Assignment 2

Due Date 6pm, October 18, 2023.

Submission Instructions Please submit your response in two files via Canvas:

- a `pdf` file that includes your answers to all the problems, and
- a `zip` file that includes your completed `Jupyter` notebooks.

The completed `Jupyter` notebooks should include missing code as well as the code output.

Accompanying Code The assignment includes an accompanying `zip` file with

- starter code (for which you need to fill in the missing code),
- test code (to help you develop your solution, available for a subset of problems), and
- grader code (to evaluate your solution, please do not change this code)

for the programming problems. The code is packaged in two `Jupyter` notebooks. To execute and complete this starter code you will need Python 3 and `Jupyter`. You can use either your local Python 3 run-time or the one provided by Google Colaboratory (available using your Rice NetID). If you have any questions about working with the code, please contact the teaching staff.

Collaboration Policy Collaborating on assignments is permitted, provided the submission lists the students who you collaborated with. As per the Rice Honor System Handbook, this means that students are allowed to develop answers to specific problems together and check answers with each other. However, collaboration does not confer the right for students to submit the exact same document— students must write down the answers themselves. While the "core" of the answer can be the same, the wording cannot be identical unless precise wording is necessary to answer the question. Students must be able to demonstrate that they worked together when developing responses and that one student did not copy off the other. This means that students are barred from dividing questions among themselves.

Citation Policy Students don't have an obligation to cite class slides, lectures, or class textbooks on assignments; these are considered common knowledge. An academic citation style is required for all other sources (such as research articles and blogs).

Late Policy Assignments should be turned on time via Canvas. Assignments handed in late will be marked off 10% per day. Assignments more than 3 days late will not be accepted. In turn, you can expect the teaching staff to grade your assignments and provide feedback in a timely manner.

Grading The assignment is worth 20% of your final grade.

Updates to the Assignment In case there are any updates to the assignment (e.g., additional clarifications, typo fixes, hints, etc.), they will be indicated via the following table.

Version	Date	Note
v1	September 28	Assignment released

1. Policy Evaluation 20pts

1.1 5pts

In the class, we derived the following relation between the Q_π and V_π values of a policy π

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V_\pi(s'). \tag{1}$$

In this equation, $R(s,a)$ denotes the expected value of reward received after selecting action a in state s, i.e., $R(s,a) = \mathbb{E}[r'|s,a]$.

For the case when the reward signal is a deterministic function (denoted by $R_d(s,a,s')$) of the state, action, and next state, show that Eq. 1 can be written as

$$Q_\pi(s,a) = \sum_{s' \in S} T(s'|s,a)\{R_d(s,a,s') + \gamma V_\pi(s')\}. \tag{2}$$

Simplify the above equation for the case when the reward signal is a deterministic function (denoted by $R_d(s')$) of only the next state.

1.2 10pts

Next, we will implement the policy evaluation algorithm for the Frozen Lake environment from Gymnasium (formerly OpenAI Gym). The reward signal in the Frozen Lake environment is a deterministic function of the (state, action, next state)-tuple, so you should use Eq. 2 to inform your implementation.

Starter code for this problem is provided in the accompanying `Jupyter` notebook. The started code also includes test (to help check your answer) and a grader code (to help with the grading).

Fill in the missing code to compute the V_π values of a policy π. Report the output of the grader code for Problem 1.2.

1.3 5pts

Fill in the missing code to compute the Q_π values of a policy π. Report the output of the grader code for Problem 1.3.

2. Value Iteration 20pts

In this problem, we will use the value iteration algorithm to compute the optimal policy for the Frozen Lake environment from Gymnasium.

2.1 10pts

The accompanying `Jupyter` notebook provides starter code for the value iteration algorithm. Complete this code to compute the optimal policy π_*, V_*, and Q_*. Report the output of the grader code for Problem 2.1.

2.2 3pts

For a discount factor of $\gamma = 0.99$, generate and report the following two plots

- $\max_s |V(s) - V_{\text{previous\_iteration}}(s)|$ with respect to iteration number
- $\max_s |V(s) - V_{\text{final\_iteration}}(s)|$ with respect to iteration number

2.3 2pts

For a discount factor of $\gamma = 0.99$, report the number of iterations after which the algorithm converged. Are you certain that the value iteration algorithm has found the optimal policy? In a sentence or two, please justify your answer.

2.4 3pts

Plot the number of iterations after which the algorithm converged with respect to discount factor for $\gamma \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$. You should observe that the number of iterations is positively correlated with γ. In a sentence or two, please explain the reason behind this trend.

2.5 2pts

The number of iterations required for value iteration algorithm depends on its initialization. In the accompanying code, you are given an educated guess of the optimal policy. Use this guess to warm start the value iteration algorithm. Report the output of the grader code for Problem 2.5.

3. Policy Iteration 20pts

In this problem, we will use the policy iteration algorithm to compute the optimal policy for the Frozen Lake environment from Gymnasium.

3.1 5pts

Consider a brute force approach to solve an MDP:

Algorithm 1: A brute force algorithm for solving MDPs

```
input  : MDP $(S, A, T, R, \gamma)$
output: Optimal policy of the MDP $(\pi_*)$
$\pi_* \leftarrow$ Arbitrary policy ;                                   /* Initialization */
$V_* \leftarrow$ PolicyEvaluation$(S, A, T, R, \gamma, \pi_*)$ ;
foreach deterministic policy $(\pi)$ of the MDP do
    $V \leftarrow$ PolicyEvaluation$(S, A, T, R, \gamma, \pi)$ ;
    if $V > V_*$ then
        $\pi_* \leftarrow \pi$ ;
        $V_* \leftarrow V$ ;
    end
end
return $\pi_*$
```

Recall that for each MDP there exists an optimal policy that is deterministic. Hence, even though the algorithm is searching only in the space of deterministic policies (and not all policies), it is guaranteed to find the optimal policy.

How many policy evaluations would this brute force algorithm need to do?

Hint: Calculate the number of deterministic policies of an MDP with $|S|$ states and $|A|$ actions.

3.2 12pts

The accompanying `Jupyter` notebook provides starter code for the policy iteration algorithm. Complete this code to compute the optimal policy π_*. Report the output of the grader code for Problem 3.2.

3.3 3pts

How many policy evaluations did it take for the policy iteration algorithm to obtain the optimal policy? Is this number lower or higher than the brute force approach? In a sentence or two, please describe why.

4. Monte Carlo Control 20pts

In this problem, we will use Monte Carlo methods to learn the optimal policy for the Blackjack game implemented as part of the RL Card toolkit. Starter code for this problem is provided in the accompanying `Jupyter` notebook. Complete this code and report the output of the grader code for Problem 4.

5. Temporal Difference Learning 20pts

In this problem, we will use the Q learning algorithm to learn the optimal policy for the Blackjack game implemented as part of the RL Card toolkit. Starter code for this problem is provided in the accompanying `Jupyter` notebook. Complete this code and report the output of the grader code for Problem 5.