Tabular Methods for RL

Ganesh Ravisankar
Department of Aerospace Engineering)

University of Illinois, Urbana-Champaign
Champaign, Illinois
ganeshr3@illinois.edu

I. Introduction

In this document, we report the findings of Homework 2, where we compare three different model-free RL algorithms in a tabular setting: on-policy first visit MC control, SARSA, and Q-learning. We use these three frameworks to approximate the optimal value function and policy for Gymnasium's "FrozenLake-v1" environment, which is a finite-dimensional, discrete MDP.

The remainder of the document is organized as follows: Section II covers a precise problem statement in terms of a Markov Decision Process (MDP); Section III describes the approaches taken to solve the problem and the algorithms; Section IV discusses the results and presents relevant graphs; conclude the report in Section V.

II. Problem Statement

We consider the tabular FrozenLake problem as a discounted, infinite-horizon Markov Decision Process (MDP):

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \rho_0 \rangle, \qquad \gamma = 0.95.$$

a) Grid and indexing: The environment is a 4×4 grid

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

where S is the start cell, G is the goal, H denotes holes, and F denotes frozen floor.

The start distribution is deterministic at the upper–left cell:

$$\rho_0(s) = \mathbb{I}[s = 0].$$

The terminal condition is $\mathcal{T} = \{s : \text{cell}(s) \in \{H, G\}\}$.

b) State and action spaces: The state space is finite with $|\mathcal{S}| = 16$. The action space is $\mathcal{A} = \{0, 1, 2, 3\}$, corresponding to Left (0), Down (1), Right (2), and Up (3).

c) Transition Probability: For each nonterminal state s and action a, the transition kernel is given by:

$$P[s][a] \equiv \{(p, s', r, \texttt{done})\},$$

where $p = \Pr(s'|s, a)$, $r = R(s, a, s')$, and $\texttt{done} = \mathbb{I}[s' \in \mathcal{T}]$.

Further, we consider two different types of dynamics:

- *Deterministic (non-slippery)*: the next state is the result of applying action a once. Thus $P(\cdot \mid s, a)$ has a single deterministic outcome.
- *Stochastic (slippery)*: the executed action is a random turn of the intended action:

  $$a_{\text{exec}} \in \{(a{-}1) \bmod 4,\ a,\ (a{+}1) \bmod 4\} \quad \text{with} \quad \Pr = \tfrac{1}{3} \text{ each},$$

  followed by a one-step move.

All terminal states are absorbing, that is, for any $s \in \mathcal{T}$ and any a,

$$P(s' = s \mid s, a) = 1, \qquad \texttt{done} = \text{True}.$$

d) Reward: Rewards depend on the next state:

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

Entering a hole yields 0 and terminates. Once in the terminal state, subsequent self-loops yield 0.

e) Objective: We seek to find a stationary policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the discounted return:

$$\max_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right].$$

In the next section, we discuss the approach to solve the problem.

III. Methods

Our main objective is to obtain the optimal value function $V^*(\cdot)$ and the optimal policy $\pi^*(\cdot|\cdot)$ using three distinct model-free approaches: on-policy MC control, SARSA, and Q-learning. Previously, we were able to use dynamic programming, a model-based approach, because we made use of the state transition function, P, of the MDP. While the MDP defines the state transition function, in practice, it is very difficult to define the state transition function, particularly in continuous state and action spaces and when the number of states and actions is very large. Model-free methods, such as those mentioned above, circumnavigate these issues by directly estimating the optimal value function and policy through a data-driven approach. In the following discussions, we outline each of the three model-free approaches and provide pseudocode for the implementation.

A. On-Policy Monte Carlo Control

The pseudocode for on-policy MC control is given in Algorithm 1 [1]. On-policy first-visit MC control learns an action-value function Q and improves a stochastic behavior policy π using episodes generated by that same policy. For each episode, and for each state-action pair (s_t, a_t) the first time it appears in the episode, we compute the return

$$G_t \;=\; \sum_{k=0}^{T-t-1} \gamma^k R_{t+1+k},$$

and update Q toward this target. After the episode, we imrove π to be ε-greedy (i.e., ε-soft) with respect to the current Q:

$$\pi(a \mid s) = \begin{cases} 1-\varepsilon+\dfrac{\varepsilon}{|\mathcal{A}(s)|}, & a \in \arg\max_{a'} Q(s,a'), \\ \dfrac{\varepsilon}{|\mathcal{A}(s)|}, & \text{otherwise.} \end{cases}$$

Using a diminishing exploration schedule ensures every (s,a) is visited infinitely often; then in finite tabular MDPs, on-policy first-visit MC control converges to Q and a greedy optimal policy π.

Algorithm 1: On-Policy First-Visit MC Control (ε-greedy)

```
   Input: Discount $\gamma$, episodes $K$, schedule $\{\varepsilon_k\}$
1  Initialize $Q(s,a)$ arbitrarily; $Returns(s,a) \leftarrow 0$;
     initialize $\pi$ to be $\varepsilon_1$-soft;
2  for $k = 1$ to $K$ do
3      Generate episode $(s_0, a_0, r_1, \ldots, s_T)$ by following
         $\pi$;
4      $G \leftarrow 0$; for $t = T-1$ to $0$ do
5          $G \leftarrow \gamma G + R_{t+1}$;
6          if $(S_t, A_t)$ does not appear in
             $(S_0,A_0),(S_1,A_1),\ldots,(S_{t-1},A_{t-1})$ then
7              Append $G$ to Returns$(S_t,A_t)$;
                 $Q(S_t,A_t) \leftarrow$ average(Returns$(S_t,A_t)$);
                 $A^* \leftarrow \arg\max_a Q(S_t,a)$ forall $a \in \mathcal{A}(S_t)$
                 do
8                    $\pi(a \mid S_t) \leftarrow \begin{cases} 1-\varepsilon+\dfrac{\varepsilon}{|\mathcal{A}(S_t)|}, & a = A^*, \\ \dfrac{\varepsilon}{|\mathcal{A}(S_t)|}, & a \neq A^* \end{cases}$
```

B. SARSA

The pseudocode for SARSA is given in Algorithm 2 [1]. The SARSA algorithm is an on-policy temporal-difference control method that updates the action-value function $Q(s,a)$ based on the current state–action pair, the reward received, and the next state–action pair sampled from the same behavior policy. Unlike Monte Carlo methods, SARSA is more bootstrapping, as it updates Q after each step rather than waiting until the end of the episode.

At each time step t, the agent selects an action A_t according to an ε-greedy policy derived from Q, observes the reward R_{t+1} and next state S_{t+1}, selects the next action A_{t+1} from the same ε-greedy policy, and updates:

$$Q(S_t,A_t) \;\leftarrow\; Q(S_t,A_t) + \alpha\big[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)\big].$$

The process continues until a terminal state is reached, after which the next episode begins. Over time, with sufficient exploration, Q converges to the optimal action-value function Q^*, and the greedy policy converges to the optimal policy π^*.

Algorithm 2: SARSA (On-Policy TD(0)) Control (ε-greedy)

```
    Input: Step size $\alpha \in (0,1]$, discount $\gamma$, small $\varepsilon > 0$
1   Initialize $Q(s,a)$ arbitrarily for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$,
      except that $Q(\text{terminal},\cdot) = 0$;
2   repeat
3       Initialize $S$;
4       Choose $A$ from $S$ using policy derived from $Q$
          (e.g., $\varepsilon$-greedy);
5       repeat
6           Take action $A$, observe $R$ and $S'$;
7           Choose $A'$ from $S'$ using policy derived from
              $Q$ (e.g., $\varepsilon$-greedy);
8             $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
            $S \leftarrow S'$;    $A \leftarrow A'$;
9       until for each step of episode;
        $S$ is terminal
10  until for each episode;
```

C. Q-Learning

The pseudocode for Q-Learning is given in Algorithm 3 [1]. Q-learning is an off-policy temporal-difference (TD) control algorithm that learns the optimal action-value function Q^* independent of the agent's behavior policy. It differs from SARSA by updating Q toward the value of the greedy action at the next state rather than the action actually taken.

At each step, the agent selects an action A_t using an ε-greedy policy derived from Q, observes reward R_{t+1} and next state S_{t+1}, and updates its estimate:

$$Q(S_t,A_t) \;\leftarrow\; Q(S_t,A_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)\big].$$

Here, the target uses $\max_a Q(S_{t+1},a)$, representing the greedy action, while the behavior policy may still be ε-greedy for exploration. With sufficient exploration (e.g., a GLIE schedule) and appropriate learning rate decay, Q-learning converges to the optimal Q^* and the corresponding optimal policy π^*.

Algorithm 3: Q-Learning (Off-Policy TD(0)) Control (ε-greedy)

```
   Input: Step size $\alpha \in (0, 1]$, discount $\gamma$, small $\varepsilon > 0$
1  Initialize $Q(s, a)$ arbitrarily for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$,
     except that $Q(\text{terminal}, \cdot) = 0$;
2  repeat
3      Initialize $S$;
4      repeat
5          Choose $A$ from $S$ using policy derived from $Q$
             (e.g., $\varepsilon$-greedy);
6          Take action $A$, observe $R$ and $S'$;
7
             $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
           $S \leftarrow S'$;
8      until for each step of episode;
         $S$ is terminal
9  until for each episode;
```

IV. Results and Discussion

In this section, we present the results and discuss them. In particular, we examine the convergence of each approach to reach the optimal return value as well as the number of time steps to reach it in both stochastic and deterministic conditions. Additionally, we also present grid maps for the values of the value function and policy preference in each grid.

a) Evaluation Setup.: We evaluated on-policy MC control, SARSA, and off-policy Q-learning on the 4×4 Frozen Lake environment under two transition models: the stochastic $(\mathtt{is}_slippery = True) and deterministic($

b) Slippery environment (stochastic transitions).: As shown in Figure 1, all three algorithms gradually improve their evaluation returns over time despite the inherent stochasticity of state transitions. Q-learning (green) demonstrates faster learning and higher asymptotic performance than SARSA (orange) and Monte Carlo control (blue). Because Q-learning is off-policy, it can more aggressively exploit optimal actions via the $\max_a Q(s', a)$ target, leading to higher variance but faster convergence toward the optimal value. SARSA, being on-policy, updates using the next action actually taken under the ε-greedy policy; this makes it more conservative, yielding smoother but slower learning. Monte Carlo control exhibits the slowest early learning due to its reliance on complete episodes before updates, though it eventually approaches near-optimal returns. Overall, Q-learning achieves the best balance between sample efficiency and asymptotic performance under stochastic dynamics.

c) Deterministic environment (non-slippery).: In the deterministic case (Figure 2), all algorithms converge much faster and achieve returns close to the optimal policy obtained by Value Iteration. Q-learning again shows the most rapid rise to optimality, reaching the plateau almost immediately. SARSA and Monte Carlo control follow closely, with SARSA exhibiting slightly smoother convergence due to on-policy consistency. Because transitions are deterministic, the variance of returns is low, allowing all methods to eventually match the optimal expected return. This highlights that the primary difference among the methods lies in their sample efficiency and sensitivity to stochasticity rather than their final achievable performance.

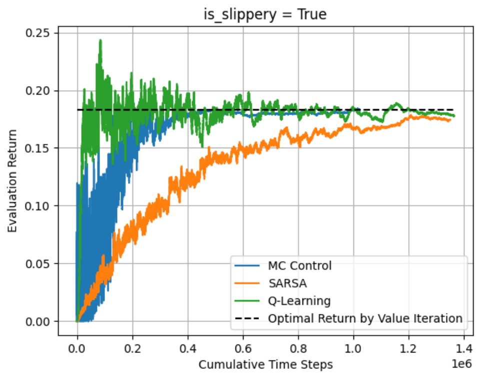

Fig. 1. Evaluation return vs. cumulative time steps for `is_slippery=True`.

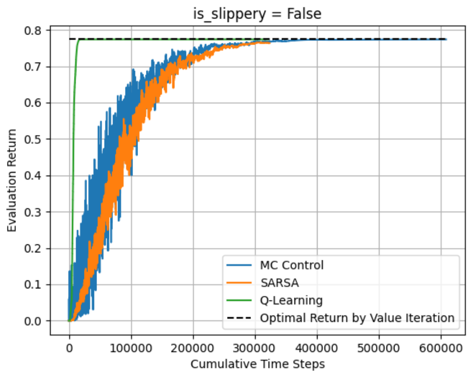

Fig. 2. Evaluation return vs. cumulative time steps for `is_slippery=False`.

d) Summary.: In summary:

- **Q-learning** is the fastest and most sample-efficient but exhibits higher variance early on.
- **SARSA** is more stable but slower to reach optimal returns, particularly in noisy environments.
- **Monte Carlo Control** learns purely from episodic experience and thus converges slowly but reliably.

These observations align with theoretical expectations, confirming that bootstrapped TD methods (SARSA, Q-learning) achieve faster convergence than Monte Carlo methods, and that off-policy learning can outperform on-policy methods under sufficient exploration.

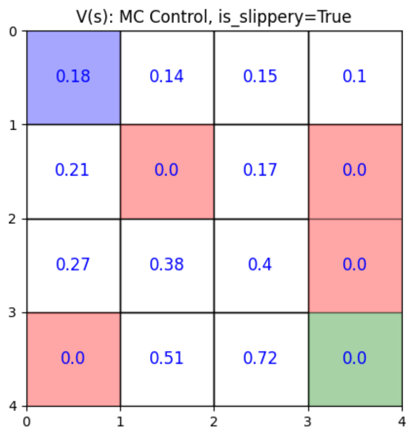

Fig. 3. Monte Carlo Grid Value Function: Slippery Condition

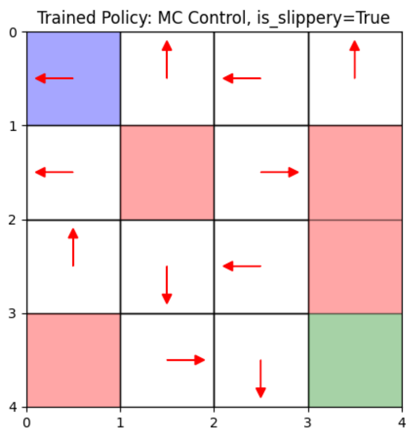

Fig. 4. Monte Carlo Grid Policy: Slippery Condition

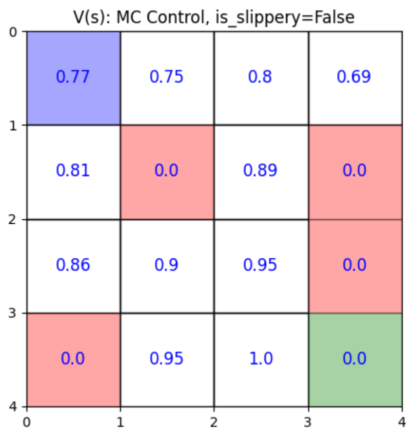

Fig. 5. Monte Carlo Grid Value Function: Deterministic Condition

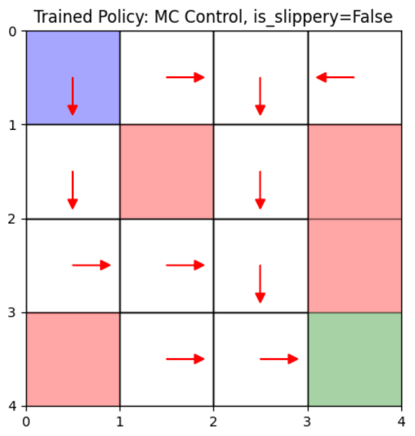

Fig. 6. Monte Carlo Grid Policy: Deterministic Condition

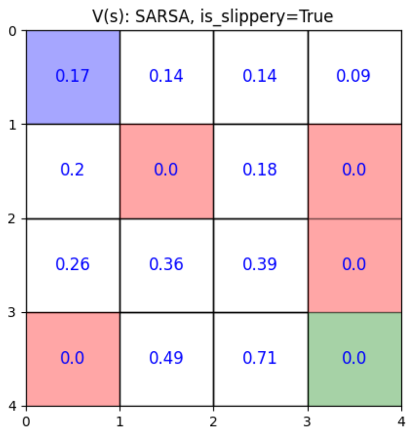

Fig. 7. SARSA Grid Value Function: Slippery Condition

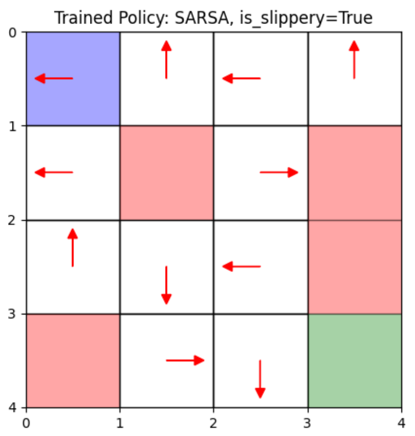

Fig. 8. SARSA Grid Policy: Slippery Condition

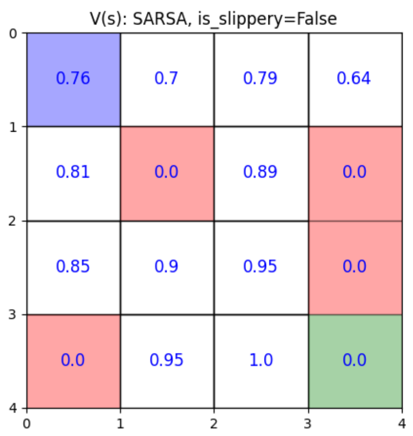

Fig. 9. SARSA Grid Value Function: Deterministic Condition

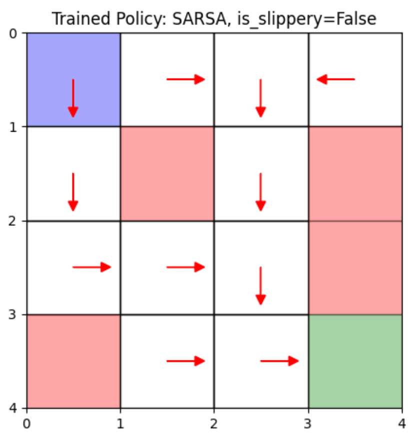

Fig. 10. SARSA Grid Policy: Deterministic Condition

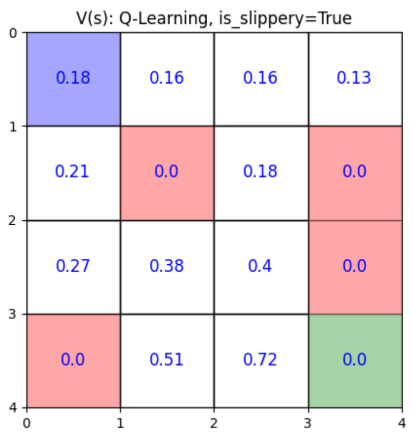

Fig. 11. Q-Learning Grid Value Function: Slippery Condition

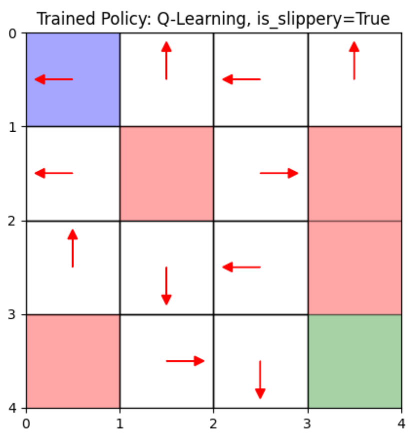

Fig. 12. Q-Learning Grid Policy: Slippery Condition

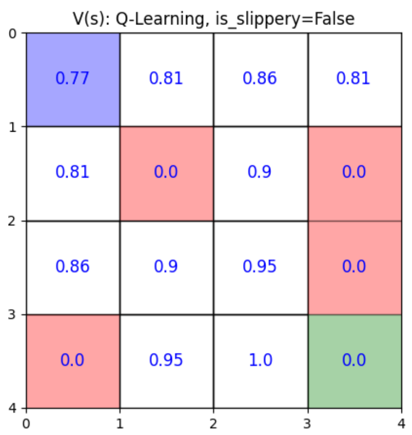

Fig. 13. Q-Learning Grid Value Function: Deterministic Condition

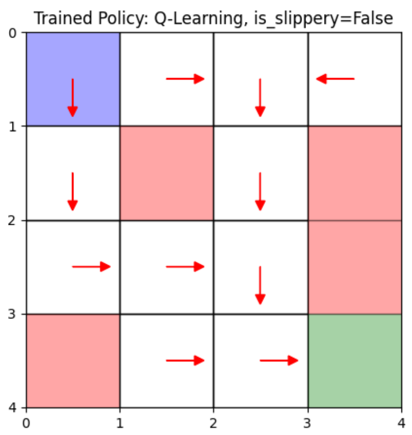

Fig. 14. Q-Learning Grid Policy: Deterministic Condition

V. Conclusion

This work compared three tabular, model-free control methods—on-policy first-visit Monte Carlo (MC), SARSA, and Q-learning—on the 4×4 FrozenLake MDP under both stochastic (`is_slippery=True`) and deterministic (`is_slippery=False`) dynamics. Using Value Iteration as an oracle baseline, we measured evaluation return versus cumulative interaction steps.

The key findings were as follows:

- **Sample efficiency.** Q-learning was consistently the most sample-efficient, reaching near-optimal performance in both settings and doing so faster in the deterministic case.
- **Stability vs. speed.** SARSA exhibited smoother (more stable) learning but converged more slowly, especially under stochastic transitions, reflecting its on-policy bootstrap target.
- **Episodic updates.** MC control improved reliably but was slowest early on due to episode-level targets; nevertheless it approached the optimal return with sufficient data.
- **Effect of dynamics.** All methods converged faster and closer to the VI optimum in the deterministic environment, highlighting the impact of transition stochasticity on variance and learning speed.

For small, discrete MDPs, **Q-learning** offers the best trade-off between speed and final performance; **SARSA** is preferable when stability under exploration noise is desired; and **MC control** provides a simple, model-free baseline that converges with enough experience.

References

[1] Richard S. Sutton and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*, MIT Press.

[2] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, MIT Press.