

# Artificial Intelligence

BY

DR. ANUPAM GHOSH

DATE: 19.08.2023

Email: [anupam.ghosh@rediffmail.com](mailto:anupam.ghosh@rediffmail.com)

LinkedIn Profile:  
<https://www.linkedin.com/in/anupam-ghosh-1504273b/>

Research Profile:  
[https://www.researchgate.net/profile/Anupam\\_Ghosh14](https://www.researchgate.net/profile/Anupam_Ghosh14)

# What is Artificial Intelligence?

- ▶ The power of a machine to copy intelligent human behaviour

## Artificial Intelligence Tests: Turing Test

- ▶ Developed by Alan Turing
- ▶ Involves an interpreter, a human, and a computer.
- ▶ The computer and human have separate conversations with the interpreter.
- ▶ If the interpreter can't guess which is the computer or if the interpreter gets it wrong then the computer has Artificial Intelligence.

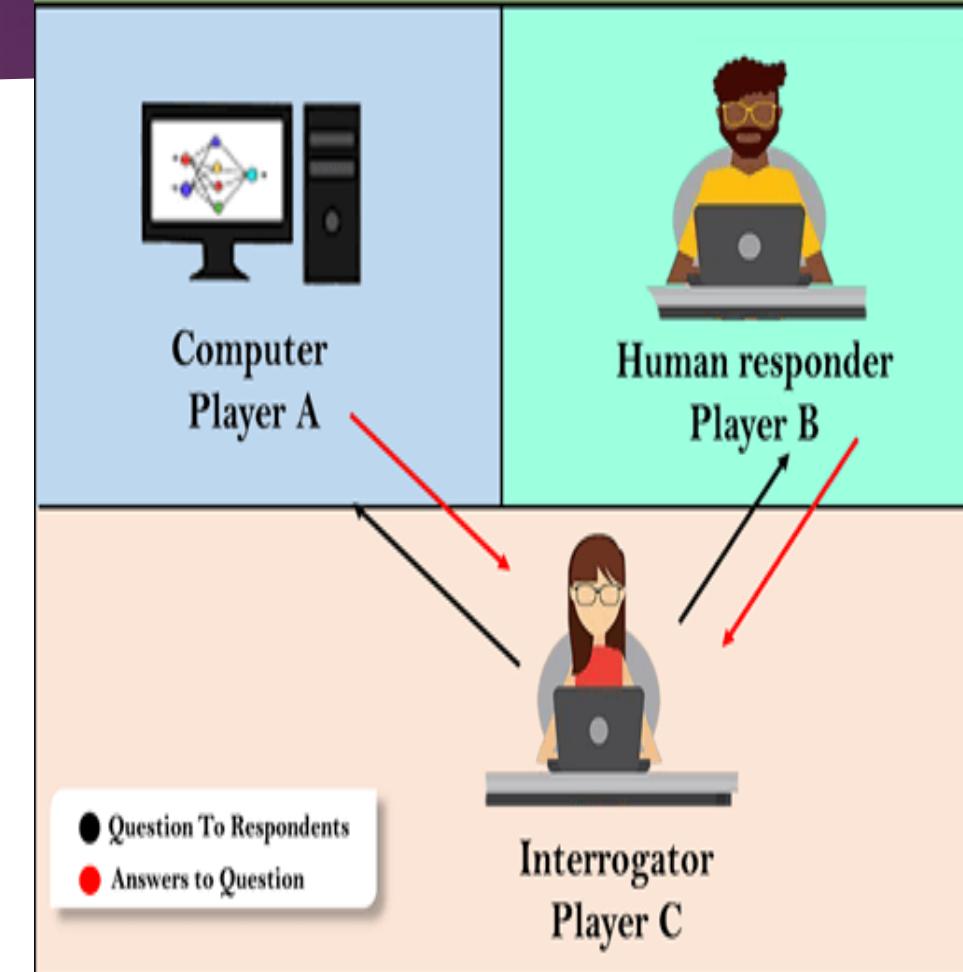
# Turing Test in AI

- ▶ 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 intelligent if it can mimic human response under specific conditions.
- ▶ 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 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.

## Turing Test



# 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.

# Intelligence - some definitions

- ▶ Intelligence:- ability to adapt oneself adequately to relatively new situations in life. (R. Pintner).
- ▶ Intelligence:- having learned or the ability to learn to adjust oneself to the environment. (Colvin)
- ▶ Intelligence:- the ability to carry out abstract thinking. (Terman)
- ▶ Intelligence:- innate general cognitive ability (Burt)
- ▶ Intelligence:- appropriate and adaptable behaviour in given circumstances. (Psihologija, group of authors, SK, Zagreb, 1992)
- ▶ Intelligence:- manifests itself only relative to specific social and cultural contexts. (J. Weizenbaum, 1975)

- ▶ A branch of computer science: Mathematical Sciences -> Computer Science -> Artificial Intelligence
- ▶ The branches of Artificial Intelligence (according to Association of Computing Machinery, ACM):
  - (1) General AI (cognitive modelling, philosophical foundations)
  - (2) Expert systems and applications
  - (3) Automated programming
  - (4) Deduction and theorem proving
  - (5) Formalisms and methods for knowledge representation
  - (6) Machine learning
  - (7) Understanding and processing of natural and artificial languages
  - (8) Problem solving, control methods, and state space search
  - (9) Robotics
  - (10) Computer vision, pattern recognition, and scene analysis
  - (11) Distributed artificial intelligence

# Goals of AI

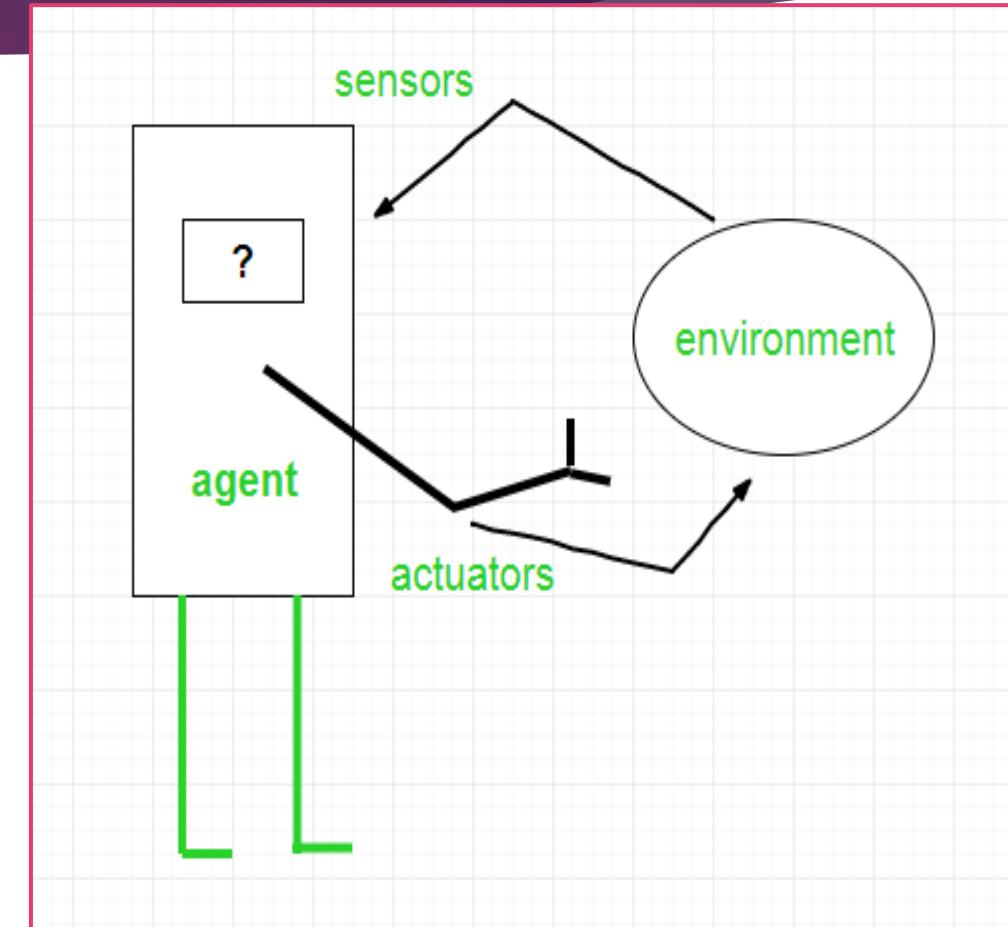
- ▶ To Create Expert Systems: The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users
- ▶ To Implement Human Intelligence in Machines: Creating systems that understand, think, learn, and behave like humans

# Agents in Artificial Intelligence

- ▶ Artificial intelligence is defined as the study of rational agents. A rational agent could be anything that makes decisions, as a person, firm, machine, or software. It carries out an action with the best outcome after considering past and current percepts(agent's perceptual inputs at a given instance). An AI system is composed of an agent and its environment. The agents act in their environment. The environment may contain other agents.

An agent is anything that can be viewed as :

- ▶ perceiving its environment through sensors and
- ▶ acting upon that environment through actuators



# Agent = Architecture + Agent Program

- ▶ Architecture is the machinery that the agent executes on. It is a device with sensors and actuators, for example, a robotic car, a camera, a PC. Agent program is an implementation of an agent function. An agent function is a map from the percept sequence(history of all that an agent has perceived to date) to an action.

## Examples of Agent:

- ▶ A software agent has Keystrokes, file contents, received network packages which act as sensors and displays on the screen, files, sent network packets acting as actuators.
- ▶ A Human-agent has eyes, ears, and other organs which act as sensors, and hands, legs, mouth, and other body parts acting as actuators.
- ▶ A Robotic agent has Cameras and infrared range finders which act as sensors and various motors acting as actuators.



# Types of Agents

Agents can be grouped into five classes based on their degree of perceived intelligence and capability :

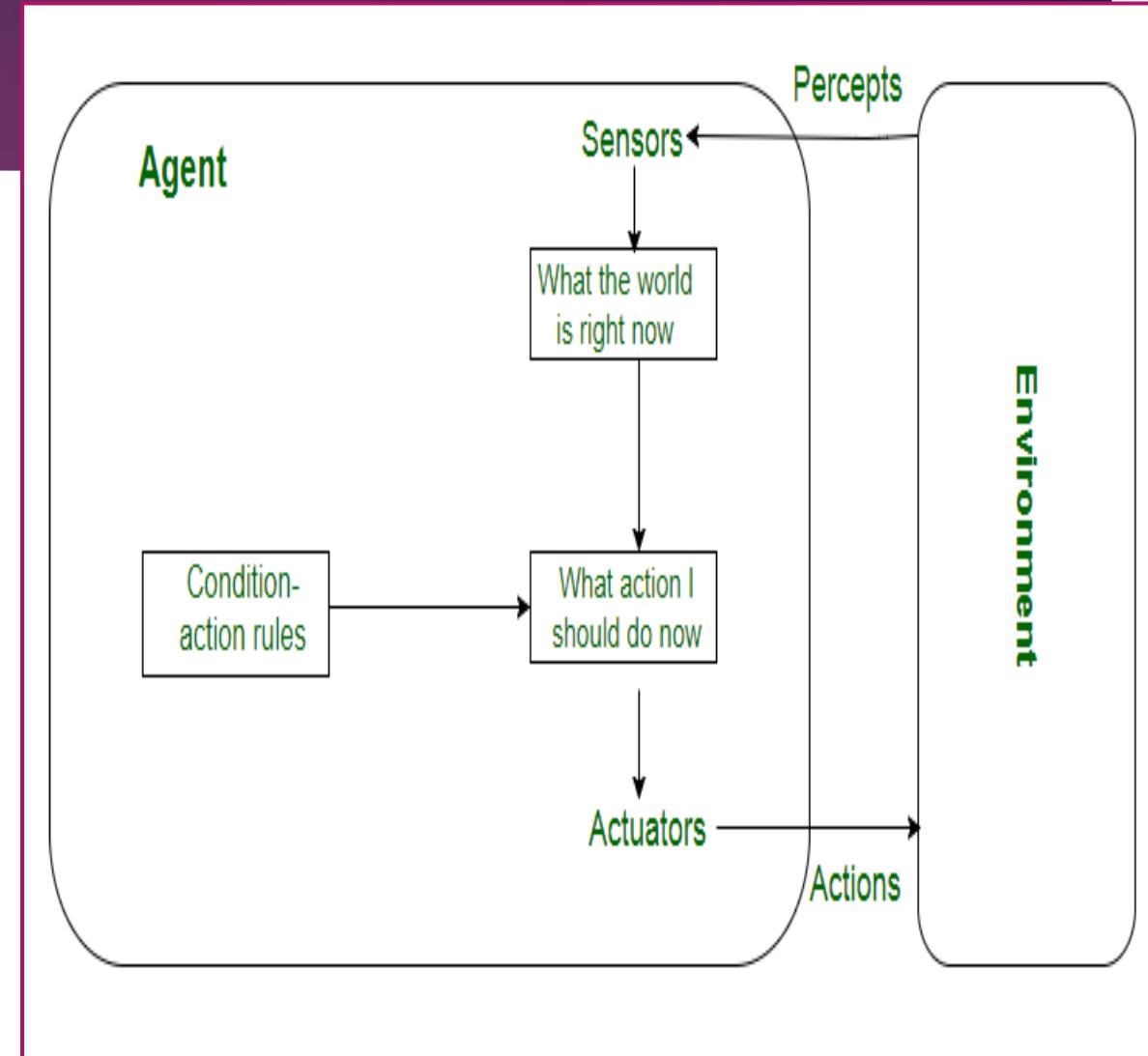
- ▶ Simple Reflex Agents
- ▶ Model-Based Reflex Agents
- ▶ Goal-Based Agents
- ▶ Utility-Based Agents
- ▶ Learning Agent

# Simple reflex agents

- ▶ Simple reflex agents ignore the rest of the percept history and act only on the basis of the **current percept**.
- ▶ Percept history is the history of all that an agent has perceived to date.
- ▶ The agent function is based on the **condition-action rule**. If the condition is true, then the action is taken, else not. For simple reflex agents operating in partially observable environments, infinite loops are often unavoidable. It may be possible to escape from infinite loops if the agent can randomize its actions.

## Problems with Simple reflex agents are :

- ▶ Very limited intelligence.
- ▶ No knowledge of non-perceptual parts of the state.
- ▶ Usually too big to generate and store.
- ▶ If there occurs any change in the environment, then the collection of rules need to be updated.

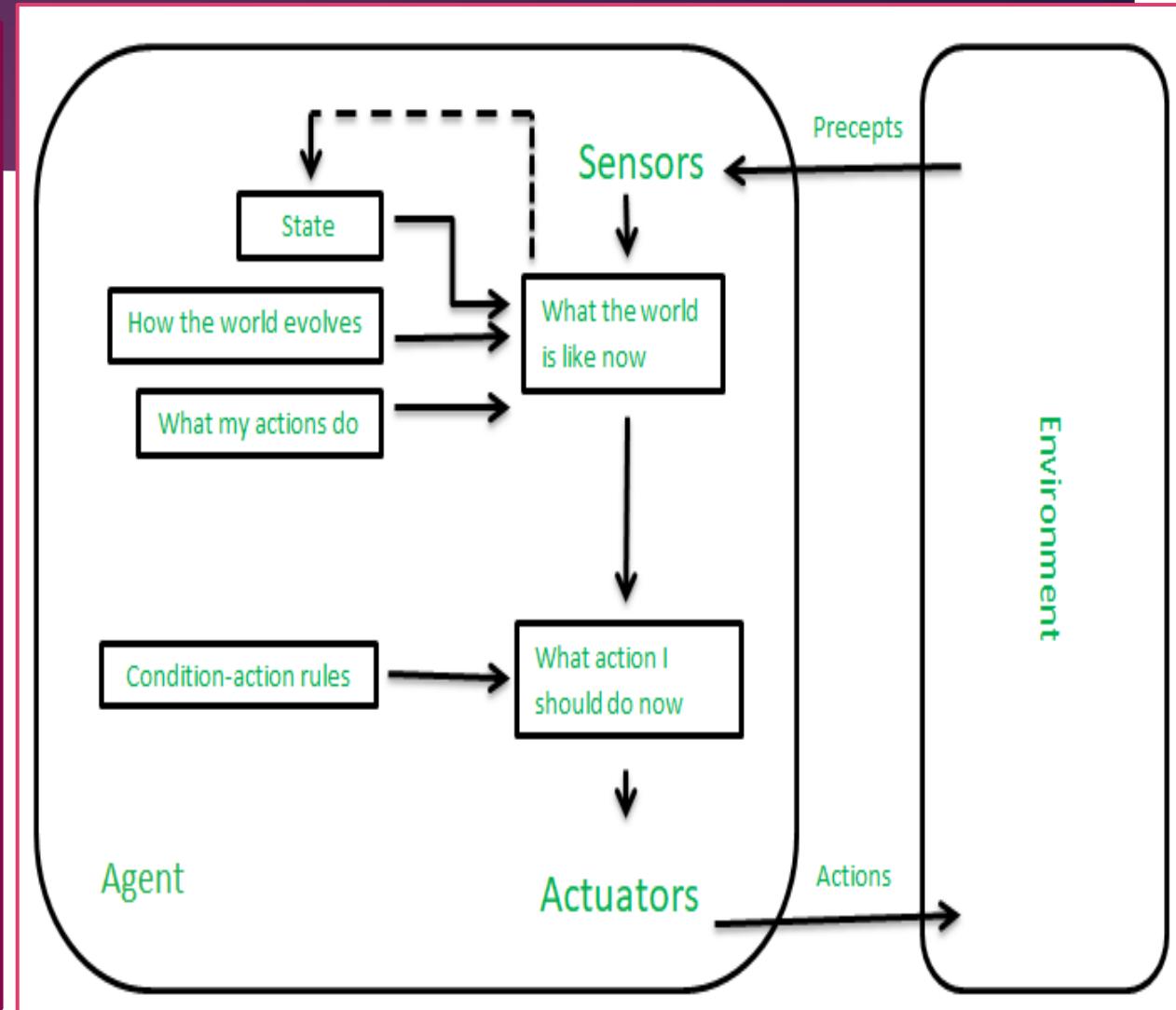


# Model-based reflex agents

- It works by finding a rule whose condition matches the current situation. A model-based agent can handle **partially observable environments** by the use of a model about the world. The agent has to keep track of the **internal state** which is adjusted by each percept and that depends on the percept history. The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.

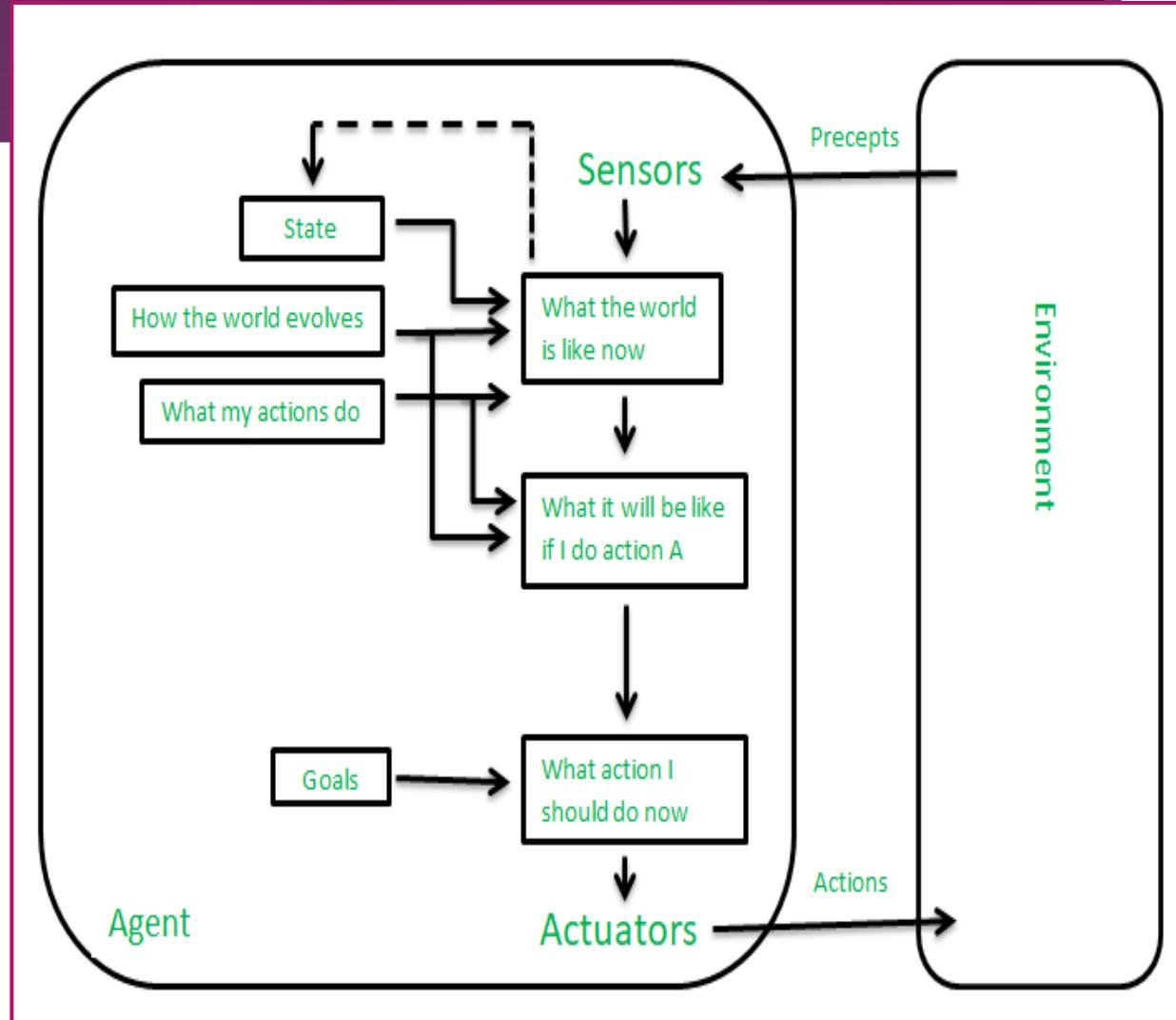
Updating the state requires information about :

- how the world evolves independently from the agent, and
- how the agent's actions affect the world.



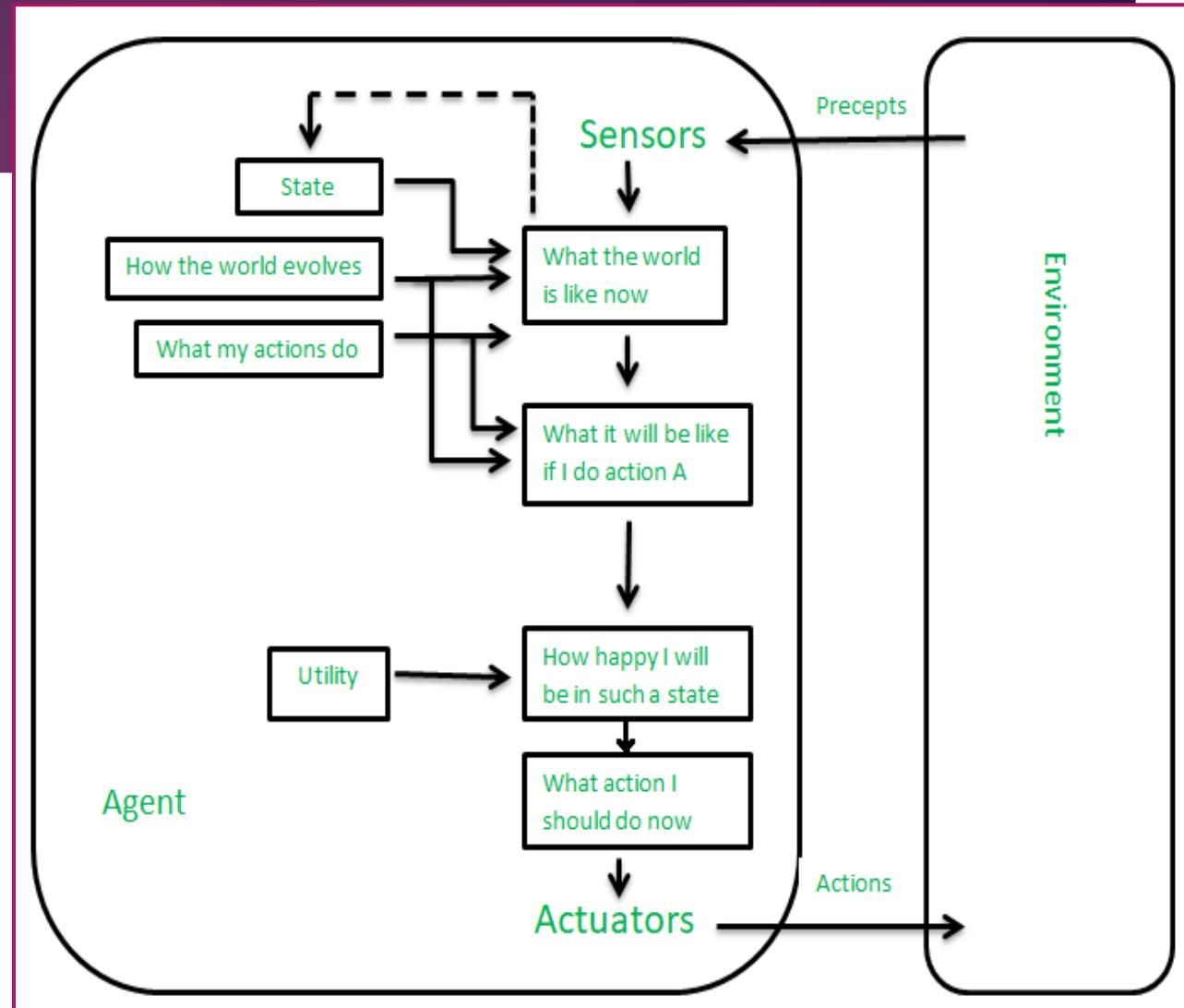
# Goal-based agents

- ▶ These kinds of agents take decisions based on how far they are currently from their **goal**(description of desirable situations).
- ▶ Their every action is intended to reduce its distance from the goal. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state.
- ▶ The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible.
- ▶ They usually require search and planning. The goal-based agent's behavior can easily be changed.



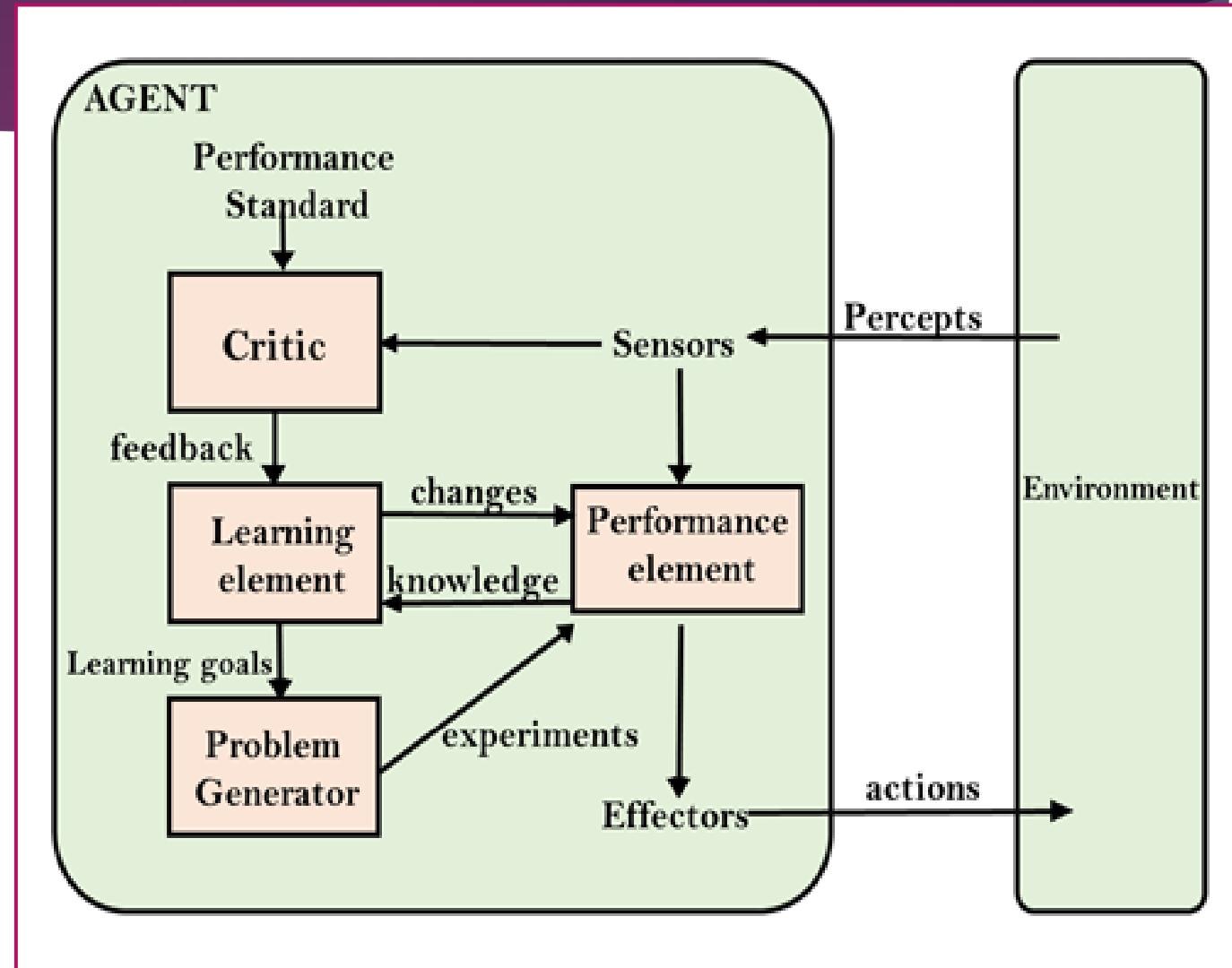
# Utility-based agents

- The agents which are developed having their end uses as building blocks are called utility-based agents.
- When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used.
- They choose actions based on a **preference (utility)** for each state. Sometimes achieving the desired goal is not enough.
- We may look for a quicker, safer, cheaper trip to reach a destination. Agent happiness should be taken into consideration.
- Utility describes how “**happy**” the agent is. Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility.
- A utility function maps a state onto a real number which describes the associated degree of happiness.



# Learning Agent

- ▶ A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities.
- ▶ It starts to act with basic knowledge and then is able to act and adapt automatically through learning.
- ▶ A learning agent has mainly four conceptual components, which are:
- ▶ **Learning element:** It is responsible for making improvements by learning from the environment
- ▶ **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
- ▶ **Performance element:** It is responsible for selecting external action
- ▶ **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



# Applications of AI

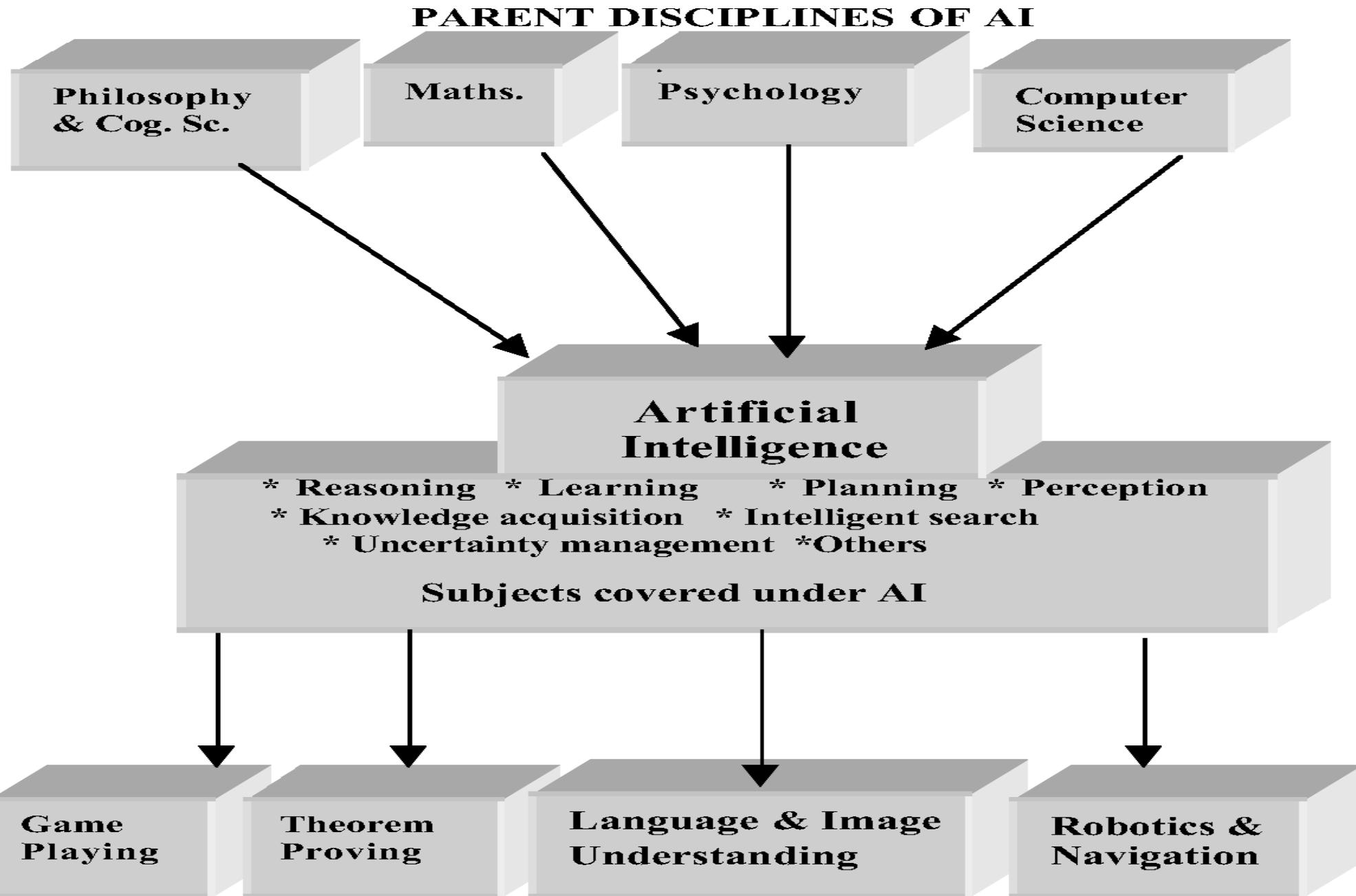
- ▶ **Gaming:** AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
- ▶ **Natural Language Processing:** It is possible to interact with the computer that understands natural language spoken by humans.
- ▶ **Expert Systems:** There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.
- ▶ **Vision Systems:** These systems understand, interpret, and comprehend visual input on the computer. For example,
  - Doctors use clinical expert system to diagnose the patient.
  - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
- ▶ **Speech Recognition:** Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's voice due to cold, etc.
- ▶ **Handwriting Recognition:** The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.
- ▶ **Intelligent Robots:** Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

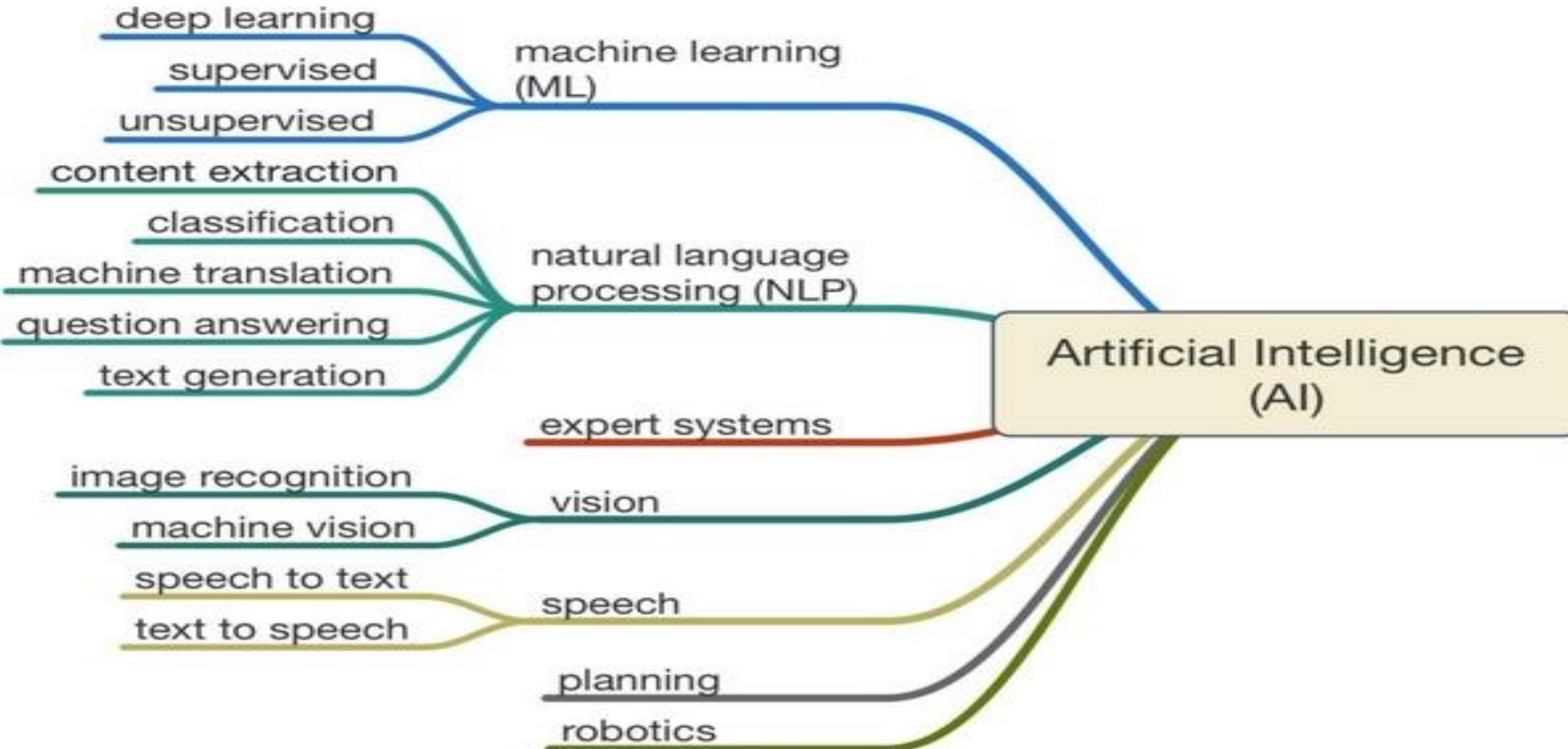
# Types of Intelligence

- ▶ **Linguistic intelligence:** The ability to speak, recognize, and use mechanisms of phonology (speech sounds), syntax (grammar), and semantics (meaning). Ex: Narrators, Orators
- ▶ **Musical intelligence:** The ability to create, communicate with, and understand meanings made of sound, understanding of pitch, rhythm. Ex: Musicians, Singers, Composers
- ▶ **Logical-mathematical intelligence:** The ability of use and understand relationships in the absence of action or objects. Understanding complex and abstract ideas. Ex: Mathematicians, Scientists
- ▶ **Spatial intelligence:** The ability to perceive visual or spatial information, change it, and re-create visual images without reference to the objects, construct 3D images, and to move and rotate them. Ex: Map readers, Astronauts, Physicists
- ▶ **Bodily-Kinesthetic intelligence:** The ability to use complete or part of the body to solve problems or fashion products, control over fine and coarse motor skills, and manipulate the objects. Ex: Players, Dancers
- ▶ **Intra-personal intelligence:** The ability to distinguish among one's own feelings, intentions, and motivations.
- ▶ **Interpersonal intelligence:** The ability to recognize and make distinctions among other people's feelings, beliefs, and intentions. Ex: Mass Communicators, Interviewers

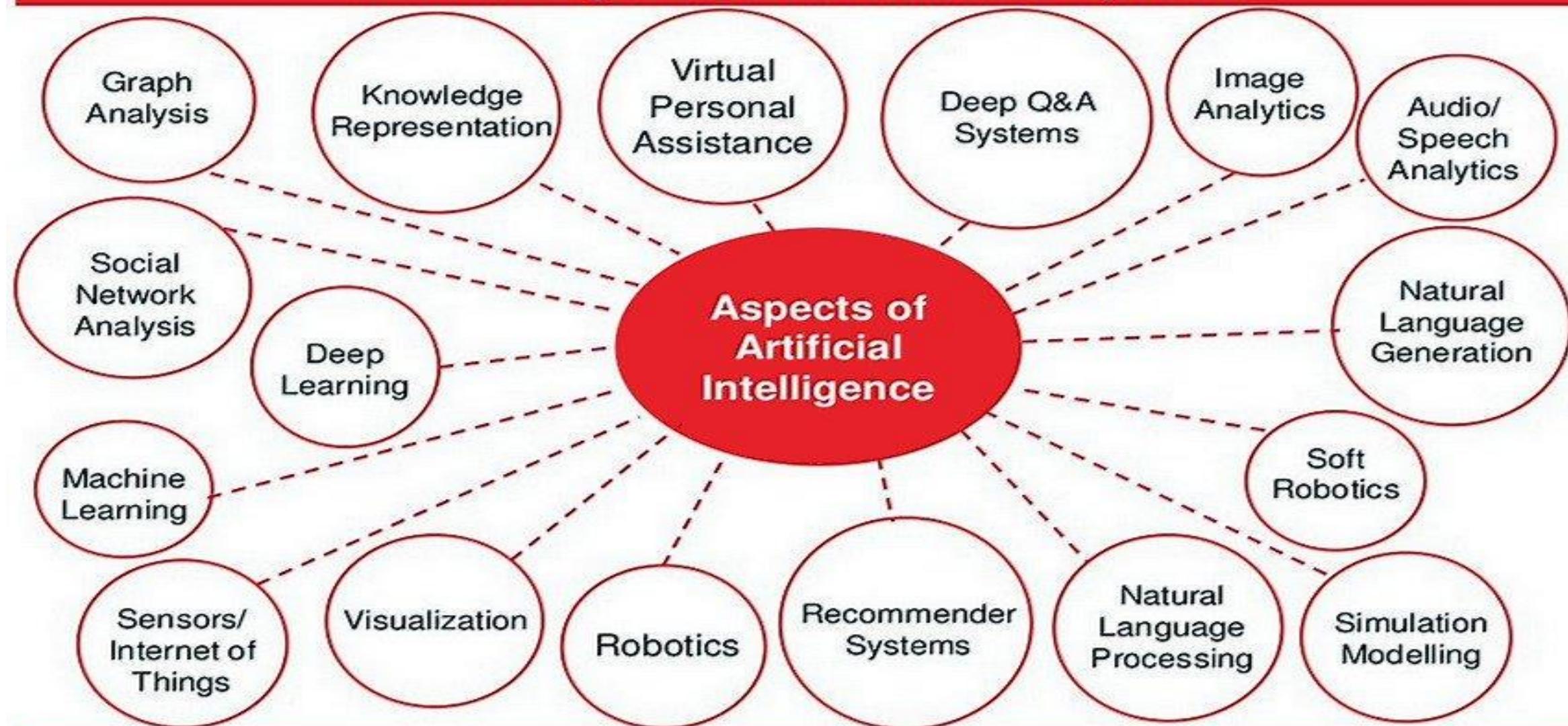
# What's involved in Intelligence?

- ▶ Ability to interact with the real world:
  - To perceive, understand, and act
    - e.g., speech recognition and understanding and synthesis, image understanding, ability to take actions, have an effect
- ▶ Reasoning and Planning:
  - Modelling the external world, given input solving new problems, planning, and making decisions, ability to deal with unexpected problems, uncertainties
- ▶ Learning and Adaptation:
  - we are continuously learning and adapting our internal models are always being “updated”
    - e.g., a baby learning to categorize and recognize animals





## Artificial Intelligence is a multi-dimensional subject area



# What Contributors to AI?

## Programming Without AI

- ▶ A computer program without AI can answer the specific questions it is meant to solve.
- ▶ Modification in the program leads to change in its structure.
- ▶ Modification is not quick and easy. It may lead to affecting the program adversely.

## Programming With AI

- ▶ A computer program with AI can answer the generic questions it is meant to solve.
- ▶ AI programs can absorb new modifications by putting highly independent pieces of information together. Hence you can modify even a minute piece of information of program without affecting its structure.
- ▶ Quick and Easy program modification.

# Example: Water Jug Problem

- ▶ Consider the following problem: A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon 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 gallons of water in the 4-gallon jug?

# Solution: Phase-I

- ▶ 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-gallon jug. Note  $0 \leq x \leq 4$ , and  $0 \leq y \leq 3$ . Our initial state:  $(0,0)$
- ▶ Goal Predicate state =  $(2,y)$  where  $0 \leq y \leq 3$ .

# Solution: Phase-II:--Operators

1. Fill 4-gal jug:  $(x,y) \rightarrow (4,y)$
2. Fill 3-gal jug:  $(x,y) \rightarrow (x,3)$
3. Empty 4-gal jug on ground:  $(x,y) \rightarrow (0,y)$
4. Empty 3-gal jug on ground:  $(x,y) \rightarrow (x,0)$
5. Pour water from 3-gal jug to fill 4-gal jug :  $(x,y) \rightarrow (4, y - (4 - x))$

6. Pour water from 4-gal jug to fill 3-gal jug :  $(x,y) \rightarrow (x-(3-y), 3)$
7. Pour all of water from 3-gal jug into 4-gal jug:  $(x,y) \rightarrow (x+y, 0)$
8. Pour all of water from 4-gal jug into 3-gal jug:  $(x,y) \rightarrow (0, x+y)$

# Solution: Phase-III:--Rules

Gals in 4-gal jug	Gals in 3-gal jug	Rule Applied
0	0	
4	0	1. Fill 4
1	3	6. Pour 4 into 3 to fill
1	0	4. Empty 3
0	1	8. Pour all of 4 into 3
4	1	1. Fill 4
2	3	6. Pour into 3

# Why Searching algorithms in AI?

- ▶ In 1997 ‘Deep Blue’ an AI beat the legendary Gary Kasparov in Chess , and in 2016 ‘Alpha Go’ defeated the champion of the game Go. The ability of artificial intelligence to mimic humans and surpass their mental capabilities has exceeded over time.
- ▶ Searching algorithms form the base of such programs , they are utilized in areas like course and cost optimization, action planning, information mining, mechanical technology, independent driving, computational science, programming and equipment check, hypothesis demonstrating and so on.
- ▶ As it were, a considerable lot of the AI issues can be considered such that the objective is to reach the final goal from an initial point by means of state change rules. So the search space or options are characterized as a diagram (or a tree) and the point is to arrive at the objective from the underlying state through the shortest path.
- ▶ The searching algorithms can be classified into two types:
  - Uninformed methods : in this method no additional information is provided
  - Informed methods : also known as Heuristic method where search is carried out by using additional information to find out the next step to take.

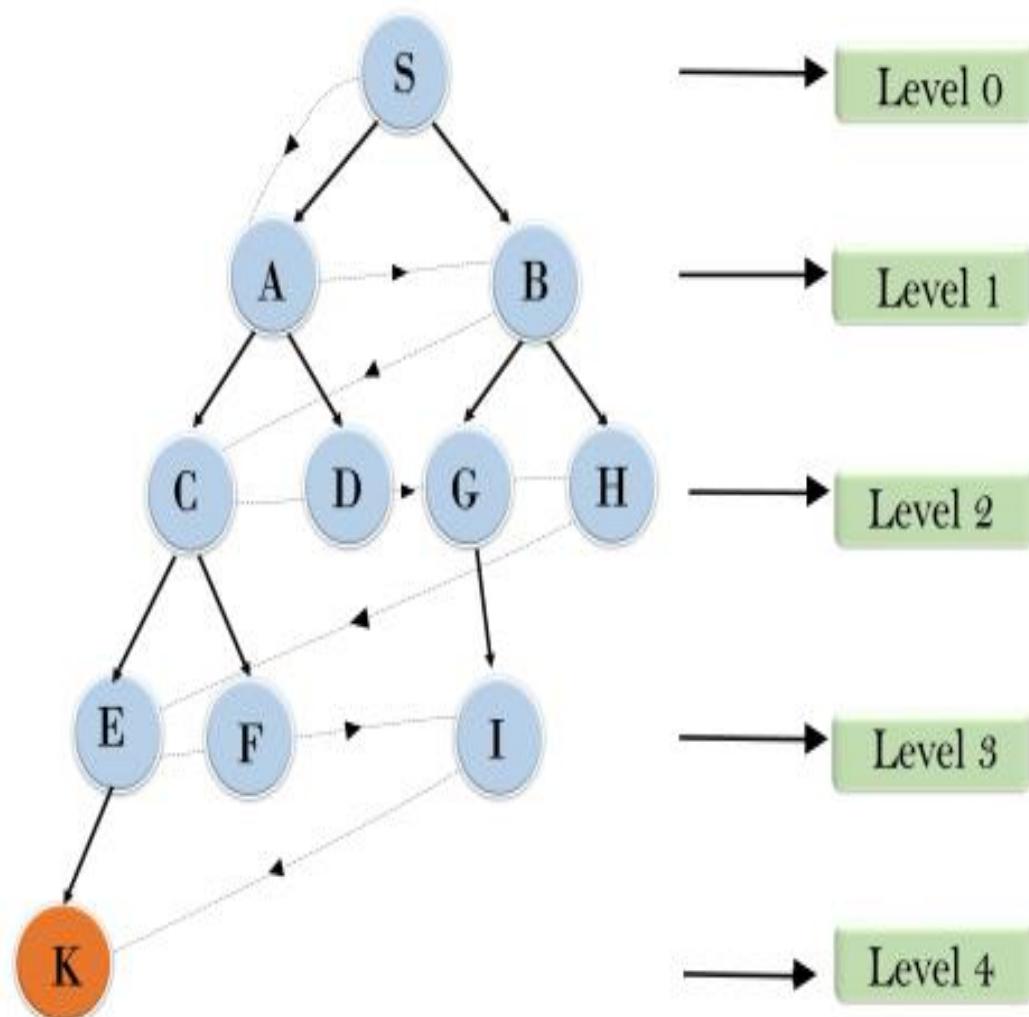
# Breadth-first Search:

- ▶ Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadth wise 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 solutions 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.

# 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
- 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.
- $T(b) = 1+b^2+b^3+\dots+b^d = O(b^d)$
- 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.

## Breadth First Search



# Depth-first Search

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

## **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).

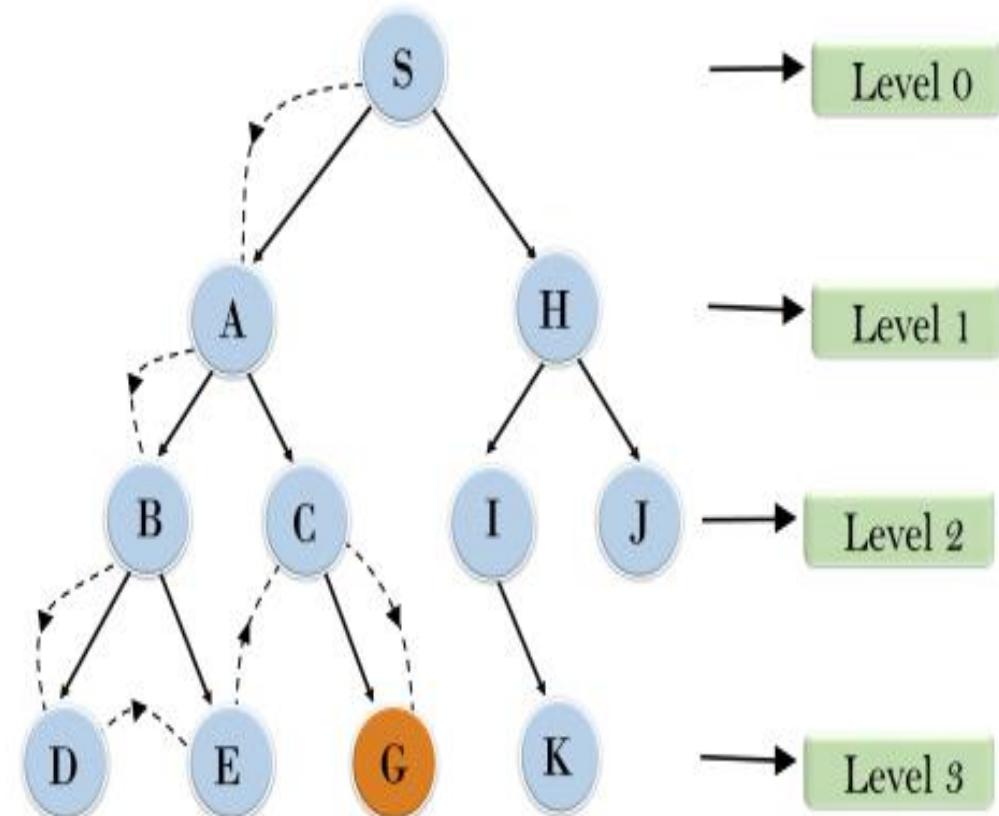
## **Disadvantage:**

- ▶ 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.

# Example

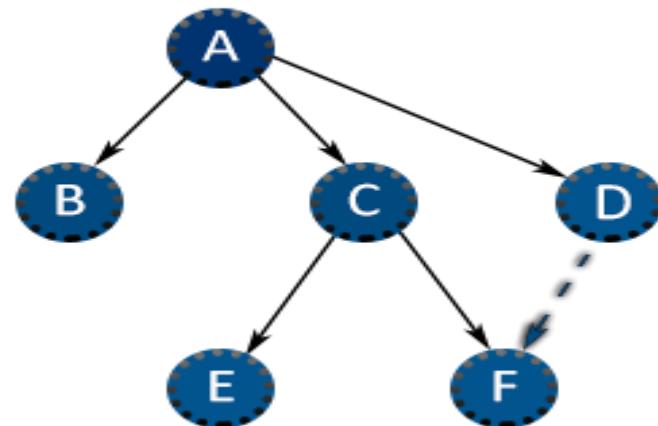
- In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:
- Root node--->Left node ----> right node.
- It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor 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.
- 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:  
 $T(n) = 1 + n^2 + n^3 + \dots + n^m = O(n^m)$
- 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.

## Depth First Search



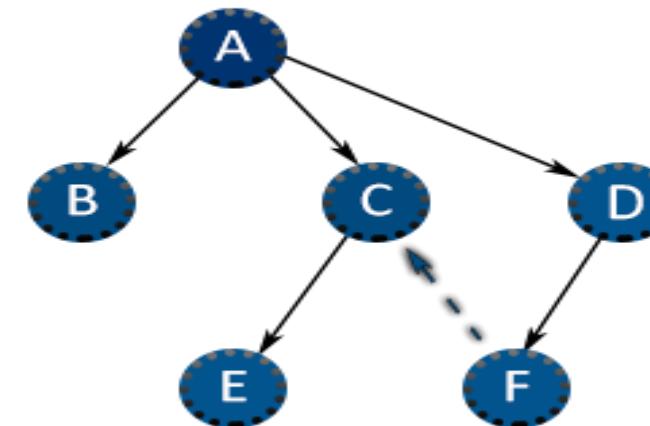
# BFS and DFS

**BFS**



**A B C D E F**

**DFS**



**A D F C E B**

# Depth-Limited Search Algorithm

- ▶ 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.

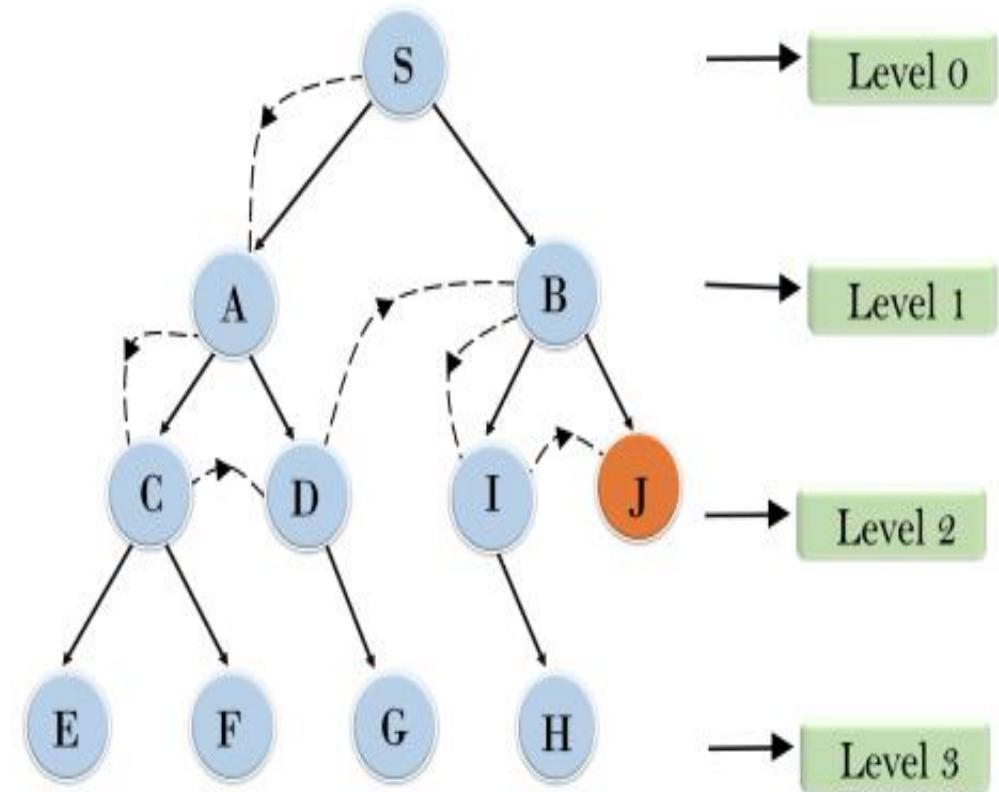
## **Disadvantages:**

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

# Example:

- ▶ **Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.
- ▶ **Time Complexity:** Time complexity of DLS algorithm is  $O(b^{\ell})$ .
- ▶ **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  $\ell > d$ .

## Depth Limited Search



# Uniform-cost Search Algorithm

- ▶ Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs from the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

## **Advantages:**

- ▶ Uniform cost search is optimal because at every state the path with the least cost is chosen.

## **Disadvantages:**

- ▶ It does not care about the number of steps involved in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

# Example

## Completeness:

- ▶ Uniform-cost search is complete, such as if there is a solution, UCS will find it.

## Time Complexity:

- ▶ Let **C\*** is Cost of the optimal solution, and  $\epsilon$  is each step to get closer to the goal node. Then the number of steps is  $= C^*/\epsilon + 1$ . Here we have taken +1, as we start from state 0 and end to  $C^*/\epsilon$ .
- ▶ Hence, the worst-case time complexity of Uniform-cost search is  $O(b^{1 + [C^*/\epsilon]})$ .

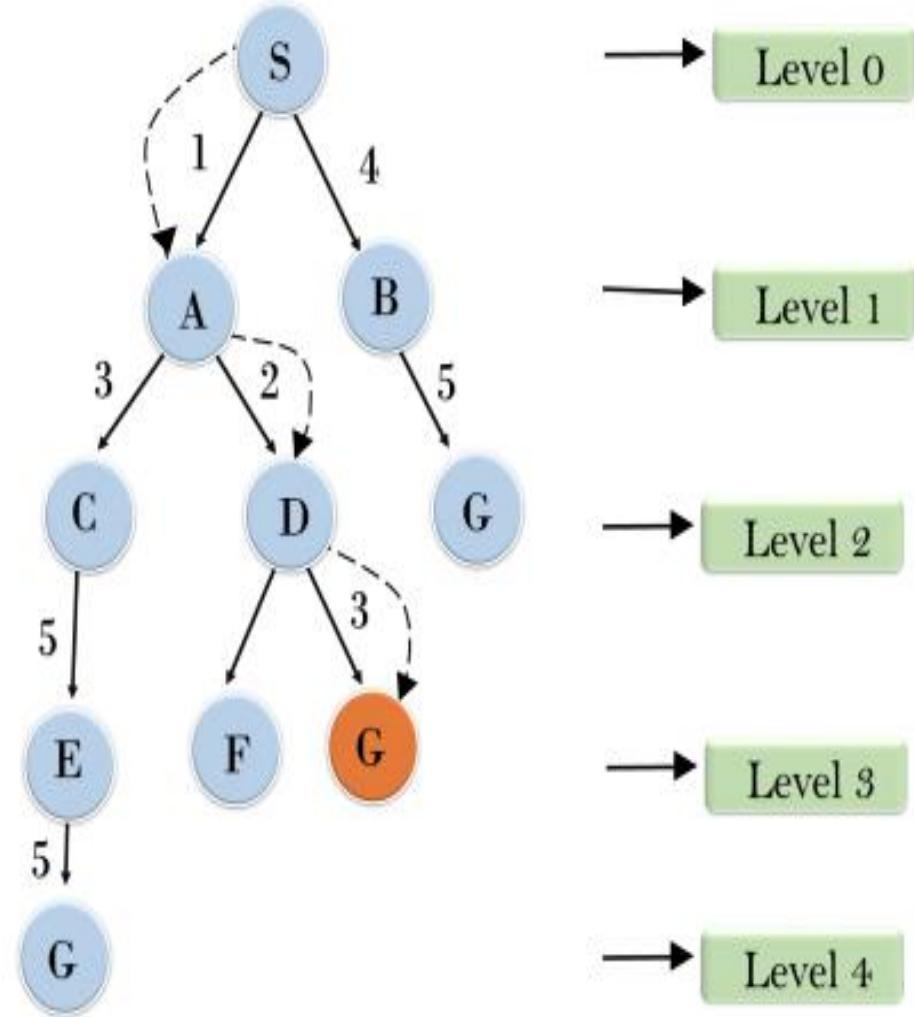
## Space Complexity:

- ▶ The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is  $O(b^{1 + [C^*/\epsilon]})$ .

## Optimal:

- ▶ Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

## Uniform Cost Search



# Iterative deepening depth-first Search

- ▶ 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.

## **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:

- ▶ 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 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<sup>d</sup>)**.

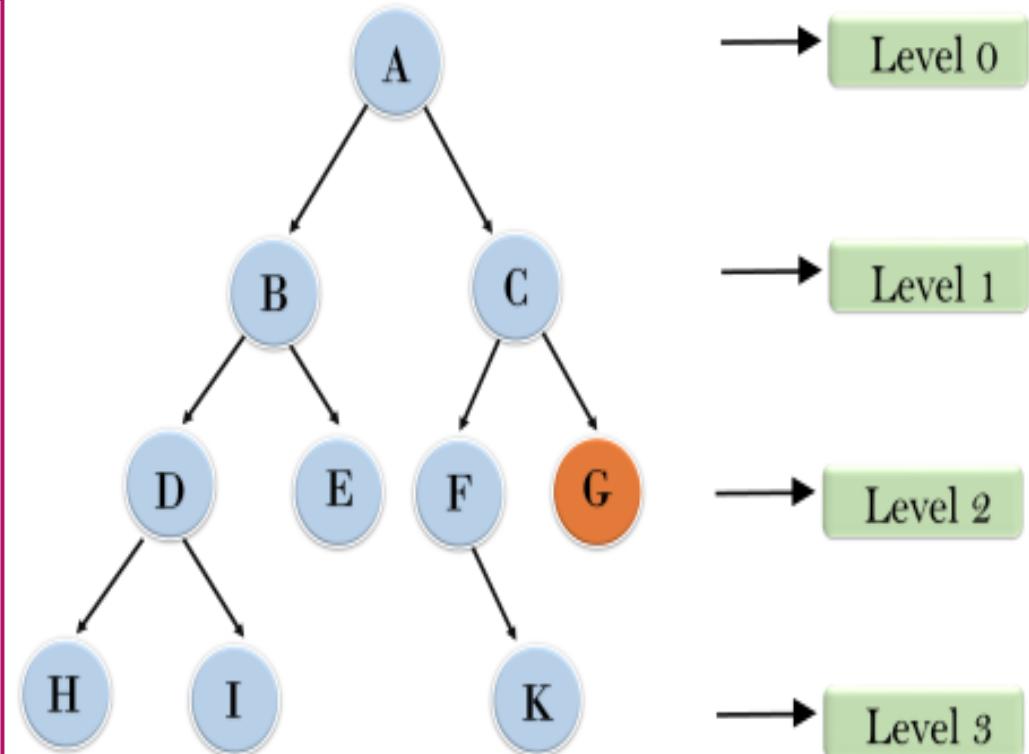
## Space Complexity:

- ▶ 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.

## Iterative deepening depth first search



# Bidirectional Search Algorithm

- ▶ Bidirectional search algorithm runs two simultaneous searches, one from initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small sub graphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.
- ▶ Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

## **Advantages:**

- ▶ Bidirectional search is fast.
- ▶ Bidirectional search requires less memory

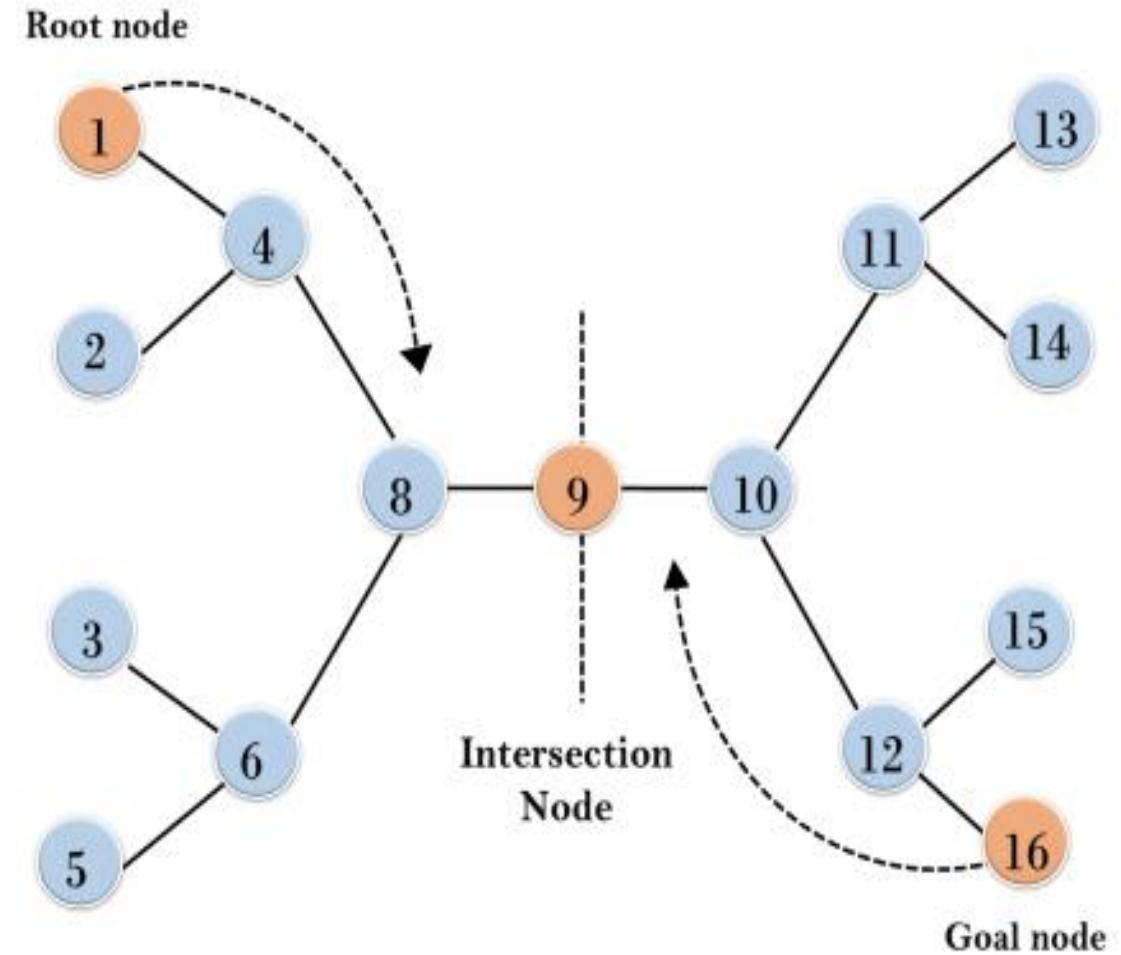
## **Disadvantages:**

- ▶ Implementation of the bidirectional search tree is difficult.
- ▶ In bidirectional search, one should know the goal state in advance.

# Example

- In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
- The algorithm terminates at node 9 where two searches meet.
- Completeness:** Bidirectional Search is complete if we use BFS in both searches.
- Time Complexity:** Time complexity of bidirectional search using BFS is  $O(b^d)$ .
- Space Complexity:** Space complexity of bidirectional search is  $O(b^d)$ .
- Optimal:** Bidirectional search is Optimal.

Bidirectional Search



# Informed search methods

# Definition

► known as Heuristic method where search is carried out by using additional information to find out the next step to take.

# A\* Search Algorithm

- ▶ A\* search is the most commonly known form of best-first search. It uses heuristic function  $h(n)$ , and cost to reach the node  $n$  from the start state  $g(n)$ . It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A\* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A\* algorithm is similar to UCS except that it uses  $g(n)+h(n)$  instead of  $g(n)$ .
- ▶ In A\* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.

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

Estimated cost  
of the cheapest  
solution.

Cost to reach  
node  $n$  from  
start state.

Cost to reach  
from node  $n$  to  
goal node

# Algorithm of A\* search

**Step 1:** 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:**

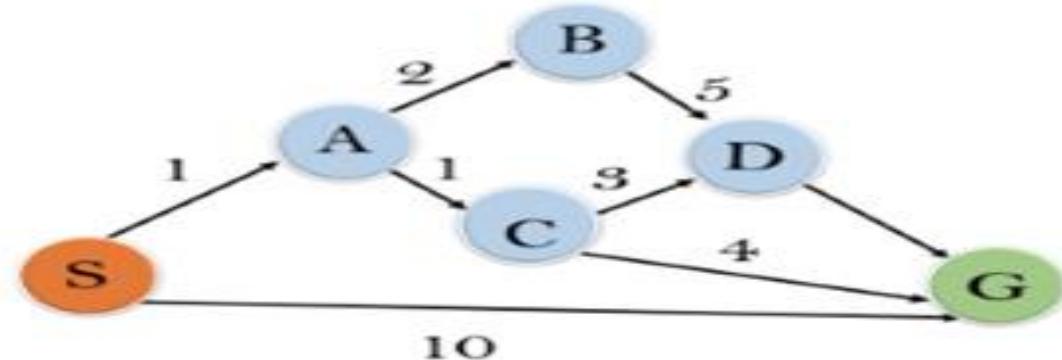
- A\* search algorithm is the best algorithm than other search algorithms.
- A\* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

► **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:

- ▶ In this example, we will traverse the given graph using the A\* algorithm. The heuristic value of all states is given in the below table so we will calculate the  $f(n)$  of each state using the formula  $f(n) = g(n) + h(n)$ , where  $g(n)$  is the cost to reach any node from start state.
- ▶ Here we will use OPEN and CLOSED list.



State	$h(n)$
S	5
A	3
B	4
C	2
D	6
G	0

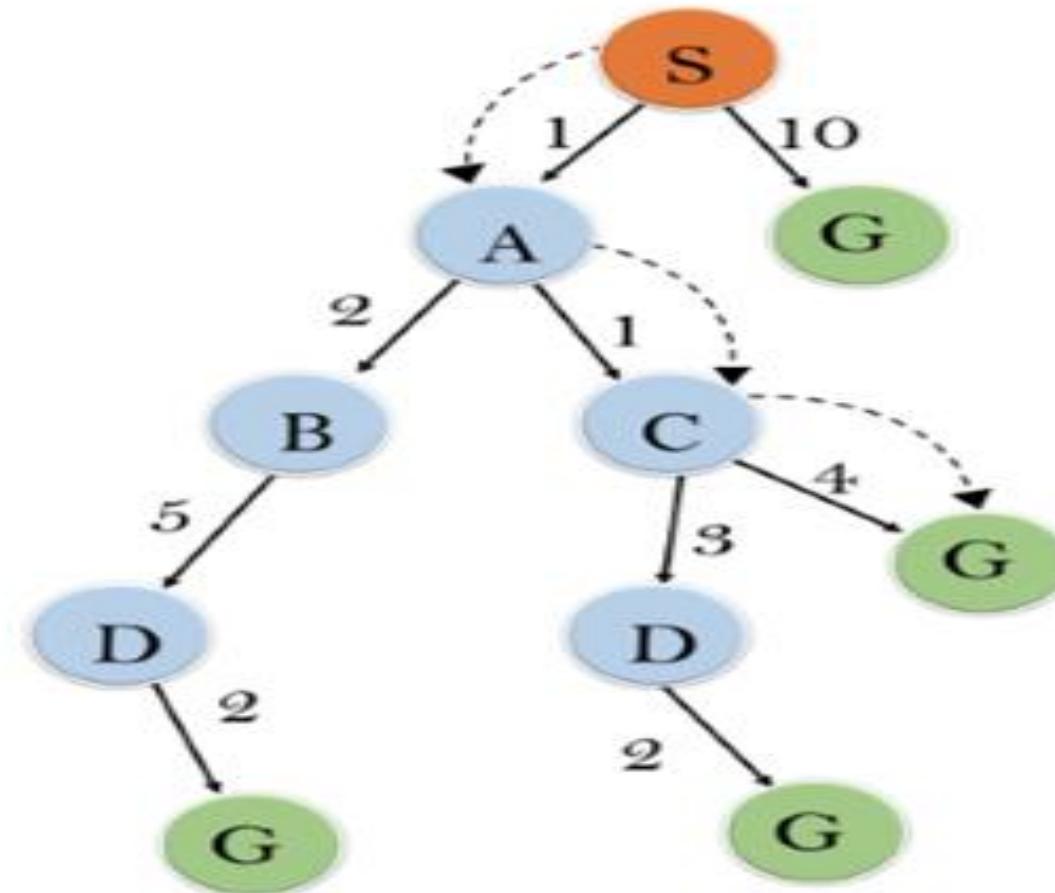
**Initialization:**  $\{(S, 5)\}$

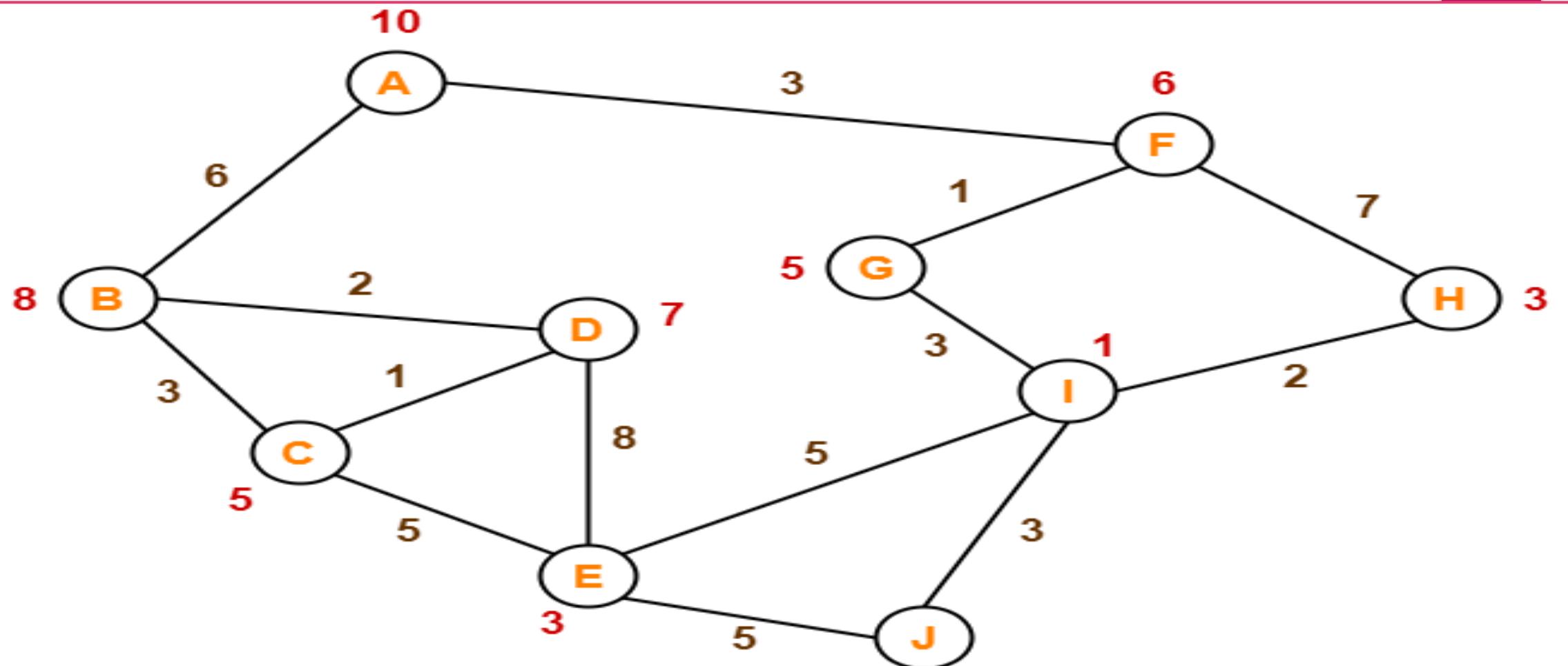
**Iteration1:**  $\{(S \rightarrow A, 4), (S \rightarrow G, 10)\}$

**Iteration2:**  $\{(S \rightarrow A \rightarrow C, 4), (S \rightarrow A \rightarrow B, 7), (S \rightarrow G, 10)\}$

**Iteration3:**  $\{(S \rightarrow A \rightarrow C \rightarrow G, 6), (S \rightarrow A \rightarrow C \rightarrow D, 11), (S \rightarrow A \rightarrow B, 7), (S \rightarrow G, 10)\}$

**Iteration 4** will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

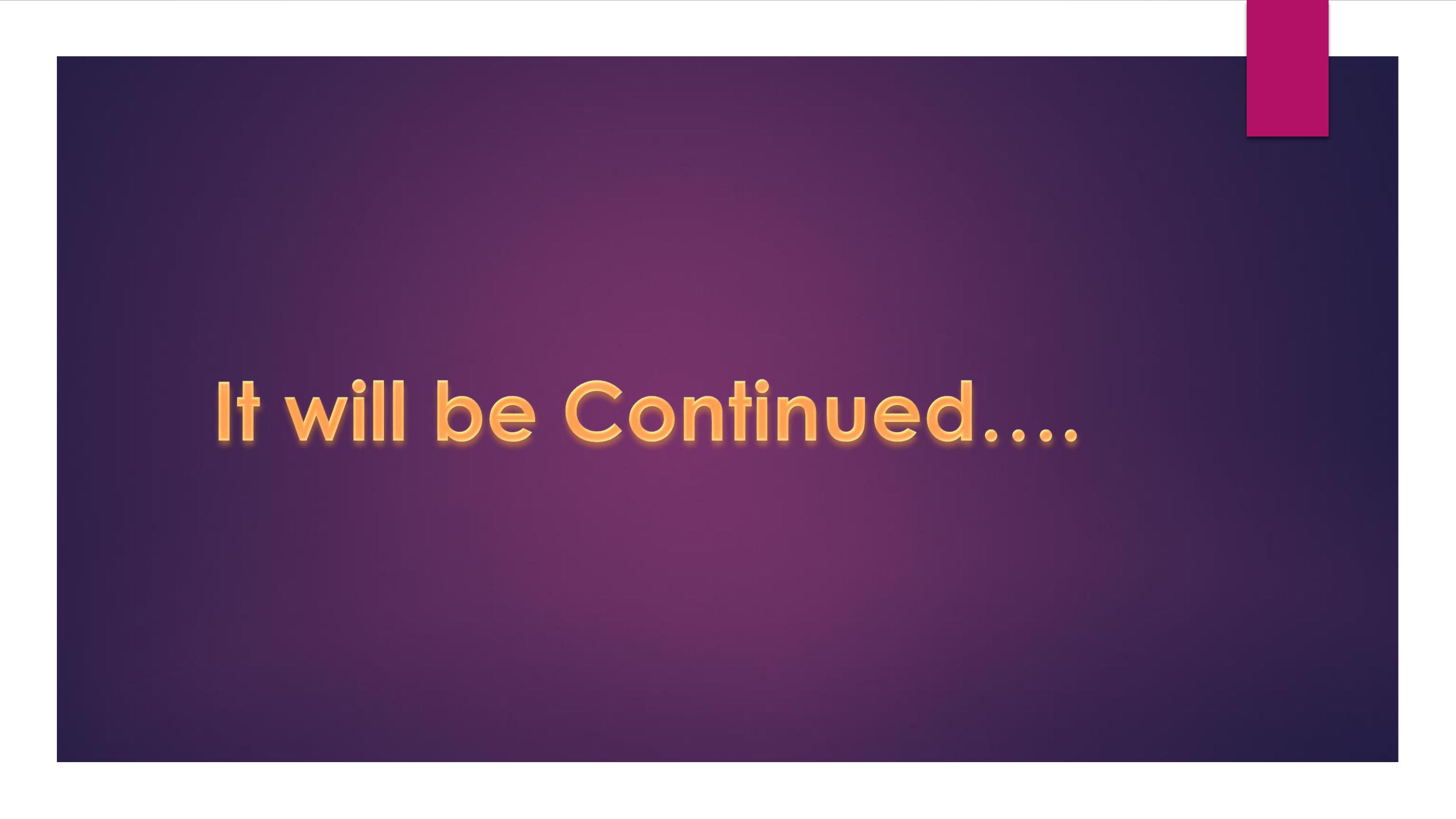




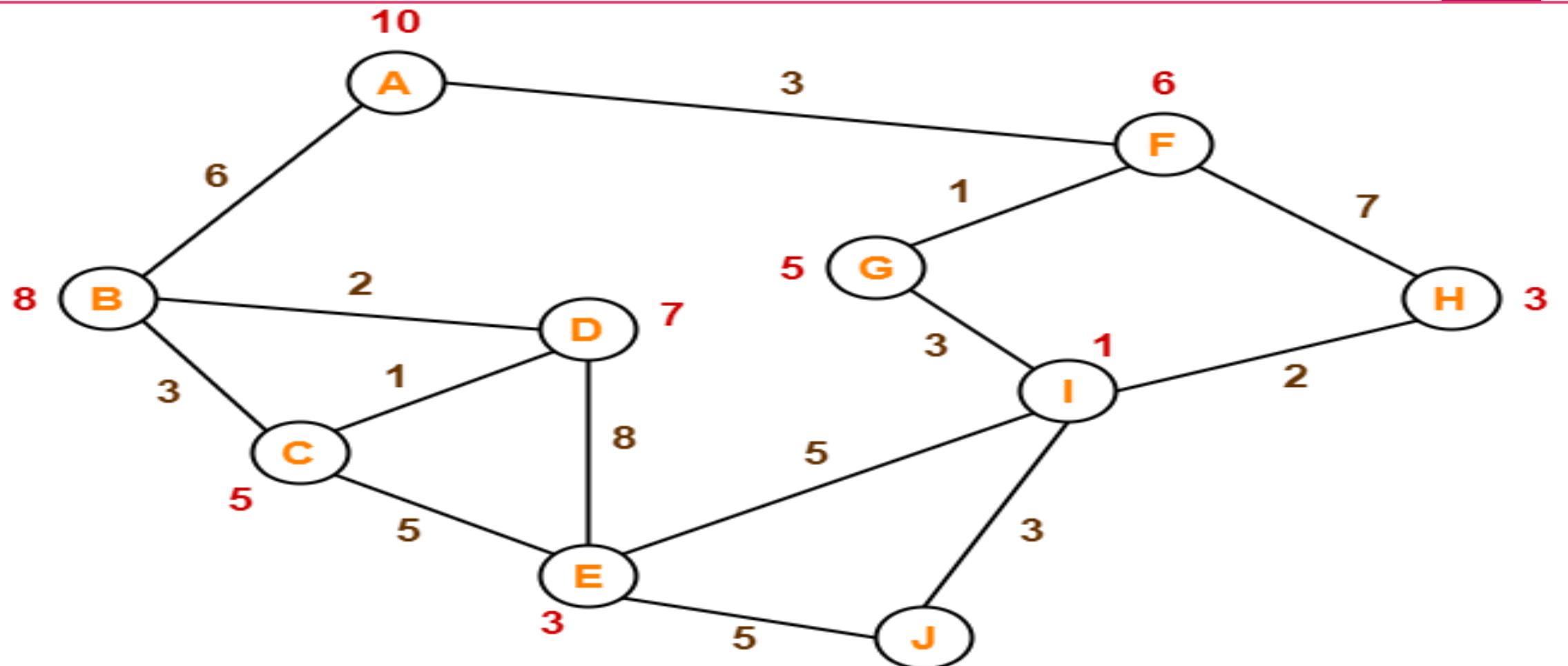
The numbers written on edges represent the distance between the nodes.

The numbers written on nodes represent the heuristic value.

Find the most cost-effective path to reach from start state A to final state J using A\* Algorithm.



**It will be Continued....**



The numbers written on edges represent the distance between the nodes.

The numbers written on nodes represent the heuristic value.

Find the most cost-effective path to reach from start state A to final state J using A\* Algorithm.

### **Step-01:**

We start with node A.

Node B and Node F can be reached from node A.

A\* Algorithm calculates  $f(B)$  and  $f(F)$ .

$$f(B) = 6 + 8 = 14$$

$$f(F) = 3 + 6 = 9$$

Since  $f(F) < f(B)$ , so it decides to go to node F.

Path-  $A \rightarrow F$

### **Step-02:**

Node G and Node H can be reached from node F.

A\* Algorithm calculates  $f(G)$  and  $f(H)$ .

$$f(G) = (3+1) + 5 = 9$$

$$f(H) = (3+7) + 3 = 13$$

Since  $f(G) < f(H)$ , so it decides to go to node G.

Path-  $A \rightarrow F \rightarrow G$

**Step-03:**

Node I can be reached from node G.

A\* Algorithm calculates  $f(I)$ .

$$f(I) = (3+1+3) + 1 = 8$$

It decides to go to node I.

Path- A → F → G → I

**Step-04:**

Node E, Node H and Node J can be reached from node I.

A\* Algorithm calculates  $f(E)$ ,  $f(H)$  and  $f(J)$ .

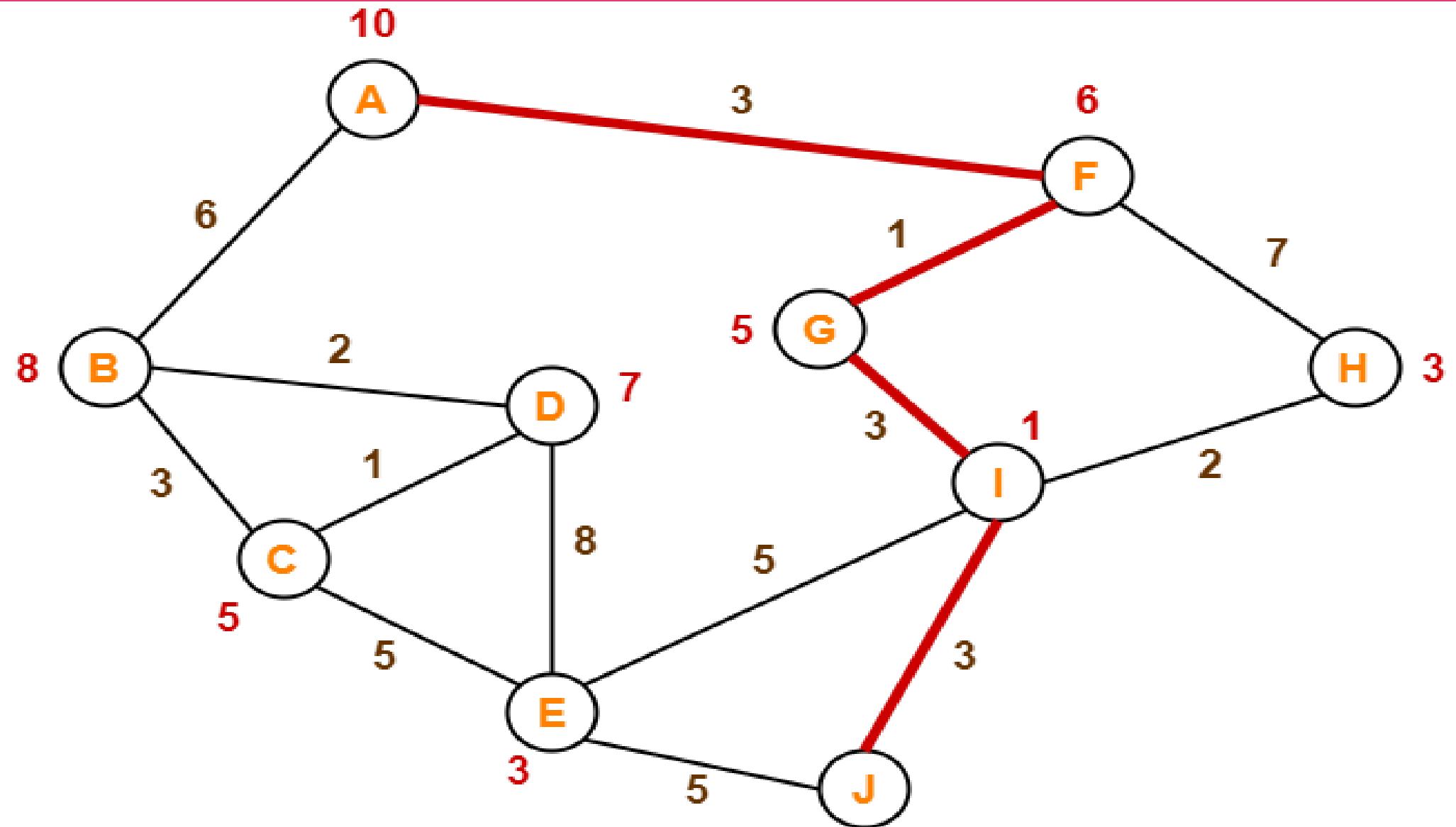
$$f(E) = (3+1+3+5) + 3 = 15$$

$$f(H) = (3+1+3+2) + 3 = 12$$

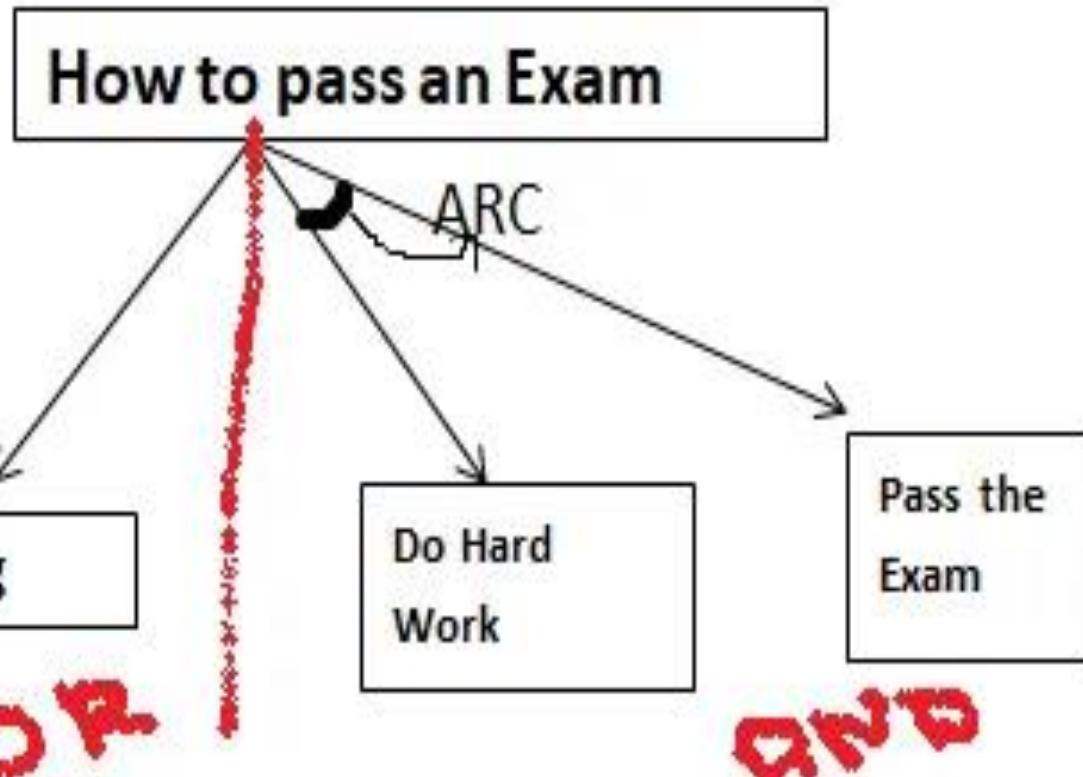
$$f(J) = (3+1+3+3) + 0 = 10$$

Since  $f(J)$  is least, so it decides to go to node J.

Path- A → F → G → I → J



# AND-OR Graph

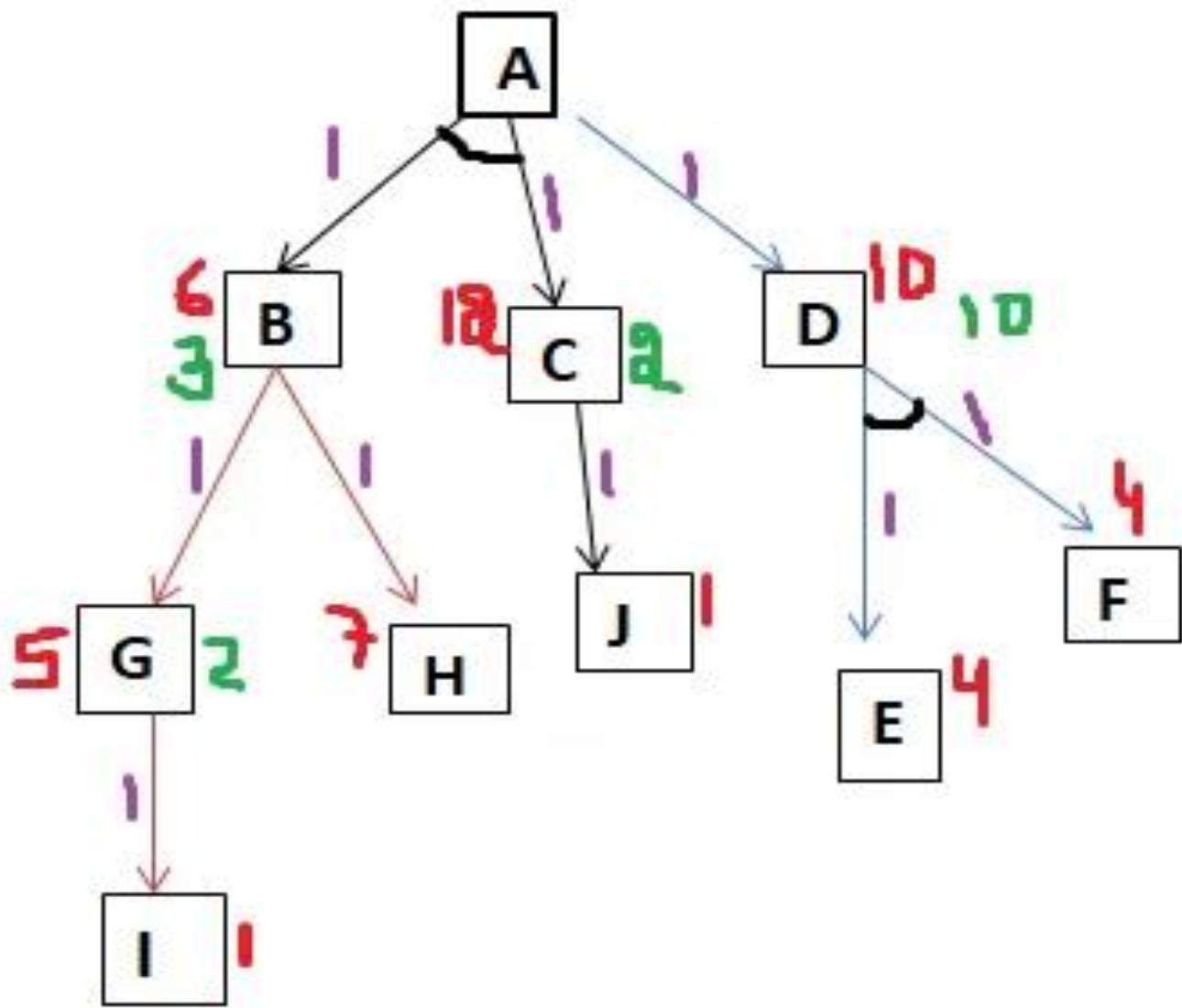


The figure shows an AND-OR graph

1. To pass any exam, we have two options, either cheating or hard work.
2. In this graph we are given two choices, first do cheating **or** (The red line) work hard and **(The arc)** pass.
3. When we have more than one choice and we have to pick one, we apply **OR condition** to choose one.(That's what we did here).
4. Basically the **ARC** here denote **AND condition**.
5. Here we have replicated the arc between the work hard and the pass because by doing the hard work possibility of passing an exam is more than cheating.

# A\* Vs AO\*

- ▶ Both are part of informed search technique and use heuristic values to solve the problem.
- ▶ The solution is guaranteed in both algorithm.
- ▶ A\* always gives an optimal solution (shortest path with low cost) But It is not guaranteed to that AO\* always provide an optimal solutions.
- ▶ Reason: Because AO\* does not explore all the solution path once it got solution.
- ▶ AO\* algorithm requires lesser memory compared to A\* algorithm.
- ▶ AO\* algorithm doesn't go into infinite loop whereas the A\* algorithm can go into an infinite loop.



The algorithm always moves towards a lower cost value.

Basically, We will calculate the cost function here ( $F(n) = G(n) + H(n)$ )

H: heuristic/ estimated value of the nodes. and G: actual cost or edge value (here unit value).

Here we have taken the edges value 1 , meaning we have to focus solely on the heuristic value.

The Purple color values are edge values (here all are same that is one).

The Red color values are Heuristic values for nodes.

The Green color values are New Heuristic values for nodes.

# Procedure:

- ▶ In the above diagram we have two ways from A to D or A to B-C (because of and condition). calculate cost to select a path
- ▶  $F(A-D) = 1 + 10 = 11$       and       $F(A-BC) = 1 + 1 + 6 + 12 = 20$
- ▶ As we can see  $F(A-D)$  is less than  $F(A-BC)$  then the algorithm choose the path  $F(A-D)$ .
- ▶ From D we have one choice that is F-E.
- ▶  $F(A-D-FE) = 1 + 1 + 4 + 4 = 10$
- ▶ Basically 10 is the cost of reaching FE from D. And Heuristic value of node D also denote the cost of reaching FE from D. So, the new Heuristic value of D is 10.
- ▶ And the Cost from A-D remain same that is 11.

## Let's Explore the other path:

1.In the above diagram we have two ways from **A to D** or **A to B-C** (because of and condition). calculate cost to select a path

**2.** $F(A-D) = 1+10 = 11$       and       $F(A-BC) = 1 + 1 + 6 + 12 = 20$

3.As we know the cost is more of **F(A-BC)** but let's take a look

4.Now from B we have two path G and H , let's calculate the cost

**5.** $F(B-G) = 5+1 = 6$    and  $F(B-H) = 7 + 1 = 8$

6.So, cost from **F(B-H)** is more than **F(B-G)** we will take the path B-G.

7.The Heuristic value from **G to I** is 1 but let's calculate the cost form **G to I**.

**8.** $F(G-I) = 1 + 1 = 2$ . which is less than Heuristic value 5. So, the new Heuristic value form **G to I** is 2.

9.If it is a new value, then the cost from **G to B** must also have changed. Let's see the new cost form (**B to G**)

10. $F(B-G) = 1+2 = 3$  . Mean the New Heuristic value of B is 3.

**11.**But A is associated with both B and C .

12.As we can see from the diagram C only have one choice or one node to explore that is J. The Heuristic value of C is 12.

13.Cost form C to J=  $F(C-J) = 1+1= 2$  Which is less than Heuristic value

**14.**Now the New Heuristic value of C is 2.

**15.**And the New Cost from A- BC that is  $F(A-BC) = 1+1+2+3 = 7$  which is less than  $F(A-D)=11$ .

16.In this case Choosing path A-BC is more cost effective and good than that of A-D.

# Decision Theory

Supervised Learning

# Which Attribute is "best"?

- ▶ We would like to select the attribute that is most useful for classifying examples.
- ▶ • ***Information gain*** measures how well a given attribute separates the training examples according to their target classification.
- ▶ • ID3 uses this *information gain* measure to select among the candidate attributes at each step while growing the tree.
- ▶ • In order to define information gain precisely, we use a measure commonly used in information theory, called ***entropy***
- ▶ • ***Entropy*** characterizes the (im)purity of an arbitrary collection of examples.

# Information Theory –ID3 (Iterative Dichotomiser 3)

- ❖ ID3 algorithm invented by Ross Quinlan and uses information gain as its attribute selection measure
- ❖ This measure is based on pioneering work by Claude Shannon on information theory, which studied the value or “information content” of messages
- ❖ Let node N represent or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node N
- ❖ This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions
- ❖ The expected information needed to classify a tuple in D is given by

$$\text{Info}(D) = - \sum_{i=1}^m p_i \log_2(p_i),$$

- ▶ Let D, the data partition, be a training set of class-labeled tuples. Suppose the class label attribute has m distinct values defining m distinct classes,  $C_i$  (here  $i = 1$  to  $m$ );  $p_i = s_i/s$ ;  $s$  = no. of samples;  $s_i$  = no. of samples in class label  $C_i$ ;  $\text{Info}(D)$  is also known as the **entropy** of  $D$

# ID3--Continued

- ▶ suppose we were to partition the tuples in  $D$  on some attribute  $A$  having  $v$  distinct values,  $[a_1, a_2, \dots, a_v]$ , as observed from the training data. If  $A$  is discrete-valued, these values correspond directly to the  $v$  outcomes of a test on  $A$ . Attribute  $A$  can be used to split  $D$  into  $v$  partitions or subsets,  $[D_1, D_2, \dots, D_v]$ , where  $D_j$  contains those tuples in  $D$  that have outcome  $a_j$  of  $A$

$$\text{Info}_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{Info}(D_j).$$

- ▶ Here,  $|D_j| / |D|$  acts as the weight of the  $j$ th partition;  $\text{Info}_A(D)$  is the expected information required to classify a tuple from  $D$  based on the partitioning by  $A$ .
- ▶  $\text{Info}(D_j) = -\sum_{i=1}^m p_{ij} \log_2(p_{ij})$ ;  $p_{ij} = s_{ij} / |D_j|$ ;  $s_{ij}$  = no. of samples belongs to class label  $C_i$  and having the attribute value  $a_j$

# ID3--Continued

- ▶ Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on  $A$ ).

$$Gain(A) = Info(D) - Info_A(D).$$

- ▶ In other words,  $Gain(A)$  tells us how much would be gained by branching on  $A$ . It is the expected reduction in the information requirement caused by knowing the value of  $A$ . The attribute  $A$  with the highest information gain,  $Gain(A)$ , is chosen as the splitting attribute at node  $N$ .

# Problem statement: Find out Test Attribute

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

## Solution:

- Class P: *buys\_computer* = “yes”
- Class N: *buys\_computer* = “no”

$$\text{Entropy}(D) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

- Compute the expected information requirement for each attribute: start with the attribute *age*

$$\begin{aligned} & \text{Gain}(\text{age}, D) \\ &= \text{Entropy}(D) - \sum_{v \in \{\text{Youth, Middle-aged, Senior}\}} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \\ &= \text{Entropy}(D) - \frac{5}{14} \text{Entropy}(S_{\text{youth}}) - \frac{4}{14} \text{Entropy}(S_{\text{middle-aged}}) - \frac{5}{14} \text{Entropy}(S_{\text{senior}}) \\ &= 0.246 \end{aligned}$$

$$\text{Gain}(\text{income}, D) = 0.029$$

$$\text{Gain}(\text{student}, D) = 0.151$$

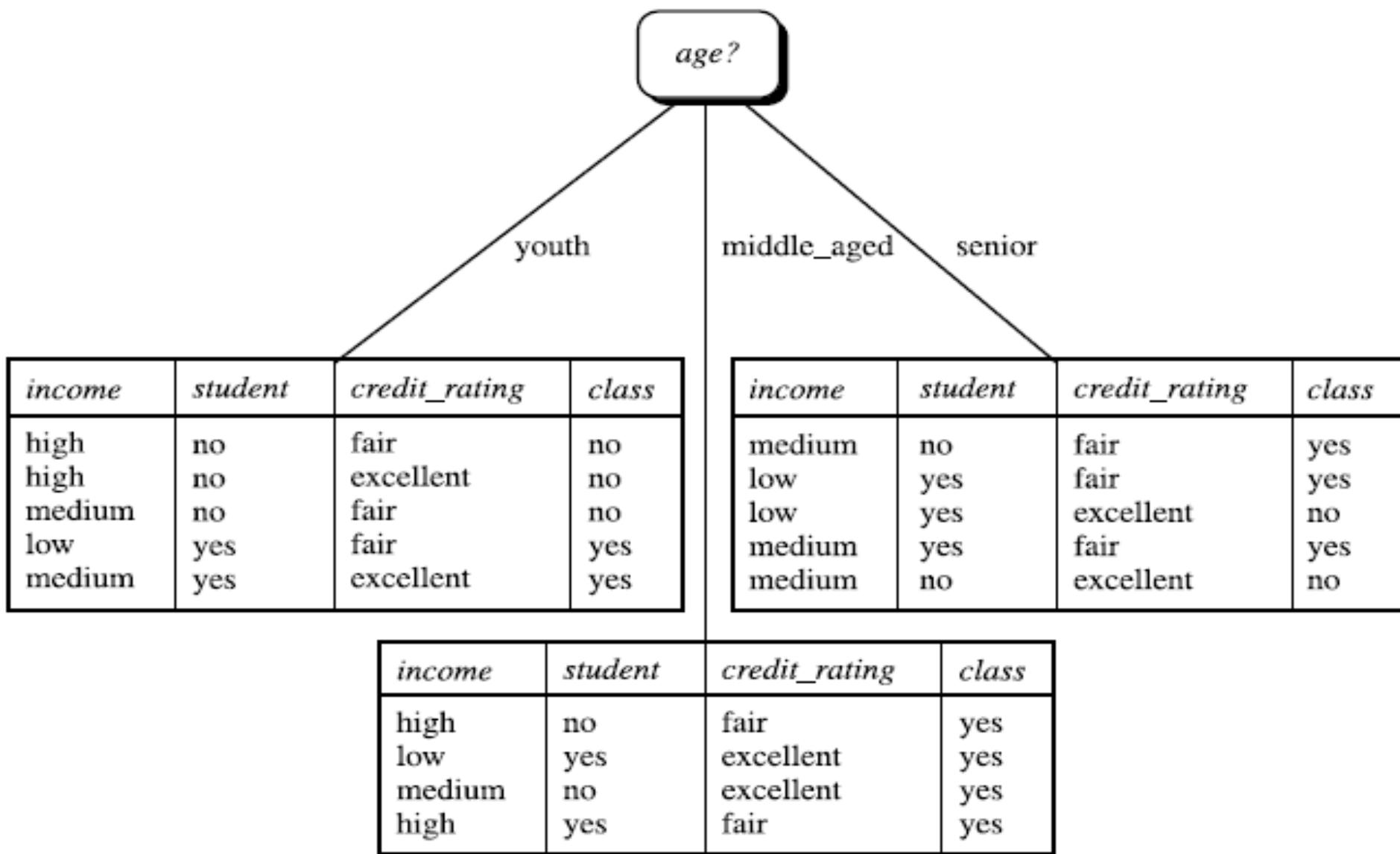
$$\text{Gain}(\text{credit\_rating}, D) = 0.048$$

$$\begin{aligned} \text{Entropy}(S_{\text{youth}}) &= - \sum_{i=1}^2 p_{i1} \log_2(p_{i1}) \\ &= - p_{11} \log_2(p_{11}) - p_{21} \log_2(p_{21}) \\ &= -2/5 \log_2(2/5) - 3/5 \log_2(3/5) \\ &= 0.971 \end{aligned}$$

Here,  $p_{11} = s_{11}/|D_1| = 2/5$   
 $p_{21} = s_{21}/|D_1| = 3/5$   
 $\log_2 X = \log_{10} X / \log_{10} 2$

$$\begin{aligned} \text{Entropy}(S_{\text{middle}}) &= - \sum_{i=1}^2 p_{i2} \log_2(p_{i2}) \\ &= - p_{12} \log_2(p_{12}) - p_{22} \log_2(p_{22}) \\ &= -4/4 \log_2(4/4) - 0/4 \log_2(0/4) \\ &= 0 \end{aligned}$$

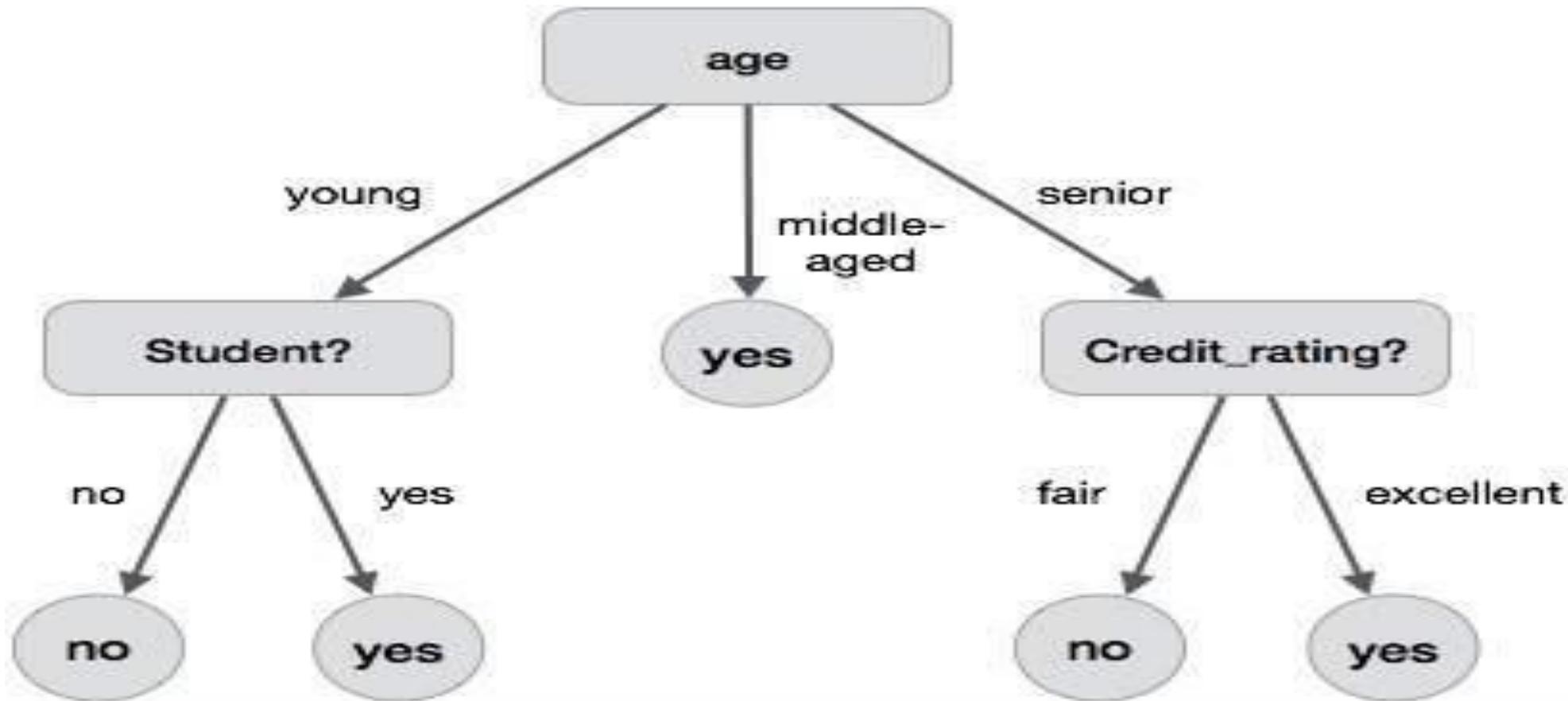
Here,  $p_{12} = s_{12}/|D_2| = 4/4$   
 $p_{22} = s_{22}/|D_2| = 0/4$



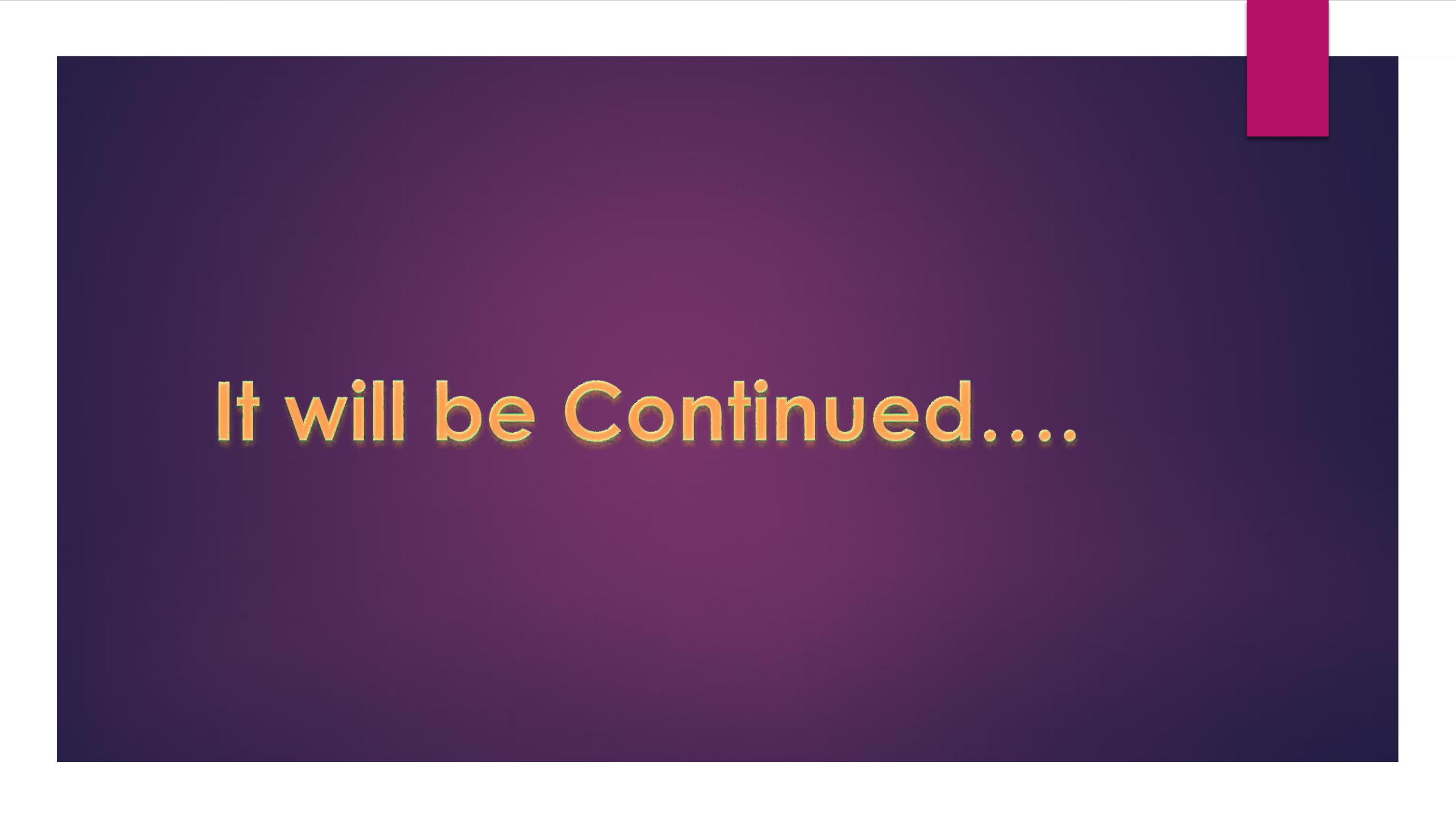
# Extracting Rules from Decision Tree

- R1: IF *age* = *youth* AND *student* = *no* THEN *buys\_computer* = *no*
- R2: IF *age* = *youth* AND *student* = *yes* THEN *buys\_computer* = *yes*
- R3: IF *age* = *middle\_aged* THEN *buys\_computer* = *yes*
- R4: IF *age* = *senior* AND *credit\_rating* = *excellent* THEN *buys\_computer* = *no*
- R5: IF *age* = *senior* AND *credit\_rating* = *fair* THEN *buys\_computer* = *yes*

# Decision Tree



$X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit} = \text{fair})$  Class label=?



**It will be Continued...**

# Association Rule Mining

## CART

# Assignment:

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

# Gini Index

- Many alternative measures to Information Gain
- Most popular alternative: Gini index
  - used in e.g., in CART (Classification And Regression Trees)
  - impurity measure (instead of entropy)

$$Gini(S) = 1 - \sum_i p_i^2$$

- average Gini index (instead of average entropy / information)

$$\text{Gini}(S, A) = \sum_j \frac{\text{card}(Sj)}{\text{card}(S)} Gini(Sj)$$

- Gini Gain
  - could be defined analogously to information gain
  - but typically avg. Gini index is minimized instead of maximizing Gini gain

# Dataset

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

## Gini index

Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

$$\text{Gini} = 1 - \sum (P_i)^2 \text{ for } i=1 \text{ to number of classes}$$

## Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$$\text{Gini}(\text{Outlook}=\text{Sunny}) = 1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$$

$$\text{Gini}(\text{Outlook}=\text{Overcast}) = 1 - (4/4)^2 - (0/4)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Rain}) = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$$

Then, we will calculate weighted sum of gini indexes for outlook feature.

$$\text{Gini}(\text{Outlook}) = (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 = 0.171 + 0 + 0.171 = 0.342$$

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

$$\text{Gini}(\text{Temp}=\text{Hot}) = 1 - (2/4)^2 - (2/4)^2 = 0.5$$

$$\text{Gini}(\text{Temp}=\text{Cool}) = 1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$\text{Gini}(\text{Temp}=\text{Mild}) = 1 - (4/6)^2 - (2/6)^2 = 1 - 0.444 - 0.111 = 0.445$$

We'll calculate weighted sum of gini index for temperature feature

$$\text{Gini}(\text{Temp}) = (4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 = 0.142 + 0.107 + 0.190 = 0.439$$

## Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

$$\text{Gini}(\text{Humidity}=\text{High}) = 1 - (3/7)^2 - (4/7)^2 = 1 - 0.183 - 0.326 = 0.489$$

$$\text{Gini}(\text{Humidity}=\text{Normal}) = 1 - (6/7)^2 - (1/7)^2 = 1 - 0.734 - 0.02 = 0.244$$

Weighted sum for humidity feature will be calculated next

$$\text{Gini}(\text{Humidity}) = (7/14) \times 0.489 + (7/14) \times 0.244 = 0.367$$

# Wind

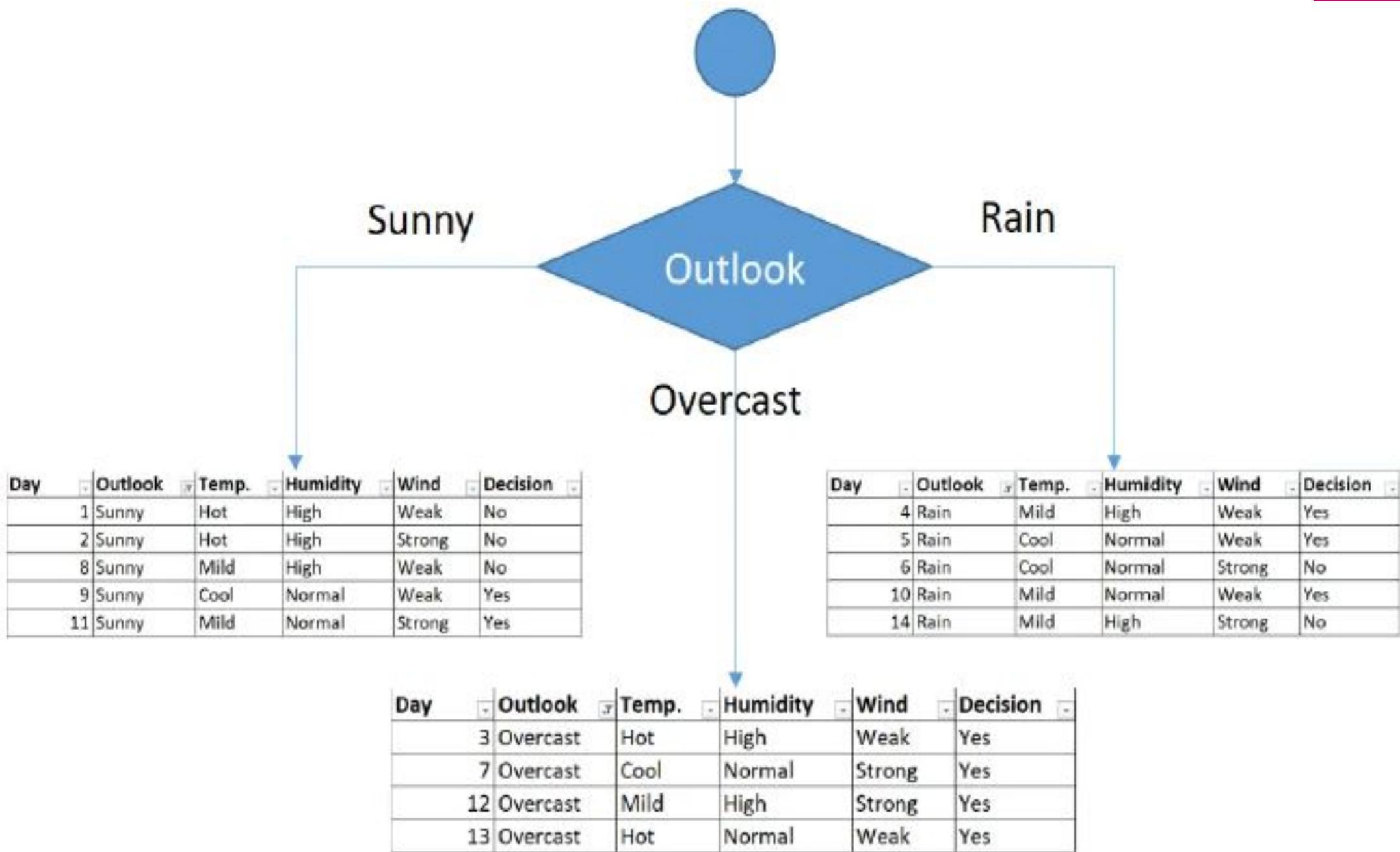
Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

$$\text{Gini}(\text{Wind}=\text{Weak}) = 1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$$

$$\text{Gini}(\text{Wind}=\text{Strong}) = 1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$$

$$\text{Gini}(\text{Wind}) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$$



We will apply same principles to those sub datasets in the following steps.

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

## Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Temp.}=\text{Hot}) = 1 - (0/2)^2 - (2/2)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Temp.}=\text{Cool}) = 1 - (1/1)^2 - (0/1)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Temp.}=\text{Mild}) = 1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Temp.}) = (2/5) \times 0 + (1/5) \times 0 + (2/5) \times 0.5 = 0.2$$

## Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Humidity}=\text{High}) = 1 - (0/3)^2 - (3/3)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Humidity}=\text{Normal}) = 1 - (2/2)^2 - (0/2)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Humidity}) = (3/5) \times 0 + (2/5) \times 0 = 0$$

## Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Wind}=\text{Weak}) = 1 - (1/3)^2 - (2/3)^2 = 0.266$$

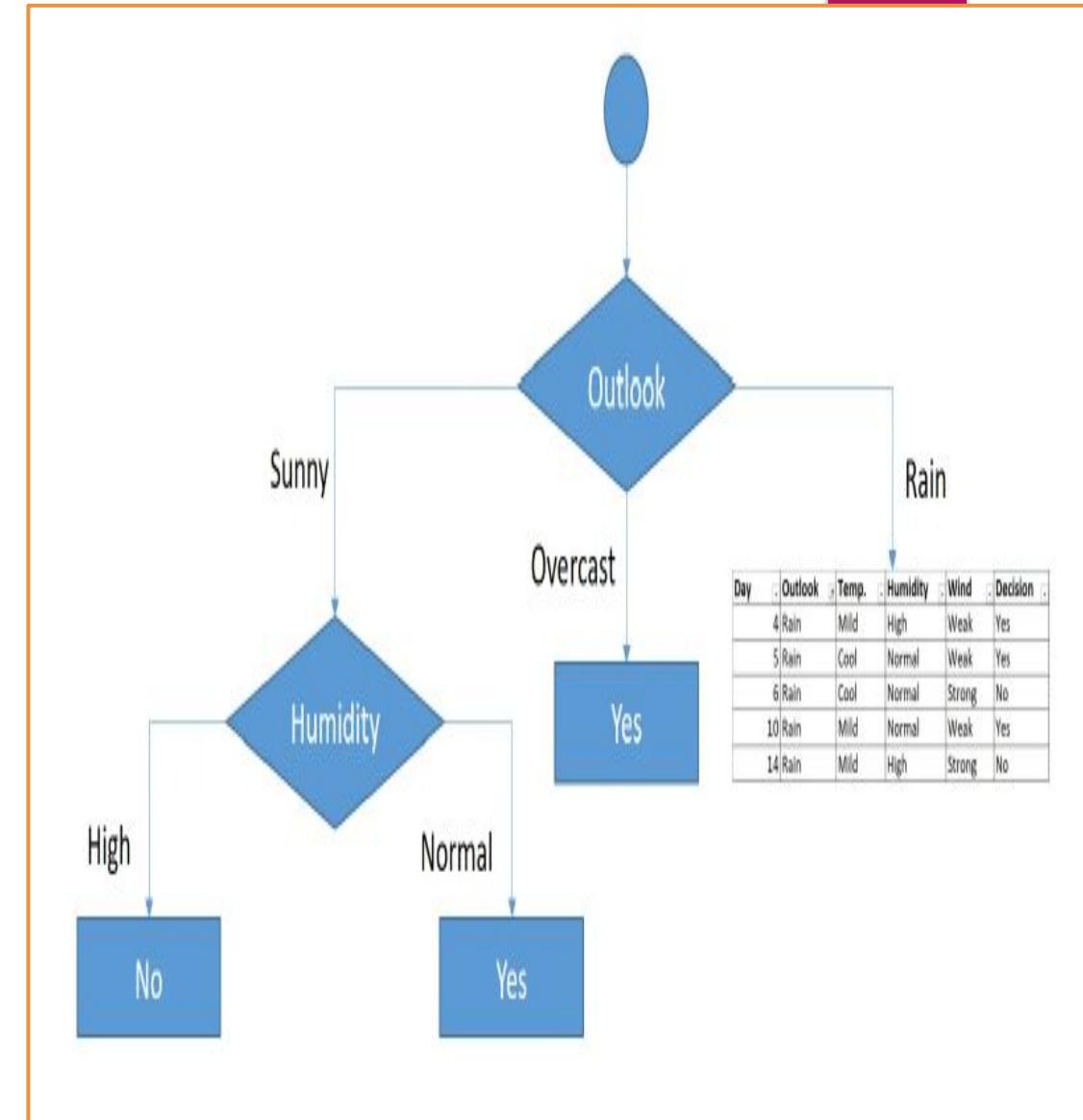
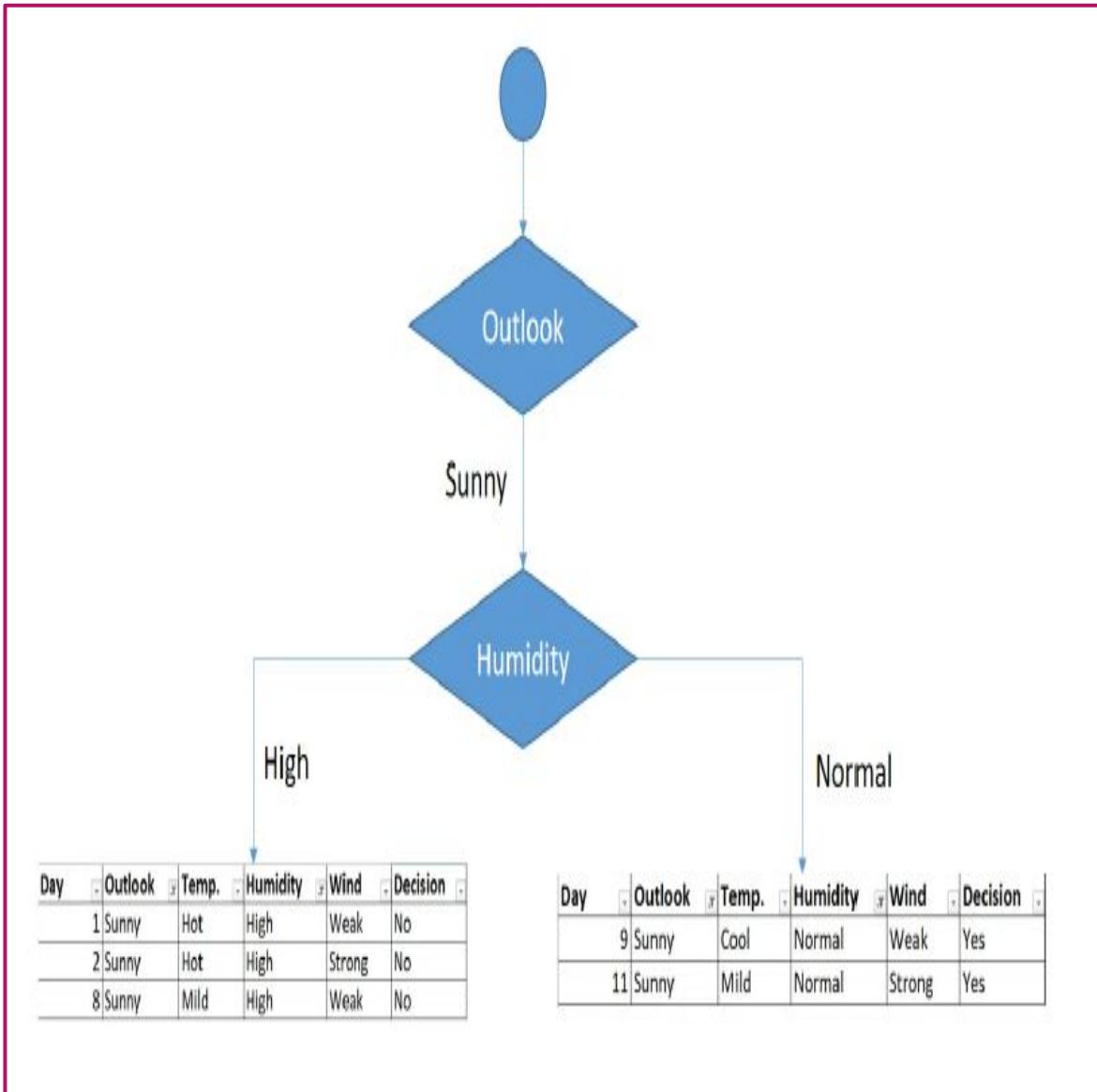
$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Wind}=\text{Strong}) = 1 - (1/2)^2 - (1/2)^2 = 0.2$$

$$\text{Gini}(\text{Outlook}=\text{Sunny} \text{ and } \text{Wind}) = (3/5) \times 0.266 + (2/5) \times 0.2 = 0.466$$

## Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466



# Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

## Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Temp.}=\text{Cool}) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Temp.}=\text{Mild}) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Temp.}) = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$

## Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Humidity}=\text{High}) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Humidity}=\text{Normal}) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Humidity}) = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$

## Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Wind}=\text{Weak}) = 1 - (3/3)^2 - (0/3)^2 = 0$$

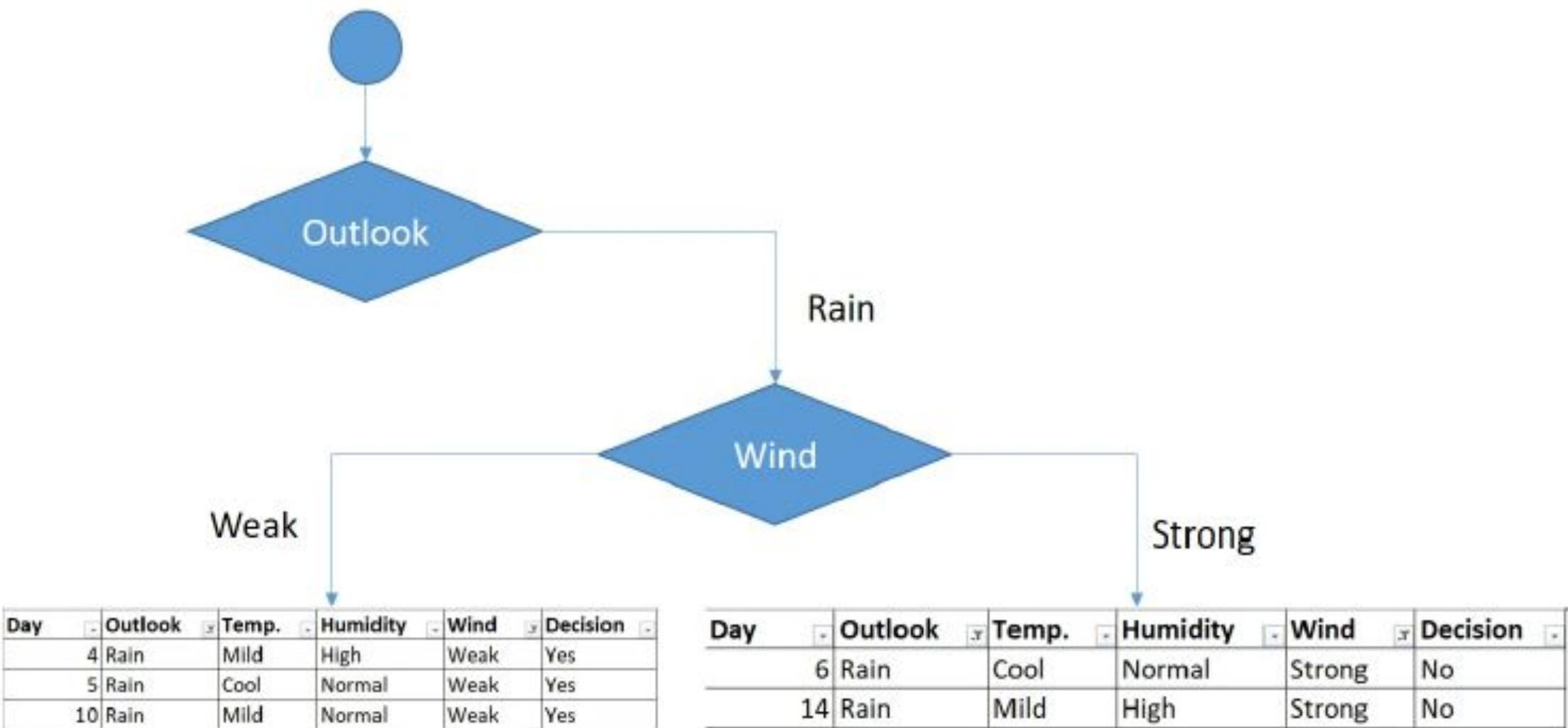
$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and } \text{Wind}=\text{Strong}) = 1 - (0/2)^2 - (2/2)^2 = 0$$

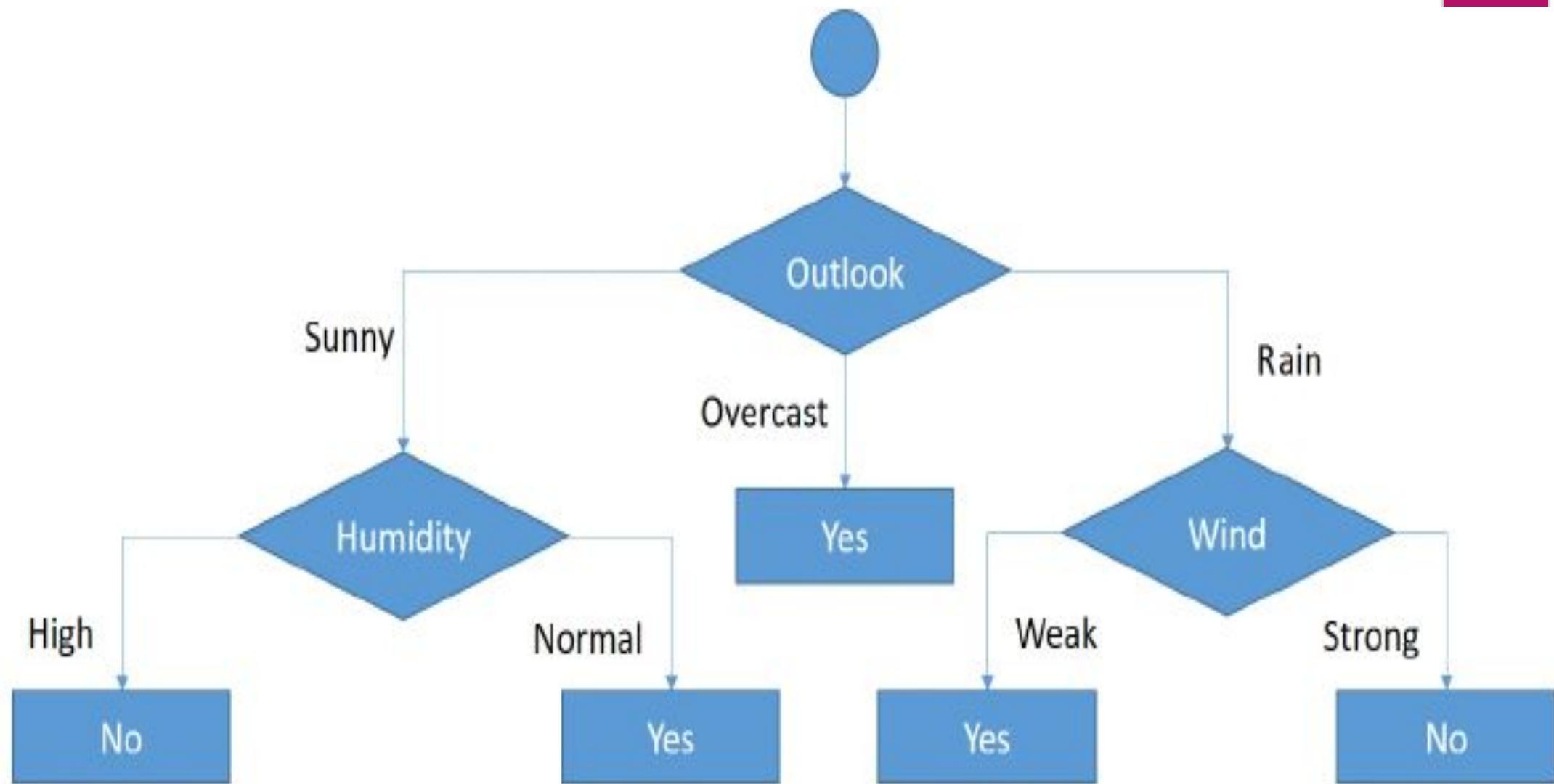
$$\text{Gini}(\text{Outlook}=\text{Rain} \text{ and Wind}) = (3/5) \times 0 + (2/5) \times 0 = 0$$

## Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0





# Association Rule Mining

## Decision Theory- Naïve Bayes

**Supervised Learning**

# Naïve Bayesian Classifier

According to Bayes' theorem, the probability that we want to compute  $P(H|X)$  can be expressed in terms of probabilities  $P(H)$ ,  $P(X|H)$ , and  $P(X)$  as

$$P(H|X) = \frac{P(X|H) P(H)}{P(X)},$$

and these probabilities may be estimated from the given data.

## Naive Bayesian Classifier

The naive Bayesian classifier works as follows:

- Let  $T$  be a training set of samples, each with their class labels. There are  $k$  classes,  $C_1, C_2, \dots, C_k$ . Each sample is represented by an  $n$ -dimensional vector,  $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$ , depicting  $n$  measured values of the  $n$  attributes,  $A_1, A_2, \dots, A_n$ , respectively.

That is  $\mathbf{X}$  is predicted to belong to the class  $C_i$  if and only if

$$P(C_i|\mathbf{X}) > P(C_j|\mathbf{X}) \quad \text{for } 1 \leq j \leq m, \ j \neq i.$$

Thus we find the class that maximizes  $P(C_i|\mathbf{X})$ . The class  $C_i$  for which  $P(C_i|\mathbf{X})$  is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i) P(C_i)}{P(\mathbf{X})}.$$

As  $P(\mathbf{X})$  is the same for all classes, only  $P(\mathbf{X}|C_i)P(C_i)$  need be maximized. If the class a priori probabilities,  $P(C_i)$ , are not known, then it is commonly assumed that the classes are equally likely, that is,  $P(C_1) = P(C_2) = \dots = P(C_k)$ , and we would therefore maximize  $P(\mathbf{X}|C_i)$ . Otherwise we maximize  $P(\mathbf{X}|C_i)P(C_i)$ . Note that the class a priori probabilities may be estimated by  $P(C_i) = \text{freq}(C_i, T)/|T|$ .

$$P(\mathbf{X}|C_i) \approx \prod_{k=1}^n P(x_k|C_i).$$

The probabilities  $P(x_1|C_i), P(x_2|C_i), \dots, P(x_n|C_i)$  can easily be estimated from the training set. Recall that here  $x_k$  refers to the value of attribute  $A_k$  for sample  $\mathbf{X}$ .

- (a) If  $A_k$  is categorical, then  $P(x_k|C_i)$  is the number of samples of class  $C_i$  in  $T$  having the value  $x_k$  for attribute  $A_k$ , divided by  $\text{freq}(C_i, T)$ , the number of sample of class  $C_i$  in  $T$ .
- (b) If  $A_k$  is continuous-valued, then we typically assume that the values have a Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$  defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{(x - \mu)^2}{2\sigma^2},$$

$$p(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

We need to compute  $\mu_{C_i}$  and  $\sigma_{C_i}$ , which are the mean and standard deviation of values of attribute  $A_k$  for training samples of class  $C_i$ .

In order to predict the class label of  $\mathbf{X}$ ,  $P(\mathbf{X}|C_i)P(C_i)$  is evaluated for each class  $C_i$ . The classifier predicts that the class label of  $\mathbf{X}$  is  $C_i$  if and only if it is the class that maximizes  $P(\mathbf{X}|C_i)P(C_i)$ .

## Problem Statement

RID	age	income	student	credit	$C_i$ : buy
1	youth	high	no	fair	$C_2$ : no
2	youth	high	no	excellent	$C_2$ : no
3	middle-aged	high	no	fair	$C_1$ : yes
4	senior	medium	no	fair	$C_1$ : yes
5	senior	low	yes	fair	$C_1$ : yes
6	senior	low	yes	excellent	$C_2$ : no
7	middle-aged	low	yes	excellent	$C_1$ : yes
8	youth	medium	no	fair	$C_2$ : no
9	youth	low	yes	fair	$C_1$ : yes
10	senior	medium	yes	fair	$C_1$ : yes
11	youth	medium	yes	excellent	$C_1$ : yes
12	middle-aged	medium	no	excellent	$C_1$ : yes
13	middle-aged	high	yes	fair	$C_1$ : yes
14	senior	medium	no	excellent	$C_2$ : no

$X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit} = \text{fair})$  Class label=?

# Solution

We need to maximize  $P(\mathbf{X}|C_i)P(C_i)$ , for  $i = 1, 2$ .  $P(C_i)$ , the a priori probability of each class, can be estimated based on the training samples:

$$P(\text{buy} = \text{yes}) = \frac{9}{14}$$

$$P(\text{buy} = \text{no}) = \frac{5}{14}$$

To compute  $P(\mathbf{X}|C_i)$ , for  $i = 1, 2$ , we compute the following conditional probabilities:

$$P(\text{age} = \text{youth} | \text{buy} = \text{yes}) = \frac{2}{9}$$

$$P(\text{age} = \text{youth} | \text{buy} = \text{no}) = \frac{3}{5}$$

$$P(\text{income} = \text{medium} | \text{buy} = \text{yes}) = \frac{4}{9}$$

$$P(\text{income} = \text{medium} | \text{buy} = \text{no}) = \frac{2}{5}$$

$$P(\text{student} = \text{yes} | \text{buy} = \text{yes}) = \frac{6}{9}$$

$$P(\text{student} = \text{yes} | \text{buy} = \text{no}) = \frac{1}{5}$$

$$P(\text{credit} = \text{fair} | \text{buy} = \text{yes}) = \frac{6}{9}$$

$$P(\text{credit} = \text{fair} | \text{buy} = \text{no}) = \frac{2}{5}$$

Using the above probabilities, we obtain

$$\begin{aligned} P(\mathbf{X} | \text{buy} = \text{yes}) &= P(\text{age} = \text{youth} | \text{buy} = \text{yes}) \\ &\quad P(\text{income} = \text{medium} | \text{buy} = \text{yes}) \\ &\quad P(\text{student} = \text{yes} | \text{buy} = \text{yes}) \\ &\quad P(\text{credit} = \text{fair} | \text{buy} = \text{yes}) \\ &= \frac{2}{9} \frac{4}{9} \frac{6}{9} \frac{6}{9} = 0.044. \end{aligned}$$

$$P(\mathbf{X}|buy = no) = \frac{3}{5} \frac{2}{5} \frac{1}{5} \frac{2}{5} = 0.019$$

To find the class that maximizes  $P(\mathbf{X}|C_i)P(C_i)$ , we compute

$$P(\mathbf{X}|buy = yes)P(buy = yes) = 0.028$$

$$P(\mathbf{X}|buy = no)P(buy = no) = 0.007$$

Thus the naive Bayesian classifier predicts  $buy = yes$  for sample  $\mathbf{X}$ .

# Pros & Cons

Advantages :

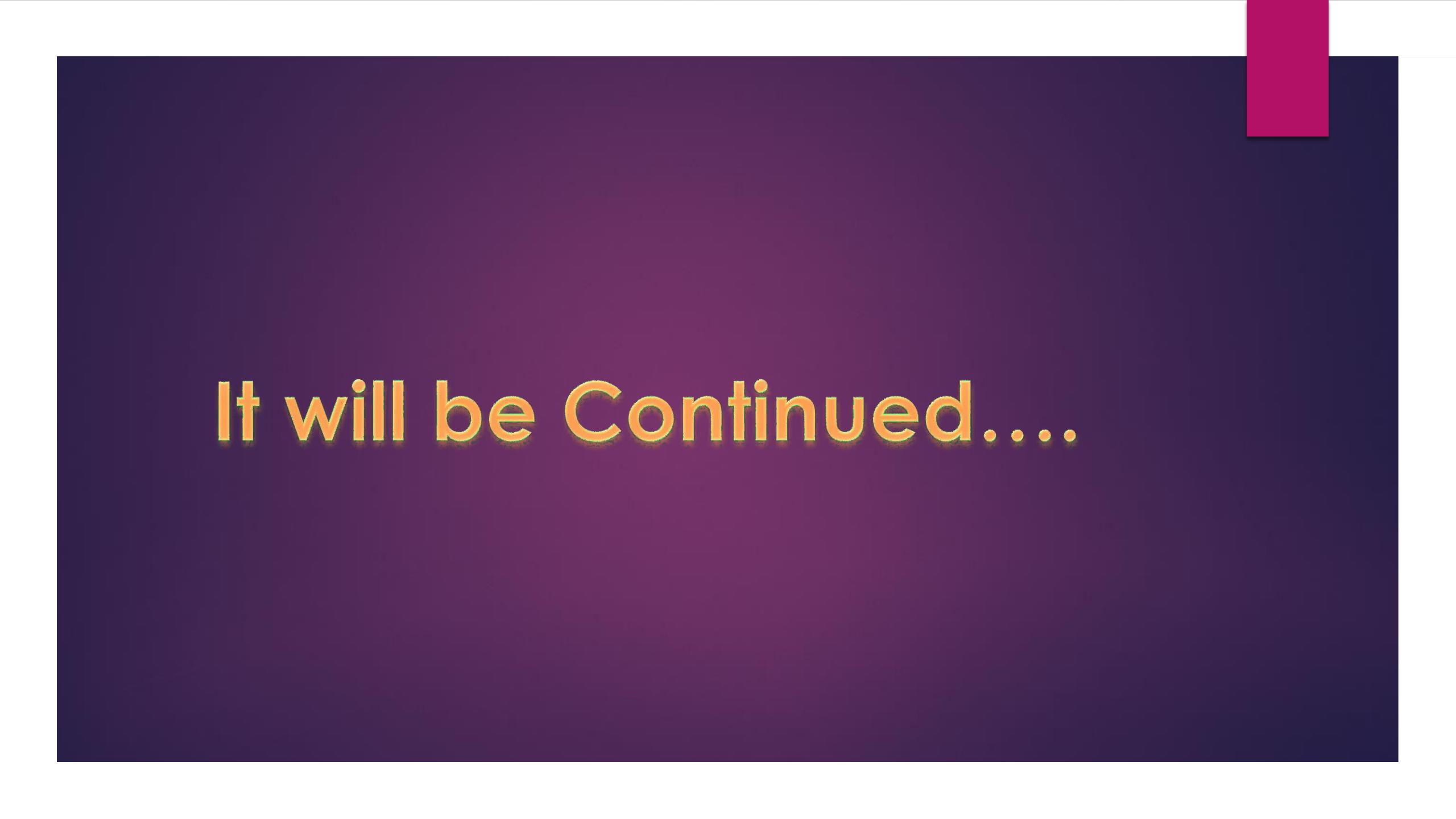
- No preprocessing needed on data.
- No assumptions on distribution of data.
- Handles co linearity efficiently.
- Decision trees can provide understandable explanation over the prediction.

Disadvantages :

- Chances for overfitting the model if we keep on building the tree to achieve high purity. decision tree pruning can be used to solve this issue.
- Prone to outliers.
- Tree may grow to be very complex while training complicated datasets.
- Loses valuable information while handling continuous variables.

# Decision tree vs naive Bayes :

- Decision tree is a discriminative model, whereas Naive bayes is a generative model.
- Decision trees are more flexible and easy.
- Decision tree pruning may neglect some key values in training data, which can lead the accuracy.
- A major advantage to Naive Bayes classifiers is that they are not prone to overfitting, thanks to the fact that they “ignore” irrelevant features.
- Naive Bayes classifiers are easily implemented and highly scalable, with a linear computational complexity with respect to the number of data entries.



It will be Continued...

# Learning

BY

DR. ANUPAM GHOSH

DATE: 02.09.2023

Email: [anupam.ghosh@rediffmail.com](mailto:anupam.ghosh@rediffmail.com)

LinkedIn Profile:  
<https://www.linkedin.com/in/anupam-ghosh-1504273b/>

Research Profile:  
[https://www.researchgate.net/profile/Anupam\\_Ghosh14](https://www.researchgate.net/profile/Anupam_Ghosh14)

# Supervised vs. Unsupervised

fruit	length	width	weight	label
fruit 1	165	38	172	Banana
fruit 2	218	39	230	Banana
fruit 3	76	80	145	Orange
fruit 4	145	35	150	Banana
fruit 5	90	88	160	Orange
...	...	...	...	...
fruit n	...	...	...	...

## Unsupervised learning:

Learning a model from **unlabeled** data.

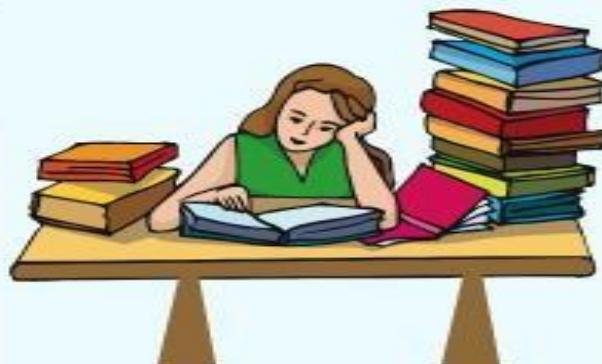
## Supervised learning:

Learning a model from **labeled** data.

## Supervised Learning



## Unsupervised Learning



# Styles of Learning

## Supervised

- Data has **known labels** or output

## Unsupervised

- Labels or output unknown
- Focus on **finding patterns and gaining insight** from the data

## Semi-Supervised

- Labels or output known for a **subset of data**
- A blend of supervised and unsupervised learning

## Reinforcement

- Focus on **making decisions** based on previous experience
- Policy-making with feedback

- Insurance underwriting
- Fraud detection

- Customer clustering
- Association rule mining

- Medical predictions (where tests and expert diagnoses are expensive, and only part of the population receives them)

- Game AI
- Complex decision problems
- Reward systems

# Unsupervised Learning

# Clustering

- Clustering: Intuitively, finding clusters of points in the given data such that similar points lie in the same cluster
- Can be formalized using distance metrics in several ways
  - Group points into  $k$  sets (for a given  $k$ ) such that the average distance of points from the centroid of their assigned group is minimized
    - ▶ Centroid: point defined by taking average of coordinates in each dimension.
    - Another metric: minimize average distance between every pair of points in a cluster
- Has been studied extensively in statistics, but on small data sets
  - Data mining systems aim at clustering techniques that can handle very large data sets
  - E.g., the Birch clustering algorithm (more shortly)

# Clustering

- ▶ What is clustering?

# Clustering

- ▶ Definition
  - ▶ Assignment of a set of observations into subsets so that observations in the same subset are similar in some sense
- ▶ Hard vs. Soft
  - ▶ Hard: same object can only belong to single cluster
  - ▶ Soft: same object can belong to different clusters
- ▶ Flat vs. Hierarchical
  - ▶ Flat: clusters are flat
  - ▶ Hierarchical: clusters form a tree
    - ▶ Agglomerative
    - ▶ Divisive

# Similarity measures

- ▶ How to determine similarity between data points
  - ▶ using various distance metrics
- ▶ Let  $\mathbf{x} = (x_1, \dots, x_n)$  and  $\mathbf{y} = (y_1, \dots, y_n)$  be n-dimensional vectors of data points of objects  $g_1$  and  $g_2$ 
  - ▶  $g_1, g_2$  can be two different genes in microarray data
  - ▶ n can be the number of samples

# Summary of similarity measures

- ▶ Using different measures for clustering can yield different clusters
- ▶ Euclidean distance and correlation distance are the most common choices of similarity measure for microarray data
- ▶ Euclidean vs Correlation Example
  - ▶  $g1 = (1,2,3,4,5)$
  - ▶  $g2 = (100,200,300,400,500)$
  - ▶  $g3 = (5,4,3,2,1)$
  - ▶ Which genes are similar according to the two different measures?

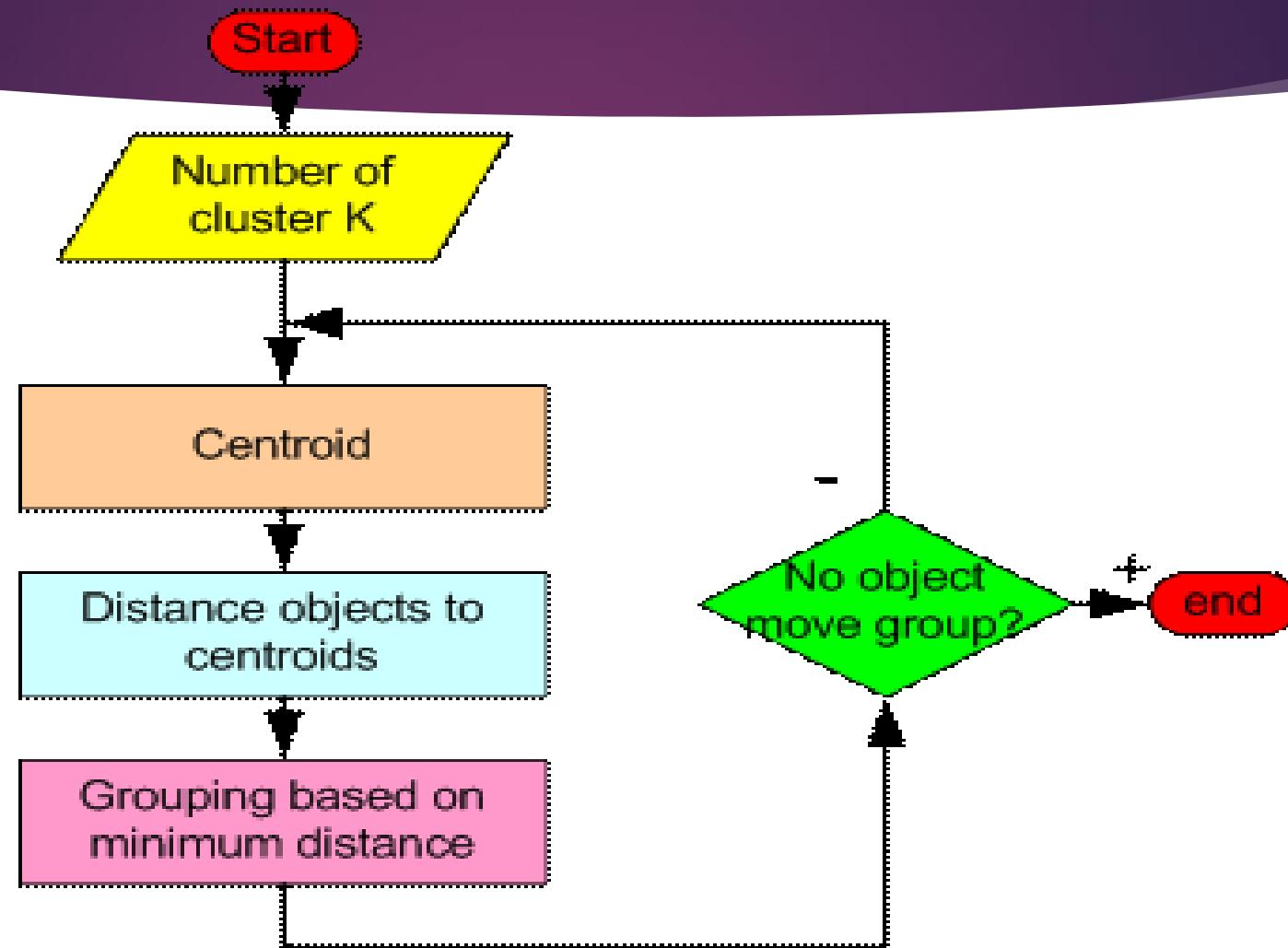
# K-MEANS CLUSTERING

- ▶ The **k-means algorithm** is an algorithm to cluster  $n$  objects based on attributes into  $k$  partitions, where  $k < n$ .
  - ▶ It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
  - ▶ It assumes that the object attributes form a vector space.
- 
- ▶ An algorithm for partitioning (or clustering)  $N$  data points into  $K$  disjoint subsets  $S_j$  containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^K \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where  $x_n$  is a vector representing the  $n^{\text{th}}$  data point and  $\mu_j$  is the geometric centroid of the data points in  $S_j$ .

# How the K-Mean Clustering algorithm works?



- ▶ **Step 1:** Begin with a decision on the value of  $k$  = number of clusters .
- ▶ **Step 2:**  
Put any initial partition that classifies the data into  $k$  clusters. You may assign the training samples randomly, or systematically as the following:
  - 1.Take the first  $k$  training sample as single-element clusters
  2. Assign each of the remaining  $(N-k)$  training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

- ▶ **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters.

If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

- ▶ **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

# A Simple example showing the implementation of k-means algorithm (using K=2)

Individual	Variable 1	Variable 2
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

## Step 1:

Initialization: Randomly we choose following two centroids ( $k=2$ ) for two clusters.

In this case the 2 centroid are:  $m_1=(1.0, 1.0)$  and  $m_2=(5.0, 7.0)$ .

Individual	Variable 1	Variable 2
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

Individual	Mean Vector
Group 1	(1.0, 1.0)
Group 2	(5.0, 7.0)

## Step 2:

- Thus, we obtain two clusters containing:

$\{1,2,3\}$  and  $\{4,5,6,7\}$ .

- Their new centroids are:

$$m_1 = \left( \frac{1}{3}(1.0 + 1.5 + 3.0), \frac{1}{3}(1.0 + 2.0 + 4.0) \right) = (1.83, 2.33)$$

$$\begin{aligned} m_2 &= \left( \frac{1}{4}(5.0 + 3.5 + 4.5 + 3.5), \frac{1}{4}(7.0 + 5.0 + 5.0 + 4.5) \right) \\ &= (4.12, 5.38) \end{aligned}$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 2) = \sqrt{|1.0 - 1.5|^2 + |1.0 - 2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0 - 1.5|^2 + |7.0 - 2.0|^2} = 6.10$$

### Step 3:

- ▶ Now using these centroids we compute the Euclidean distance of each object, as shown in table.
- ▶ Therefore, the new clusters are:  
 $\{1,2\}$  and  $\{3,4,5,6,7\}$
- ▶ Next centroids are:  
 $m_1 = (1.25, 1.5)$  and  $m_2 = (3.9, 5.1)$

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.84	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

► Step 4 :

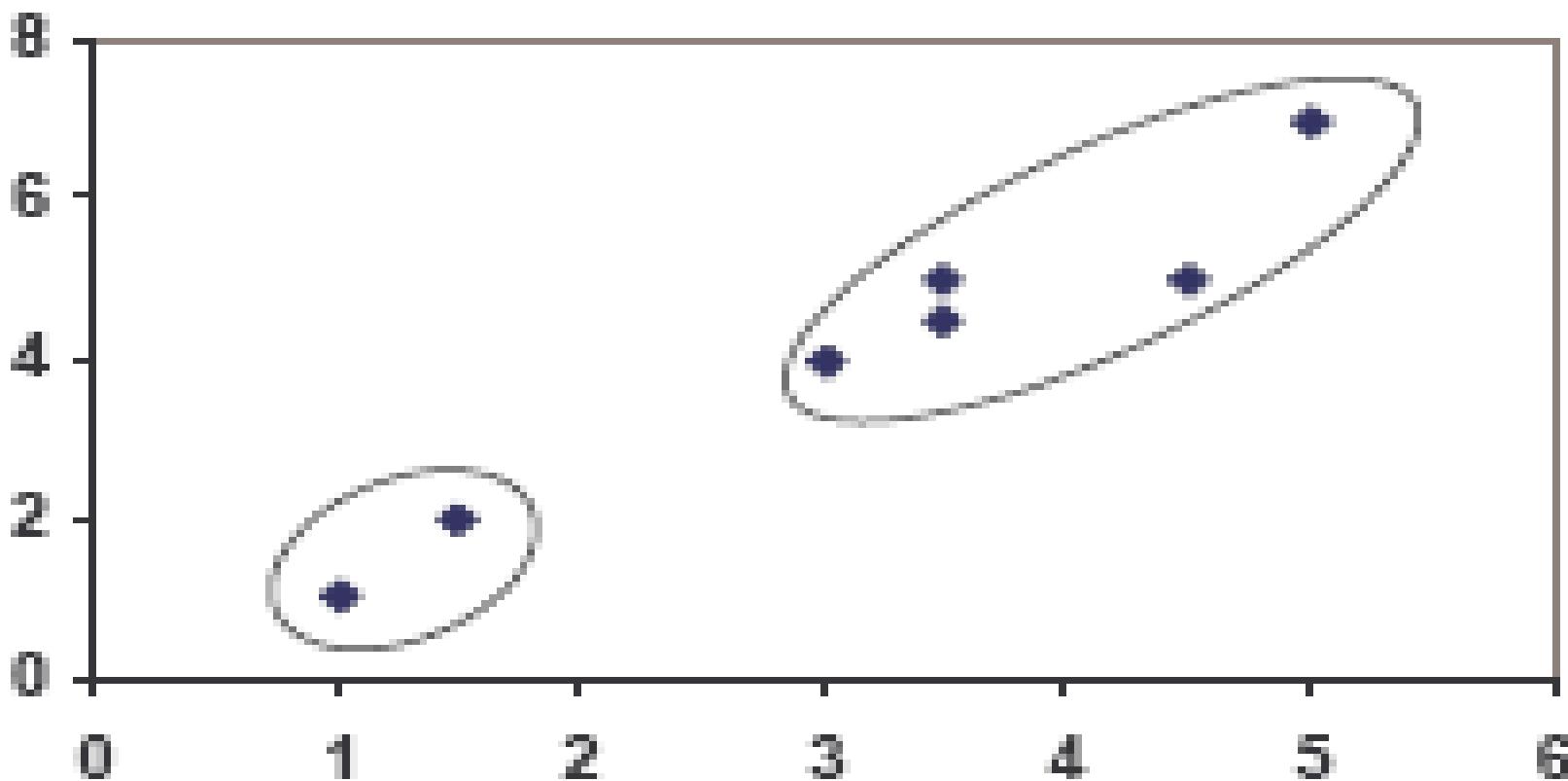
The clusters obtained are:

{1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.58	5.02
2	0.58	3.92
3	3.05	1.42
4	6.88	2.20
5	4.18	0.41
6	4.78	0.61
7	3.75	0.72

# PLOT

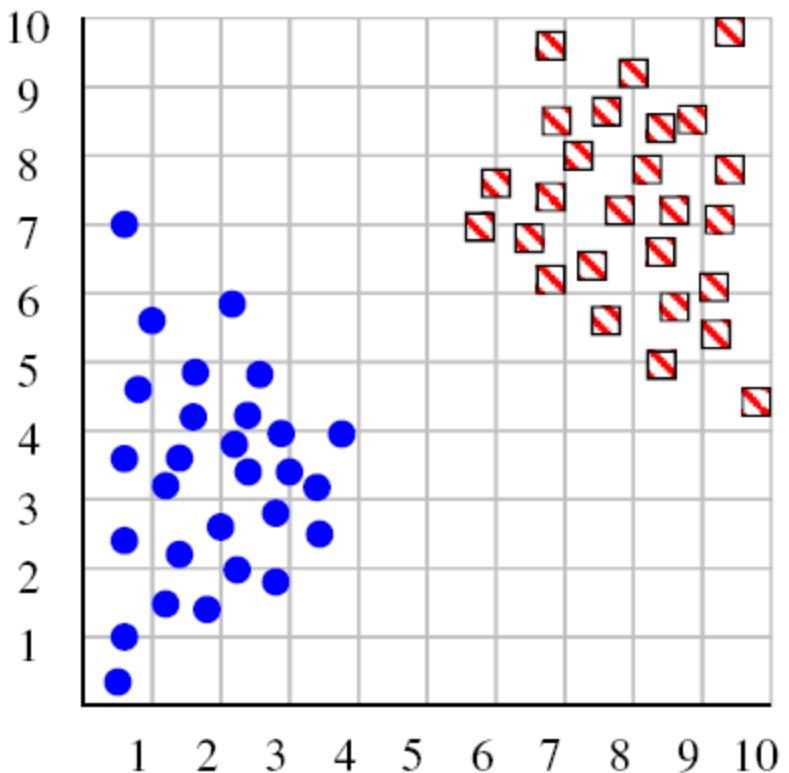


# K-means: summary

- ▶ Algorithmically, very simple to implement
- ▶ K-means converges, but it finds a local minimum of the cost function
- ▶ Works only for numerical observations
- ▶  $K$  is a user input; alternatively BIC (Bayesian information criterion) or MDL (minimum description length) can be used to estimate  $K$
- ▶ Outliers can considerable trouble to K-means

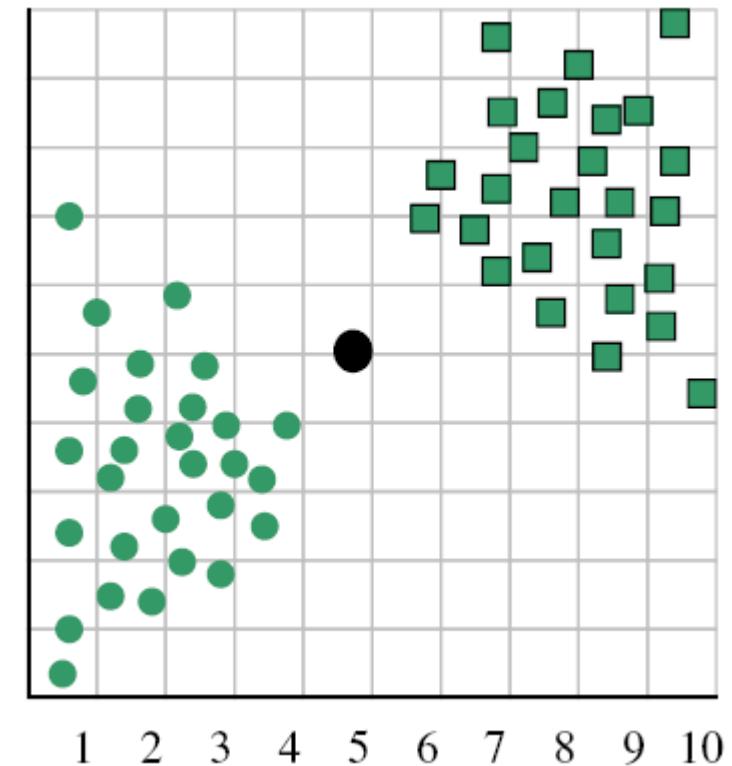
# Deciding K

- ▶ Try a couple of K



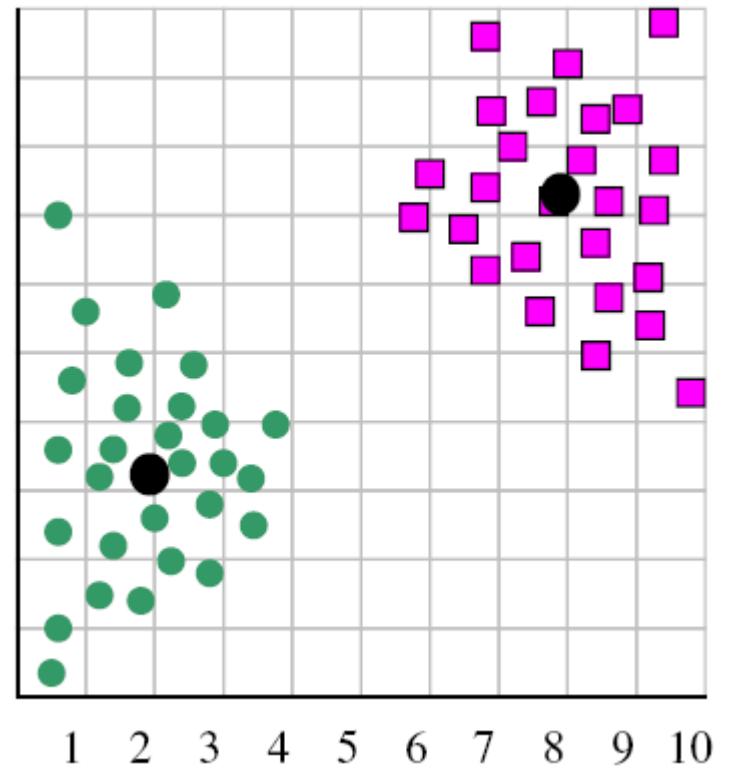
# Deciding K

- ▶ When  $k = 1$ , the objective function is 873.0



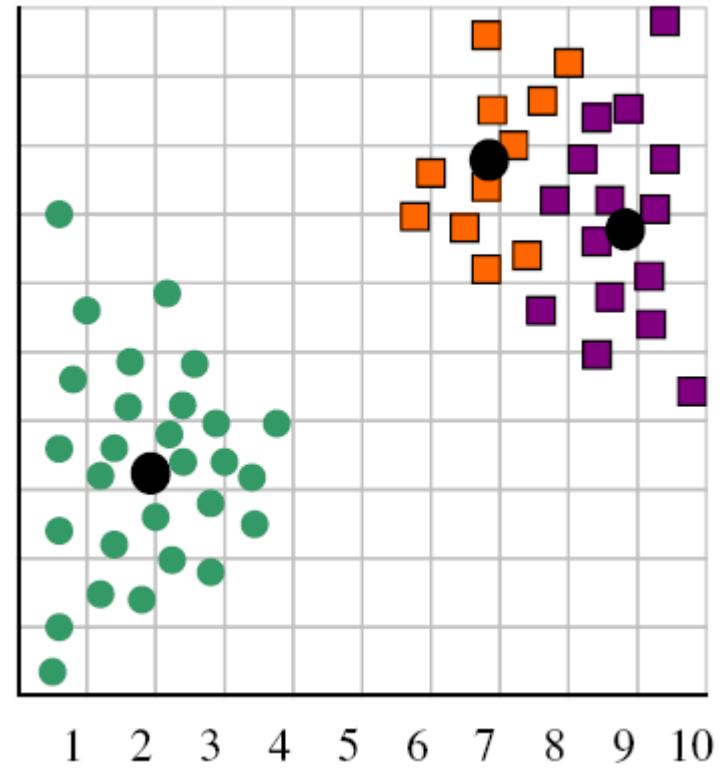
# Deciding K

- ▶ When  $k = 2$ , the objective function is 173.1



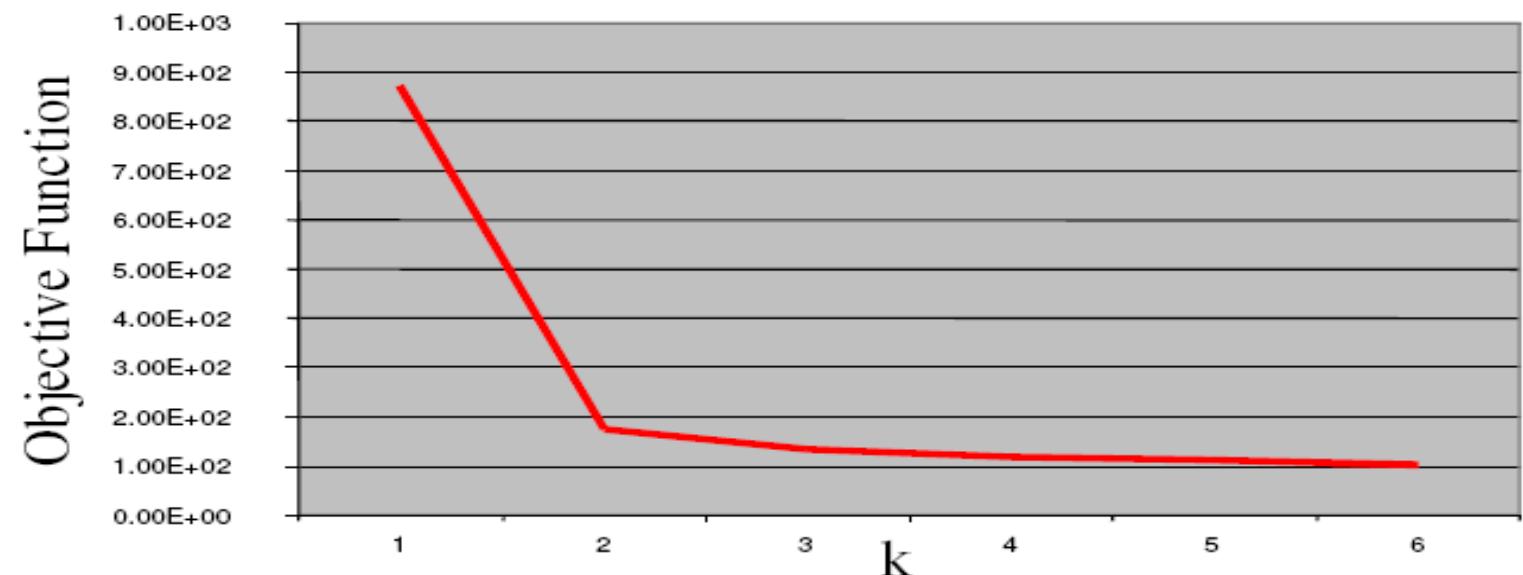
# Deciding K

- ▶ When  $k = 3$ , the objective function is 133.6



# Deciding K

- ▶ We can plot objective function values for k=1 to 6
- ▶ The abrupt change at k=2 is highly suggestive of two clusters
- ▶ “knee finding” or “elbow finding”
- ▶ Note that the results are not always as clear cut as in this toy example



Back

# Weaknesses of K-Means Clustering

1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K, must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the local optimum.

# Applications of K-Means Clustering

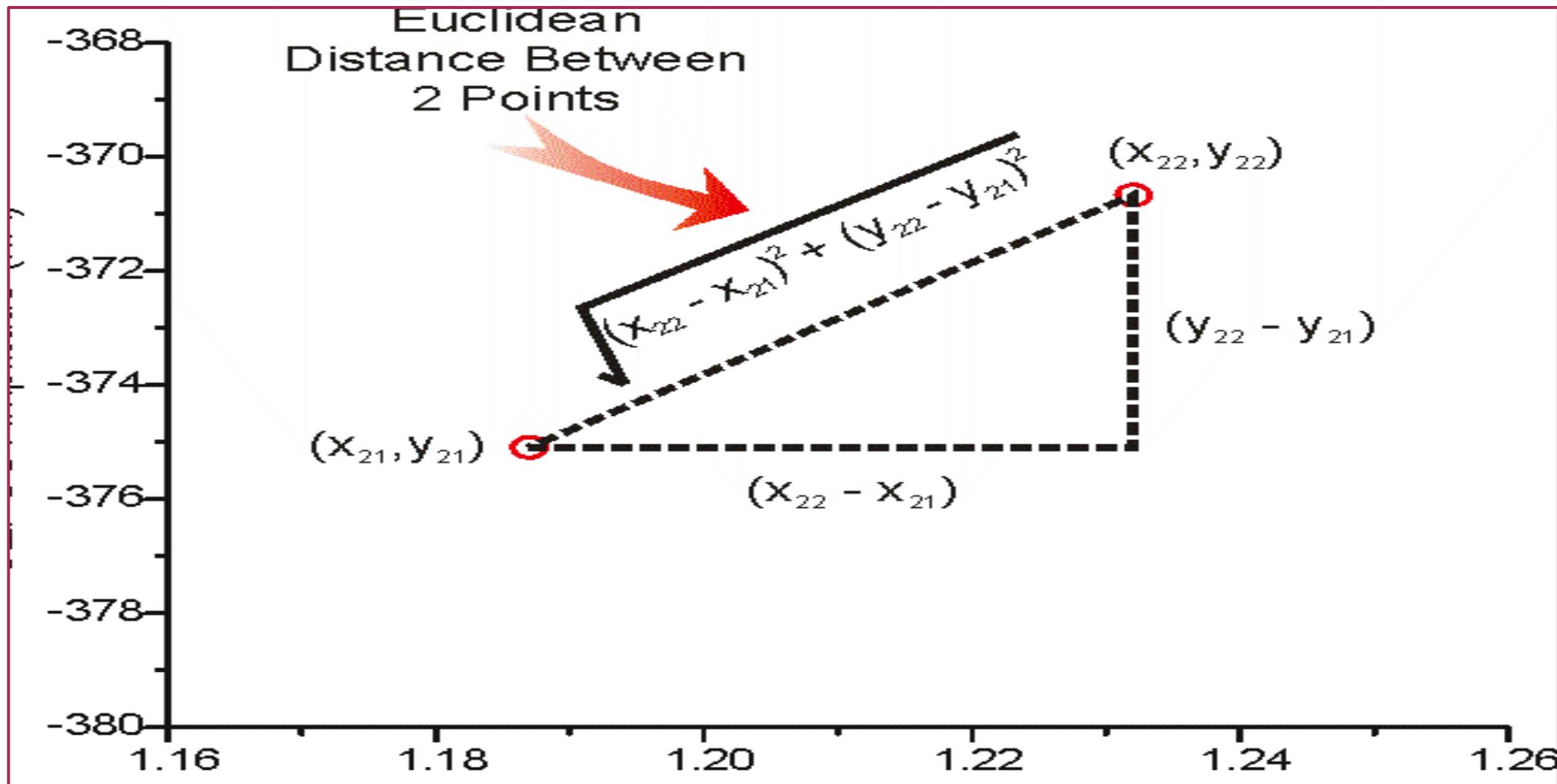
- ▶ It is relatively *efficient and fast*. It computes result at  **$O(tkn)$** , where n is number of objects or points, k is number of clusters and t is number of iterations.
- ▶ k-means clustering can be applied to *machine learning or data mining*
- ▶ Used on *acoustic data in speech understanding* to convert waveforms into one of k categories (known as *Vector Quantization or Image Segmentation*).
- ▶ Also used for choosing color palettes on *old fashioned graphical display devices and Image Quantization*.

# Geometric Distance Measure

- ▶ Geometric distance metrics, primarily, tends to measure the similarity between two or more vectors solely based on the distance between two points in multi-dimensional space.
- ▶ The examples of such type of geometric distance measures are Minkowski distance, Euclidean distance and Manhattan distance.
- ▶ Minkowski distance is the general form of Euclidean and Manhattan distance. Mathematically, it can be represented as the following:

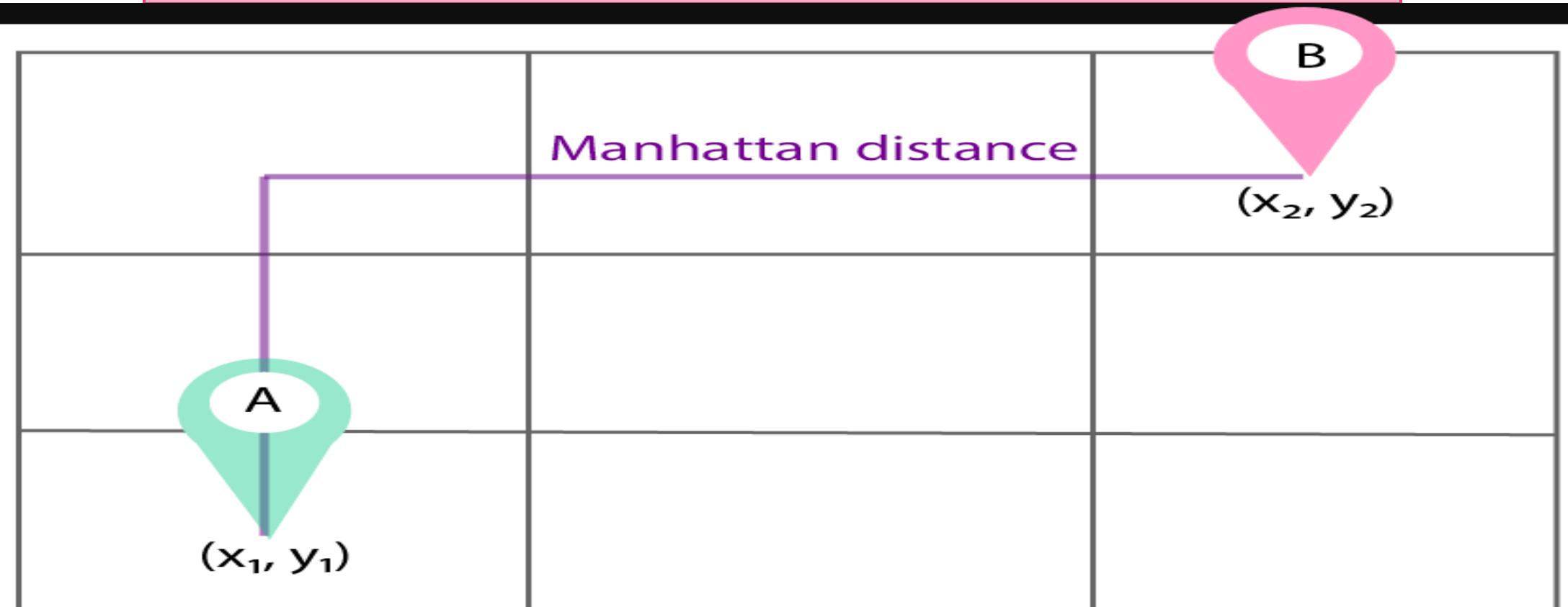
$$D(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{l=1}^d |x_{il} - x_{jl}|^{1/p} \right)^p$$

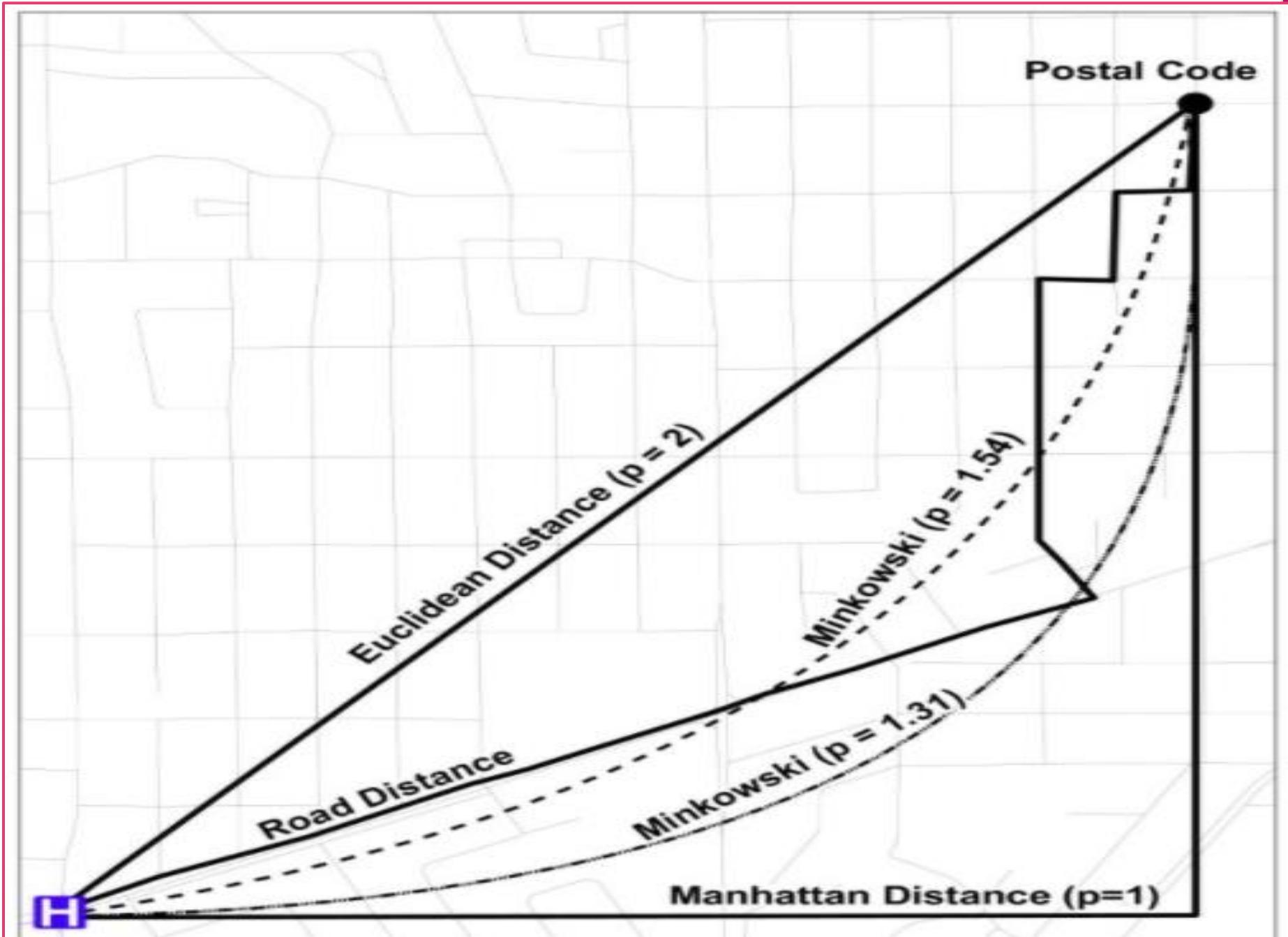
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



# Manhattan Distance

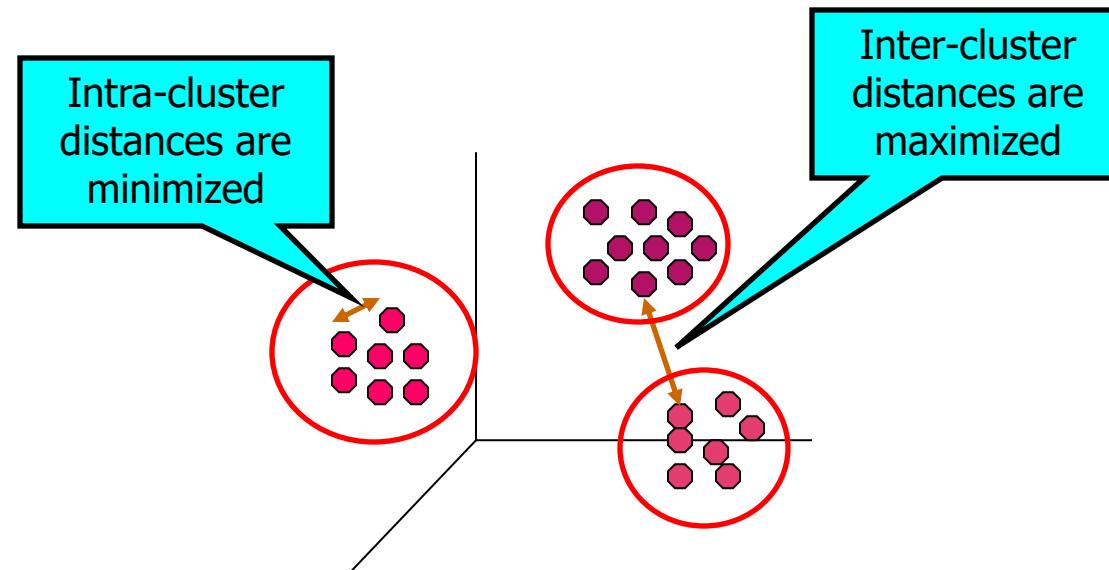
- ▶ If you want to find Manhattan distance between two different points  $(x_1, y_1)$  and  $(x_2, y_2)$  such as the following, it would look like the following:
- ▶ Manhattan distance =  $(x_2 - x_1) + (y_2 - y_1)$





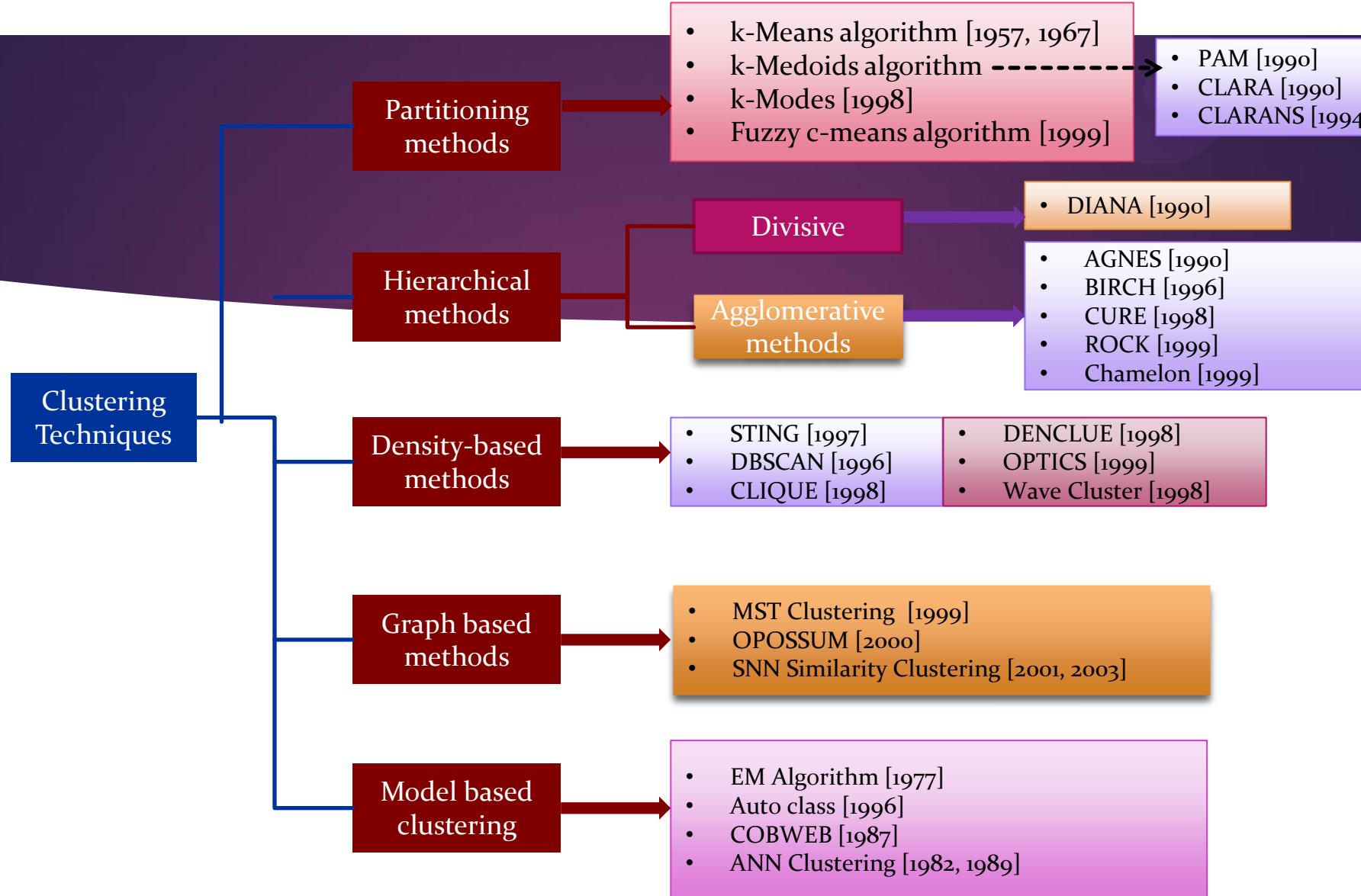
# What is Cluster Analysis?

- ▶ Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



# Quality: What Is Good Clustering?

- ▶ A good clustering method will produce high quality clusters with
  - ▶ high intra-class similarity
  - ▶ low inter-class similarity
- ▶ The quality of a clustering result depends on both the similarity measure used by the method and its implementation
- ▶ The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns



## K-MEDOIDS CLUSTERING

The mean in k-means clustering is sensitive to outliers. Since an object with an extremely high value may substantially distort the distribution of data.

Hence we move to k-medoids.

Instead of taking mean of cluster we take the most centrally located point in cluster as it's center.

These are called medoids.

## **k-Medoids**

- It is also called as Partitioning Around Medoid algorithm.
- A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.
- The dissimilarity of the medoid( $C_i$ ) and object( $P_i$ ) is calculated by using  $E = |P_i - C_i|$

### **Algorithm:**

1. Initialize: select k random points out of the n data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.
3. While the cost decreases:

For each medoid m, for each data o point which is not a medoid:

1. Swap m and o, associate each data point to the closest medoid, recompute the cost.
2. If the total cost is more than that in the previous step, undo the swap.

# K-MEDOIDS - PAM ALGORITHM

**PAM** stands for Partitioning Around Medoids.

**GOAL:** To find Clusters that have minimum average dissimilarity between objects that belong to same cluster.

**ALGORITHM:**

1. Start with initial set of medoids.
2. Iteratively replace one of the medoids with a non-medoid if it reduces total sum of SSE of resulting cluster.

SSE is calculated as below:

$$SSE(X) = \sum_{i=1}^k \sum_{x \in C_i} dist^2(m_i, x)$$

Where k is number of clusters and x is a data point in cluster  $C_i$  and  $M_i$  is medoid of  $C_i$

## **ADVANTAGES and DISADVANTAGES of PAM**

### **Advantages:**

PAM is more flexible as it can use any similarity measure.

PAM is more robust than k-means as it handles noise better.

### **Disadvantages:**

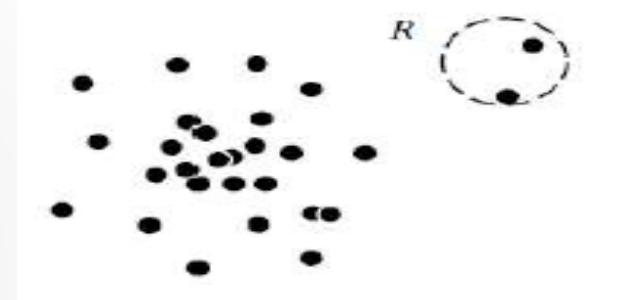
PAM algorithm for K-medoid clustering works well for dataset but cannot scale well for large data set due to high computational overhead.

**PAM COMPLEXITY :**  $O(k(n-k)^2)$  this is because we compute distance of  $n-k$  points with each  $k$  point, to decide in which cluster it will fall and after this we try to replace each of the medoid with a non medoid and find it's distance with  $n-k$  points.

# Outlier Detection in Clustering

- An **outlier** is a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism.
- Outliers are referred as “abnormal” data.
- Outliers are different from noisy data. Noise is a random error or variance in a measured variable.
- Outliers are interesting because they are suspected of not being generated by the same mechanisms as the rest of the data.
- Outlier detection is also related to novelty detection in evolving data sets.

*The objects in region R are outliers*



## Types of Outliers

- ✓ Global Outliers
- ✓ Contextual Outliers
- ✓ Collective Outliers

# **Outlier detection techniques.**

## **Supervised, Semi-Supervised, and Unsupervised Methods**

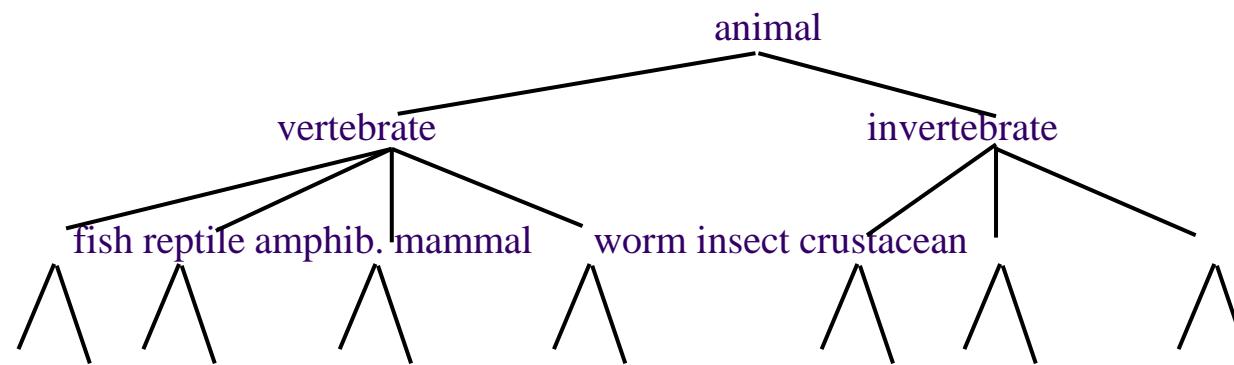
- Supervised methods model data normality and abnormality.
- In some application scenarios, objects labeled as “normal” or “outlier” are **not available**. Thus, an unsupervised learning method has to be used.
- In some cases where only a small set of the normal and/or outlier objects are labeled, but most of the data are unlabeled.

## **Statistical Methods, Proximity-Based Methods and Clustering-Based Methods**

- Statistical methods (also known as model-based methods) make assumptions of data normality.
- The effectiveness of proximity-based methods relies heavily on the proximity (or distance) measure used.
- Clustering-based methods assume that the normal data objects belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or do not belong to any clusters.

# Hierarchical Clustering

- ▶ Build a tree-based hierarchical taxonomy (dendrogram) from a set of documents.

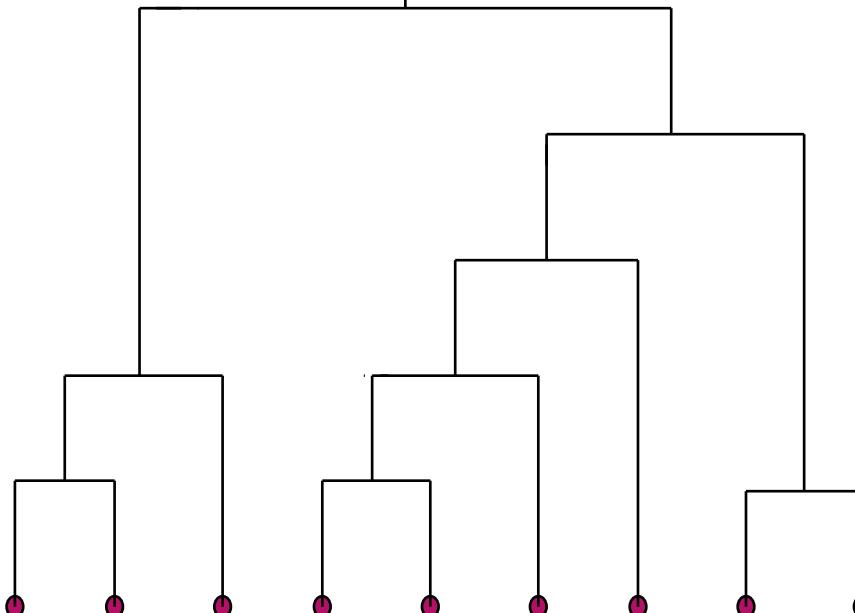


- ▶ One approach: recursive application of a partitional clustering algorithm.

# Dendrogram: Hierarchical Clustering

- ▶ Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.

▶ dendrogram

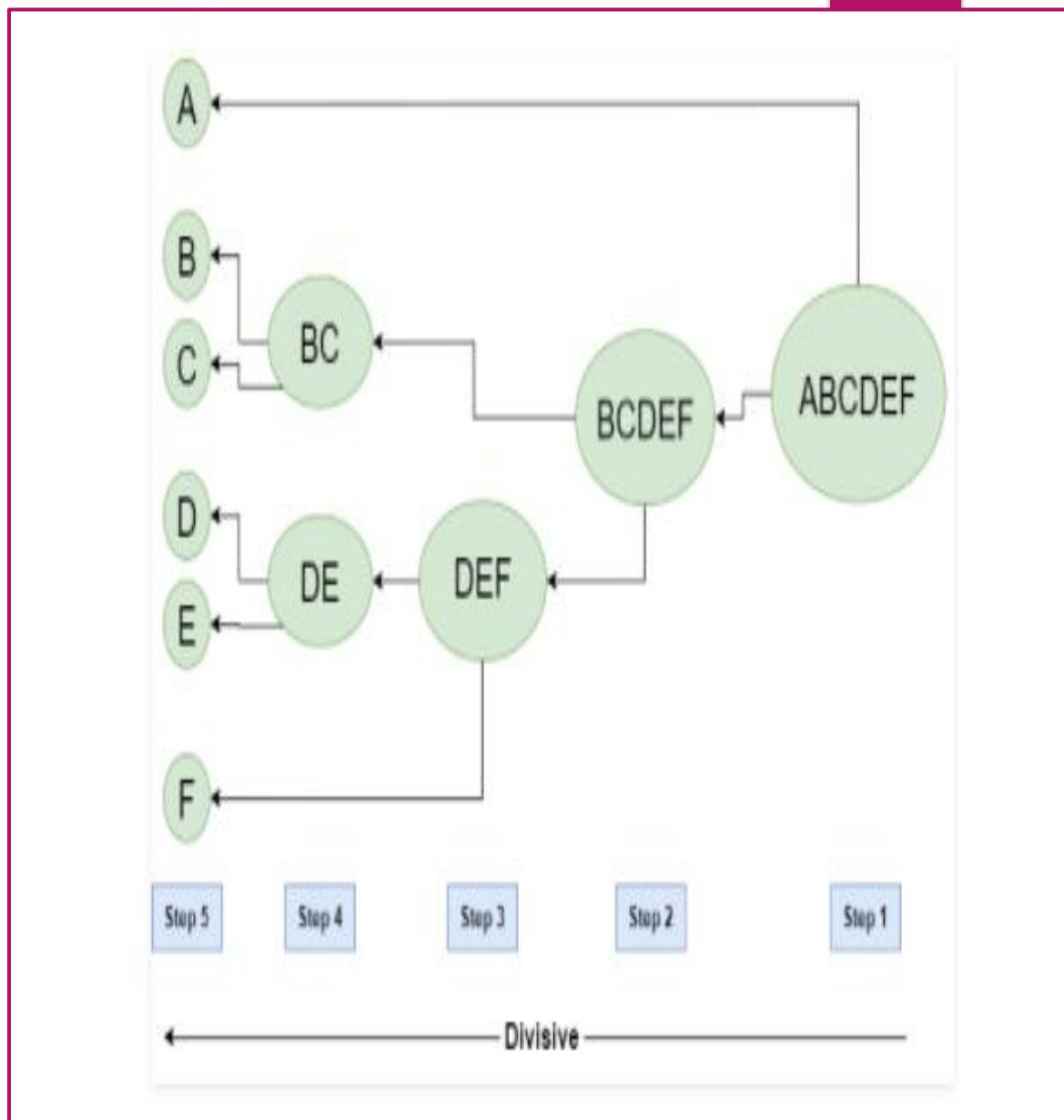
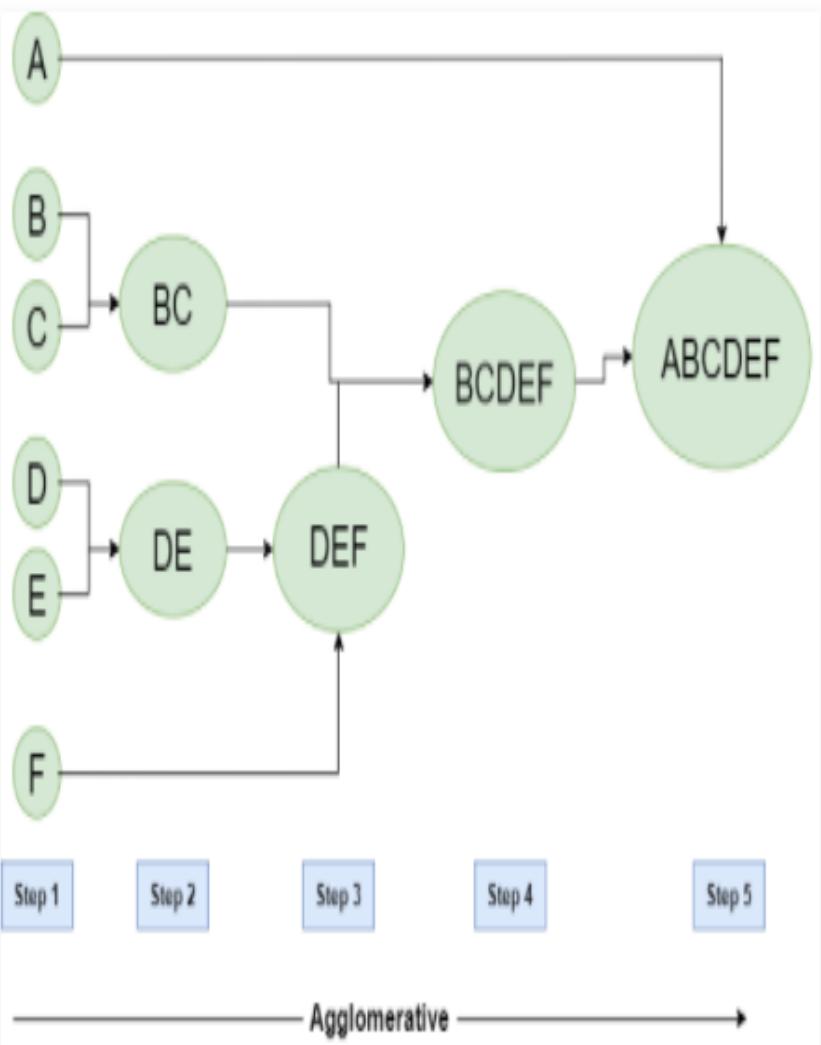


# Hierarchical Agglomerative Clustering (HAC)

Sec. 17.1

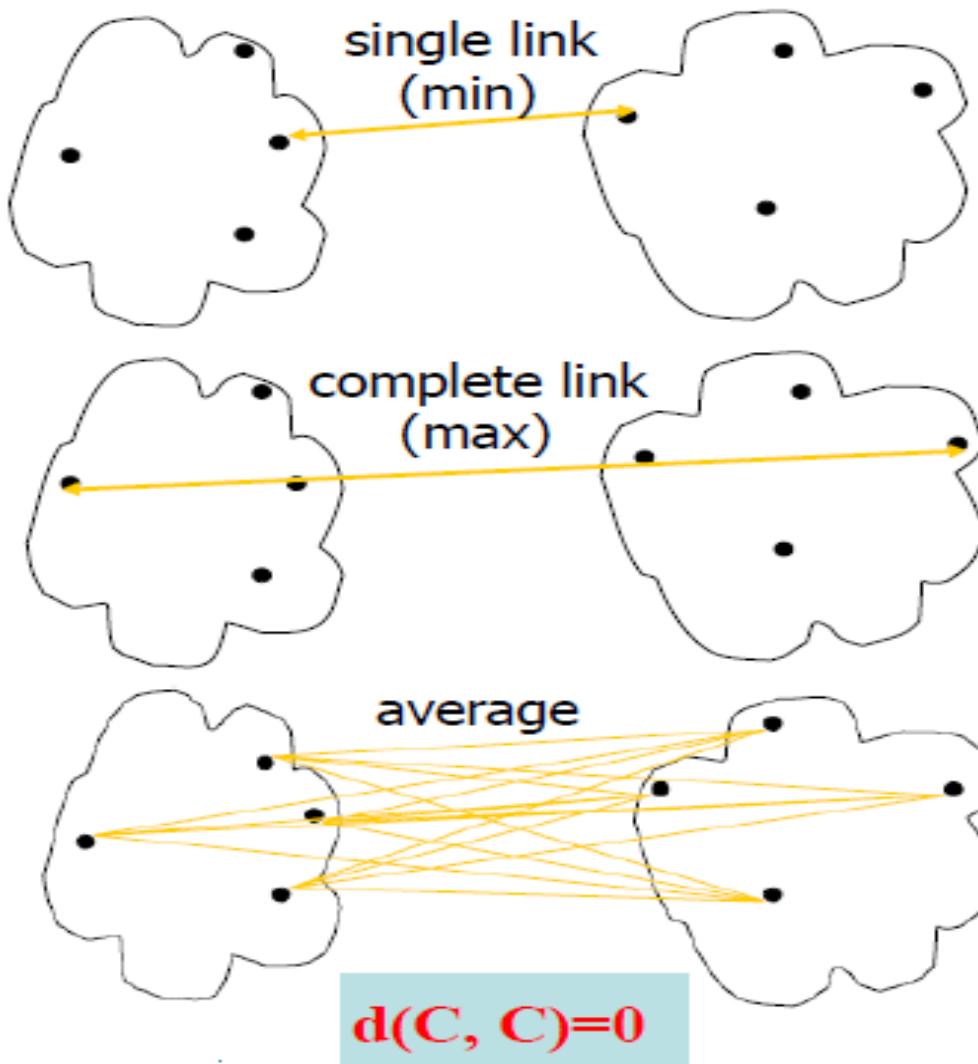
- ▶ Starts with each doc in a separate cluster
  - ▶ then repeatedly joins the closest pair of clusters, until there is only one cluster.
- ▶ The history of merging forms a binary tree or hierarchy.

Let's say we have six data points A, B, C, D, E, F.



# Cluster Distance Measures

- **Single link:** smallest distance between an element in one cluster and an element in the other, i.e.,  
 $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$
- **Complete link:** largest distance between an element in one cluster and an element in the other, i.e.,  
 $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$
- **Average:** avg distance between elements in one cluster and elements in the other, i.e.,  
 $d(C_i, C_j) = \text{avg}\{d(x_{ip}, x_{jq})\}$



# Cluster Distance Measures

**Example:** Given a data set of five objects characterised by a single continuous feature, assume that there are two clusters:  $C_1: \{a, b\}$  and  $C_2: \{c, d, e\}$ . (Minkowski distance for distance matrix)

	a	b	c	d	e
Feature	1	2	4	5	6

1. Calculate the distance matrix .

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

2. Calculate three cluster distances between  $C_1$  and  $C_2$ .

## Single link

$$\begin{aligned} \text{dist}(C_1, C_2) &= \min\{d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)\} \\ &= \min\{3, 4, 5, 2, 3, 4\} = 2 \end{aligned}$$

## Complete link

$$\begin{aligned} \text{dist}(C_1, C_2) &= \max\{d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)\} \\ &= \max\{3, 4, 5, 2, 3, 4\} = 5 \end{aligned}$$

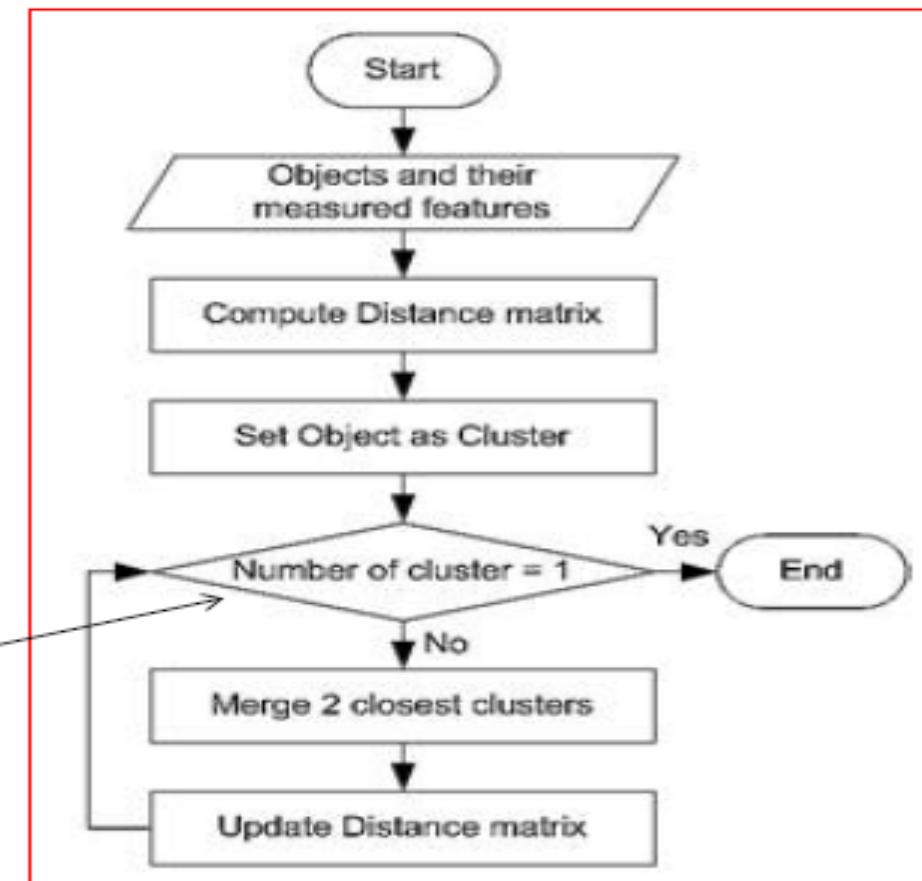
## Average

$$\begin{aligned} \text{dist}(C_1, C_2) &= \frac{d(a, c) + d(a, d) + d(a, e) + d(b, c) + d(b, d) + d(b, e)}{6} \\ &= \frac{3 + 4 + 5 + 2 + 3 + 4}{6} = \frac{21}{6} = 3.5 \end{aligned}$$

# Agglomerative Algorithm

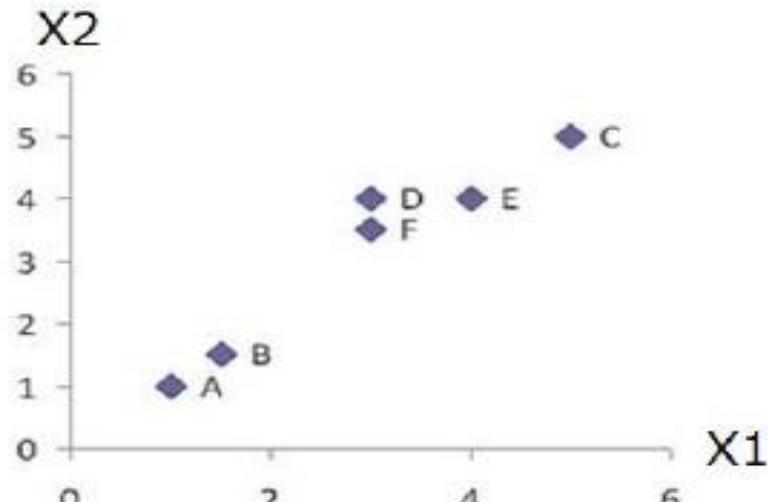
The *Agglomerative* algorithm is carried out in three steps:

- 1) Convert all object features into a distance matrix
- 2) Set each object as a cluster (thus if we have  $N$  objects, we will have  $N$  clusters at the beginning)
- 3) Repeat until number of cluster is one (or known # of clusters)
  - Merge two closest clusters
  - Update “distance matrix”



# Example

Problem: clustering analysis with agglomerative algorithm



$$d_{AB} = \sqrt{(1-1.5)^2 + (1-1.5)^2} = \sqrt{\frac{1}{2}} = 0.7071$$

$$d_{DF} = \sqrt{(3-3)^2 + (4-3.5)^2} = 0.5$$

Euclidean distance

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

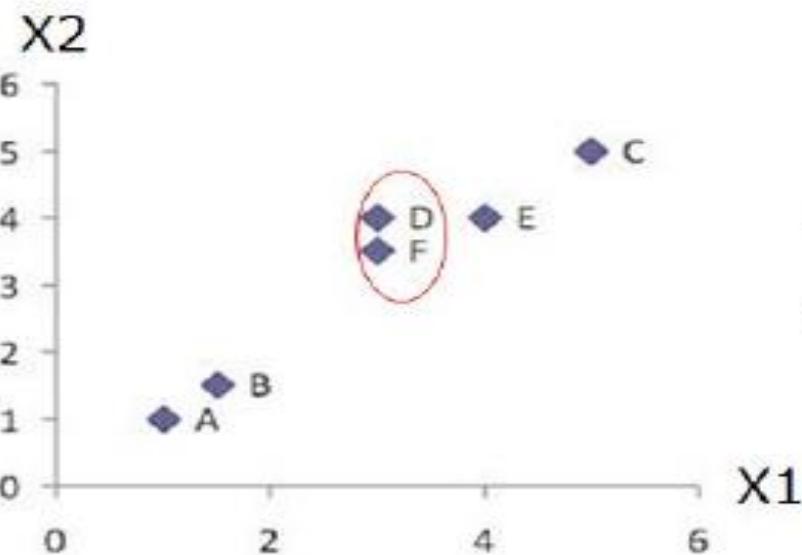
data matrix

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

distance matrix

# Example

Merge two closest clusters (iteration 1)



Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

- Update distance matrix (iteration 1)

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

$$d_{(D,F) \rightarrow A} = \min(d_{DA}, d_{FA}) = \min(3.61, 3.20) = 3.20$$

$$d_{(D,F) \rightarrow B} = \min(d_{DB}, d_{FB}) = \min(2.92, 2.50) = 2.50$$

$$d_{(D,F) \rightarrow C} = \min(d_{DC}, d_{FC}) = \min(2.24, 2.50) = 2.24$$

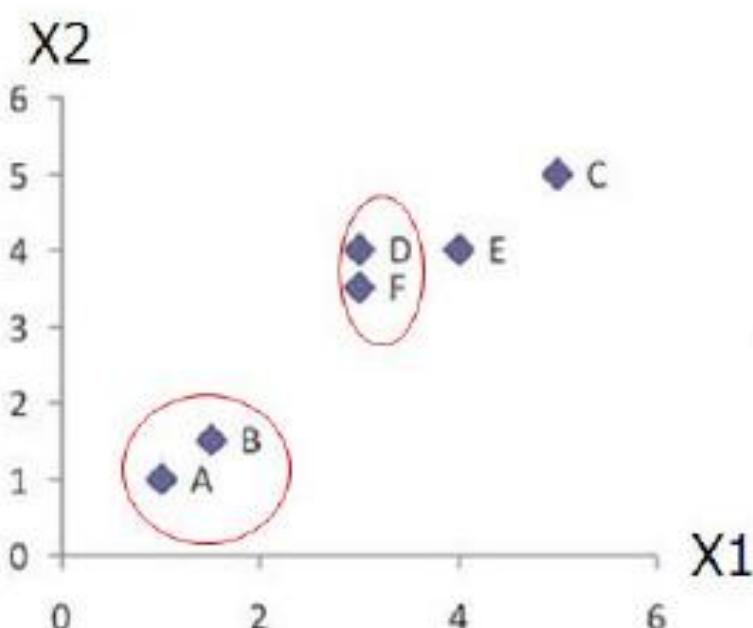
$$d_{B \rightarrow (D,F)} = \min(d_{BD}, d_{BF}) = \min(1.00, 1.12) = 1.00$$

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

- Merge two closest clusters (iteration 2)



Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

- Update distance matrix (iteration 2)

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	3.20	2.50	2.24	0.00	1.00
E	4.24	3.54	1.41	1.00	0.00

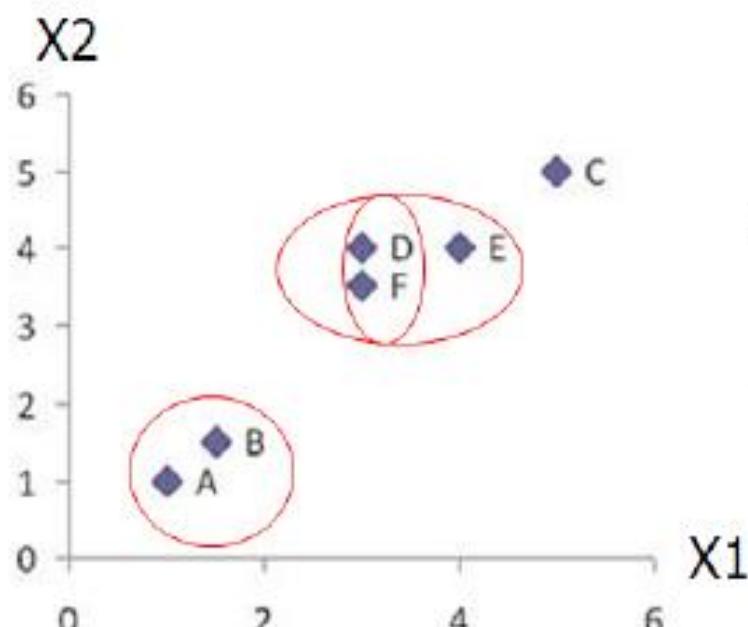
$d_{C \rightarrow (A,B)} = \min(d_{CA}, d_{CB}) = \min(5.66, 4.95) = 4.95$   
 $d_{(D,F) \rightarrow (A,B)} = \min(d_{DA}, d_{DB}, d_{FA}, d_{FB}) = \min(3.61, 2.92, 3.20, 2.50) = 2.50$   
 $d_{E \rightarrow (A,B)} = \min(d_{EA}, d_{EB}) = \min(4.24, 3.54) = 3.54$

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

- Merge two closest clusters/update distance matrix  
(iteration 3)



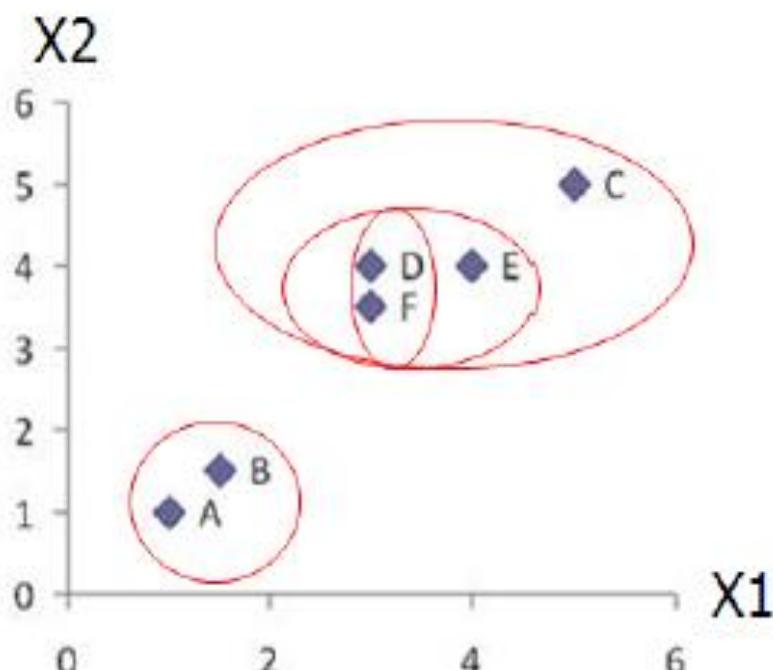
Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D, F)	2.50	2.24	0	1.00
E	3.54	1.41	1.00	0

Min Distance (Single Linkage)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	2.50	1.41	0.00

- Merge two closest clusters/update distance matrix  
(iteration 4)



Min Distance (Single Linkage)

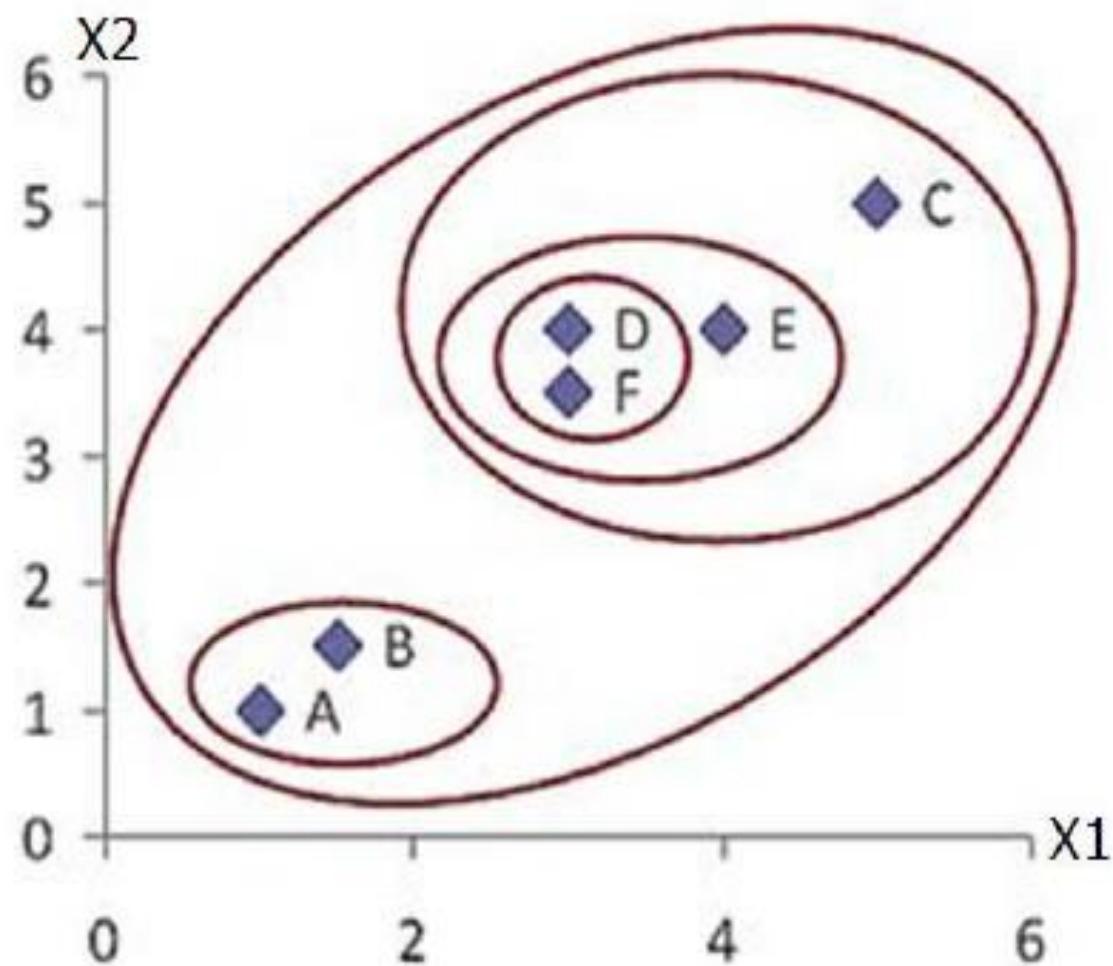
Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	2.50	1.41	0.00

Min Distance (Single Linkage)

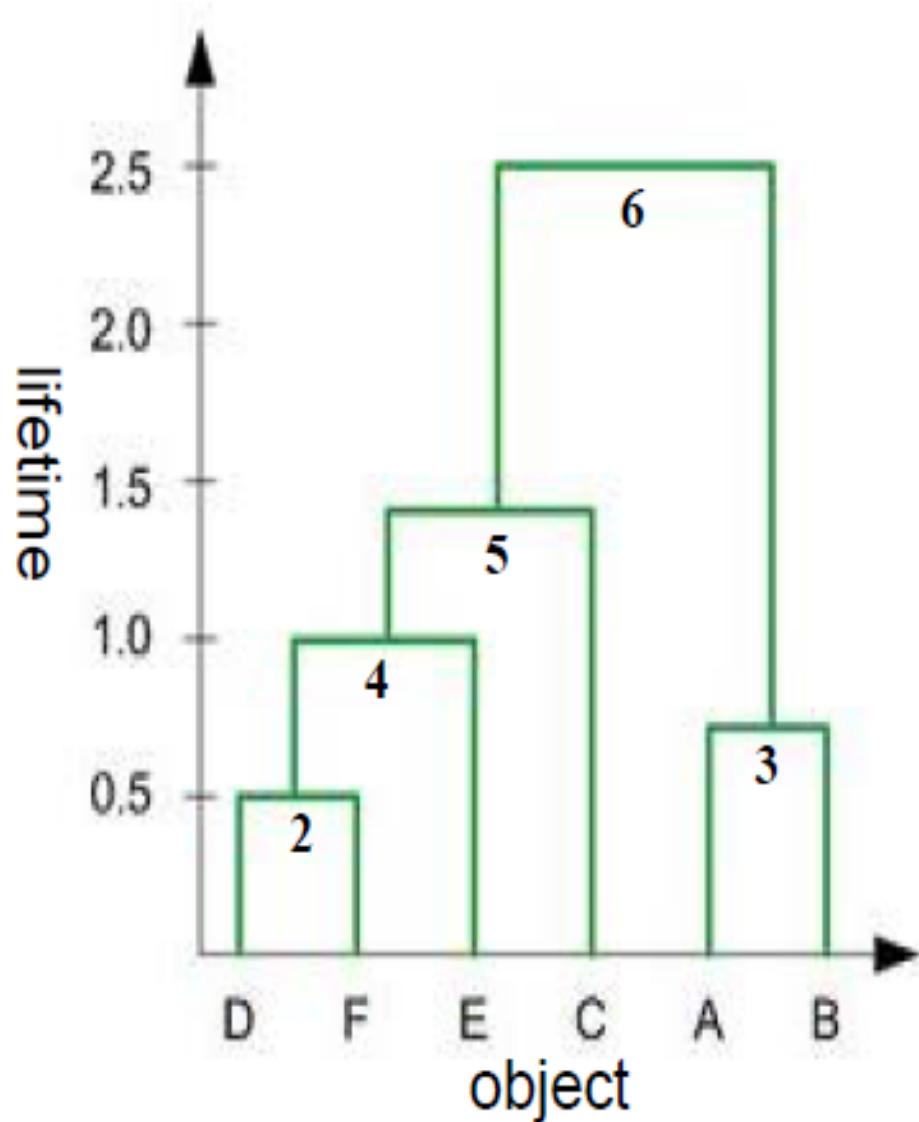
Dist	(A,B)	((D, F), E), C
(A,B)	0.00	2.50
((D, F), E), C	2.50	0.00

- Final result (meeting termination condition)

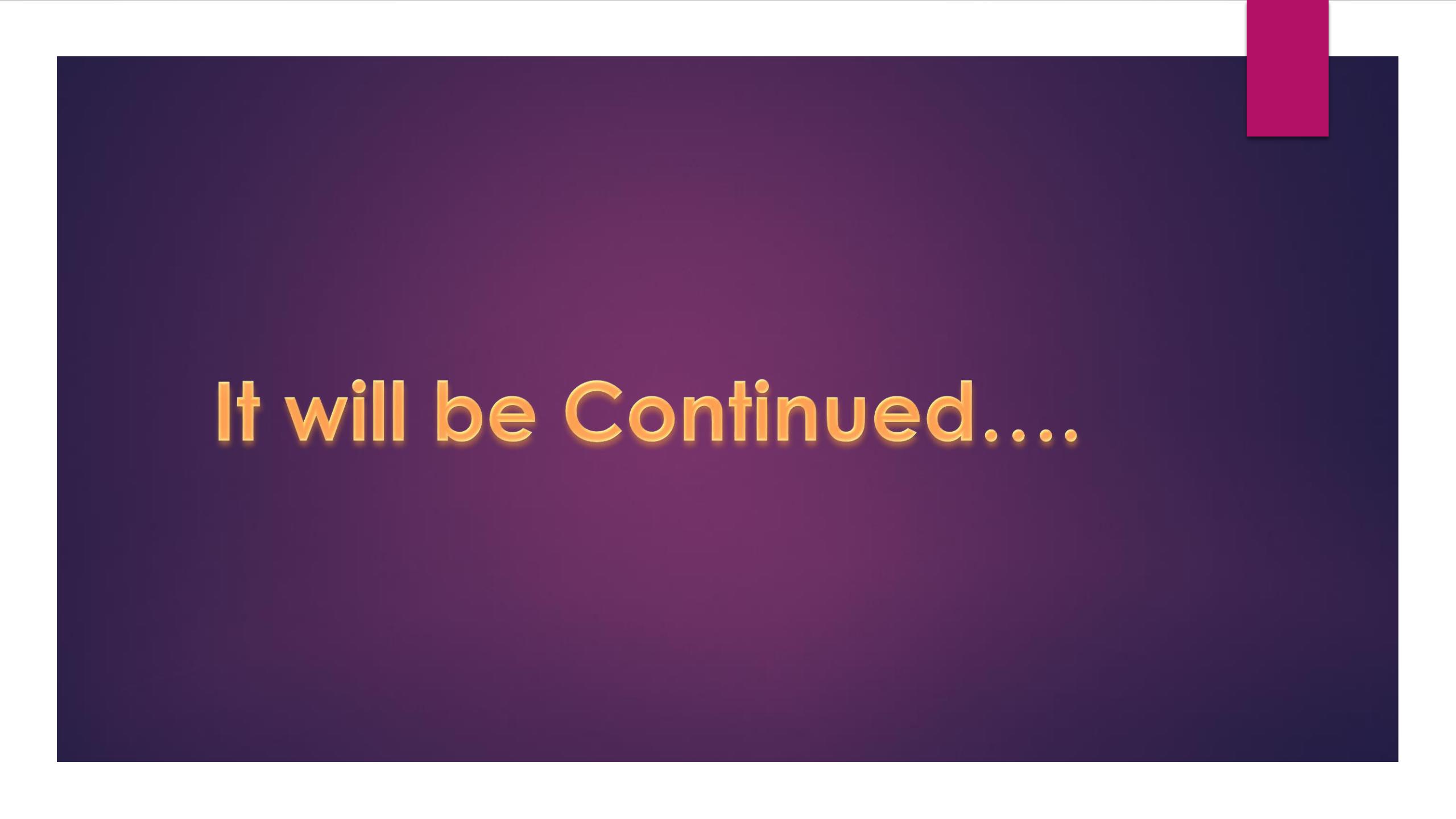
	$x_1$	$x_2$
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



- Dendrogram tree representation



- In the beginning we have 6 clusters: A, B, C, D, E and F
- We merge clusters D and F into cluster (D, F) at distance 0.50
- We merge cluster A and cluster B into (A, B) at distance 0.71
- We merge clusters E and (D, F) into ((D, F), E) at distance 1.00
- We merge clusters ((D, F), E) and C into (((D, F), E), C) at distance 1.41
- We merge clusters (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
- The last cluster contain all the objects, thus conclude the computation



**It will be Continued....**

# Fuzzy Logic

BY

DR. ANUPAM GHOSH

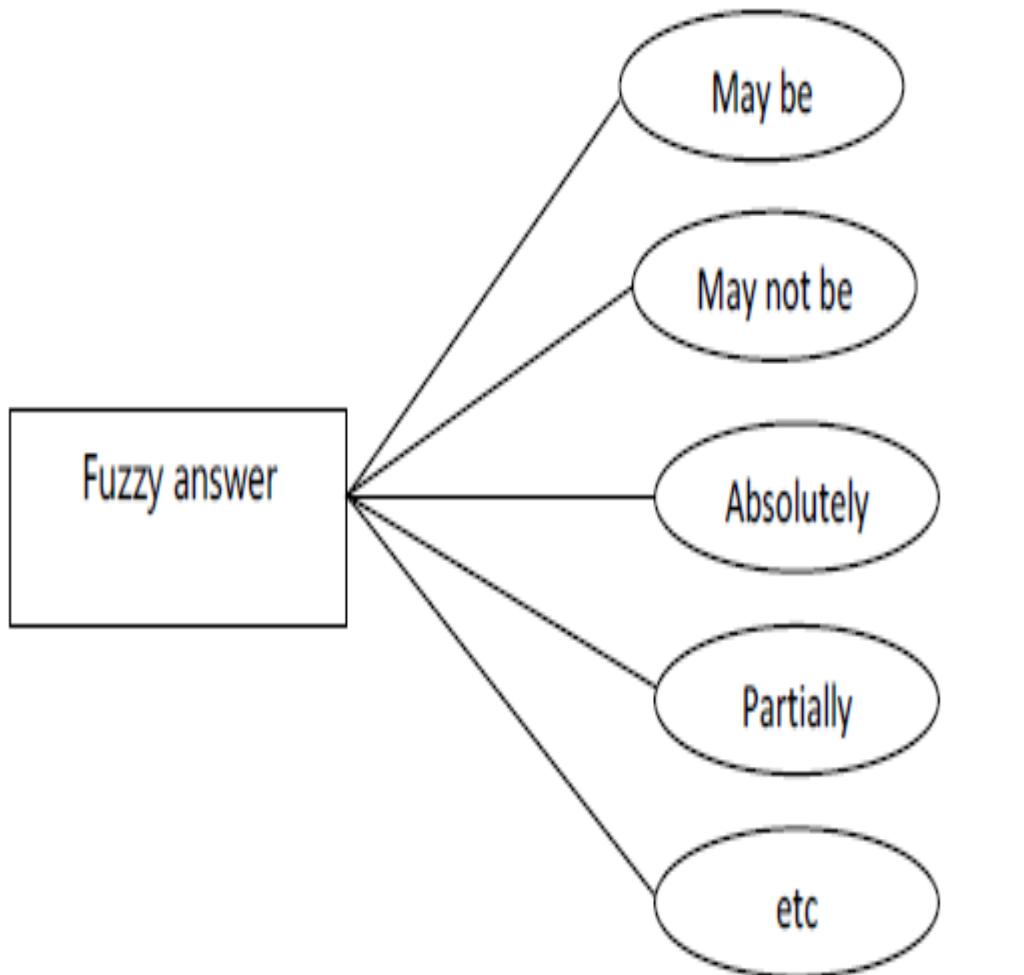
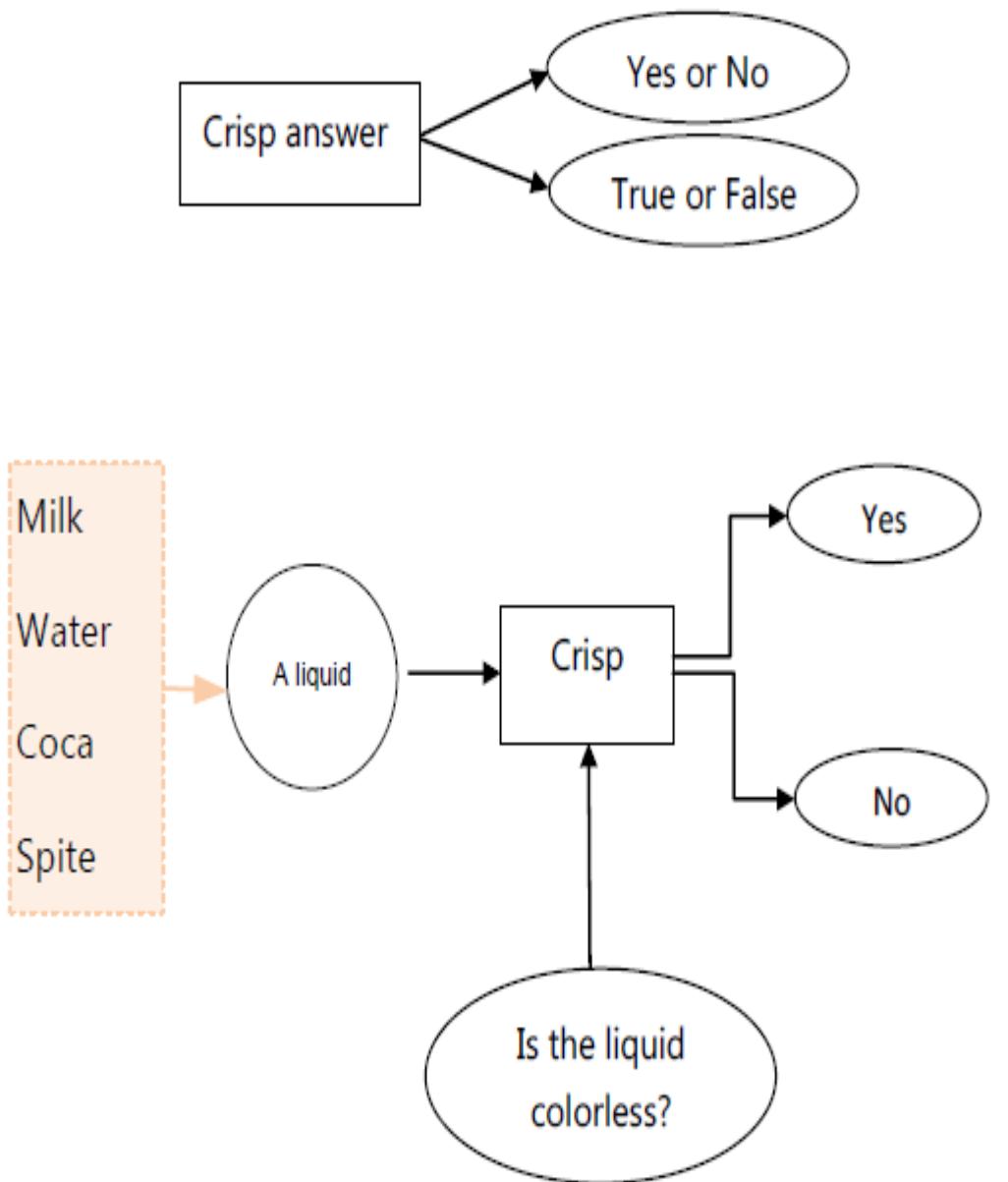
DATE: 09.09.2023

# History of fuzzy logic

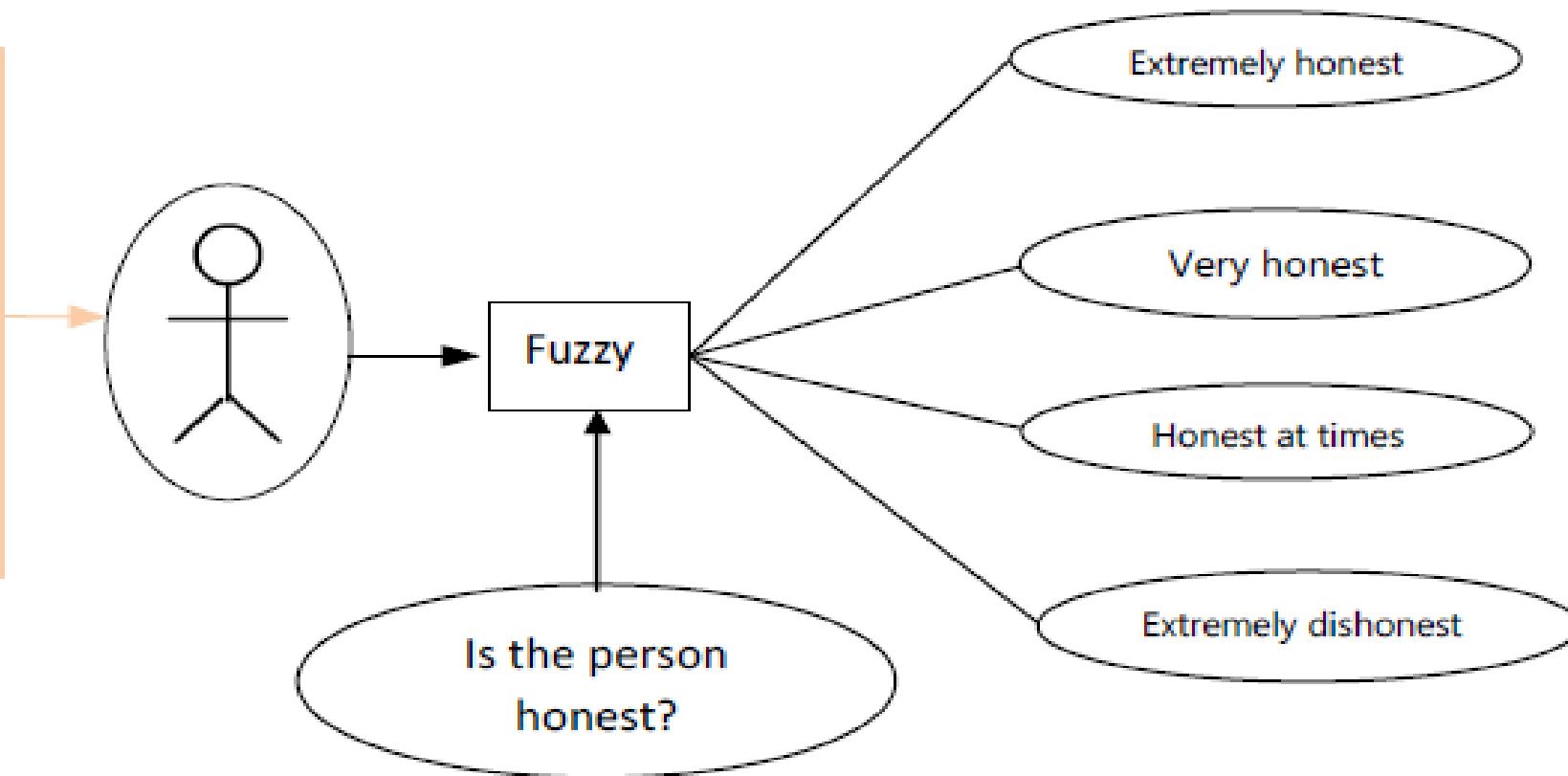
- The opposite word (antonym) of fuzzy is crisp. **Fuzzy** means un-clear or ambiguous, and **crisp** means clear, clean, and sharp.
- Fuzzy logic was proposed by Zadeh in 1965, and was applied in steam engine control by Mamdani in 1974.
- Fuzzy logic was made practically useful in Japan in the 1990s.



- Uncertainty
- Vagueness
- Imprecision
- Inexactness



- Ankit
- Rajesh
- Santosh
- Kabita
- Salmon



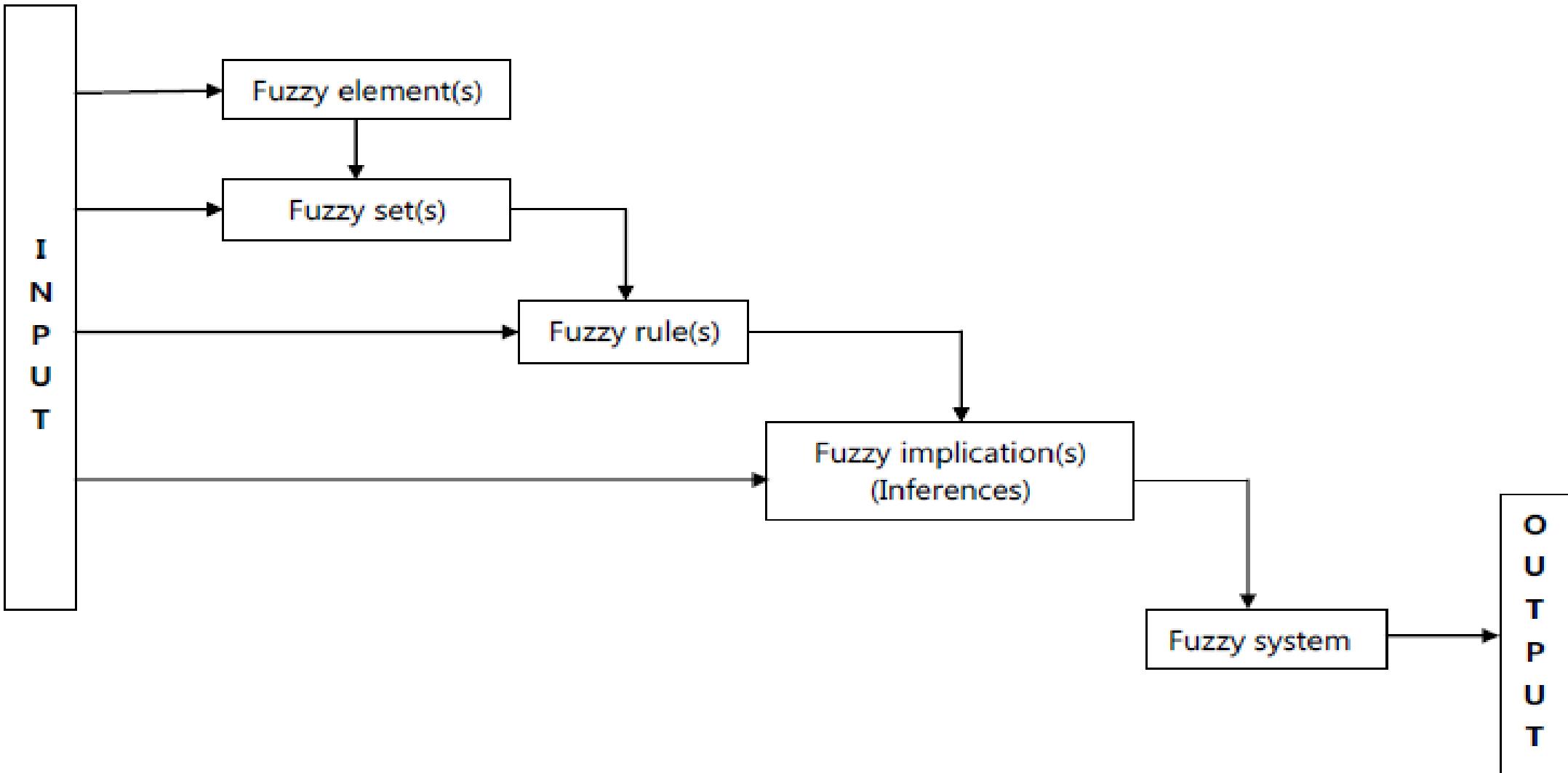
Score

99

75

55

35



# Conventional set vs. fuzzy set

- $X$ : universe of discourse. Set  $A$  is defined by

$$A = \{x \mid \mu(x) = T \wedge x \in X\}$$

where  $\mu(x)$  is a **membership function**.

- For conventional set, the range of  $\mu(x)$  is {T, F}
- For fuzzy set, the range of  $\mu(x)$  is [0,1]. So, the above definition cannot be used for fuzzy set. We cannot say clearly if  $x$  is in  $A$  or not when  $0 < \mu(x) < 1$ .

Examples of fuzzy set

- ✓ young people, old people, kind people
- ✓ temperature is hot, just good, a little bit cold

# Crisp vs Fuzzy

Crisp Set	Fuzzy Set
1. $S = \{ s \mid s \in X \}$	1. $F = (s, \mu) \mid s \in X$ and $\mu(s)$ is the degree of $s$ .
2. It is a collection of elements.	2. It is collection of ordered pairs.
3. Inclusion of an element $s \in X$ into $S$ is crisp, that is, has strict boundary <b>yes or no</b> .	3. Inclusion of an element $s \in X$ into $F$ is fuzzy, that is, if present, then with a degree of <b>membership</b> .

## **Definition 1: Membership function (and Fuzzy set)**

If  $X$  is a universe of discourse and  $x \in X$ , then a fuzzy set  $A$  in  $X$  is defined as a set of ordered pairs, that is

$A = \{(x, \mu_A(x)) | x \in X\}$  where  $\mu_A(x)$  is called the **membership function** for the fuzzy set  $A$ .

### **Note:**

$\mu_A(x)$  map each element of  $X$  onto a membership grade (or membership value) between 0 and 1 (both inclusive).

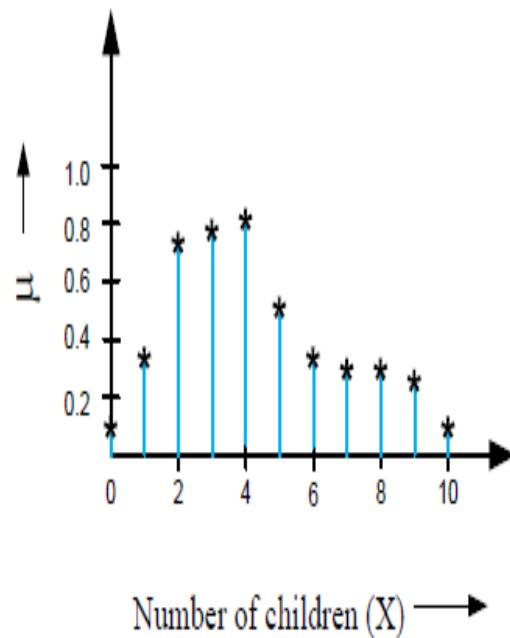
### **Example:**

$X$  = All cities in India

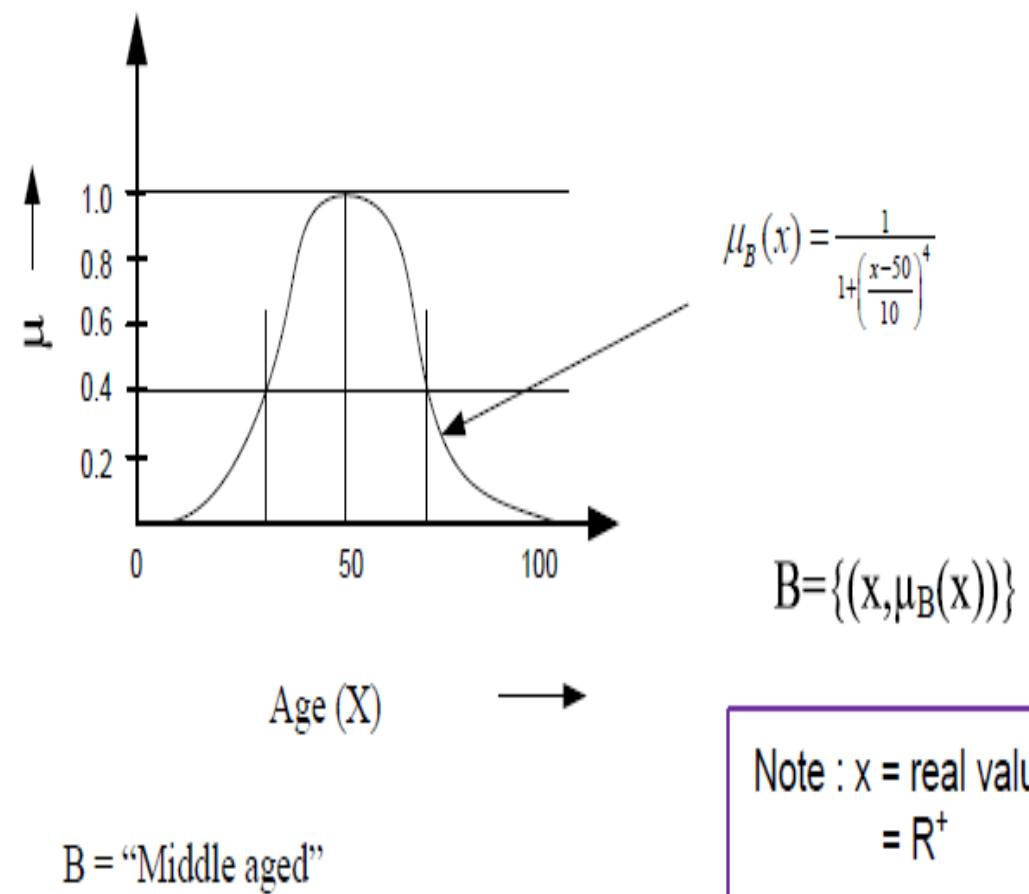
$A$  = City of comfort

$A = \{(New\ Delhi, 0.7), (Bangalore, 0.9), (Chennai, 0.8), (Hyderabad, 0.6), (Kolkata, 0.3), (Kharagpur, 0)\}$

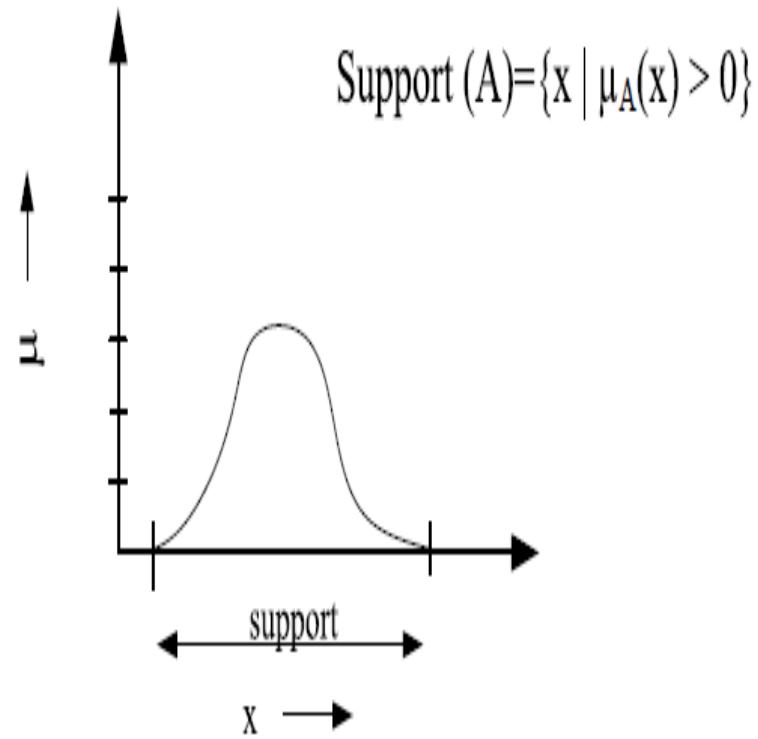
Either elements or their membership values (or both) also may be of discrete values.



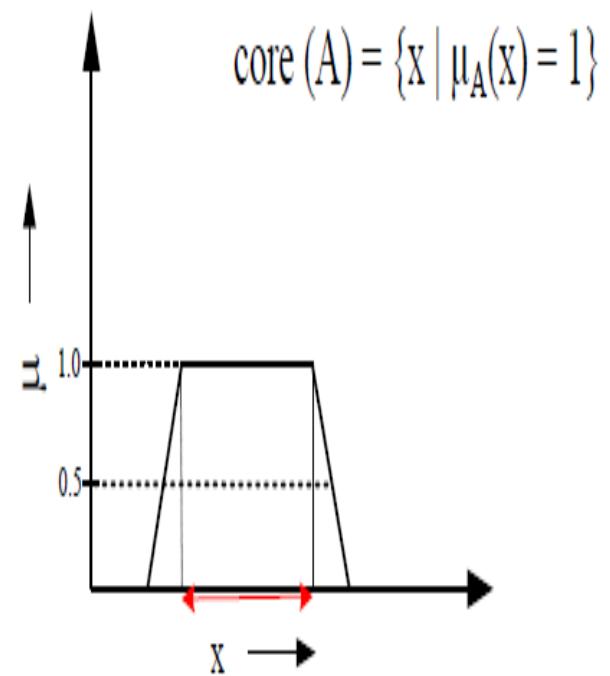
A = "Happy family"



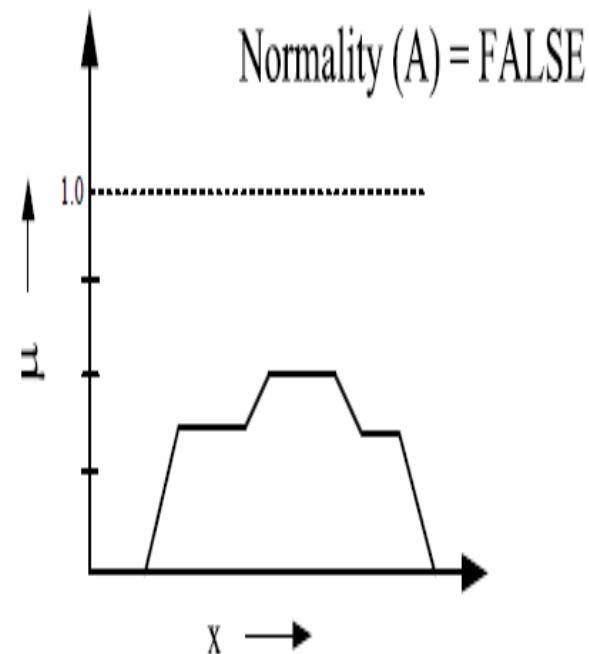
**Support:** The support of a fuzzy set  $A$  is the set of all points  $x \in X$  such that  $\mu_A(x) > 0$



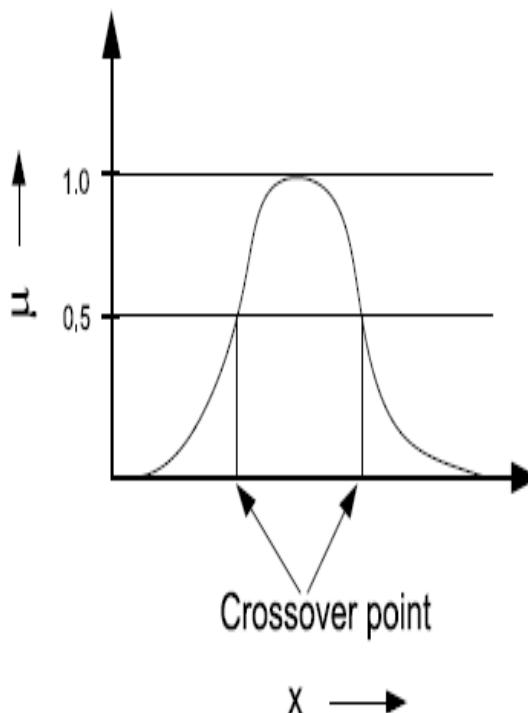
**Core:** The core of a fuzzy set  $A$  is the set of all points  $x$  in  $X$  such that  $\mu_A(x) = 1$



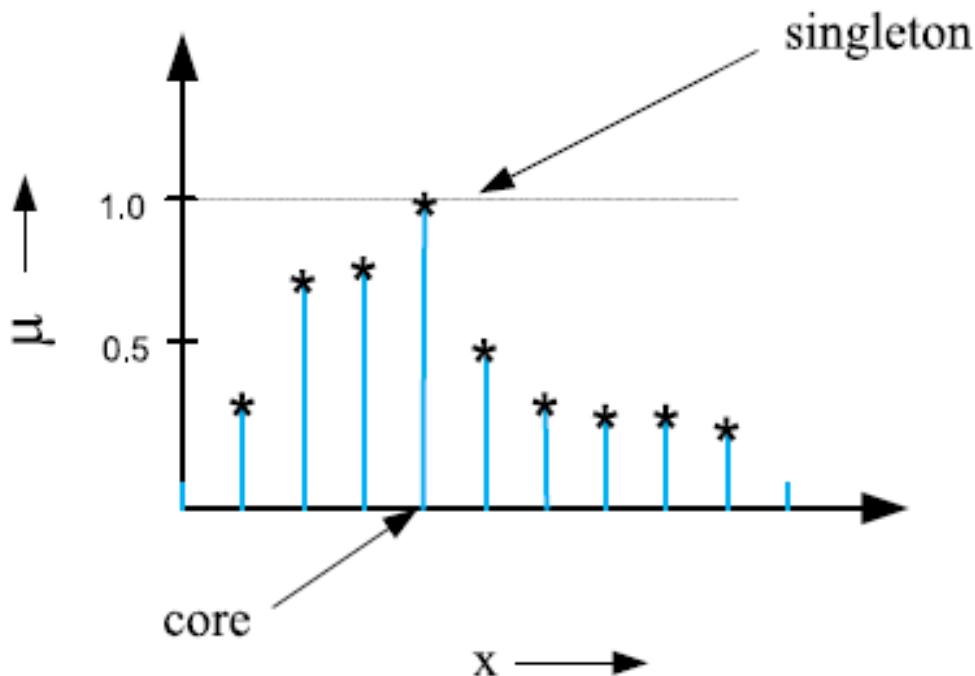
**Normality** : A fuzzy set  $A$  is normal if its core is non-empty. In other words, we can always find a point  $x \in X$  such that  $\mu_A(x) = 1$ .



**Crossover point** : A crossover point of a fuzzy set  $A$  is a point  $x \in X$  at which  $\mu_A(x) = 0.5$ . That is  
 $\text{Crossover}(A) = \{x | \mu_A(x) = 0.5\}$ .



**Fuzzy Singleton** : A fuzzy set whose support is a single point in  $X$  with  $\mu_A(x) = 1$  is called a fuzzy singleton. That is  $|A| = |\{ x \mid \mu_A(x) = 1\}| = 1$ . Following fuzzy set is not a fuzzy singleton.



# Description of fuzzy set

- Membership function of a fuzzy set A:  $\mu_A: X \rightarrow [0, 1]$
- If the universe of discourse  $X = \{x_1, x_2, \dots, x_N\}$ , A is described as follows:

$$\begin{aligned}A &= \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \dots + \mu_A(x_N)/x_N \\&= \sum \mu_A(x_i)/x_i\end{aligned}$$

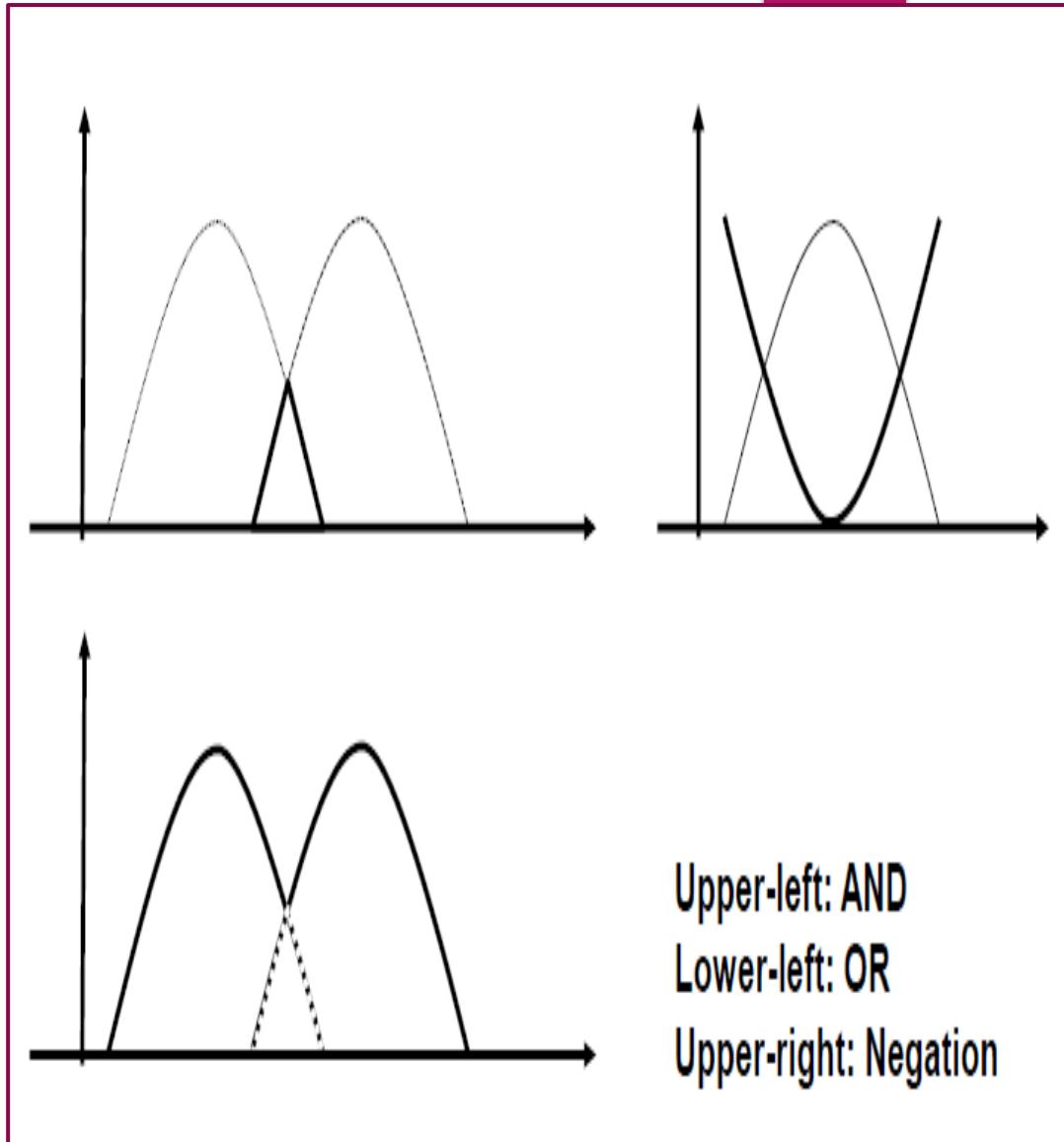
where / is a separator, and + is the logic OR.

- If X is a continuous space, integral is used instead of the summation.



# Operations of fuzzy sets

Meaning	Set notation	Logic notation	Membership function
Equivalence	$A = B$	$\mu_A(x) \Leftrightarrow \mu_B(x)$	$\mu_A(x) = \mu_B(x)$
Implication	$A \subseteq B$	$\mu_A(x) \Rightarrow \mu_B(x)$	$\mu_A(x) \leq \mu_B(x)$
Complement (negation)	$\bar{A}$	$\neg \mu_A(x)$	$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$
Union, OR (disjunction)	$A \cup B$	$\mu_A(x) \vee \mu_B(x)$	$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$
Intersection, AND (conjunction)	$A \cap B$	$\mu_A(x) \wedge \mu_B(x)$	$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$



The universe of discourse:

$$X = \{\text{Masahiro, Tsuyoshi, Takuya, Saburo, Masami}\}$$

Fuzzy sets A=“young persons” and B=“Tall persons” are defined by

- $A = 0.4/\text{Masahiro} + 0.6/\text{Tsuyoshi} + 0.8/\text{Takuya} + 1.0/\text{Saburo} + 0.9/\text{Masami}$
- $B = 0.3/\text{Masahiro} + 0.5/\text{Tsuyoshi} + 0.9/\text{Takuya} + 0.6/\text{Saburo} + 0.9/\text{Masami}$

The OR and AND of A and B are as follows:

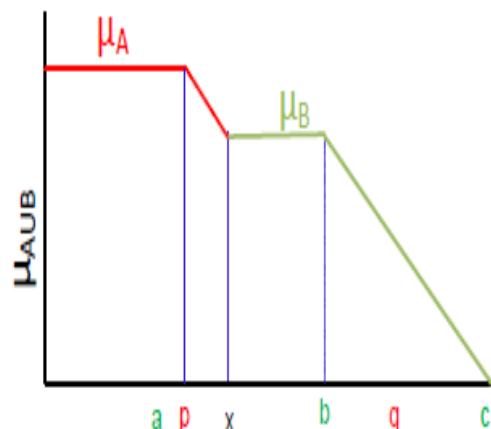
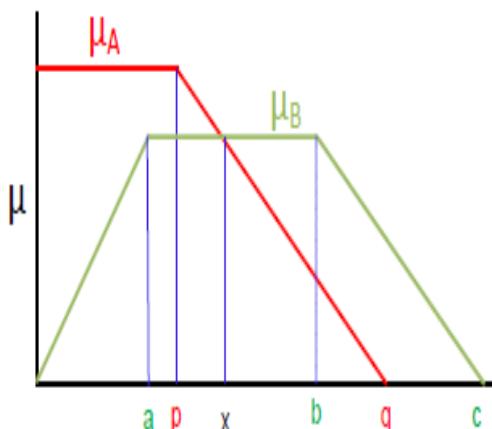
- $A \cup B = 0.4/\text{Masahiro} + 0.6/\text{Tsuyoshi} + 0.9/\text{Takuya} + 1.0/\text{Saburo} + 0.9/\text{Masami}$
- $A \cap B = 0.3/\text{Masahiro} + 0.5/\text{Tsuyoshi} + 0.8/\text{Takuya} + 0.6/\text{Saburo} + 0.9/\text{Masami}$

## Union ( $A \cup B$ ):

$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$$

Example:

$$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\} \text{ and}$$
$$B = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\};$$
$$C = A \cup B = \{(x_1, 0.5), (x_2, 0.3), (x_3, 0.5)\}$$

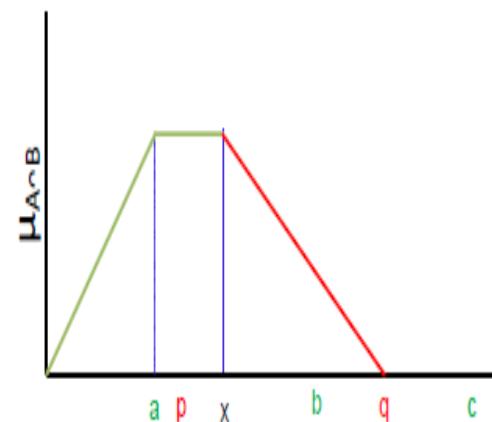
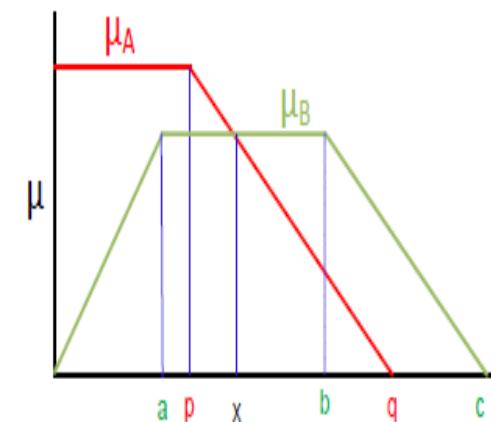


## Intersection ( $A \cap B$ ):

$$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$$

Example:

$$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\} \text{ and}$$
$$B = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5)\};$$
$$C = A \cap B = \{(x_1, 0.2), (x_2, 0.1), (x_3, 0.4)\}$$



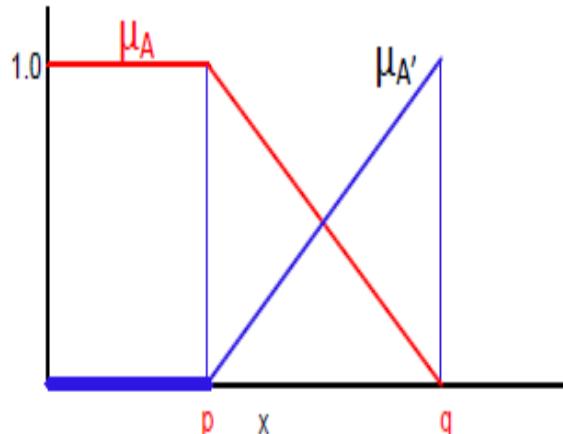
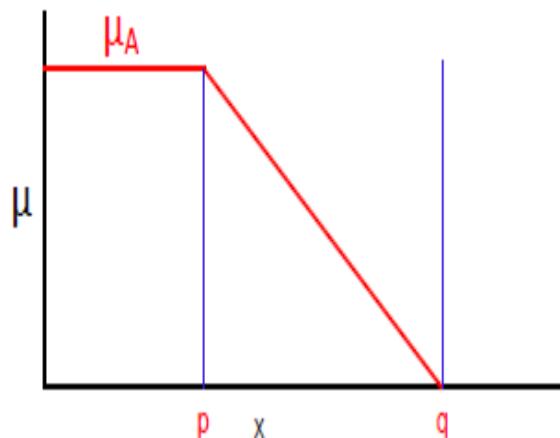
**Complement ( $A^C$ ):**

$$\mu_{A^C}(x) = 1 - \mu_A(x)$$

**Example:**

$$A = \{(x_1, 0.5), (x_2, 0.1), (x_3, 0.4)\}$$

$$C = A^C = \{(x_1, 0.5), (x_2, 0.9), (x_3, 0.6)\}$$



**Algebraic product or Vector product ( $A \bullet B$ ):**

$$\mu_{A \bullet B}(x) = \mu_A(x) \bullet \mu_B(x)$$

**Scalar product ( $\alpha \times A$ ):**

$$\mu_{\alpha A}(x) = \alpha \cdot \mu_A(x)$$

**Sum** ( $A + B$ ):

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$$

**Difference** ( $A - B = A \cap B^C$ ):

$$\mu_{A-B}(x) = \mu_{A \cap B^C}(x)$$

**Disjunctive sum:**  $A \oplus B = (A^C \cap B) \cup (A \cap B^C)$

**Bounded Sum:**  $| A(x) \oplus B(x) |$

$$\mu_{|A(x) \oplus B(x)|} = \min\{1, \mu_A(x) + \mu_B(x)\}$$

**Bounded Difference:**  $| A(x) \ominus B(x) |$

$$\mu_{|A(x) \ominus B(x)|} = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$$

**Equality** ( $A = B$ ):

$$\mu_A(x) = \mu_B(x)$$

**Power of a fuzzy set**  $A^\alpha$ :

$$\mu_{A^\alpha}(x) = \{\mu_A(x)\}^\alpha$$

- If  $\alpha < 1$ , then it is called *dilation*
- If  $\alpha > 1$ , then it is called *concentration*

## Basic fuzzy set operations: Cartesian product

**Cartesian Product ( $A \times B$ ):**

$$\mu_{A \times B}(x, y) = \min\{\mu_A(x), \mu_B(y)\}$$

**Example 3:**

$$A(x) = \{(x_1, 0.2), (x_2, 0.3), (x_3, 0.5), (x_4, 0.6)\}$$

$$B(y) = \{(y_1, 0.8), (y_2, 0.6), (y_3, 0.3)\}$$

$$A \times B = \min\{\mu_A(x), \mu_B(y)\} =$$

	$y_1$	$y_2$	$y_3$
$x_1$	0.2	0.2	0.2
$x_2$	0.3	0.3	0.3
$x_3$	0.5	0.5	0.3
$x_4$	0.6	0.6	0.3

## Properties of fuzzy sets

Commutativity :

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

Associativity :

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

Distributivity :

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Idempotence :

$$A \cup A = A$$

$$A \cap A = \emptyset$$

$$A \cup \emptyset = A$$

$$A \cap \emptyset = \emptyset$$

Transitivity :

If  $A \subseteq B, B \subseteq C$  then  $A \subseteq C$

Involution :

$$(A^c)^c = A$$

De Morgan's law :

$$(A \cap B)^c = A^c \cup B^c$$

$$(A \cup B)^c = A^c \cap B^c$$

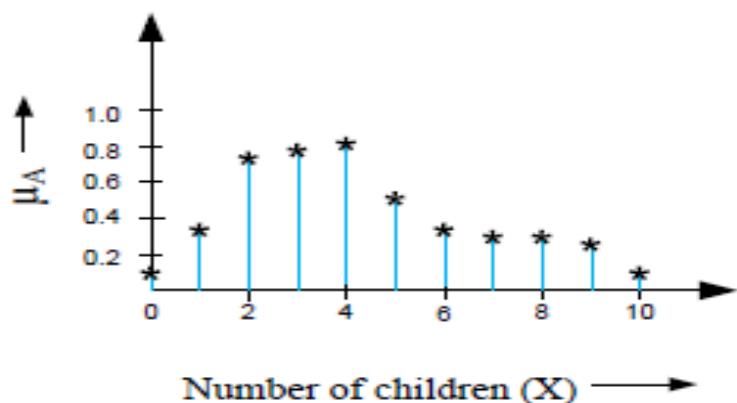
# Membership Functions

A fuzzy set is completely characterized by its membership function (sometimes abbreviated as *MF* and denoted as  $\mu$ ). So, it would be important to learn how a membership function can be expressed (mathematically or otherwise).

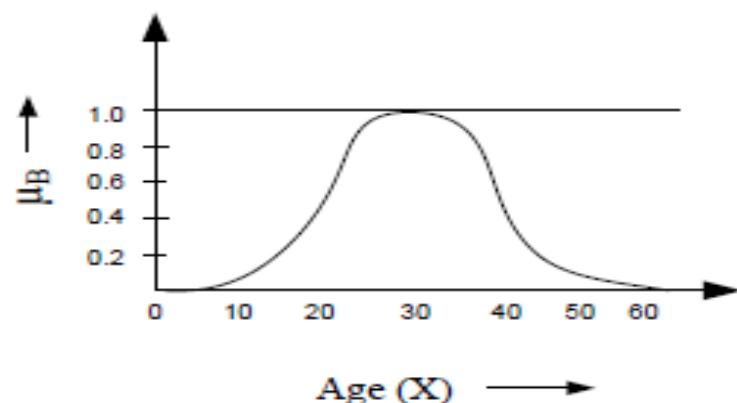
**Note:** A membership function can be on

- (a) a discrete universe of discourse and
- (b) a continuous universe of discourse.

**Example:**



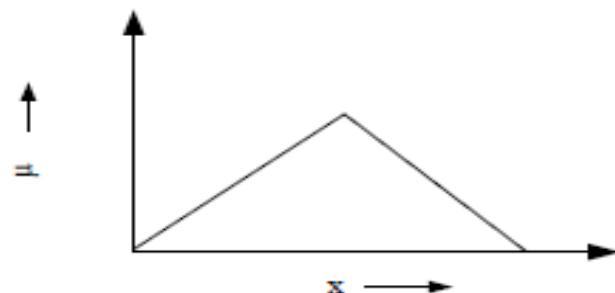
A = Fuzzy set of “Happy family”



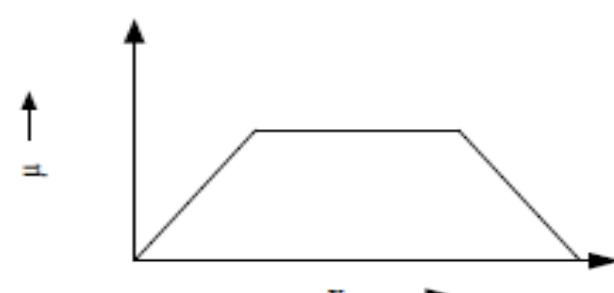
B = “Young age”

So, membership function on a discrete universe of course is trivial. However, a membership function on a continuous universe of discourse needs a special attention.

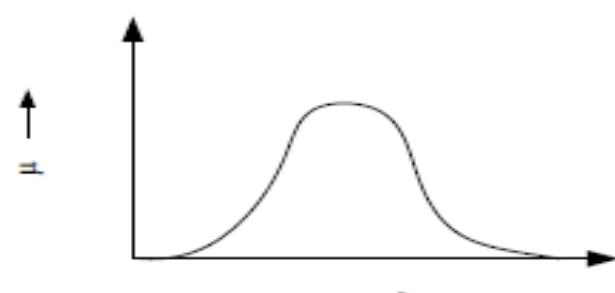
Following figures shows a typical examples of membership functions.



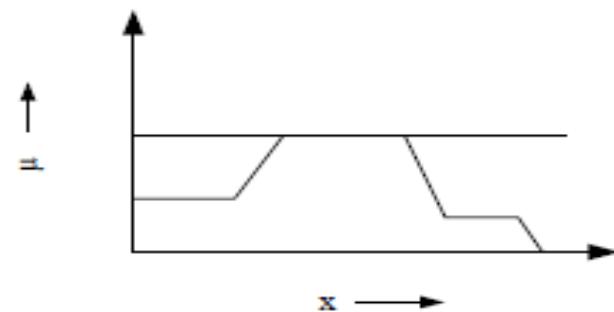
< triangular >



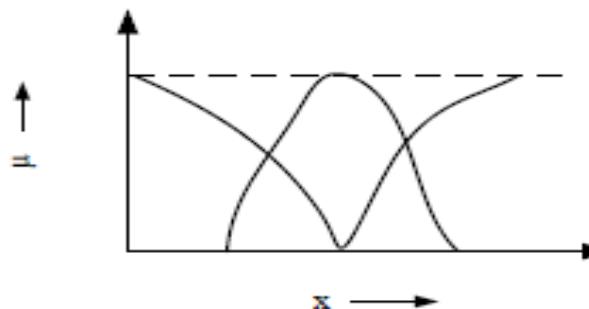
< trapezoidal >



< curve >



< non-uniform >



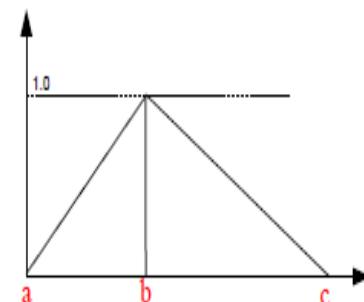
< non-uniform >

## Fuzzy MFs : Formulation and parameterization

In the following, we try to parameterize the different MFs on a continuous universe of discourse.

**Triangular MFs :** A triangular MF is specified by three parameters  $\{a, b, c\}$  and can be formulated as follows.

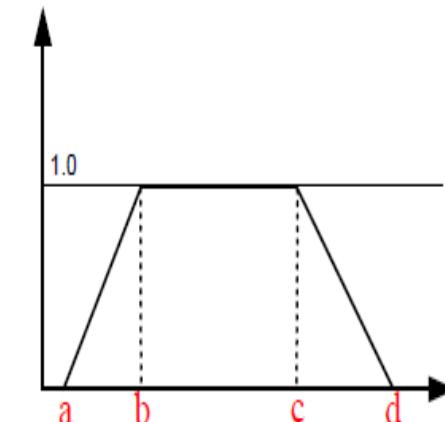
$$\text{triangle}(x; a, b, c) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ \frac{c-x}{c-b} & \text{if } b \leq x \leq c \\ 0 & \text{if } c \leq x \end{cases}$$



## Fuzzy MFs: Trapezoidal

A trapezoidal MF is specified by four parameters  $\{a, b, c, d\}$  and can be defined as follows:

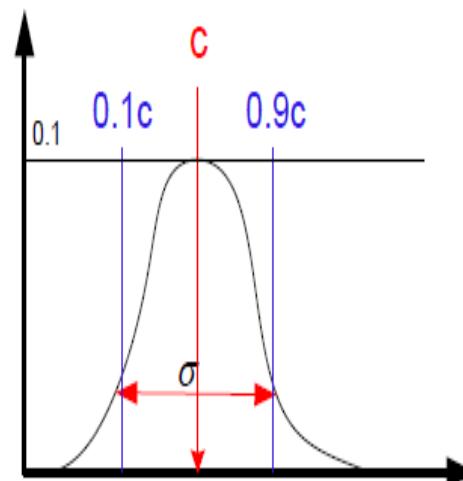
$$\text{trapeziod}(x; a, b, c, d) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{if } c \leq x \leq d \\ 0 & \text{if } d \leq x \end{cases}$$



## Fuzzy MFs: Gaussian

A Gaussian MF is specified by two parameters  $\{c, \sigma\}$  and can be defined as below:

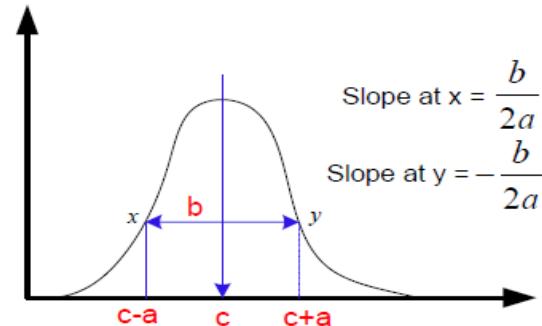
$$\text{gaussian}(x; c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2}.$$



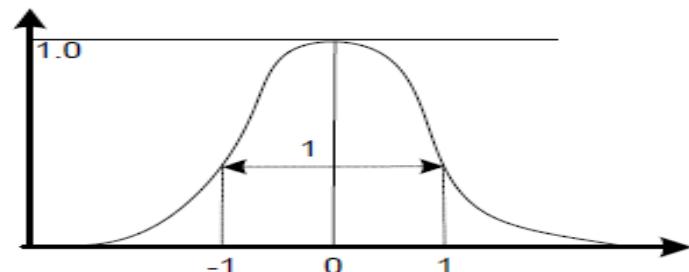
## Fuzzy MFs: Generalized bell

It is also called [Cauchy MF](#). A generalized bell MF is specified by three parameters  $\{a, b, c\}$  and is defined as:

$$\text{bell}(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}}$$



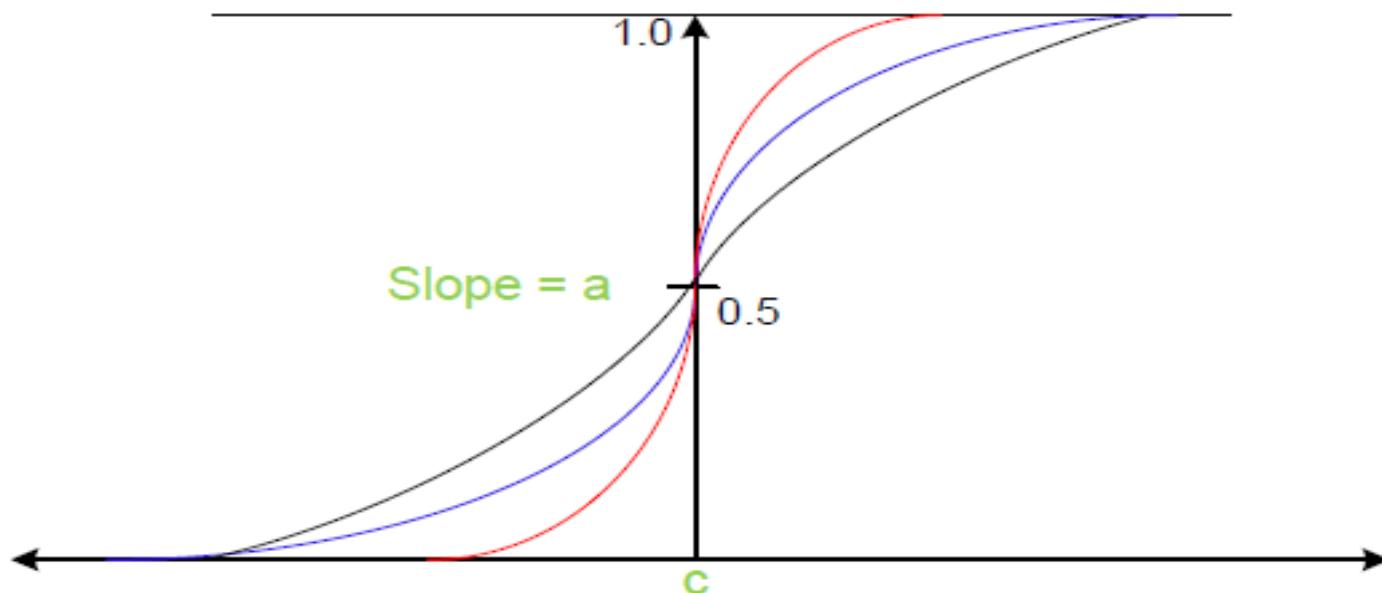
Example:  $\mu(x) = \frac{1}{1+x^2}$  ;  
 $a = b = 1$  and  $c = 0$ ;



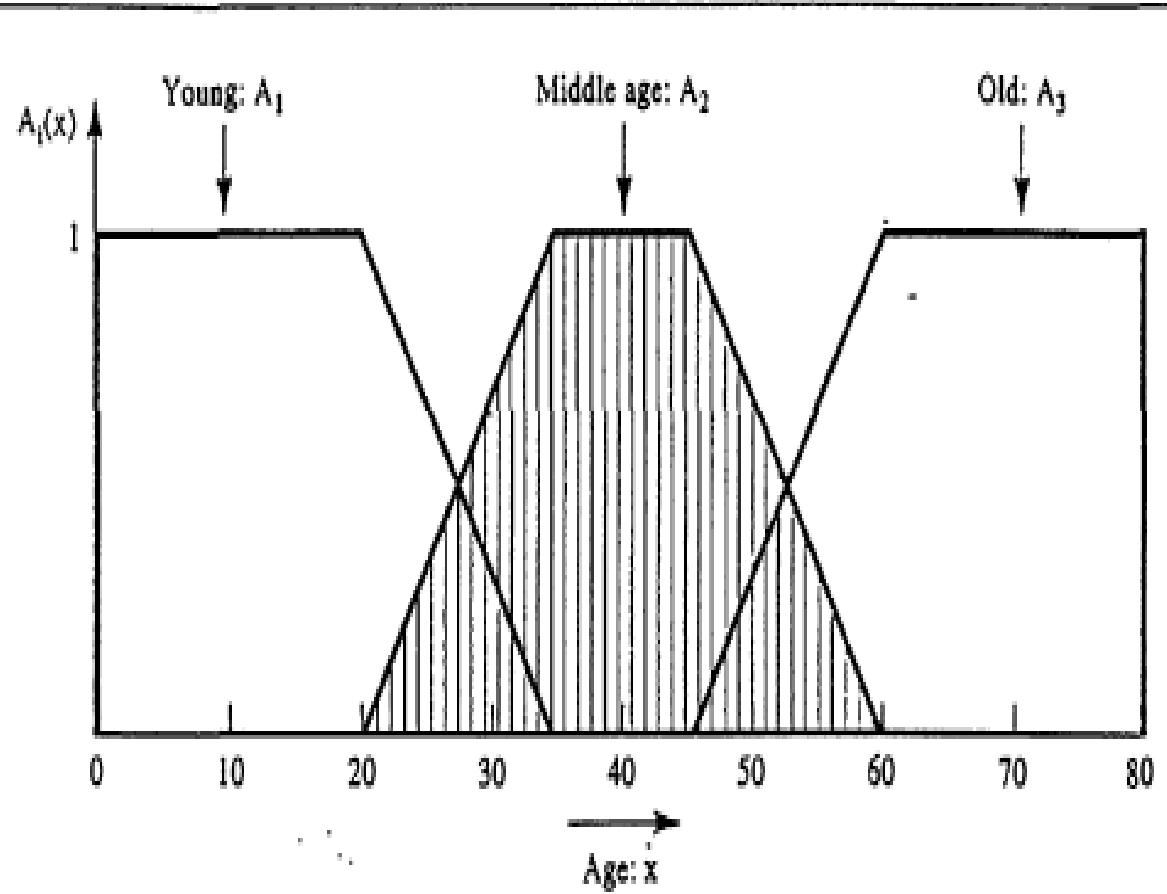
## Fuzzy MFs: Sigmoidal MFs

Parameters:  $\{a, c\}$  ; where  $c$  = crossover point and  $a$  = slope at  $c$ ;

$$\text{Sigmoid}(x;a,c) = \frac{1}{1+e^{-[\frac{a}{x-c}]}}$$



Example: 0 to 35 young aged; 20 to 60 middle aged; and 45 to 80 old aged persons. Draw the membership curve and define the membership functions of "Age"

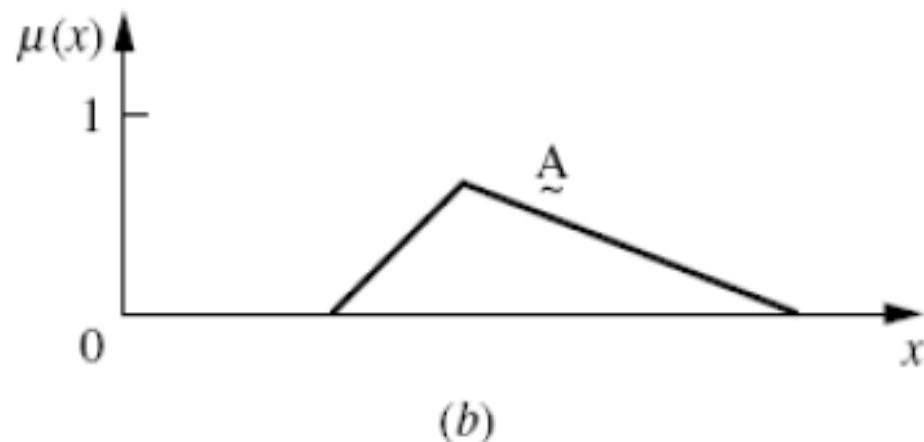
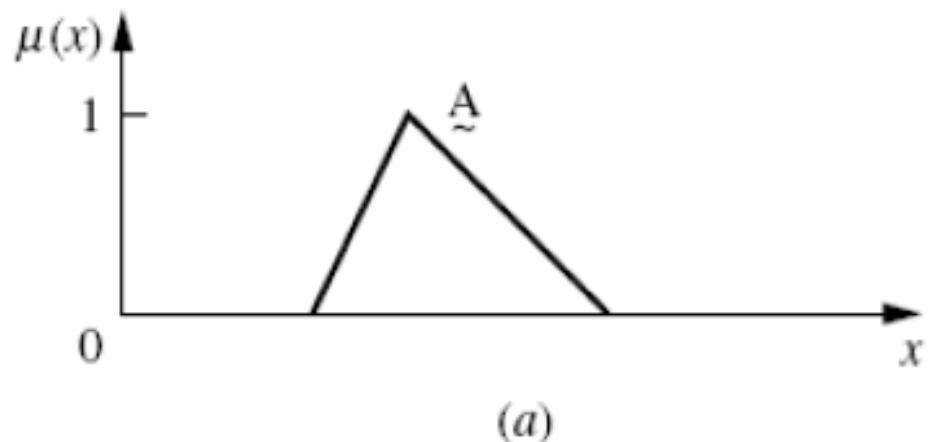


$$A_1(x) = \begin{cases} 1 & \text{when } x \leq 20 \\ (35-x)/15 & \text{when } 20 < x < 35 \\ 0 & \text{when } x \geq 35 \end{cases}$$

$$A_2(x) = \begin{cases} 0 & \text{when either } x \leq 20 \text{ or } x \geq 60 \\ (x-20)/15 & \text{when } 20 < x < 35 \\ (60-x)/15 & \text{when } 35 < x < 60 \\ 1 & \text{when } 35 \leq x \leq 45 \end{cases}$$

$$A_3(x) = \begin{cases} 0 & \text{when } x \leq 45 \\ (x-45)/15 & \text{when } 45 < x < 60 \\ 1 & \text{when } x \geq 60 \end{cases}$$

- A *normal* fuzzy set is one whose membership function has at least one element  $x$  in the universe whose membership value is unity.



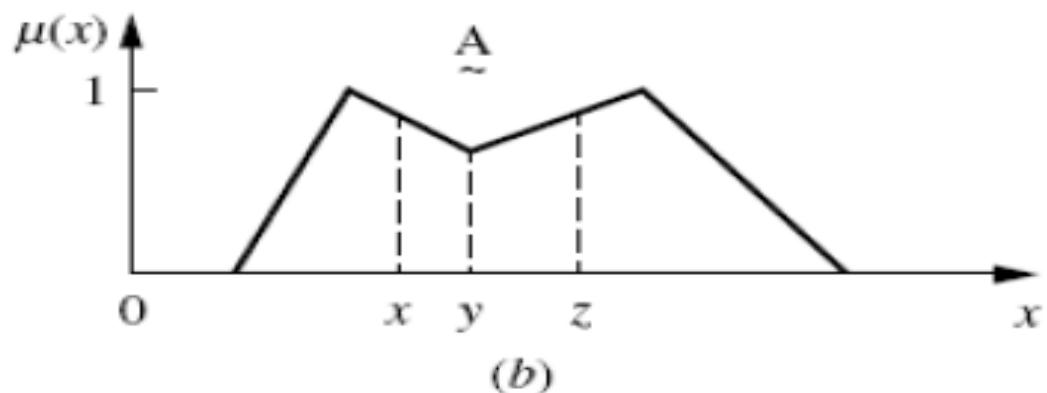
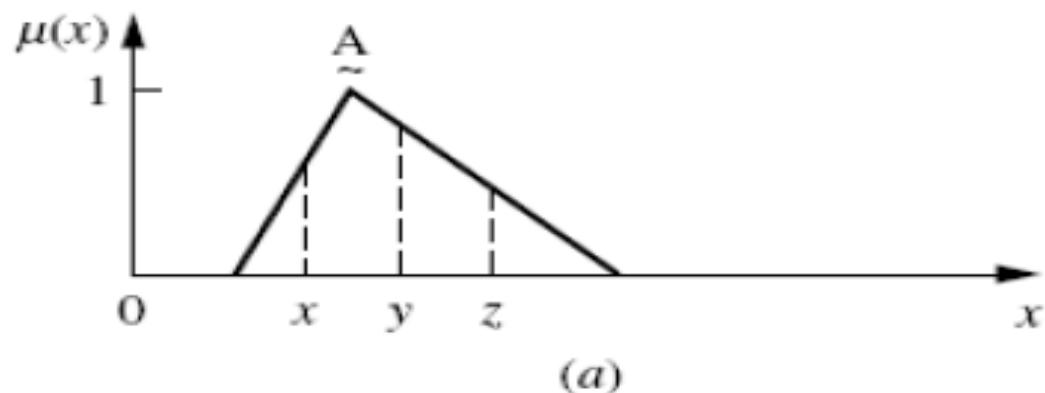
# Convex fuzzy set

- A *convex* fuzzy set is described by a membership function whose membership values are strictly monotonically increasing, or strictly monotonically decreasing, or strictly monotonically increasing then strictly monotonically decreasing with increasing values for elements in the universe.

(Said another way, if, for any elements  $x$ ,  $y$ , and  $z$  in a fuzzy set  $\underline{A}$ , the relation  $x < y < z$  implies that

$$\mu_{\underline{A}}(y) \geq \min[\mu_{\underline{A}}(x), \mu_{\underline{A}}(z)]$$

then  $\underline{A}$  is said to be a convex fuzzy set.)



**FIGURE**

Convex, normal fuzzy set (a) and nonconvex, normal fuzzy set (b).

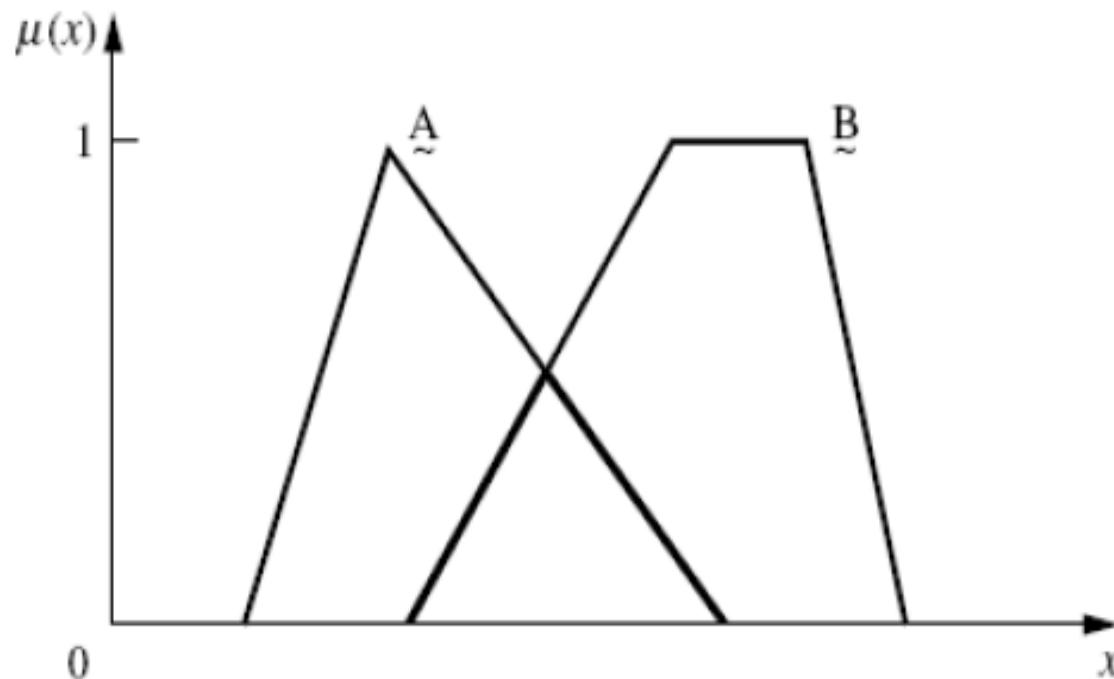
this definition of convexity is *different* from some definitions of the same term [in mathematics](#). In some areas of mathematics, convexity of shape has to do with whether a straight line through any part of the shape goes outside the boundaries of that shape. This definition of convexity is *not* used here.

- The *crossover points* of a membership function are defined as the elements in the universe for which a particular fuzzy set  $\underline{A}$  has values equal to 0.5, i.e., for which  $\mu_{\underline{A}}(x) = 0.5$ .
- The *height* of a fuzzy set  $A$  is the maximum value of the membership function, i.e.,

$$\text{hgt}(\underline{A}) = \max\{\mu_{\underline{A}}(x)\}$$

- If the  $\text{hgt}(\underline{A}) < 1$ , the fuzzy set is said to be *subnormal*.
- If  $\underline{A}$  is a convex single-point normal fuzzy set defined on the real line, then  $\underline{A}$  is often termed a *fuzzy number*.

A special property of two convex fuzzy sets, say  $\tilde{A}$  and  $\tilde{B}$ , is that the intersection of these two convex fuzzy sets is also a convex fuzzy set, as shown in Fig. 4.4. That is, for  $\tilde{A}$  and  $\tilde{B}$ , which are both convex,  $\tilde{A} \cap \tilde{B}$  is also convex.



### FIGURE

The intersection of two convex fuzzy sets produces a convex fuzzy set.

# Defuzzyfication

# Defuzzification Methods

- ▶ Fuzzy rule based systems evaluate linguistic if-then rules using fuzzification, inference and composition procedures. They produce fuzzy results which usually have to be converted into crisp output.
- ▶ To transform the fuzzy results into crisp, defuzzification is performed.
- ▶ Defuzzification is the process of converting a fuzzified output into a single crisp value with respect to a fuzzy set. The defuzzified value in FLC (Fuzzy Logic Controller) represents the action to be taken in controlling the process.
- ▶ **Different Defuzzification Methods**
  - Center of Sums Method (COS)
  - Center of gravity (COG) / Centroid of Area (COA) Method
  - Center of Area / Bisector of Area Method (BOA)
  - Weighted Average Method
  - Maxima Methods
    - First of Maxima Method (FOM)
    - Last of Maxima Method (LOM)
    - Mean of Maxima Method (MOM)

# Center of gravity (COG) / Centroid of Area (COA) Method

This method provides a crisp value based on the center of gravity of the fuzzy set. The total area of the membership function distribution used to represent the combined control action is divided into a number of sub-areas. The area and the center of gravity or centroid of each sub-area is calculated and then the summation of all these sub-areas is taken to find the defuzzified value for a discrete fuzzy set.

For discrete membership function, the defuzzified value denoted as  $x^*$  using COG is defined as:

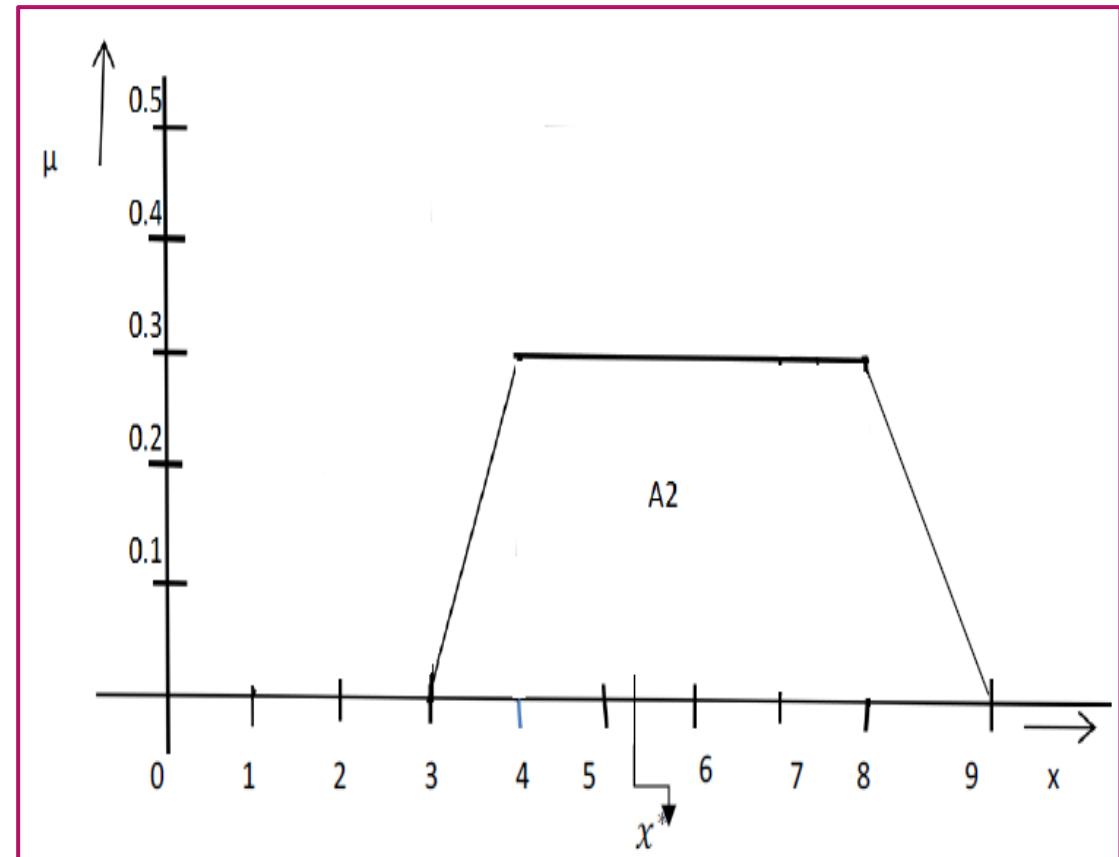
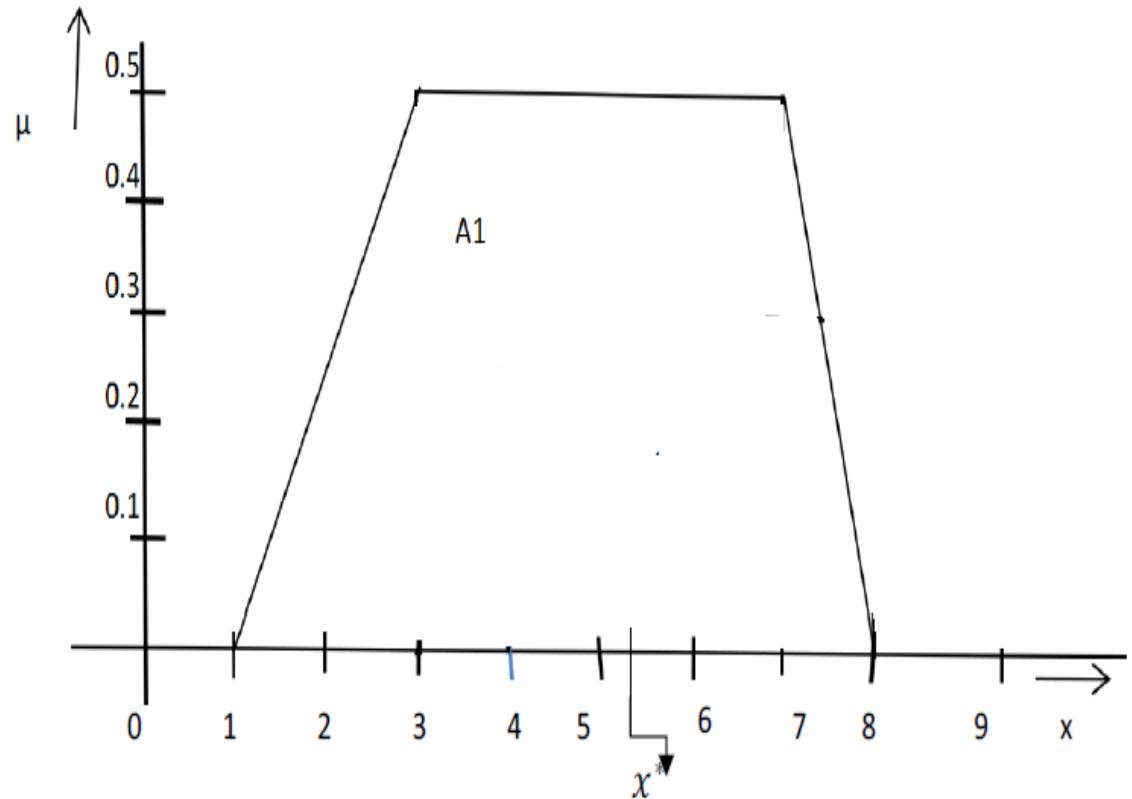
$$x^* = \frac{\sum_{i=1}^n x_i \cdot \mu(x_i)}{\sum_{i=1}^n \mu(x_i)}, \text{ Here } x_i \text{ indicates the sample element, } \mu(x_i) \text{ is}$$

the membership function, and n represents the number of elements in the sample.

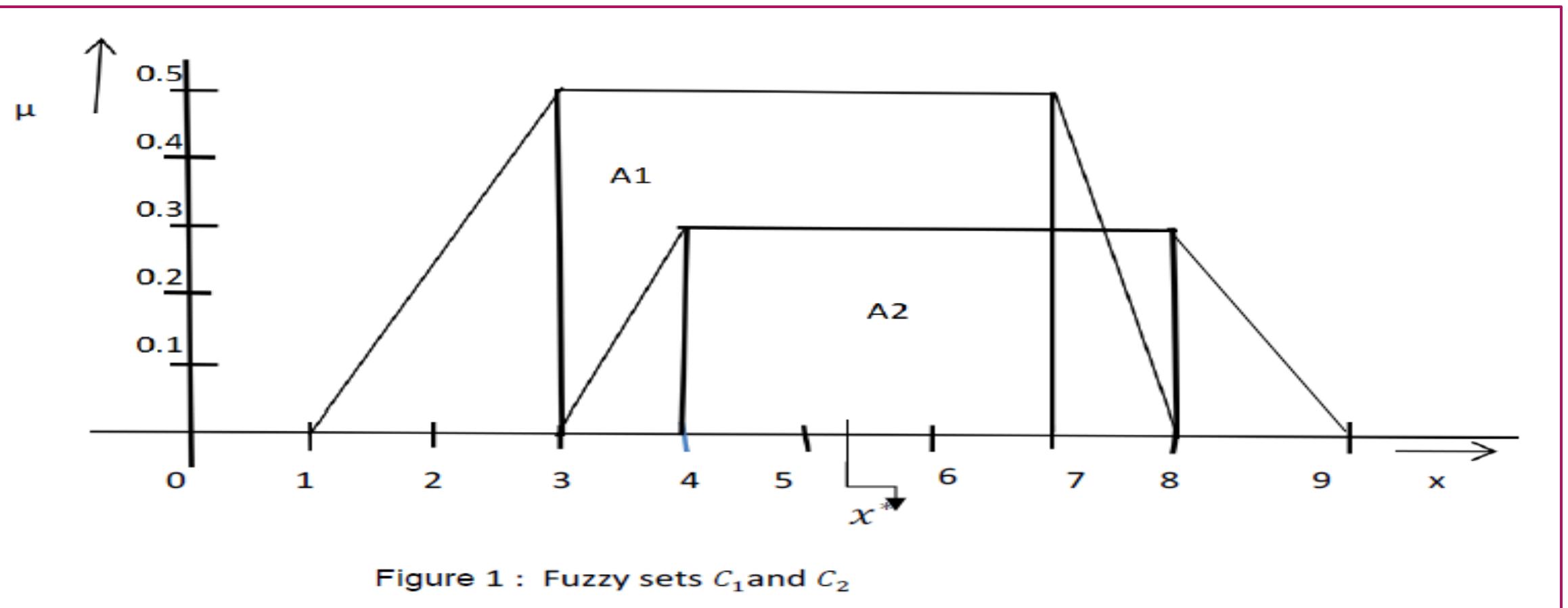
For continuous membership function,  $x^*$  is defined as :

$$x^* = \frac{\int x \mu_A(x) dx}{\int \mu_A(x) dx}$$

# Example: Fuzzy Set C1 and C2



# Combined Fuzzy Set



# Calculation of Area

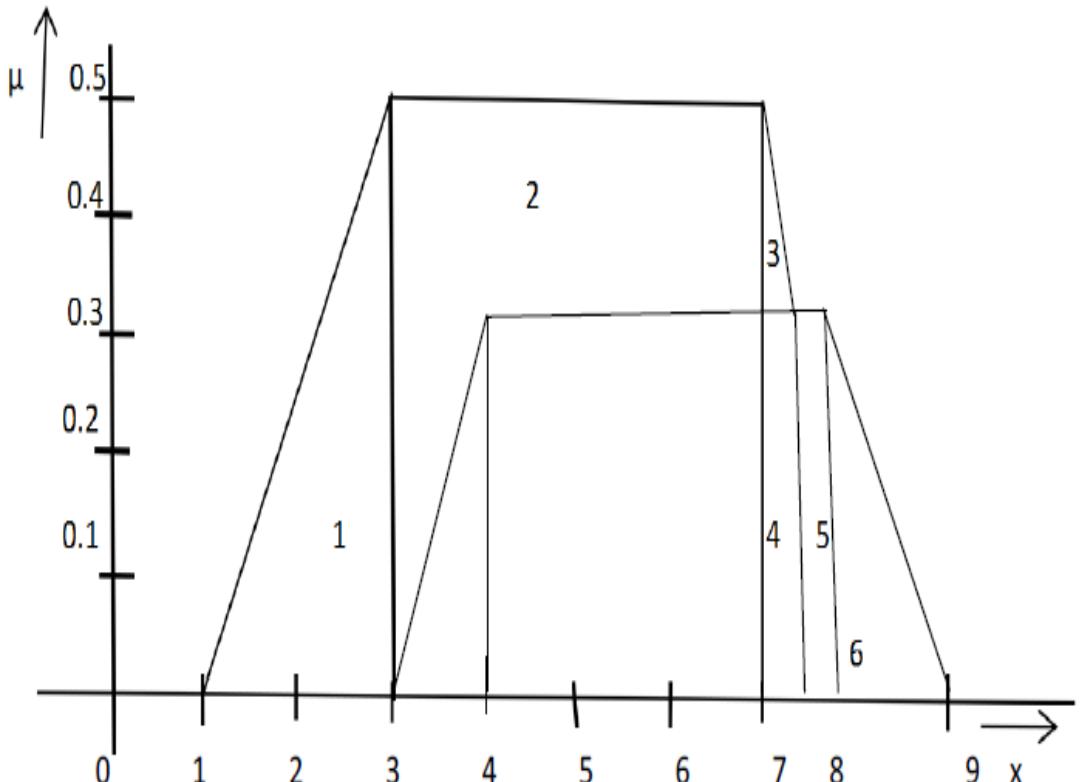


Figure 2 : Fuzzy sets C1 and C2

The defuzzified value  $x^*$  using COG is defined as:

$$x^* = \frac{\sum_{i=1}^N A_i \times \bar{x}_i}{\sum_{i=1}^N A_i}, \text{ Here } N \text{ indicates the number of sub-areas, } A_i \text{ and}$$

$\bar{x}_i$  represents the area and centroid of area, respectively, of  $i^{th}$  sub-area.

In the aggregated fuzzy set as shown in figure 2., the total area is divided into six sub-areas.

For COG method, we have to calculate the area and centroid of area of each sub-area.

These can be calculated as below.

The total area of the sub-area 1 is  $\frac{1}{2} * 2 * 0.5 = 0.5$

The total area of the sub-area 2 is  $(7-3) * 0.5 = 4 * 0.5 = 2$

The total area of the sub-area 3 is  $\frac{1}{2} * (7.5-7) * 0.2 = 0.5 * 0.5 * 0.2 = 0.05$

The total area of the sub-area 4 is  $0.5 * 0.3 = .15$

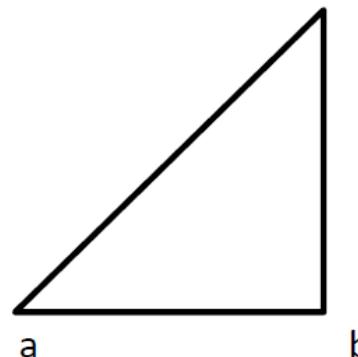
The total area of the sub-area 5 is  $0.5 * 0.3 = .15$

The total area of the sub-area 6 is  $\frac{1}{2} * 1 * 0.3 = .15$

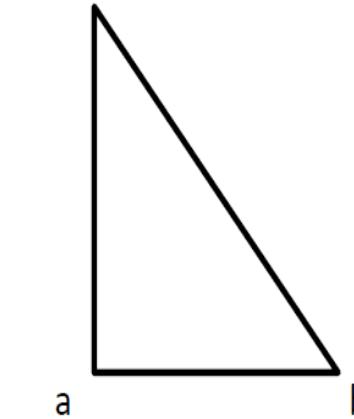
# Calculation of Centroids of primary shapes



$$\text{Centroid} = (a+b)/2$$



$$\text{Centroid} = a + (a+b)/3$$



$$\text{Centroid} = a + (b-a)/3$$

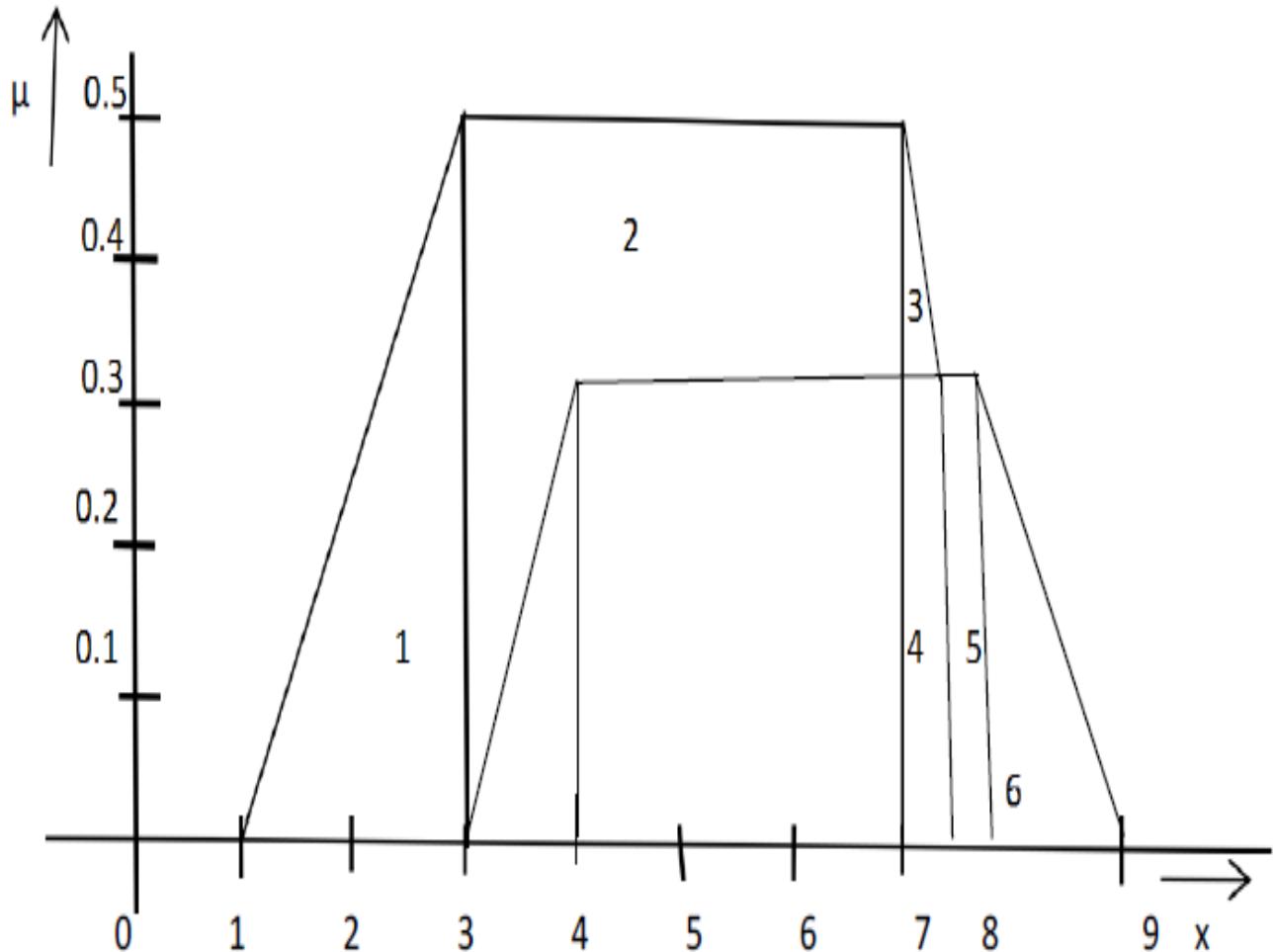


Figure 2 : Fuzzy sets  $C_1$  and  $C_2$

Calculation of Centroids:

$$\text{Centroid of } 1 = 1 + (1+3)/3 = 2.333$$

$$\text{Centroid of } 2 = (3+7)/2 = 5.00$$

$$\text{Centroid of } 3 = 7+(7.5-7)/3 = 7.166$$

$$\text{Centroid of } 4 = (7.5+7)/2 = 7.25$$

$$\text{Centroid of } 5 = (8+7.5)/2 = 7.75$$

$$\text{Centroid of } 6 = 8+(9-8)/3 = 8.333$$

Table 1

Sub-area number	Area( $A_i$ )	Centroid of area( $\bar{x}_i$ )	$A_i \cdot \bar{x}_i$
1	0.5	2.333	1.1665
2	02	5	10
3	.05	7.166	0.3583
4	.15	7.25	1.0875
5	.15	7.75	1.1625
6	.15	8.333	1.2499

The defuzzified value  $x^*$  will be

$$\frac{\sum_{i=1}^N A_i \times \bar{x}_i}{\sum_{i=1}^N A_i}$$

$$= \frac{(1.1665 + 10 + 0.3583 + 1.0875 + 1.1625 + 1.2499)}{(0.5 + 2 + .05 + .15 + .15 + .15)}$$

$$= (15.0247)/3 = 5.008$$

$$x^* = 5.008$$



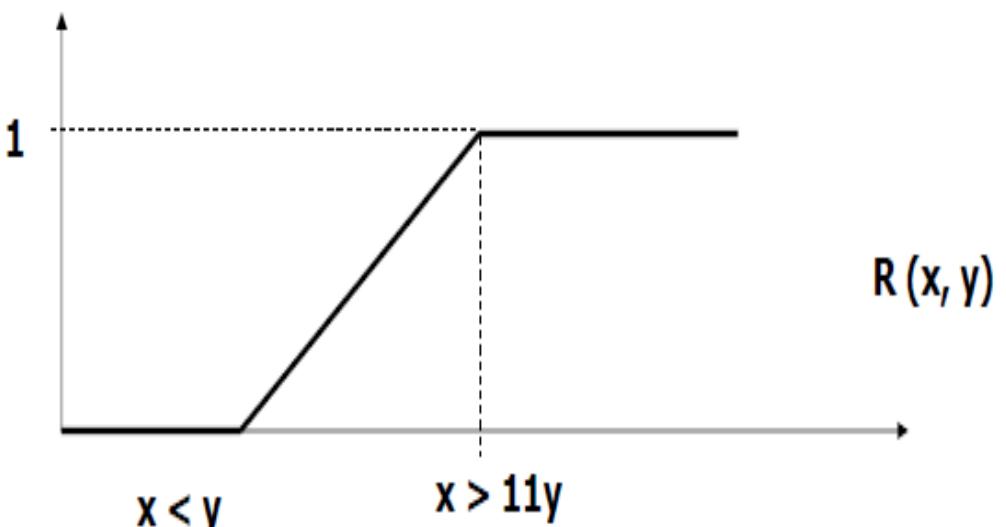
**It will be Continued....**

# Fuzzy Relation

# Definition

- Fuzzy relations are mapping elements of one universe, to those of another universe,  $Y$ , through the Cartesian product of two universes.  $X$ , Universe  $X = \{1, 2, 3\}$
- $R(X, Y) = \{[(x, y), \mu_R(x, y)] \mid (x, y) \in (X \times Y)\}$
- Where the fuzzy relation  $R$  has membership function
- $\mu_R(x, y) = \mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y))$
- It represents the strength of association between elements of the two sets
- Ex:  $R = "x \text{ is considerably larger than } y"$
- $R(X, Y) = \text{Relation between sets } X \text{ and } Y$
- $R(x, y) = \text{membership function for the relation } R(X, Y)$
- $R(X, Y) = \{R(x, y) / (x, y) \mid (x, y) \in (X \times Y)\}$

$$\triangleright R(x, y) = \begin{cases} 0 & \text{for } x \leq y \\ \frac{(x-y)}{(10-y)}, & \text{for } y < x \leq 11y \\ 1 & \text{for } x > 11y \end{cases}$$



### **Cartesian Product**

- Let  $A_1, A_2, \dots, A_n$  be fuzzy sets in  $U_1, U_2, \dots, U_n$ , respectively.

The Cartesian product of  $A_1, A_2, \dots, A_n$  is a fuzzy set in the space  $U_1 \times U_2 \times \dots \times U_n$  with the membership function as:

$$\mu_{A_1 \times A_2 \times \dots \times A_n}(x_1, x_2, \dots, x_n) = \min [\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n)]$$

- So, the Cartesian product of  $A_1, A_2, \dots, A_n$  are denoted by  $A_1 \times A_2 \times \dots \times A_n$

### **Crisp Relations**

- The relation between any two sets is the Cartesian product of the elements of  $A_1 \times A_2 \times \dots \times A_n$
- For  $X$  and  $Y$  universes  $X \times Y = \{(x, y) | x \in X, y \in Y\}$

$$\mu_{X \times Y}(x, y) = \begin{cases} 1, & (x, y) \in X \times Y \\ 0, & (x, y) \notin X \times Y \end{cases}$$

- This relation can be represented in a matrix format

### **Cartesian Product: Example**

- Let  $A = \{(3, 0.5), (5, 1), (7, 0.6)\}$
- Let  $B = \{(3, 1), (5, 0.6)\}$
- Find the product
- The product is all set of pairs from  $A$  and  $B$  with the minimum associated memberships
- $A \times B = \{[(3, 3), \min(0.5, 1)], [(5, 3), \min(1, 1)], [(7, 3), \min(0.6, 1)], [(3, 5), \min(0.5, 0.6)], [(5, 5), \min(1, 0.6)], [(7, 5), \min(0.6, 0.6)]\}$
- $= \{[(3, 3), 0.5], [(5, 3), 1], [(7, 3), 0.6], [(3, 5), 0.5], [(5, 5), 0.6], [(7, 5), 0.6]\}$

# Operations on Fuzzy Relations

- Since the fuzzy relation from  $X$  to  $Y$  is a fuzzy set in  $X \times Y$ , then the operations on fuzzy sets can be extended to fuzzy relations. Let  $R$  and  $S$  be fuzzy relations on the Cartesian space  $X \times Y$  then:
- Union:  $\mu_{R \cup S}(x, y) = \max [\mu_R(x, y), \mu_S(x, y)]$
- Intersection:  $\mu_{R \cap S}(x, y) = \min [\mu_R(x, y), \mu_S(x, y)]$
- Complement:  $\mu_{\bar{R}}(x, y) = 1 - \mu_R(x, y)$

- Assume two Universes:  $A = \{3, 4, 5\}$  and  $B = \{3, 4, 5, 6, 7\}$

$$\mu_R(x, y) = \begin{cases} (y-x)/(y+x+2) & \text{if } y > x \\ 0, & \text{if } y \leq x \end{cases}$$

- This can be expressed as follow:

$$R = \begin{matrix} 3 & 0 & 0.11 & 0.2 & 0.27 & 0.33 \\ 4 & 0 & 0 & 0.09 & 0.17 & 0.23 \\ 5 & 0 & 0 & 0 & 0.08 & 0.14 \\ 3 & 4 & 5 & 6 & 7 \end{matrix}$$

- This matrix represents the membership grades between elements in  $X$  and  $Y$
- $\mu_R(x, y) = \{[0/(3, 3)], [0.11/(3, 4)], [0.2/(3, 5)], \dots, [0.14/(5, 7)]\}$

## Fuzzy Relations: Example

- Assume two fuzzy sets:  $A = \{0.2/x_1 + 0.5/x_2 + 1/x_3\}$
- $B = \{0.3/y_1 + 0.9/y_2\}$
- Find the fuzzy relation (the Cartesian product)

$$A \times B = R = \begin{matrix} & x_1 & x_2 & x_3 \\ x_1 & \begin{pmatrix} 0.2 & 0.2 \\ 0.3 & 0.5 \\ 0.3 & 0.9 \end{pmatrix} \\ x_2 & & & \\ x_3 & & & \\ y_1 & & & \\ y_2 & & & \end{matrix}$$

- Consider the fuzzy relation. Express  $R$  using the resolution principle

$$R = \begin{pmatrix} 0.4 & 0.5 & 0 \\ 0.9 & 0.5 & 0 \\ 0 & 0 & 0.3 \\ 0.3 & 0.9 & 0.4 \end{pmatrix}$$

$$R = R_{0.3} + R_{0.4} + R_{0.5} + R_{0.9}$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

# Composition of Fuzzy Relations

- ***Composition of fuzzy relations used to combine fuzzy relations on different product spaces***
  
- ***Having a fuzzy relation; R (X × Y) and S (Y × Z), then Composition is used to determine a relation T (X × Z),***

➤ Consider two fuzzy relation;  $R (X \times Y)$  and  $S (Y \times Z)$ , then a relation  $T (X \times Z)$ , can be expressed as (max-min composition)

$$T = R \circ S$$

$$\begin{aligned}\mu_T(x, z) &= \text{max-min} [\mu_R(x, y), \mu_S(y, z)] \\ &= V [\mu_R(x, y) \wedge \mu_S(y, z)]\end{aligned}$$

➤ If algebraic product is adopted, then max-product composition is adopted:

$$T = R \circ S$$

$$\begin{aligned}\mu_T(x, z) &= \text{max} [\mu_R(x, y) \cdot \mu_S(y, z)] \\ &= V [\mu_R(x, y) \cdot \mu_S(y, z)]\end{aligned}$$

- The max-min composition can be interpreted as indicating the strength of the existence of relation between the elements of  $X$  and  $Z$

**Calculations of  $(R \circ S)$**  is almost similar to matrix multiplication

**Fuzzy relations composition have the same properties of:**

$$\text{Distributivity: } R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$$

$$\text{Associativity: } R \circ (S \circ T) = (R \circ S) \circ T$$

- Assume the following universes:  $X = \{x_1, x_2\}$ ,  $Y = \{y_1, y_2\}$ , and  $Z = \{z_1, z_2, z_3\}$ , with the following fuzzy relations.

$$R = \begin{matrix} & x_1 & x_2 \\ x_1 & \left[ \begin{array}{cc} 0.7 & 0.5 \\ 0.8 & 0.4 \end{array} \right] \\ & y_1 & y_2 \end{matrix} \quad \text{and} \quad S = \begin{matrix} & y_1 & y_2 \\ y_1 & \left[ \begin{array}{ccc} 0.9 & 0.6 & 0.2 \\ 0.1 & 0.7 & 0.5 \end{array} \right] \\ & z_1 & z_2 & z_3 \end{matrix}$$

- Find the fuzzy relation between  $X$  and  $Z$  using the max-min and max-product composition

- By max-min composition

$$\mu_T(x_1, z_1) = \max [\min(0.7, 0.9), \min(0.5, 0.1)] = 0.7$$

$$T = \begin{matrix} & x_1 & x_2 \\ x_1 & \left[ \begin{array}{ccc} 0.7 & 0.6 & 0.5 \\ 0.8 & 0.6 & 0.4 \end{array} \right] \\ & z_1 & z_2 & z_3 \end{matrix}$$

- By max-product composition

$$\mu_T(x_2, z_2) = \max [(0.8, 0.6), (0.4, 0.7)] = 0.48$$

$$T = \begin{matrix} & x_1 & x_2 \\ x_1 & \left[ \begin{array}{ccc} 0.63 & 0.42 & 0.25 \\ 0.72 & 0.48 & 0.20 \end{array} \right] \\ & z_1 & z_2 & z_3 \end{matrix}$$

Let us consider two kinds of troubles a PC may suffer from, viz., the *system hangs while running*, and the *system does not boot*. We symbolize the former by  $h$  and the later by  $b$  and define the set  $A = \{h, b\}$  of PC troubles. Two possible causes of these troubles are *computer virus* ( $v$ ) and *disc crash* ( $c$ ) and they form the set  $B = \{v, c\}$  of PC trouble makers. And finally, let the sources of the causes mentioned above are *internet* ( $i$ ) and *obsolescence* ( $o$ ) and  $C = \{i, o\}$  is the set of PC trouble causes. The relation between PC troubles and their causes is expressed by  $R$ , a fuzzy relation over  $A \times B$ . Similarly,  $S$  is the fuzzy relation over  $B \times C$ , i.e., the relation between the causes of troubles and the sources of those causes. The relations  $R$  and  $S$  in terms of their relation matrices are shown below.

$$R = \begin{matrix} v & c \\ h & b \end{matrix} \begin{bmatrix} 0.7 & 0.2 \\ 0.5 & 0.8 \end{bmatrix}, \quad S = \begin{matrix} i & o \\ v & c \end{matrix} \begin{bmatrix} 0.9 & 0.7 \\ 0.1 & 0.2 \end{bmatrix}$$

The relation between PC troubles and their ultimate sources, i.e., between  $A$  and  $C$ , can be computed on the basis of  $R$  and  $S$  above as the max–min composition  $R \circ S$ . The first element of  $R \circ S$ , expressed as  $(R \circ S)(h, i)$  is computed as follows.

$$\begin{aligned} (R \circ S)(h, i) &= \max \{\min(R(h, v), S(v, i)), \min(R(h, c), S(c, i))\} \\ &= \max \{\min(0.7, 0.9), \min(0.2, 0.1)\} \\ &= \max \{0.7, 0.1\} \\ &= 0.7 \end{aligned}$$

The rest of the elements of  $R \circ S$  can be found in a similar fashion.

$$(R \circ S)(h, o) = 0.7$$

$$(R \circ S)(b, i) = 0.5$$

$$(R \circ S)(b, o) = 0.5$$

And finally we get,

$$\begin{matrix} & i & o \\ h & 0.7 & 0.7 \\ b & 0.5 & 0.5 \end{matrix}$$

$$R \circ S = \begin{bmatrix} 0.7 & 0.7 \\ 0.5 & 0.5 \end{bmatrix}$$

Let  $A = \{\text{Mimi, Bob, Kitty, Jina}\}$  be a set of four children,  $B = \{\text{Tintin, Asterix, Phantom, Mickey}\}$  be a set of four comic characters, and  $C = \{\text{funny, cute, dreamy}\}$  be a set of three attributes. The fuzzy relations  $R = x$  Likes  $y$  is defined on  $A \times B$  and  $S = x$  IS  $y$  is defined on  $B \times C$  as shown in Table 2.20 and Table 2.21. Find  $R \circ S$ .

**Table 2.20.** Relation matrix for  $R = x \text{ Likes } y$ 

	$R \equiv \text{Likes}$			
	Tintin	Asterix	Phantom	Mickey
Mimi	0.8	0.5	0.7	0.8
Bob	0.4	0.9	0.3	0.3
Kitty	0.6	0.7	0.4	0.9
Jina	0.3	0.8	0.2	0.5

**Table 2.21.** Relation matrix for  $S = x \text{ IS } y$ 

	$S \equiv \text{IS}$		
	funny	cute	dreamy
Tintin	0.6	0.7	0.3
Asterix	0.8	0.4	0.2
Phantom	0.1	0.2	0.1
Mickey	0.9	0.8	0.3

**Table 2.22.** Relation matrix for  $R \circ S$ 

	$ROS$		
	funny	cute	dreamy
Mimi	0.8	0.8	0.3
Bob	0.8	0.4	0.3
Kitty	0.9	0.8	0.3
Jina	0.8	0.5	0.3

## Zadeh's Max-Min rule

If **x is A then y is B** with the implication of Zadeh's max-min rule can be written equivalently as :

$$R_{mm} = (A \times B) \cup (\bar{A} \times Y)$$

**Here**,  $Y$  is the universe of discourse with membership values for all  $y \in Y$  is 1, that is ,  $\mu_Y(y) = 1 \forall y \in Y$ .

**Suppose**  $X = \{a, b, c, d\}$  and  $Y = \{1, 2, 3, 4\}$

and  $A = \{(a, 0.0), (b, 0.8), (c, 0.6), (d, 1.0)\}$

$B = \{(1, 0.2), (2, 1.0), (3, 0.8), (4, 0.0)\}$  are two fuzzy sets.

We are to determine  $R_{mm} = (A \times B) \cup (\bar{A} \times Y)$

The computation of  $R_{mm} = (A \times B) \cup (\bar{A} \times Y)$  is as follows:

$$A \times B =$$

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ a & [0 & 0 & 0 & 0] \\ b & [0.2 & 0.8 & 0.8 & 0] \\ c & [0.2 & 0.6 & 0.6 & 0] \\ d & [0.2 & 1.0 & 0.8 & 0] \end{array}$$

and

$$\bar{A} \times Y =$$

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ a & [1 & 1 & 1 & 1] \\ b & [0.2 & 0.2 & 0.2 & 0.2] \\ c & [0.4 & 0.4 & 0.4 & 0.4] \\ d & [0 & 0 & 0 & 0] \end{array}$$

Therefore,

$$R_{mm} = (A \times B) \cup (\bar{A} \times Y) =$$

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ a & [1 & 1 & 1 & 1] \\ b & [0.2 & 0.8 & 0.8 & 0.2] \\ c & [0.4 & 0.6 & 0.6 & 0.4] \\ d & [0.2 & 1.0 & 0.8 & 0] \end{array}$$

Determine the implication relation :  
**If x is A then y is B**

This R represents **If x is A then y is B**

IF  $x$  is A THEN  $y$  is B ELSE  $y$  is C.

The relation  $R$  is equivalent to

$$R = (A \times B) \cup (\bar{A} \times C)$$

The membership function of  $R$  is given by

$$\mu_R(x, y) = \max[\min\{\mu_A(x), \mu_B(y)\}, \min\{\mu_{\bar{A}}(x), \mu_C(y)\}]$$

$$X = \{a, b, c, d\}$$

$$Y = \{1, 2, 3, 4\}$$

$$A = \{(a, 0.0), (b, 0.8), (c, 0.6), (d, 1.0)\}$$

$$B = \{(1, 0.2), (2, 1.0), (3, 0.8), (4, 0.0)\}$$

$$C = \{(1, 0), (2, 0.4), (3, 1.0), (4, 0.8)\}$$

Determine the implication relation :

**If  $x$  is A then  $y$  is B else  $y$  is C**

$$\text{Here, } A \times B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ a & 0 & 0 & 0 & 0 \\ b & 0.2 & 0.8 & 0.8 & 0 \\ c & 0.2 & 0.6 & 0.6 & 0 \\ d & 0.2 & 1.0 & 0.8 & 0 \end{bmatrix}$$

$$\text{and } \bar{A} \times C = \begin{bmatrix} 1 & 2 & 3 & 4 \\ a & 0 & 0.4 & 1.0 & 0.8 \\ b & 0 & 0.2 & 0.2 & 0.2 \\ c & 0 & 0.4 & 0.4 & 0.4 \\ d & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 2 & 3 & 4 \\ a & 0 & 0.4 & 1.0 & 0.8 \\ b & 0.2 & 0.8 & 0.8 & 0.2 \\ c & 0.2 & 0.6 & 0.6 & 0.4 \\ d & 0.2 & 1.0 & 0.8 & 0 \end{bmatrix}$$

Let  $R$  : If '*job is risky*' Then '*compensation is high*' be a fuzzy rule. There are four jobs  $job_1, job_2, job_3$  and  $job_4$ , constituting the universe  $job = \{job_1, job_2, job_3, job_4\}$ . Also, there are four categories of compensation  $c_1, c_2, c_3$ , and  $c_4$  in ascending order. Hence the universe for compensations is  $compensation = \{c_1, c_2, c_3, c_4\}$ . The fuzzy sets *risky-job* and *high-compensation* are defined on the universes  $job$  and  $compensation$  respectively as given below.

$$risky\text{-}job = \frac{0.3}{job_1} + \frac{0.8}{job_2} + \frac{0.7}{job_3} + \frac{0.9}{job_4}$$

$$high\text{-}compensation = \frac{0.2}{c_1} + \frac{0.4}{c_2} + \frac{0.6}{c_3} + \frac{0.8}{c_4}$$

Using Zadeh's interpretation, the truth value of rule  $R$  is expressed by the relation  
 $R = (\text{risky-job} \times \text{high-compensation}) \cup (\overline{\text{risky-job}} \times \text{compensation})$

Now,

$$\text{risky-job} \times \text{high-compensation} = \begin{matrix} & C_1 & C_2 & C_3 & C_4 \\ job_1 & [0.2 & 0.3 & 0.3 & 0.3] \\ job_2 & [0.2 & 0.4 & 0.6 & 0.8] \\ job_3 & [0.2 & 0.4 & 0.6 & 0.7] \\ job_4 & [0.2 & 0.4 & 0.6 & 0.8] \end{matrix},$$

$$\text{and, } \overline{\text{risky-job}} \times \text{compensation} = \begin{matrix} & C_1 & C_2 & C_3 & C_4 \\ job_1 & [0.7 & 0.7 & 0.7 & 0.7] \\ job_2 & [0.2 & 0.2 & 0.2 & 0.2] \\ job_3 & [0.3 & 0.4 & 0.6 & 0.7] \\ job_4 & [0.2 & 0.4 & 0.6 & 0.8] \end{matrix},$$

and finally,  $R = (\text{risky-job} \times \text{high-compensation}) \cup (\overline{\text{risky-job}} \times \text{compensation})$

$$= \begin{matrix} & C_1 & C_2 & C_3 & C_4 \\ job_1 & [0.7 & 0.7 & 0.7 & 0.7] \\ job_2 & [0.2 & 0.4 & 0.6 & 0.8] \\ job_3 & [0.3 & 0.4 & 0.6 & 0.7] \\ job_4 & [0.2 & 0.4 & 0.6 & 0.8] \end{matrix}$$

Hence the matrix  $R$  obtained above embodies the information in the fuzzy implication IF  $\text{job is risky}$  THEN  $\text{compensation is high}$  on the basis of the fuzzy concepts  $\text{risky-job}$  and  $\text{high-compensation}$  as defined above.

# Fuzzy number

- Definition: a fuzzy set  $A$  on  $\mathbf{R}$  must posses at least the following three properties
  - $A$  must be a normal fuzzy set
  - $A_\alpha$  must be a closed interval for every  $\alpha \in (0, 1]$  (convex)
  - the support of fuzzy set  $A$ ,  $A_{0+}$ , must be bounded

# Operations of Interval

Operation of fuzzy number can be generalized from that of crisp interval.  
Let's have a look at the operations of interval.

$\forall a_1, a_3, b_1, b_3 \in \Re$

$$A = [a_1, a_3], B = [b_1, b_3]$$

Assuming  $A$  and  $B$  as numbers expressed as interval, main operations of interval are

(1) Addition

$$[a_1, a_3] (+) [b_1, b_3] = [a_1 + b_1, a_3 + b_3]$$

(2) Subtraction

$$[a_1, a_3] (-) [b_1, b_3] = [a_1 - b_3, a_3 - b_1]$$

(3) Multiplication

$$\begin{aligned} [a_1, a_3] (\bullet) [b_1, b_3] &= [a_1 \bullet b_1 \wedge a_1 \bullet b_3 \wedge a_3 \bullet b_1 \wedge a_3 \bullet b_3, \\ &\quad a_1 \bullet b_1 \vee a_1 \bullet b_3 \vee a_3 \bullet b_1 \vee a_3 \bullet b_3] \end{aligned}$$

(4) Division

$$[a_1, a_3] (/) [b_1, b_3] = [a_1 / b_1 \wedge a_1 / b_3 \wedge a_3 / b_1 \wedge a_3 / b_3, a_1 / b_1 \vee a_1 / b_3 \vee a_3 / b_1 \vee a_3 / b_3]$$

excluding the case  $b_1 = 0$  or  $b_3 = 0$

(5) Inverse interval

$$[a_1, a_3]^{-1} = [1 / a_1 \wedge 1 / a_3, 1 / a_1 \vee 1 / a_3]$$

excluding the case  $a_1 = 0$  or  $a_3 = 0$

**Example**

There are two intervals A and B,

$$A = [3, 5], B = [-2, 7]$$

then following operation might be set.

$$A(+)B = [3 - 2, 5 + 7] = [1, 12]$$

$$A(-)B = [3 - 7, 5 - (-2)] = [-4, 7]$$

$$\begin{aligned}A(\bullet)B &= [3 \bullet (-2) \wedge 3 \bullet 7 \wedge 5 \bullet (-2) \wedge 5 \bullet 7, 3 \bullet (-2) \vee \dots] \\&= [-10, 35]\end{aligned}$$

$$\begin{aligned}A(/)B &= [3 / (-2) \wedge 3 / 7 \wedge 5 / (-2) \wedge 5 / 7, 3 / (-2) \vee \dots] \\&= [-2.5, 5/7]\end{aligned}$$

$$B^{-1} = [-2, 7]^{-1} = \left[ \frac{1}{(-2)} \wedge \frac{1}{7}, \frac{1}{(-2)} \vee \frac{1}{7} \right] = \left[ -\frac{1}{2}, \frac{1}{7} \right]$$

# Operations of Fuzzy Number

- $\alpha$ -cut

- Addition:  $A(+)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(+)B_\alpha)$

$$A = [a_1, a_3] \quad a_1, a_3 \in \Re$$

$$A_\alpha = [a_1^{(\alpha)}, a_3^{(\alpha)}], \forall \alpha \in [0, 1], a_1^{(\alpha)}, a_3^{(\alpha)} \in \Re$$

$$B = [b_1, b_3], \quad b_1, b_3, \in \Re$$

$$B_\alpha = [b_1^{(\alpha)}, b_3^{(\alpha)}], \forall \alpha \in [0, 1], b_1^{(\alpha)}, b_3^{(\alpha)} \in \Re$$

operations between  $A_\alpha$  and  $B_\alpha$  can be described as follows :

$$[a_1^{(\alpha)}, a_3^{(\alpha)}] (+) [b_1^{(\alpha)}, b_3^{(\alpha)}] = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_3^{(\alpha)} + b_3^{(\alpha)}]$$

- Subtraction:  $A(-)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(-)B_\alpha)$

- Multiplication:  $A(\times)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(\times)B_\alpha)$

- Division:  $A(/)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(/)B_\alpha)$

- Minimum:  $A(\wedge)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(\wedge)B_\alpha)$

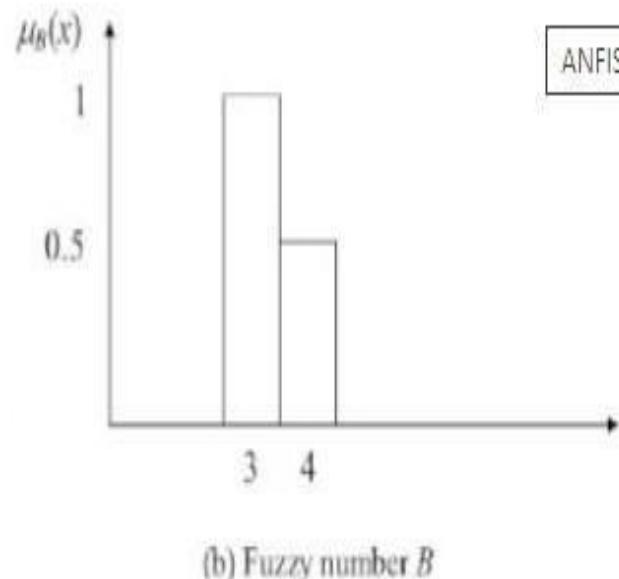
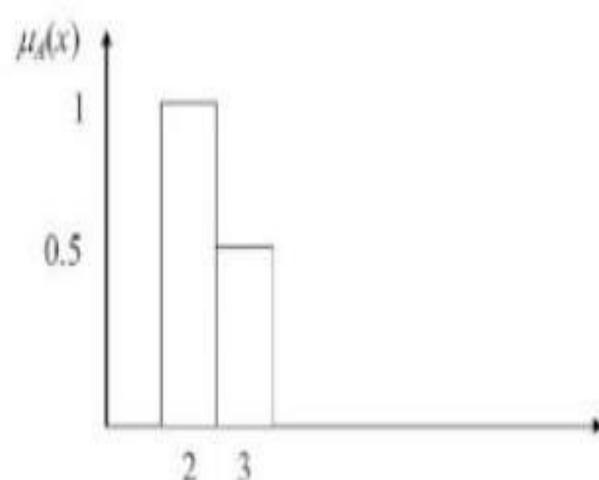
- Maximum:  $A(\vee)B = \bigcup_{\alpha \in [0,1]} \alpha(A_\alpha(\vee)B_\alpha)$

### Example : Addition A(+) $B$

For further understanding of fuzzy number operation, let us consider two fuzzy sets  $A$  and  $B$ . Note that these fuzzy sets are defined on discrete numbers for simplicity.

$$A = \{(2, 1), (3, 0.5)\}, B = \{(3, 1), (4, 0.5)\}$$

First of all, our concern is addition between  $A$  and  $B$ . To induce  $A(+) $B$ , for all  $x \in A, y \in B, z \in A(+) $B$ , we check each case as follows(Fig 5.4) :$$



- $\alpha = 0.5$

- $- A_{0.5} = [2, 3], B_{0.5} = [3, 4]$

- $- A_{0.5}(+)B_{0.5} = [5, 7]$

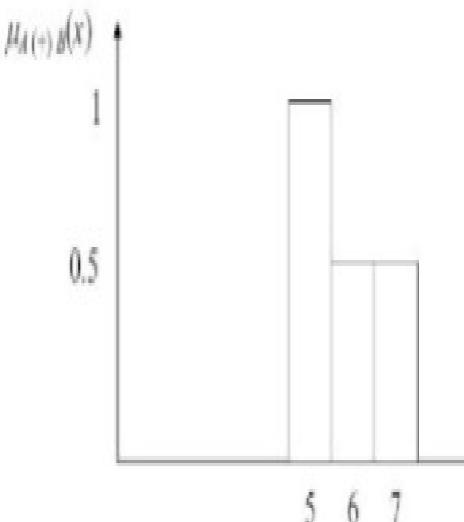
- $\alpha = 1.0$

- $- A_{1.0} = 2, B_{1.0} = 3$

- $- A_{1.0}(+)B_{1.0} = 5$

- $- A(+) $B =  $0.5(A_{0.5}(+)B_{0.5}) \cup 1.0(A_{1.0}(+)B_{1.0}) = \frac{0.5}{[5, 7]} \cup \frac{1.0}{5}$$$

$$A(+) $B = \frac{1.0}{5} + \frac{0.5}{6} + \frac{0.5}{7}$$$



## Extension principle

$\forall x, y, z \in \Re$

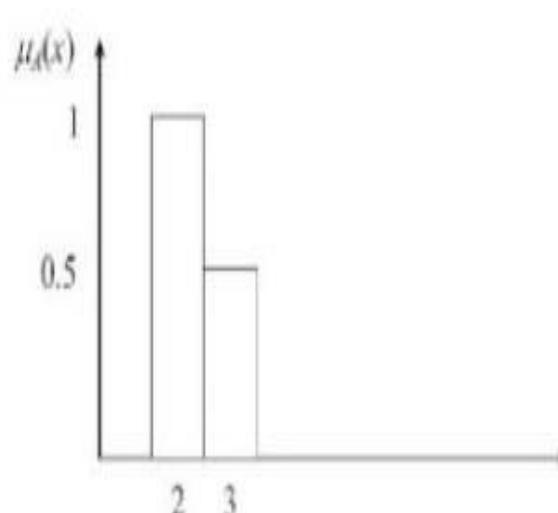
- Addition:  $A(+)B \quad \mu_{A(+)}(z) = \vee_{z=x+y} (\mu_A(x) \wedge \mu_B(y))$
- Subtraction:  $A(-)B \quad \mu_{A(-)}(z) = \vee_{z=x-y} (\mu_A(x) \wedge \mu_B(y))$
- Multiplication:  $A(\bullet)B \quad \mu_{A(\bullet)}(z) = \vee_{z=x \bullet y} (\mu_A(x) \wedge \mu_B(y))$
- Division:  $A(/)B \quad \mu_{A(/)}(z) = \vee_{z=x/y} (\mu_A(x) \wedge \mu_B(y)) \quad \square$
- Minimum:  $A(\wedge)B \quad \mu_{A(\wedge)}(z) = \vee_{z=x \wedge y} (\mu_A(x) \wedge \mu_B(y))$
- Maximum:  $A(\vee)B \quad \mu_{A(\vee)}(z) = \vee_{z=x \vee y} (\mu_A(x) \wedge \mu_B(y))$

### Example Addition $A(+B)$

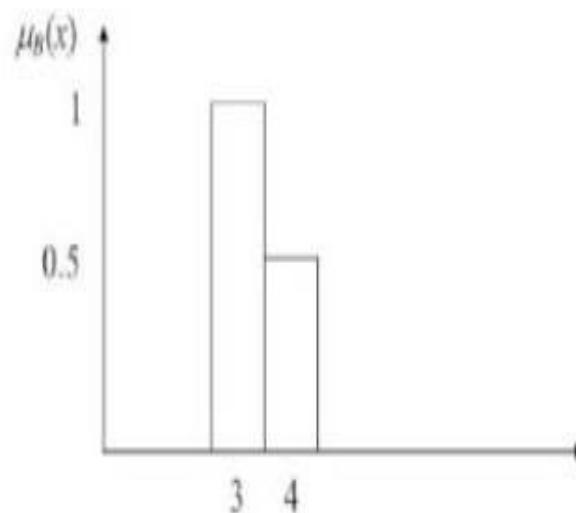
For further understanding of fuzzy number operation, let us consider two fuzzy sets  $A$  and  $B$ . Note that these fuzzy sets are defined on discrete numbers for simplicity.

$$A = \{(2, 1), (3, 0.5)\}, B = \{(3, 1), (4, 0.5)\}$$

First of all, our concern is addition between  $A$  and  $B$ . To induce  $A(+B)$ , for all  $x \in A, y \in B, z \in A(+B)$ , we check each case as follows(Fig 5.4) :



(a) Fuzzy set  $A$



(b) Fuzzy number  $B$

i) for  $z < 5$ ,

$$\mu_{A(+B)}(z) = 0$$

ii)  $z = 5$

results from  $x + y = 2 + 3$

$$\mu_A(2) \wedge \mu_B(3) = 1 \wedge 1 = 1$$

$$\mu_{A(+B)}(5) = \bigvee_{z=2+3} (1) = 1$$

iii)  $z = 6$

results from  $x + y = 3 + 3$  or  $x + y = 2 + 4$

$$\mu_A(3) \wedge \mu_B(3) = 0.5 \wedge 1 = 0.5$$

$$\mu_A(2) \wedge \mu_B(4) = 1 \wedge 0.5 = 0.5$$

$$\mu_{A(+B)}(6) = \bigvee_{z=3+3} (0.5, 0.5) = 0.5$$

iv)  $z = 7$

results from  $x + y = 3 + 4$

$$\mu_A(3) \wedge \mu_B(4) = 0.5 \wedge 0.5 = 0.5$$

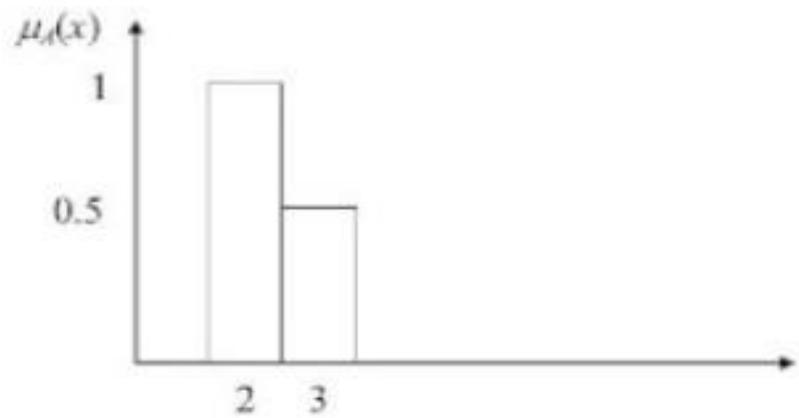
$$\mu_{A(+B)}(7) = \bigvee_{z=3+4} (0.5) = 0.5$$

v) for  $z > 7$

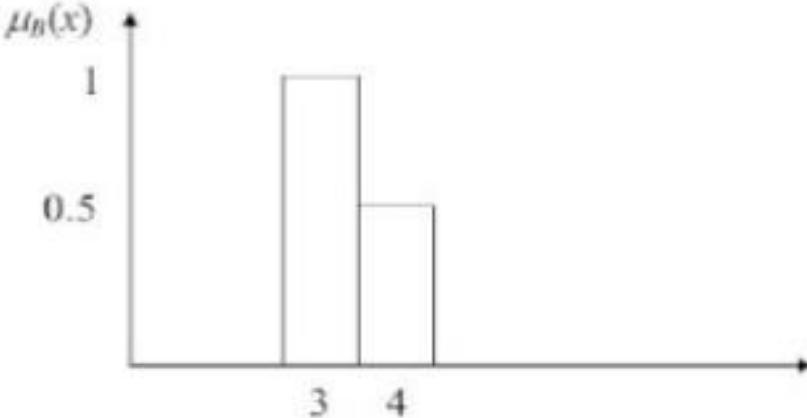
$$\mu_{A(+B)}(z) = 0$$

so  $A(+B)$  can be written as

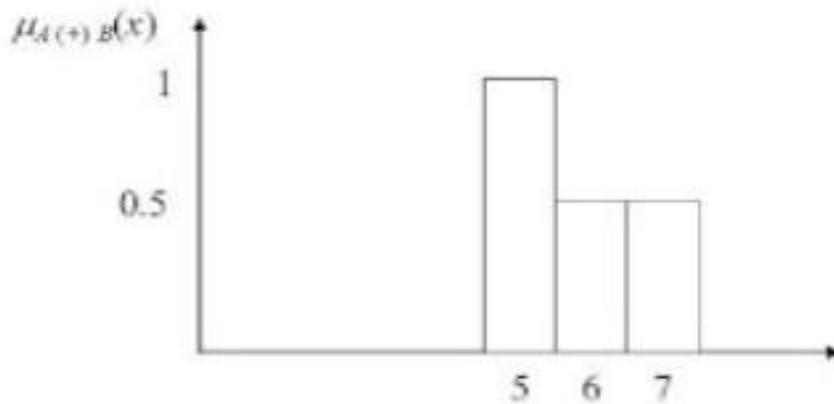
$$A(+B) = \{(5, 1), (6, 0.5), (7, 0.5)\}$$



(a) Fuzzy set  $A$



(b) Fuzzy number  $B$



(c) Fuzzy set  $A (+) B$

### Example Subtraction $A(-)B$

Let's manipulate  $A(-)B$  between our previously defined fuzzy set  $A$  and  $B$ . For  $x \in A, y \in B, z \in A(-)B$ , fuzzy set  $A(-)B$  is defined as follows (Fig 5.5).

i) For  $z < -2$ ,

$$\mu_{A(-)B}(z) = 0$$

ii)  $z = -2$

results from  $x - y = 2 - 4$

$$\mu_A(2) \wedge \mu_B(4) = 1 \wedge 0.5 = 0.5$$

$$\mu_{A(-)B}(-2) = 0.5$$

iii)  $z = -1$

results from  $x - y = 2 - 3$  or  $x - y = 3 - 4$

$$\mu_A(2) \wedge \mu_B(3) = 1 \wedge 1 = 1$$

$$\mu_A(3) \wedge \mu_B(4) = 0.5 \wedge 0.5 = 0.5$$

$$\mu_{A(-)B}(-1) = \bigvee_{\substack{-1=2-3 \\ -1=3-4}} (1, 0.5) = 1$$

iv)  $z = 0$

results from  $x - y = 3 - 3$

$$\mu_A(3) \wedge \mu_B(3) = 0.5 \wedge 1 = 0.5$$

$$\mu_{A(-)B}(0) = 0.5$$

v) For  $z \geq 1$

$$\mu_{A(-)B}(z) = 0$$

so  $A(-)B$  is expressed as

$$A(-)B = \{(-2, 0.5), (-1, 1), (0, 0.5)\}$$

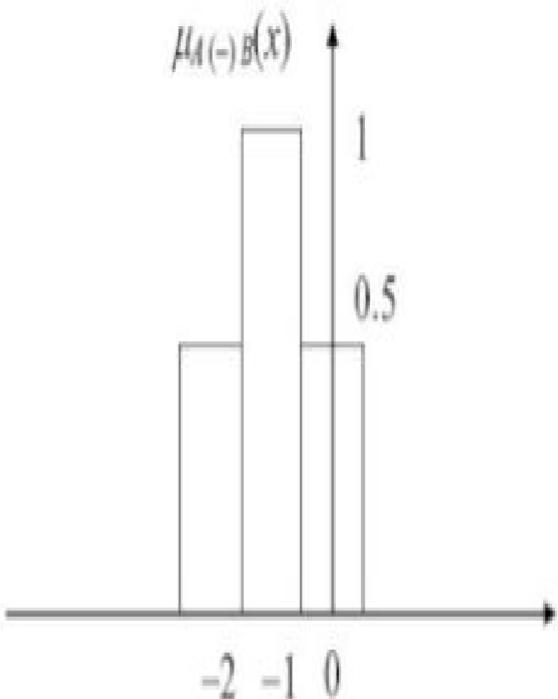


Fig. 5.5. Fuzzy number  $A(-)B$

## Example Max operation $A(\vee)B$

Let's deal with the operation Max  $A(\vee)B$  between  $A$  and  $B$  for  $x \in A, y \in B, z \in A(\vee)B$ , fuzzy set  $A(\vee)B$  is defined by  $\mu_{A(\vee)B}(z)$ .

i)  $z \leq 2$

$$\mu_{A(\vee)B}(z) = 0$$

ii)  $z = 3$

from  $x \vee y = 2 \vee 3$  and  $x \vee y = 3 \vee 3$

$$\mu_A(2) \wedge \mu_B(3) = 1 \wedge 1 = 1$$

$$\mu_A(3) \wedge \mu_B(3) = 0.5 \wedge 1 = 0.5$$

$$\mu_{A(\vee)B}(3) = \begin{cases} 1, & \text{if } 3 = 2 \vee 3 \\ 0.5, & \text{if } 3 = 3 \vee 3 \end{cases}$$

iii)  $z = 4$

from  $x \vee y = 2 \vee 4$  and  $x \vee y = 3 \vee 4$

$$\mu_A(2) \wedge \mu_B(4) = 1 \wedge 0.5 = 0.5$$

$$\mu_A(3) \wedge \mu_B(4) = 0.5 \wedge 0.5 = 0.5$$

$$\mu_{A(\vee)B}(4) = \begin{cases} 0.5, & \text{if } 4 = 2 \vee 4 \\ 0.5, & \text{if } 4 = 3 \vee 4 \end{cases}$$

v)  $z > 5$

impossible  $\mu_{A(\vee)B}(z) = 0$

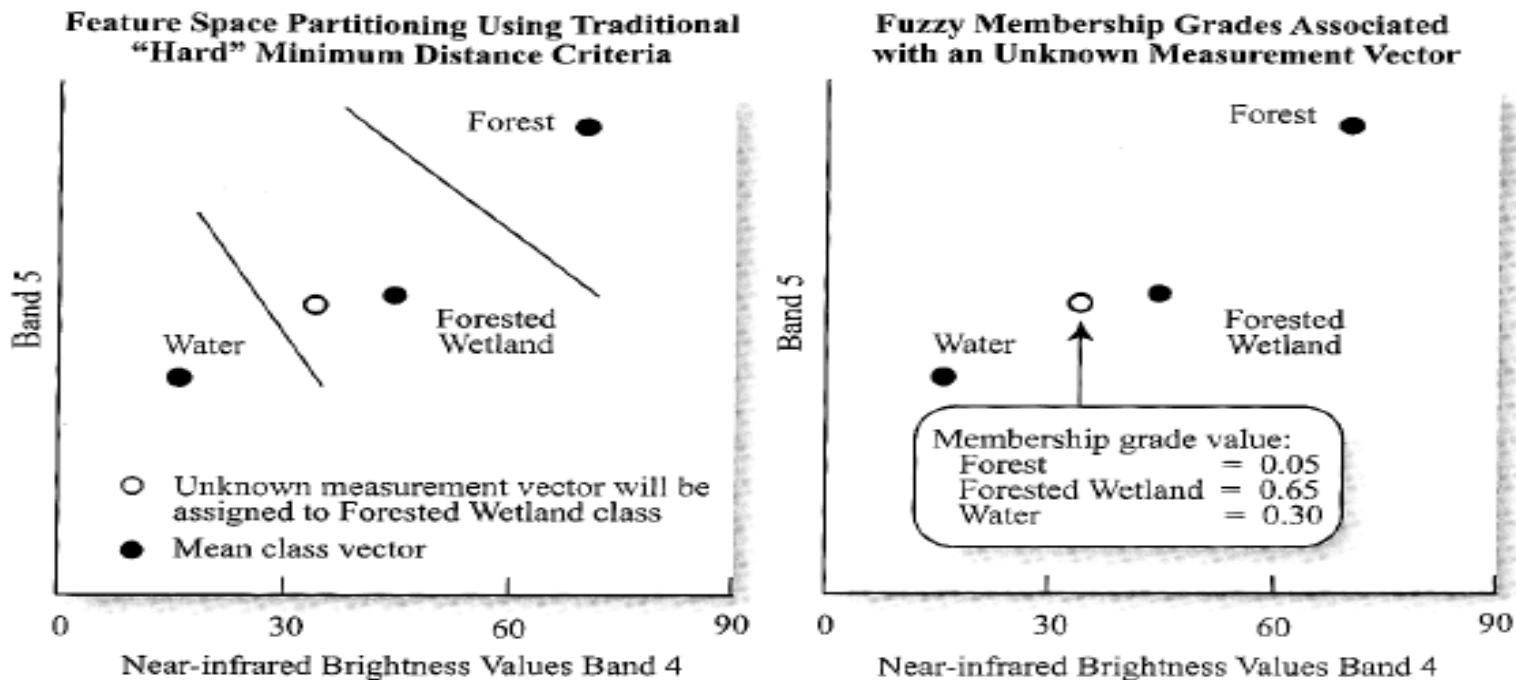
so  $A(\vee)B$  is defined to be

$$A(\vee)B = \{(3, 1), (4, 0.5)\}$$

so far we have seen the results of operations are fuzzy sets, and thus we come to realize that the extension principle is applied to the operation of fuzzy number.

# Hard vs Soft Classification

## Hard- versus soft-classifiers



## Why use soft-classifiers?

- Sub-pixel classification
- Uncertainty of classification/scheme
- Incorporating ancillary data (hardeners)

# Fuzzy Clustering

Let  $X = \{x_1, x_2, \dots, x_n\}$  be a set of given data. A fuzzy pseudopartition or fuzzy  $c$ -partition of  $X$  is a family of fuzzy subsets of  $X$ , denoted by  $\mathcal{P} = \{A_1, A_2, \dots, A_c\}$ , which satisfies

$$\sum_{i=1}^c A_i(x_k) = 1$$

for all  $k \in \mathbb{N}_n$  and

$$0 < \sum_{k=1}^n A_i(x_k) < n$$

for all  $i \in \mathbb{N}_c$ , where  $c$  is a positive integer.

For instance, given  $X = \{x_1, x_2, x_3\}$  and

$$A_1 = .6/x_1 + 1/x_2 + .1/x_3,$$

$$A_2 = .4/x_1 + 0/x_2 + .9/x_3,$$

$$v_i = \frac{\sum_{k=1}^n [A_i(x_k)]^m x_k}{\sum_{k=1}^n [A_i(x_k)]^m}$$

$$J_m(\mathcal{P}) = \sum_{k=1}^n \sum_{i=1}^c [A_i(x_k)]^m \|x_k - v_i\|^2,$$

# Fuzzy c-means

The algorithm is based on the assumption that the desired number of clusters  $c$  is given and, in addition, a particular distance, a real number  $m \in (1, \infty)$ , and a small positive number  $\varepsilon$ , serving as a stopping criterion, are chosen.

**Step 1.** Let  $t = 0$ . Select an initial fuzzy pseudopartition  $\mathcal{P}^{(0)}$ .

**Step 2.** Calculate the  $c$  cluster centers  $\mathbf{v}_1^{(t)}, \dots, \mathbf{v}_c^{(t)}$  by (13.3) for  $\mathcal{P}^{(t)}$  and the chosen value of  $m$ .

**Step 3.** Update  $\mathcal{P}^{(t+1)}$  by the following procedure: For each  $\mathbf{x}_k \in X$ , if  $\|\mathbf{x}_k - \mathbf{v}_i^{(t)}\|^2 > 0$  for all  $i \in \mathbb{N}_c$ , then define

$$A_i^{(t+1)}(\mathbf{x}_k) = \left[ \sum_{j=1}^c \left( \frac{\|\mathbf{x}_k - \mathbf{v}_i^{(t)}\|^2}{\|\mathbf{x}_k - \mathbf{v}_j^{(t)}\|^2} \right)^{\frac{1}{m-1}} \right]^{-1};$$

if  $\|\mathbf{x}_k - \mathbf{v}_i^{(t)}\|^2 = 0$  for some  $i \in I \subseteq \mathbb{N}_c$ , then define  $A_i^{(t+1)}(\mathbf{x}_k)$  for  $i \in I$  by any nonnegative real numbers satisfying

$$\sum_{i \in I} A_i^{(t+1)}(\mathbf{x}_k) = 1,$$

and define  $A_i^{(t+1)}(\mathbf{x}_k) = 0$  for  $i \in \mathbb{N}_c - I$ .

**Step 4.** Compare  $\mathcal{P}^{(t)}$  and  $\mathcal{P}^{(t+1)}$ . If  $|\mathcal{P}^{(t+1)} - \mathcal{P}^{(t)}| \leq \varepsilon$ , then stop; otherwise, increase  $t$  by one and return to Step 2.

In Step 4,  $|\mathcal{P}^{(t+1)} - \mathcal{P}^{(t)}|$  denotes a distance between  $\mathcal{P}^{(t+1)}$  and  $\mathcal{P}^{(t)}$  in the space  $\mathbb{R}^{n \times c}$ . An example of this distance is

$$|\mathcal{P}^{(t+1)} - \mathcal{P}^{(t)}| = \max_{i \in \mathbb{N}_c, k \in \mathbb{N}_n} |A_i^{(t+1)}(\mathbf{x}_k) - A_i^{(t)}(\mathbf{x}_k)|.$$

P=	A1	A2	A3	
X1	0.2	0.4	0.4	=1
X2	0.3	0.5	0.2	
X3	0.1	0.9	0.0	
X4	0.8	0.1	0.1	
X5	0.9	0.1	0.0	

$$A_1(X1) = 0.2$$

$$U_{A1}(X1) = 0.2$$

$$\mathbf{v}_i = \frac{\sum_{k=1}^n [A_i(\mathbf{x}_k)]^m \mathbf{x}_k}{\sum_{k=1}^n [A_i(\mathbf{x}_k)]^m}$$

# Sample calculation

- ▶  $A_i(X_k)$   $A_1(X_1)=0.2$
- ▶  $X_1=(1,1)=(x_{11},x_{12})$
- ▶  $X_2=(2,3)$
- ▶  $X_3=(-4,-3)$
- ▶  $X_4=(0,0)$
- ▶  $X_5=(4,-4)$

$$V_1=(V_{11}, V_{12}), V_2=(V_{21}, V_{22})$$

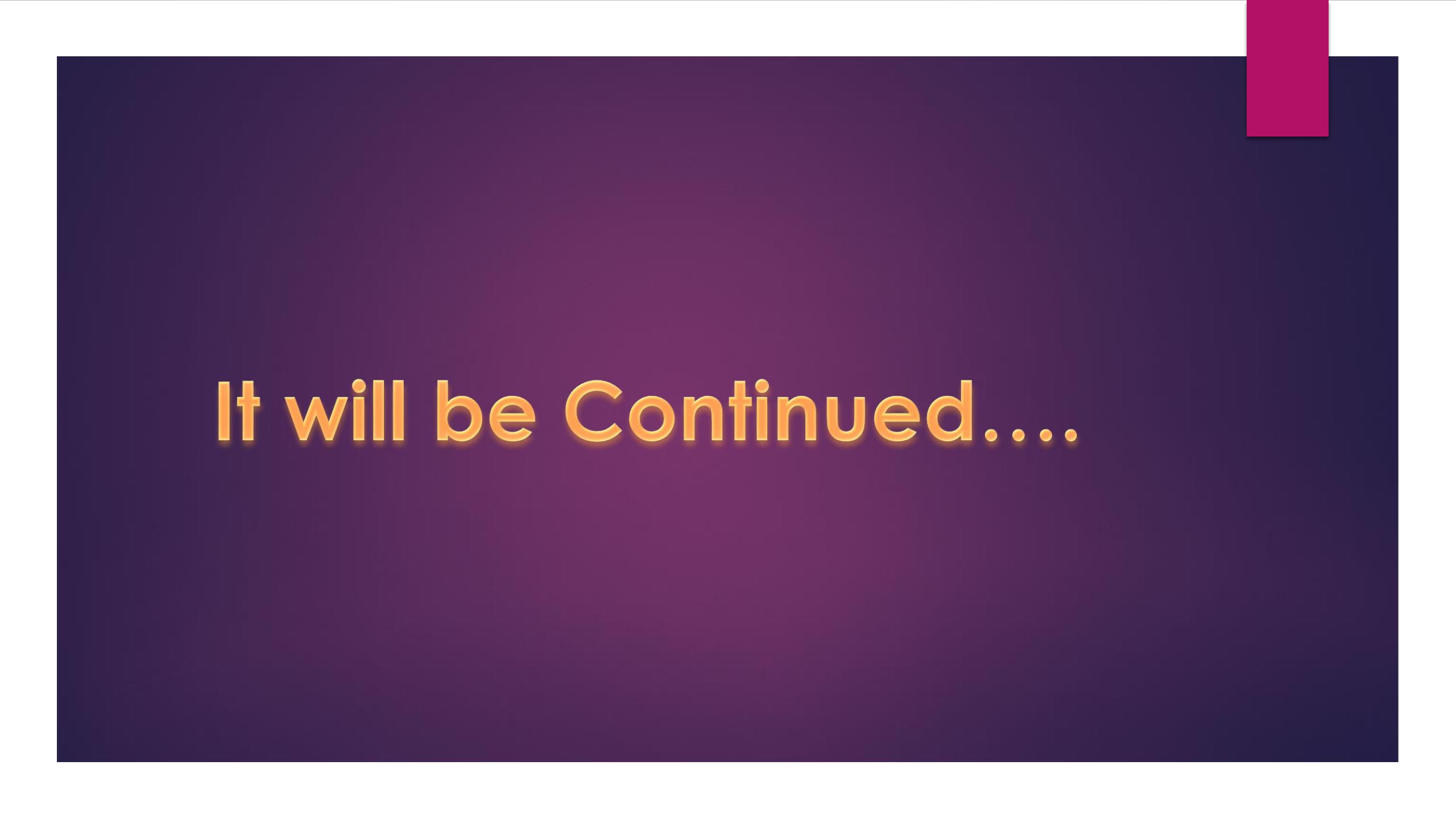
$$V_{11} = [A_1(X_1)*A_1(X_1)*X_{11} + A_1(X_2)*A_1(X_2)*X_{21} + A_1(X_3)*A_1(X_3)*X_{31} + A_1(X_4)*A_1(X_4)*X_{41} + A_1(X_5)*A_1(X_5)*X_{51}] / [A_1(X_1)*A_1(X_1) + A_1(X_2)*A_1(X_2) + A_1(X_3)*A_1(X_3) + A_1(X_4)*A_1(X_4) + A_1(X_5)*A_1(X_5)]$$

$$V_{11} = [0.2*0.2*1 + 0.3*0.3*2 + 0.1*0.1*(-4) + 0.8*0.8*0 + 0.9*0.9*4] / [0.2*0.2 + 0.3*0.3 + 0.1*0.1 + 0.8*0.8 + 0.9*0.9] = 3.42/1.59 = 2.15$$

$$V_{12} = -0.02; \quad V_1 = \{2.15, -0.02\}; \quad V_2 = ?$$

P=	A1	A2	A3
X1	0.2	0.4	0.4
X2	0.3	0.5	0.2
X3	0.1	0.9	0.0
X4	0.8	0.1	0.1
X5	0.9	0.1	0.0

$$V_i = \frac{\sum_{k=1}^n [A_i(x_k)]^m x_k}{\sum_{k=1}^n [A_i(x_k)]^m}$$



**It will be Continued....**

# Proposition Logic

BY

DR. ANUPAM GHOSH

DATE: 07.10.2023

# Logic is a truth-preserving system of inference

*Truth-preserving:*  
If the initial statements are true, the inferred statements will be true

*System:* a set of mechanistic transformations, based on syntax alone

*Inference:* the process of deriving (inferring) new statements from old statements

# Logic: Other definitions

Any ‘formal system’ can be considered a logic if it has:

- ▶ – a well-defined syntax;
- ▶ – a well-defined semantics; and
- ▶ – a well-defined proof-theory
- ▶ The syntax of a logic defines the syntactically acceptable objects of the language, which are properly called well-formed formulae (wff). (We shall just call them formulae.)
- ▶ The semantics of a logic associate each formula with a meaning.
- ▶ The proof theory is concerned with manipulating formulae according to certain rules.

# Proposition Logic

- ▶ The simplest, and most abstract logic we can study is called propositional logic.
- ▶ Definition: A proposition is a statement that can be either true or false; it must be one or the other, and it cannot be both
- ▶ It is possible to determine whether any given statement is a proposition by prefixing it with:

It is true that ...

and seeing whether the result makes grammatical sense.
- ▶ A number of connectives which will allow us to build up complex propositions

# Basic Operations

The NOT operation:  $\neg$

The AND operation:  $\wedge$

The OR operation:  $\vee$

The “implication” operation:  $\rightarrow$

The “equivalence” operation:  $\leftrightarrow$  or  $\equiv$

## NOT Operation

- In plain text, we can describe the operation to be the opposite of what the original value is.

Example:      Let  $p$  = “cat”  
                  Then  $\neg p$  will be “not a cat”

Example:      Let  $p$  = “tired”  
                  Then  $\neg p$  = “not tired”

# AND Operation

- In plain text,

Example: Let  $p$  = "tired" and  $q$  = "hungry"  
Then  $p \wedge q$  = "tired and hungry"  
Also,  $\neg p \wedge q$  = "not tired and hungry"  
Or,  $p \wedge \neg q$  = "tired and not hungry"

# OR Operation

- In plain text,

Example: Let  $p$  = "tired" and  $q$  = "hungry"  
If "tired and not hungry",  $p \vee q$  is "true"  
If "not tired and not hungry",  $p \vee q$  is "false"

# "Implication" Operation

- For implication, it is slightly more complicated and requires two variables. It will return "true" if the initial condition is false, regardless of the value of the second variable, and when both variables are true. (See the truth table below)
- Denoted by the symbol " $\rightarrow$ "
- Example:  $p \rightarrow q$

Truth Table:

<b>p</b>	<b>q</b>	<b><math>p \rightarrow q</math></b>
0	0	1
0	1	1
1	0	0
1	1	1

# “Implication” Operation

- $p \rightarrow q$  is also equivalent (the same as)  $\neg p \vee q$
- We can verify this using truth tables:

p	q	$\neg p \vee q$
0	0	1
0	1	1
1	0	0
1	1	1

## Equivalence

- Equivalence requires two variables and will return “true” if both variables have the same value
- Often denoted by the symbol “ $\leftrightarrow$ ” or “ $\equiv$ ”
- Example:  $p \leftrightarrow q$  or  $p \equiv q$

Truth Table:

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

## Biconditional $\leftrightarrow$

- The biconditional of p and q:  $p \leftrightarrow q \triangleq (p \rightarrow q) \wedge (q \rightarrow p)$ 
  - True only when p and q have identical truth values
- If and only if (iff)

Known operators:

$\wedge$  conjunction (and),

$\vee$  disjunction (or),

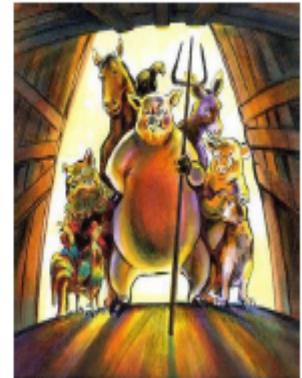
$\neg$  negation,

$\rightarrow$  conditional (if then)

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

## Operator Precedence

- From high to low:  $( )$ ,  $\neg$ ,  $\wedge\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$
- When equal priority instances of *binary connectives* are not separated by  $( )$ , the *leftmost one has precedence*. E.g.  $p \rightarrow q \rightarrow r \equiv (p \rightarrow q) \rightarrow r$
- When instances of  $\neg$  are not separated by  $( )$ , the *rightmost one has precedence*:  
E.g.  $\neg\neg\neg p \equiv \neg(\neg(\neg p))$



# Tautology

- Tautology is very similar to logical equivalence
  - When all values are “true” that is a tautology

Example:  $p \equiv q$  if and only if  $p \leftrightarrow q$  is a tautology

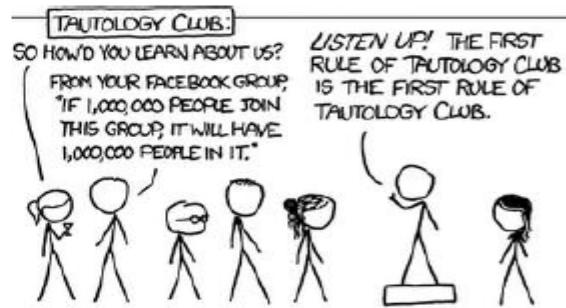
Example:  $p \equiv \neg\neg p$  is a tautology

An expression that always gives a true value is called a **tautology**.

Example:  $p \vee (\neg p) \equiv T$

$p$	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

Always  
true!



Example: Show that  $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$  ← statement

## Contradiction

A statement that is always false is called a **contradiction**.

Example: This course is easy  
'and' this course is not easy

$$p \wedge (\neg p) \equiv F$$



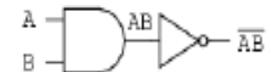
## De Morgan's Law

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

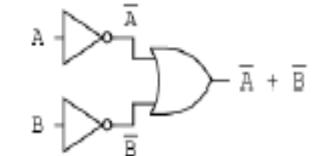
$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$



Augustus De Morgan  
(1806-1871)



... is equivalent to ...



$$\overline{AB} = \overline{A} + \overline{B}$$

p	q	$\neg p$	$\neg q$	$p \wedge q$	$\neg(p \wedge q)$	$\neg p \vee \neg q$
T	T	F	F	T	F	F
T	F	F	T	F	T	T
F	T	T	F	F	T	T
F	F	T	T	F	T	T

## Logical Equivalences

- Useful laws to **transform** one logical expression to an equivalent one.
- **Axioms** ( $T \equiv$  tautology,  $C \equiv$  contradiction):

$$\neg T \equiv F \quad \neg F \equiv T \quad \neg T \equiv C \equiv F \quad \neg C \equiv T \equiv T$$

- De Morgan:

$$\begin{aligned}\neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q\end{aligned}$$

- Commutativity:

$$\begin{aligned}p \wedge q &\equiv q \wedge p \\ p \vee q &\equiv q \vee p\end{aligned}$$



## Logical Equivalence Laws

- **double negation:**  $\neg(\neg p) \equiv p$
- **idempotent:**  $p \wedge p \equiv p$  and  $p \vee p \equiv p$
- **absorption:**  $p \vee (p \wedge q) \equiv p$  and  $p \wedge (p \vee q) \equiv p$

# Quantifiers

## □ Universal quantification

- $(\forall x)P(x)$  means P holds for **all** values of x in domain associated with variable
- E.g.,  $(\forall x) \text{ dolphin}(x) \rightarrow \text{mammal}(x)$

## □ Existential quantification

- $(\exists x)P(x)$  means P holds for **some** value of x in domain associated with variable
- E.g.,  $(\exists x) \text{ mammal}(x) \wedge \text{lays\_eggs}(x)$
- Permits one to make a statement about some object without naming it

# Translating English to FOL

**Every gardener likes the sun**

$$\forall x \text{gardener}(x) \rightarrow \text{likes}(x, \text{Sun})$$

**You can fool some of the people all of the time**

$$\exists x \forall t \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x, t)$$

**You can fool all of the people some of the time**

$$\forall x \exists t (\text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x, t))$$

$$\exists t \forall x (\text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x, t))$$

**All purple mushrooms are poisonous**

$$\forall x (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$$

# Modus Ponens & Modus Tollens

## ► Modus Ponens

1. If A, then B.

2. A

So, 3. B.

### Example:

1. If it is raining, then the ground is wet.

2. It is raining.

So, 3. The ground is wet

## ► Modus Tollens

1. If A, then B.

2. Not B.

So, 3. Not A.

Example:

1. If everything it says in the Bible is true, then the world was created in six days.

2. The world was not created in six days.

So, 3. Therefore, not everything it says in the Bible is true.

# Predicates

- ▶ Terms represent specific objects in the world and can be constants, variables or functions.  
Predicate Symbols refer to a particular relation among objects.
- ▶ Sentences represent facts, and are made of terms, quantifiers and predicate symbols.  
Functions allow us to refer to objects indirectly (via some relationship).
- ▶ Quantifiers and variables allow us to refer to a collection of objects without explicitly naming each object

# Example

- ▶ Predicates: Brother, Sister, Mother , Father
- ▶ Objects: Bill, Hillary, Chelsea, Roger
- ▶ Facts expressed as atomic sentences a.k.a.
- ▶ literals:
  - Father(Bill,Chelsea)
  - Mother(Hillary,Chelsea)
  - Brother(Bill,Roger)

## Unification

We can get the inference immediately if we can find a substitution  $\theta$  such that  $King(x)$  and  $Greedy(x)$  match  $King(John)$  and  $Greedy(y)$ .

$\theta = \{x/John, y/John\}$  works

$\text{UNIFY}(\alpha, \beta) = \theta$  if  $\alpha\theta = \beta\theta$

$p$	$q$	$\theta$
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, SteveJobs)$	$\{x/SteveJobs, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, SteveJobs)$	$fail$

## Standardizing variables apart

- ▶ *Standardizing apart* eliminates overlap of variables.
- ▶ Rename all variables so that variables bound by different quantifiers have unique names.
- ▶ For example

$$\forall x \text{ } Apple(x) \implies \text{ } Fruit(x)$$

$$\forall x \text{ } Spider(x) \implies \text{ } Arachnid(x)$$

is the same as

$$\forall x \text{ } Apple(x) \implies \text{ } Fruit(x)$$

$$\forall y \text{ } Spider(y) \implies \text{ } Arachnid(y)$$

## Resolution

Full first-order version:

$$\frac{\ell_1 \vee \dots \vee \ell_k, \quad m_1 \vee \dots \vee m_n}{(\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)\theta}$$

where  $\text{UNIFY}(\ell_i, \neg m_j) = \theta$ .

For example,

$$\begin{array}{c} \neg \text{Rich}(x) \vee \text{Unhappy}(x) \\ \text{Rich}(\text{Ken}) \\ \hline \end{array}$$

$\text{Unhappy}(\text{Ken})$

with  $\theta = \{x/\text{Ken}\}$ .

# Conjunctive Normal Form (CNF)

- Resolution works best when the formula is of the special form: it is an  $\wedge$  of  $\vee$ s of (possibly negated,  $\neg$ ) variables (called literals).
- This form is called a **Conjunctive Normal Form**, or **CNF**.
  - $(y \vee \neg z) \wedge (\neg y) \wedge (y \vee z)$  is a CNF
  - $(x \vee \neg y \wedge z)$  is not a CNF
- An AND ( $\wedge$ ) of CNF formulas is a CNF formula.
  - So if all premises are CNF and the negation of the conclusion is a CNF, then AND of premises AND NOT conclusion is a CNF.

- To convert a formula into a CNF.
  - Open up the implications to get ORs.
  - Get rid of double negations.
  - Convert  $F \vee (G \wedge H)$  to  $(F \vee G) \wedge (F \vee H)$   
*//distributivity*
- Example:  $A \rightarrow (B \wedge C)$   
 $\equiv \neg A \vee (B \wedge C)$   
 $\equiv (\neg A \vee B) \wedge (\neg A \vee C)$
- In general, CNF can become quite big, especially when have  $\leftrightarrow$ . There are tricks to avoid that ...

## CNF and DNF

- Every truth table (Boolean function) can be written as either a conjunctive normal form (CNF) or disjunctive normal form (DNF)
- **CNF is an  $\wedge$  of  $\vee$ s**, where  $\vee$  is over variables or their negations (literals); an  $\vee$  of literals is also called a **clause**.
- **DNF is an  $\vee$  of  $\wedge$ s**; an  $\wedge$  of literals is called a **term**.

# Why CNF and DNF?

- Convenient normal forms
- **Resolution** works best for formulas in CNF
- Useful for constructing formulas given a **truth table**
  - DNF: take a disjunction (that is,  $\vee$ ) of all **satisfying** truth assignments
  - CNF: take a conjunction ( $\wedge$ ) of **negations** of **falsifying** truth assignments

- Any propositional formula is tautologically equivalent to some formula in disjunctive normal form.
- Any propositional formula is tautologically equivalent to some formula in conjunctive normal form.

## Conversion to CNF

1. Eliminate biconditionals and implications.
2. Reduce the scope of  $\neg$ : move  $\neg$  inwards.
3. Standardize variables apart: each quantifier should use a different variable name.
4. Skolemize: a more general form of existential instantiation.  
Each existential variable is replaced by a *Skolem function* of the enclosing universally quantified variables.
5. Drop all universal quantifiers: It's allright to do so now.
6. Distribute  $\wedge$  over  $\vee$ .
7. Make each conjunct a separate clause.
8. Standardize the variables apart again.

► All people who are graduating are happy.

All happy people smile.

JohnDoe is graduating.

Is JohnDoe smiling?

► First convert to predicate logic

$\forall x \text{graduating}(x) \Rightarrow \text{happy}(x)$

$\forall x \text{happy}(x) \Rightarrow \text{smiling}(x)$

$\text{graduating}(\text{JohnDoe})$

$\text{smiling}(\text{JohnDoe})$       negate this:  $\neg \text{smiling}(\text{JohnDoe})$

► Then convert to canonical form.

1.  $\forall x \text{graduating}(x) \Rightarrow \text{happy}(x)$

2.  $\forall x \text{happy}(x) \Rightarrow \text{smiling}(x)$

3.  $\text{graduating}(\text{JohnDoe})$

4.  $\neg \text{smiling}(\text{JohnDoe})$

Step 1. Eliminate  $\Rightarrow$

1.  $\forall x \neg \text{graduating}(x) \vee \text{happy}(x)$

2.  $\forall x \neg \text{happy}(x) \vee \text{smiling}(x)$

3.  $\text{graduating}(\text{JohnDoe})$

4.  $\neg \text{smiling}(\text{JohnDoe})$

1.  $\forall x \neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\forall x \neg\text{happy}(x) \vee \text{smiling}(x)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

Step 2. Move  $\neg$  inwards. (not needed)

Step 3. Standardize variables apart.

1.  $\forall x \neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\forall y \neg\text{happy}(y) \vee \text{smiling}(y)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

1.  $\forall x \neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\forall y \neg\text{happy}(y) \vee \text{smiling}(y)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

Step 4. Skolemize. (not needed)

Step 5. Drop all  $\forall$ .

1.  $\neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\neg\text{happy}(y) \vee \text{smiling}(y)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

1.  $\neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\neg\text{happy}(y) \vee \text{smiling}(y)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

Step 6. Distribute  $\wedge$  over  $\vee$ . (not needed)

Step 7. Make each conjunct a separate clause. (not needed)

Step 8. Standardize the variables apart again. (not needed)

Ready for resolution!

1.  $\neg\text{graduating}(x) \vee \text{happy}(x)$
2.  $\neg\text{happy}(y) \vee \text{smiling}(y)$
3.  $\text{graduating}(\text{JohnDoe})$
4.  $\neg\text{smiling}(\text{JohnDoe})$

Resolve 4 and 2 using  $\theta = \{y / \text{JohnDoe}\}$ :

5.  $\neg\text{happy}(\text{JohnDoe})$

Resolve 5 and 1 using  $\theta = \{x / \text{JohnDoe}\}$ :

6.  $\neg\text{graduating}(\text{JohnDoe})$

Resolve 6 and 3:

7.  $\perp$

# Resolution

- ❑ Resolution is a **sound** and **complete** inference procedure for FOL
- ❑ Reminder: Resolution rule for propositional logic:
  - ❑  $P_1 \vee P_2 \vee \dots \vee P_n$
  - ❑  $\neg P_1 \vee Q_2 \vee \dots \vee Q_m$
  - ❑ Resolvent:  $P_2 \vee \dots \vee P_n \vee Q_2 \vee \dots \vee Q_m$
- ❑ Examples
  - ❑  $P$  and  $\neg P \vee Q$  : derive  $Q$  (Modus Ponens)
  - ❑  $(\neg P \vee Q)$  and  $(\neg Q \vee R)$  : derive  $\neg P \vee R$
  - ❑  $P$  and  $\neg P$  : derive False [contradiction!]
  - ❑  $(P \vee Q)$  and  $(\neg P \vee \neg Q)$  : derive True

# Resolution in first-order logic

- Given sentences

$$P_1 \vee \dots \vee P_n$$

$$Q_1 \vee \dots \vee Q_m$$

- in *conjunctive normal form*:

- each  $P_i$  and  $Q_i$  is a literal, i.e., a positive or negated predicate symbol with its terms,

- if  $P_j$  and  $\neg Q_k$  unify with substitution list  $\theta$ , then derive the resolvent sentence:

$$\text{subst}(\theta, P_1 \vee \dots \vee P_{j-1} \vee P_{j+1} \dots P_n \vee Q_1 \vee \dots \vee Q_{k-1} \vee Q_{k+1} \vee \dots \vee Q_m)$$

- Example

- from clause  $P(x, f(a)) \vee P(x, f(y)) \vee Q(y)$

- and clause  $\neg P(z, f(a)) \vee \neg Q(z)$

- derive resolvent  $P(z, f(y)) \vee Q(y) \vee \neg Q(z)$

- using  $\theta = \{x/z\}$

Example:

Assume:  $E_1 \vee E_2$  playing tennis or raining

and  $\neg E_2 \vee E_3$  not raining or working

---

Then:  $E_1 \vee E_3$  playing tennis or working

"Resolvent"

General Rule:

Assume:  $E \vee E_{12} \vee \dots \vee E_{1k}$

and  $\neg E \vee E_{22} \vee \dots \vee E_{2l}$

---

Then:  $E_{12} \vee \dots \vee E_{1k} \vee E_{22} \vee \dots \vee E_{2l}$

Note:  $E_{ij}$  can be negated.

# Resolution refutation

- Given a consistent set of axioms  $\text{KB}$  and goal sentence  $Q$ , show that  $\text{KB} \models Q$
- Proof by contradiction: Add  $\neg Q$  to  $\text{KB}$  and try to prove false.  
i.e.,  $(\text{KB} \dashv Q) \leftrightarrow (\text{KB} \wedge \neg Q \dashv \text{False})$
- Resolution is **refutation complete**: it can establish that a given sentence  $Q$  is entailed by  $\text{KB}$ , but can't (in general) be used to generate all logical consequences of a set of sentences

## Resolution example

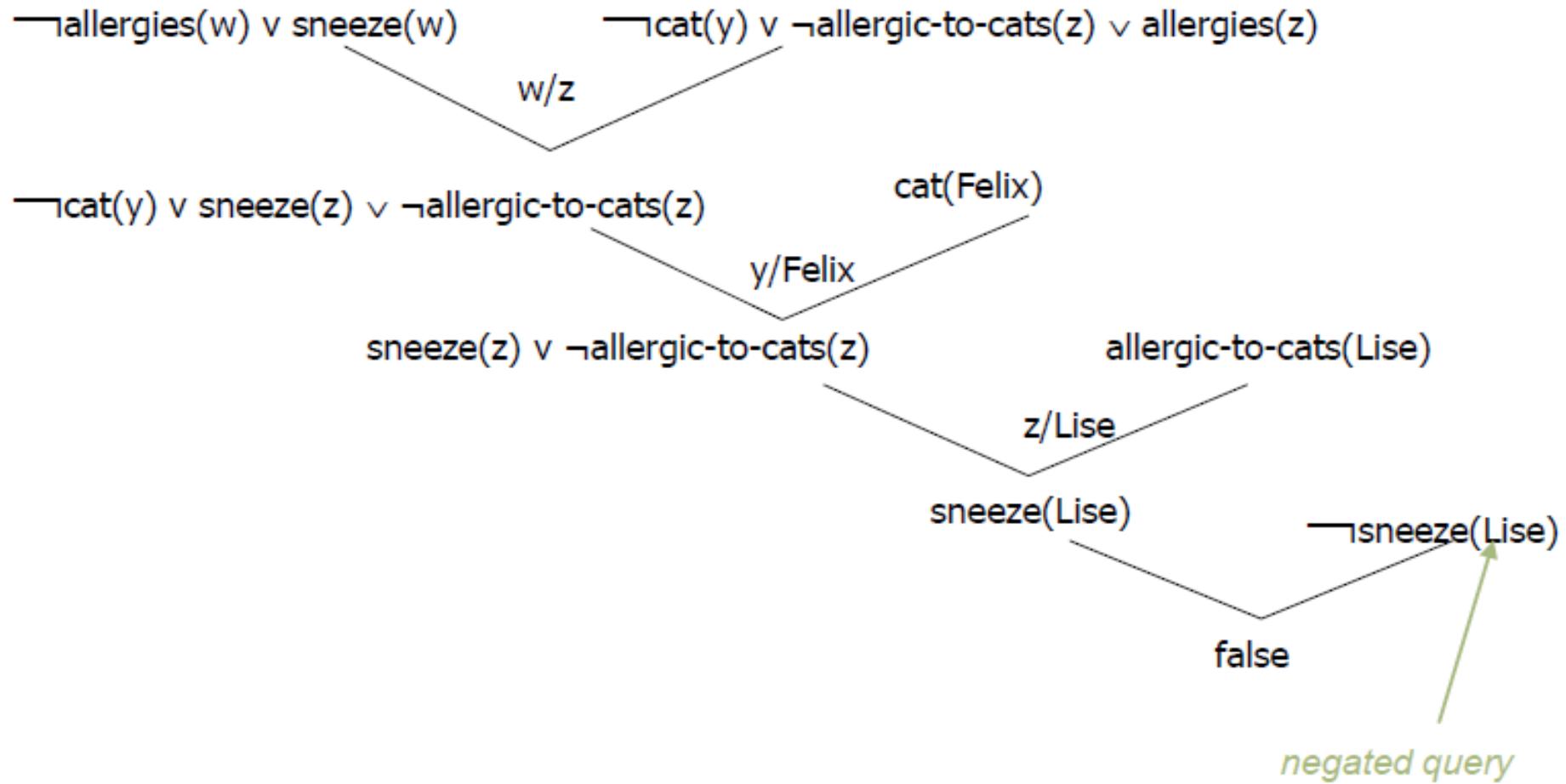
❑ KB:

- ❑ allergies(X) → sneeze(X)
- ❑ cat(Y) ∧ allergic-to-cats(X) → allergies(X)
- ❑ cat(Felix)
- ❑ allergic-to-cats(Lise)

❑ Goal:

- ❑ sneeze(Lise)

# Refutation resolution proof tree



# Skolemization

# FOL: Conversion to CNF

- Everyone who loves all animals is loved by someone

$$\forall x [\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{Loves}(y, x)]$$

- Steps to convert to CNF

- Eliminate biconditionals  $\Leftrightarrow$  and implications  $\Rightarrow$
- Move  $\neg$  inwards  $\neg\forall x, p \equiv \exists x \neg p, \neg\exists x p \equiv \forall x \neg p$
- Standardize the variables: each quantifier should use a different variable
- Skolemize: A more general form of existential instantiation
  - Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables
- Drop universal quantifiers
- Distribute  $\wedge$  over  $\vee$

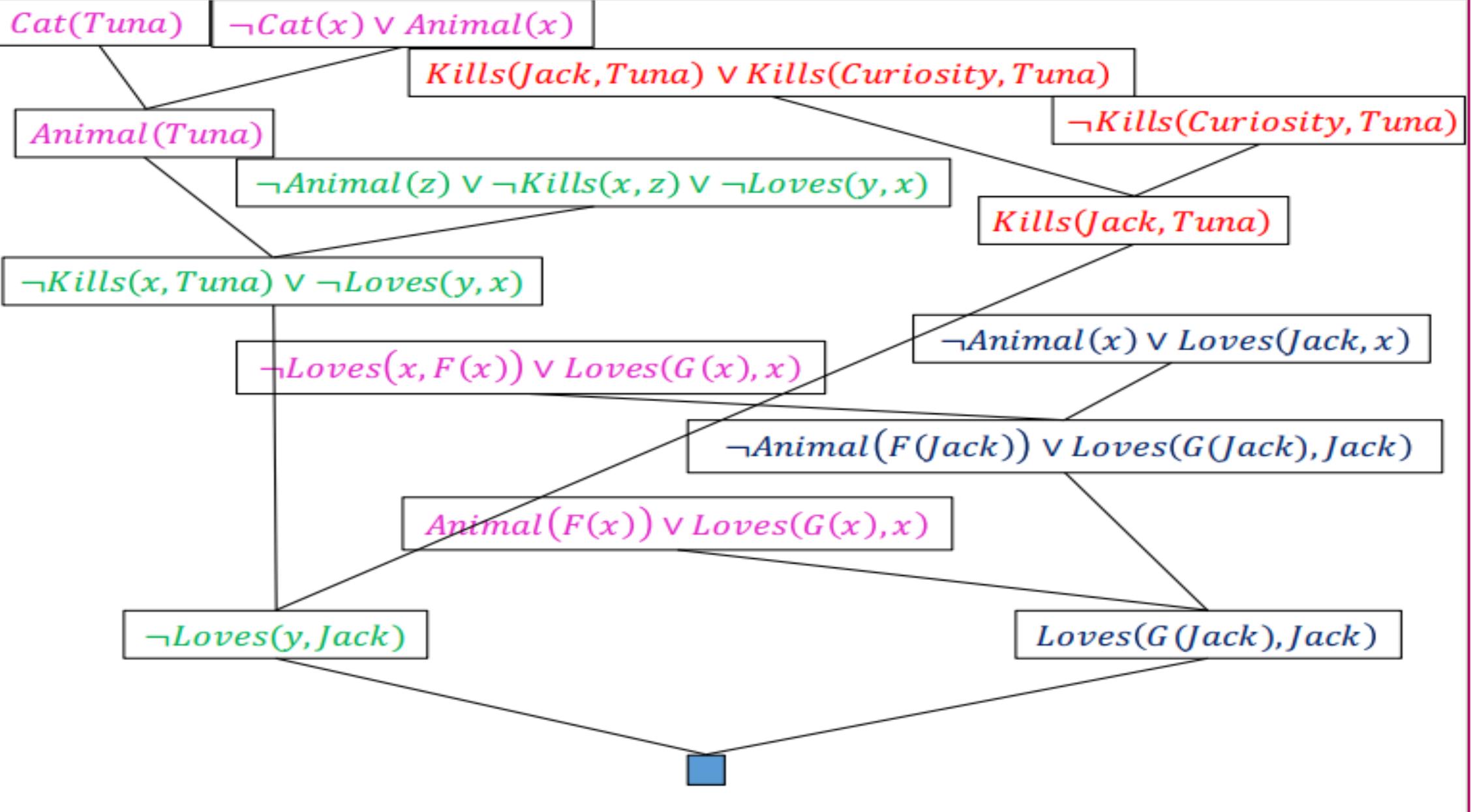
# FOL Resolution Example

- Everyone who loves all animals is loved by someone  $\forall x[\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{Loves}(y, x)]$
- Anyone who kills an animal is loved by no one  $\forall x[\exists z \text{Animal}(z) \wedge \text{Kills}(x, z)] \Rightarrow [\forall y \neg \text{Loves}(y, x)]$
- Jack loves all animals  $\forall x \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$
- Either Jack or Curiosity killed the cat, who is named Tuna  $\text{Kills}(\text{Jack}, \text{Tuna}) \vee \text{Kills}(\text{Curiosity}, \text{Tuna})$
- Did Curiosity kill the cat?
  - $\text{Kills}(\text{Curiosity}, \text{Tuna})?$
  - $\forall x \text{Cat}(x) \Rightarrow \text{Animal}(x)$
  - $\text{Cat}(\text{Tuna})$

$$\forall x [\forall y \text{Animal}(y) \Rightarrow \text{Loves}(x, y)] \Rightarrow [\exists y \text{Loves}(y, x)]$$
$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x, y)] \vee [\exists y \text{Loves}(y, x)]$$
$$\forall x [\exists y \neg (\neg \text{Animal}(y) \vee \text{Loves}(x, y))] \vee [\exists y \text{Loves}(y, x)]$$
$$\forall x [\exists y (\text{Animal}(y) \wedge \neg \text{Loves}(x, y))] \vee [\exists y \text{Loves}(y, x)]$$
$$\forall x [\exists y (\text{Animal}(y) \wedge \neg \text{Loves}(x, y))] \vee [\exists z \text{Loves}(z, x)]$$
$$\forall x [\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$
$$[\text{Animal}(F(x)) \wedge \neg \text{Loves}(x, F(x))] \vee \text{Loves}(G(x), x)$$
$$[\text{Animal}(F(x)) \vee \text{Loves}(G(x), x)] \wedge [\neg \text{Loves}(x, F(x)) \vee \text{Loves}(G(x), x)]$$

$$\forall x [\exists z Animal(z) \wedge Kills(x, z)] \Rightarrow [\forall y \neg Loves(y, x)]$$
$$\forall x [\neg \exists z Animal(z) \wedge Kills(x, z)] \vee [\forall y \neg Loves(y, x)]$$
$$\forall x [\forall z \neg (Animal(z) \wedge Kills(x, z))] \vee [\forall y \neg Loves(y, x)]$$
$$\forall x [\forall z \neg Animal(z) \vee \neg Kills(x, z)] \vee [\forall y \neg Loves(y, x)]$$
$$\neg Animal(z) \vee \neg Kills(x, z) \vee \neg Loves(y, x)$$

$$\forall x \text{Animal}(x) \Rightarrow \text{Loves}(\text{Jack}, x)$$
$$\forall x \text{Cat}(x) \Rightarrow \text{Animal}(x)$$
$$\forall x \neg \text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)$$
$$\forall x \neg \text{Cat}(x) \vee \text{Animal}(x)$$
$$\neg \text{Animal}(x) \vee \text{Loves}(\text{Jack}, x)$$
$$\neg \text{Cat}(x) \vee \text{Animal}(x)$$



# Example

Jack owns a dog.

Every dog owner is an animal lover.

No animal lover kills an animal.

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

Did Curiosity kill the cat?

1.  $\exists x : Dog(x) \wedge Owns(Jack, x)$
2.  $\forall x; (\exists y \ Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$
3.  $\forall x; AnimalLover(x) \rightarrow (\forall y \ Animal(y) \rightarrow \neg Kills(x, y))$
4.  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
5.  $Cat(Tuna)$
6.  $\forall x : Cat(x) \rightarrow Animal(x)$

1.  $\exists x : Dog(x) \wedge Owns(Jack, x)$

2.  $\forall x; (\exists y Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$

3.  $\forall x; AnimalLover(x) \rightarrow (\forall y Animal(y) \rightarrow \neg Kills(x, y))$

4.  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

5.  $Cat(Tuna)$

6.  $\forall x : Cat(x) \rightarrow Animal(x)$

## Conjunctive Normal Form

$Dog(D)$

(D is a placeholder for the dogs unknown name  
(i.e. Skolem symbol/function). Think of D like  
"JohnDoe")

$Owns(Jack, D)$

$\neg Dog(y) \vee \neg Owns(x, y) \vee AnimalLover(x)$

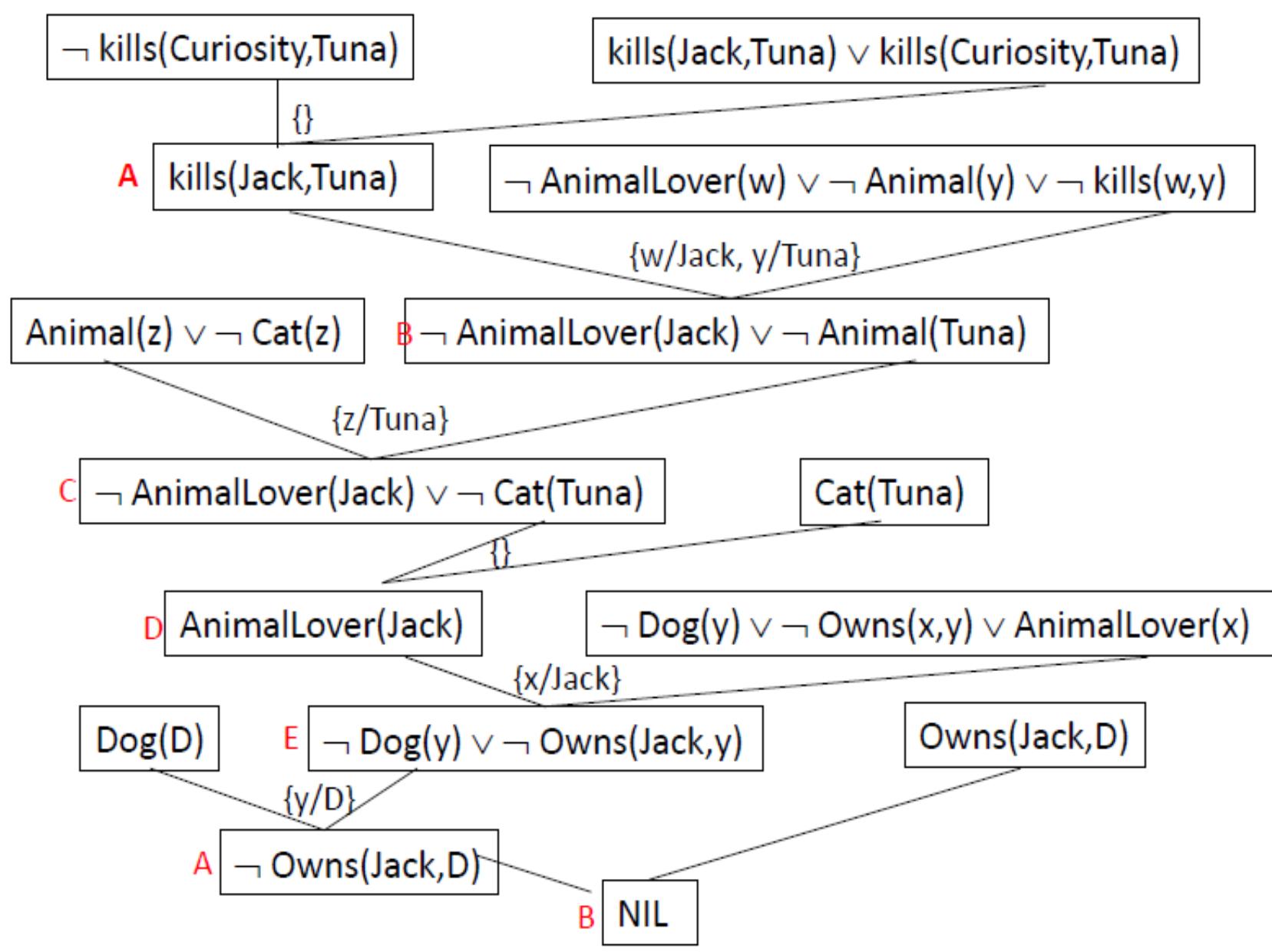
$\neg AnimalLover(w) \vee \neg Animal(y) \vee \neg Kills(w, y)$

$Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

$Cat(Tuna)$

$\neg Cat(z) \vee Animal(z)$

$\neg Kills(Curiosity, Tuna)$



## Conjunctive Normal Form

$\text{Dog}(D)$

(D is a placeholder for the dog's unknown name (i.e. Skolem symbol/function). Think of D like "JohnDoe")

$\text{Owns}(\text{Jack}, D)$

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

$\neg \text{AnimalLover}(w) \vee \neg \text{Animal}(y) \vee \neg \text{Kills}(w,y)$

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

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

$\neg \text{Cat}(z) \vee \text{Animal}(z)$

$\neg \text{Kills}(\text{Curiosity}, \text{Tuna})$

# Assignment

## Knowledge Base in a Natural Language

- ▶ 1. Marcus was a man.
- ▶ 2. Marcus was a Pompeian.
- ▶ 3. All Pompeians were Roman.
- ▶ 4. Caesar was a ruler.
- ▶ 5. All Romans were either loyal to Caesar or hate Caesar.
- ▶ 6. Everyone is loyal to someone.
- ▶ 7. People only try to assassinate rulers to whom they are not loyal.
- ▶ 8. Marcus tried to assassinate Caesar.
- ▶ Query: Does Marcus hate Caesar?

1. Marcus was a man.

- ▶ man(Marcus)

2. Marcus was a Pompeian

- ▶ pompeian(Marcus)

3. All Pompeians were Romans.

- ▶  $\forall x: \text{pompeian}(x) \rightarrow \text{roman}(x)$
- ▶  $\neg \text{pompeian}(x) \vee \text{roman}(x)$

4. Caesar was a ruler.

- ▶ ruler(Caesar)

5. All Romans were either loyal to Caesar or hated him.

- ▶  $\forall x: \text{roman}(x) \rightarrow (\text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar}))$
- ▶  $\neg \text{roman}(y) \vee \text{loyalto}(y, \text{Caesar}) \vee \text{hate}(y, \text{Caesar})$

6. Everyone is loyal to someone.

- ▶  $\forall x \exists y: \text{loyalto}(x, y)$
- ▶  $\text{loyalto}(z, f(z))$  //  $f$  is a skolem function and returns a person that  $z$  is loyal to

7. People only try to assassinate rulers they aren't loyal to.

- ▶  $\forall x \forall y: \text{man}(x) \wedge \text{ruler}(y) \wedge \neg \text{loyalto}(x, y) \rightarrow \text{trytoassassinate}(x, y)$
- ▶  $\neg(\text{man}(a) \wedge \text{ruler}(b) \wedge \neg \text{loyalto}(a, b)) \vee \text{trytoassassinate}(a, b)$
- ▶  $= \neg \text{man}(a) \vee \neg \text{ruler}(b) \vee \neg \text{loyalto}(a, b) \vee \text{trytoassassinate}(a, b)$

8. Marcus tried to assassinate Caesar.

- ▶  $\text{trytoassassinate}(\text{Marcus}, \text{Caesar})$

Introduce 9.  $\sim \text{hated}(\text{Marcus}, \text{Caesar})$ .

Resolve 9 with 5 unifying Marcus to y yields

10.  $\sim \text{roman}(\text{Marcus}) \vee \text{loyalto}(\text{Marcus}, \text{Caesar})$

Resolve 10 with 3 unifying Marcus to x yields

11.  $\sim \text{pompeian}(\text{Marcus}) \vee \text{loyalto}(\text{Marcus}, \text{Caesar})$

Resolve 11 with 2 yields

12.  $\text{loyalto}(\text{Marcus}, \text{Caesar})$

Resolve 12 with 7 unifying Marcus to a and Caesar to b yields

13.  $\sim \text{man}(\text{Marcus}) \vee \sim \text{ruler}(\text{Caesar}) \vee \text{trytoassassinate}(\text{Marcus}, \text{Caesar})$

Resolve 13 with 1 yields

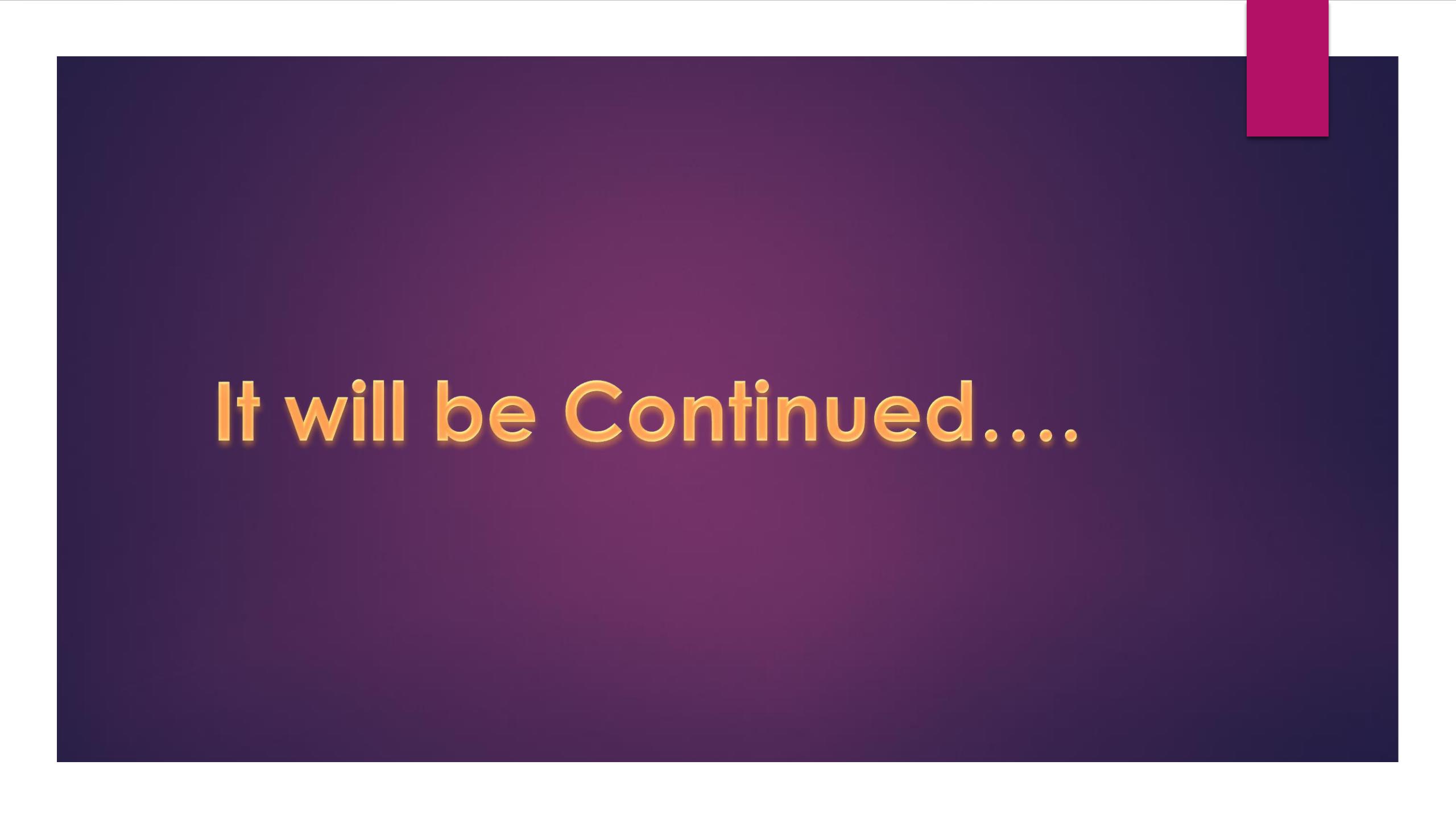
14.  $\sim \text{ruler}(\text{Caesar}) \vee \text{trytoassassinate}(\text{Marcus}, \text{Caesar})$

Resolve 14 with 4 yields

15.  $\text{trytoassassinate}(\text{Marcus}, \text{Caesar})$

Resolve 15 with 8 yields the null hypothesis.

Therefore,  $\sim \text{hate}(\text{Marcus}, \text{Caesar})$  is false, so  $\text{hate}(\text{Marcus}, \text{Caesar})$  is true.



**It will be Continued....**

# Artificial Neural Network----- Back-propagation learning

**Supervised Learning**

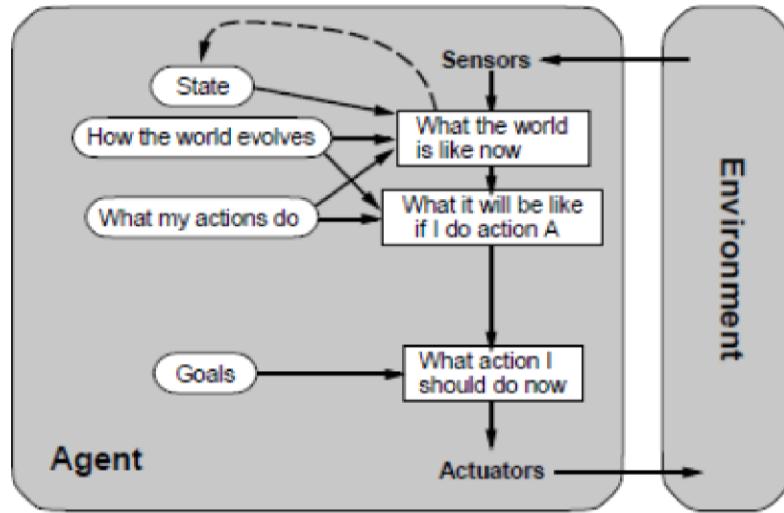
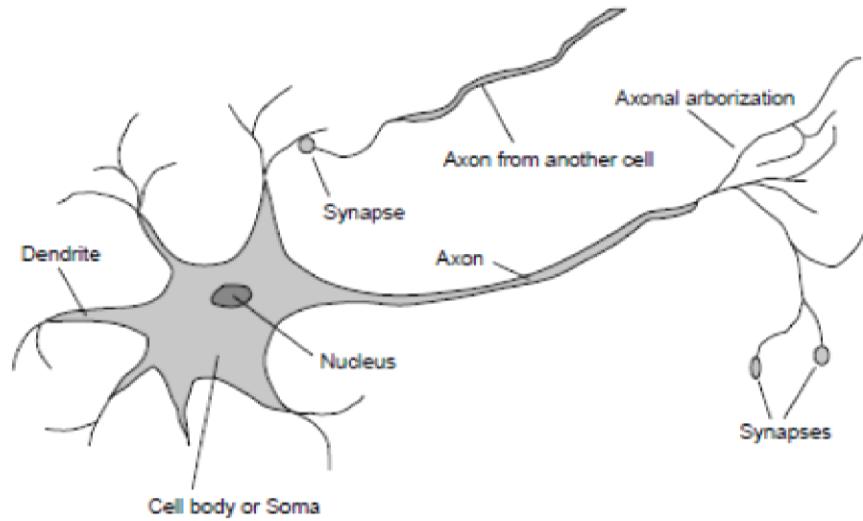
BY

DR. ANUPAM GHOSH

10<sup>TH</sup> OCT, 2023

# History of AI

mid-80's



- ▶ NN are back
- ▶ Statistical approaches rise
- ▶ AI is a science (formalization, specialization, complexity, ...)
- ▶ From mid 80's to mid 90's : **AI winter**
  - ▶ Over-optimistic promises?
  - ▶ Funding agencies (public and private) have expected too much
- ▶ Since mid 90's: unifying approach « intelligent agent »

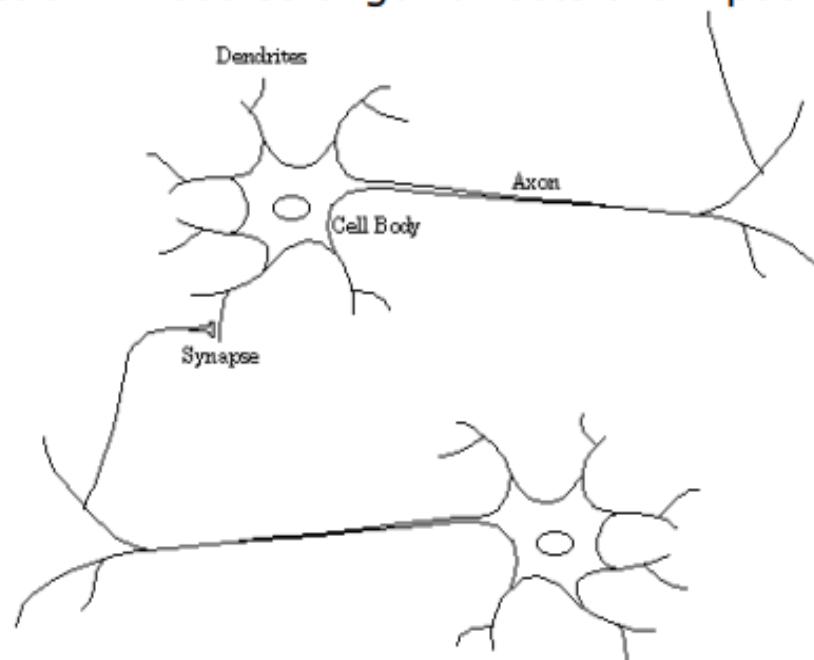
## Biological Inspirations

Humans perform complex tasks like vision, motor control, or language understanding very well.

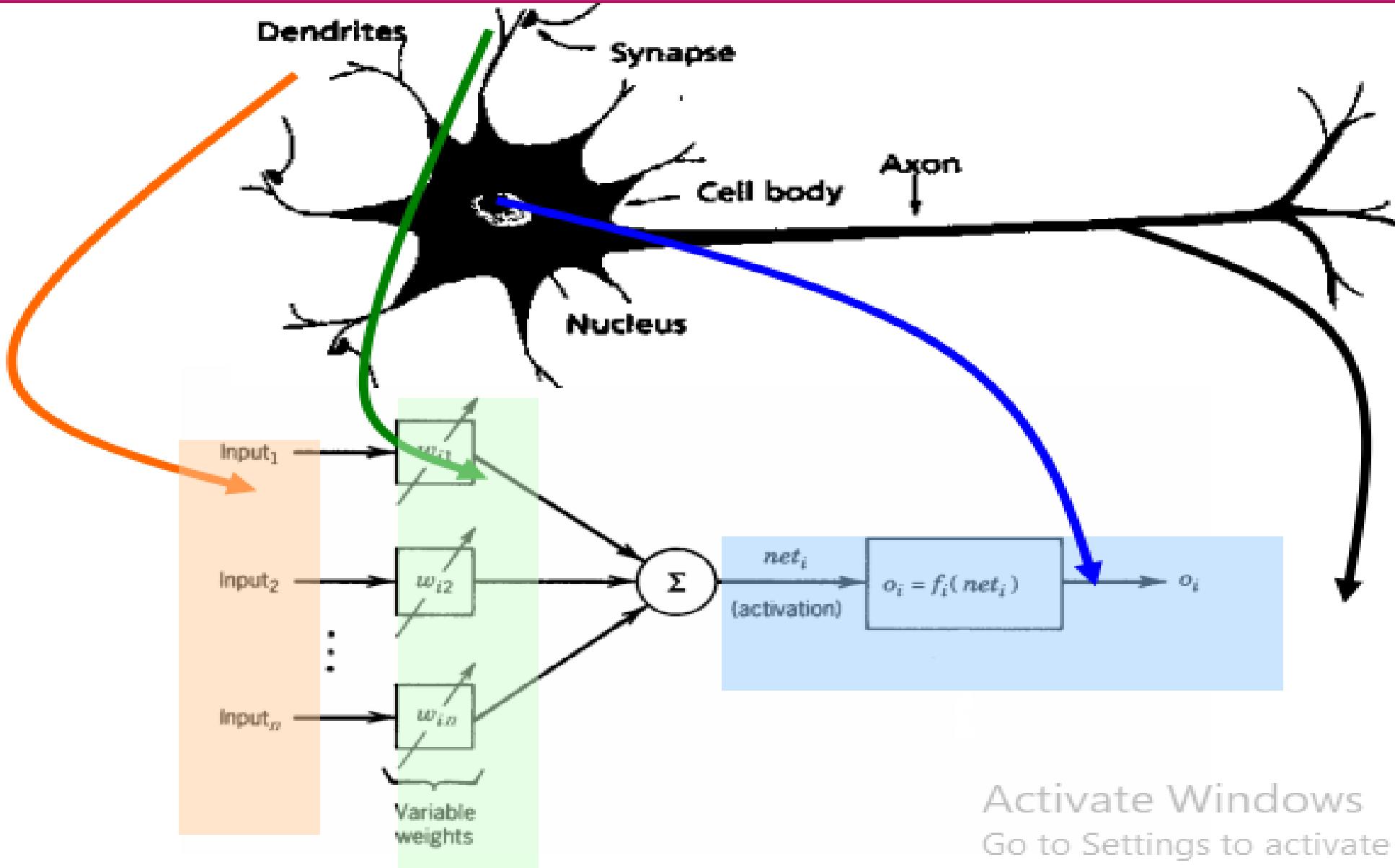
One way to build intelligent machines is to try to imitate the (organizational principles of) human brain.

## Biological Neuron

- **dendrites:** nerve fibres carrying electrical signals to the cell
- **cell body:** computes a non-linear function of its inputs
- **axon:** single long fiber that carries the electrical signal from the cell body to other neurons
- **synapse:** the point of contact between the axon of one cell and the dendrite of another, regulating a chemical connection whose strength affects the input to the cell.



Activate Window  
Go to Settings to ad



# Artificial Neural Networks

Computational models inspired by the human brain:

- Massively parallel, distributed system, made up of simple processing units (neurons)
- Synaptic connection strengths among neurons are used to store the acquired knowledge.
- Knowledge is acquired by the network from its environment through a learning process

# Properties of ANNs

## Learning from examples

- labeled or unlabeled

## Adaptivity

- changing the connection strengths to learn things

## Non-linearity

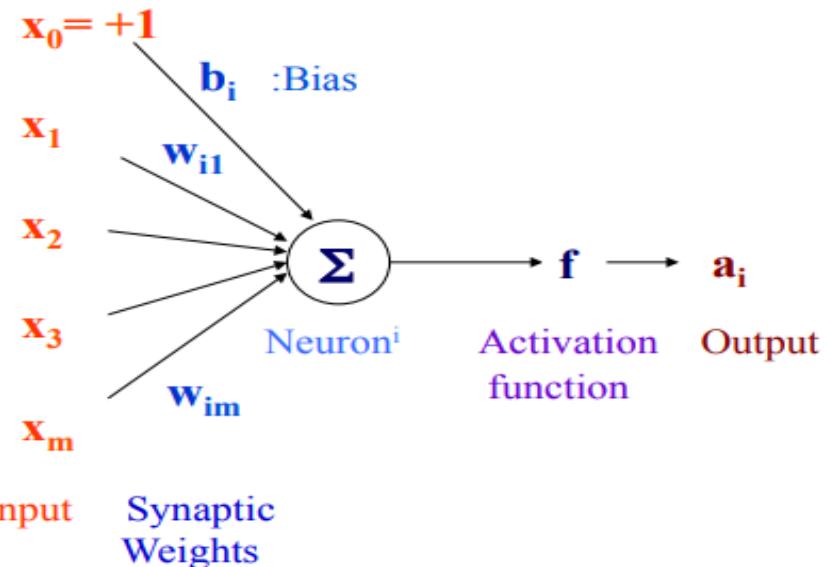
- the non-linear activation functions are essential

## Fault tolerance

- if one of the neurons or connections is damaged, the whole network still works quite well

Thus, they might be better alternatives than classical solutions for problems characterised by:

- high dimensionality, noisy, imprecise or imperfect data; and
- a lack of a clearly stated mathematical solution or algorithm



## Bias

$$a_i = f(n_i) = f(\sum_{j=1}^n w_{ij}x_j + b_i)$$

An artificial neuron:

- computes the weighted sum of its input (called its **net input**)
- adds its bias
- passes this value through an activation function

We say that the neuron "fires" (i.e. becomes active) if its output is above zero.

# *Activation Functions*

- Identity

$$f(x) = x$$

- Binary step

$$f(x) = 1 \text{ if } x \geq \theta$$

$$f(x) = 0 \text{ otherwise}$$

- Binary sigmoid

$$f(x) = 1 / (1 + e^{-\sigma x})$$

- Bipolar sigmoid

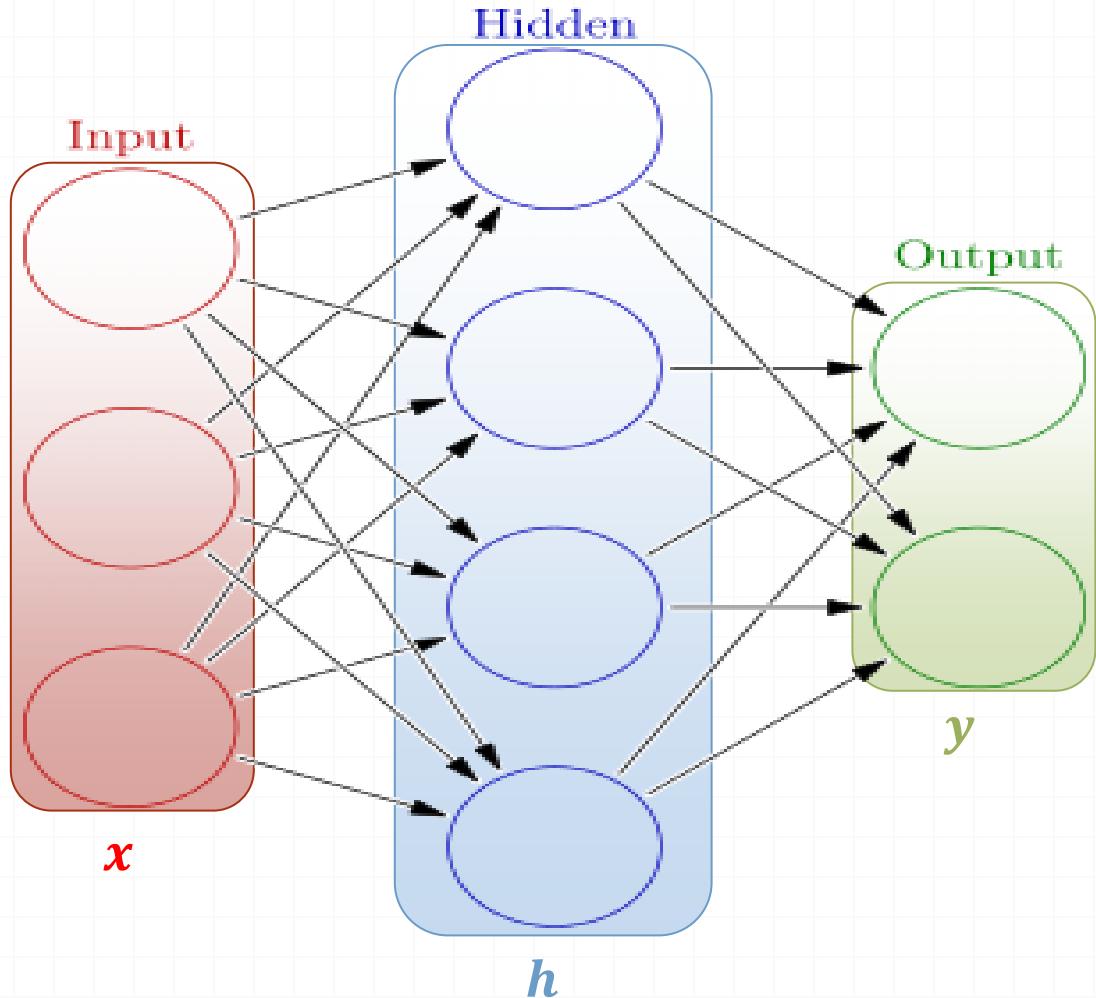
$$f(x) = -1 + 2 / (1 + e^{-\sigma x})$$

- Hyperbolic tangent

$$f(x) = (e^x - e^{-x}) / (e^x + e^{-x})$$

Activate Windows  
Go to Settings to activate

# Neural Network Intro



Demo

Weights

$$h = \sigma(W_1x + b_1)$$
$$y = \sigma(W_2h + b_2)$$

Activation functions

How do we train?

$4 + 2 = 6$  neurons (not counting inputs)

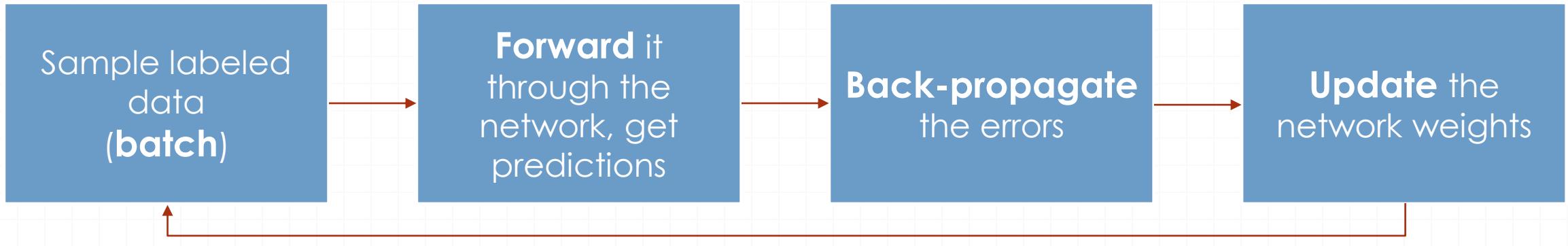
$[3 \times 4] + [4 \times 2] = 20$  weights

$4 + 2 = 6$  biases

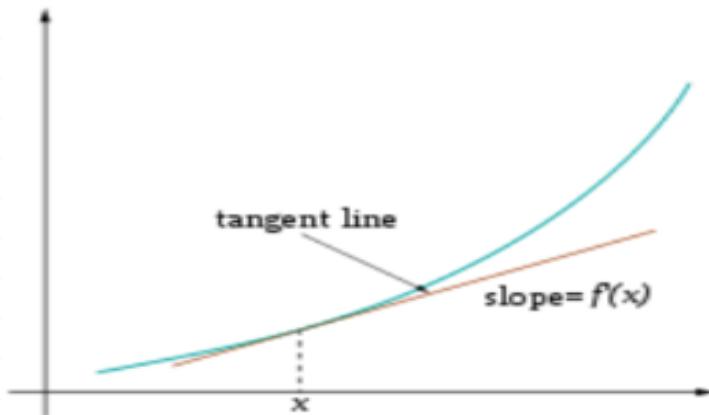
---

26 learnable **parameters**

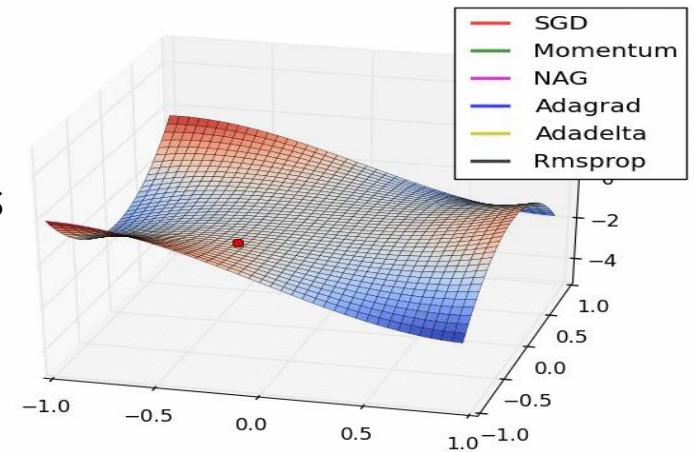
# Training



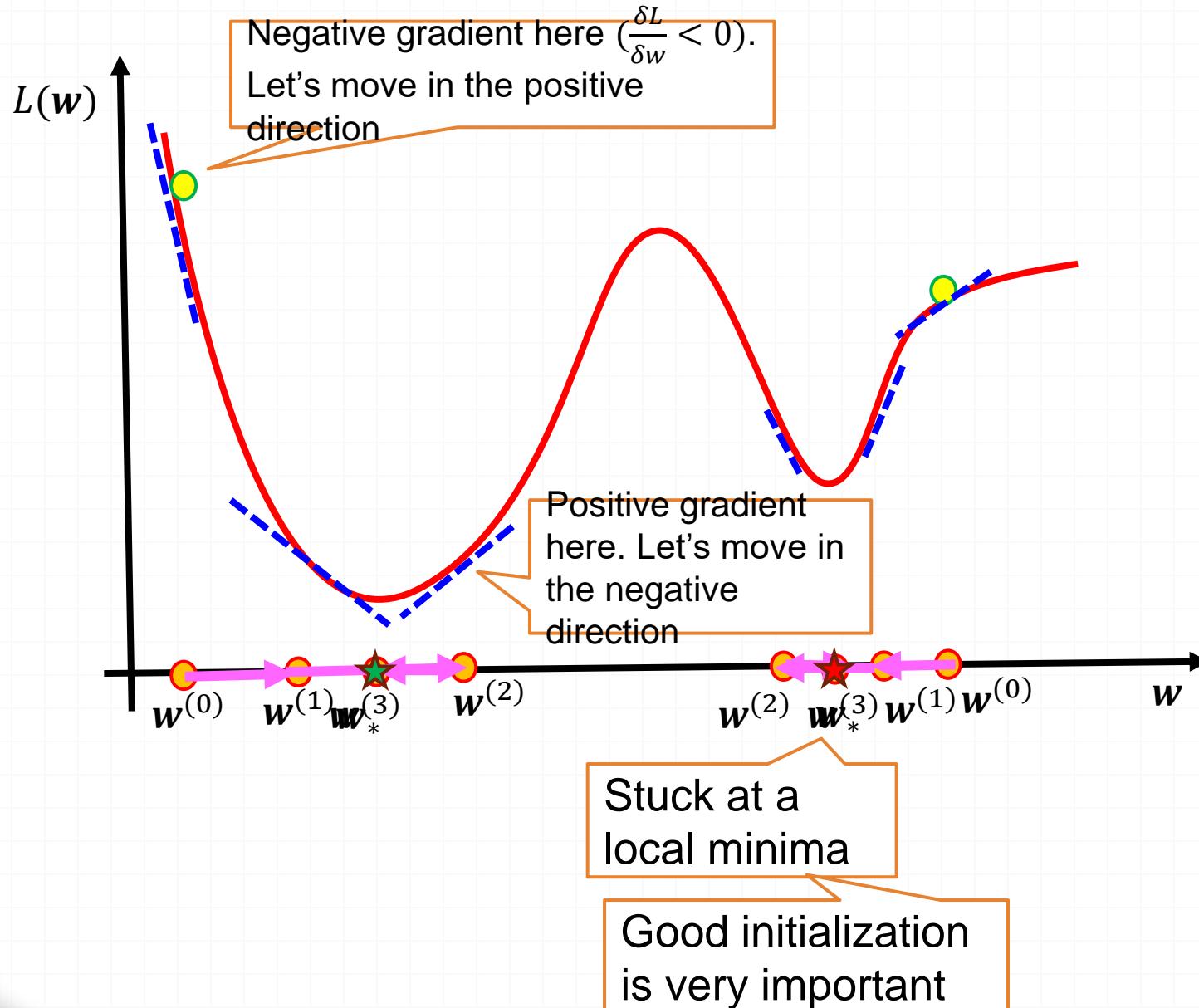
Optimize (min. or max.) **objective/cost function  $J(\theta)$**   
Generate **error signal** that measures difference between predictions and target values



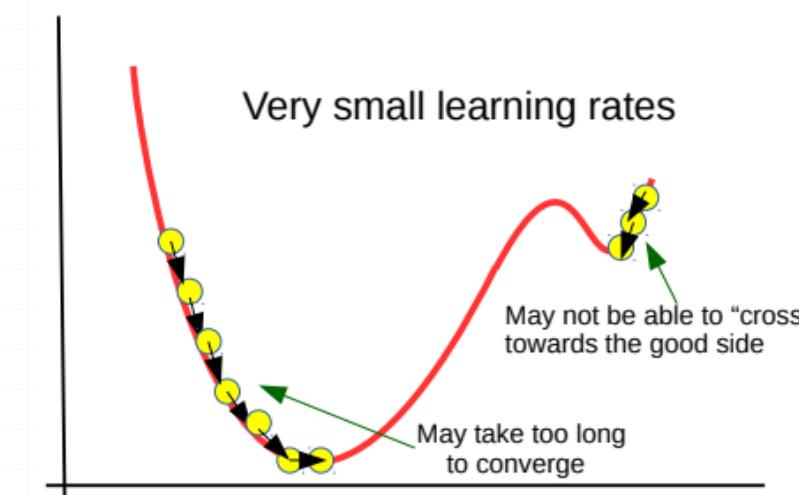
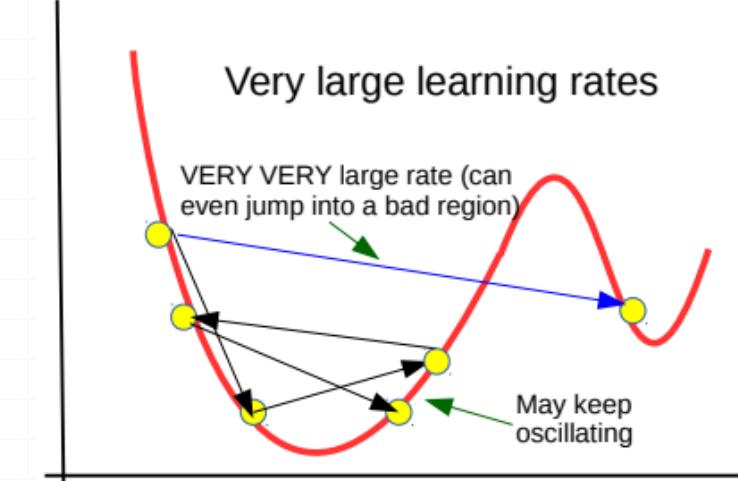
Use error signal to change the **weights** and get more accurate predictions  
Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**



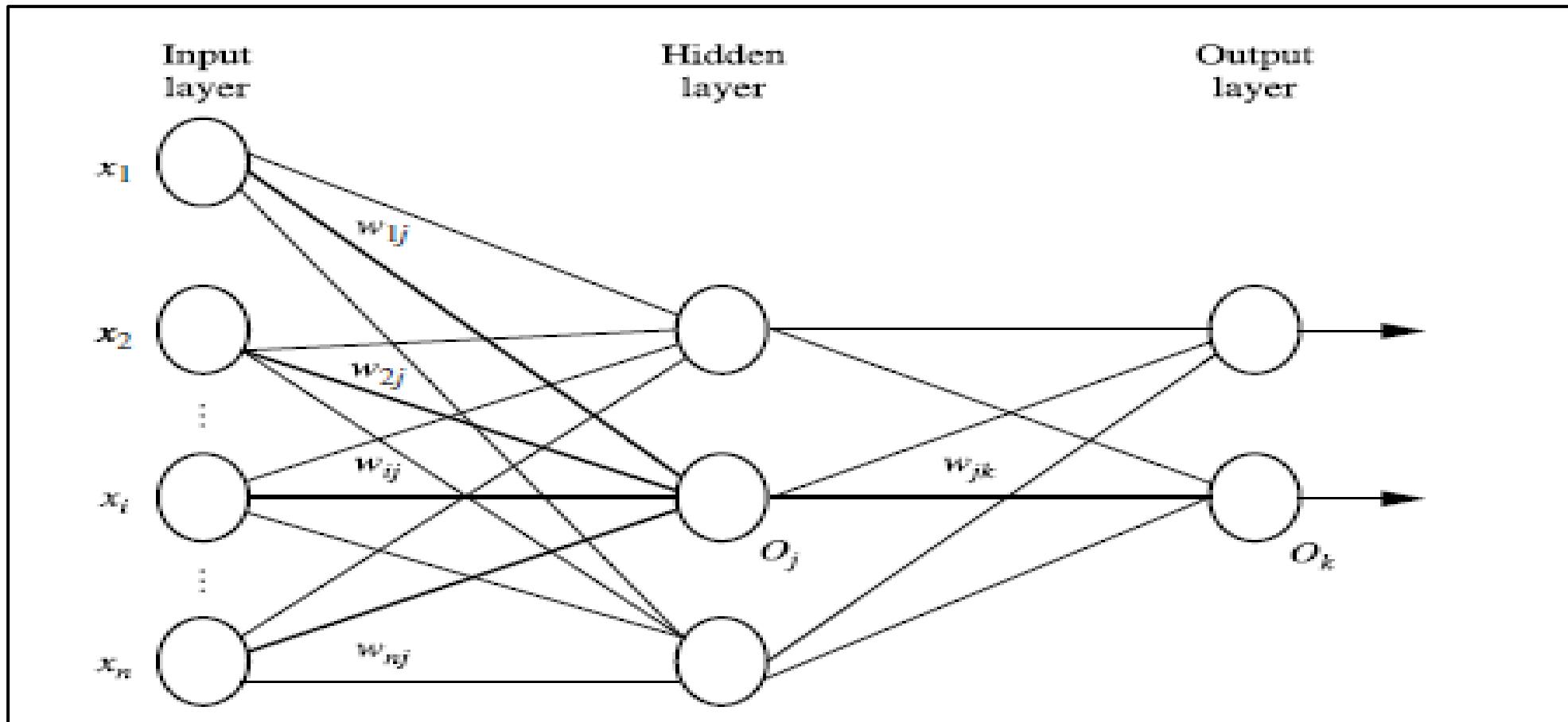
# Gradient Descent: An Illustration



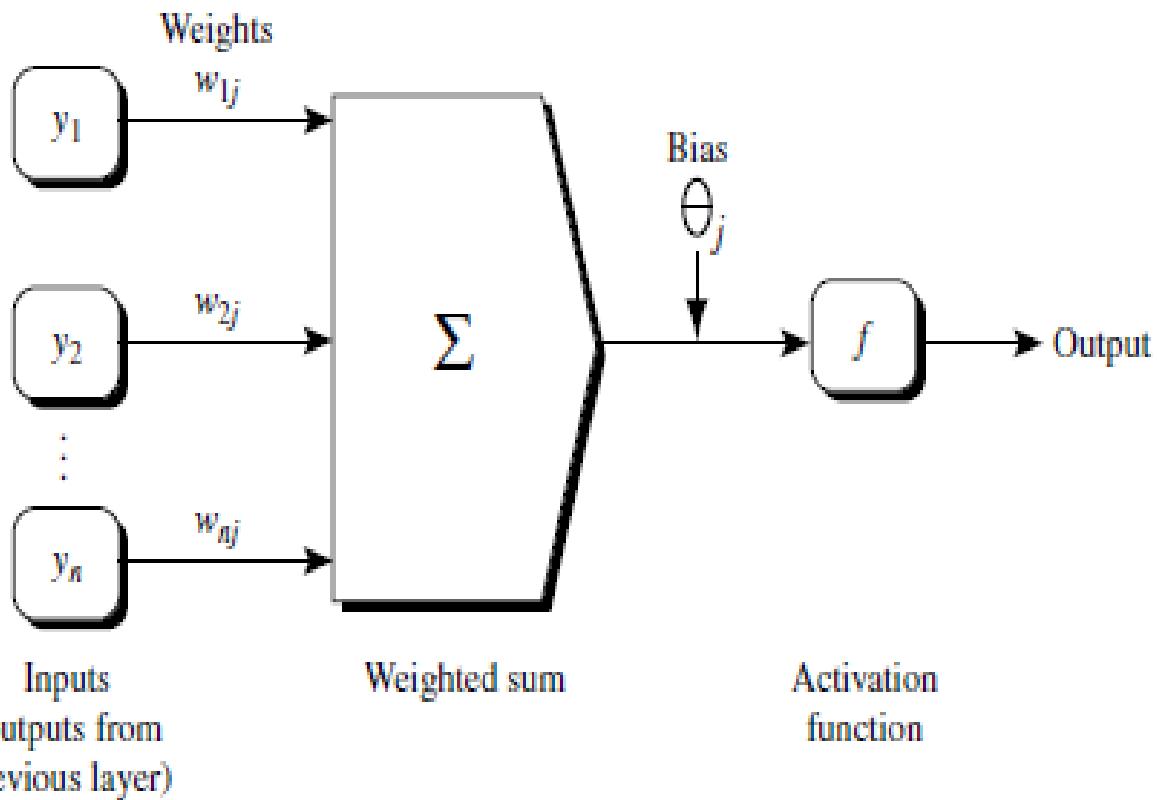
Learning rate is very important



# Multilayer Feed-Forward Neural Network



# Back propagation Learning

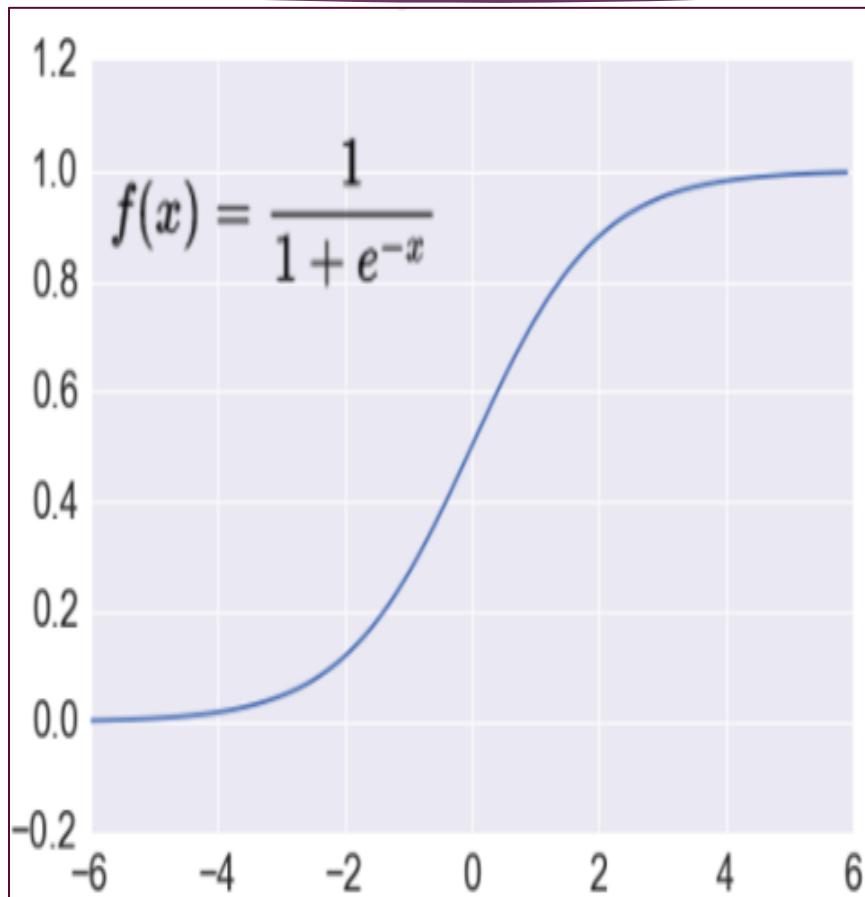


- ▶ **Initialize the weights:** The weights in the network are initialized to small random numbers (e.g., ranging from -1.0 to 1.0, or -0.5 to 0.5). Each unit has a *bias* associated with it. The biases are similarly initialized to small random numbers
- ▶ Each training tuple,  $X$ , is processed by the following steps.
- ▶ **Propagate the inputs forward:**

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

# Calculate Output: Activation: Sigmoid

- ▶ Property of Sigmoid function: (Activation)
- ▶ Takes a real-valued number and "squashes" it into range between 0 and 1.
- ▶ Nice interpretation as the firing rate of a neuron
  - 0 = not firing at all
  - 1 = fully firing
- Sigmoid neurons saturate and kill gradients, thus NN will barely learn
  - when the neuron's activation are 0 or 1 (saturate)  
gradient at these regions almost zero  
almost no signal will flow to its weights  
if initial weights are too large then most neurons would saturate



- ▶ Calculate the output using Sigmoid function: (Activation)

$$O_j = \frac{1}{1 + e^{-I_j}}$$

# Back propagate the error

- ▶ The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction. For a unit  $j$  in the output layer, the error is computed by

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

- ▶ To compute the error of a hidden layer unit  $j$ , the weighted sum of the errors of the units connected to unit  $j$  in the next layer are considered. The error of a hidden layer unit  $j$  is

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$

$w_{jk}$  is the weight of the connection from unit  $j$  to a unit  $k$  in the next higher layer, and  $Err_k$  is the error of unit  $k$

# Updating of weights and biases

The weights and biases are updated to reflect the propagated errors

The variable  $l$  is the learning rate, a constant typically having a value between 0.0 and 1.0

$$\Delta w_{ij} = l Err_j O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}.$$

$$\Delta \theta_j = l Err_j.$$
$$\theta_j = \theta_j + \Delta \theta_j.$$

Back propagation learns using a gradient descent method to search for a set of weights that fits the training data so as to minimize the mean-squared distance between the network's class prediction and the known target value of the tuples

The learning rate helps avoid getting stuck at a local minimum in decision space (i.e., where the weights appear to converge, but are not the optimum solution) and encourages finding the global minimum

If the learning rate is too small, then learning will occur at a very slow pace. If the learning rate is too large, then oscillation between inadequate solutions may occur. A rule of thumb is to set the learning rate to  $l=1/t$ , where  $t$  is the number of iterations through the training set so far.

# Terminating condition

## Training stops when

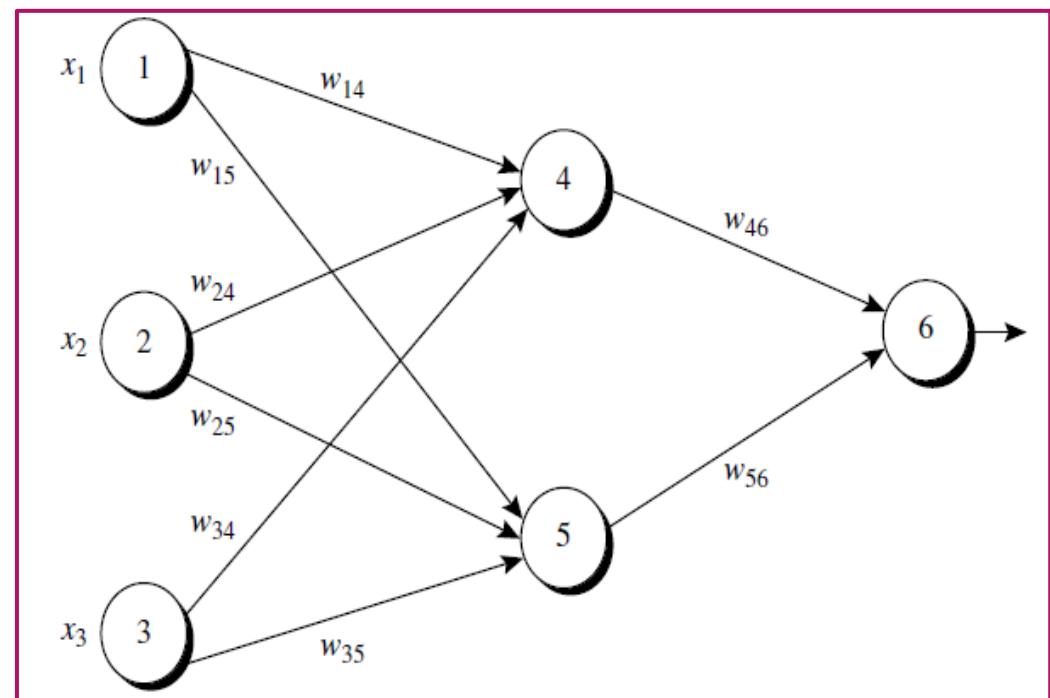
- ▶ All  $\Delta w$  in the previous epoch are so small as to be below some specified threshold,  
or
- The percentage of tuples misclassified in the previous epoch is below some  
threshold,  
or
- ▶ A pre-specified number of epochs has expired

# Demonstration:

Figure shows a multilayer feed-forward neural network. Let the learning rate be 0.9. The initial weight and bias values of the network are given in Table 9.1, along with the first training tuple,  $\mathbf{X} = (1, 0, 1)$ , with a class label of 1

Initial Input, Weight, and Bias Values

$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1



# Solution:

## Net Input and Output Calculations

<i>Unit, j</i>	<i>Net Input, I<sub>j</sub></i>	<i>Output, O<sub>j</sub></i>
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

## Calculation of the Error at Each Node

<i>Unit, j</i>	<i>Err<sub>j</sub></i>
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

# Weight and bias values after first iteration

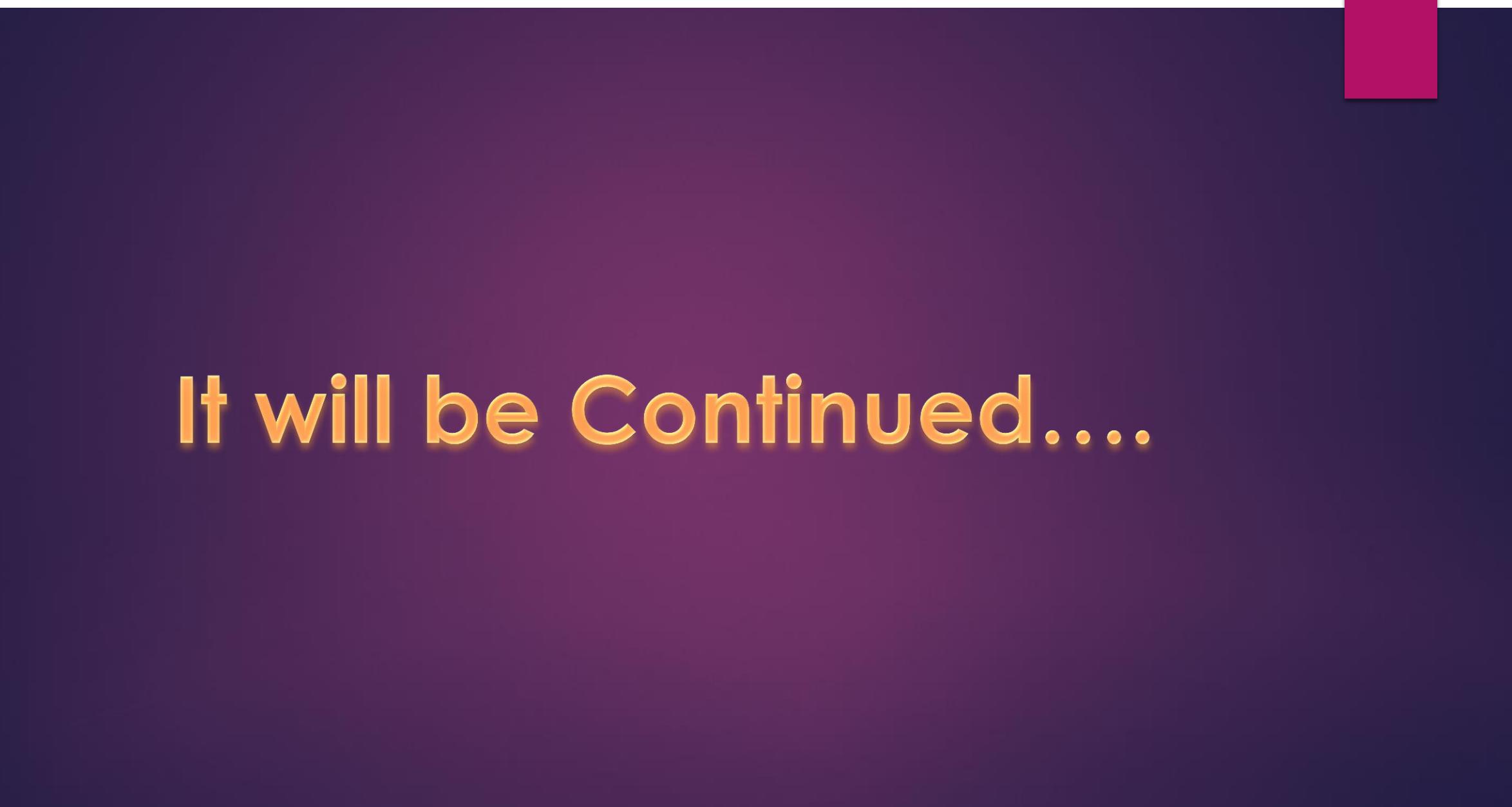
## Calculations for Weight and Bias Updating

Weight or Bias	New Value
$w_{46}$	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
$w_{56}$	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
$w_{14}$	$0.2 + (0.9)(-0.0087)(1) = 0.192$
$w_{15}$	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
$w_{24}$	$0.4 + (0.9)(-0.0087)(0) = 0.4$
$w_{25}$	$0.1 + (0.9)(-0.0065)(0) = 0.1$
$w_{34}$	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
$w_{35}$	$0.2 + (0.9)(-0.0065)(1) = 0.194$
$\theta_6$	$0.1 + (0.9)(0.1311) = 0.218$
$\theta_5$	$0.2 + (0.9)(-0.0065) = 0.194$
$\theta_4$	$-0.4 + (0.9)(-0.0087) = -0.408$

# Problem Statement

- ▶ Network topology:- 3-(2-2)-2
- ▶ Weight and bias values are initialized to 0
- ▶ For input set {1,0,1} the class label will be {1,0}

Q: What will be the updated weight and bias values after 2<sup>nd</sup> iteration if learning rate = 1  
?



**It will be Continued....**

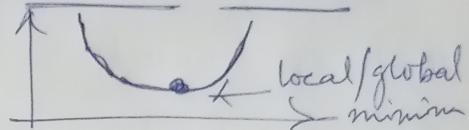
## Backpropagation → justification

### Cost function / Loss function

$$J = \frac{1}{2} \sum (t_j - o_j)^2 \Rightarrow \text{squared error.}$$

Minimize the cost function  $\rightarrow$  Gradient descent-

$$\Delta w = -\eta \frac{\partial J}{\partial w}$$



From ANN model we are getting  $\rightarrow$   $net_j = \sum x_j w_j + b_j$

$$o_j = \frac{1}{1 + e^{-net_j}}$$

Hence  $\rightarrow$

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_j}$$

$$J = \frac{1}{2} \cdot \sum (t_j - o_j)^2$$

$$\therefore \frac{\partial J}{\partial o_j} = \frac{1}{2} \cdot 2 \cdot (t_j - o_j) (-1) = -(t_j - o_j)$$

As we know,

$$o_j = \frac{1}{1 + e^{-net_j}}$$

Assume that

$$y = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}$$

$$\therefore \frac{\partial y}{\partial x} = \frac{(1 + e^x) \cdot e^x - e^x \cdot e^x}{(1 + e^x)^2} = \frac{e^x}{(1 + e^x)^2} = \frac{e^x}{(1 + e^x)} \left[ 1 - \frac{e^x}{1 + e^x} \right] \\ = y(1 - y)$$

therefore,

$$\frac{dy}{dx} = \gamma(1-y)$$

likewise,

$$\frac{\partial o_j}{\partial \text{Net}_j} = o_j(1-o_j)$$

Moreover,

$$\frac{\partial \text{Net}_j}{\partial w_j} = x_j$$

Therefore,

$$\begin{aligned}\cancel{\frac{\partial J}{\partial w_j}} &= \cancel{\frac{\partial J}{\partial o_j}} \cdot \cancel{\frac{\partial o_j}{\partial \text{Net}_j}} \cdot \cancel{\frac{\partial \text{Net}_j}{\partial w_j}} \\ &= -(T_j - o_j) o_j(1-o_j) x_j\end{aligned}$$

Now,  $x_j = o_i$  that output of  $i^{\text{th}}$  Neuron.

$$\therefore \boxed{\frac{\partial J}{\partial w_j} = -(T_j - o_j) o_j(1-o_j) o_i}$$

From perceptron learning rule we are getting-

$$\Delta w = l \cdot \underline{(T_j - o_j)} \cdot o_i = \underline{l \cdot \text{Err}_j \cdot o_i} \quad \text{--- } ①$$

from gradient decent we are getting- $\rightarrow$

$$\Delta w = -l \cdot \frac{\partial J}{\partial w} = l \cdot (T_j - o_j) o_j(1-o_j) o_i \quad \text{--- } ②$$

from ① and ② we are getting.

$$\text{Err}_j = (T_j - o_j) o_j(1-o_j)$$

$$\therefore \boxed{\text{Err}_j = o_j(1-o_j)(T_j - o_j)}$$

# Perceptron Learning

BY

DR. ANUPAM GHOSH

# Perceptron-Learning

- ▶ The input vectors are allowed to be either binary or bipolar. However, the outputs must be in bipolar form
- ▶ The bias  $W_0$  is adjustable but the threshold  $\Theta$  used in the activation function is fixed
- ▶ Learning takes place only when the computed output does not match the target output.
- ▶ The threshold  $\Theta$  of the activation function may be interpreted as a separating band of width  $2\Theta$  between the region of positive response and negative response.
- ▶ The band separating the regions of positive response from that of the negative response is defined by the pair of lines

$$w_0x_0 + w_1x_1 + w_2x_2 = \Theta$$

$$w_0x_0 + w_1x_1 + w_2x_2 = -\Theta$$

# Algorithm

**Procedure** Perceptron-Learning

**Step 1.** Initialize all weights,  $w_0, \dots, w_m$ .

**Step 2.** Set learning rate  $\eta$  such that  $0 < \eta \leq 1$ , and threshold  $\theta$ .

**Step 3.** For each training pair  $s : t$  do Steps 4-8.

**Step 4.** Activate the input units,  $x_i = s_i$ , for  $i = 0, \dots, m$ .

**Step 5.** Compute the net input to the output unit

$$y\_in = \sum_{i=0}^m w_i x_i$$

**Step 6.** Compute the activation of the output unit using the function

$$y\_out = \begin{cases} 1, & \text{if } y\_in > \theta \\ 0, & \text{if } -\theta \leq y\_in \leq \theta \\ -1, & \text{if } y\_in < -\theta \end{cases}$$

**Step 7.** If there is an error, i.e.,  $y\_out \neq t$ , then adjust the weights as follows

$$w_i \text{ (new)} = w_i \text{ (old)} + \eta \times t \times x_i$$

If, however, no error has occurred, the weights are kept unchanged.

**Step 8.** If there were no error, i.e.,  $y\_out = t$ , for the entire set of training pairs, then stop. Otherwise go to Step 3.

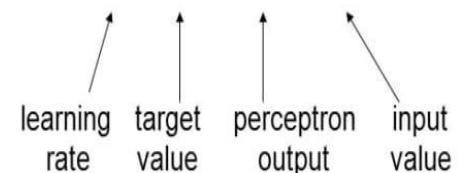
## Perceptron Training Rule

- Weights modified for each example
- Update Rule:

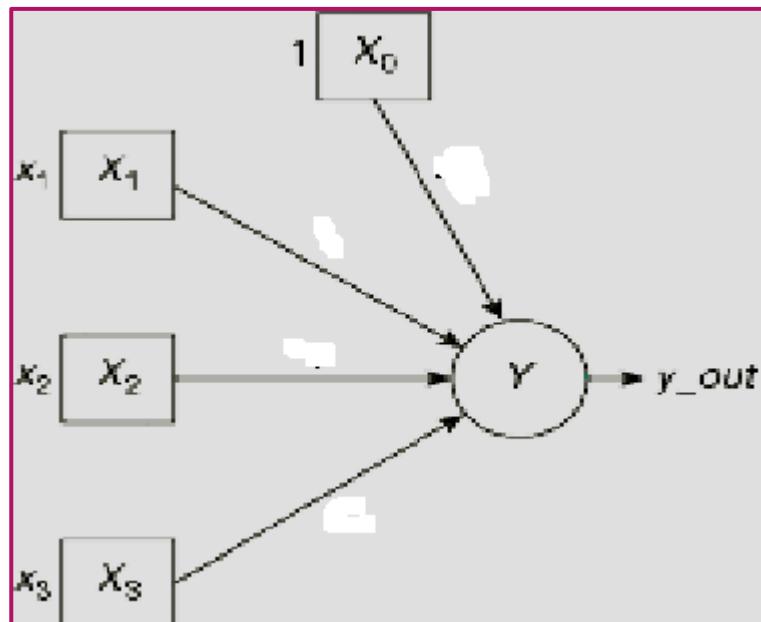
$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$



# Sample Demonstration



Input pattern			Target output ( $t$ )
$x_1$	$x_2$	$x_3$	
-1	1	1	-1
1	-1	1	-1
1	1	-1	-1
1	1	1	1

the initial weights are all kept at 0 and the learning rate is set to  $\eta = 1$ . Both the inputs and the outputs are presented in bipolar form.

#	Input				Net input	Out- put	Tar- get	Weight adjustments				Weights			
	$x_0$	$x_1$	$x_2$	$x_3$				$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												0	0	0	0
1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
2	1	-1	1	1	2	1	-1	-1	1	-1	-1	0	2	0	0
3	1	1	-1	1	2	1	-1	-1	-1	1	-1	-1	1	1	-1
4	1	1	1	-1	2	1	-1	-1	-1	-1	1	-2	0	0	0

Epoch #1

#	Input				Net input	Out- put	Tar- get	Weight adjustments				Weights			
	$x_0$	$x_1$	$x_2$	$x_3$				$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												-2	0	0	0
1	1	1	1	1	-2	-1	1	1	1	1	1	-1	1	1	1
2	1	-1	1	1	0	0	-1	-1	1	-1	-1	-2	2	0	0
3	1	1	-1	1	0	0	-1	-1	-1	1	-1	-3	1	1	-1
4	1	1	1	-1	0	0	-1	-1	-1	-1	1	-4	0	0	0

Epoch #2

#	Input				Net input	Out- put	Tar- get	Weight adjustments				Weights			
	$x_0$	$x_1$	$x_2$	$x_3$				$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												-4	0	0	0
1	1	1	1	1	-4	-1	1	1	1	1	1	-3	1	1	1
2	1	-1	1	1	-2	-1	-1	0	0	0	0	-3	1	1	1
3	1	1	-1	1	-2	-1	-1	0	0	0	0	-3	1	1	1
4	1	1	1	-1	-2	-1	-1	0	0	0	0	-3	1	1	1

Epoch #3

#	Input				Net input	Out- put	Tar- get	Weight adjustments				Weights			
	$x_0$	$x_1$	$x_2$	$x_3$	$y_{in}$	$y_{out}$	$t$	$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												-3	1	1	1
1	1	1	1	1	0	0	1	1	1	1	1	-2	2	2	2
2	1	-1	1	1	0	0	-1	-1	1	-1	-1	-3	3	1	1
3	1	1	-1	1	0	0	-1	-1	-1	1	-1	-4	2	2	0
4	1	1	1	-1	0	0	-1	-1	-1	-1	1	-5	1	1	1

Epoch #4

#	Input		Net input	Out- put	Tar- get	Weight adjustments				Weights					
	$x_0$	$x_1$	$x_2$	$x_3$	$y_{in}$	$y_{out}$	$t$	$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												-5	1	1	1
1	1	1	1	1	-2	-1	1	1	1	1	1	-4	2	2	2
2	1	-1	1	1	-2	-1	-1	-1	1	-1	-1	-4	2	2	2
3	1	1	-1	1	-2	-1	-1	-1	-1	1	-1	-4	2	2	2
4	1	1	1	-1	-2	-1	-1	-1	-1	-1	1	-4	2	2	2

#	Input				Net input	Out- put	Tar- get	Weight adjustments				Weights			
	$x_0$	$x_1$	$x_2$	$x_3$	$y_{in}$	$y_{out}$	$t$	$\Delta w_0$	$\Delta w_1$	$\Delta w_2$	$\Delta w_3$	$w_0$	$w_1$	$w_2$	$w_3$
0												-4	2	2	2
1	1	1	1	1	2	1	1	0	0	0	0	-4	2	2	2
2	1	-1	1	1	-2	-1	-1	0	0	0	0	-4	2	2	2
3	1	1	-1	1	-2	-1	-1	0	0	0	0	-4	2	2	2
4	1	1	1	-1	-2	-1	-1	0	0	0	0	-4	2	2	2

Epoch #6



# ADALINE & MADALINE

# ADALINE

The ADALINE (Adaptive Linear Neuron), introduced by Widrow and Hoff in 1960, is a single output unit neural net with several input units. One of the input units acts as the bias and is permanently fixed at 1. An ADALINE is trained with the help of the delta, or LMS (Least Mean Square), or Widrow-Hoff learning rule.

The salient features of *ADALINE* are

- ▶ Both the inputs and the outputs are presented in bipolar form.
- ▶ The net is trained through the delta, or LMS, or Widrow-Hoff rule. This rule tries to minimize the mean-squared error between activation and the target value.
- ▶ *ADALINE* employs the identity activation function at the output unit *during training*. This implies that during training  $y_{out} = y_{in}$ .
- ▶ During application the following bipolar step function is used for activation.

$$y = \begin{cases} 1, & \text{if } y_{in} \geq 0 \\ -1, & \text{if } y_{in} < 0 \end{cases}$$

- ▶ Learning by an *ADALINE* net is sensitive to the value of the learning rate. A too large learning rate may prevent the learning process to converge. On the other hand, if the learning rate is too low, the learning process is extremely slow. Usually, the learning rate is set on the basis of the inequality  $.1 \leq m \times \eta \leq 1.0$ , where  $m$  is the number of input units.

# MADALINE

- ▶ Several *ADALINES* arranged in a multilayer net is known as *Many ADALINES*, or *Many Adaptive Linear Neurons*, or *MADALINE* in short. The architecture of a two input, one output, one hidden layer consisting of two hidden *MADALINE* is shown in Figure. *MADALINE* is computationally more powerful than *ADALINE*. The enhanced computational power of the *MADALINE* is due to the hidden *ADALINE* units. Salient features of *MADALINE* are mentioned below.
- ▶ All units, except the inputs, employ the same activation function as in *ADALINE*, i.e.,  
$$f(x) = \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$
- ▶ As mentioned earlier, the enhanced computational power of the *MADALINE* is due to the hidden *ADALINE* units. However, existence of the hidden units makes the training process more complicated.
- ▶ There are two training algorithms for *MADALINE*, viz., *MR-I* and *MR-II*.

**Procedure MADALINE-MR-I-Learning**

**Step 1.** Initialize  $v_0, v_1, v_2$  with 0.5 and other weights  $w_{01}, w_{11}, w_{12}, w_{21}, w_{12}$  and  $w_{22}$  by small random values. All bias inputs are set to 1.

**Step 2.** Set the learning rate  $h$  to a suitable value.

**Step 3.** For each bipolar training pair  $s : t$ , do Steps 4-6

**Step 4.** Activate the input units:  $x_1 = s_1, x_2 = s_2$ , all biases are set to 1 permanently.

**Step 5.** Propagate the input signals through the net to the output unit  $y$ .  
5.1 Compute net inputs to the hidden units.

$$z_{in_1} = 1 \times w_{01} + x_1 \times w_{11} + x_2 \times w_{21}$$

$$z_{in_2} = 1 \times w_{02} + x_1 \times w_{12} + x_2 \times w_{22}$$

5.2 Compute activations of the hidden units  $z_{out_1}$  and  $z_{out_2}$  using the bipolar step function

$$z_{out} = \begin{cases} 1, & \text{if } z_{in} \geq 0 \\ -1, & \text{if } z_{in} < 0. \end{cases}$$

5.3 Compute net input to the output unit

$$y_{in} = 1 \times v_0 + z_{out_1} \times v_1 + z_{out_2} \times v_2$$

5.4 Find the activation of the output unit  $y_{out}$  using the same activation function as in Step 5.2, i.e.,

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} \geq 0 \\ -1, & \text{if } y_{in} < 0. \end{cases}$$

**Step 6.** Adjust the weights of the hidden units, if required, according to the following rules:

i) If ( $y_{out} = t$ ) then the net yields the expected result. Weights need not be updated.

ii) If ( $y_{out} \neq t$ ) then apply one of the following rules whichever is applicable.

**Case I:**  $t = 1$

Find the hidden unit  $z_i$  whose net input  $z_{in_i}$  is closest to 0. Adjust the weights attached to  $z_i$  according to the formula

$$w_{ij} (\text{new}) = w_{ij} (\text{old}) + h \times (1 - z_{in_j}) \times x_i, \text{ for all } i.$$

**Case II:**  $t = -1$

Adjust the weights attached to those hidden units  $z_i$  that have positive net input.

$$w_{ij} (\text{new}) = w_{ij} (\text{old}) + h \times (-1 - z_{in_j}) \times x_i, \text{ for all } i.$$

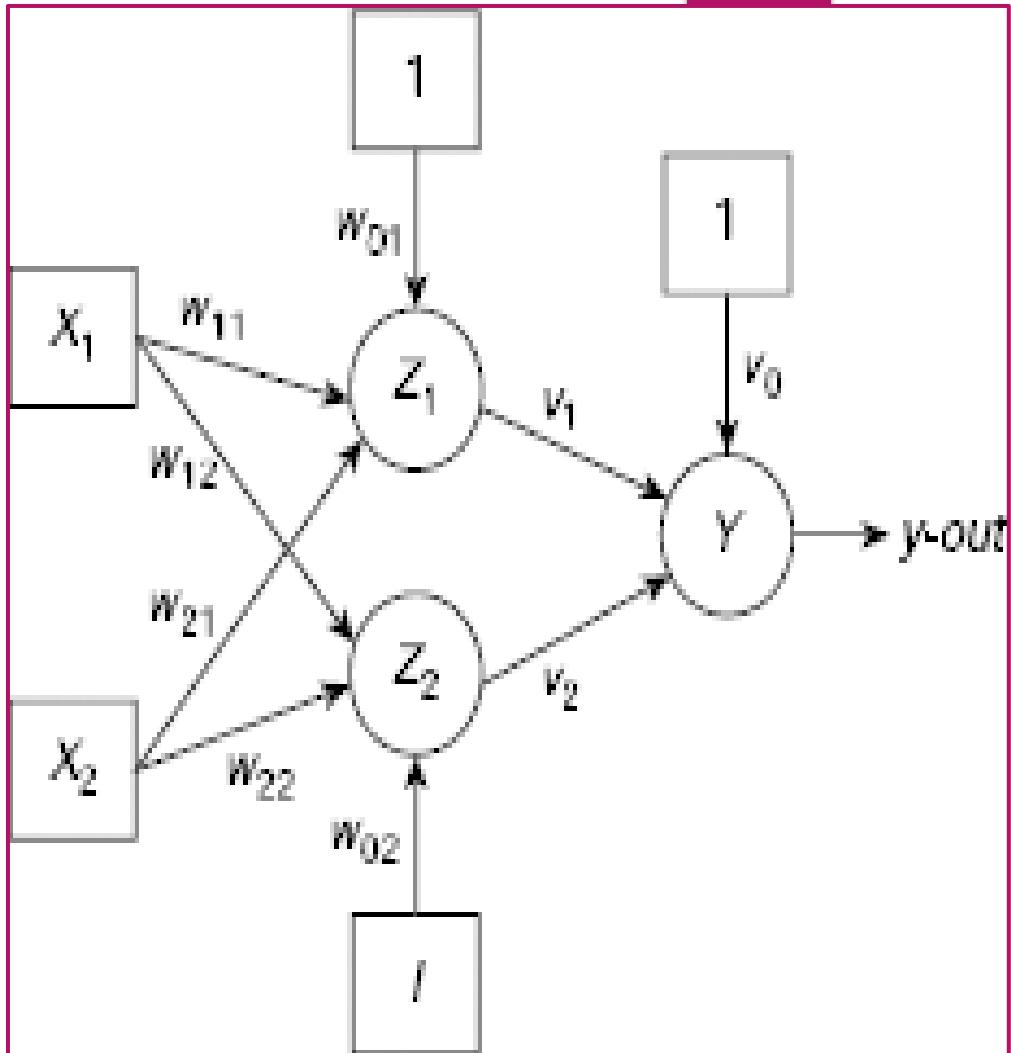
**Step 7.** Test for stopping condition. It can be any one of the following:

i) No change of weight occurs in Step 6.

ii) The weight adjustments have reached an acceptable level.

iii) A predefined number of iterations have been carried out.

If the stopping condition is satisfied then stop. Otherwise go to Step 3.



# Sample Run

Table Bipolar training set for XOR function

$x_0$	$x_1$	$x_2$	$t$
1	1	1	-1
1	1	-1	1
1	-1	1	1
1	-1	-1	-1

Table 7.10. Initial weights and the fixed learning rate

$w_{01}$	$w_{11}$	$w_{21}$	$w_{02}$	$w_{12}$	$w_{22}$	$\eta$
.2	.3	.2	.3	.2	.1	.5

Step 5.1 Compute net inputs  $z\_in_1$  and  $z\_in_2$  to the hidden units  $z_1$  and  $z_2$ .

Step 5.2 Compute activations of the hidden units  $z\_out_1$  and  $z\_out_2$  using the bipolar step function.

$$\begin{aligned} z\_in_1 &= 1 \times w_{01} + x_1 \times w_{11} + x_2 \times w_{21} \\ &= 1 \times .2 + 1 \times .3 + 1 \times .2 = .7 \end{aligned}$$

$$z\_out_1 = 1$$

$$\begin{aligned} z\_in_2 &= 1 \times w_{02} + x_1 \times w_{12} + x_2 \times w_{22} \\ &= 1 \times .3 + 1 \times .2 + 1 \times .1 = .6 \end{aligned}$$

$$\therefore z\_out_2 = 1$$

Step 5.3 Compute net inputs  $y\_in$  to the output units.

Step 5.4 Find the activation of the output unit  $y\_out$  using the same activation function as in Step 5.2, i.e.,

$$\begin{aligned} y\_in &= 1 \times v_0 + z\_out_1 \times v_1 + z\_out_2 \times v_2 \\ &= 1 \times .5 + 1 \times .5 + 1 \times .5 = 1.5 \end{aligned}$$

$$\therefore y\_out = 1$$

#### Step 6      Adjust the weights of the hidden units, if required.

Since  $t = -1$ ,  $y_{out} = 1 \neq t$ . Moreover, since  $t = -1$ , CASE II of Step 6 is applicable here. Therefore, we have to update weights on all units that have positive net inputs. Hence in this case we need to update the values of  $w_{01}$ ,  $w_{11}$ ,  $w_{21}$  as well as those of  $w_{02}$ ,  $w_{12}$ ,  $w_{22}$ . The computations for the said adjustments are shown below.

$$\begin{aligned}w_{01}(\text{new}) &= w_{01}(\text{old}) + \eta \times (-1 - z_{in1}) \\&= .2 + .5 \times (-1 - .7) \\&= .2 - .85 \\&= -.65\end{aligned}$$

$$\begin{aligned}w_{11}(\text{new}) &= w_{11}(\text{old}) + \eta \times (-1 - z_{in1}) \\&= .3 - .85 \\&= -.55\end{aligned}$$

$$\begin{aligned}w_{21}(\text{new}) &= w_{21}(\text{old}) + \eta \times (-1 - z_{in1}) \\&= .2 - .85 \\&= -.65\end{aligned}$$

$$\begin{aligned}w_{02}(\text{new}) &= w_{02}(\text{old}) + \eta \times (-1 - z_{in2}) \\&= .3 + .5 \times (-1 - .6) \\&= .3 - .8 \\&= -.5\end{aligned}$$

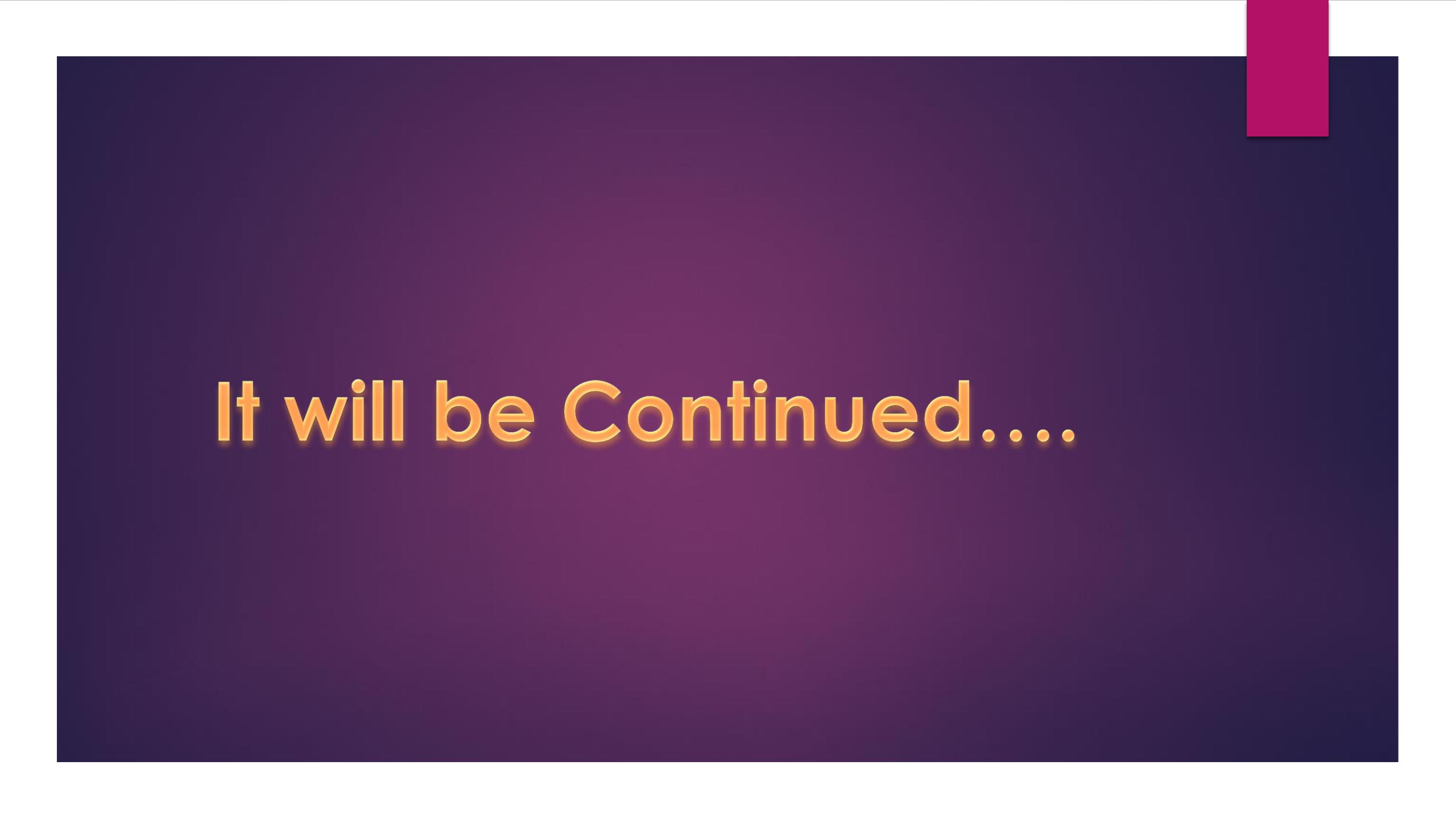
$$\begin{aligned}w_{12}(\text{new}) &= w_{12}(\text{old}) + \eta \times (-1 - z_{in2}) \\&= .2 - .8 \\&= -.6\end{aligned}$$

$$\begin{aligned}w_{22}(\text{new}) &= w_{22}(\text{old}) + \eta \times (-1 - z_{in2}) \\&= .1 - .8 \\&= -.7\end{aligned}$$

Hence the new set of weights after training with the first training pair  $(1, 1) : -1$  in the first epoch is obtained as

$$W = \begin{bmatrix} w_{01} & w_{02} \\ w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} -.65 & -.5 \\ -.55 & -.6 \\ -.65 & -.7 \end{bmatrix}$$

**Table 1** .. MADALINE Learning of XOR Function through MR-I algorithm



**It will be Continued....**

```

Procedure MADALINE-MR-I-Learning
Step 1. Initialize  $v_0$ ,  $v_1$ ,  $v_2$  with 0.5 and other weights  $w_{01}$ ,  $w_{11}$ ,  $w_{12}$ ,  $w_{02}$ ,  $w_{21}$  and  $w_{22}$  by small random values. All bias inputs are set to 1.
Step 2. Set the learning rate  $h$  to a suitable value.
Step 3. For each bipolar training pair  $s : t$ , do Steps 4-6
Step 4. Activate the input units:  $x_1 = s_1$ ,  $x_2 = s_2$ , all biases are set to 1 permanently.
Step 5. Propagate the input signals through the net to the output unit  $y$ .
  5.1 Compute net inputs to the hidden units.
    
$$z_{in_1} = 1 \times w_{01} + x_1 \times w_{11} + x_2 \times w_{21}$$

    
$$z_{in_2} = 1 \times w_{02} + x_1 \times w_{12} + x_2 \times w_{22}$$

  5.2 Compute activations of the hidden units  $z_{out_1}$  and  $z_{out_2}$  using the bipolar step function
    
$$z_{out} = \begin{cases} 1, & \text{if } z_{in} \geq 0 \\ -1, & \text{if } z_{in} < 0. \end{cases}$$

  5.3 Compute net input to the output unit
    
$$y_{in} = 1 \times v_0 + z_{out_1} \times v_1 + z_{out_2} \times v_2$$

  5.4 Find the activation of the output unit  $y_{out}$  using the same activation function as in Step 5.2, i.e.,
    
$$y_{out} = \begin{cases} 1, & \text{if } y_{in} \geq 0 \\ -1, & \text{if } y_{in} < 0. \end{cases}$$

Step 6. Adjust the weights of the hidden units, if required, according to the following rules:
  i) If ( $y_{out} = t$ ) then the net yields the expected result. Weights need not be updated.
  ii) If ( $y_{out} \neq t$ ) then apply one of the following rules whichever is applicable.
    Case I:  $t = 1$   

      Find the hidden unit  $z_j$  whose net input  $z_{in_j}$  is closest to 0.  

      Adjust the weights attached to  $z_j$  according to the formula  

      
$$w_{ij} (\text{new}) = w_{ij} (\text{old}) + h \times (1 - z_{in_j}) \times x_i, \text{ for all } i.$$

    Case II:  $t = -1$   

      Adjust the weights attached to those hidden units  $z_j$  that have positive net input.  

      
$$w_{ij} (\text{new}) = w_{ij} (\text{old}) + h \times (-1 - z_{in_j}) \times x_i, \text{ for all } i.$$

Step 7. Test for stopping condition. It can be any one of the following:
  i) No change of weight occurs in Step 6.
  ii) The weight adjustments have reached an acceptable level.
  iii) A predefined number of iterations have been carried out.
  If the stopping condition is satisfied then stop. Otherwise go to Step 3.

```

# Genetic Algorithm

BY

DR. ANUPAM GHOSH

DATE: 25.11.2023

# Limitations of the traditional optimization approaches

- Computationally expensive
- For a discontinuous objective function, methods may fail.
- Method may not be suitable for parallel computing.
- Discrete (integer) variables are difficult to handle.
- Methods may not necessarily adaptive.
  
- Evolutionary algorithms have been evolved to address the above mentioned limitations of solving optimization problems with traditional approaches.

# Evolutionary Algorithms

The algorithms, which follow some biological and physical behaviors:

## **Biologic behaviors:**

- ❑ Genetics and Evolution -> Genetic Algorithms (GA)
- ❑ Behavior of ant colony -> Ant Colony Optimization (ACO)
- ❑ Human nervous system -> Artificial Neural Network (ANN)
- ❑ In addition to that there are some algorithms inspired by some physical behaviors:

## **Physical behaviors:**

- ❑ Annealing process -> Simulated Annealing (SA)
- ❑ Swarming of particle -> Particle Swarming Optimization (PSO)
- ❑ Learning -> Fuzzy Logic (FL)

# A brief account on evolution

## Evolution : Natural Selection

### Four primary premises:

- ① ***Information propagation:*** An offspring has many of its characteristics of its parents (i.e. information passes from parent to its offspring). [**Heredity**]
- ② ***Population diversity:*** Variation in characteristics in the next generation. [**Diversity**]
- ③ ***Survival for existence:*** Only a small percentage of the offspring produced survive to adulthood. [**Selection**]
- ④ ***Survival of the best:*** Offspring survived depends on their inherited characteristics. [**Ranking**]

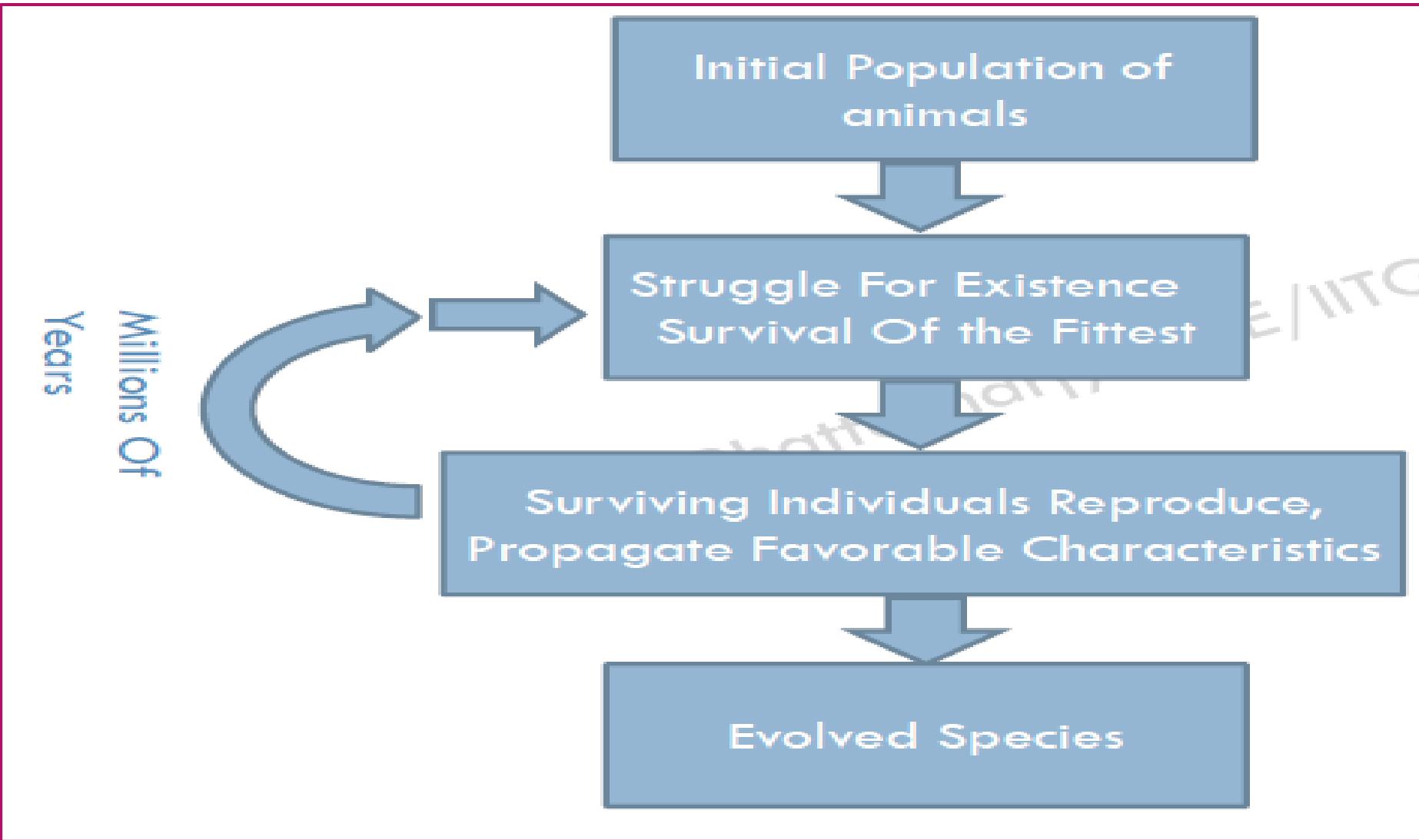
## History of GAs

- As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments.
- By the 1975, the publication of the book *Adaptation in Natural and Artificial Systems*, by Holland and his students and colleagues.
- early to mid-1980s, genetic algorithms were being applied to a broad range of subjects.
- In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method "genetic programming" (GP).

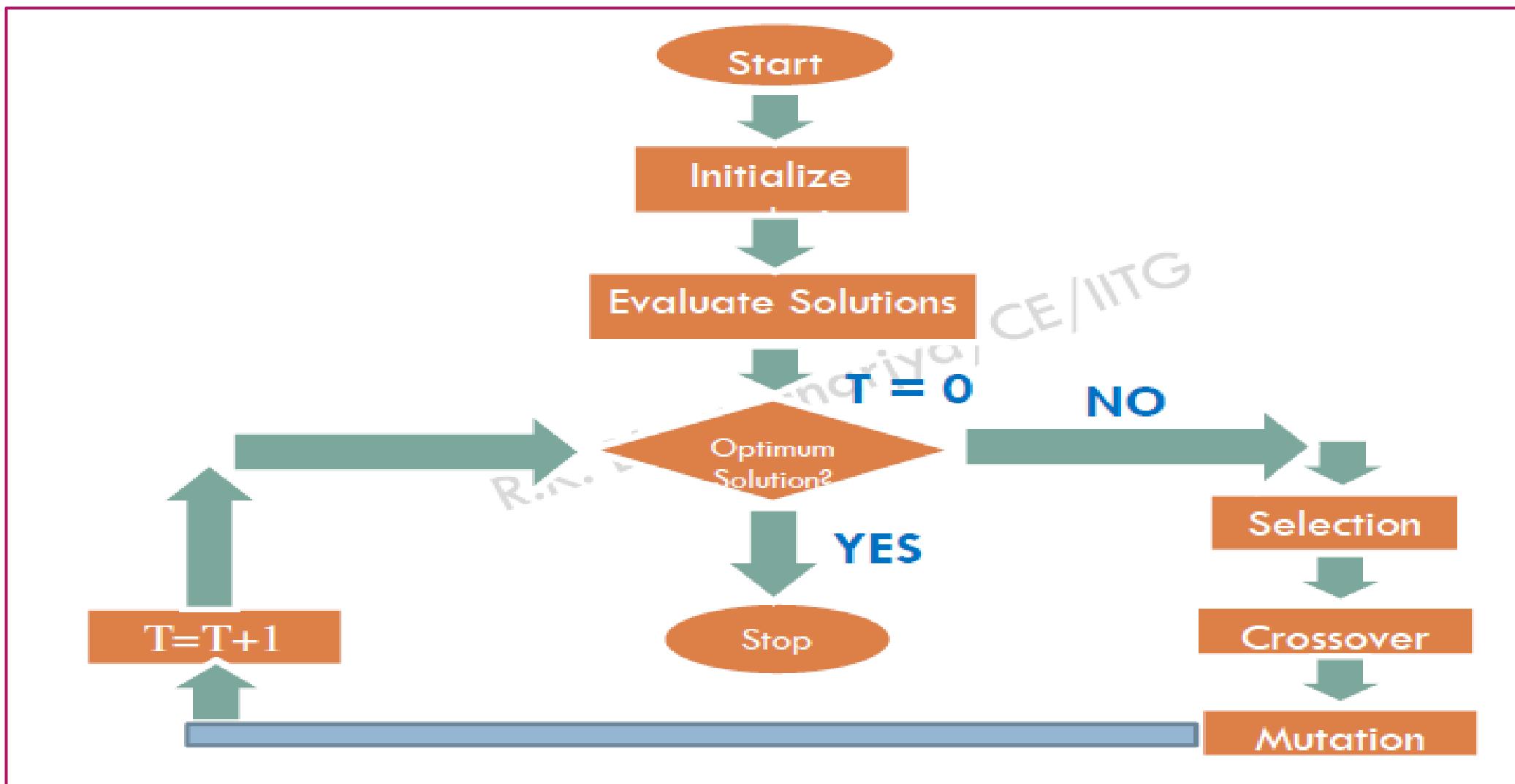
# Introduction

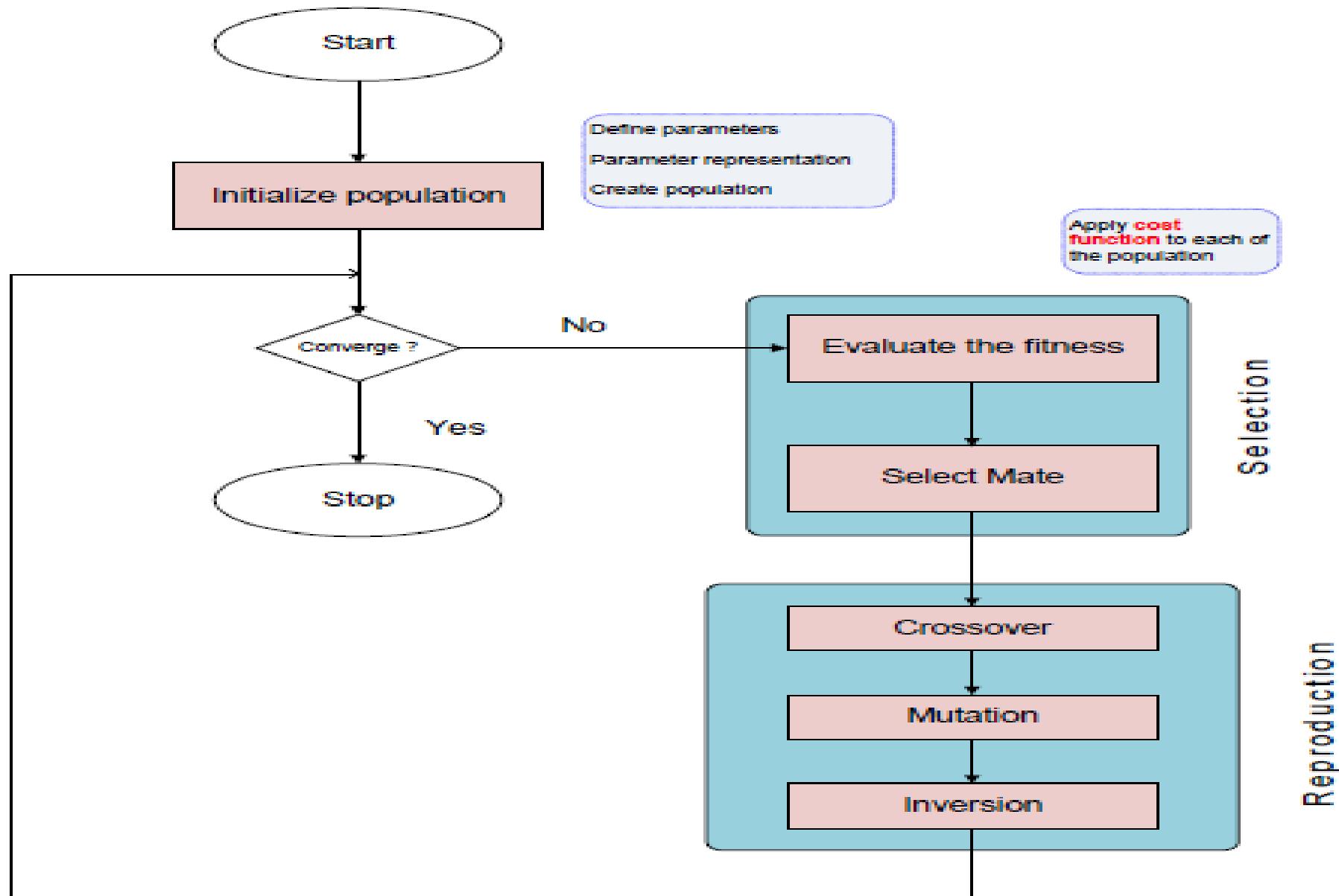
- ▶ Genetic Algorithms are the heuristic search and optimization techniques that mimic the process of natural evolution
- ▶ Genetic algorithms implement the optimization strategies by simulating evolution of species through natural selection
- ▶ GA Operators and Parameters:
  - Selection
  - Crossover
  - Mutation

## Evolution of species



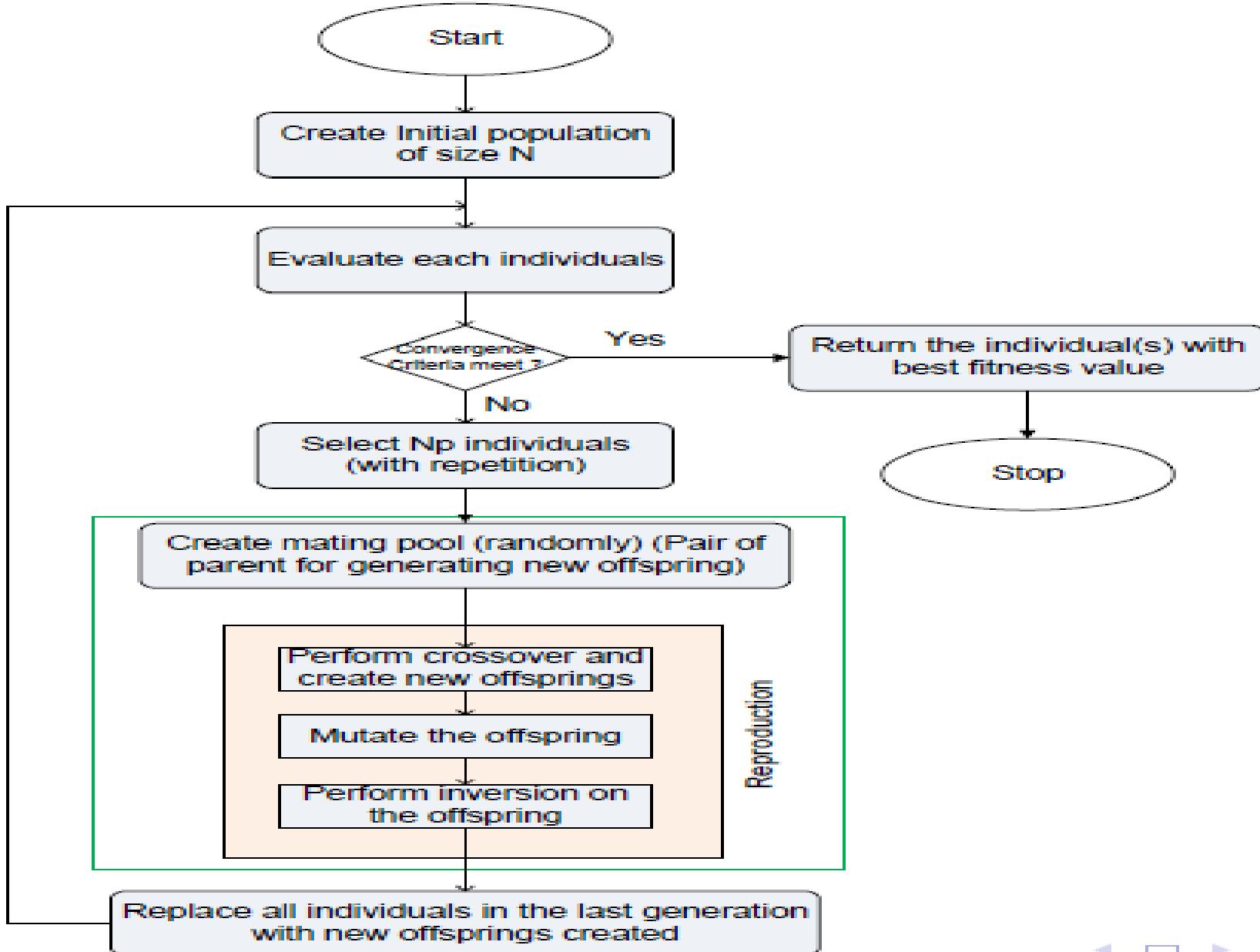
# Simple Genetic Algorithms





# GA Operators

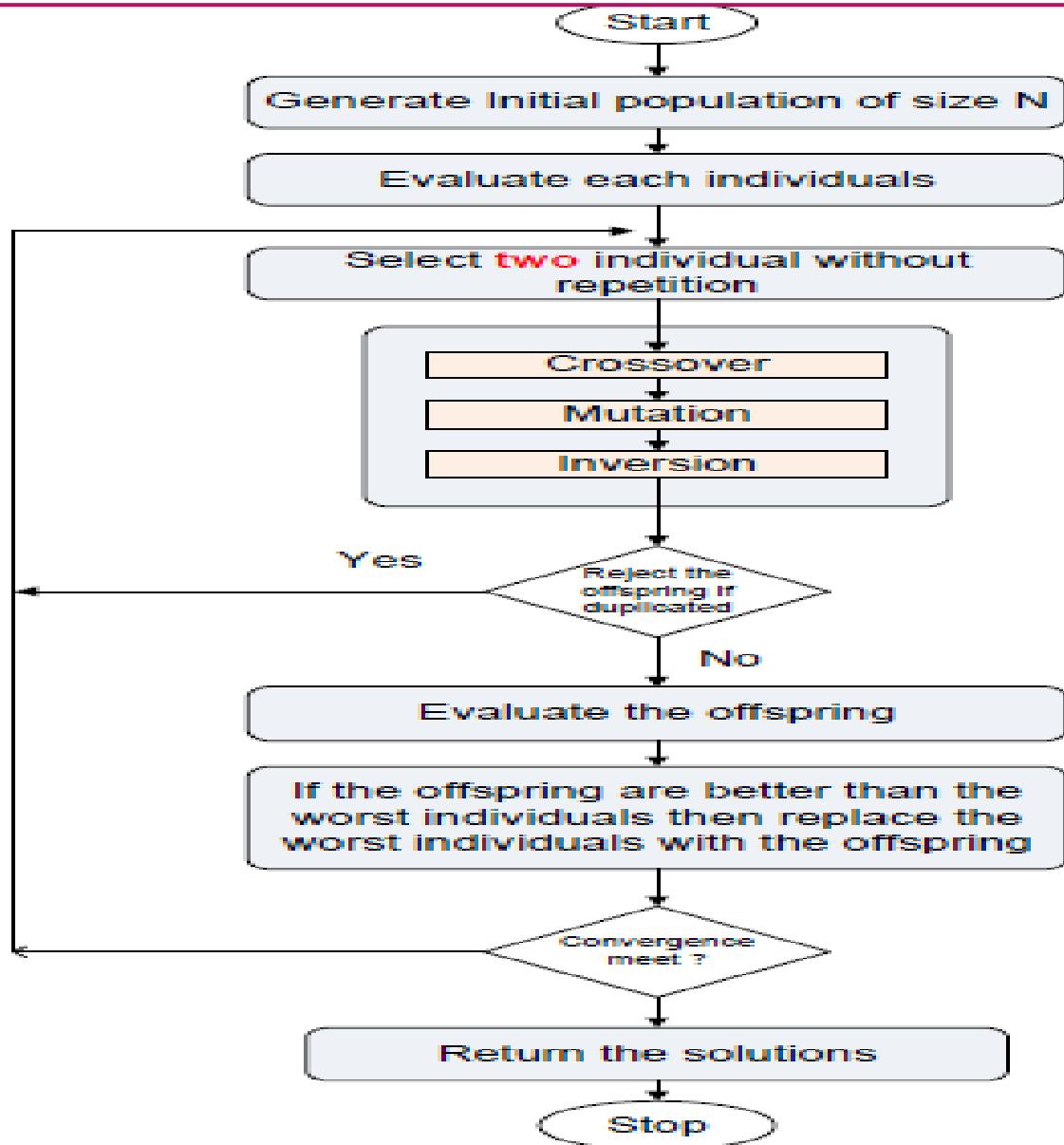
- ① **Encoding:** How to represent a solution to fit with GA framework.
- ② **Convergence:** How to decide the termination criterion.
- ③ **Mating pool:** How to generate next solutions.
- ④ **Fitness Evaluation:** How to evaluate a solution.
- ⑤ **Crossover:** How to make the diverse set of next solutions.
- ⑥ **Mutation:** To explore other solution(s).
- ⑦ **Inversion:** To move from one optima to other.



# Simple GA features

- Have overlapping generation (Only fraction of individuals are replaced).
- Computationally expensive.
- Good when initial population size is large.
- In general, gives better results.
- Selection is biased toward more highly fit individuals; Hence, the average fitness (of overall population) is expected to increase in succession.
- The best individual may appear in any iteration.

# Steady State Genetic Algorithm (SSGA)



## SGA Features:

- Generation gap is small.  
Only two offspring are produced in one generation.
- It is applicable when
  - Population size is small
  - Chromosomes are of longer length
  - Evaluation operation is less computationally expensive (compare to duplicate checking)

## Limitations in SSGA:

- There is a chance of stuck at local optima, if crossover/mutation/inversion is not strong enough to diversify the population).
- Premature convergence may result.
- It is susceptible to stagnation. Inferiors are neglected or removed and keeps making more trials for very long period of time without any gain (i.e. long period of localized search).

# Selection

- ▶ The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.
- ▶ The primary objective of the selection operator is to emphasize the good solutions and eliminate the bad solutions in a population while keeping the population size constant.
- ▶ “Selects the best, discards the rest”

# Functions of Selection operator

- ❑ Identify the good solutions in a population
- ❑ Make multiple copies of the good solutions
- ❑ Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population
- ❑ There are different techniques to implement selection in Genetic Algorithms:

Tournament selection. Roulette wheel selection Proportionate selection Rank selection Steady state selection

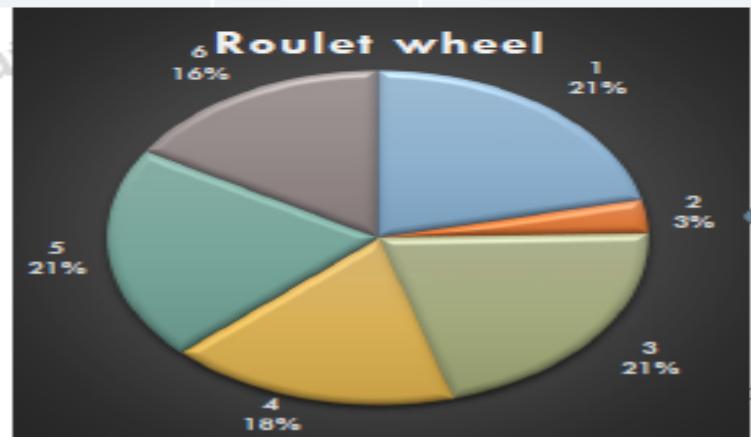
- ❑ A fitness value can be assigned to evaluate the solutions
- ❑ A fitness function value quantifies the optimality of a solution.
- ❑ The value is used to rank a particular solution against all the other solutions
- ❑ A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem

# Roulette wheel and proportionate selection

Parents are selected according to their fitness values

The better chromosomes have more chances to be selected

Chrom #	Fitness	% of RW	EC	AC
1	50	26.88	1.61	2
2	6	3.47	0.19	0
3	36	20.81	1.16	1
4	30	17.34	0.97	1
5	36	20.81	1.16	1
6	28	16.18	0.90	1
	186	100.00	6	6



November 2013

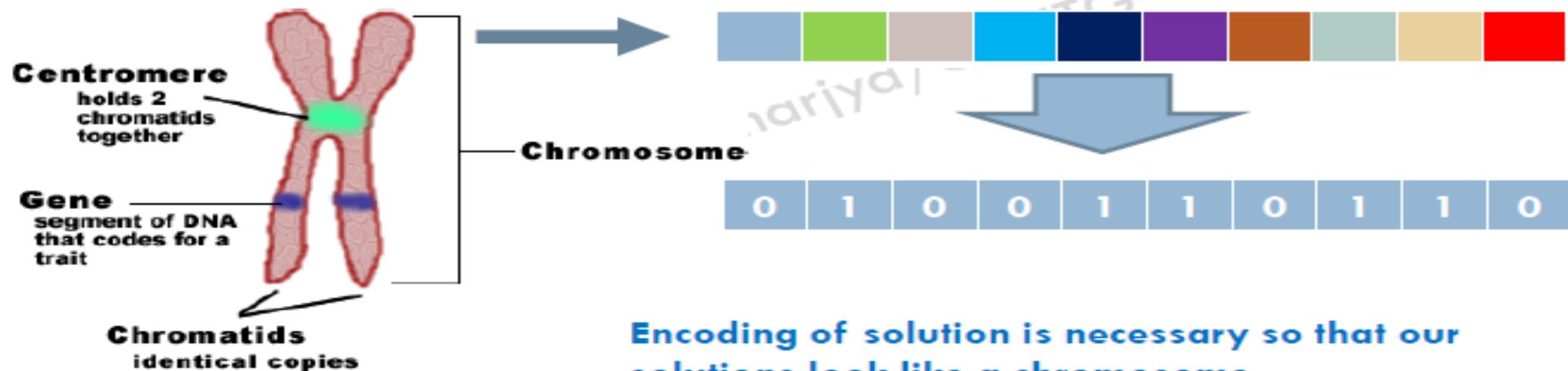
# Tournament selection

- ▶ In tournament selection several tournaments are played among a few individuals.
- ▶ The individuals are chosen at random from the population.
- ▶ The winner of each tournament is selected for next generation.
- ▶ Selection pressure can be adjusted by changing the tournament size.
- ▶ Weak individuals have a smaller chance to be selected if tournament size is large.

# How to implement crossover

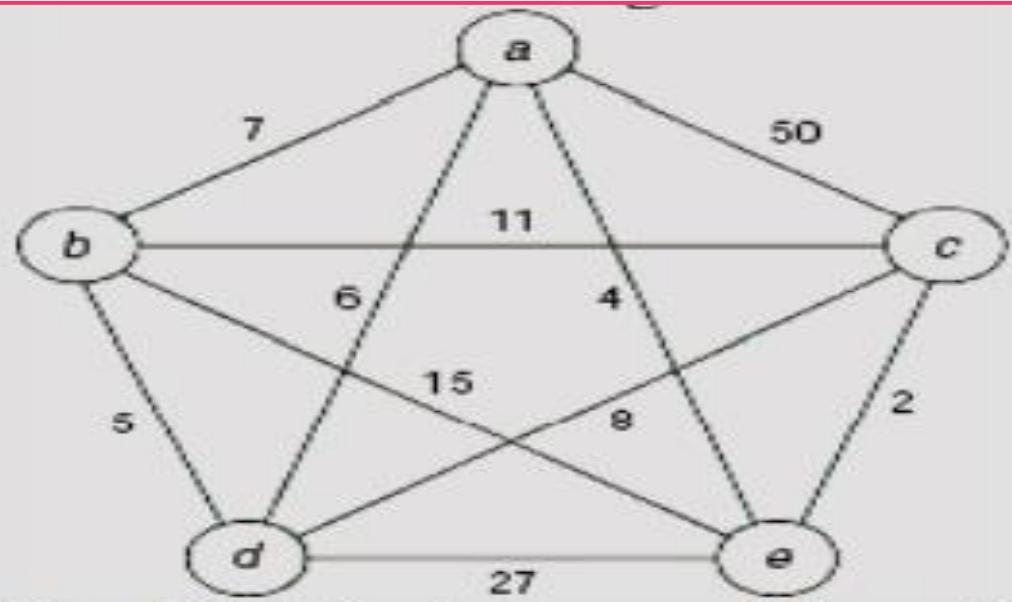
The crossover operator is used to create new solutions from the existing solutions available in the mating pool after applying selection operator.

This operator exchanges the gene information between the solutions in the mating pool.

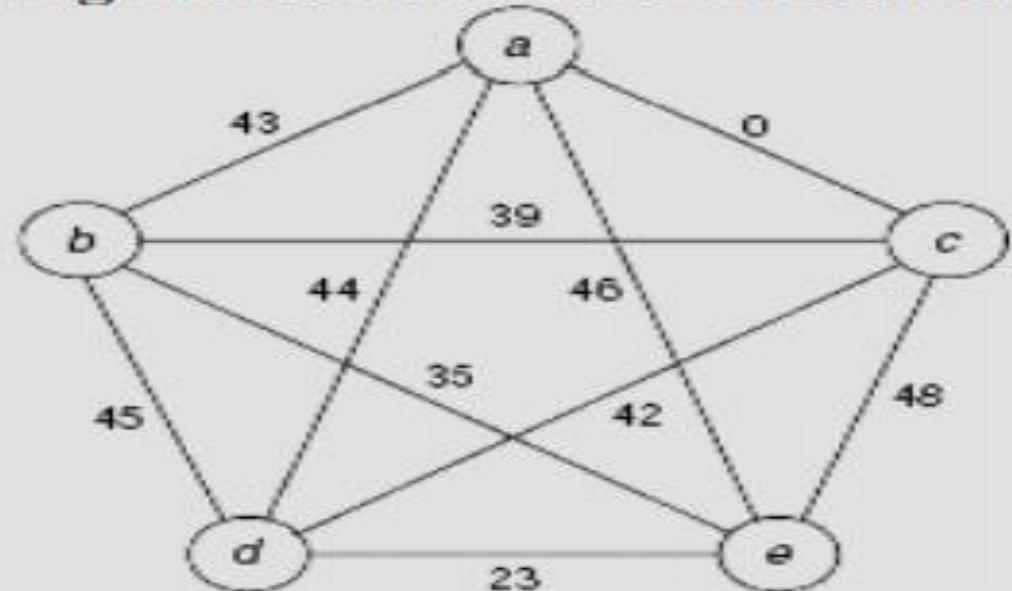


# Encoding

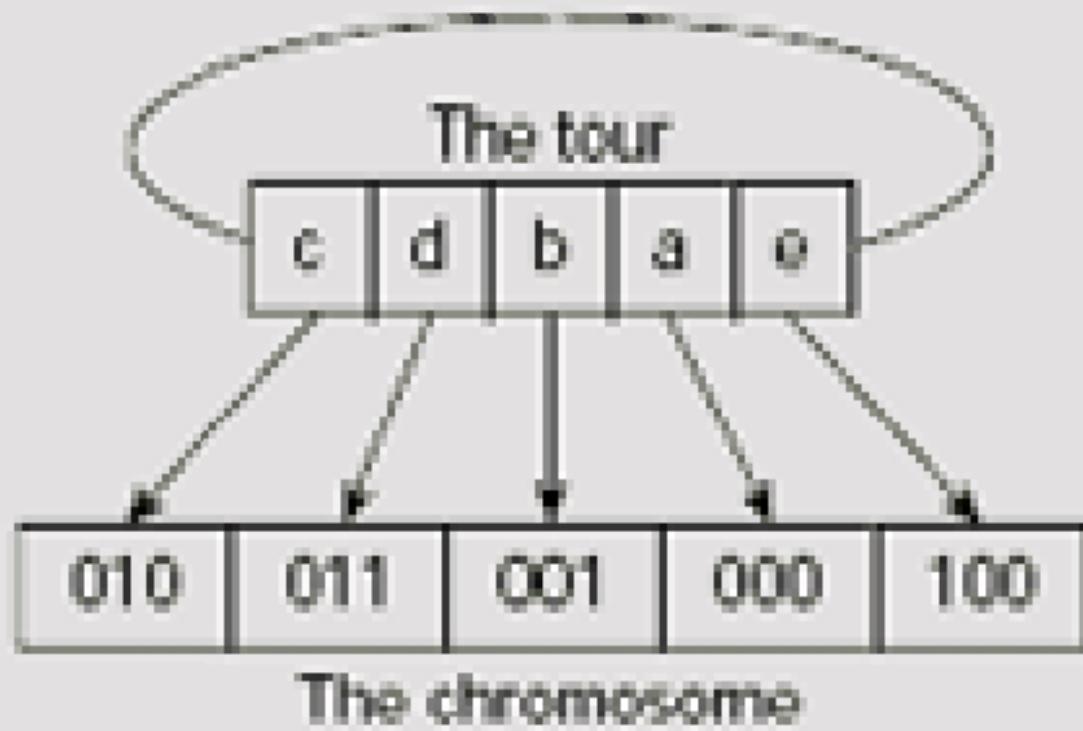
- ▶ The process of representing a solution in the form of a string that conveys the necessary information.
- ▶ Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution.



**Fig. 12.4** An instance of Travelling Salesperson Problem



#	Node	3-bit Code
0	a	000
1	b	001
2	c	010
3	d	011
4	e	100
5		101
6	Unused	110
7		111



$$\begin{aligned}\text{Reward} &= 42 + 45 + 43 \\ &\quad + 46 + 48 = 224\end{aligned}$$

**chromosome ch = 101 011 001 110 001 will be interpreted as the tour a → d → b → c → e.**

**Table 12.1** Randomly Generated Initial Population

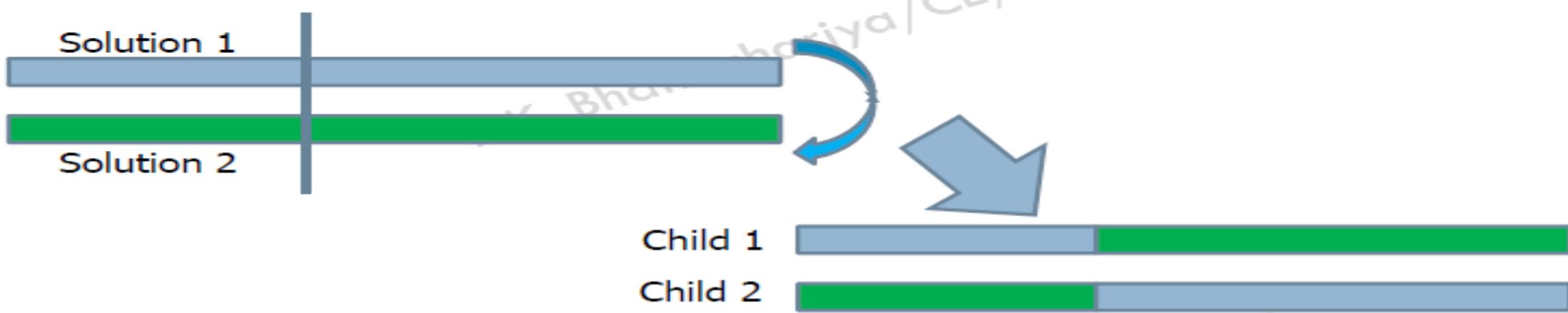
#	Chromosome	Tour	Fitness
1	011 101 111 011 001	d-e-a-b-c	193
2	010 100 001 101 110	c-e-b-d-a	172
3	100 100 001 111 010	e-a-b-c-d	193
4	011 100 110 001 101	d-e-a-b-c	193
5	110 011 101 110 001	a-d-e-b-c	141
6	010 100 010 011 011	c-e-d-a-b	197
7	100 100 011 111 110	e-a-d-b-c	222
8	010 101 011 011 001	c-d-e-a-b	193
9	100 110 101 011 110	e-a-b-d-c	224
10	111 011 101 100 100	a-d-e-b-c	141

# Crossover operator

The most popular crossover selects any two solution strings randomly from the mating pool and some portion of the strings is exchanged between the strings.

The selection point is selected randomly.

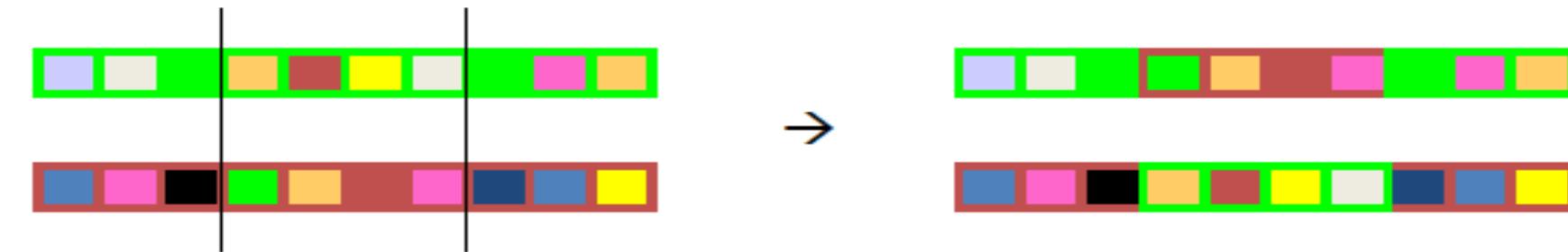
A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.

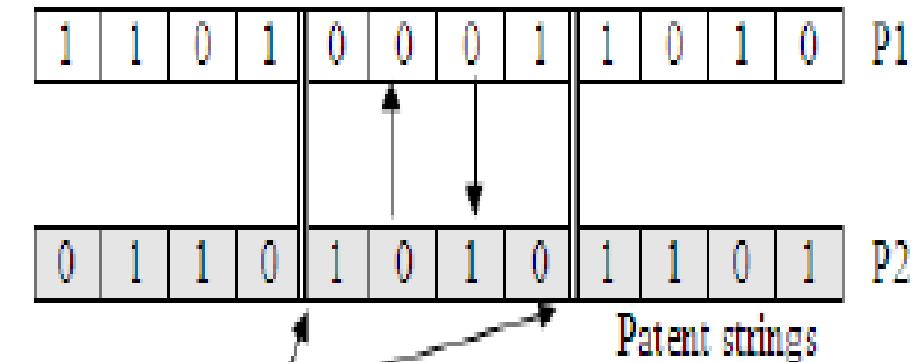
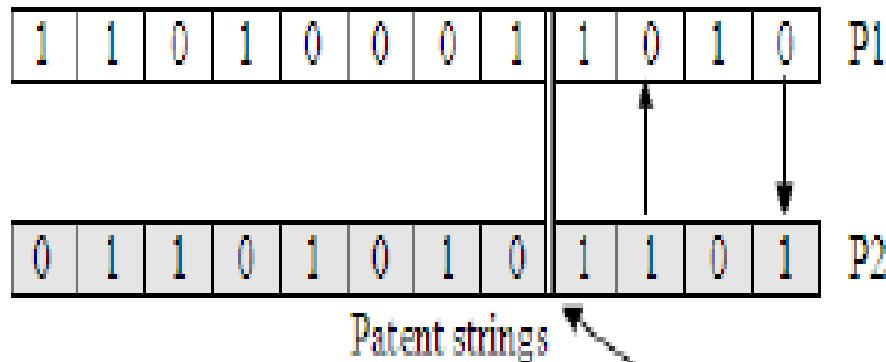


- Single point crossover



- Two point crossover (Multi point crossover)





# Mutation operator

Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population.

Though crossover has the main responsibility to search for the optimal solution, mutation is also used for this purpose.



Before mutation



After mutation

Mutation operator changes a 1 to 0 or vise versa, with a mutation probability of .

The mutation probability is generally kept low for steady convergence.

A high value of mutation probability would search here and there like a random search technique.

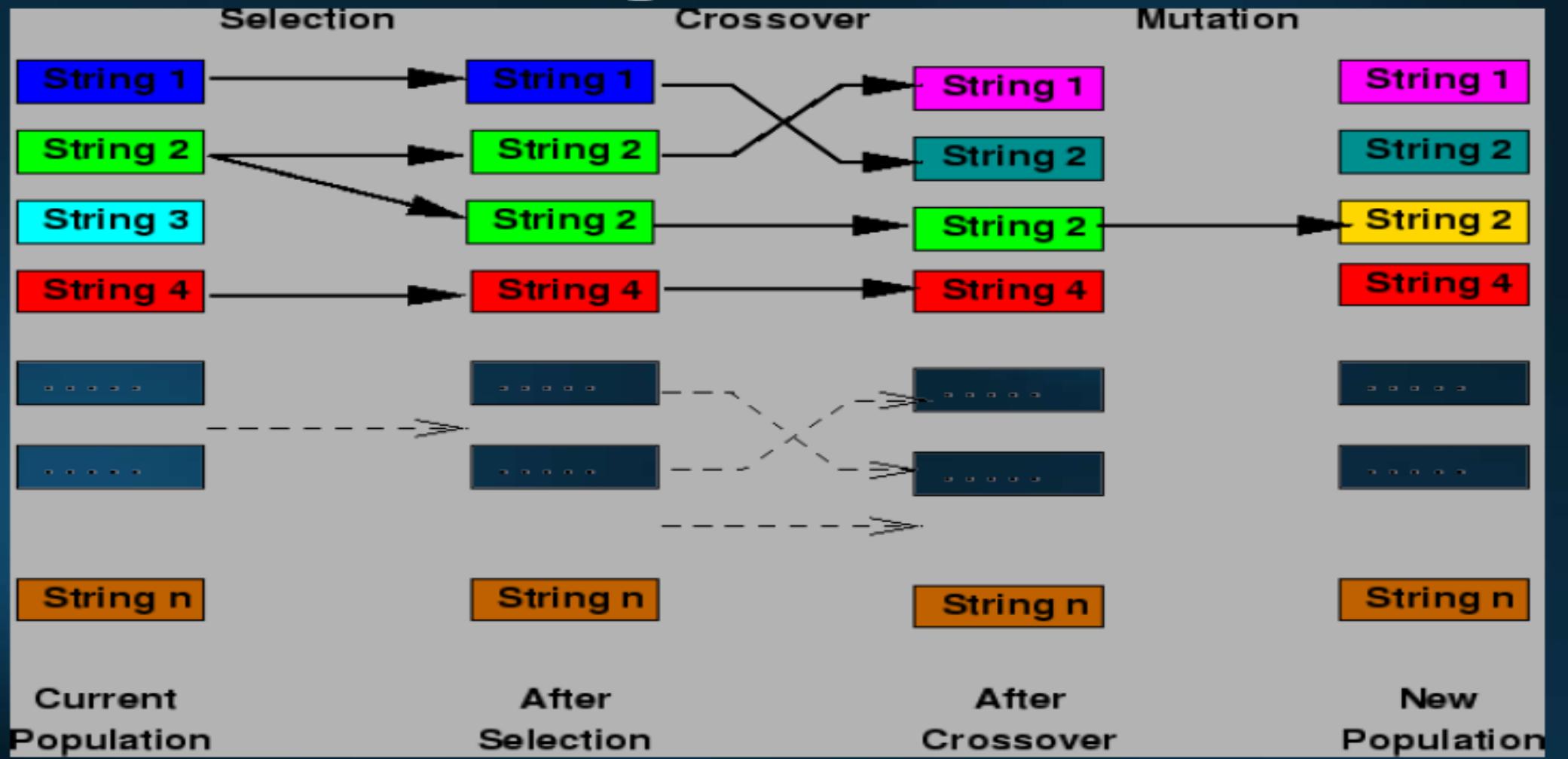
# Elitism

Crossover and mutation may destroy the best solution of the population pool

Elitism is the preservation of few best solutions of the population pool

Elitism is defined in percentage or in number

# Working Principle of Genetic Algorithm

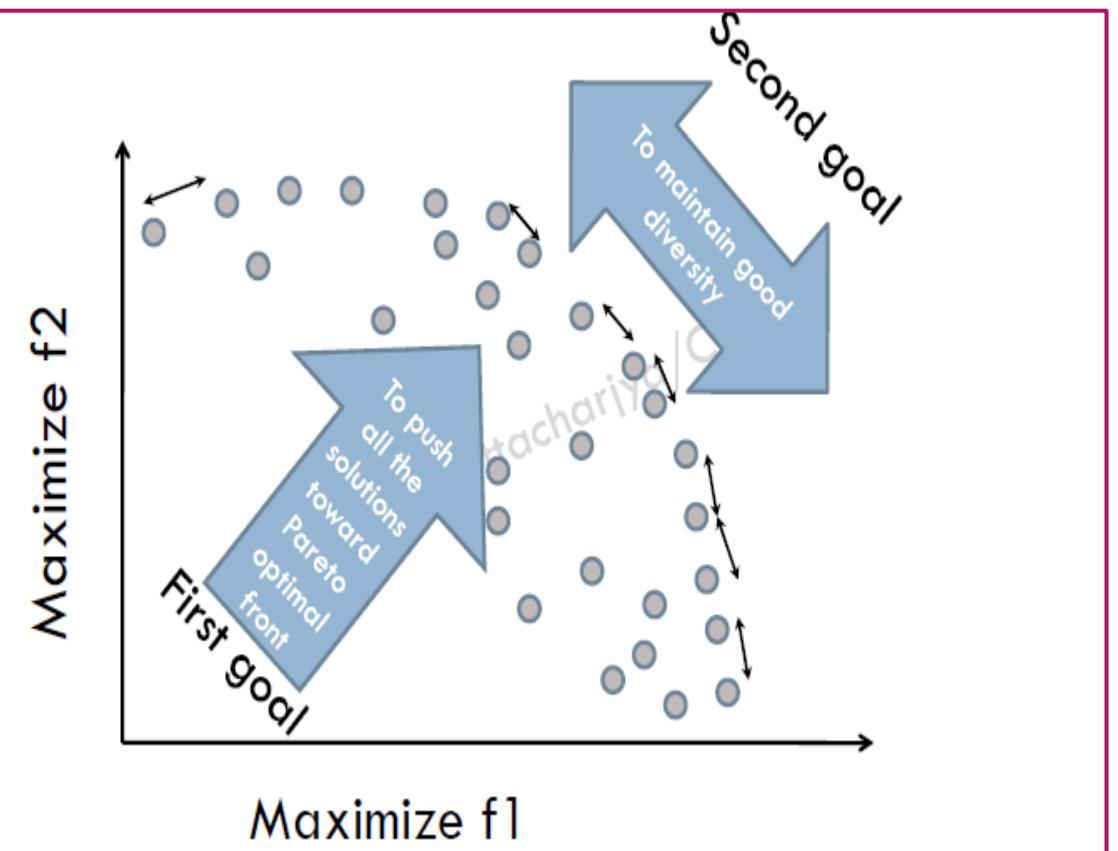
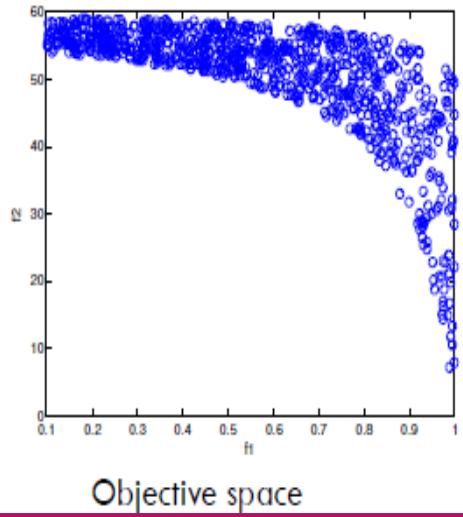
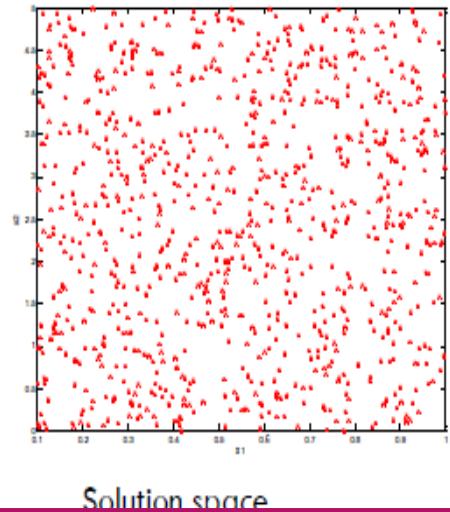


# Multi-objective optimization

- More than one objectives
- Objectives are conflicting in nature
- Dealing with two search space
- Decision variable space
- Objective space
- Unique mapping between the objectives and often the mapping is non-linear
- Properties of the two search space are not similar
- Proximity of two solutions in one search space does not mean a proximity in other search space

# Multiple objective genetic algorithm (MOGA)

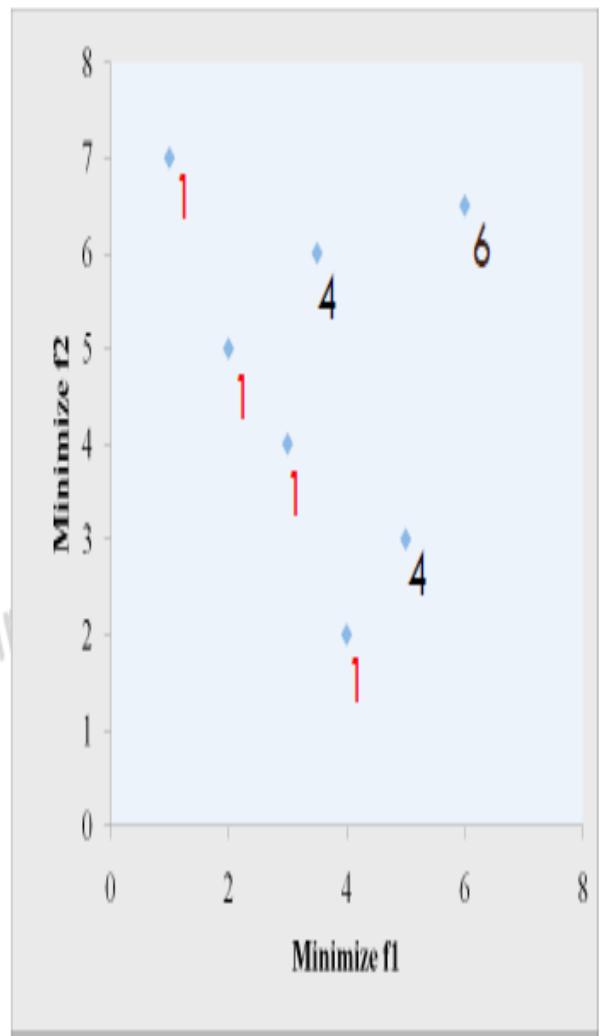
Maximize  $f_1 = 1.1 - x_1$   
Maximize  $f_2 = 60 - \frac{1+x_1}{x_2}$   
Subject to  $0.1 \leq x_1 \leq 1$   
 $0 \leq x_2 \leq 5$



Fonseca and Fleming (1993) first introduced multiple objective genetic algorithm (MOGA)

The assigned fitness value based on the non-dominated ranking.

The rank is assigned as  $r_i = 1 + n_i$ , where  $r_i$  is the ranking of the  $i^{th}$  solution and  $n_i$  is the number of solutions that dominate the solution.



□ Fonseca and Fleming (1993) maintain the diversity among the non-dominated solution using niching among the solution of same rank.

□ The normalize distance was calculated as,

$$d_{i,j} = \sqrt{\sum_{k=1}^M \left( \frac{f_k^i - f_k^j}{f_k^{\max} - f_k^{\min}} \right)^2}$$

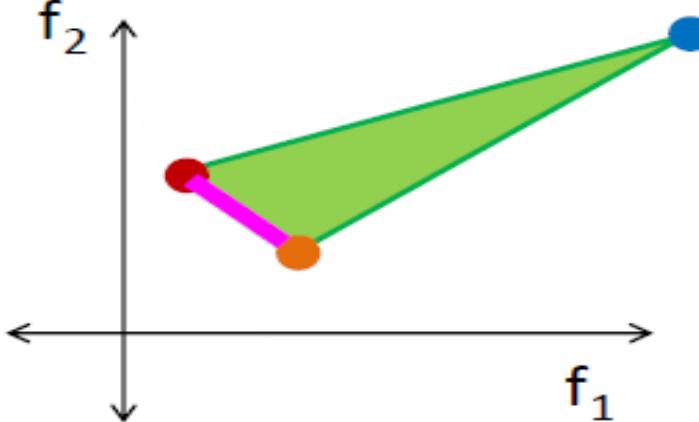
□ The niche count was calculated as,

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij})$$

# Pareto Optimality

In the business example, we were trying to minimize time and cost.

Note that the orange point in criterion space is the lowest value of  $f_2$  (time) and the red point is the lowest value of  $f_1$  (cost). The edge between them is called the *Pareto Front*.



Any point on this front is considered “Pareto optimal”. By moving along the curve, you could minimize cost at the expense of time, or minimize time at the expense of cost, but you can not improve both at once.

## The Pareto-optimal Front

The Pareto-optimal front is defined in terms of dominance relation between solution vectors. Assuming all the objective functions to be minimized, a solution vector  $\bar{x}_1$  is said to dominate another solution vector  $\bar{x}_2$  if all the objective functional values of  $\bar{x}_1$  are less than or equal to those of  $\bar{x}_2$  and there is at least one objective function for which  $\bar{x}_1$  has a value which is strictly less than that of  $\bar{x}_2$ .

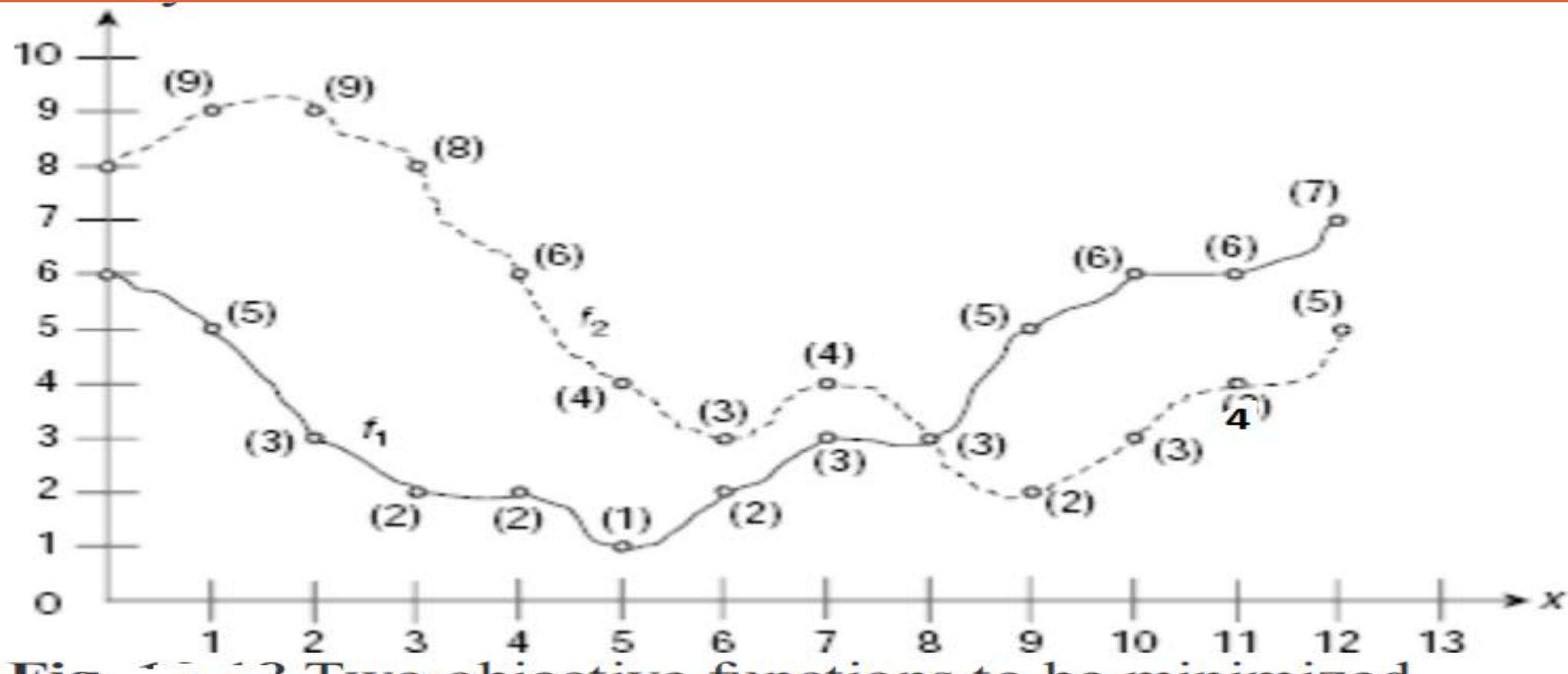


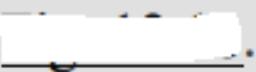
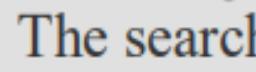
Fig. 17.3 Two objective functions to be minimized

**Definition 12.1** (*Dominance relation between solution vectors*) Let  $\vec{x}$  denote the vector of objective functions of a MOO problem with  $K$  objective functions. Without loss of generality we assume that the functions  $f_1(\vec{x}), f_2(\vec{x}), \dots, f_K(\vec{x})$  all need to be minimized. We say that solution  $\vec{x}_1$  dominates another solution  $\vec{x}_2$ , written as  $\vec{x}_1 \prec \vec{x}_2$ , if and only if

$$f_i(\vec{x}_1) \leq f_i(\vec{x}_2) \quad \forall i \in \{1, 2, \dots, K\}, \text{ and}$$

$$\exists i \in \{1, 2, \dots, K\} \ni f_i(\vec{x}_1) < f_i(\vec{x}_2)$$

### **Example**      (*Dominance relation between solution vectors*)

Let us consider an MOO with two objective functions  $f_1$  and  $f_2$  both to be minimized. For simplicity we assume that the solution vector consists of a single parameter  $x$ . The shapes of  $f_1$  and  $f_2$  plotted against  $x$  are shown in . The search process produces sample solutions at  $x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ . It is evident from  that the minima for  $f_1$  and  $f_2$  do not coincide. In fact  $f_1$  attains its minimum value of  $f_1(x) = 1$  at  $x = 5$ , while  $f_2(x)$  attains its minimal value of 2 at  $x = 9$ .  shows positions of the objective function vector  $f(x) = (f_1(x), f_2(x))$  corresponding to various values of  $x \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ . Now consider the points  $e (3, 3)$  and  $g (3, 4)$ . Here  $e$  dominates  $g$ . Similarly  $g$  dominates  $i, j, k, l$  and  $m$ . However,  $d (2, 6)$  and  $e (3, 3)$  are non-dominating with respect to each other. In fact, the points in the set  $\{a, b, c\}$  are mutually non-dominating. Similarly  $\{d, e\}$ ,  $\{f, g, h\}$ ,  $\{i, j\}$ , and  $\{k, l, m\}$  are non-dominating sets of solutions.

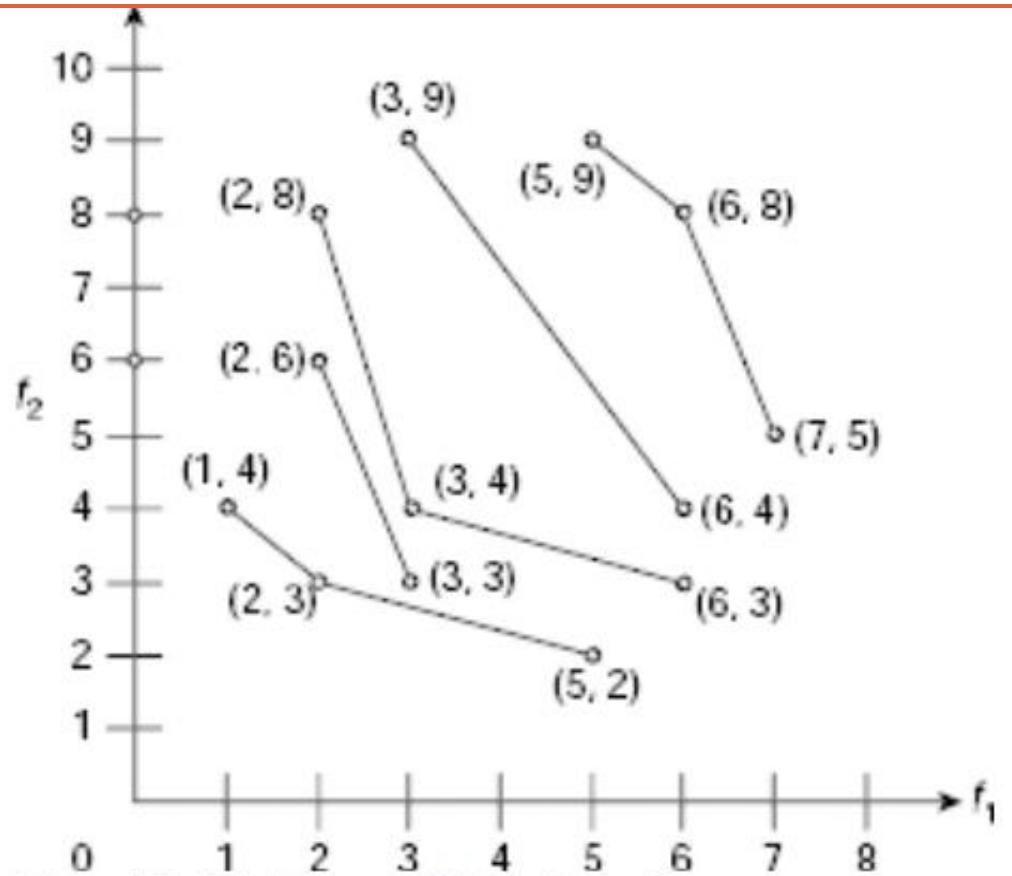


Fig. 12.19 Two-objective function vectors

**Definition 12.2 (Pareto-optimal Front)** Let  $P$  be a population of solutions to an MOO problem. The pareto-optimal front,  $F_{PO}$  is the set of all non-dominated candidates of  $P$ .

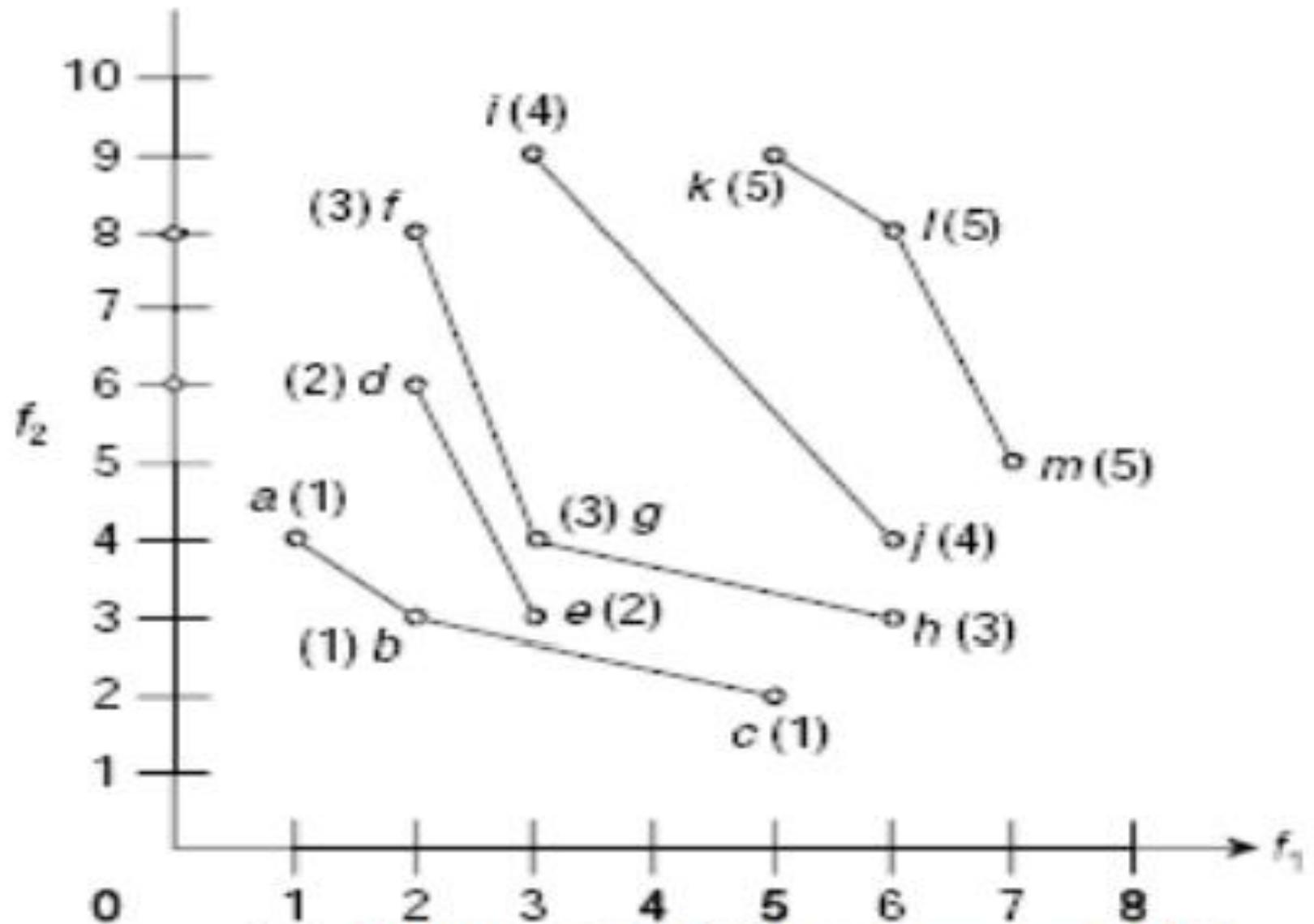
$$F_{PO} = \{ \vec{x} \mid \neg \exists \vec{y} \exists (f(\vec{y}) \prec f(\vec{x})) \}$$

The pareto-ranking method described above is often improvised to obtain other ranking schemes. For example, Fonseca and Fleming (1993) followed a ranking method that penalizes solutions located in the regions of the objective function space which are dominated, or covered, by densely populated sections of the pareto-fronts. They used the formula

$$r(\vec{x}) = 1 + nd(\vec{x}) \quad (12.3)$$

Here  $nd(\vec{x})$  is the number of solutions who dominate  $\vec{x}$ .

$$\begin{aligned} F_{P01} &= \{a(1, 4), b(2, 3), c(5, 2)\}, F_{P02} = \{d(2, 6), e(3, 3)\}, \\ F_{P03} &= \{f(2, 8), g(3, 4), h(6, 3)\}, F_{P04} = \{i(3, 9), j(6, 4)\}, \\ F_{P05} &= \{k(5, 9), l(6, 8), m(7, 5)\} \end{aligned}$$



Pareto-Ranking as per Goldberg's method

- Between every pair of solutions  $\vec{x}, \vec{y} \in P$ , calculate the Euclidean distance  $df(\vec{x}, \vec{y})$  in the normalized objective space using the formula

$$df(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^K \left( \frac{f_i(\vec{x}) - f_i(\vec{y})}{f_i^{\max} - f_i^{\min}} \right)^2} \quad (12.6)$$

In Formula 12.6,  $f_i^{\max}$  and  $f_i^{\min}$  are the maximum and the minimum values of the objective function  $f_i$  observed so far in the search.

- For each  $\vec{x} \in P$ , calculate the niche count  $nc(\vec{x})$  using the formula

$$nc(\vec{x}) = \sum_{\substack{r(\vec{x})=r(\vec{y}) \\ df(\vec{x}, \vec{y}) \leq \sigma}} \frac{\sigma - df(\vec{x}, \vec{y})}{\sigma} \quad (12.7)$$

Here  $\sigma$  is the size of the niche. The value of  $\sigma$  is to be supplied by the user.

Fitness of a chromosome is initially calculated with the help of Formula 12.8 given below.

$$f(\vec{x}) = PopSize - \sum_{i=1}^{r(\vec{x})-1} n_i - \frac{n_{r(\vec{x})} - 1}{2} \quad (12.8)$$

In Formula 12.8,  $n_i$  is the size of the  $i^{\text{th}}$  pareto front, and  $r(\vec{x})$  is the rank of  $\vec{x}$ . Then *shared* fitness of an individual solution is computed using the niche count as per Formula 12.9.

$$f'(\vec{x}) \leftarrow \frac{f(\vec{x})}{nc(\vec{x})} \quad (12.9)$$

Finally, the *normalized* fitness values are calculated using the shared fitness with the help of Formula 12.10.

$$f''(\vec{x}) = \frac{f'(\vec{x}) \times n_{r(\vec{x})}}{\sum_{\substack{\vec{y} \in P \\ r(\vec{y})=r(\vec{x})}} f'(\vec{y})} \times f(\vec{x}) \quad (12.10)$$

### **Procedure Multi-Objective-GA**

- Step 1.** Generate the initial population randomly.
- Step 2.** Determine the pareto-optimal fronts  $F_{\text{P}01}, F_{\text{P}02}, \dots, F_{\text{P}0K}$ .
- Step 3.** If stopping criteria is satisfied then Return the pareto-optimal front  $F_{\text{P}01}$  and **Stop**.
- Step 4.** For each solution  $\bar{x} \in P$ , evaluate the fitness as follows:
  - Step 4.1.** Assign a rank  $r(\bar{x})$  using Goldberg's method, or Formula 12.3 (Fonseca and Fleming), or Formula 12.4 (Lu and Yen).
  - Step 4.2.** Compute the basic fitness value using Formula 12.8.
  - Step 4.3.** Compute the shared fitness value using Formula 12.9.
  - Step 4.4.** Compute the normalized fitness value using Formula 12.10.
- Step 5.** Generate the mating pool  $MP$  from population  $P$  applying appropriate selection operator.
- Step 6.** Apply crossover and mutation operations on the chromosomes of the mating pool to produce the next generation  $P'$  of population from  $MP$ .
- Step 7.** Replace the old generation of population  $P$  by the new generation of population  $P'$ .
- Step 8.** Goto Step 2.

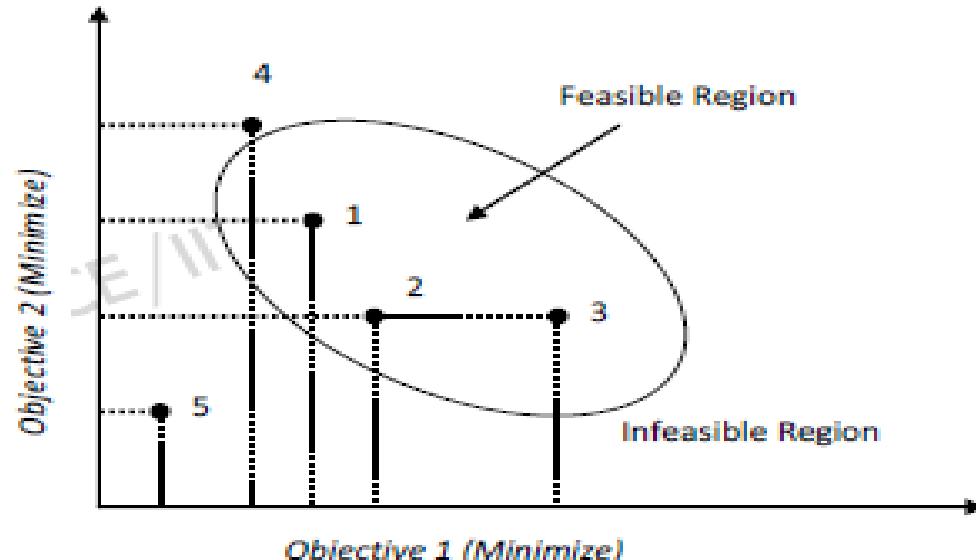
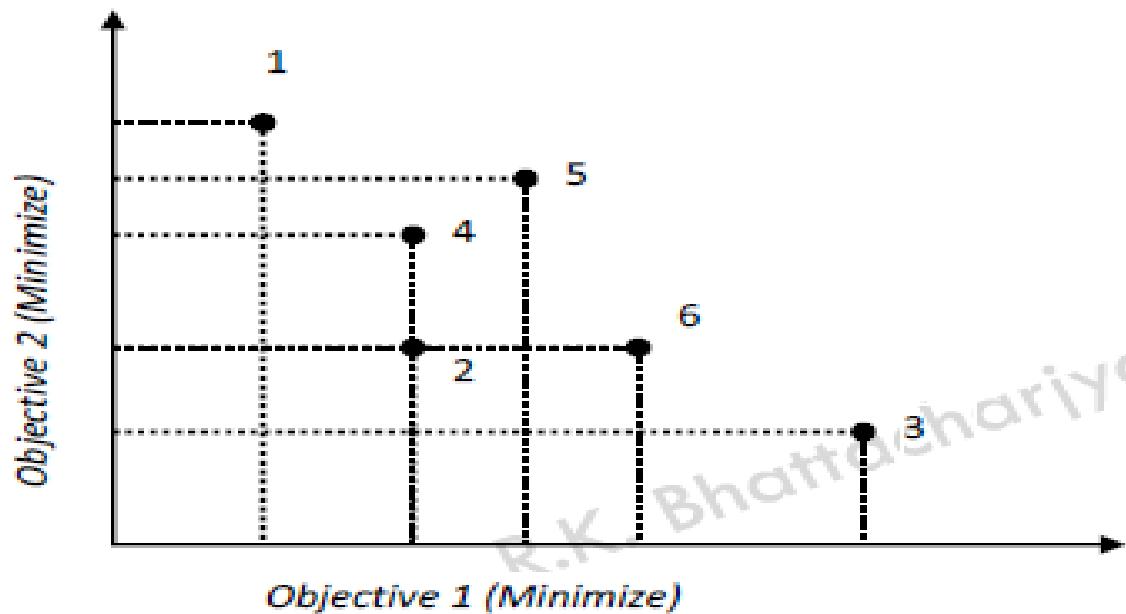
## Non-dominated sorting

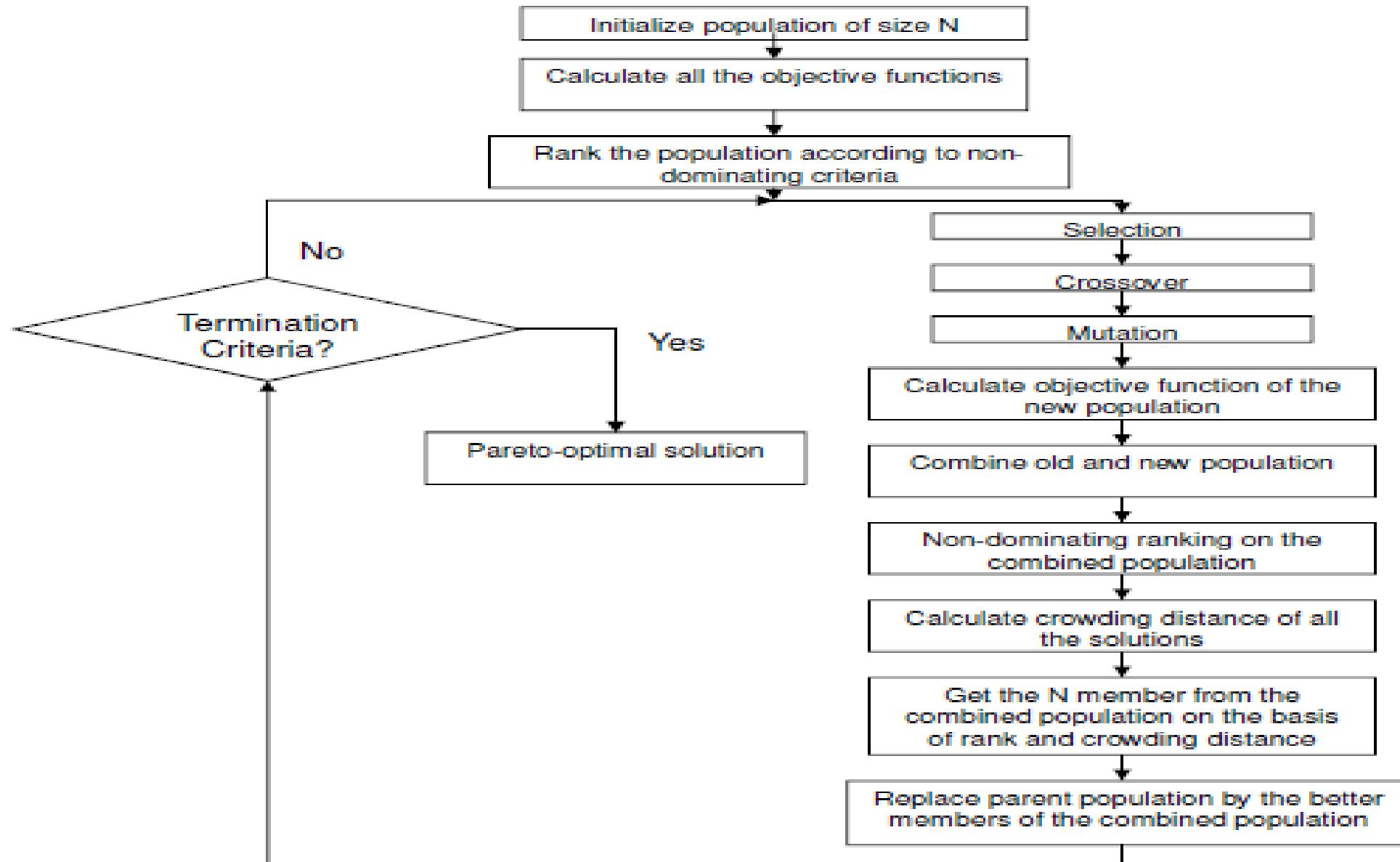
- Srinivas and Deb (1994) proposed NSGA
- The algorithm is based on the non-dominated sorting.
- The spread on the Pareto optimal front is maintained using sharing function

$$d_{i,j} = \sqrt{\sum_{k=1}^{P_1} \left( \frac{x_k^i - x_k^j}{x_k^{max} - x_k^{min}} \right)^2}$$

**Non-dominated Sorting Genetic Algorithms:**  
NSGA II is an elitist non-dominated sorting Genetic Algorithm to solve multi-objective optimization problem developed by Prof. K. Deb and his student at IIT Kanpur.

It has been reported that NSGA II can converge to the global Pareto-optimal front and can maintain the diversity of population on the Pareto-optimal front

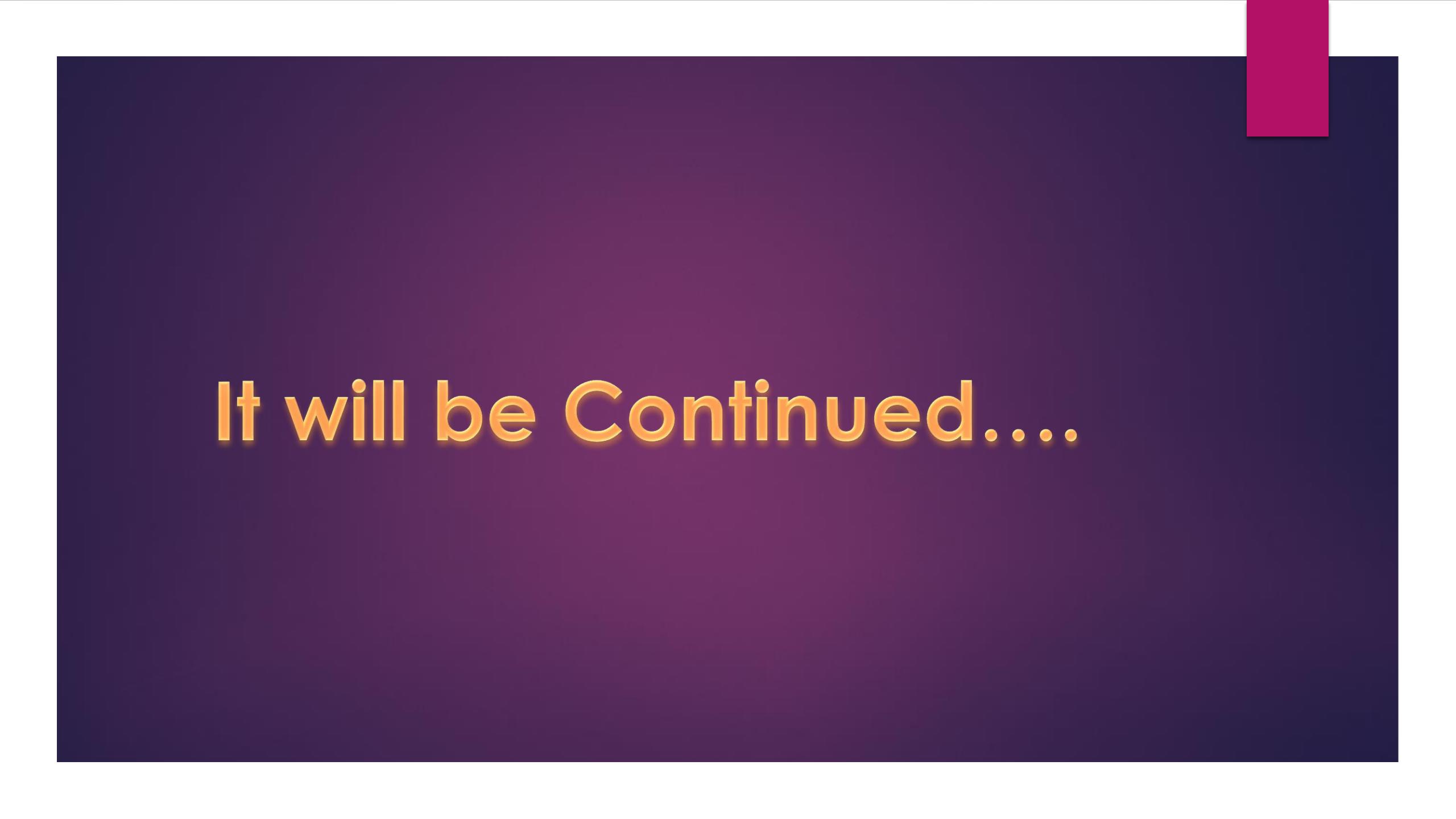






# Genetic Algorithm Applications

Domains	Application Types
Control	Gas pipeline, pole balancing, missile evasion, pursuit
Robotics	Trajectory planning
Signal Processing	Filter design
Game Playing	Poker, checker, prisoner's dilemma
Scheduling	Manufacturing facility, scheduling, resource allocation
Design	Semiconductor layout, aircraft design, keyboard configuration, communication networks
Combinatorial Optimisation	Set covering, travelling salesmen, routing, bin packing, graph coloring and partitioning



**It will be Continued....**

# Expert System

BY

DR. ANUPAM GHOSH

DATE: 21.12.2023

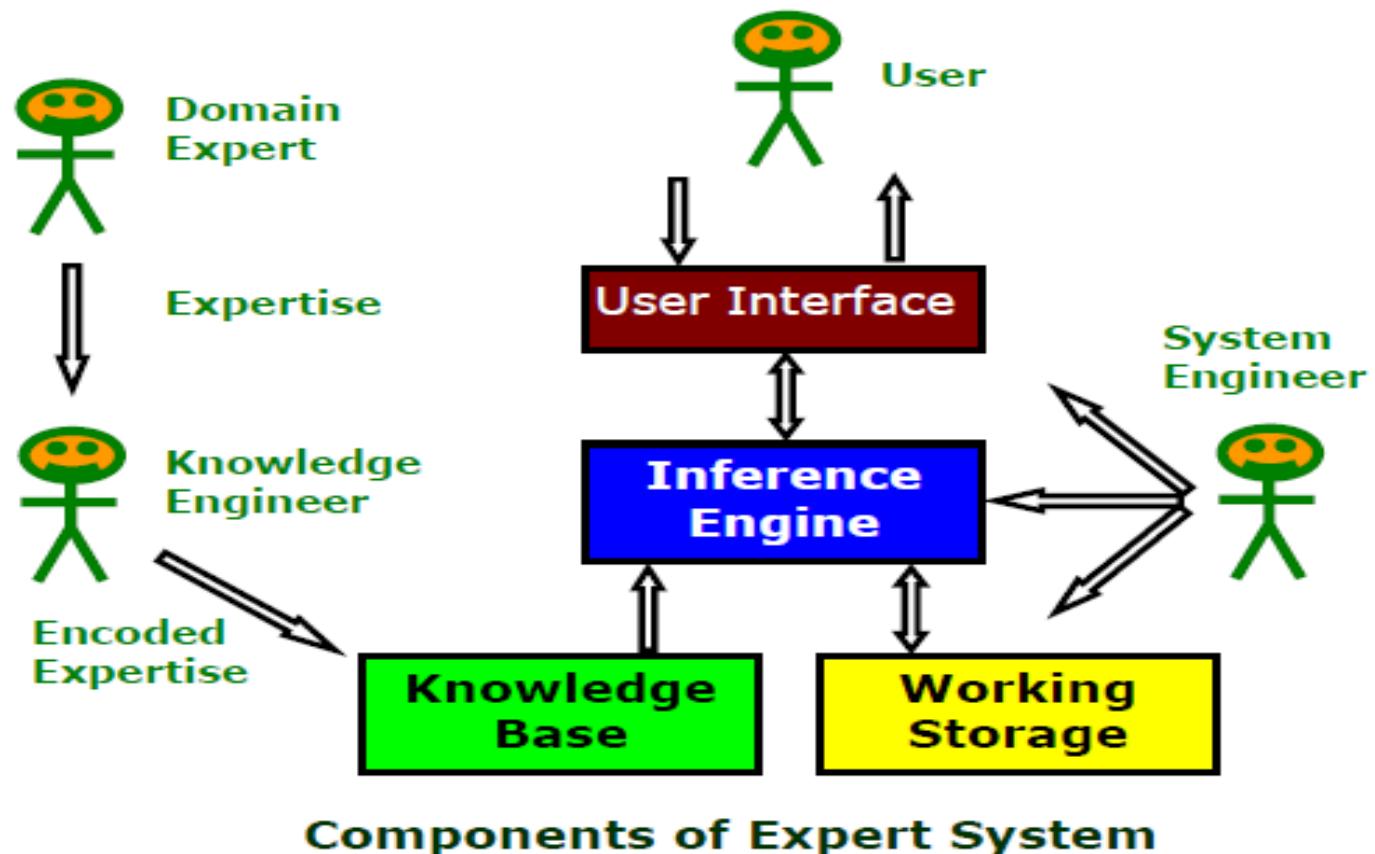
# Definition

- An expert system, is an interactive computer-based decision tool that uses both facts and heuristics to solve difficult decision making problems, based on knowledge acquired from an expert.
- An expert system is a model and associated procedure that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert.
- An expert system compared with traditional computer :

Inference engine + Knowledge = Expert system  
( Algorithm + Data structures = Program in traditional computer )
- First expert system, called DENDRAL, was developed in the early 70's at Stanford University.

## Expert System Components And Human Interfaces

Expert systems have a number of major system components and interface with individuals who interact with the system in various roles. These are illustrated below.



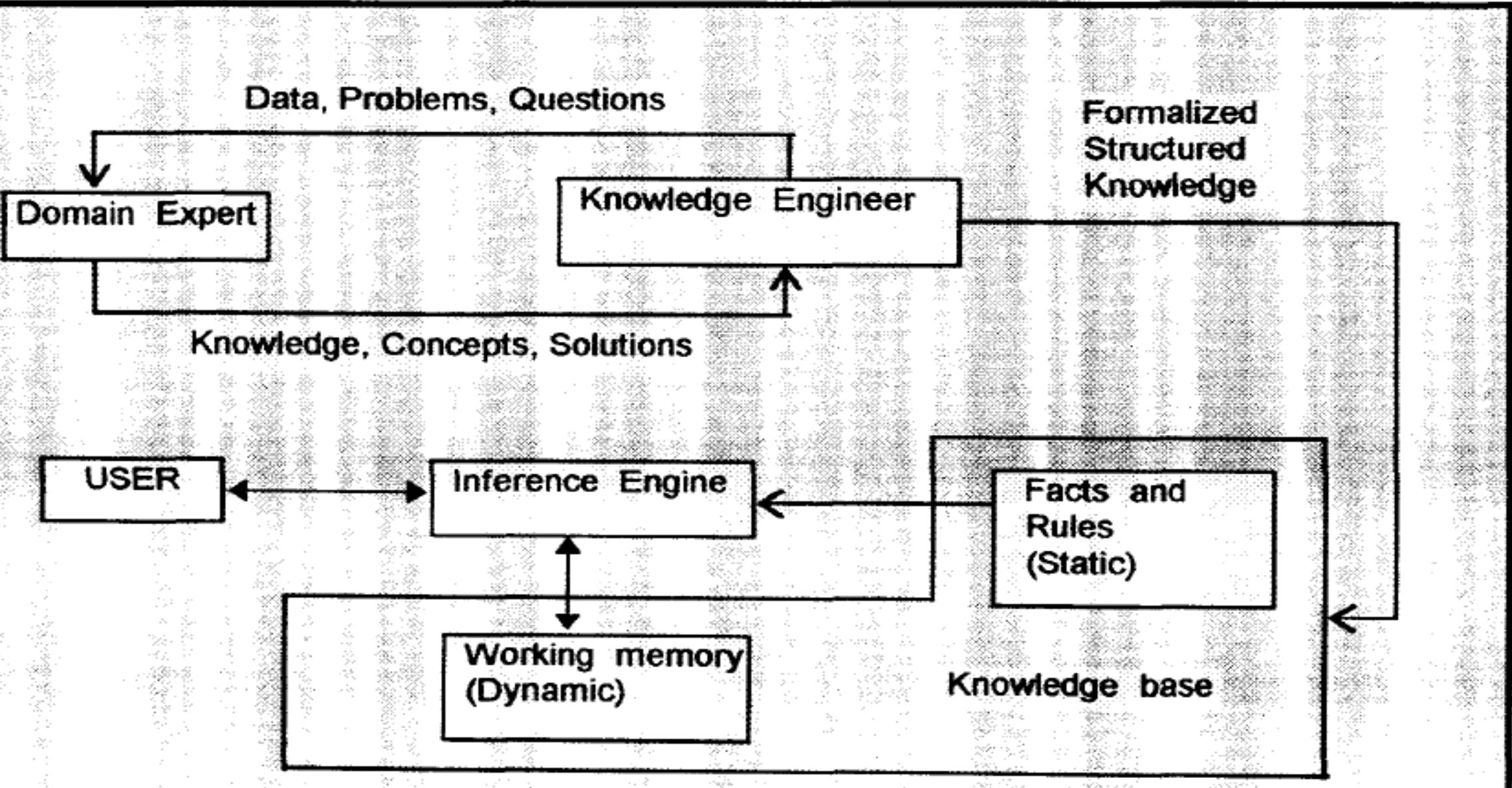
## ■ Components and Interfaces

- # **Knowledge base** : A declarative representation of the expertise; often in IF THEN rules ;
- # **Working storage** : The data which is specific to a problem being solved;
- # **Inference engine** : The code at the core of the system which derives recommendations from the knowledge base and problem-specific data in working storage;
- # **User interface** : The code that controls the dialog between the user and the system.

## ■ Roles of Individuals who interact with the system

- # **Domain expert** : The individuals who currently are experts 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;
- # **User** : The individual who will be consulting with the system to get advice which would have been provided by the expert.

# Architecture of a typical Expert system



## ■ Expert System Shells

Many expert systems are built with products called expert system shells. A shell is a piece of software which contains the user interface, a format for declarative knowledge in the knowledge base, and an inference engine. The knowledge and system engineers uses these shells in making expert systems.

- # **Knowledge engineer** : uses the shell to build a system for a particular problem domain.
- # **System engineer** : builds the user interface, designs the declarative format of the knowledge base, and implements the inference engine.

Depending on the size of the system, the knowledge engineer and the system engineer might be the same person.

# Characteristics

## ■ Operates as an interactive system

This means an expert system :

- # Responds to questions
- # Asks for clarifications
- # Makes recommendations
- # Aids the decision-making process.

## ■ Tools have ability to sift (filter) knowledge

- # Storage and retrieval of knowledge
- # Mechanisms to expand and update knowledge base on a continuing basis.

## ■ Make logical inferences based on knowledge stored

- # Simple reasoning mechanisms is used
- # Knowledge base must have means of exploiting the knowledge stored, else it is useless; e.g., learning all the words in a language, without knowing how to combine those words to form a meaningful sentence.

## ■ Ability to Explain Reasoning

- # Remembers logical chain of reasoning; therefore user may ask
  - ◊ for explanation of a recommendation
  - ◊ factors considered in recommendation
- # Enhances user confidence in recommendation and acceptance of expert system

## ■ Domain-Specific

- # A particular system caters a narrow area of specialization; e.g., a medical expert system cannot be used to find faults in an electrical circuit.
- # Quality of advice offered by an expert system is dependent on the amount of knowledge stored.

## ■ Capability to assign Confidence Values

- # Can deliver quantitative information
- # Can interpret qualitatively derived values
- # Can address imprecise and incomplete data through assignment of confidence values.

## **Expert System Features**

The features which commonly exist in expert systems are :

- Goal Driven Reasoning or Backward Chaining**

An inference technique which uses IF-THEN rules to repetitively break a goal into smaller sub-goals which are easier to prove;

- Coping with Uncertainty**

The ability of the system to reason with rules and data which are not precisely known;

- Data Driven Reasoning or Forward Chaining**

An inference technique which uses IF-THEN rules to deduce a problem solution from initial data;

- Data Representation**

The way in which the problem specific data in the system is stored and accessed;

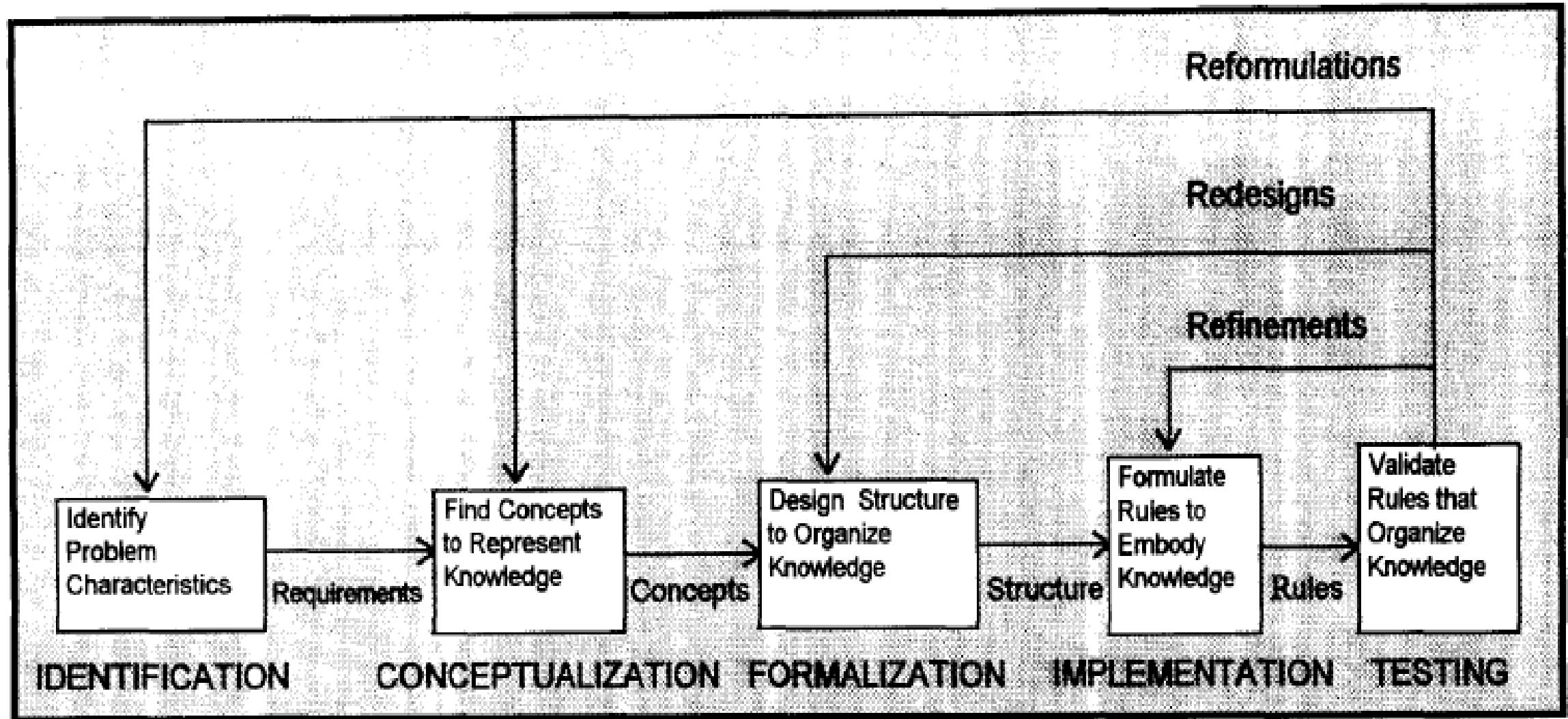
- User Interface**

That portion of the code which creates an easy to use system;

- Explanations**

The ability of the system to explain the reasoning process that it used to reach a recommendation.

# Major Stages of Expert System



# Applications

## # Diagnosis and Troubleshooting of Devices and Systems

Medical diagnosis was one of the first knowledge areas to which Expert system technology was applied in 1976. However, the diagnosis of engineering systems quickly surpassed medical diagnosis.

## # Planning and Scheduling

The Expert system's commercial potential in planning and scheduling has been recognized as very large. Examples are airlines scheduling their flights, personnel, and gates; the manufacturing process planning and job scheduling;

## # Configuration of Manufactured Objects from sub-assemblies

Configuration problems are synthesized from a given set of elements related by a set of constraints. The Expert systems have been very useful to find solutions. For example, modular home building and manufacturing involving complex engineering design.

## # Financial Decision Making

The financial services are the vigorous user of expert system techniques. Advisory programs have been created to assist bankers in determining whether to make loans to businesses and individuals. Insurance companies to assess the risk presented by the customer and to determine a price for the insurance. ES are used in typical applications in the financial markets / foreign exchange trading.

## # Knowledge Publishing

This is relatively new, but also potentially explosive area. Here the primary function of the Expert system is to deliver knowledge that is relevant to the user's problem. The two most widely known Expert systems are : one, an **advisor** on appropriate grammatical usage in a text; and the other, is a **tax advisor** on tax strategy, tactics, and individual tax policy.

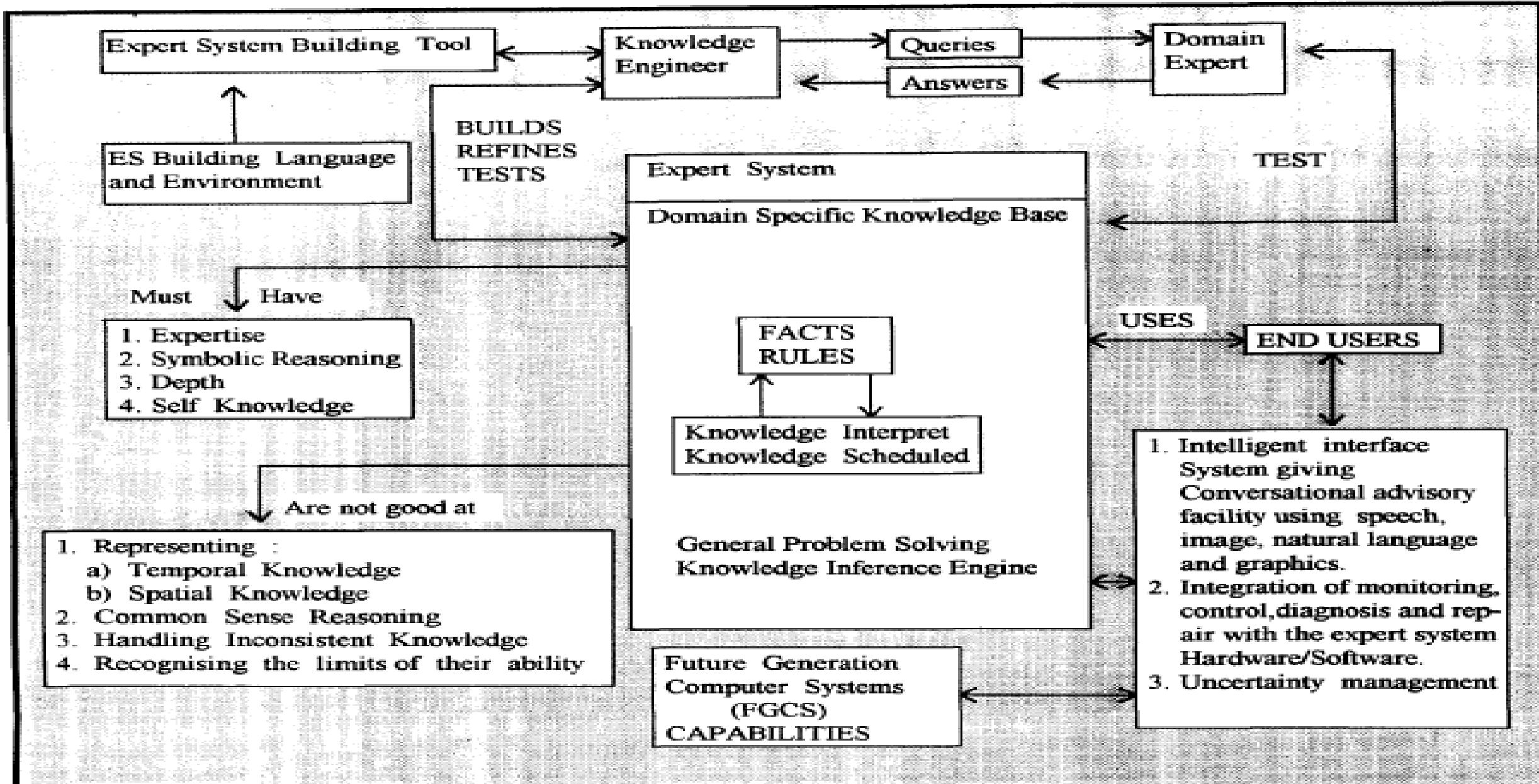
## # Process Monitoring and Control

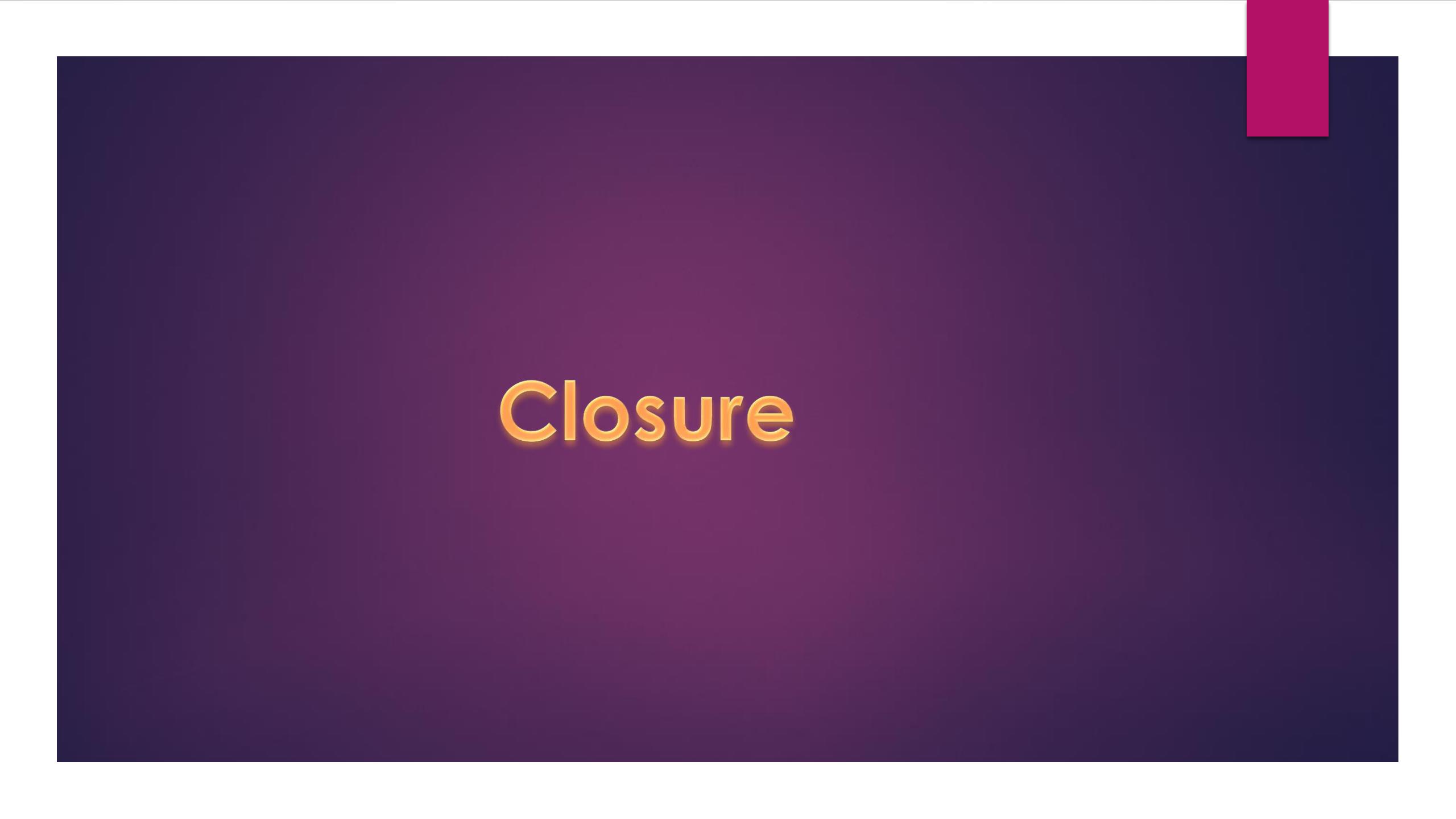
Here Expert system does analysis of real-time data from physical devices, looking for anomalies, predicting trends, controlling optimality and failure correction. Examples of real-time systems that actively monitor processes are found in the steel making and oil refining industries.

## # Design and Manufacturing

Here the Expert systems assist in the design of physical devices and processes, ranging from high-level conceptual design of abstract entities all the way to factory floor configuration of manufacturing processes.

## Block diagram indicating Expert System development environment





# Closure