**CSCI 4450/8456 Introduction to Artificial Intelligence**

**Instructor: Dasgupta**

**Homework 4**

**Full Points: 80**

Given on November 8, 2018 (Thursday)

Due on November 20, 2018 (Tuesday) 11:59 PM via loki

Submission directory CSCI-4450-DG-F17-A4. As usual, include a README text file with instructions on how to compile and run your code for programming problems.

**IMPORTANT READ**: Note on programming language used: As mentioned from the first assignment, it is your responsibility to make sure that your program compiles and runs properly on loki. If your submitted program does not run (version issues, etc.) on loki, you will not get points for it.

Java is the main supported language in the UNO Computer Science program and JDK1.8 is available on loki. If you are using another language, like Python, for your programming, it is your responsibility to make sure that your submitted programs run with the versions of Python available on loki, which are python 3.4.3 and python 2.7.3. For grading, we will not be able to run your programs on different systems in case they do not run on loki.

**Problem 1. Maze Navigation Reinforcement Learning** **(40 points)**

In class, we discussed three different approaches for reinforcement learning (RL): direct utility estimates, adaptive dynamic programming and temporal difference learning. In this question, we will explore these RL algorithms for maze navigation

For the maze, assume bottom left cell is at (1, 1) and the format for the coordinates is (<col>, <row>)

1. 4 X 3 world with one obstacle in (2, 2), reward +1 at (4, 3) and -1 at (4, 2) (same environment as in textbook Figure 17.1 (a))
2. 10 X 10 world with no obstacles and reward +1 at (5, 5)
3. 10 X 10 world with four obstacles at (4, 4) (6, 4) (4, 6) (6, 6) and reward +1 at (5, 5),
4. 10 X 10 world with four obstacles at (4, 4) (6, 4) (4, 6) (6, 6) and reward +1 at (5, 5), reward -1 at (5, 7) and (4, 5)

Write a program that prompts the user via a start menu to first select the world (a) – (d) above, and then to select the RL algorithm (1) DUE, (2) ADP, (3) TD. Your program should select an appropriate number of trials or epochs to learn the utilities and/or model.

When the algorithm finishes, your program should again prompt the user to input a start state (two integer coordinates separated by a space, with check for input being a valid state – inside environment, not obstacle). From this start state, your agent should navigate until it reaches a terminal state (correct operation should reach the +1 terminal state). Your program should then printout the coordinates of the states the agent navigated through until it reached the terminal state.

Remember to return the program to the start menu after each run, and add an exit option to the start menu, so that the program can be tested multiple times.

**Problem 2. Tic-tac-toe with Q-learning (40 points)**

Write a program that uses Q-learning to teach a learning agent how to play tic-tac-toe. Once the training is complete, the program uses this agent to play tic-tac-toe against a human player. There are many online resources available for solving this problem, some resources are below. You can refer to those for ideas about implementing, but cannot directly copy those solutions. Your program will be checked against plagiarism against available online solutions. Your program will be scored on the interface your design for your game and the quality of the solution -how well it learned to play the game determined by how it performs against a human opponent.

**Resources for RL based tic-tac-toe game play learning** lass report giving some implementation ideas: <http://www.cs.dartmouth.edu/~lorenzo/teaching/cs134/Archive/Spring2009/final/PengTao/final_report.pdf>

**Notes on RL-based Algorithm Implementation** RL algorithms like direct utility estimation (DUE), Adaptive Dynamic Programming (ADP) and Temporal Difference (TD) have similar objective – learn the utility of a policy. In other words, they calculate the utility of each state in the state space for a given policy – this is the output for each of these algorithms. The policy that the algorithm is determining the utility for is given as input to each algorithm. As you know, the way to go about doing this is to simulate several trials or epochs. Each epoch is a training sequence. In each epoch the agent starts from any non-terminal state in the environment and takes successive actions until it reaches a terminal state. The algorithms differ in how they calculate the utility of the states, as discussed in class and as given in the textbook. Below are a few notes on the setup, or how the DUE/ADP/TD algorithm would fit into the overall program.

Q: Within an epoch, how is the action that takes the agent to the next state determined?

A: It is given by the policy that is provided as input to the DUE/ADP/TD algorithm. Although the policy gives a specific action for each state, the actual action taken by the agent is determined by the transition model (e.g., 80% of time it goes UP when it wants to do UP, and 10% of the time it goes LEFT when it wants to do UP and 10% of the time it goes RIGHT when it wants to do UP). Note that the RL does not know this transition model, but the agent’s movement, which is a simulation of its real-life behavior effects this transition model. So, in your program, you need to have a method called, for example, simulateMove, that takes as input the current state of the agent and action for that state given by the input policy, implements the transition model and returns the actual action that the agent will take.

Q: Where does the input policy come from?

A: It is given as input. It remains fixed for all epochs or trials. Although it is given as input, like the action taken at each state inside a training sequence (above question), the policy is also simulating the actual behavior of the agent. So, in the example of the 4 X 3 world maze navigation, the input policy would be the policy calculated by MDP value or policy iteration with R(s) = -0.04, as shown in the diagram alongside (same as Figure 21.1(a)). For your assignment Q1, you have to use the same 4 X 3 world and same policy, as in this figure.