Artificial Intelligence Laboratory
End-Semester Report
TeamLoki

Abhishek Singh	Gaurav Singh	Harish Kumar	Janmejay	Varun Gupta
201851005	201851044	201851046	201851053	201851141

CONTENTS

I. Week 9: 22-26 March, 2021 Lab Assignment 9

Learning Objective: Understanding Exploitation - Exploration in simple n-arm bandit reinforcement learning task, epsilon-greedy algorithm.

Problem(1) Consider a binary bandit with two rewards 1-success, 0-failure. The bandit returns 1 or 0 for the action that you select, i.e. 1 or 2. The rewards are stochastic (but stationary). Use an epsilon-greedy algorithm discussed in class and decide upon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.

Solution: We have used e-greedy algorithm and in the banditA we have given probability of 0.1 and 0.2 of getting reward 1 else 0 for each action, so after some iteration it will reach the expected reward near to 0.1 and 0.2 respectively for each action and the same inference also apply to the banditB in which we have 0.8 and 0.9 probability of getting reward of 1 else 0 for each action after some iteration it will reach to the expected reward near to 0.8 and 0.9 respectively for each action.

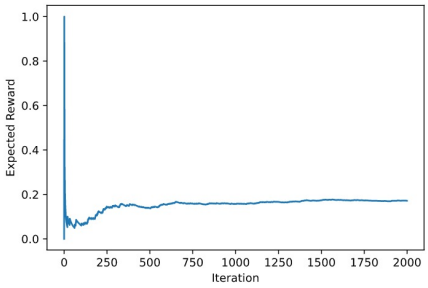

Fig. 1. Bandit-A

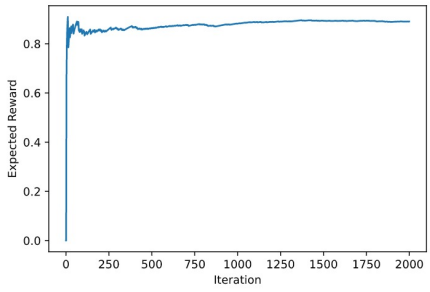

Fig. 2. Bandit-B

Problem(2) Develop a 10-armed bandit in which all ten mean-rewards start out equal and then take independent random walks (by adding a normally distributed increment with mean zero and standard deviation 0.01 to all mean-rewards on each time step). function [value] = bandit-nonstat(action).
Solution: In this case, we have initialized the mean rewards array by 1 and we applied an e-greedy algorithm in which it performed both exploration based on some epsilon value probability, and most time we exploit the same action with 1 - epsilon so that bandit will not stick with the same action. For each iteration, we generate an array of ten values so that they are normally distributed with mean 0 and std 0.01 and add it to mean reward, and choose the reward for any action from this mean reward and mean reward gets update each time. The rewards given in this case are non-stationary.

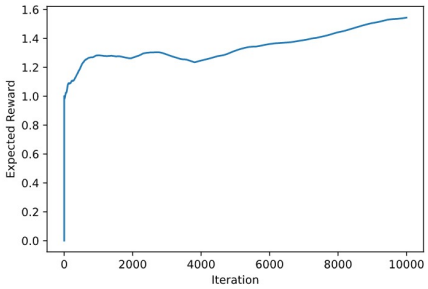

Fig. 3.

Problem(3) The 10-armed bandit that you developed (bandit-nonstat) is difficult to crack with a standard epsilon-greedy algorithm since the rewards are non-stationary. We did discuss how to track non-stationary rewards in class. Write a modified epsilon-greedy agent and show whether it is able to latch onto correct actions or not. (Try at least 10000 time steps before commenting on results)

Solution: In the above, we faced a non-stationary problem so we will change our strategy of the averaging method to update our estimation of action reward to more recent samples will be important and hence we could use a constant discounting factor alpha = 0.6.

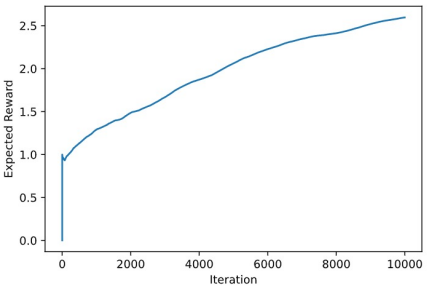

Fig. 4.

Code Link: code for bandit

II. Week 8 : 15 March - 19 March 2021 Lab Assignment - 8

Learning Objective: Basics of data structure needed for state space search tasks and use of random numbers required for MDP and RL

(Problem Statement:) Read the reference on MENACE by Michie and check for its implementations.Pick the one that you like the most and go through the code carefully.Highlights the parts that you feel crucial.If possible,try to code the MENACE in any programming language of your liking.
Solution:
Here are some highlighted parts from the paper.

The problem of artificial intelligence consists in the reduction of these processes to the elementary operations of arithmetic and logic.The work consists of trial and error learning based and mental tasks consist of the game Tic-tac-toe. If the player is able to learn from experience, the choices which have led up to a given outcome receive reinforcements in the light of the outcome value.Trial and Error uses the concepts and terminology of the experimental psychologist.MENACE stands for Matchbox Educable Noughts And Crosses Engine.

The position encounter by an opening player is represented by a separate box. The face bears the drawing of position and code number for indexing. If we want to play against machines then in order to first move, we remove the box corresponding to the opening position, shake it and tilt it forwards. The beads in the color white, lilac and gold run to the forwards.where V partition selects the first to arrive. The colors define the opening move. The human opponent will generate a fresh position. The box corresponding to this position is located, shake and tilted and then machine will move next and so on If we apply reinforcement then if the machine has done badly then it will be punished. if the machine has done well then it will be rewarded. it is obvious that the strength of reinforcement should be related to the stage of the game being maximal for terminal moves and decreasing towards the beginning.

For MENACE'S maiden tournament against a human opponent a draw was reckoned a good result, and received a unit bonus. A win was regarded as an exceptionally good result and was rewarded by three extra beads to each open box.

The reinforcement system differs from that of the matchbox machine in two main ways. First the stage of play to which a move belongs is reckoned backwards from the end of the play. Thus, the opening move of a long play might stand as much as eight moves from the end and hence receive relatively mild reinforcement, since the strength of reinforcement decays for moves successively further from the end.

Secondly, it is related to moving probability. computer program handles these in the form of odds, where odds equal to p divides 1-p, where p being the probability with which a given move is selected. Figure shows the adjustment of multipliers to sliding origin. The figure shows the 3 different outcomes WIN, DRAWN, LOST corresponding value of Rn which decides the Reinforcement. Here mu is calculated from decay factor which is represented as D and Mn is the un-adjusted multiplier for nth stage.

OUTCOME	REINFORCEMENT
Won	$R_n = M_n^{-\mu+1}$
Drawn	$R_n = M_n^{-\mu}$
Lost	$R_n = M_n^{-\mu-1}$

Fig. 5. Outcome and their Rn value

The table shows the different value of Rn for Reinforcement learning for different values of outcomes like win,drawn or lost.We can say that value for Won and lost are same and for drawn it is different.The values are depends on the mu which is a parameter.which decides the value of reinforcement.
Here are some parts that are implemented using Python Programming language.Here is the link for the some functions used in MENACE problem.
Code Link: code for menace problem

III. Week 2 : 18 Jan - 22 Jan 2021 Lab Assignment - 2

Learning Objective: To understand the use of Heuristic function for reducing the size of the search space. Explore non-classical search algorithms for large problems.

A: Read about the game of marble solitaire. Figure shows the initial board configuration. The goal is to reach the board configuration where only one marble is left at the centre. To solve marble solitaire,

A valid move is to jump a peg orthogonally over an adjacent peg into a hole two positions away and then to remove the jumped peg. (1) Priority queue based search considering path cost,

(2) Suggest two different heuristic functions with justification,

Manhattan This heuristic's value of a board is the sum, for each peg on the board, of the Manhattan distance from the center.

Exponential Distance This is similar, but with a larger bias towards the center. If H,V are the horizontal and vertical distances from the center respectively, then the heuristic's value is 2max(H,V).

(3) Implement best first search algorithm,

```
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
        only element
  explored ← an empty set
  loop do
    if EMPTY?( frontier)then return
                        failure
    /* chooses the shallowest node in
                        frontier*/
    node ← POP(frontier)
     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)
```

(4) Implement A*,

The A* algorithm turned out to be very good with a sufficiently fine tuned heuristic. There is a variety of methods to estimate how "promising" a given board is. All of them have to do with trying to concentrate the pegs on the center (when we played it ourselves it became evident that leaving pegs isolated in one of the edges was a bad strategy, and one should always try and concentrate them on the center).

B: Write a program to randomly generate k-SAT problems. The program must accept values for k, m the number of clauses in the formula, and n the number of variables. Each clause of length k must contain distinct variables or their negation. Instances generated by this algorithm belong to fixed clause length models of SAT and are known as uniform random k-SAT problems.

This is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In this the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.If this is the case, the formula is called satisfiable .

We have taken the number of variables n , number of clause m and number of variable in one clause k. Then we randomly selected English alphabet characters and used there negations.

Code Link: code for K-sat problem generator

C: Write programs to solve a set of uniform random 3-SAT problems for different combinations of m and n, and compare their performance. Try the Hill-Climbing, Beam-Search with beam widths 3 and 4, Variable-Neighborhood-Descent with 3 neighborhood functions. Use two different heuristic functions and compare them with respect to penetrance .

In 3-Sat clause is limited to at most three literals. we have used the k-sat generation function implemented before then we implemented the hill -Climbing, Beam-Search with beam widths 3 and 4, Variable-Neighborhood-Descent with 3 neighborhood functions .

3-SAT Problem performace comparison for different variables and clause

		number of states Explored			
Number of Variables n	Number of Clause m	Hill Climbing	Beam Search beam width 3	Beam Search beam width 4	Variable Nei.. Descent : 3
3	3	1	1	1	1
3	5	1	1	1	1
4	5	1	1	1	1
5	9	1	1	1	1
5	13	2	4	5	4
6	12	1	1	1	1
6	15	2	4	5	4
9	9	1	1	1	1
9	22	2	4	5	4
12	23	2	4	5	4
13	13	3	13	21	13

Fig. 6. 3-sat problem performance comparison

The figure shows the performance comparison between different algorithms for solving 3-sat satisfiability problem for different combinations of number of variables n and number of clause m. **Code Link:** code for 3-Sat problem solver

IV. Week 7 : 8 - 12 March, 2021 Lab Assignment for FUN

Learning Objective: Objective: (unsupervised learning) Smoothing filter as a clustering mechanism, Hierarchical Agglomerative Clustering

Problem:(1) Given the noisy image test noisy.jpg (available in the codes folder). Use the adaptive smoothing (Ref. Diffusion and Confusions in Signal and Image Processing) process to remove the noise. Explain the selected values of the parameters in the algorithm.

Solution: An image is a two-dimensional analog of the discrete-time series. Denote the gray value at a pixel I(i,j) = I(xi, yj) and we assume equal spacing in the x and y directions. Denoising the image will clearly necessitate a smoothing process. The special nature of images demand the average to be taken over pixels that "resemble" the center pixel I(i,j). This average should be taken over projections of the same object and be independent of pixels that describe different objects or that are simply in a region far from the pixel of interest.

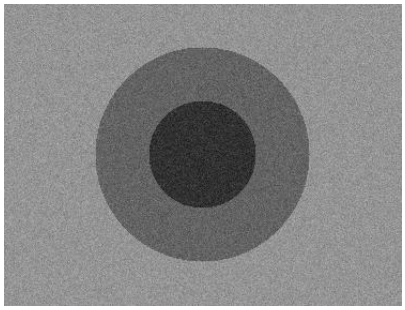

Fig. 7. Before Adaptive smoothing

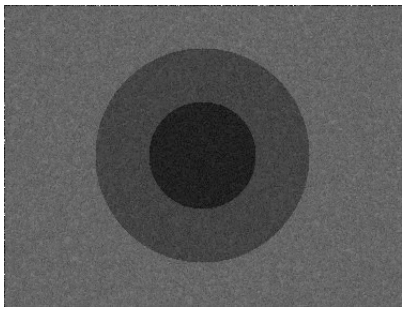

Fig. 8. After Adaptive smoothing

Code Link: code for adaptive smoothing

Problem:(2) Use HAC with Euclidean/ Manhattan distance as a measure (Single link, complete link, Ward's distance, Group average, Centroid, Clusteroid) cluster the states of India based on the feature vector comprising of the following parameters (for one of the financial year values available in the data-set)
a. Percentage of schools with electricity
b. Percentage of schools with girls toilet
c. Percentage of schools with drinking water
d. Percentage of schools with boys toilet

Solution: The process of Hierarchical Clustering involves either clustering sub-clusters(data points in the first iteration) into larger clusters in a bottom-up manner or dividing a larger cluster into smaller sub-clusters in a top-down manner. During both the types of hierarchical clustering, the distance between two sub-clusters needs to be computed. The different types of linkages describe the different approaches to measure the distance between two sub-clusters of data points

(A) Single Linkage: For two clusters R and S, the single linkage returns the minimum distance between two points i and j such that i belongs to R and j belongs to S.

(B) Complete Linkage: For two clusters R and S, the single linkage returns the maximum distance between two points i and j such that i belongs to R and j belongs to S.

(C) Average Linkage: For two clusters R and S, first for the distance between any data-point i in R and any data-point j in S and then the arithmetic mean of these distances are calculated. Average Linkage returns this value of the arithmetic mean.

(D) Ward's Method: Ward's method says that the distance between two clusters, A and B, is how much the sum of squares will increase when we merge them.

(E) Centroid's Method: In centroid method, the distance between two clusters is the distance between the two mean vectors of the clusters. At each stage of the process we combine the two clusters that have the smallest centroid distance.

Code Link: code for HAC

V. REFERENCES

1) MIT Paper for MENACE Problem
2) http://people.csail.mit.edu/brooks/idocs/matchbox.pdf
3) Artificial Intelligence – A Modern Approach, S. Russell, P. Norvig, Pearson Education, New Delhi, 2002.
4) https://www.cs.huji.ac.il/ ai/projects/2013/PegSolitaire
5) Sochen, N., Kimmel, R. Bruckstein, A. Diffusions and Confusions in Signal and Image Processing. Journal of Mathematical Imaging and Vision 14, 195–209 (2001). https://doi.org/10.1023/A:1011277827470