CBCS SCHEME

USN | 1 | C | R | 2 | 1 | C | S | 1 | 4 | 6 | **21CS54**

Fifth Semester B.E. Degree Examination, Dec.2023/Jan.2024
Artificial Intelligence and Machine Learning

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Identify the Turing test approach to provide a satisfactory operational definition of Intelligence. **(04 Marks)**

b. Make use of the state space of the vacuum world and define the components to solve this problem. **(06 Marks)**

c. Illustrate the properties and the algorithm for Breadth-first search technique. **(10 Marks)**

OR

2 a. Explain the concepts of thinking rationally and acting rationally. **(04 Marks)**

b. Explain the tree search and graph search algorithms. **(06 Marks)**

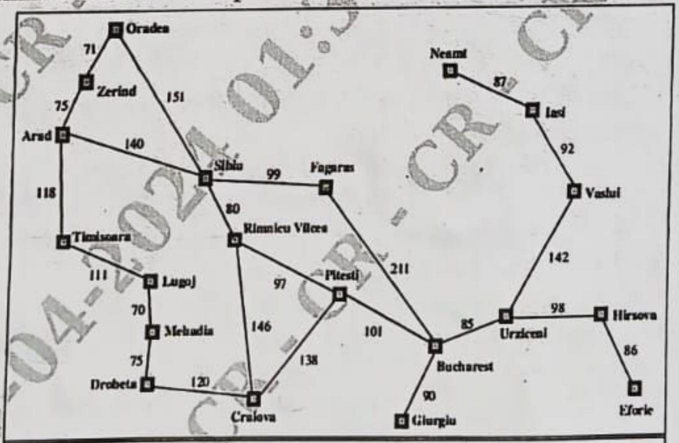
c. Explain problem solving agents alongwith the algorithm and illustrate the incremental formulation of 8-Queens problem. **(10 Marks)**

Module-2

3 a. Identify the differences between supervised and unsupervised learning. **(04 Marks)**

b. Explain the types of Big data. **(06 Marks)**

c. Apply A* algorithm to find the best path from Arad to Bucharest. [Refer Fig.Q3(c)].



A simplified road map of part of Romania.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Values of h_{SLD}—straight-line distances to Bucharest.

Fig.Q3(c) **(10 Marks)**

OR

4 a. Explain the machine learning process model along with diagram. **(06 Marks)**

b. Consider the table given below which contains the machine learning course registration done by both boys and girls. There are 50 boys and 50 girls in the class and the registration of the course is given in the table. Apply Chi-square test and find out whether any differences exist between boys and girls for course registration.

Table 4(b)

Gender	Registered	Not Registered	Total
Boys	35	15	50
Girls	25	25	50
Total	60	40	100

(06 Marks)

c. Apply the heuristic search algorithm on the given 8 puzzle problem to reach the goal state from the given initial state.

Initial State			Final State		
1	2	3	1	2	3
	4	6	4	5	6
7	5	8	7	8	

Fig.Q4(c) **(08 Marks)**

Module-3

5 a. Consider the training dataset of 4 instances shown in the table below. Apply Find-S algorithm to find the final hypothesis.

Table 5(a)

CGPA	Interactiveness	Practical Knowledge	Communication Skills	Logical Thinking	Interest	Job Offer
≥9	Yes	Excellent	Good	Fast	Yes	Yes
≥9	Yes	Good	Good	Fast	Yes	Yes
≥8	No	Good	Good	Fast	No	No
≥9	Yes	Good	Good	Slow	No	Yes

(08 Marks)

b. Explain why Instance based learners are called lazy learners and compare instance based learning and model based learning. **(06 Marks)**

c. Explain the types of Regression methods with diagram. **(06 Marks)**

OR

6 a. Consider the student performance training dataset of 8 data instances in the below table. Based on the performance of a student, classify the test instance (6.1, 40, 5) to check whether the student will pass or fail in that course using KNN approach (K = 3).

Table 6(a)

S.No.	CGPA	Assessment	Project Submitted	Result
1	9.2	85	8	Pass
2	8	80	7	Pass
3	8.5	81	8	Pass
4	6	45	5	Fail
5	6.5	50	4	Fail
6	8.2	72	7	Pass
7	5.8	38	5	Fail
8	8.9	91	9	Pass

(12 Marks)

b. Explain version space and the candidate elimination algorithm explaining the algorithm steps. **(08 Marks)**

Module-4

7 a. Explain the advantages and disadvantages of decision trees. **(06 Marks)**

b. Explain validating and pruning of decision trees. **(06 Marks)**

c. Explain Bayes optimal classifier and solve to find whether a patient is diagnosed as COVID positive or COVID negative using the table given below.

Table 7(c)

$P(h_i/T)$	P(COVID positive)	P(COVID negative/h_i)
0.3	0	1
0.1	1	0
0.2	1	0
0.1	1	0

(08 Marks)

OR

8 a. Explain the procedure to construct a decision tree using ID3 algorithm. **(06 Marks)**

b. Explain Bayes theorem, Maximum A Posteriori (MAP) Hypothesis (h_{MAP}) and Maximum Likelihood (ML) Hypothesis (h_{ML}). **(06 Marks)**

c. Illustrate the algorithm of Naïve Bayes and explain the popular variants of Bayesian classifier. **(08 Marks)**

Module-5

9 a. Explain the different activation functions used in ANN. **(06 Marks)**

b. Illustrate the various types of Artificial Neural Networks. **(08 Marks)**

c. Illustrate the applications and challenges of Clustering algorithms. **(06 Marks)**

OR

10 a. Explain the perceptron model and the algorithm. **(08 Marks)**

b. Consider the following set of data given in the below table. Cluster it using K-means algorithm with the initial value of objects 2 and 5 with the coordinate values (4, 6) and (12, 4) as initial seeds.

Table 10(b)

Objects	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

(12 Marks)

* * * * *

21CS54 – Artificial Intelligence and Machine Learning

5th sem – B.E – Dec. 2023 / Jan. 2024

Time : 3 Hrs Max. Marks : 100

Scheme of Valuation

Module – 1

1.(a) Identify the Turing test approach to provide a satisfactory operational definition of Intelligence. (4)

Ans:

– Acting humanly – Turing test approach

– Interrogator can't differentiate human's and computer's response. – Pass the test.

– Required capabilities for computer:

1. Natural language processing
2. Knowledge representation
3. Automated reasoning
4. Machine learning
5. Computer vision
6. Robotics

(1b) Make use of the state space of the vaccium world and define the components to solve this problem. (6)

Ans:

States : – agent location and dirt location

Agent's location:

– location might or might not contain dirt.

$2 \times 2^2 = 8$ possible world states

Initial state:

- Any state

Actions:

- Left, Right and Suck

Transitional model:

- actions have their expected effects

Exceptions:

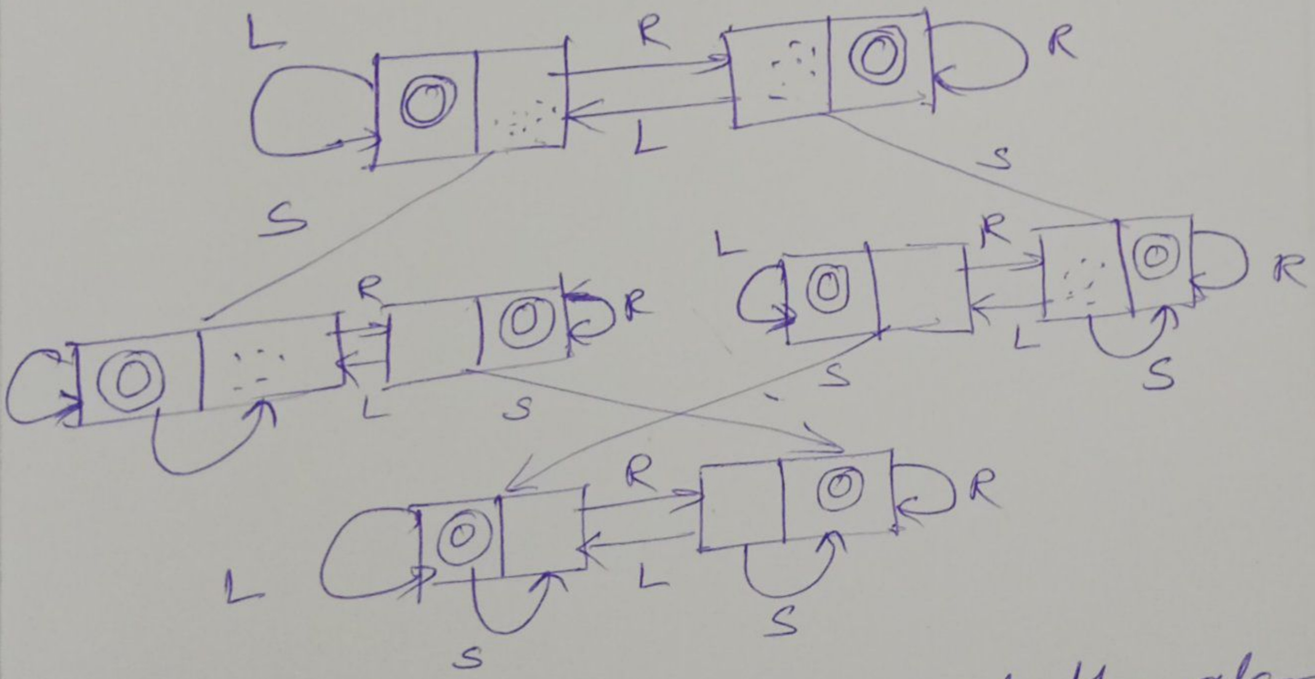
- moving left in the leftmost square.
- moving right in the rightmost square.
- sucking ~~in a~~ in a clean square

Goal test:

- all the squares are clean

Path cost:

- number of steps in the path



the algorithm

1C)

c. Illustrate the properties and the algorithm for Breadth-first search technique. (10 Marks)

3.4.1 Breadth-first search

BREADTH-FIRST SEARCH

Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then *their* successors, and so on. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

Breadth-first search is an instance of the general graph-search algorithm (Figure 3.7) in which the *shallowest* unexpanded node is chosen for expansion. This is achieved very simply by using a FIFO queue for the frontier. Thus, new nodes (which are always deeper than their parents) go to the back of the queue, and old nodes, which are shallower than the new nodes, get expanded first. There is one slight tweak on the general graph-search algorithm, which is that the goal test is applied to each node when it is *generated* rather than when it is selected for expansion. This decision is explained below, where we discuss time complexity. Note also that the algorithm, following the general template for graph search, discards any new path to a state already in the frontier or explored set; it is easy to see that any such path must be at least as deep as the one already found. Thus, breadth-first search always has the shallowest path to every node on the frontier.

Pseudocode is given in Figure 3.11. Figure 3.12 shows the progress of the search on a simple binary tree.

How does breadth-first search rate according to the four criteria from the previous section? We can easily see that it is *complete*—if the shallowest goal node is at some finite depth d, breadth-first search will eventually find it after generating all shallower nodes (provided the branching factor b is finite). Note that as soon as a goal node is generated, we know it is the shallowest goal node because all shallower nodes must have been generated already and failed the goal test. Now, the *shallowest* goal node is not necessarily the *optimal* one;

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
     if EMPTY?(frontier) then return failure
     node ← POP(frontier)  /* chooses the shallowest node in frontier */
     add node.STATE to explored
     for each action in problem.ACTIONS(node.STATE) do
        child ← CHILD-NODE(problem, node, action)
        if child.STATE is not in explored or frontier then
           if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
           frontier ← INSERT(child, frontier)
```

Figure 3.11 Breadth-first search on a graph.

technically, breadth-first search is optimal if the path cost is a nondecreasing function of the depth of the node. The most common such scenario is that all actions have the same cost.

So far, the news about breadth-first search has been good. The news about time and space is not so good. Imagine searching a uniform tree where every state has b successors. The root of the search tree generates b nodes at the first level, each of which generates b more nodes, for a total of b^2 at the second level. Each of *these* generates b more nodes, yielding b^3 nodes at the third level, and so on. Now suppose that the solution is at depth d. In the worst case, it is the last node generated at that level. Then the total number of nodes generated is

$$b + b^2 + b^3 + \cdots + b^d = O(b^d)\ .$$

(If the algorithm were to apply the goal test to nodes when selected for expansion, rather than when generated, the whole layer of nodes at depth d would be expanded before the goal was detected and the time complexity would be $O(b^{d+1})$.)

As for space complexity: for any kind of graph search, which stores every expanded node in the *explored* set, the space complexity is always within a factor of b of the time complexity. For breadth-first graph search in particular, every node generated remains in memory. There will be $O(b^{d-1})$ nodes in the *explored* set and $O(b^d)$ nodes in the frontier,

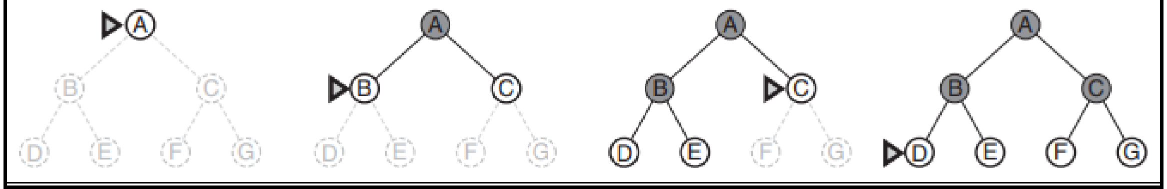


Figure 3.12 Breadth-first search on a simple binary tree. At each stage, the node to be expanded next is indicated by a marker.

so the space complexity is $O(b^d)$, i.e., it is dominated by the size of the frontier. Switching to a tree search would not save much space, and in a state space with many redundant paths, switching could cost a great deal of time.

An exponential complexity bound such as $O(b^d)$ is scary. Figure 3.13 shows why. It lists, for various values of the solution depth d, the time and memory required for a breadth-first search with branching factor $b = 10$. The table assumes that 1 million nodes can be generated per second and that a node requires 1000 bytes of storage. Many search problems fit roughly within these assumptions (give or take a factor of 100) when run on a modern personal computer.

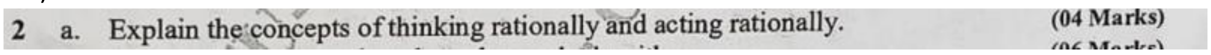
Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

Two lessons can be learned from Figure 3.13. First, *the memory requirements are a bigger problem for breadth-first search than is the execution time.* One might wait 13 days for the solution to an important problem with search depth 12, but no personal computer has the petabyte of memory it would take. Fortunately, other strategies require less memory.

The second lesson is that time is still a major factor. If your problem has a solution at depth 16, then (given our assumptions) it will take about 350 years for breadth-first search (or indeed any uninformed search) to find it. In general, *exponential-complexity search problems cannot be solved by uninformed methods for any but the smallest instances.*

2 a)



2 a. Explain the concepts of thinking rationally and acting rationally. **(04 Marks)**

1.1.3 Thinking rationally: The “laws of thought” approach

The Greek philosopher Aristotle was one of the first to attempt to codify “right thinking,” that is, irrefutable reasoning processes. His **syllogisms** provided patterns for argument structures that always yielded correct conclusions when given correct premises—for example, “Socrates is a man; all men are mortal; therefore, Socrates is mortal.” These laws of thought were supposed to govern the operation of the mind; their study initiated the field called **logic**.

Logicians in the 19th century developed a precise notation for statements about all kinds of objects in the world and the relations among them. (Contrast this with ordinary arithmetic notation, which provides only for statements about *numbers*.) By 1965, programs existed that could, in principle, solve *any* solvable problem described in logical notation. (Although if no solution exists, the program might loop forever.) The so-called **logicist** tradition within artificial intelligence hopes to build on such programs to create intelligent systems.

There are two main obstacles to this approach. First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain. Second, there is a big difference between solving a problem “in principle” and solving it in practice. Even problems with just a few hundred facts can exhaust the computational resources of any computer unless it has some guidance as to which reasoning steps to try first. Although both of these obstacles apply to *any* attempt to build computational reasoning systems, they appeared first in the logicist tradition.

1.1.4 Acting rationally: The rational agent approach

An **agent** is just something that acts (*agent* comes from the Latin *agere*, to do). Of course, all computer programs do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals. A **rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.

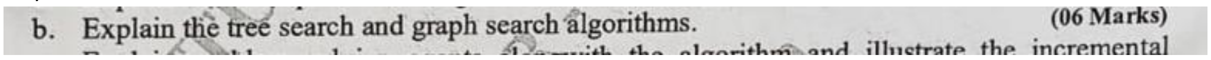
In the “laws of thought” approach to AI, the emphasis was on correct inferences. Making correct inferences is sometimes *part* of being a rational agent, because one way to act rationally is to reason logically to the conclusion that a given action will achieve one’s goals and then to act on that conclusion. On the other hand, correct inference is not *all* of rationality; in some situations, there is no provably correct thing to do, but something must still be done. There are also ways of acting rationally that cannot be said to involve inference. For example, recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.

All the skills needed for the Turing Test also allow an agent to act rationally. Knowledge representation and reasoning enable agents to reach good decisions. We need to be able to generate comprehensible sentences in natural language to get by in a complex society. We need learning not only for erudition, but also because it improves our ability to generate effective behavior.

The rational-agent approach has two advantages over the other approaches. First, it is more general than the “laws of thought” approach because correct inference is just one of several possible mechanisms for achieving rationality. Second, it is more amenable to scientific development than are approaches based on human behavior or human thought. The standard of rationality is mathematically well defined and completely general, and can be "unpacked" to generate agent designs that provably achieve it. Human behavior, on the other hand, is well adapted for one specific environment and is defined by, well, the sum total of all the things that humans do. *This book therefore concentrates on general principles of rational agents and on components for constructing them.* We will see that despite the apparent simplicity with which the problem can be stated, an enormous variety of issues come up when we try to solve it. Chapter 2 outlines some of these issues in more detail.

One important point to keep in mind: We will see before too long that achieving perfect rationality—always doing the right thing—is not feasible in complicated environments. The computational demands are just too high. For most of the book, however, we will adopt the working hypothesis that perfect rationality is a good starting point for analysis. It simplifies the problem and provides the appropriate setting for most of the foundational material in the field. Chapters 5 and 17 deal explicitly with the issue of **limited rationality**—acting appropriately when there is not enough time to do all the computations one might like.

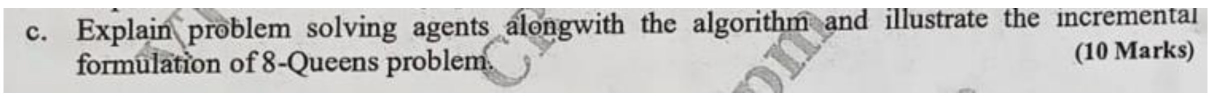
2b)


b. Explain the tree search and graph search algorithms. **(06 Marks)**

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

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

2c)


c. Explain problem solving agents alongwith the algorithm and illustrate the incremental formulation of 8-Queens problem. **(10 Marks)**

The problem-solving approach has been applied to a vast array of task environments. We list some of the best known here, distinguishing between *toy* and *real-world* problems. A **toy problem** is intended to illustrate or exercise various problem-solving methods. It can be given a concise, exact description and hence is usable by different researchers to compare the performance of algorithms. A **real-world problem** is one whose solutions people actually care about. Such problems tend not to have a single agreed-upon description, but we can give the general flavor of their formulations.

The goal of the **8-queens problem** is to place eight queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.) Figure 3.5 shows an attempted solution that fails: the queen in the rightmost column is attacked by the queen at the top left.

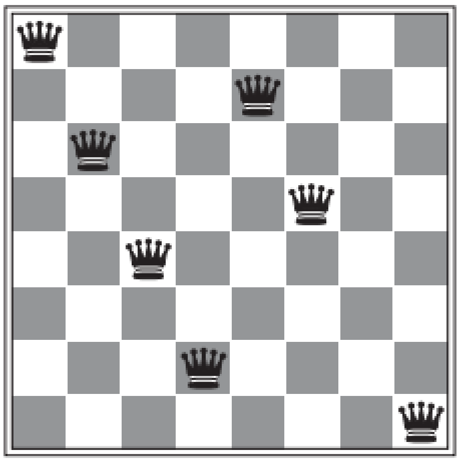


Figure 3.5 Almost a solution to the 8-queens problem. (Solution is left as an exercise.)

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

- **States**: Any arrangement of 0 to 8 queens on the board is a state.
- **Initial state**: No queens on the board.
- **Actions**: Add a queen to any empty square.
- **Transition model**: Returns the board with a queen added to the specified square.
- **Goal test**: 8 queens are on the board, none attacked.

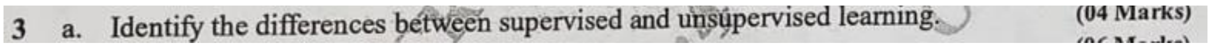
In this formulation, we have $64 \cdot 63 \cdots 57 \approx 1.8 \times 10^{14}$ possible sequences to investigate. A better formulation would prohibit placing a queen in any square that is already attacked:

- **States**: All possible arrangements of n queens $(0 \leq n \leq 8)$, one per column in the leftmost n columns, with no queen attacking another.
- **Actions**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.

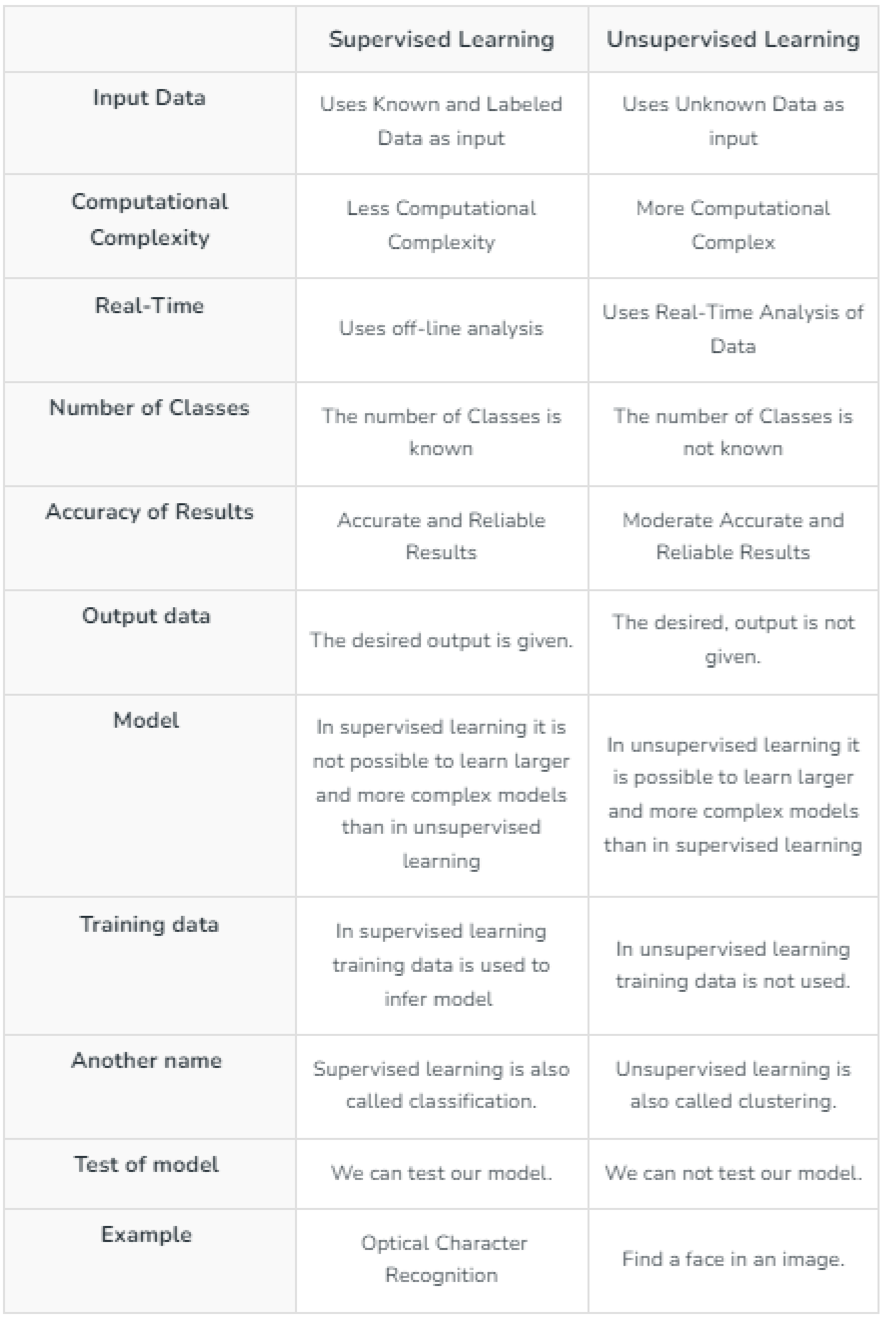
This formulation reduces the 8-queens state space from 1.8×10^{14} to just 2,057, and solutions are easy to find. On the other hand, for 100 queens the reduction is from roughly 10^{400} states to about 10^{52} states (Exercise 3.5)—a big improvement, but not enough to make the problem tractable. Section 4.1 describes the complete-state formulation, and Chapter 6 gives a simple algorithm that solves even the million-queens problem with ease.

Module 2

3a)



3 a. Identify the differences between supervised and unsupervised learning. (04 Marks)



	Supervised Learning	Unsupervised Learning
Input Data	Uses Known and Labeled Data as input	Uses Unknown Data as input
Computational Complexity	Less Computational Complexity	More Computational Complex
Real-Time	Uses off-line analysis	Uses Real-Time Analysis of Data
Number of Classes	The number of Classes is known	The number of Classes is not known
Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results
Output data	The desired output is given.	The desired, output is not given.
Model	In supervised learning it is not possible to learn larger and more complex models than in unsupervised learning	In unsupervised learning it is possible to learn larger and more complex models than in supervised learning
Training data	In supervised learning training data is used to infer model	In unsupervised learning training data is not used.
Another name	Supervised learning is also called classification.	Unsupervised learning is also called clustering.
Test of model	We can test our model.	We can not test our model.
Example	Optical Character Recognition	Find a face in an image.

3b)

b. Explain the types of Big data. (06 Marks)

Types of data:

1) Structured data: tabular form

- Data can be retrieved using tools like SQL.

a) Record data

- collection of measurements taken from a process.

- rows: entities,cases or records.

- columns: attributes/features/fields.

- table is filled with observed data.

- ‘label’ – is used to describe the individual observations.

b)Data matrix

- variant of record type. It consists of numeric attributes.

c)Graph data

- it involves the relationship among objects.

E.g. A web page can refer to another web page. This can be modeled as a graph.

Nodes are web pages and the hyperlink is an edge that connects the nodes.

d)Ordered data

- Ordered data objects involve attributes that have an implicit order among them.

E.g. Temporal data: attributes are associated with time (series of measurements over time)

Sequence data: doesn’t have time stamps (it involves the sequence of words or letters. E.g. DNA data is a sequence of 4 characters – A T G C

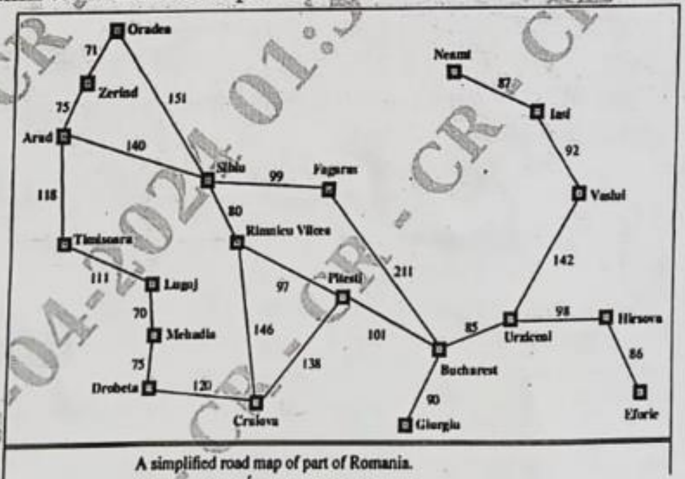
Spatial Data: attributes – positions or areas. E.g. Maps are spatial data where points are related by location.

2) Unstructured data: Video, image, audio, textual documents, programs and blog data. 80% of the data are unstructured data.

3) Semi-structured data: Partially structured and unstructured data. XML/JSON data, RSS feeds and hierarchical data.

3c)

c. Apply A* algorithm to find the best path from Arad to Bucharest. [Refer Fig.Q3(c)].



A simplified road map of part of Romania.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Values of h_{SLD}—straight-line distances to Bucharest.

Fig.Q3(c) (10 Marks)

A* Search

- a form of best-first search.

- Minimizing the total estimated solution cost.

- It evaluates nodes by combining g(n) and f(n).

g(n) ->the path cost from the start node to the node **n.**

h(n) -> the estimated cost of the cheapest path from the node **n** to the goal.

f(n) = g(n) + h(n) [f(n) : estimated cost of the cheapest solution through **n.**]

- finding the node with the lowest value of g(n)+h(n) to find the solution.

- A* search algorithm is both complete and optimal.

Conditions for optimality: (Admissibility and consistency)

- (1) The first condition for optimality is that **h(n)** be an admissible heuristic (it never overestimates the cost to reach the goal).

- **g(n)** is the actual cost to reach **n** along the current path

- f(n) never overestimates the true cost of a solution along the current path through **n.**

- h_{SLD} is admissible because the shortest path between any two points is a straight line so the straight line cannot be an overestimate.

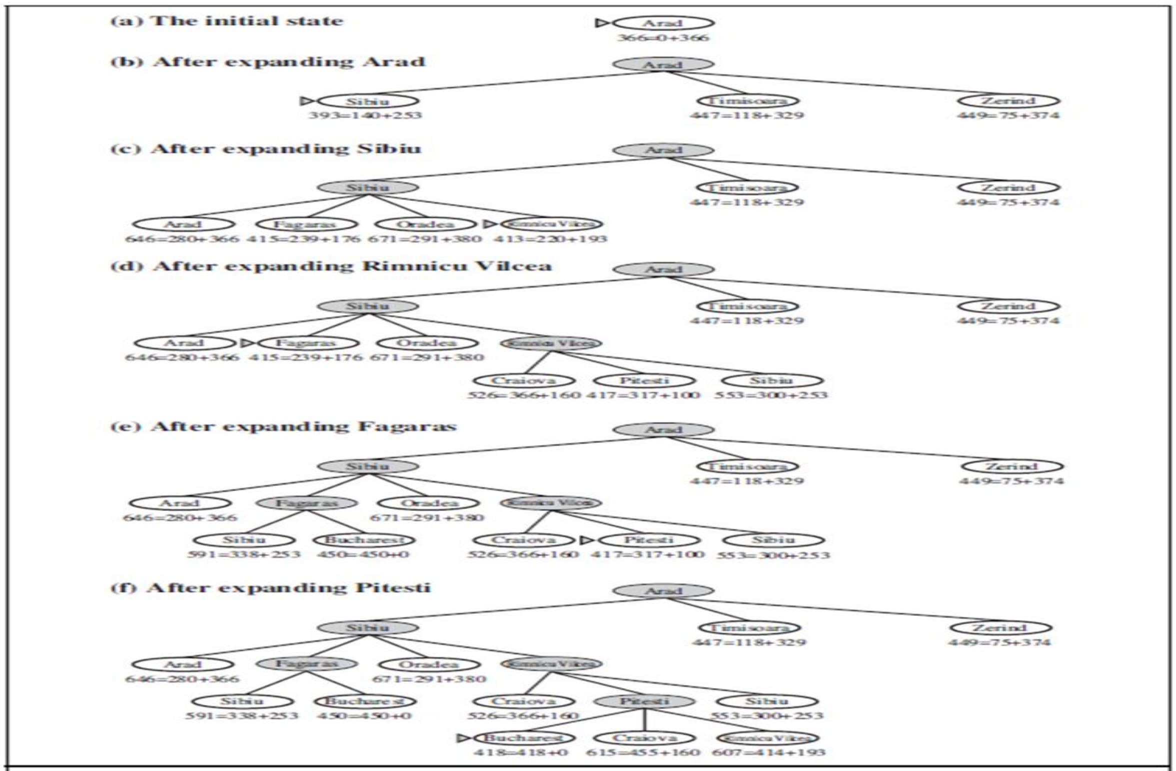


Figure 3.24 Stages in an A* search for Bucharest. Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 3.22.

- The values of **g** are computed from the step costs (in Kms Figure 3.2- Romania map) and h_{SLD} is given in Figure 3.22.

- In A* search tree(Figure 3.24), nodes are labelled with f=g+h.

- **h** values are the straight line distances to **Bucharest.**

- Bucharest first appears on the frontier at **step e** but it is not selected for expansion because its **f** cost is 450 is higher than of Piteshi(417).

- There might be a solution through the node **Piteshi.**

The second condition for optimality (consistency or monotonicity):

- It is required only for **applications** of A* to graph search.
- h(n) is consistent if 'for every node **n** and every successor **n'** of **n** generated by any action **a**, the estimated cost of reaching the goal from **n** is no greater than the step cost of getting to **n** plus the estimated cost of reaching the goal from n' :

$$h(n) <= c(n,a,n') + h(n') \quad \text{[form of triangle inequality: } a< b+c]$$

- Here, the triangle is formed by **n, n'** and goal node **G_n** closest to **n.**
- Every consistent heuristic is also admissible.
- Consistency is a stricter requirement than admissibility.
- All admissible heuristics are also consistent.

- The straight line distance between n and n’ is no greater than c(n,a,n’). h_{SLD} is a consistent heuristic.

A* has properties:

- tree search version of A* is optimal if h(n) is admissible while the graph search version is optimal if h(n) is consistent.

(1) if h(n) is consistent, then the values of f(n) along any path are nondecreasing.

Proof:

suppose n’ is a successor of n, then

$g(n')=g(n)+c(n,a,n')$

and

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

$=g(n)+c(n,a,n')+h(n')$

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

$= f(n)$

(2) Whenever A* selects a node n for expansion, the optimal path to that node has been found.

- If it is happened, there would be another frontier node n on the optimal path from start node to n.

4 a. Explain the machine learning process model along with diagram. (06 Marks)

Machine Learning Process(1)

- emerging process model for data mining for business

Organization is: Cross Industry Standard Process – Data

Mining (CRISP-DM).

- 6 steps:

1) Understanding the business: objectives and requirements

- problem formulation

2) Understanding the data: data collection, study

characteristics of the data, hypothesis formulation,

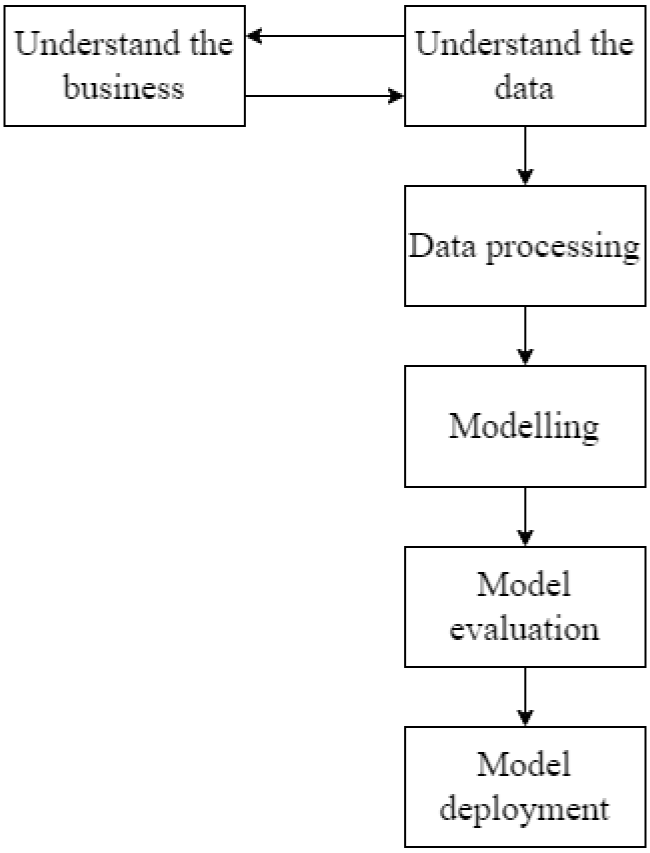
matching patterns to the selected hypothesis.

3) Preparation of data/ data preprocessing: cleaning raw data and handling missing values.

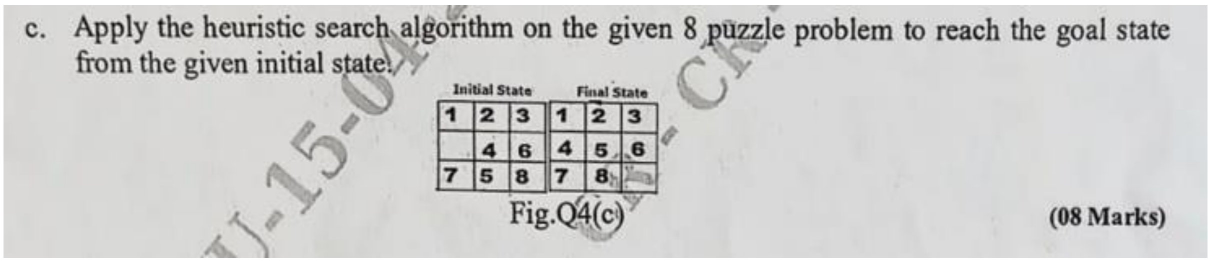
4) Modelling: Obtain a model or pattern.

5) Evaluate: Evaluate the results using statistical analysis and visualization methods.

6) Deployment: Improve the existing process.



4c)



c. Apply the heuristic search algorithm on the given 8 puzzle problem to reach the goal state from the given initial state.

Initial State

1	2	3
	4	6
7	5	8

Final State

1	2	3
4	5	6
7	8	

Fig.Q4(c)

(08 Marks)

- Slide the tiles horizontally or vertically into the empty space until the configuration matches the goal configuration.

- Average solution cost: 22 steps (on average, it takes 22 moves to transition from a random initial state to a goal state).

- branching factor: 3(on average, each state has 3 possible successor states).

- When the empty tile is in the middle, 4 moves are possible.

- When the empty tile is in a corner, 2 moves are possible.

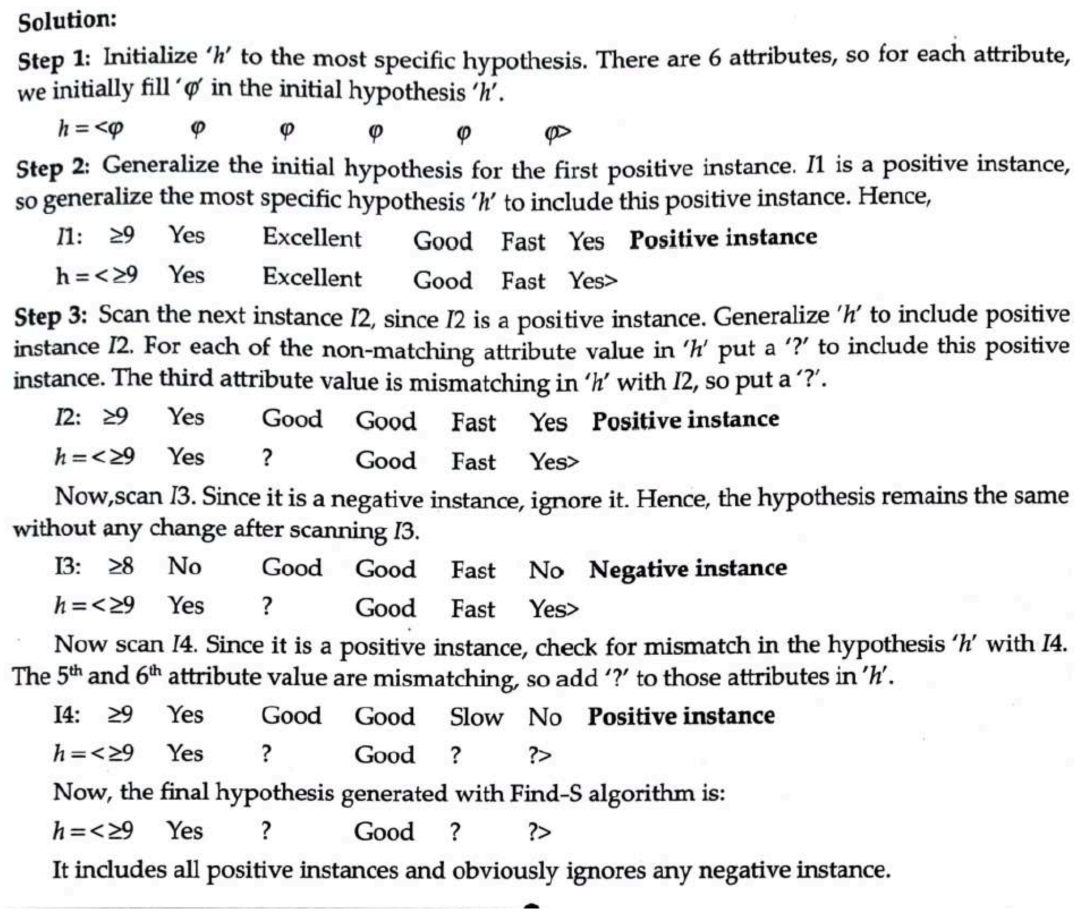
- When the empty tile is along an edge, 3 moves are possible.

- Tree search to depth: $3^{22} \sim 3.1\text{x}10^{10}$ states.

Answer Key for Module 3 and 4

Module 3

Answer 5-a



Solution:

Step 1: Initialize 'h' to the most specific hypothesis. There are 6 attributes, so for each attribute, we initially fill 'φ' in the initial hypothesis 'h'.

h = <φ φ φ φ φ φ>

Step 2: Generalize the initial hypothesis for the first positive instance. $I1$ is a positive instance, so generalize the most specific hypothesis 'h' to include this positive instance. Hence,

$I1$: ≥9 Yes Excellent Good Fast Yes **Positive instance**

h = < ≥9 Yes Excellent Good Fast Yes>

Step 3: Scan the next instance $I2$, since $I2$ is a positive instance. Generalize 'h' to include positive instance $I2$. For each of the non-matching attribute value in 'h' put a '?' to include this positive instance. The third attribute value is mismatching in 'h' with $I2$, so put a '?'.

$I2$: ≥9 Yes Good Good Fast Yes **Positive instance**

h = < ≥9 Yes ? Good Fast Yes>

Now,scan $I3$. Since it is a negative instance, ignore it. Hence, the hypothesis remains the same without any change after scanning $I3$.

I3: ≥8 No Good Good Fast No **Negative instance**

h = < ≥9 Yes ? Good Fast Yes>

Now scan $I4$. Since it is a positive instance, check for mismatch in the hypothesis 'h' with $I4$. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in 'h'.

I4: ≥9 Yes Good Good Slow No **Positive instance**

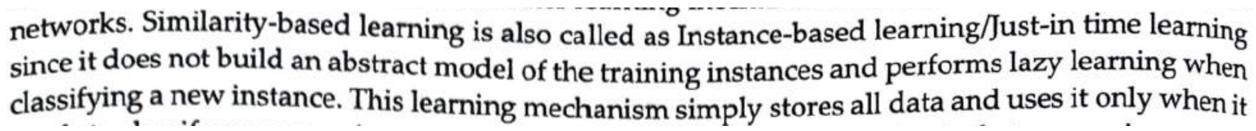
h = < ≥9 Yes ? Good ? ?>

Now, the final hypothesis generated with Find-S algorithm is:

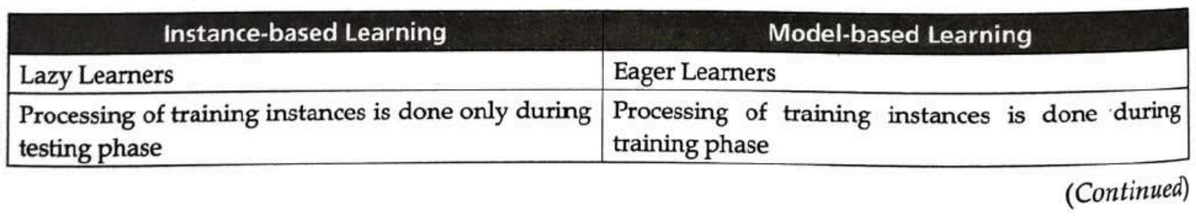
h = < ≥9 Yes ? Good ? ?>

It includes all positive instances and obviously ignores any negative instance.

Answer 5-b

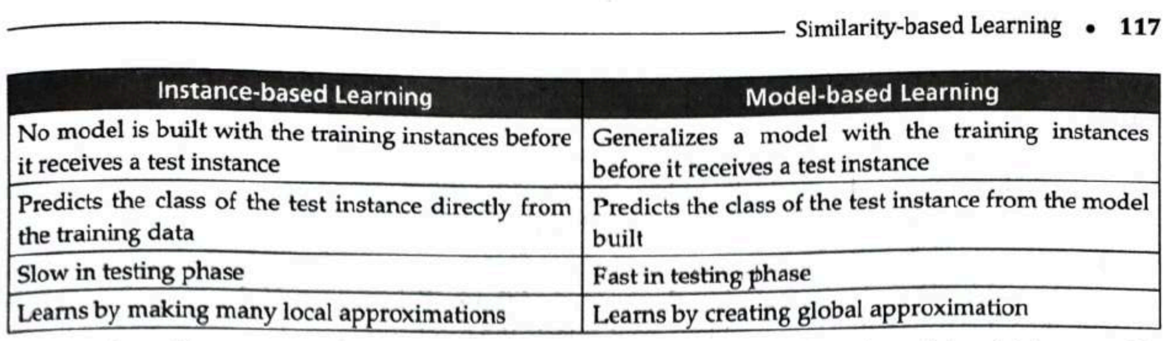


networks. Similarity-based learning is also called as Instance-based learning/Just-in time learning since it does not build an abstract model of the training instances and performs lazy learning when classifying a new instance. This learning mechanism simply stores all data and uses it only when it



Instance-based Learning	Model-based Learning
Lazy Learners	Eager Learners
Processing of training instances is done only during testing phase	Processing of training instances is done during training phase

(*Continued*)



Instance-based Learning	Model-based Learning
No model is built with the training instances before it receives a test instance	Generalizes a model with the training instances before it receives a test instance
Predicts the class of the test instance directly from the training data	Predicts the class of the test instance from the model built
Slow in testing phase	Fast in testing phase
Learns by making many local approximations	Learns by creating global approximation

Answer 5-c

Types of Regression Methods

The classification of regression methods is shown in Figure 5.3.

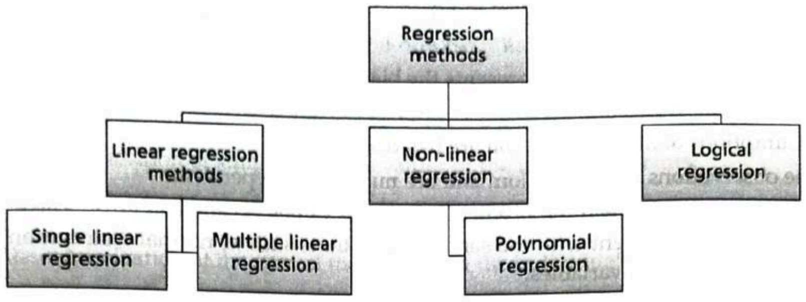


Figure 5.3: Types of Regression Methods

Linear Regression It is a type of regression where a line is fitted upon given data for finding the linear relationship between one independent variable and one dependent variable to describe relationships.

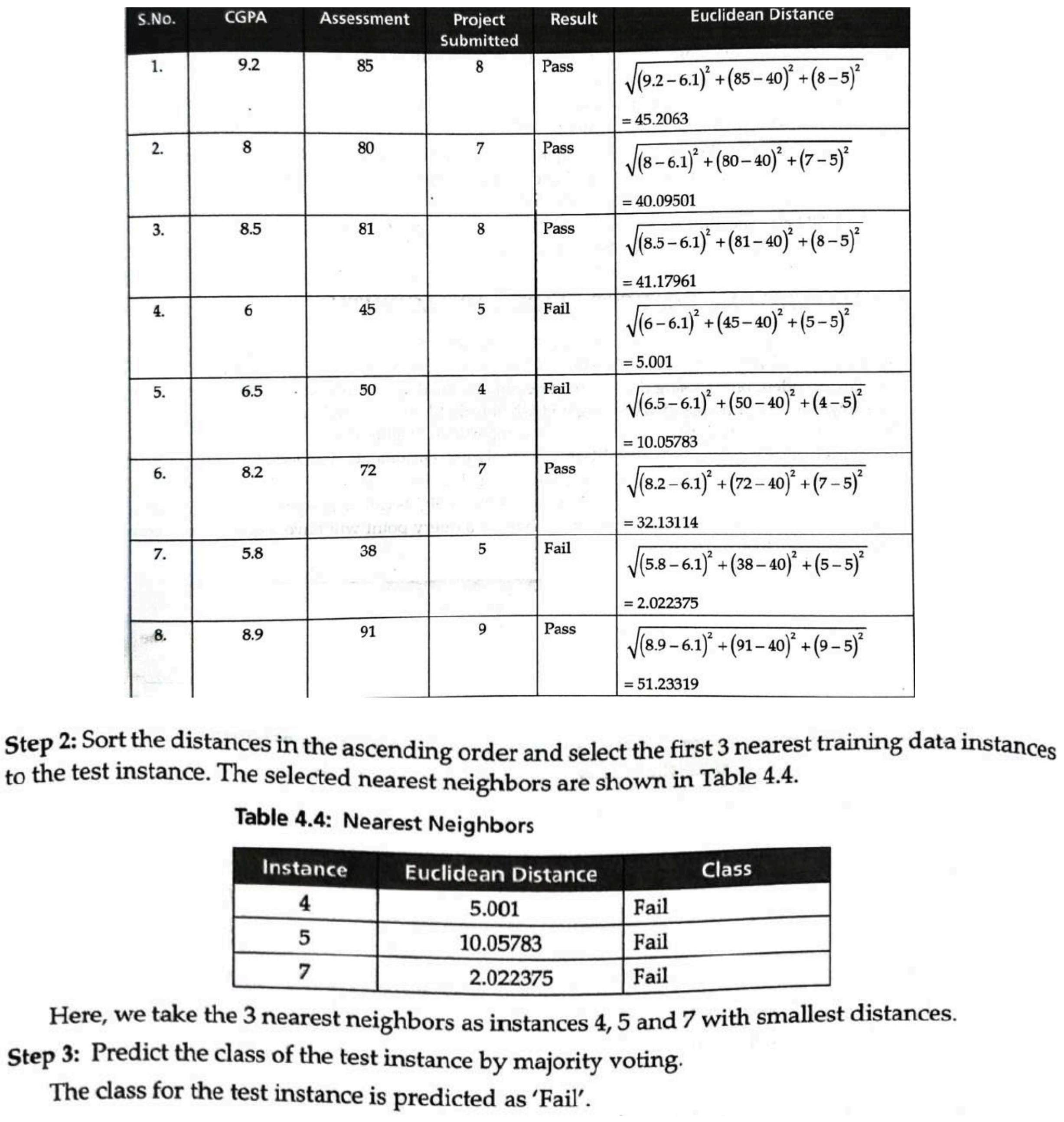
Multiple Regression It is a type of regression where a line is fitted for finding the linear relationship between two or more independent variables and one dependent variable to describe relationships among variables.

Polynomial Regression It is a type of non-linear regression method of describing relationships among variables where N^{th} degree polynomial is used to model the relationship between one independent variable and one dependent variable. Polynomial multiple regression is used to model two or more independent variables and one dependant variable.

Logistic Regression It is used for predicting categorical variables that involve one or more independent variables and one dependent variable. This is also known as a binary classifier.

Lasso and Ridge Regression Methods These are special variants of regression method where regularization methods are used to limit the number and size of coefficients of the independent variables.

Answer 6-a



S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-6.1)^2+(85-40)^2+(8-5)^2}$ = 45.2063
2.	8	80	7	Pass	$\sqrt{(8-6.1)^2+(80-40)^2+(7-5)^2}$ = 40.09501
3.	8.5	81	8	Pass	$\sqrt{(8.5-6.1)^2+(81-40)^2+(8-5)^2}$ = 41.17961
4.	6	45	5	Fail	$\sqrt{(6-6.1)^2+(45-40)^2+(5-5)^2}$ = 5.001
5.	6.5	50	4	Fail	$\sqrt{(6.5-6.1)^2+(50-40)^2+(4-5)^2}$ = 10.05783
6.	8.2	72	7	Pass	$\sqrt{(8.2-6.1)^2+(72-40)^2+(7-5)^2}$ = 32.13114
7.	5.8	38	5	Fail	$\sqrt{(5.8-6.1)^2+(38-40)^2+(5-5)^2}$ = 2.022375
8.	8.9	91	9	Pass	$\sqrt{(8.9-6.1)^2+(91-40)^2+(9-5)^2}$ = 51.23319

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.4.

Table 4.4: Nearest Neighbors

Instance	Euclidean Distance	Class
4	5.001	Fail
5	10.05783	Fail
7	2.022375	Fail

Here, we take the 3 nearest neighbors as instances 4, 5 and 7 with smallest distances.

Step 3: Predict the class of the test instance by majority voting.

The class for the test instance is predicted as 'Fail'.

Answer 6-b

The set of hypotheses that can be generated by a learning algorithm can be further reduced by specifying a language bias.

The subset of hypothesis space that is consistent with all-observed training instances is called as **Version Space.** Version space represents the only hypotheses that are used for the classification.

For example, each of the attribute given in the Table 3.1 has the following possible set of values.

Horns - Yes, No
Tail - Long, Short
Tusks - Yes, No
Paws - Yes, No
Fur - Yes, No
Color - Brown, Black, White
Hooves - Yes, No
Size - Medium, Big

Algorithm 3.3: Candidate Elimination

```
Input: Set of instances in the Training dataset
Output: Hypothesis G and S
  1. Initialize G, to the maximally general hypotheses.
  2. Initialize S, to the maximally specific hypotheses.
     • Generalize the initial hypothesis for the first positive instance.
  3. For each subsequent new training instance,
     • If the instance is positive,
        o Generalize S to include the positive instance,
           ➢ Check the attribute value of the positive instance and S,
              ▪ If the attribute value of positive instance and S are different, fill that
                field value with '?'.
              ▪ If the attribute value of positive instance and S are same, then do no
                change.
        o Prune G to exclude all inconsistent hypotheses in G with the positive instance.
     • If the instance is negative,
        o Specialize G to exclude the negative instance,
           ➢ Add to G all minimal specializations to exclude the negative example and
             be consistent with S.
              ▪ If the attribute value of S and the negative instance are different, then
                fill that attribute value with S value.
              ▪ If the attribute value of S and negative instance are same, no need to
                update 'G' and fill that attribute value with '?'.
        o Remove from S all inconsistent hypotheses with the negative instance.
```

Module 4

Answer 7-a

Advantages of Decision Trees

1. Easy to model and interpret
2. Simple to understand
3. The input and output attributes can be discrete or continuous predictor variables.
4. Can model a high degree of nonlinearity in the relationship between the target variables and the predictor variables
5. Quick to train

Disadvantages of Decision Trees

Some of the issues that generally arise with a decision tree learning are that:

1. It is difficult to determine how deeply a decision tree can be grown or when to stop growing it.
2. If training data has errors or missing attribute values, then the decision tree constructed may become unstable or biased.
3. If the training data has continuous valued attributes, handling it is computationally complex and has to be discretized.
4. A complex decision tree may also be over-fitting with the training data.
5. Decision tree learning is not well suited for classifying multiple output classes.
6. Learning an optimal decision tree is also known to be NP-complete.

Answer 7-b

To avoid overfitting of the tree, we need to prune the trees and construct an optimal decision tree. Trees can be pre-pruned or post-pruned. If tree nodes are pruned during construction or the construction is stopped earlier without exploring the nodes' branches, then it is called as pre-pruning whereas if tree nodes are pruned after the construction is over then it is called as post-pruning. Basically, the dataset is split into three sets called training dataset, validation dataset and test dataset. Generally, 40% of the dataset is used for training the decision tree and the remaining 60% is used for validation and testing. Once the decision tree is constructed, it is validated with the validation dataset and the misclassifications are identified. Using the number of instances correctly classified and number of instances wrongly classified, Average Squared Error (ASE) is computed. The tree nodes are pruned based on these computations and the resulting tree is validated until we get a tree that performs better. Cross validation is another way to construct an optimal decision tree. Here, the dataset is split into k-folds, among which k–1 folds are used for training the decision tree and the k^{th} fold is used for validation and errors are computed. The process is repeated for randomly k–1 folds and the mean of the errors is computed for different trees. The tree with the lowest error is chosen with which the performance of the tree is improved. This tree can now be tested with the test dataset and predictions are made.

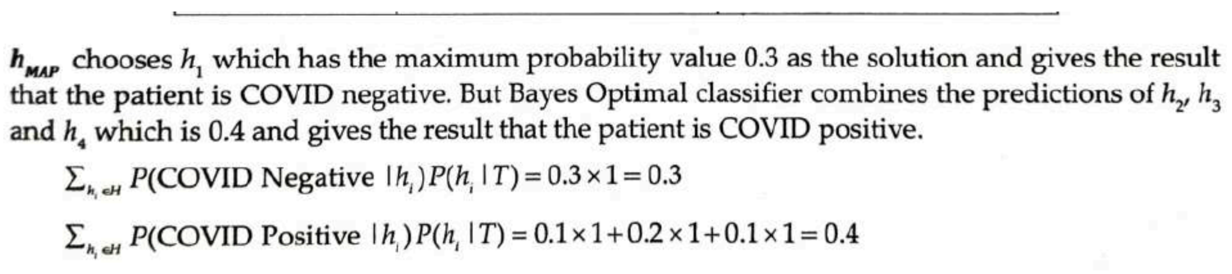
Another approach is that after the tree is constructed using the training set, statistical tests like error estimation and Chi-square test are used to estimate whether pruning or splitting is required for a particular node to find a better accurate tree.

The third approach is using a principle called Minimum Description Length which uses a complexity measure for encoding the training set and the growth of the decision tree is stopped when the encoding size (i.e., (size(tree)) + size(misclassifications(tree)) is minimized. CART and C4.5 perform post-pruning, that is, pruning the tree to a smaller size after construction in order to minimize the misclassification error. CART makes use of 10-fold cross validation method to validate and prune the trees, whereas C4.5 uses heuristic formula to estimate misclassification error rates.

Some of the tree pruning methods are listed below:

1. Reduced Error Pruning
2. Minimum Error Pruning (MEP)
3. Pessimistic Pruning
4. Error–based Pruning (EBP)
5. Optimal Pruning
6. Minimum Description Length (MDL) Pruning
7. Minimum Message Length Pruning
8. Critical Value Pruning

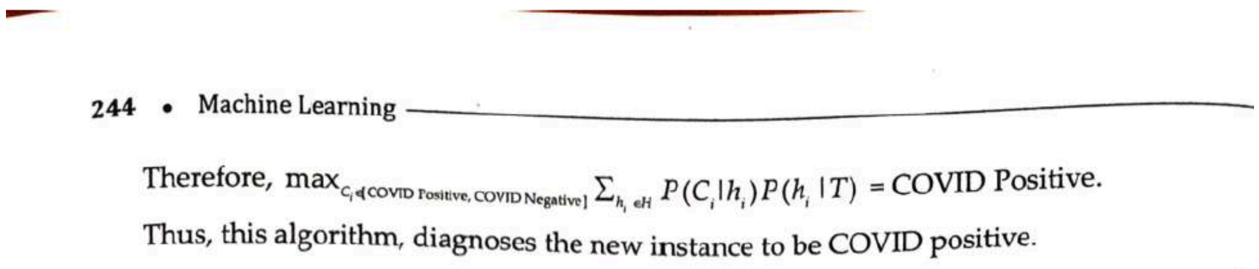
Answer 7-c



h_{MAP} chooses h_1 which has the maximum probability value 0.3 as the solution and gives the result that the patient is COVID negative. But Bayes Optimal classifier combines the predictions of h_2, h_3 and h_4 which is 0.4 and gives the result that the patient is COVID positive.

$$\Sigma_{h_i \in H} P(\text{COVID Negative} \mid h_i)P(h_i \mid T) = 0.3 \times 1 = 0.3$$

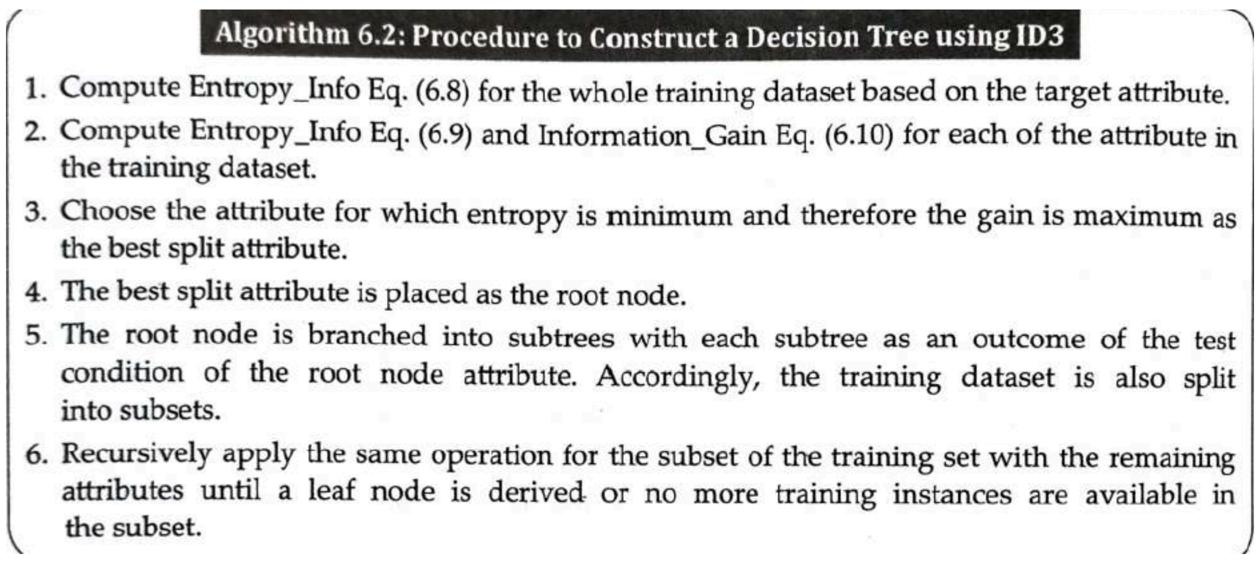
$$\Sigma_{h_i \in H} P(\text{COVID Positive} \mid h_i)P(h_i \mid T) = 0.1 \times 1 + 0.2 \times 1 + 0.1 \times 1 = 0.4$$



Therefore, $\max_{C_i \in [\text{COVID Positive, COVID Negative}]} \Sigma_{h_i \in H} P(C_i \mid h_i)P(h_i \mid T)$ = COVID Positive.

Thus, this algorithm, diagnoses the new instance to be COVID positive.

Answer 8-a



Algorithm 6.2: Procedure to Construct a Decision Tree using ID3

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9) and Information_Gain Eq. (6.10) for each of the attribute in the training dataset.
3. Choose the attribute for which entropy is minimum and therefore the gain is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Answer 8-b

Naïve Bayes Model relies on Bayes theorem that works on the principle of three kinds of probabilities called prior probability, likelihood probability, and posterior probability.

Prior Probability

It is the general probability of an uncertain event before an observation is seen or some evidence is collected. It is the initial probability that is believed before any new information is collected.

Likelihood Probability

Likelihood probability is the relative probability of the observation occurring for each class or the sampling density for the evidence given the hypothesis. It is stated as *P* (Evidence | Hypothesis), which denotes the likeliness of the occurrence of the evidence given the parameters.

Posterior Probability

It is the updated or revised probability of an event taking into account the observations from the training data. *P* (Hypothesis | Evidence) is the posterior distribution representing the belief about the hypothesis, given the evidence from the training data. Therefore,

Posterior probability = prior probability + new evidence

[illegible]

It can be written as:

$$P\,(\text{Hypothesis } h \mid \text{Evidence } E) = \frac{P(\text{Evidence } E \mid \text{Hypothesis } h)\ P(\text{Hypothesis } h)}{P(\text{Evidence } E)}$$

[illegible]

Maximum A Posteriori (MAP) Hypothesis, h_{MAP}

Given a set of candidate hypotheses, the hypothesis which has the maximum value is considered as the *maximum probable hypothesis* or *most probable hypothesis*. This most probable hypothesis is called the Maximum A Posteriori Hypothesis h_{MAP}. Bayes theorem Eq. (8.1) can be used to find the h_{MAP}.

$$\begin{aligned} h_{MAP} &= \max_{h\epsilon H} P(Hypothesis h \mid Evidence\, E) \\ &= \max_{h\epsilon H} \frac{P(Evidence\, E \mid Hypothesis\ h)P(Hypothesis\ h)}{P(Evidence\, E)} \\ &= \max_{h\epsilon H} P(Evidence\, E \mid Hypothesis\ h)P(Hypothesis\ h) \end{aligned} \quad (8.2)$$

Maximum Likelihood (ML) Hypothesis, h_{ML}

Given a set of candidate hypotheses, if every hypothesis is equally probable, only *P* (*E* | *h*) is used to find the *most probable hypothesis*. The hypothesis that gives the maximum likelihood for *P* (*E* | *h*) is called the Maximum Likelihood (ML) Hypothesis, h_{ML}.

$$h_{ML} = \max_{h\epsilon H} P(Evidence\, E \mid Hypothesis\ h) \quad (8.3)$$

Answer 8-c

Algorithm 8.1: Naïve Bayes

1. Compute the prior probability for the target class.
2. Compute Frequency matrix and likelihood Probability for each of the feature.
3. Use Bayes theorem Eq. (8.1) to calculate the probability of all hypotheses.
4. Use Maximum A Posteriori (MAP) Hypothesis, h_{MAP} Eq. (8.2) to classify the test object to the hypothesis with the highest probability.

8.5 OTHER POPULAR TYPES OF NAIVE BAYES CLASSIFIERS

Some of the popular variants of Bayesian classifier are listed below:

Bernoulli Naive Bayes Classifier

Bernoulli Naive Bayes works with discrete features. In this algorithm, the features used for making predictions are Boolean variables that take only two values either 'yes' or 'no'. This is particularly useful for text classification where all features are binary with each feature containing two values whether the word occurs or not.

Multinomial Naive Bayes Classifier

This algorithm is a generalization of the Bernoulli Naive Bayes model that works for categorical data or particularly integer features. This classifier is useful for text classification where each feature will have an integer value that represents the frequency of occurrence of words.

Multi-class Naïve Bayes Classifier

This algorithm is useful for classification problems with more than two classes where the target feature contains multiple classes and test instance has to be predicted with the class it belongs to.

9	a.	Explain the different activation functions used in ANN.	(06 Marks)
	b.	Illustrate the various types of Artificial Neural Networks.	(08 Marks)
	c.	Illustrate the applications and challenges of Clustering algorithms.	(06 Marks)

Activation functions are mathematical functions associated with each neuron in the neural network that map input signals to output signals. It decides whether to fire a neuron or not based on the input signals the neuron receives. These functions normalize the output value of each neuron either between 0 and 1 or between –1 and +1. Typical activation functions can be linear or non-linear.

Linear functions are useful when the input values can be classified into any one of the two groups and are generally used in binary perceptrons. Non-linear functions, on the other hand, are continuous functions that map the input in the range of (0, 1) or (–1, 1), etc. These functions are useful in learning high-dimensional data or complex data such as audio, video and images.

Below are some of the activation functions used in ANNs:

1. Identity Function or Linear Function

$$f(x) = x \ \forall x \tag{10.4}$$

The value of $f(x)$ increases linearly or proportionally with the value of x. This function is useful when we do not want to apply any threshold. The output would be just the weighted sum of input values. The output value ranges between $-\infty$ and $+\infty$.

2. Binary Step Function

$$f(x) = \begin{cases} 1 \ \text{if } f(x) \geq \theta \\ 0 \ \text{if } f(x) < \theta \end{cases} \tag{10.5}$$

The output value is binary, i.e., 0 or 1 based on the threshold value θ. If value of $f(x)$ is greater than or equal to θ, it outputs 1 or else it outputs 0.

3. Bipolar Step Function

$$f(x) = \begin{cases} 1 \ \text{if } f(x) \geq \theta \\ -1 \ \text{if } f(x) < \theta \end{cases} \tag{10.6}$$

The output value is bipolar, i.e., +1 or –1 based on the threshold value θ. If value of $f(x)$ is greater than or equal to θ, it outputs +1 or else it outputs –1.

4. Sigmoidal Function or Logistic Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{10.7}$$

It is a widely used non-linear activation function which produces an S-shaped curve and the output values are in the range of 0 and 1. It has a vanishing gradient problem, i.e., no change in the prediction for very low input values and very high input values.

5. Bipolar Sigmoid Function

$$\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{10.8}$$

It outputs values between –1 and +1.

6. Ramp Functions

$$f(x) = \begin{cases} 1 \ \text{if } x > 1 \\ x \ \text{if } 0 \leq x \leq 1 \\ 0 \ \text{if } x < 0 \end{cases} \tag{10.9}$$

It is a linear function whose upper and lower limits are fixed.

7. Tanh – Hyperbolic Tangent Function

The Tanh function is a scaled version of the sigmoid function which is also non-linear. It also suffers from the vanishing gradient problem. The output values range between –1 and 1.

$$\tan h(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{10.10}$$

8. ReLu – Rectified Linear Unit Function

This activation function is a typical function generally used in deep learning neural network models in the hidden layers. It avoids or reduces the vanishing gradient problem. This function outputs a value of 0 for negative input values and works like a linear function if the input values are positive.

$$r(x) = \max(0, x) = \begin{cases} x \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases} \tag{10.11}$$

9. Softmax Function

This is a non-linear function used in the output layer that can handle multiple classes. It calculates the probability of each target class which ranges between 0 and 1. The probability of the input belonging to a particular class is computed by dividing the exponential of the given input value by the sum of the exponential values of all the inputs.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \text{ where } i = 0 \ldots k \tag{10.12}$$

ANNs consist of multiple neurons arranged in layers. There are different types of ANNs that differ by the network structure, activation function involved and the learning rules used. In an ANN, there are three layers called input layer, hidden layer and output layer. Any general ANN would consist of one input layer, one output layer and zero or more hidden layers.

10.5.1 Feed Forward Neural Network

This is the simplest neural network that consists of neurons which are arranged in layers and the information is propagated only in the forward direction. This model may or may not contain a hidden layer and there is no back propagation. Based on the number of hidden layers they are further classified into single-layered and multi-layered feed forward networks. These ANNs are simple to design and easy to maintain. They are fast but cannot be used for complex learning. They are used for simple classification and simple image processing, etc. The model of a Feed Forward Neural Network is shown in Figure 10.7.

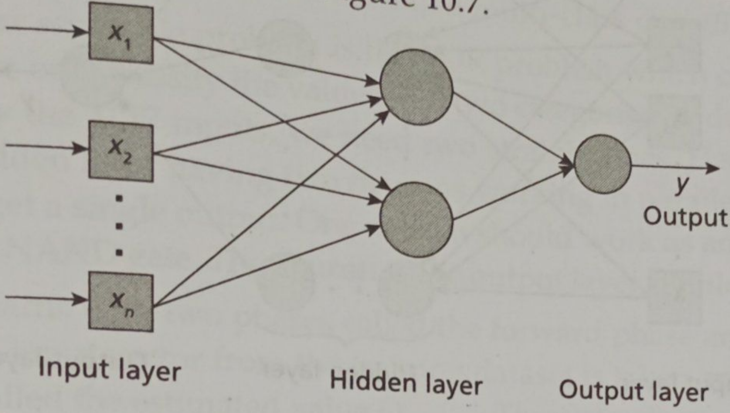


Figure 10.7: Model of a Feed Forward Neural Network

10.5.2 Fully Connected Neural Network

Fully connected neural networks are the ones in which all the neurons in a layer are connected to all other neurons in the next layer. The model of a fully connected neural network is shown in Figure 10.8.

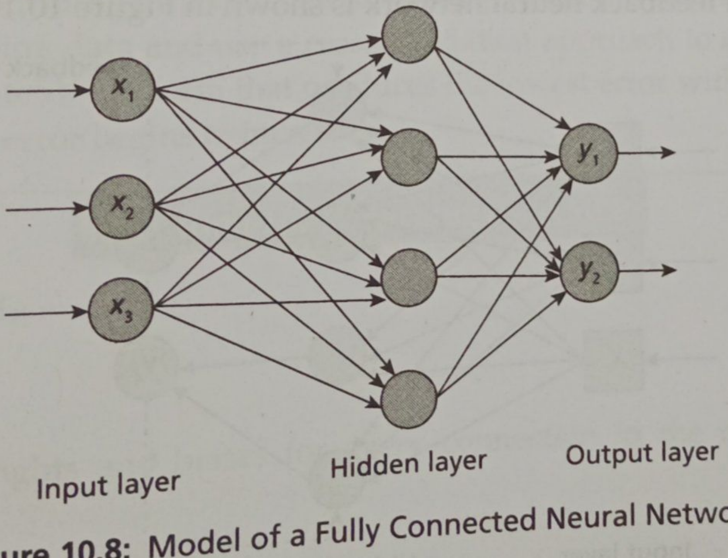


Figure 10.8: Model of a Fully Connected Neural Network

10.5.3 Multi-Layer Perceptron (MLP)

This ANN consists of multiple layers with one input layer, one output layer and one or more hidden layers. Every neuron in a layer is connected to all neurons in the next layer and thus they are fully connected. The information flows in both the directions. In the forward direction, the inputs are multiplied by weights of neurons and forwarded to the activation function of the neuron and output is passed to the next layer. If the output is incorrect, then in the backward direction, error is back propagated to adjust the weights and biases to get correct output. Thus, the network learns with the training data. This type of ANN is used in deep learning for complex classification, speech recognition, medical diagnosis, forecasting, etc. They are comparatively complex and slow. The model of an MLP is shown in Figure 10.9.

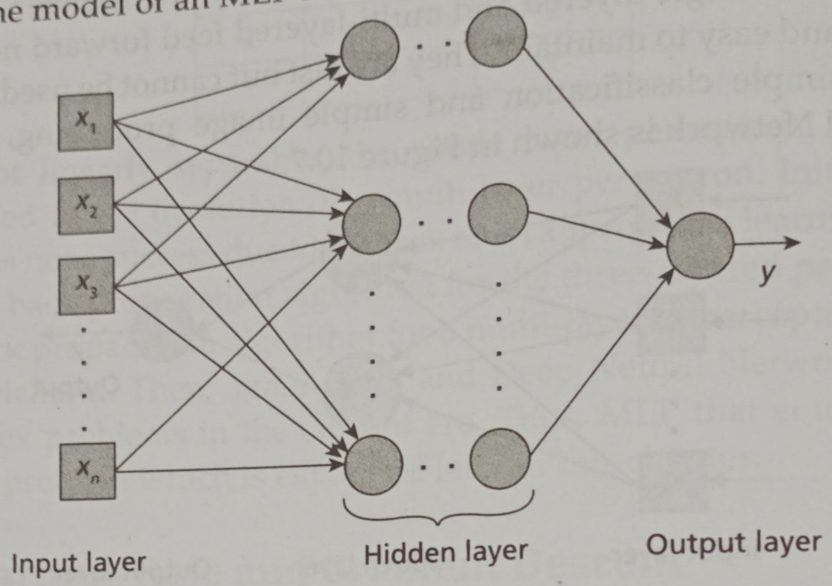


Figure 10.9: Model of a Multi-Layer Perceptron

10.5.4 Feedback Neural Network

Feedback neural networks have feedback connections between neurons that allow information flow in both directions in the network. The output signals can be sent back to the neurons in the same layer or to the neurons in the preceding layers. Hence, this network is more dynamic during training. The model of a feedback neural network is shown in Figure 10.10.

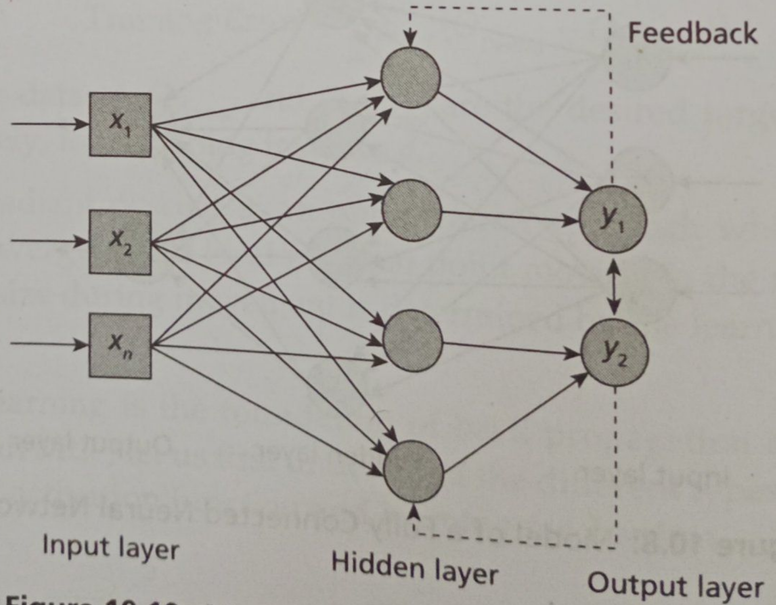


Figure 10.10: Model of a Feedback Neural Network

Once a label is assigned, it does not change

Applications of Clustering

1. Grouping based on customer buying patterns
2. Profiling of customers based on lifestyle
3. In information retrieval applications (like retrieval of a document from a collection of documents)
4. Identifying the groups of genes that influence a disease
5. Identification of organs that are similar in physiology functions
6. Taxonomy of animals, plants in Biology
7. Clustering based on purchasing behaviour and demography
8. Document indexing
9. Data compression by grouping similar objects and finding duplicate objects

Challenges of Clustering Algorithms

A huge collection of data with higher dimensions (i.e., features or attributes) can pose a problem for clustering algorithms. With the arrival of the internet, billions of data are available for clustering algorithms. This is a difficult task, as scaling is always an issue with clustering algorithms. Scaling is an issue where some algorithms work with lower dimension data but do not perform well for higher dimension data. Also, units of data can post a problem, like some weights in kg and some in pounds can pose a problem in clustering. Designing a proximity measure is also a big challenge.

The advantages and disadvantages of the cluster analysis algorithms are given in Table 13.2.

Table 13.2: Advantages and Disadvantages of Clustering Algorithms

S.No.	Advantages	Disadvantages
1.	Cluster analysis algorithms can handle missing data and outliers.	Cluster analysis algorithms are sensitive to initialization and order of the input data.
2.	Can help classifiers in labelling the unlabelled data. Semi-supervised algorithms use cluster analysis algorithms to label the unlabelled data and then use classifiers to classify them.	Often, the number of clusters present in the data have to be specified by the user.

(Continued)

S.No.	Advantages	Disadvantages
3.	It is easy to explain the cluster analysis algorithms and to implement them.	Scaling is a problem.
4.	Clustering is the oldest technique in statistics and it is easy to explain. It is also relatively easy to implement.	Designing a proximity measure for the given data is an issue.

10 a. Explain the perceptron model and the algorithm. (08 Marks)

b. Consider the following set of data given in the below table. Cluster it using K-means algorithm with the initial value of objects 2 and 5 with the coordinate values (4, 6) and (12, 4) as initial seeds.

Table 10(b)

Objects	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

(12 Marks)

The first neural network model 'Perceptron', designed by Frank Rosenblatt in 1958, is a linear binary classifier used for supervised learning. He modified the McCulloch & Pitts Neuron model by combining two concepts, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights. He introduced variable weight values and an extra input that represents bias to this model. He proposed that artificial neurons could actually learn weights and thresholds from data and came up with a supervised learning algorithm that enabled the artificial neurons to learn the correct weights from training data by itself. The perceptron model (shown in Figure 10.5) consists of 4 steps:

1. Inputs from other neurons
2. Weights and bias
3. Net sum
4. Activation function

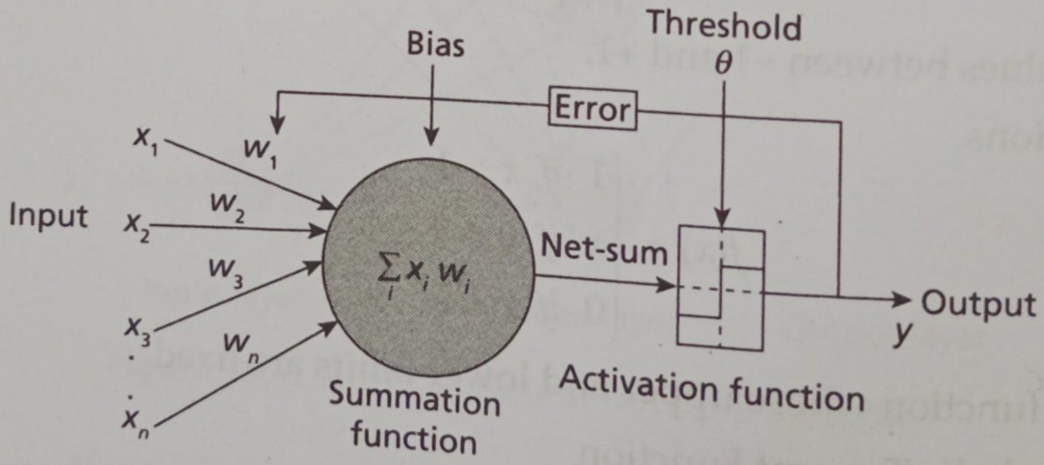


Figure 10.5: Perceptron Model

Thus, the modified neuron model receives a set of inputs $x_1, x_2, \ldots, x_n$, their associated weights $w_1, w_2, \ldots, w_n$ and a bias. The summation function 'Net-sum' Eq. (10.13) computes the weighted sum of the inputs received by the neuron.

$$\text{Net-sum} = \sum_{i=1}^{n} x_i w_i \tag{10.13}$$

After computing the 'Net-sum', bias value is added to it and inserted in the activation function as shown below:

$$f(x) = \text{Activation function } (Net\text{-}sum + bias) \tag{10.14}$$

The activation function is a binary step function which outputs a value 1 if $f(x)$ is above the threshold value θ, and a 0 if $f(x)$ is below the threshold value θ. Then, output of a neuron:

$$Y = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases} \tag{10.15}$$

Before learning how a neural network works, let us learn about how a perceptron model works.

Algorithm 10.1: Perceptron Algorithm

```
Set initial weights $w_1, w_2, \ldots, w_n$ and bias $\theta$ to a random value in the range [–0.5, 0.5].
  For each Epoch,
    1. Compute the weighted sum by multiplying the inputs with the weights and add the
       products.
    2. Apply the activation function on the weighted sum:
           $Y = \text{Step}((x_1 w_1 + x_2 w_2) - \theta)$
    3. If the sum is above the threshold value, output the value as positive else output the
       value as negative.
    4. Calculate the error by subtracting the estimated output $Y_{estimated}$ from the desired output
       $Y_{desired}$:
           error $e(t) = Y_{desired} - Y_{estimated}$
         [If error $e(t)$ is positive, increase the perceptron output $Y$ and if it is negative,
       decrease the perceptron output $Y$.]
    5. Update the weights if there is an error:
           $\Delta w_i = \propto \times e(t) \times x_i$
           $w_i = w_i + \Delta w_i$
       where, $x_i$ is the input value, $e(t)$ is the error at step $t$, $\propto$ is the learning rate and $\Delta w_i$ is the
       difference in weight that has to be added to $w_i$.
```

Table 13.9: Sample Data

Objects	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

Solution: As per the problem, choose the objects 2 and 5 with the coordinate values. Hereafter, the objects' id is not important. The samples or data points (4, 6) and (12, 4) are started as two clusters as shown in Table 13.10.

Initially, centroid and data points are same as only one sample is involved.

Table 13.10: Initial Cluster Table

Cluster 1	Cluster 2
(4, 6)	(12, 4)
Centroid 1 (4, 6)	Centroid 2 (12, 4)

Iteration 1: Compare all the data points or samples with the centroid and assign to the nearest sample. Take the sample object 1 (2, 4) from Table 13.9 and compare with the centroid of

the clusters in Table 13.10. The distance is 0. Therefore, it remains in the same cluster. Similarly, consider the remaining samples. For the object 1 (2, 4), the Euclidean distance between it and the centroid is given as:

Dist (1, centroid 1) = $\sqrt{(2-4)^2 + (4-6)^2} = \sqrt{8}$

Dist (1, centroid 2) = $\sqrt{(2-12)^2 + (4-4)^2} = \sqrt{100} = 10$

Object 1 is closer to the centroid of cluster 1 and hence assign it to cluster 1. This is shown in Table 13.11. Object 2 is taken as centroid point.

For the object 3 (6, 8), the Euclidean distance between it and the centroid points is given as:

Dist (3, centroid 1) = $\sqrt{(6-4)^2 + (8-6)^2} = \sqrt{8}$

Dist (3, centroid 2) = $\sqrt{(6-12)^2 + (8-4)^2} = \sqrt{52}$

Object 3 is closer to the centroid of cluster 1 and hence remains in the same cluster 1.

Proceed with the next point object 4(10, 4) and again compare it with the centroids in Table 13.10.

Dist (4, centroid 1) = $\sqrt{(10-4)^2 + (4-6)^2} = \sqrt{40}$

Dist (4, centroid 2) = $\sqrt{(10-12)^2 + (4-4)^2} = \sqrt{4} = 2$

Object 4 is closer to the centroid of cluster 2 and hence assign it to the cluster table. Object 4 is in the same cluster. The final cluster table is shown in Table 13.11.

Obviously, Object 5 is in Cluster 3. Recompute the new centroids of cluster 1 and cluster 2. They are (4, 6) and (11, 4), respectively.

Table 13.11: Cluster Table After Iteration 1

Cluster 1	Cluster 2
(4, 6)	(10, 4)
(2, 4)	(12, 4)
(6, 8)	
Centroid 1 (4, 6)	Centroid 2 (11, 4)

The second iteration is started again with the Table 13.11.

Obviously, the point (4, 6) remains in cluster 1, as the distance of it with itself is 0. The remaining objects can be checked. Take the sample object 1 (2, 4) and compare with the centroid of the clusters in Table 13.12.

Dist (1, centroid 1) = $\sqrt{(2-4)^2 + (4-6)^2} = \sqrt{8}$

Dist (1, centroid 2) = $\sqrt{(2-11)^2 + (4-4)^2} = \sqrt{81} = 9$

Object 1 is closer to centroid of cluster 1 and hence remains in the same cluster. Take the sample object 3 (6, 8) and compare with the centroid values of clusters 1 (4, 6) and cluster 2 (11, 4) of the Table 13.12.

Dist (3, centroid 1) = $\sqrt{(6-4)^2 + (8-6)^2} = \sqrt{8}$

Dist (3, centroid 2) = $\sqrt{(6-11)^2 + (8-4)^2} = \sqrt{41}$

Object 3 is closer to centroid of cluster 1 and hence remains in the same cluster. Take the sample object 4 (10, 4) and compare with the centroid values of clusters 1 (4, 6) and cluster 2 (11, 4) of the Table 13.12:

$$\text{Dist }(4, \text{centroid } 1) = \sqrt{(10-4)^2 + (4-6)^2} = \sqrt{40}$$

$$\text{Dist }(3, \text{centroid } 2) = \sqrt{(10-11)^2 + (4-4)^2} = \sqrt{1} = 1$$

Object 3 is closer to centroid of cluster 2 and hence remains in the same cluster. Obviously, the sample (12, 4) is closer to its centroid as shown below:

$$\text{Dist }(5, \text{centroid } 1) = \sqrt{(12-4)^2 + (4-6)^2} = \sqrt{68}$$

$\text{Dist }(5, \text{centroid } 2) = \sqrt{(12-11)^2 + (4-4)^2} = \sqrt{1} = 1$. Therefore, it remains in the same cluster. Object 5 is taken as centroid point.

The final cluster Table 13.12 is given below:

Table 13.12: Cluster Table After Iteration 2

Cluster 1	Cluster 2
(4, 6)	(10, 4)
(2, 4)	(12, 4)
(6, 8)	
Centroid (4, 6)	Centroid (11, 4)

There is no change in the cluster Table 13.12. It is exactly the same; therefore, the *k*-means algorithm terminates with two clusters with data points as shown in the Table 13.12.