Homework 1

Task 1 Frozen Lake MDP

Now you will implement value iteration and policy iteration for the Frozen Lake environment from `OpenAI Gym`. We have provided custom versions of this environment.

1. **(coding)** Read through `vi_and_pi.py` and implement `policy_evaluation`, `policy_improvement` and `policy_iteration`. The stopping tolerance (defined as $max_s|V_{old}(s) - V_{new}(s)|$) is tol = 10^{-3}. Use $\gamma = 0.9$. Return the optimal value function and the optimal policy.
2. **(coding)** Implement `value_iteration` in `vi_and_pi.py`. The stopping tolerance is tol = 10^{-3}. Use $\gamma = 0.9$. Return the optimal value function and the optimal policy.

Task 2 Test Environment

It is crucial to test our code on a test environment. In this problem, you will reason about optimality in the provided test environment by hand; later, to sanity-check your code, you will verify that your implementation is able to achieve this optimality. You should be able to run your models on CPU in no more than a few minutes on the following environment:

- 4 states: 0, 1, 2, 3
- 5 actions: 0, 1, 2, 3, 4. Action $0 \leq i \leq 3$ goes to state i , while action 4 makes the agent stay in the same state.
- Rewards: Going to state i from states 0, 1 and 3 gives a reward $R(i)$, where $R(0) = 0.1, R(1) = -0.3, R(2) = 0.0, R(3) = -0.2$. If we start in state 2, then the rewards defined above are multiplied by -10. See Table 1 for the full transition and reward structure.
- One episode lasts 5 time steps (for a total of 5 actions) and always starts in state 0 (no rewards at the initial state).

An example of a trajectory (or episode) in the test environment is shown in Figure 1, and the trajectory can be represented in terms of s_t, a_t, R_t as:
$s_0 = 0, a_0 = 1, R_0 = -0.3, s_1 = 1, a_1 = 2, R_1 = 0.0, s_2 = 2, a_2 = 4, R_2 = 0.0, s_3 = 2, a_3 = 3, R_3 = 2.0, s_4 = 3, a_4 = 0, R_4 = 0.1, s_5 =$
.

State (s)	Action (a)	Next State (s')	Reward (R)
0	0	0	0.1
0	1	1	-0.3
0	2	2	0.0
0	3	3	-0.2
0	4	0	0.1
1	0	0	0.1
1	1	1	-0.3
1	2	2	0.0
1	3	3	-0.2
1	4	1	-0.3
2	0	0	-1.0
2	1	1	3.0
2	2	2	0.0
2	3	3	2.0
2	4	2	0.0
3	0	0	0.1
3	1	1	-0.3
3	2	2	0.0
3	3	3	-0.2
3	3	3	-0.2

Table 1: Transition table for the Test Environment

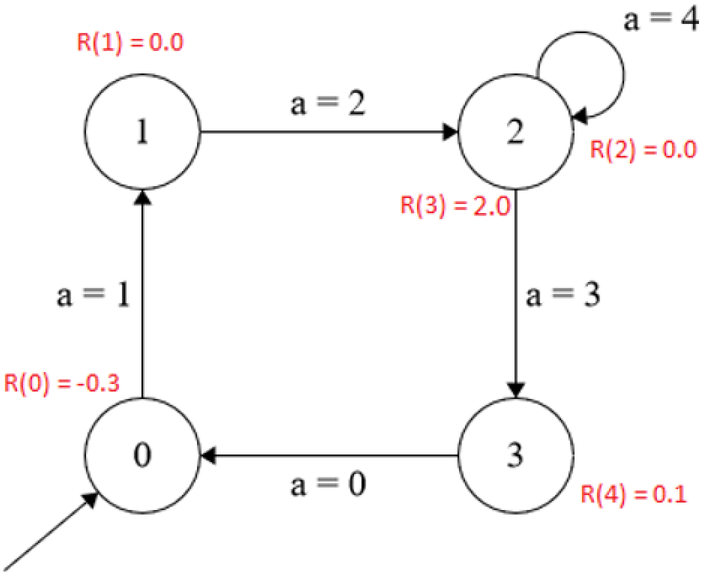

Figure 1: Example of a trajectory in the Test Environment

1. **(written)** What is the maximum sum of rewards that can be achieved in a single trajectory in the test environment, assuming γ = 1? Show first that this value is attainable in a single trajectory, and then briefly argue why no other trajectory can achieve greater cumulative reward.

Task 3 Tabular Q-Learning

If the state and action spaces are sufficiently small, we can simply maintain a table containing the value of $Q(s,a)$, an estimate of $Q^*(s,a)$, for every (s,a) pair. In this *tabular setting*, given an experience sample (s,a,r,s'), the update rule is

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \lambda max_{a'} Q(s',a') - Q(s,a))$$

where $\alpha > 0$ is the learning rate, $\lambda \in [0,1)$ the discount factor.

$\epsilon - \mathbf{greedy}$ **Exploration Strategy** For exploration, we use an $\epsilon - greedy$ strategy. This means that with probability ϵ, an action is chosen uniformly at random from $\mathcal{A}$, and with probability $1-\epsilon$, the greedy action $(i.e., argmax_{a \in \mathcal{A}} Q(s,a))$ is chosen.

1. **(coding)** Implement the `get_action` and `update` functions in `q_table.py`. Test your implementation by running `python q_table.py`.

Task 4 Maze Example

You will implement Sarsa and Q-learning for the Maze environment from `OpenAI Gym`. We have provided custom versions of this environment. In the scenario, a red rectangle (agent) are initialized at the maze made up of 4×4 grids and can only observe its location. At each timestep, agent can move to one of four neighboring grids. The reward is +1 if agent is located at the yellow grid, -1 if agent reaches the black grid, otherwise 0.

1. **(coding)** Implement **Sarsa** in `RL_sarsa.py`.
2. **(coding)** Implement **Q_learning** in `RL_q_learning.py`.

Submission

Submit a zip fie that includes your source code and PDF of report to the email(zhangyc8@mail2.sysu.edu.cn)

Naming format of zip file: RL_Student ID_Name_Assignment ID, e.g.: RL_20220315_张三_homework1

The environment code can be available at https://github.com/ZYC9894/SYSU_RL2022