Planning in the Taxi Domain

Overview

This exercise involves modeling a sequential decision-making problem. We consider the Taxi Domain problem that involves controlling a taxi agent that can pick up and drop passengers in a grid world. We will implement algorithms for arriving at a good policy offline using dynamic programming and later implement basic reinforcement learning methods for online learning. You are requested to submit your implementation and a report that includes the insights gained via this exercise.

The Taxi Domain

a. A Taxi agent is situated in a 5x5 grid world (see figure below). The goal of the taxi agent is to pick up a passenger from an initial grid cell and drop the passenger at a destination grid cell. There are four specially-designated locations in this grid world, called *depots*, that are denoted as R, G, B and Y (initials for Red, Green, Blue and Yellow). The taxi agent is shown using a curved symbol. There are a few walls situated along grid boundaries (indicated by thick black lines) which prevent the taxi from moving across grid cells. The entire grid is assumed to be enclosed by walls along the left, right, top and bottom sides.

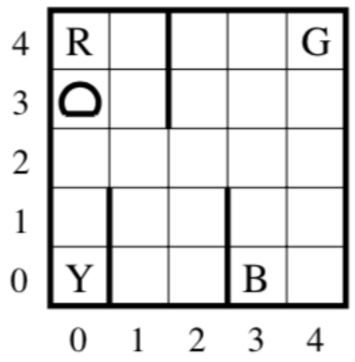

Figure: The Taxi Domain (5x5 grid world). The taxi is indicated by a curved symbol. There are 4 depots indicated as R, G, B and Y. The taxi agent aims at transporting passengers from the source depot to the destination depot.

b. There are six actions in the domain: (a) four *navigation* actions that move the taxi one grid cell to the *North*, *South*, *East* or the *West* directions, (b) a *Pickup* action, where the taxi attempts to pick up the passenger and (c) a *Putdown* action, where the taxi drops off the passenger. Each navigation action succeeds with a probability of 0.85 and moves in a random other direction with probability 0.15. Movements that attempt to cross the boundary of the grid or a wall results in no change in the state. The *Pickup* and the *Putdown* actions are deterministic and lead to their exact intended effects.

c. The taxi agent receives a reward of (-1) for performing each action. A reward of (+20) is received when the taxi agent successfully delivers the passenger at the destination grid cell. Further, a reward of (-10) is received if the taxi attempts to *Pickup* or *Putdown* a passenger when the taxi and the passenger are not located in the same grid cell.

d. The taxi agent interacts in the environment in episodes. In each episode, the taxi starts in a randomly chosen grid cell. There is a passenger at one of the four depot locations chosen randomly and that passenger wishes to be transported to one of the other depot locations. The destination is different from the source and also selected randomly. The taxi must move towards the passenger's grid cell (called the source), pick up the passenger, go to the destination location (called the destination), and drop the passenger there. The episode ends when the passenger is deposited at the destination location. Note that dropping the passenger in a location *other* than the destination will not terminate the episode.

Part A: Computing Policies

In this section, we model the taxi domain as a Markov Decision Process (MDP) and compute optimal policy for the taxi agent.

1. Formulate the taxi domain as an MDP
    a. Describe the state space, the action space, the transition model and the reward model for the problem in the report.
    b. Implement a simulator for the taxi domain which includes a model of the environment and determines the next state based on the action taken by the agent. The stochastic effects of navigation actions are simulated as part of the environment model. The agent should receive an instantaneous reward after taking each action.
    c. An instance of the taxi domain problem consists of a starting depot for the passenger, selecting a different destination depot and selecting a starting location for the taxi in the grid.
2. Implement Value Iteration for the taxi domain.
    a. The implementation should take as input an instance of the taxi domain and a parameter epsilon (denoting the maximum error allowed in the value of any state) and return the optimal policy. Use the *max-norm* distance[1] in the successive value functions to determine convergence. Initially, set the discount factor as 0.9. Report the value of epsilon you have chosen and the number of iterations required for the convergence in your write-up.
    b. Next, we study the connection between the discount factor and the rate of convergence. Repeat part (a) by varying the discount factor in the range {0.01, 0.1, 0.5, 0.8, 0.99}. For each discount factor considered, plot the iteration index on the x-axis and the *max-norm* distance along the y-axis. Describe your observations.
    c. Pick an instance such as follows: Y (passenger initial location), G (passenger destination location) and R (taxi initial location). Simulate the policy obtained for the discount factor gamma as 0.1 and 0.99. Determine the first 20 states and actions prescribed by the policy. Examine the two state-action sequences. Report any differences in the execution traces for the policies obtained for the two discount factors. Repeat the runs by keeping the goal the same and varying the start states (for the taxi and the passenger).

[1] Reference: AIMA Chapter 17.

3. Implement Policy Iteration for the problem
    a. How would you implement the Policy Evaluation step? Implement a linear algebra method and an iterative method for Policy Evaluation and report when one approach would be better than the other.
    b. Run policy iteration till convergence to determine the optimal policy. Repeat the process by computing the policy loss at each iteration between the policy at each iteration the final policy at convergence. Plot the policy loss (along y-axis) and iteration number (x-axis). Study the behaviour by varying the discount factor in the range {0.01, 0.1, 0.5, 0.8, 0.99}.

Part B: Incorporating Learning

In this section, assume that we have an *unknown* taxi domain MDP where the transition model and the reward model is not known to the taxi agent. In this case, the taxi must act only based on its past experience. Assume that the next state and reward is provided by the simulated environment when the taxi agent executes an action.

1. Implement the following (model-free) approaches to learn the optimal policy. Implement
    a. Implement Q-learning that uses a 1-step greedy look-ahead policy for the Q-learning updates. Incorporate an epsilon-greedy policy for exploration with a fixed exploration rate (epsilon) of 0.1.
    b. Implement Q-learning with an epsilon-greedy policy with a decaying exploration rate. The exploration is initialized with epsilon as 0.1 and is decayed, inversely proportional to the iteration number. Here, a single iteration refers to a single Q-learning update.
    c. Implement *State-Action-State-Action-Reward* (SARSA) for this problem. Use an epsilon-greedy policy for exploration with a fixed exploration rate (epsilon) of 0.1.
    d. As in part (b) implement SARSA with a decaying exploration rate.
2. Execute the above algorithms for *at least* 2000 episodes. Initialize the above methods with a learning rate (alpha) as 0.25 and use a discount factor (gamma) of 0.99. Assume that each episode begins from a randomly selected feasible state and terminates if the taxi agent reaches the goal state or the episode reaches a maximum length of 500 steps. Evaluate each algorithm by computing the sum of discounted rewards accumulated during the episode (averaged over 10 different runs). Plot the sum of discounted rewards accumulated during an episode (along y-axis) against the number of training episodes (along x-axis). Analyse the convergence of the algorithms.
3. From part 2, select the learning algorithm that converges to the highest accumulated reward. Execute the policy on 5 problem instances by varying the initial depot location of the passenger and the initial depot location of the taxi agent. What do you observe?
4. Consider the Q-learning method implemented in 1a. Plot the sum of discounted rewards accumulated during an episode against the number of training episodes by varying the exploration rate (epsilon) as {0, 0.05, 0.1, 0.5, 0.9} with a learning rate (alpha) kept as 0.1. Analyze the impact of using a high or a low exploration rate. Next, vary the learning rate (alpha) as {0.1, 0.2, 0.3, 0.4, 0.5} keeping the exploration rate (epsilon) as 0.1.
5. Consider the 10x10 extension of the taxi domain problem (see figure below). Select the best learning algorithm from your experiments above and try the learner on the 10x10 problem. Assume a maximum of 10,000 learning iterations. Report the accumulated

discounted reward for the learned policy from your method. Average over at least 5 instances sampling the initial states for the passenger and the taxi and the destination location for the taxi among the depots.

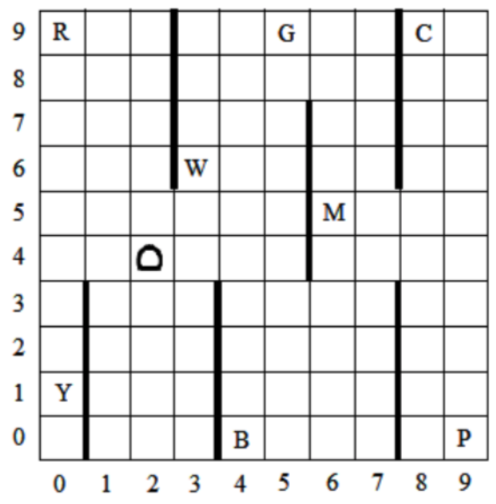

Figure: The 10x10 version of the Taxi Domain. The taxi is indicated by a curved symbol. There are 8 depots indicated as R, G, B, Y, C, P, W, M. The taxi agent aims at transporting passengers from source to destination locations.

Submission Instructions

- **This assignment is to be done individually or in pairs. The choice is entirely yours.**
- The assignment is to be submitted on Moodle. Exactly one of the team members needs to submit the zip file. The implementation should be contained in a single file named - ***A3.py.*** The implementation should be accompanied with a report in pdf format as ***report.pdf***.
- Please submit a single file named ***<A3-EntryNumber1-EntryNumber2>.zip*** or ***<A3-EntryNumber>.zip***. Exactly one of the team members needs to submit the zip file. Please note that '.rar' etc. are not permitted. Upon unzipping, this should yield a single folder with the same name as the zip file and inside the folder should be a python file named - A3.py and a pdf file named - report.pdf. The assignment is to be submitted on Moodle. Exactly one of the team members needs to submit the zip file.
- **The submission deadline is 5pm on November 26, 2021.**
- This assignment will carry 11% of the grade. This is the final assignment for the course.
- Please use Python 3.6 for your implementation. Your code should use only standard python libraries. Please do not include any dependencies on third party libraries or existing algorithmic implementations.
- Please ensure the reproducibility of graphs/results (modulo any probabilistic behaviour) for your code when run on our system. Include reasonable comments for your code to understand the implementation at a high-level. Ensure that each part of the assignment is immediately (one-click) runnable. For example, if a TA asks you to run Part A (2b) in your submission, then you should be able to run the sub-part immediately in the presence of the TA.
- The report should include the results/graphs that you produce working through the assignment. The report should be prepared thoughtfully and not simply a cut/paste of graphs. For each graph or result that you obtain, provide a brief description for the observation or insight.
- Please submit by the submission date. **No late submissions** can be accommodated.
- Please submit work only from your own efforts. Do not look at or refer to code written by anyone else. You may discuss the problem, however the code implementation must be original. Discussion will not be grounds to justify software plagiarism. Please do not copy existing assignment solutions from the internet: your submission will be compared against them using plagiarism detection software. Copying and cheating will result in a penalty of at least -10 (absolute). The department and institute guidelines will apply. More severe penalties such as F grade will follow.
- Queries (if any) should be raised on Piazza.