing inferences − It is independent of the language.
- CD representation of a sentence is not built using words in the sentence rather built using conceptual primitives which give the intended meanings of words.
- CD provides structures and specific set of primitives from which representation can be built.

I gave the man a book.
In CD from the above sentence can be represented with primitives as



It should be noted that this representation is same for different saying with same meaning.

For example − **I gave the man a book**, − **The man got book from me**, − **The book was given to man by me** etc.

In the above representation the symbols have the following meaning:

- ✓ Arrows indicate direction of dependency
- ✓ Double arrow indicates two may link between actor and the action
    - P indicates past tense
    - ATRANS is one of the primitive acts used by the theory. it indicates transfer of possession
    - O indicates the object case relation
    - R indicates the recipient case relation
    - D indicates destination

Conceptual dependency provides a str5ucture in which knowledge can be represented and also a set of building blocks from which representations can be built. A typical set of primitive actions are

ATRANS - Transfer of an abstract relationship (Eg: give)

PTRANS - Transfer of the physical location of an object (Eg: go)

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]

PROPEL - Application of physical force to an object (Eg: push)

MOVE - Movement of a body part by its owner (eg : kick)

GRASP - Grasping of an object by an actor (Eg: throw)

INGEST - Ingesting of an object by an animal (Eg: eat)

EXPEL - Expulsion of something from the body of an animal (cry)

MTRANS - Transfer of mental information (Eg: tell)

MBUILD - Building new information out of old (Eg: decide)

SPEAK - Production of sounds (Eg: say)

ATTEND - Focusing of sense organ toward a stimulus (Eg: listen)

A second set of building block is the set of allowable dependencies among the conceptualization describe in a sentence.

Artificial Intelligence	Machine Learning	Deep Learning
Artificial intelligence originated around 1950s.	Machine learning originated around 1960s.	Deep learning originated around 1970s.
AI represents simulated intelligence in machines.	Machine Learning is the practice of getting machines to make decisions without being programmed.	Deep Learning is the process of using Artificial Neural Networks to solve complex problems.
AI is a subset of Data Science.	Machine learning is a subset of AI & Data Science	Deep learning is a subset of Machine learning, AI & Data Science.
Aim is to build machines which are capable of thinking like humans.	Aim is to make machines learn through data so that they can solve problems.	Aim is to build neural networks that automatically discover patterns for feature detection.

THE END

© Deepak Bhatta Kaji (Lecturer)
Department of Computer Science – DEC [NAST]