AE 598: Reinforcement Learning HW2

Inyoung Kim
Aerospace Engineering
University of Illinois Urbana-Champaign
IL, USA
inyoung3@illinois.edu

***Abstract*—In this report, we implement three model-free RL algorithms — On-policy First-Visit Monte Carlo (MC) Control, SARSA (On-policy TD(0)), and Q-learning (Off-policy TD(0)) — in a tabular setting using the Frozen Lake MDP environment from Gymnasium.**

I. Introduction

In this task, we apply three model-free RL algorithms to the Frozen Lake environment. The problem is fully observable, has an infinite time horizon, and the discount rate is set to $\gamma = 0.95$. The goal is to compare performance by applying On-policy First-Visit Monte Carlo (MC) Control, SARSA (On-policy TD(0)), and Q-learning (Off-policy TD(0)). Furthermore, we consider both cases where the `is_slippery` option, which determines the transition probability of the environment, is set to `True` and `False`.

A. Frozen Lake MDP

A Markov Decision Process (MDP) is defined as a 5-tuple

$$\mathcal{M} = (S, A, P, R, \gamma),$$

S is a finite set of states, A is a finite set of actions, $P(s' \mid s, a)$ is the state transition probability, $R(s, a, s')$ is the reward function and $\gamma \in [0, 1]$ is the discount factor.
We define the MDP as follows. The environment is given by a 4×4 grid:

$$\begin{bmatrix} S & F & F & F \\ F & H & F & H \\ F & F & F & H \\ H & F & F & G \end{bmatrix}$$

Here, S is the start tile, F is a frozen tile, H is a tile with a hole, and G is the goal tile. Therefore, the state space $\mathcal{S}$ contains 16 states, corresponding to each cell in the grid. The holes and the goal are the terminal states. The action space $\mathcal{A}$ has four actions: left(0), down(1), right(2), up(3).
Next, for transition probabilities, if `is_slippery=False`, the agent moves in the intended direction. If `is_slippery=True`, The agent moves in the intended direction with probability $\frac{1}{3}$ and in both directions perpendicular to the intended direction with probability $\frac{1}{3}$.

The reward function $R(s, a, s')$ is defined as

$$R(s, a, s') = \begin{cases} 1, & \text{if } s' \text{ is the goal,} \\ 0, & \text{otherwise.} \end{cases}$$

when transitioning from state s to s' by taking action a.

B. Model-Free Reinforcement Learning

The three algorithms implemented in this project are all model-free reinforcement learning methods, which learn through interaction with the environment without knowledge of the transition probabilities or reward distributions. In contrast, the dynamic programming approach used in the previous assignment is a model-based learning method that directly utilizes the transition probabilities and rewards, whereas in this work, the algorithms are implemented using only sampled experiences.

C. On-policy learning and Off-policy learning

On-policy learning refers to the case where the target policy and the behavior policy are identical, while off-policy learning refers to the case where the target policy and the behavior policy are different.

D. Monte Carlo VS Temporal Difference

Monte Carlo (MC) methods are learned after an episode is completed, whereas Temporal-Difference (TD) methods are updated immediately at each step. In other words, MC does not use bootstrapping, while TD employs bootstrapping to adjust the current state's estimate based on the estimated value of the next state.

E. ϵ-greedy policy

The ϵ-greedy policy is a method for balancing exploitation and exploration, in which the agent selects the best (greedy) action with probability $1 - \epsilon$, and randomly chooses an action for exploration with probability ϵ.

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A(s)|}, & \text{if } a = \arg\max\limits_{a'} Q(s, a') \\ \dfrac{\epsilon}{|A(s)|}, & \text{otherwise} \end{cases}$$

where $|A(s)|$ denotes the number of available actions in state s.

II. Methods

In this section, we describe the three implemented algorithms. On-policy First-Visit Monte Carlo (MC) Control

A. On-policy First-Visit Monte Carlo (MC) Control

On-policy First-Visit Monte Carlo Control is an algorithm that approximates the optimal policy from the agent's experiences by updating the Q-values based on the returns of sampled episodes. In other words, it estimates $Q(s, a)$ through experience and improves the policy π in an ϵ-greedy manner, leading to convergence toward the optimal policy π^*.

Algorithm 1 On-policy First-Visit Monte Carlo Control (for ϵ-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\epsilon > 0$
Initialize:
$\pi \leftarrow$ an arbitrary ϵ-soft policy
$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
repeat: for each episode
Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
for $t = T-1, T-2, \ldots, 0$ **do**
$G \leftarrow \gamma G + R_{t+1}$
if the pair (S_t, A_t) does not appear in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$ **then**
Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
$A^* \leftarrow \arg\max_a Q(S_t, a)$: ties broken arbitrarily
for all $a \in \mathcal{A}(S_t)$ **do**

$$\pi(a \mid S_t) \leftarrow \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|\mathcal{A}(S_t)|}, & \text{if } a = A^* \\ \dfrac{\epsilon}{|\mathcal{A}(S_t)|}, & \text{if } a \neq A^* \end{cases}$$

end for
end if
end for
until converge

B. SARSA (On-policy TD(0))

SARSA is an on-policy TD(0) algorithm that, as its name implies, learns $Q_\pi(s, a)$ under the current policy π using the experienced state, action, reward, next state, and next action. Through this process, the current policy gradually converges toward the optimal policy.

Algorithm 2 SARSA (On-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize: $Q(s, a)$ for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$
repeat: for each episode
Initialize S
Choose A from S using policy derived from Q (e.g., ϵ-greedy)
repeat: for each step of episode
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
$Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
$S \leftarrow S'$
$A \leftarrow A'$
until S is terminal
until converge

C. Q-learning (Off-policy TD(0))

Q-learning is an Off-policy TD control algorithm that selects actions based on the ϵ-greedy policy. Unlike SARSA, however, Q-learning is off-policy and thus updates the Q-function greedily, directly learning the optimal action-value function Q^*.

Algorithm 3 Q-learning (Off-policy TD control) for estimating $\pi \approx \pi_*$

Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize: $Q(s, a)$ arbitrarily for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, except that $Q(\text{terminal}, \cdot) = 0$
repeat
Initialize S
repeat
Choose A from S using policy derived from Q (e.g., ϵ-greedy)
Take action A, observe R, S'
$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
$S \leftarrow S'$
until S is terminal
until convergence

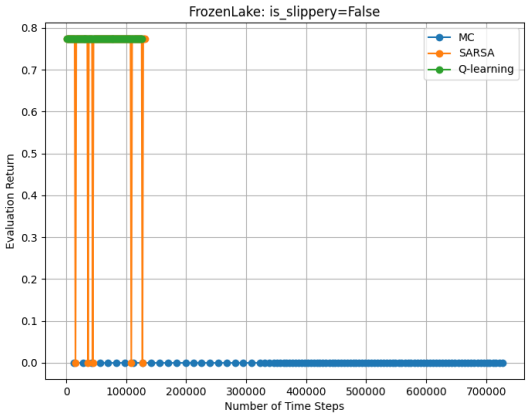
III. Result

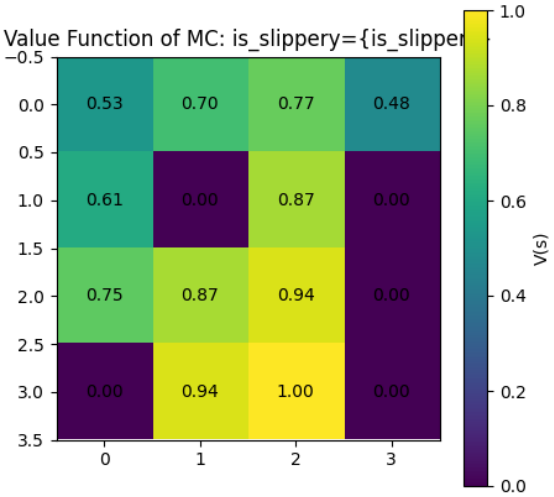
In this experiment, we evaluated a total of six scenarios, combining three algorithms (MC, SARSA, and Q-learning) with two environment settings: *is_slippery* = *False* and *is_slippery* = *True*. Figures III and III show the plots of the evaluation return versus the number of time steps, where the number of time steps refers to the *cumulative training time steps*.

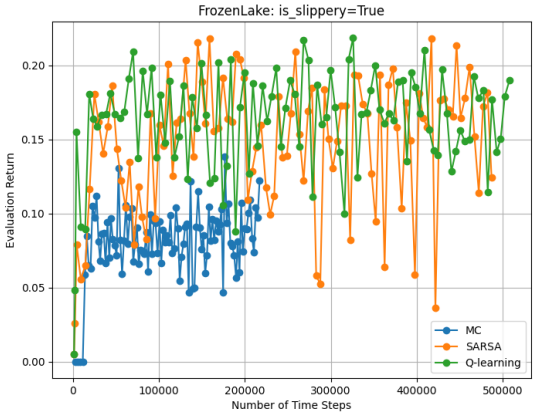
As shown in Figure III, when *is_slippery* = *False*, both Q-learning and SARSA converge to the optimal return value, while MC fails to reach the optimal return. This occurs because MC uses a *first-visit Monte Carlo* approach, where

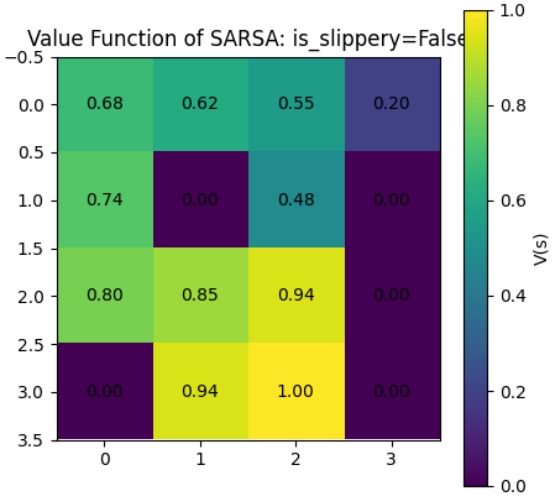
rewards are updated only after each episode. As a result, most rewards remain close to zero, leading to limited learning progress and insufficiently informative updates.

Similarly, in Figure III, when *is_slippery* = *True*, Q-learning and SARSA again show convergence toward the optimal return value, whereas MC still does not reach the optimal value for the same reason described above.

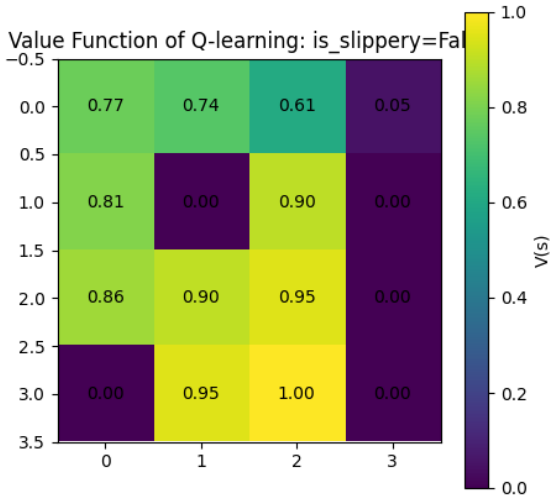


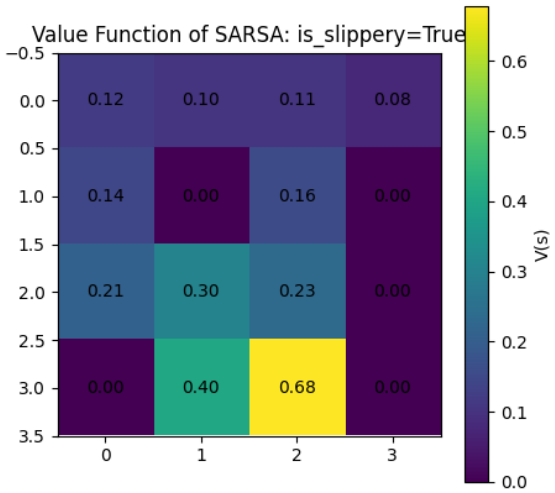


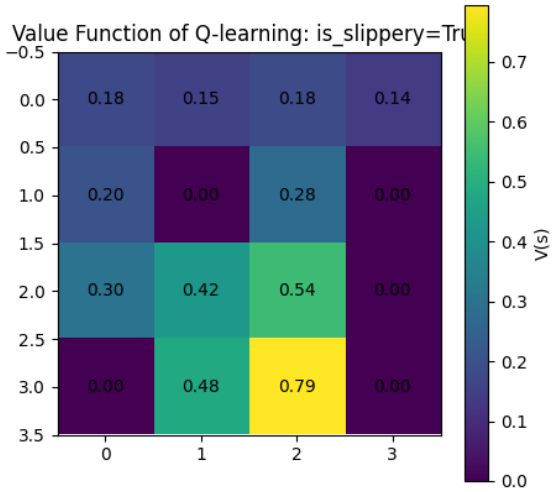


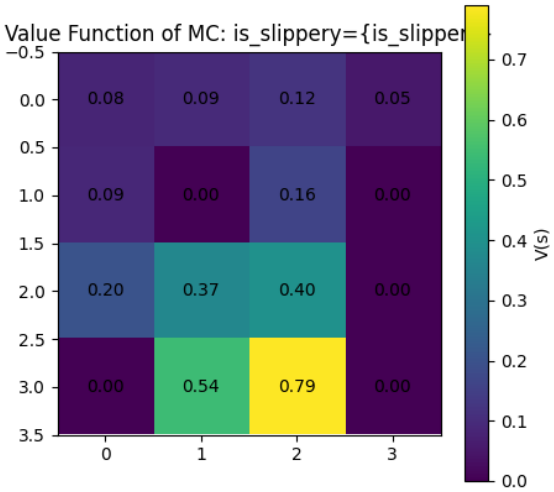


The following figures show the value functions corresponding to each trained agent.









IV. Conclusions

In this report, three model-free reinforcement learning algorithms were implemented and compared in the *FrozenLake* environment. The experimental results show that the TD-based methods, SARSA and Q-learning, exhibited faster convergence since they update values at every timestep, whereas the first-visit Monte Carlo (MC) method, which updates only after each episode, failed to reach the optimal policy.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.