CCBR 2020 : Introduction to Reinforcement Learning

Assignment #2 **Max marks: 45**

Deadline: 26th February 2020 23:55 hrs

Instructions:

1. This is an individual assignment. Collaborations and discussions are strictly prohibited.
2. **Any sort of plagiarism/cheating will be dealt very strictly. Acknowledge any source used.**
3. You have to turn in the well-documented code along with a detailed report of the results of the experiments.
4. Be precise with your explanations. Avoid verbosity. Place relevant results that bolster your conclusions. Report should be **within 10 pages** in a single column format, and with 11pt font. Results/conclusions beyond the page limit will be ignored during evaluation.
5. You can use any programming language for this assignment. However, we recommend Python or MATLAB.
6. Please email your report and code to TAs (Nirav Bhavsar, Ajay Pandey).

1 Taxi Driver Problem

Consider a problem of a taxi driver, who serves three cities A, B and C. The taxi driver can find a new ride by choosing one of the following actions.

1. Cruise the streets looking for a passenger.
2. Go to the nearest taxi stand and wait in line.
3. Wait for a call from the dispatcher (this is not possible in town B because of poor reception).

For a given town and a given action, there is a probability that the next trip will go to each of the towns A, B and C and a corresponding reward in monetary units associated with each such trip. This reward represents the income from the trip after all necessary expenses have been deducted. Please refer Table 1 below for the rewards and transition probabilities. In Table 1 below, p_{ij}^k is the probability of getting a ride to town j, by choosing an action k while the driver was in town i and r_{ij}^k is the immediate reward of getting a ride to town j, by choosing an action k while the driver was in town i.

Answer the following, for discount factors β ranging from 0 to 0.95 with intervals of 0.05: (10 + 10 + 10 marks)

1. Implement *Value Iteration* and tabulate optimal policy π^* and state values $J(s)$ as a function of discount factor β.
2. Implement the *Q-Learning* and tabulate optimal policy π^* and state-action values $Q(s, a)$ as a function of discount factor β.
3. Consider a policy that always forces the driver to go to the nearest taxi stand, irrespective of the state. Implement *TD Learning* and tabulate optimal policy π^* and state values $J(s)$ as a function of discount factor β.

Table 1: Taxi Problem: Probabilities and Rewards

Town i	**Actions** k	**Probabilities** p_{ij}^k j = A B C	**Rewards** r_{ij}^k j = A B C
A	1 2 3	$\begin{array}{ccc} A & B & C \\ 1/2 & 1/4 & 1/4 \\ 1/16 & 3/4 & 3/16 \\ 1/4 & 1/8 & 5/8 \end{array}$	$\begin{array}{ccc} A & B & C \\ 10 & 4 & 8 \\ 8 & 2 & 4 \\ 4 & 6 & 4 \end{array}$
B	1 2	$\begin{array}{ccc} A & B & C \\ 1/2 & 0 & 1/2 \\ 1/16 & 7/8 & 1/16 \end{array}$	$\begin{array}{ccc} A & B & C \\ 14 & 0 & 18 \\ 8 & 16 & 8 \end{array}$
C	1 2 3	$\begin{array}{ccc} A & B & C \\ 1/4 & 1/4 & 1/2 \\ 1/8 & 3/4 & 1/8 \\ 3/4 & 1/16 & 3/16 \end{array}$	$\begin{array}{ccc} A & B & C \\ 10 & 2 & 8 \\ 6 & 4 & 2 \\ 4 & 0 & 8 \end{array}$

2 Mountain Car Problem

We will be using python and OpenAI Gym to solve the Mountain Car problem. This link leads to the page on 'Getting started with Gym'. Please install OpenAI Gym before starting.

In the Mountain Car environment, when the problem begins the car is dropped into the valley and given an initial position and velocity as a vector. This is the **car's state**. Your agent must then tell the car to take one of three actions: **drive left, do nothing, or drive right**. This action is sent to the Mountain Car environment algorithm which returns a new state (position and velocity) as well as a reward. For each step that the car does not reach the goal, located at position 0.5, the environment returns a reward of -1. Use these rewards to solve the Mountain Car problem using **Q-Learning** algorithm.

1. Implement Q-learning algorithm with a discount factor β of 0.9. (3 marks)
2. Plot the following (compute the averages over 50 independent runs): (3 + 3 + 3 + 3 marks)
   (a) Learning curve that shows how the average reward per episode changes.
   (b) Learning curve that shows how the average number of steps to goal changes.
   (c) Learning curve that shows the car's final position at the end of each episode.
   (d) The optimal policy (Position vs Velocity).