CS6700 - Reinforcement Learning
Programming Assignment 3

April 11, 2023

1 Hierarchical Reinforcement Learning

For this assignment, we will be referring to Sutton, Precup and Singh's 1999 paper, 'Between MDPs and semi-MDPs : A Framework for Temporal Abstraction in Reinforcement Learning'. Please read the paper upto and including Section 3, it is self explanatory and a good reference leading up to the understanding and implementation of SMDP Q-learning. Section 3 of the paper talks about SMDP planning and is necessary to build intuition to solve this assignment. We will be working with a simple taxi domain environment (explained in the next section). Your tasks are to implement 1-step **SMDP Q-Learning** and **intra-option Q-Learning** on this environment.

2 Environment Description

The environment for this task is the taxi domain, illustrated in Fig. 1. It is a 5x5 matrix, where each cell is a position your taxi can stay at. There is a single passenger who can be either picked up or dropped off, or is being transported. There are four designated locations in the grid world indicated by R(ed), G(reen), Y(ellow), and B(lue). When the episode starts, the taxi starts off at a random square and the passenger is at a random location. The taxi drives to the passenger's location, picks up the passenger, drives to the passenger's destination (another one of the four specified locations), and then drops off the passenger. Once the passenger is dropped off, the episode ends.

There are 500 discrete states since there are 25 taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations. Note that there are 400 states that can actually be reached during an episode. The missing states correspond to situations in which the passenger is at the same location as their destination, as this typically signals the end of an episode. Four additional states can be observed right after a successful episodes, when both the passenger and the taxi are at the destination. This gives a total of 404 reachable discrete states.

Passenger locations: 0: R(ed); 1: G(reen); 2: Y(ellow); 3: B(lue); 4: in taxi
Destinations: 0: R(ed); 1: G(reen); 2: Y(ellow); 3: B(lue)
Rewards:

- -1 per step unless other reward is triggered.
- +20 delivering passenger.
- -10 executing "pickup" and "drop-off" actions illegally.

The discount factor is taken to be $\gamma = 0.9$.

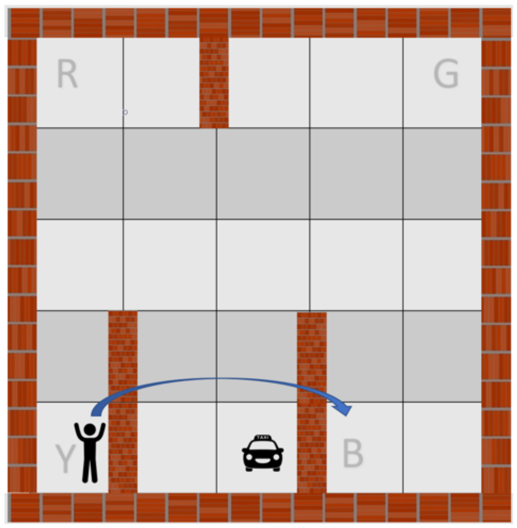

Figure 1: Taxi Domain

3 Actions and Options

Actions: There are 6 discrete deterministic actions: 0: move south; 1: move north; 2: move east; 3: move west; 4: pick passenger up; and 5: drop passenger off.
Options: Options to move the taxi to each of the four designated locations, executable when the taxi is not already there.

You will be experimenting with Gymnasium Gym's Taxi-v3 environment.

4 Tasks

First, implement the single step **SMDP Q-learning** for solving the taxi problem. A rough sketch of the algorithm is as follows: Given the set of options,

- Execute the current selected option to termination (e.g. use epsilon greedy $Q(s, o)$).
- Computer $r(s, o)$.
- Update $Q(s_t, o)$.

Second, implement **intra-option Q-Learning** on the same environment.

For each algorithm, do the following (only for the configuration with the best hyperparameters):

1. Plot reward curves and visualize the learned Q-values.
2. Provide a written description of the policies learnt and your reasoning behind why the respective algorithm learns the policy.

3. Is there an alternate set of options that you can use to solve this problem, such that this set and the given options to move the taxi are mutually exclusive? If so, run both algorithms with this alternate set of options and compare performance with the algorithms run on the options to move the taxi.

Finally, provide a comparison between the SMDP Q-Learning and intra-option Q-Learning algorithms. Do you observe any improvement with intra-option Q-Learning? If so, describe why this happens as well. Please make sure that all descriptions are **brief** and to the point.

5 Submission Instructions

You are required to submit both your report and your code. Please submit in **teams of two**. One submission per team will suffice. The due date for this programming assignment is **11:59 pm on April 25.**