Page No.

Date : / /20

Aim : To implement tic-tac-toe game.

Theory :

Tic-tac-toe is a simple game for two players that we enjoy. The game involves 2 players placing their respective symbols in a 3×3 grid. The player who manages to place three of their symbols in horizontal / vertical / diagonal row within the game. If either player fails to do so the game ends in draw.

If both players play their optimal strategies the game always ends in a draw

i) Since the grid is small and there are only two players involved the number of possible moves for every board state is limited thus allowing tree based search algorithm to provide a computationally feasible and exact solution to building a computer based tic-tac-toe players.

ii) We look at approximate approach to game game.

iii) The idea is to pose the tic-tac-toe game as a well posed learning problem. The learning system is described in brief Tic-tac-toe learning system.

The basic idea behind the learning system.

Vaishali

is that the system should be able to improve its performance (P) w.r.t. a set of tables (T) by learning from training experience (E).

The training experience (E) can be a direct (predefined set of club with individual labels) indirect feedback (No labels for each learning example)

* Take (T) playing tic-tac-toe.
* Performance (P) percentage of games won against humans.
* Experience (E): Indirect feedback is solution based (game history) generated from game played against itself (a clone)

Learning from Experience (E):

Ideally a function needs to be learned that gives the best moves possible for any given board state. In our problem, we represent our T/f to be a linear function (w) that maps a given board state to used value. Then we use appropriate algorithms (LMS) to estimate IEF from solution.

i) v (board state) → R (R- secure value for board state)

ii) v.net (board state) ↔ R (R- W.T)* x, (w=weight of target function, x= features extracted from given board state)

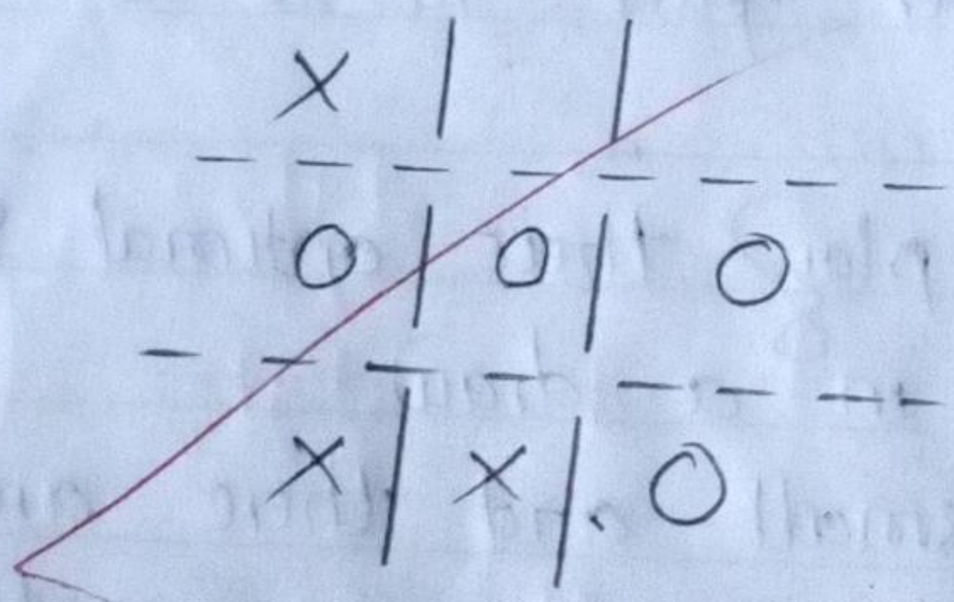
iii) LMS training rule to update given weights w.r.t.

Vaishali

Computer - O

Human - X

Computer Starts First



w_i + α (v-net (board state) - v.net (successor (board state))) * x_i (i-ith training example, learning rule)

iv) The score (R) for each non-final board state is assigned with the estimated score of successor board state. The first board state is assigned a score based on the end result at the game.

v) v (board state) ← v. net (successor board state)

vi) v (final board state) ← 100 (win) / 0 (Draw) / -100 (lose)

IMPLEMENTATION:

The final design is split into two modules

i) Experiment Generation:

It's job is to generate new problem statement at the beginning of every training epoch. In our case it just returns an empty initial board state.

ii) Performance System:

The module takes an input problem provided by the experiment generation and then uses the unproved learning algorithm to produce a solution base of the game of every pach.

iii) Critic:

The module takes the solution base and outputs a set of training example to be input to the generalizer.

Vaishali

Page No.

Date: / /20

iv) Generalizer :

The module uses the training example provided by the critic to update. Improve the get function by learning the desired weights using the sms weights update rule at each epoch.

Conclusion :

We have implemented a tic-tac-toe game.

Rish
3/10/22

Vaishali

Page No.

Date: / /20

Aim: Solve 8-puzzle problem. Assume any initials configuration & define goal configuration clearly.

Theory:

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and the empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above and below) tiles into the empty space

1. DFS (Brute-Force)

We can perform a depth-first search on state space (set all configurations of a given problem i.e. all states that can be reached from the initial state) tree.

2. BFS (Brute-Force)

We can perform BFS on the state space tree. This always finds a goal state nearest to the root. But no matter what the initial state is, the algorithm attempts the same sequence of moves like DFS.

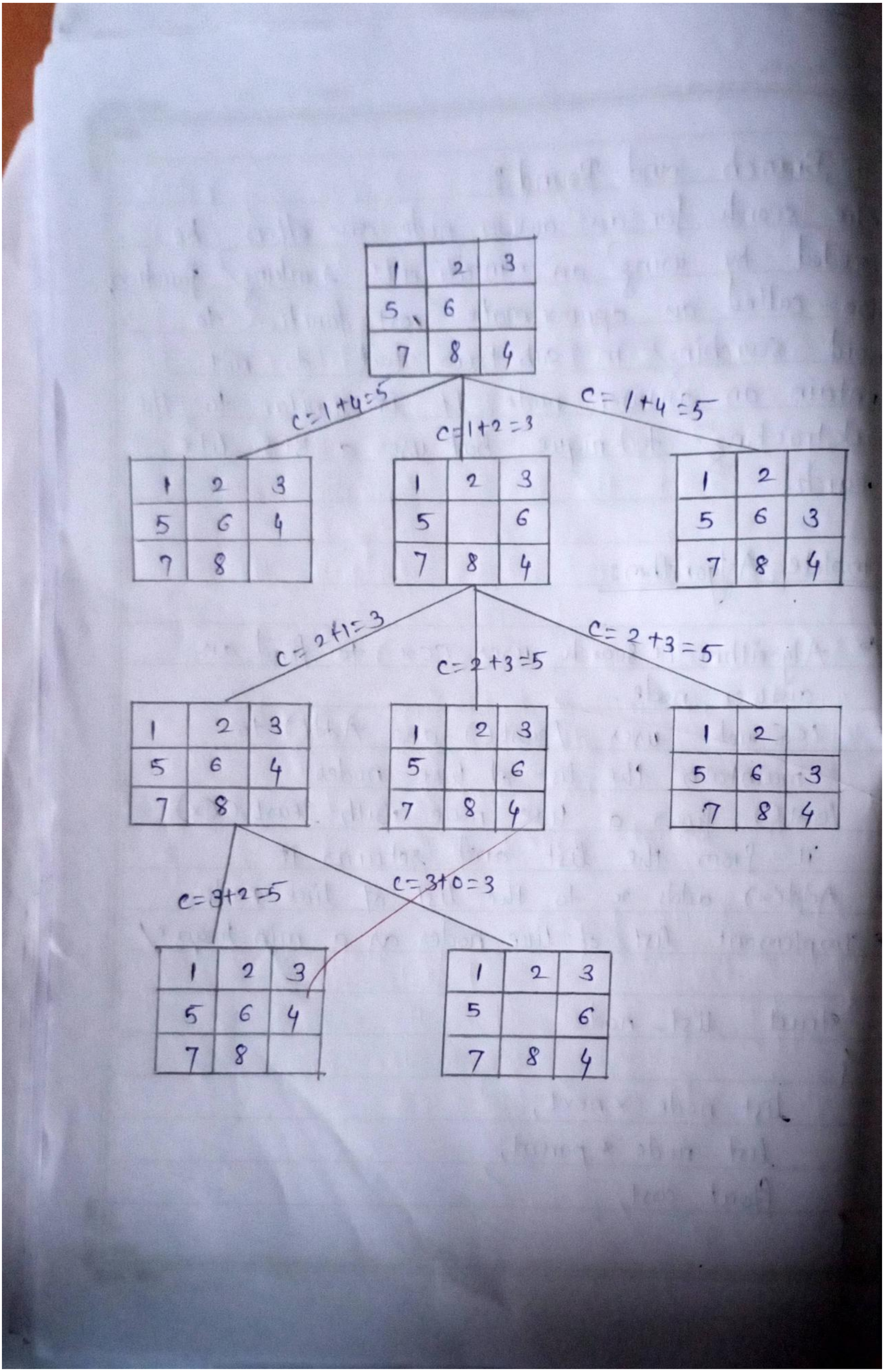
Vaishali

Initial Configuration

1	2	3
5	6	
7	8	4

Final Configuration

1	2	3
5	8	6
	7	4



```
algorithm LCSearch (list-node *t)
{
   if (*t is an answer node)
   {
      print (*t);
      return;
   }
   E = t;

Initialize the list of line nodes to be empty;
while (true)
{
   for each child x of E
   {
      if x is an answer node
      {
         print the path from x to t;
         return;
      }
      Add (x);
      x -> parent = E;
   }
   if there are no more live nodes
   {
      print ("No answer node");
      return;
   }
```

```
E = least();
// The found node is deleted from the list
// of live nodes.

}

}
```

Conclusion: Hence, we learnt the implementation of 8 puzzle problem.

30/10/22

Page No

Date / /20

Aim: Apply branch and bound techniques travelling salesman problem.

Theory: Given a set of cities and distance between every pair of cities, the problem is to find the shortest path tour that visits every cities exactly once and returns to the starting point.

Branch and Bound Solution:

In branch and bound method, for current node in tree, we compute a bound on best possible solution that we can get it we down the node. If the bound on best possible solution itself is worse than current best, then we ignore the subtree rooted with the node.

Note that the cost through a node includes 2 costs.

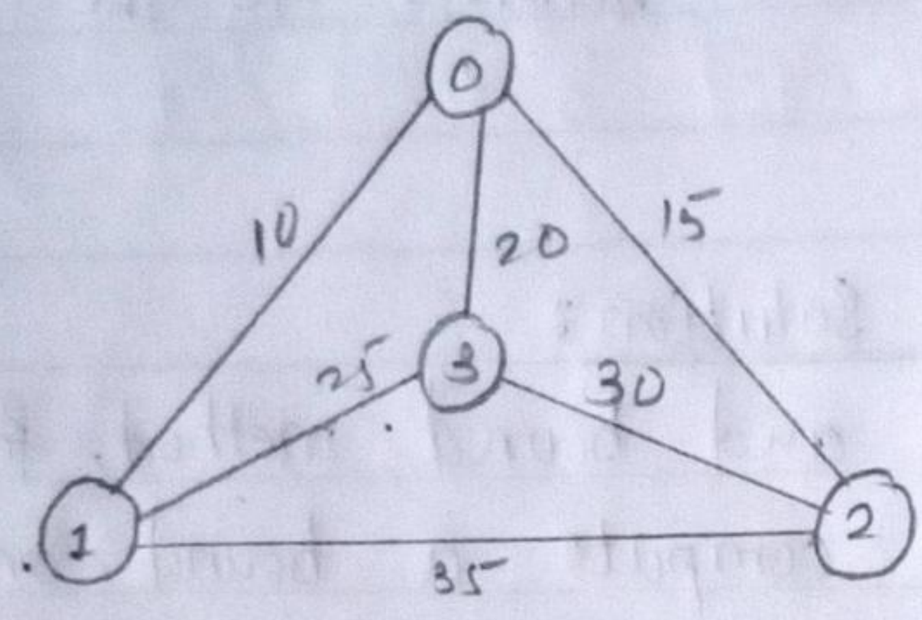
1) Cost of reaching the node from the root.

2) Cost of reaching the node from the root.

- Cost of reaching an answer from current node to a leaf.

In case of a maximization problem, an upper bound tells us the maximum possible solution if we follow the given node.

Vaishali



TSP tour is 0-1-3-2-0

The cost of tour is 10+25+30+15 which is 80

Page No.

Date: / /20

- In cases of a minimization problem, a lower bound tells us the minimum possible solution if we follow the given node, in branch bound. The challenging part is figuring out a way to compute a bound on best possible solution. Below is an idea used to compute bounds for Travelling Salesman Problem.

Cost of any tour can be written as below:

Cost of a Tour $T = (1/2) * \Sigma$ (sum of cost of two edges adjacent to u and in Tour T) where $u \in V$

For every vertex u, if we consider two edges through it in T, and sum their costs the overall sum for all vertices would be Twice of cost of Tour T (we have considered every edge twice).

(Sum of two tour edges adjacent to u) >=
(Sum of minimum weight two edges adjacent to u).

Cost of any tour >= $1/2 \Sigma$ (Sum of cost of two minimum weight edges adjacent to u) where $u \in V$.

Vaishali

Page No.

Date / /20

for example, consider the above shown graph, Below are minimum cost two edges adjacent for every node.

Node	least cost edges	Total cost
0	(0,1), (0,2)	25
1	(0,1), (1,3)	35
2	(0,2), (2,3)	45
3	(0,3), (1,3)	45

Thus, a lower bound on the cost of any tour

$= 1/2 < (25+35) + 45+45)$

$= \underline{75}$

Now, we have an idea about computation of lower bound. let us see how to apply it state space search tree. we start enumerating all possible nodes.

1. The Root Node: Without loss of enumerates all possible vertices we can go which are 1, 2, 3, ... n consider we ar calculating for vertex 1, Since we moved from 0 to 1, our tour has now included the edge 0-1. This allows us to make necessary changes in the lower bound of the root.

Vaishali

Page No.

Date: / /20

lower Bound for vertex 1 =

Old lower bound - ((minimum edge cost of
of minimum edge cost of 1)/2) +
(edge cost 0-1)

How does it work? To include edge 0-1, we add the edge cost of 0-1, and subtract and edge weight such that the lower bound remains as tight as possible which would be the sum of the minimum edges of 0 and 1 divided by 2 clearly, The edge subtracted can't be smaller than this.

Dealing with other levels: → As we move on to the next levels, we again enumerate all possible vertices. For the above case going further after 1, we check out for 2, 3, 4, ... n. Consider a lower bound for 2 as we moved from 1 to 1 we include the edge 1-2 to the tour. and after the new lower bound for this node.

lower bound (2) = old lower bound - (1 second minimum edge cost of 2)/2) + edge cost of 1-2)

Conclusion: Hence, we have implemented Branch and Bound.

Vaishali

Page No.

Date: / /20

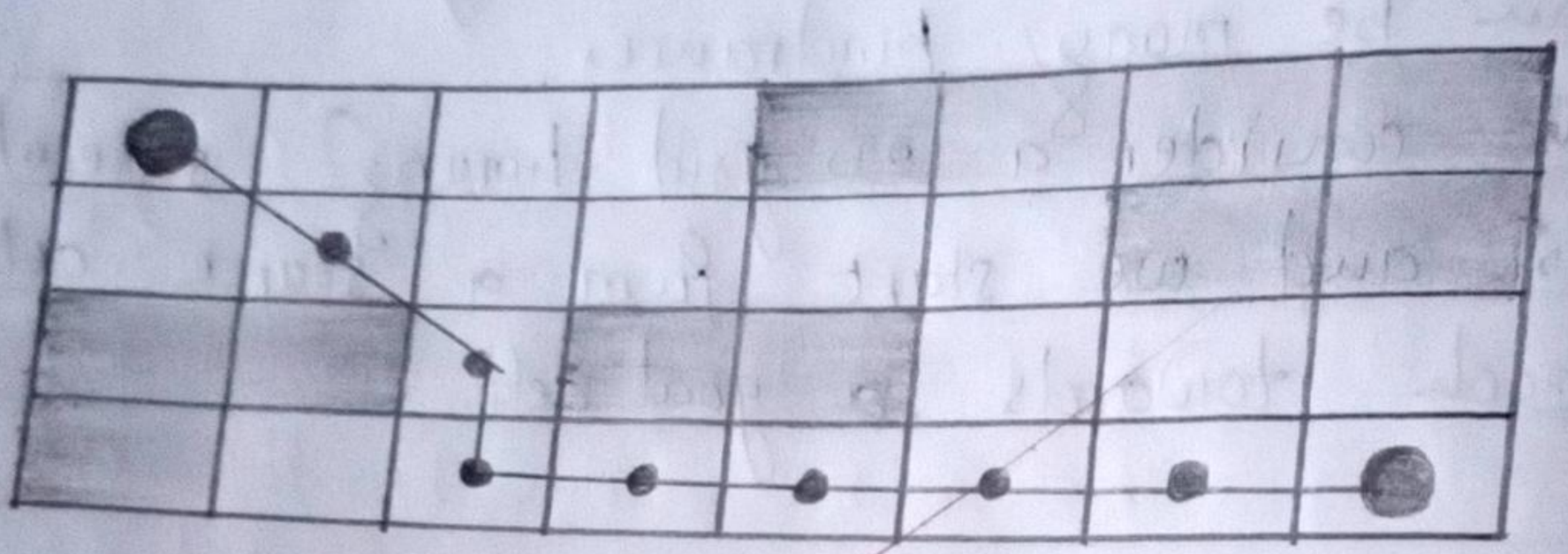
Aim: Implement A* Search Algorithm

Theory: To approximate the shortest path in real-life situations, like in maps, games where there can be many hindrances.
we can consider a 2D grid having several obstacles and we start from a source cell to reach towards a goal cell.

What is A* Search Algorithm?
A* Search Algorithm is one of the best and popular technique used in path-finding and graph traversals.

Explanation: Consider a square grid having many obstacles and we are given a starting cell and a target cell. we want to reach the target cell if possible from the starting cell as quickly as possible.
Here A* search algorithm comes to the rescue What A* Search Algorithm does is that at each step it picks the node according to a value - 'f' which is a parameter equal to the sum of two other parameters - 'g' and 'h'. At each step it picks the node/cell. we define 'g'

Vaishali

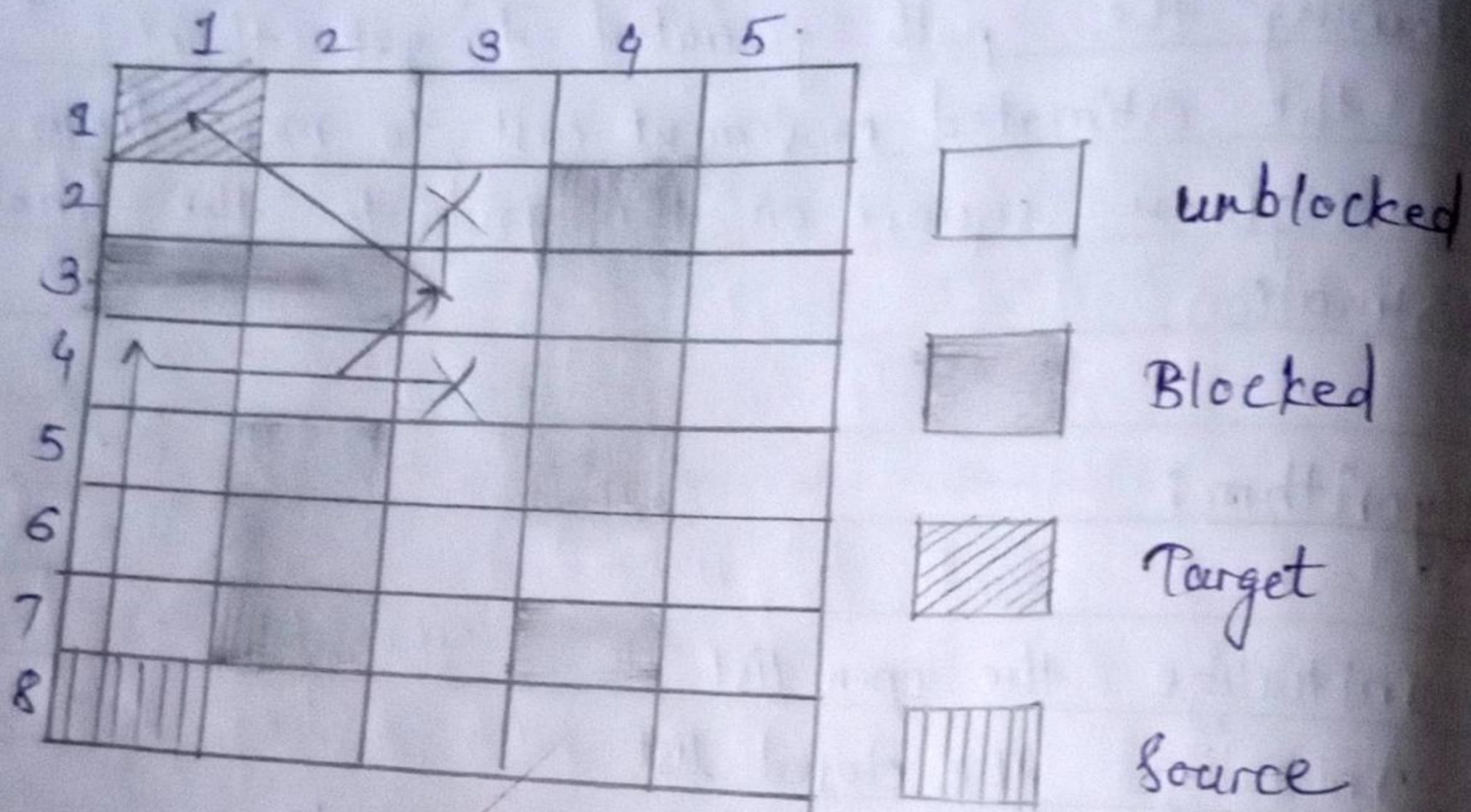


and 'h' as simply as possible.
g= the movement cost to move from that starting point to a given square on the grid, following the path generated to get there.
h= the estimated movement cost to move from that given square on the grid to the final destination.

Algorithm:

1. Initialize the open list
2. Initialize the closed list
3. While the open list is not empty
   a) Find the node with the least f on the open list, call it "q".
   b) pop q off the open list.
   c) generate q's 8 successors and set their parents to q.
   d) for each successor
      i) if successor is the goal, stop search.
      ii) else, compute both g and h for successor
      iii) if a node with the same position as successor is in the closed list which has a lower f than successor, skip this successor.

Vaishali



A* Search Algorithm

iv) if a node with same position as successor is in the closed list which has a lower f than successor, skip this successor otherwise, add the node to the open list.
end (for loop)
c) push q on the closed list
end (while loop).

Heuristics:

We can calculate g but how to calculate h?
A) Either calculate the exact value of h
OR
B) Appr. the value of h using some heuristic.

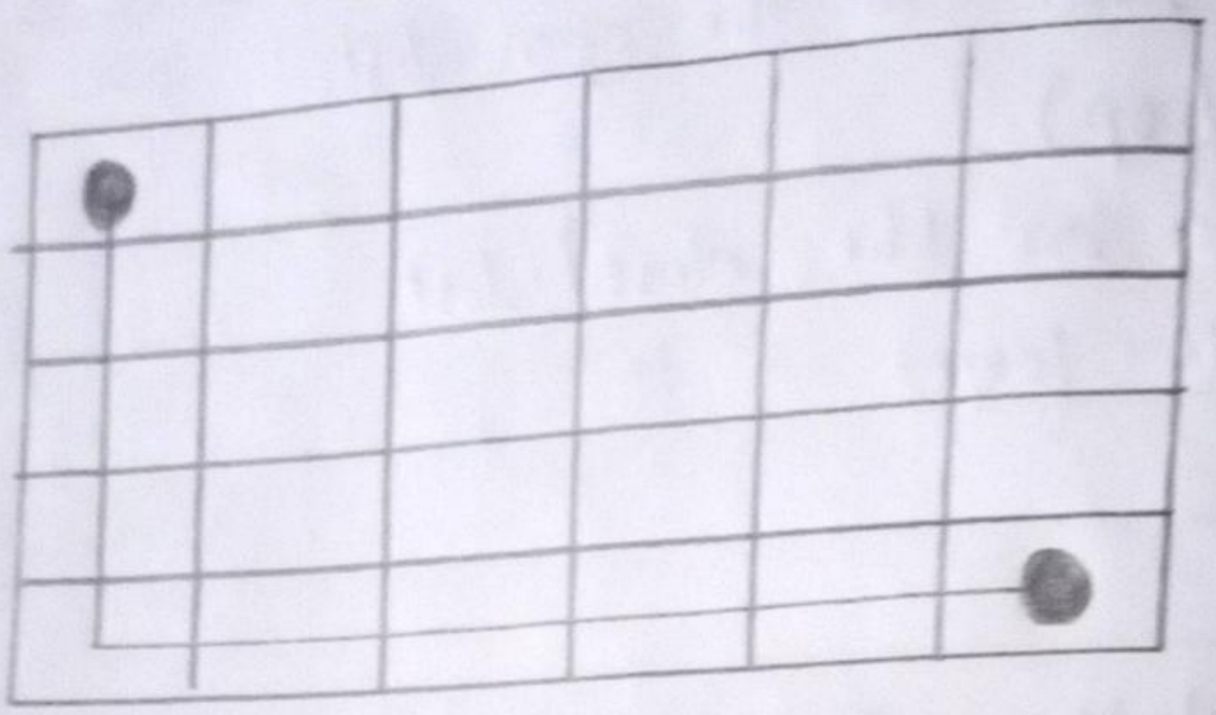
A) Exact Heuristic:

1) Pre-compute the distance between each pair of cells before running the A* search algorithm
2) If there are no blocked cells/obstacles then we can just find the exact value of h without any pre-computation.

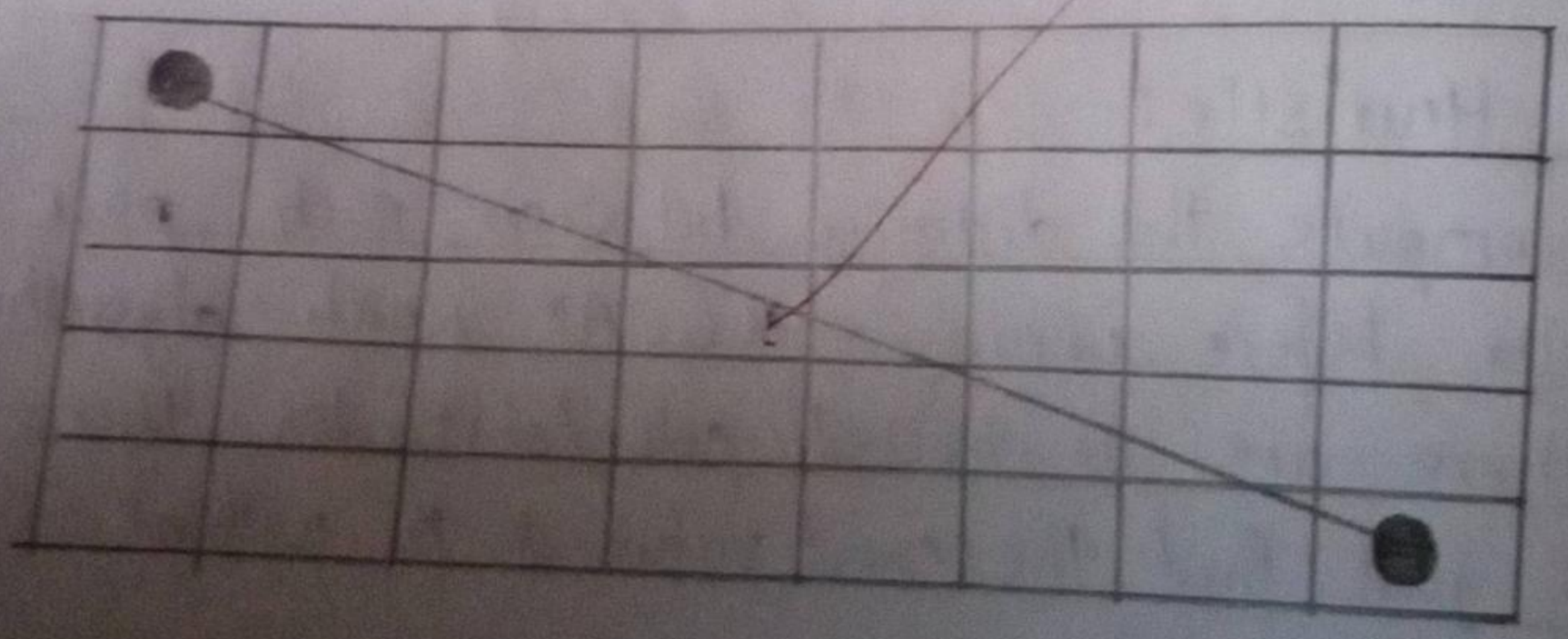
B) Approximation Heuristics:

There are generally three approximation heuristics to calculate h -

Vaishali



Manhattan Distance Heuristic



Euclidean Distance Heuristic

Page No.

Date: / /20

1) Manhattan Distance:

- It is nothing but the sum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively.

2) Diagonal Distance:

- It is nothing but the maximum of absolute values of differences in the goals x and y coordinates and the current cell's x and y coordinates respectively.

3) Euclidean Distance:

- As it is clear from its name, it is nothing but the distance between the current cell and the goal cell using the distance formula.

Conclusion: Hence, we learned and implemented A* search algorithm.

3/10/22

Page No.

Date : / /20

Aim: Apply backtracking in N-queen problem

Theory: The N-queen is the problem of placing N chess queen on an NXN chess board so that no two queen attack each other.

Backtracking Algorithm:

The idea is to place queen one by one in different columns, starting from the left most column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is the no clash. We mark this row and column as part of that solution. If we do not find such a row due to clashes. Then we, backtract and return false.

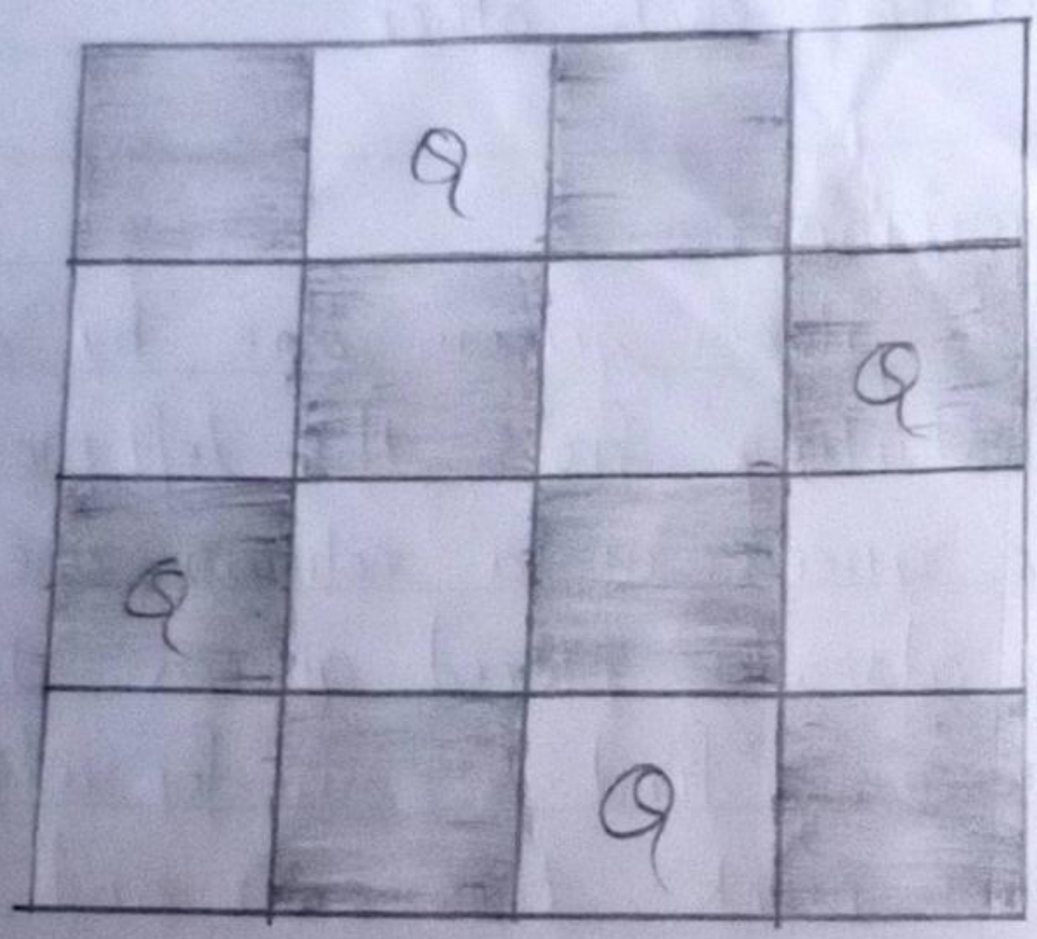
Algorithm:

1) Start in the leftmost column.
2) If all queens are placed; return true
3) Try all rows in the current column.
   Do following for every third row.
   a) If the queen can be placed safely in this row then mark this [row, column]

Vaishali

Example :

4 queen Problem



The expected output is binary matrix that has 1s for the blocks where queens are placed

Output Matrix :

{ 0, 1, 0, 0}

{ 0, 0, 0, 1}

{ 1, 0, 0, 0}

{ 0, 0, 1, 0}

as part of the solution and recursively check if placing queen here leads to a solution.
b) If placing the queen in [row, column] leads to a solution then return true.
c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
d) If all rows have been tried and nothing worked, return false to trigger backtracking.

Time Complexity : O(N!)
Space Complexity : O(N²)

Conclusion : Hence, we applied backtracking to N-queen problem.

17/11/22

Page No.

Date / /20

Aim: Design and implementation of Naive - Bayes algorithm to find the probability of playing a golf or not playing it.

Theory:

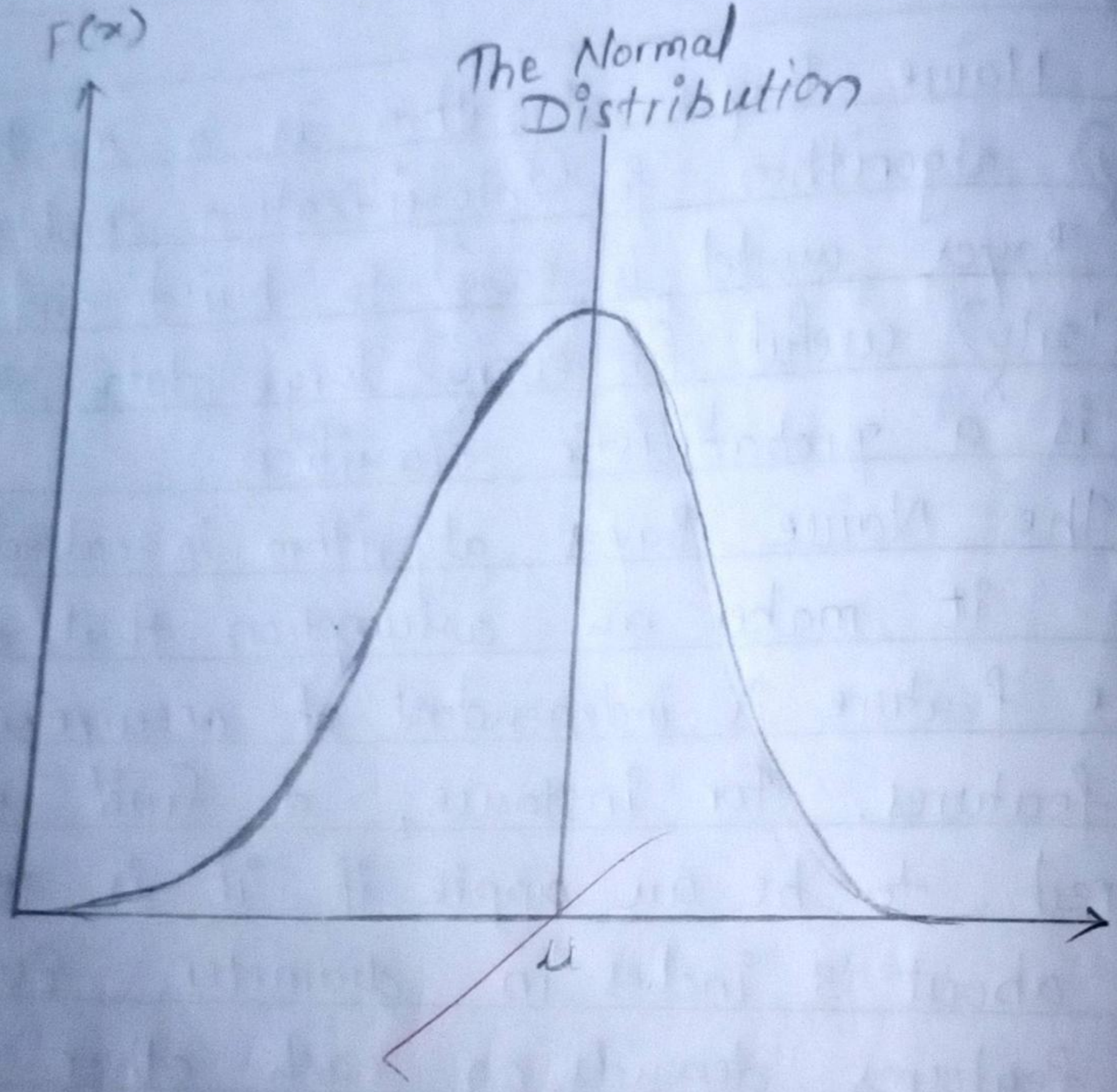
The Naive Bayes algorithm is a machine learning algorithm for classification problems. Naive Bayes model is easy to build and particularly useful for every large data sets. It is a probabilistic classifier.

The Naive Bayes algorithm is called 'Naive' because it makes an assumption that occurrence of one feature is independent of occurrence of other features. For instance, a fruit may be considered to be an apple if it is red, round and about 3 inches in diameter. Even if these features depends on each other or upon the existence of the other features. all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'naive.'

Furthermore, this classifier is based on the bayes theorem, which is why it is called naive bayes algorithm.

Vaishali

The Normal Distribution



Bayes Theorem:

Bayes theorem provides a way of calculating posterior probality $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. look at the equation below:

$$P(c|x) = \frac{P(x|c)\,P(c)}{P(x)}$$

Here,

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes)
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Gaussian Naive Bayes Classifier:

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution A gaussian distribution is also called normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

The likelihood of the features is assumed

Vaishali

to be Gaussian, hence, conditional probability is given by :

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma^2 y}} \exp\left(\frac{-(x_i - \mu_y)^2}{2\sigma^2 y}\right)$$

Conclusion :

Hence, we have implemented naive bayes algorithm to find the probability of playing a golf or not playing it.

Page No

Date / /20

Aim: Perform regression on given House price dataset considering one variable (area) & multiple variable.

Theory: linear regression is a supervised machine learning algorithm. It is a statistical method that is used for predictive analysis. linear regression makes predictions for continuous / real or numeric variables.

linear regression algorithm shows a linear relationship between a dependent (y) & one or more independent variables (x), hence called as linear regression.

The linear regression model provides a shaped straight line representing the relationship between the variables

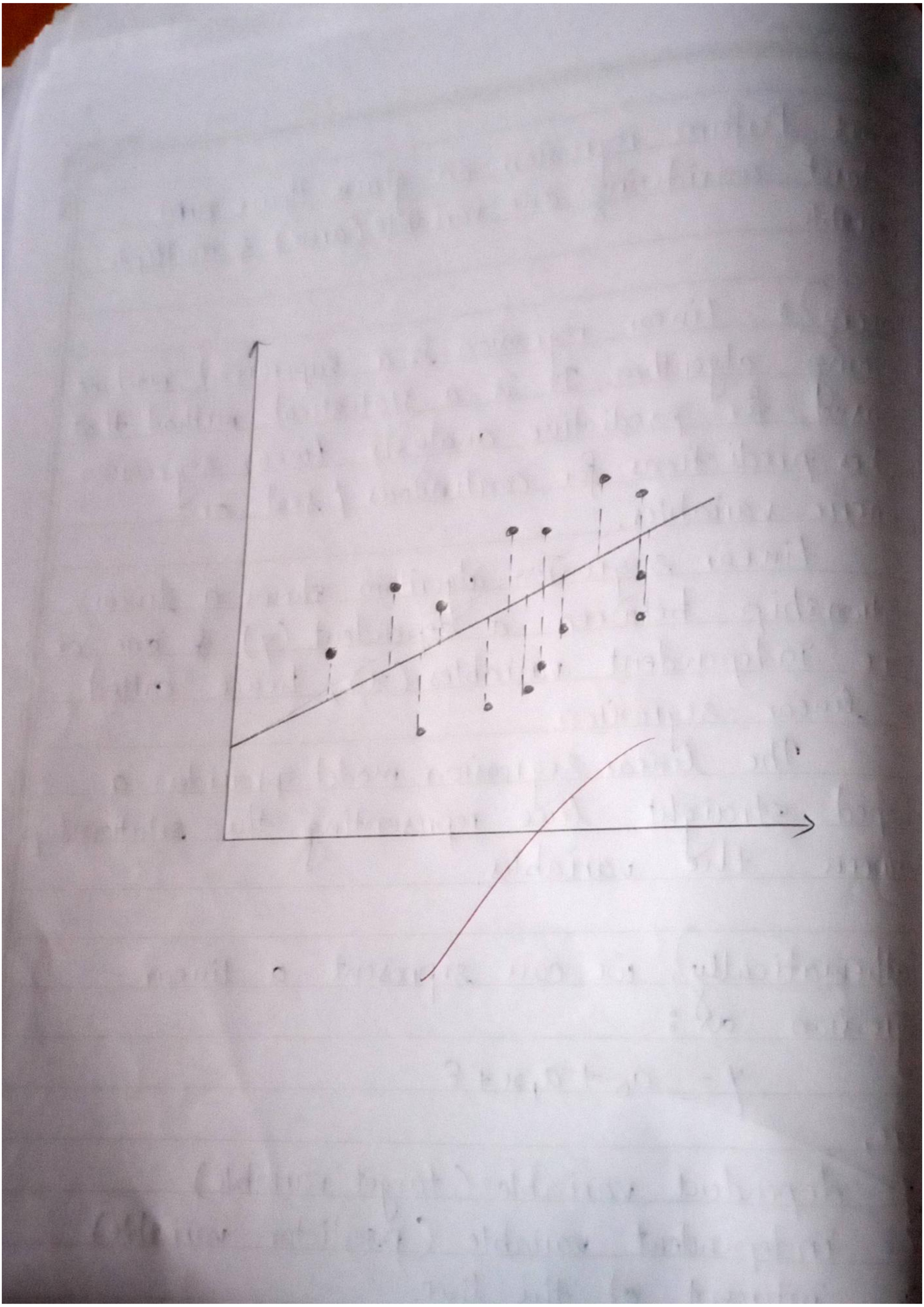
Mathematically, we can represent a linear regression as:

$$y = a_0 + a_1 x + \varepsilon$$

Here,

y = dependent variable (target variable)
x = independent variable (predictor variable)
a_0 = intercept of the line
a_1 = linear regression coefficient
ε = Error.

Vaishali



Page No.

Date / /20

Multivariable Regression:

A multivariable regression is a bit more complex than other procedures. In the below equation, w stands for weights of coefficients which requires to be elaborated.
All variables x_1, x_2 and x_3 information attributes of the observation

$$Y(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + w_0$$

House Price Prediction:

Now lets consider every step for the house price prediction using linear regression. consider a company of real estate with datasets containing the property prices of specific region. The price of a property is based on essential factors like bedrooms, area & parking. Majorly, a real estate company requires,

- Find the variable that affects the price of a house.

Vaishali

Page No.

Date: / /20

- Creating a linear model quantitatively related to the house price & variables like area, no. of rooms & bathroom.
- For finding the accuracy of a model that means how well the variables can predict the prices of house.

Conclusion:

Hence, we studied linear regression & implemented house price prediction.

14/11/22

Page No

Date / /20

Aim: Implement K-means & KNN clustering algorithm to given data set by varying the number of clusters and compare the result.

Theory):

K-Means Algorithm: The K-means algorithm is an unsupervised clustering algorithm. It takes a bunch of unlabeled points and tries to group them into "K" number of cluster.

Algorithm:

Step I: Determine K value by Elbow method and specify the number of clusters K.

Step II: Randomly assign each data point to a cluster.

Step III: Determine the cluster centroid coordinates

Step IV: Determine the distances of each data point to the centroids and re-assign each point to the closest cluster centroid based upon minimum distance.

Vaishali

Page No.

Date: / /20

Step V : Calculate cluster centroids again.

Step VI : Repeat step 4 and 5 until we reach global optima where no improvements are possible and no switching of data points from one cluster to other.

KNN Algorithm

The k-nearest neighbors algorithm is a supervised classification algorithm. It takes a bunch of labeled points and uses them to learn how to label other points. To label a new point, it looks at the labeled points closest to that new point which are its nearest neighbors, and has those neighbors vote. So, whichever label, the most of the neighbors have is the label for the new point. Here "K" in K-Nearest Neighbors is the number of neighbors it checks. It is supervised because you are trying to classify a point based on the known classification of other points.

Algorithm:

Step I: Determine the value for K.

Vaishali

Page No.

Date / /20

Step II : Calculate the distances between the new input (test data) and all the training data. The most commonly used metrics for calculating distance are Euclidean, Manhattan and Minkowski

Step III : Sort the distance and determine k nearest neighbors based on minimum distance values.

Step IV : Analyze the category of those neighbors and assign the category for the the data based on majority vote.

Step V : Return the predicted class.

Conclusion : Hence, we have Implemented K-means and KNN clustering algorithm

Vaishali

Page No.

Date / /20

Aim: Use heuristic search techniques to implement Hill climbing algorithm

Theory:

Hill climbing:

It solves the problems where we need to maximize or minimize a given real function by choosing value from the given inputs.

Heuristic Search:

This search algorithms may not find the optimal solution to the problem. However, it will give a good solution in a reasonable time.

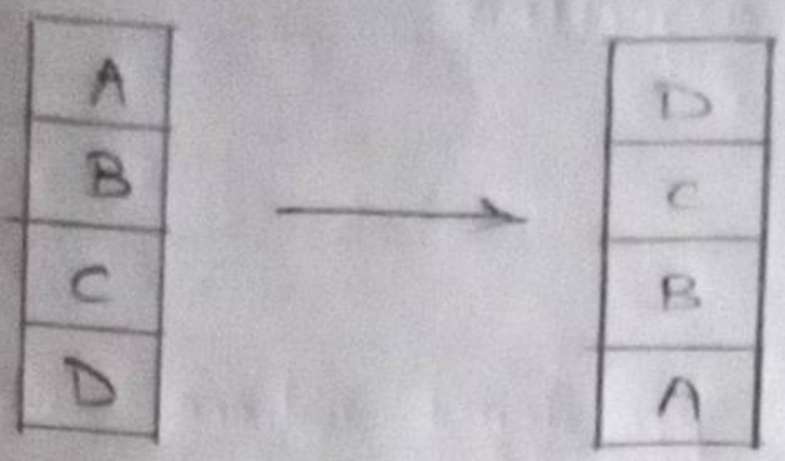
It is also called greedy local search as it only looks to its good immediate neighbour state and not beyond that.

A node of hill climbing algorithm has 2 components which are state and value.

Hill climbing is mostly used when a good heuristic is available.

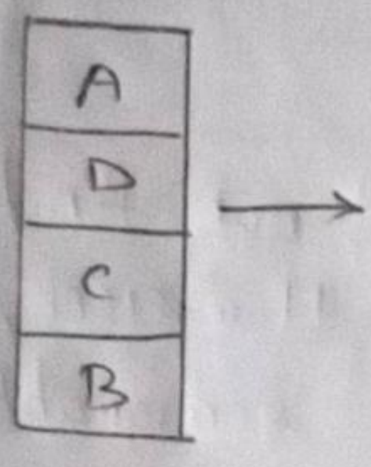
Vaishali

Example :

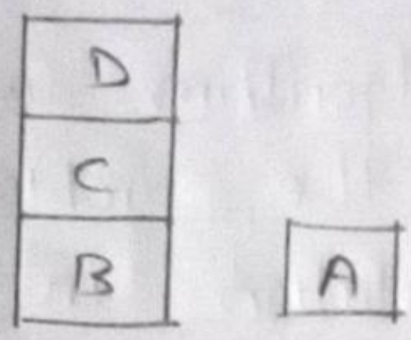


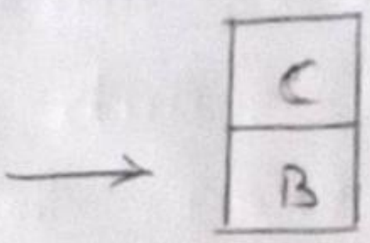
$h(1) = -6$

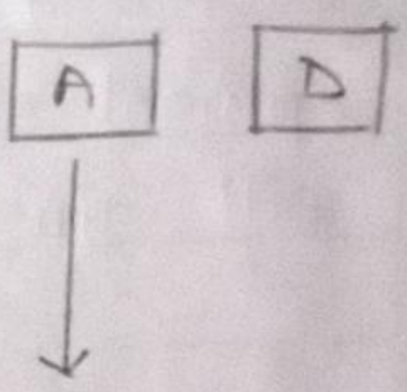
$h(3) = -1$



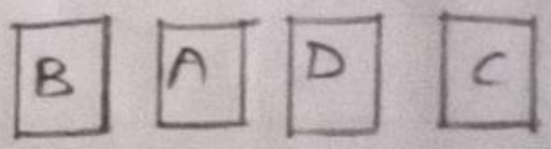
$h(2) = -3$



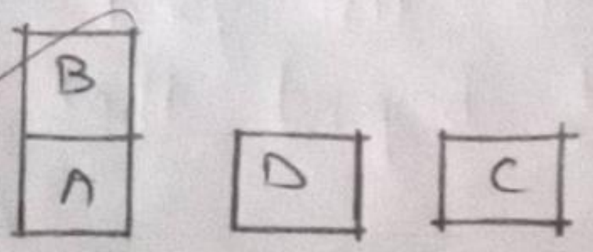


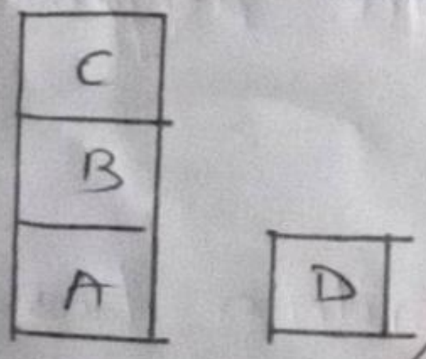


$h(4) = 0$

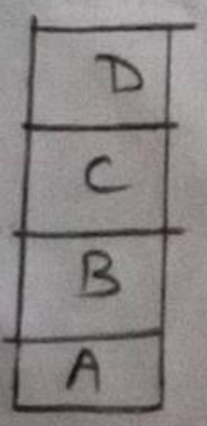


$h(5) = +1$





$h(6) = +3$



$h(7) = +6$

Page No

Date: / /20

Algorithm:

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

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

Step III: Select and apply an operator to current state.

Step IV: Check new state.

a) If it is goal state, then return success and quit.

b) Else if it is better than the current state the assign new state as a current state.

c) Else if not better than the current state, then return to step 2.

Step V: Exit

Conclusion: Hence, we used Heuristic search techniques to implement Hill Climbing algorithm.

14/11/22

Vaishali

Government College of Engineering, Aurangabad.

LIST OF EXPERIMENT IN Artificial Intelligence

CONTENTS

SR. NO.	TITLE	PAGE	DATE	REMARKS
01	Implement Tic-Tac-Toe Game.	1	29/8/22	
02	Solve 8-Puzzle Problem	5	5/9/22	
03.	Apply Branch and Bound	9	13/9/22	
04	Implement A* Search Algorithm	11	19/9/22	
05.	N-queen Problem	13	3/10/22	
06	Design & Implementation of Naive Bayes	15	10/10/22	
07	Perform Regression	18	17/10/22	
08	Implement K-means and KNN Algo	21	7/11/22	
09	Hill Climbing Algorithm	24	14/11/22	

30/11/22

Vaishali