CS-368 Reinforcement Learning
Handed on: October 1, 2024
Due Date: October 27, 2024 (11:59PM)
Assignment 1

Instructions:

- What to hand in: A full report answering all the questions. You can copy paste the screenshot of your codes. Make sure the code is readable. Code should be properly commented on explaining important details. Results from the console should be shown (**mandatory**). A few lines to explain the screenshots.
- You should hand in your assignment as below (follow the same naming convention and subdirectories)
    - firstname-lastname-assignment-1.zip
        - assignment-1-report.pdf
        - code-files
            - policy_iteration_with_stochastic_policy.py
            - policy_iteration_with_deterministic_policy.py
            - value_iteration.py
            - value_iteration_with_changed_discount.py
            - value_iteration_with_changed_transition_probabilities.py
            - value_iteration_with_changed_reward.py
- The deadline for the first assignment **will not** be extended.
- Remember Turnitin plagiarism detection software will be used. Therefore, don't put yourself in any trouble.
- **Assignment 1 is mapped on CLO-1**

In this assignment, you will solve for the optimal policy and values of a small MDP, given by the following story. You will program both value iteration and policy iteration algorithms in python.

A little autonomous rover on Mars depends on its solar panels for energy. Its goal is to collect as much energy as possible. It is close to a small hill; it can drive at the top of the hill, but it has a tendency to roll off the hill. Specifically, the rover can be in one of three states: on top of the hill, rolling down the hill, or at the bottom of the hill. There are two actions that it can take in each state: drive or don't drive.

If the rover is at the top of the hill and drives, then it is still at the top of the hill in the next period with probability 0.5 with +2 reward and rolling down in the next period with probability 0.5 with +2 reward. If it is at the top of the hill and does not drive, these probabilities are 0.5 and 0.5 with +3 and +1 reward, respectively.

If it is rolling down and drives, then with probability 0.3 it is at the top of the hill in the next period with +2 reward, with probability 0.4 it is still rolling down in the next period with +1.5 reward, and with probability 0.3 it is at the bottom in the next period with +0.5 reward. If it is rolling down and does not drive, then with probability 1 it is at the bottom in the next period and gets a +1 reward.

Finally, if it is at the bottom of the hill and drives, then in the next period it is at the top of the hill with probability 0.5, and at the bottom with probability 0.5. In both cases, it gets +2 reward. If it does not drive, then with probability 1 it is at the bottom in the next period with +1 reward.

1. (10 points). Draw the MDP graphically (handmade can also work).
2. (25 points). Using a discount factor of 0.9, solve the MDP using value iteration (until the values have become reasonably stable). You should start with the values set to zero. You should show both the optimal policy and the optimal values.
3. (40 points). Using a discount factor of 0.9, solve the MDP using policy iteration (until you have complete convergence). You should start once with a stochastic random policy that is all actions have equal probability (0.5 for driving and 0.5 for not driving) and once with a deterministic policy which never drives in any state. Again, you should show both the optimal policy and the optimal values (and of course they should be the same as in 2...). Your implementation should be in such a way that any deterministic policy can also be considered in the start and your program can still find the optimal policy with minimal changes required.
4. (25 points). Change the MDP in three different ways: by changing the discount factor, e.g., 0.75, changing the transition probabilities for a single action from a single state, and by changing a reward for a single action at a single state. Each of these changes should be performed separately starting at the original MDP, resulting in three new MDPs (which you do not have to draw), each of which is different from the original MDP in a single way. In each case, the change should be so that the optimal policy changes, and you should state what the optimal policy becomes and give a short intuitive argument for this.