AE 598 Reinforcement Learning: Homework 2

Ting-Wei Hsu (twhsu3)

I. Introduction

In this homework, we will implement three *model-free* learning algorithms, namely 1) on-policy first-visit Monte Carlo control, 2) SARSA, and 3) Q-learning, to approximate the optimal value function and policy for the Frozen Lake MDP.

A. Frozen Lake MDP

The Frozen Lake MDP is a discounted infinite-horizon MDP with a discount factor of $\gamma = 0.95$. The map of the frozen lake is a 4-by-4 grid, where each grid represents a state, denoted by $s \in \mathcal{S}$ (16 discrete states in total). $[0, 0]$ is the start state, $[3, 3]$ is the goal state, and there are some holes in between on the map, as shown in Fig. 1. The action,

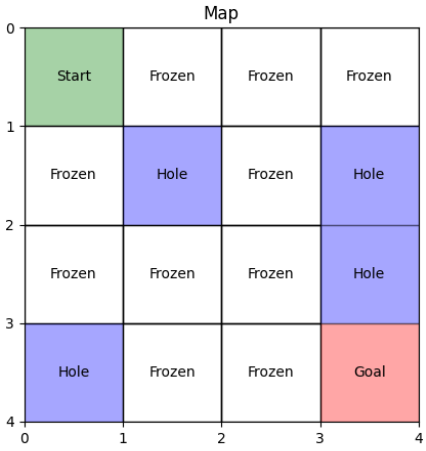


Fig. 1: Frozen lake map.

denoted by $a \in \mathcal{A}$, indicates the intended direction the player is moving, given by $a = 0$ (move left), 1 (down), 2 (right), 3 (up). The reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{>0}$ is a function of state and action. $r(s, a) = 1$ when the player reaches the goal by taking action a at state s; $r(s, a) = 0$ otherwise. The state transition is deterministic if `is_slippery=True`. However, if `is_slippery=False`, the player will move in the intended direction with probability of 1/3 and move in either perpendicular direction with equal probability of 1/3. The transition probability distribution is denoted by $P(s'|s, a)$. When the player is in an edge state and the transition attempts to move the player outside the map, the player will stay in the current state. The episode terminates whenever the player reaches the goal or a hole.

B. Dynamic Programming and Bellman Equations

The state-value function of a given policy, $\pi(s) : \mathcal{S} \to \mathcal{A}$ is given by

$$V^\pi(s) := \mathbb{E}_{\pi, P(\cdot|s,a)}\Big[\sum_{t=1}^{\infty} \gamma_{t-1} r_t \Big| s_1 = s\Big] \tag{1}$$

Similarly, we can define the state-action value function:

$$Q^\pi(s, a) := \mathbb{E}_{\pi, P(\cdot|s,a)}\Big[\sum_{t=1}^{\infty} \gamma_{t-1} r_t \Big| s_1 = s, a_1 = a, a_{2:\infty} \sim \pi\Big]. \tag{2}$$

1) Bellman Equations for Policy Evaluation: Rewriting (1) and (2), we have the Bellman equations for policy evaluations as follows:

$$\begin{aligned} V^\pi(s) &:= \mathbb{E}_{a\sim\pi}[r(s, a)] + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')] \\ &:= \mathcal{T}^\pi V^\pi(s) \end{aligned} \tag{3}$$

and

$$\begin{aligned} Q^\pi(s, a) &:= r(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')] \\ &:= \mathcal{T}^\pi Q^\pi(s, a), \end{aligned} \tag{4}$$

where $\mathcal{T}^\pi(\cdot)$ denotes the Bellman operator for policy evaluation.

2) Bellman Optimality Equations: There always exists a deterministic policy that maximizes $V^\pi(s)$ for all $s \in \mathcal{S}$ and $Q^\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Such a policy is called the optimal policy, denoted by $\pi^\star$, and the corresponding maximized value functions are denoted by $V^\star(s)$ and $Q^\star(s, a)$. The Bellman equations for optimality are given by

$$V^\star(s) = \max_{a\in\mathcal{A}} Q^\star(s, a) \tag{5}$$

and

$$Q^\star(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\star(s')]. \tag{6}$$

Therefore, we have

$$\begin{aligned} V^\star(s) &= \max_{a\in\mathcal{A}} \Big[r(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\star(s')]\Big] \\ &:= \mathcal{T}^\star V^\star(s) \end{aligned} \tag{7}$$

and

$$\begin{aligned} Q^\star(s, a) &= r(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[\max_{a\in\mathcal{A}} Q^\star(s', a)] \\ &:= \mathcal{T}^\star Q^\star(s, a), \end{aligned} \tag{8}$$

where $\mathcal{T}^\star(\cdot)$ denotes the Bellman operator for optimality. Once we have $Q^\star(s, a)$, we can ontain the optimal policy as

$$\pi^\star(s) = \arg\max_{a\in\mathcal{A}} Q^\star(s, a) \quad \forall s \in \mathcal{S} \tag{9}$$

II. METHODS

Our goal is to obtain the optimal value function, $V^\star(s)$ or $Q^\star(s,a)$, and the optimal policy, $\pi^\star(s)$ for all $s \in \mathcal{S}$. In the previous homework, since we assumed the state transition model (i.e., $P(s'|s,a)$) was known, we utilized *model-based* methods such as policy iteration and value iteration to solve the dynamic programming problem directly. In practice, however, the state transition model of an MDP is often unknown, so we can only estimate the value function and optimal policy using *model-free* methods, i.e., learning from data. In this homework, we implement three common tabular learning algorithms: 1) on-policy first-visit Monte Carlo control, 2) SARSA (on-policy TD learning), and 3) Q-learning (off-policy TD learning). In the following subsections, we will provide the implementation details of each algorithm using pseudo-code.

A. On-policy First-Visit MC Control

The pseudo-code for the on-policy first visit Monte Carlo control is given in Algorithm 1 [1]. In each episode, we generate a trajectory using an ϵ_k-greedy policy, given by

$$\pi(a|s_t) \leftarrow \begin{cases} 1-\epsilon_k+\dfrac{\epsilon_k}{|\mathcal{A}(s_t)|}, & \text{if } a = \arg\max_a Q(s_t,a) \\ \dfrac{\epsilon_k}{|\mathcal{A}(s_t)|}, & \text{if otherwise} \end{cases}, \tag{10}$$

where ϵ_k denotes the ϵ value for the k-th episode. The purpose of using the ϵ-greedy policy is to ensure that every state-action pair is visited infinitely often. Further, to ensure *greedy in the limit with infinite exploration (GLIE)*, we schedule the ϵ value such that $\epsilon_k \to 0$ as $k \to \infty$. For example, we implement

$$\epsilon_k = \epsilon_0 \cdot (0.9999)^{k-1}. \tag{11}$$

In doing so, the MC control is guaranteed to converge to the true optimal policy. At each time step t in the episode, we update the state-action value function $Q(s_t,a_t)$ if the state-action pair (s_t,a_t) is the first visit in the episode. Every time $Q(s_t,a_t)$ is updated, its ϵ-greedy policy is also updated. The next episode will be generated by the updated ϵ-greedy policy, and so on.

To improve learning stability, we also schedule the learning rate α to decay over episodes. For instance, we implement

$$\alpha_k = \begin{cases} \alpha_0, & \text{if } k < 5000, \\ \max\left(\alpha_0 \cdot 0.9999^{(k-5000)},\ \alpha_{\min}\right), & \text{if } k \geq 5000, \end{cases} \tag{12}$$

where α_k denotes the learning rate for the k-th episode, α_0 denotes the initial learning rate, and $\alpha_{\min}$ denotes the minimum learning rate. We set $\alpha_{\min} = 1e^{-4}$ for the MC control algorithm.

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

The pseudo-code for SARSA is given in Algorithm 2 [1]. Different from MC control, which requires running an entire episode before updating the policy, SARSA utilizes TD(0) learning to update the state-action value function $Q(s_t,a_t)$ at every time step, as can be seen on Line 9 in Algorithm 2. Also, SARSA is an on-policy method because the action generated by the policy is actually used to propagate the state. To guarantee GLIE, the ϵ_k-greedy policy in (10) with ϵ_k scheduled by (11) is implemented. To enhance learning stability, the learning rate is scheduled to decay by (12) with $\alpha_{\min} = 1e^{-3}$.

Algorithm 1: On-policy First-Visit MC Control

Input: MDP discount factor $\gamma = 0.95$, initial learning rate $\alpha_0 = 0.1$, initial $\epsilon_0 = 1$, number of episodes $N_e = 100000$

Output: Optimal action-value function $Q^\star$ and policy $\pi^\star$

```
Initialize Q(s,a) ← 0 for all (s,a)
for episode k = 1, 2, ..., N_e do
    Schedule ε_k by (11) to impose GLIE
    Schedule α_k by (12) to enhance learning stability
    Generate an episode {(s_0,a_0,r_0), ..., (s_T,a_T,r_T)} following the
    ε_k-greedy policy π(a|s) w.r.t. current Q
    G ← 0
    for t = T−1, T−2, ..., 0 do
        G ← γG + r_t
        if (s_t,a_t) is the first visit in the episode then
            Q(s_t,a_t) ← Q(s_t,a_t) + α_k[G − Q(s_t,a_t)]
π*(s) ← arg max_a Q(s,a),   Q* ← Q
return Q*, π*
```

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

The pseudo-code for Q-learning is given in Algorithm 3 [1]. Q-learning is similar to SARSA as they are both TD(0) control. The main difference lies in its update law, given by

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right],$$

which is implemented on Line 8 in Algorithm 3. Q-learning is an *off-policy* method due to the $\max$ operator in the update law, since the maximizing action is not actually used to propagate the state transition. To guarantee GLIE, the ϵ_k-greedy policy in (10) with ϵ_k scheduled by (11) is implemented. To enhance learning stability, the learning rate is scheduled to decay by (12) with $\alpha_{\min} = 1e^{-3}$.

III. RESULTS AND DISCUSSIONS

In this section, we demonstrate the planning results for the Frozen Lake MDP using the three learning algorithms mentioned in the previous section. For each algorithm, both `is_slippery=False` and `is_slippery=True` scenarios are considered. For each trained agent, we will

- plot the evaluation return versus the number of cumulative time steps [2];
- plot the state value function that corresponds to each trained agent.

Algorithm 2: SARSA (On-policy TD(0) Control)

Input: MDP discount factor $\gamma = 0.95$, initial learning rate $\alpha_0 = 0.1$, initial $\epsilon_0 = 1$, number of episodes $N_e = 80000$

Output: Optimal action-value function $Q^\star$ and policy $\pi^\star$

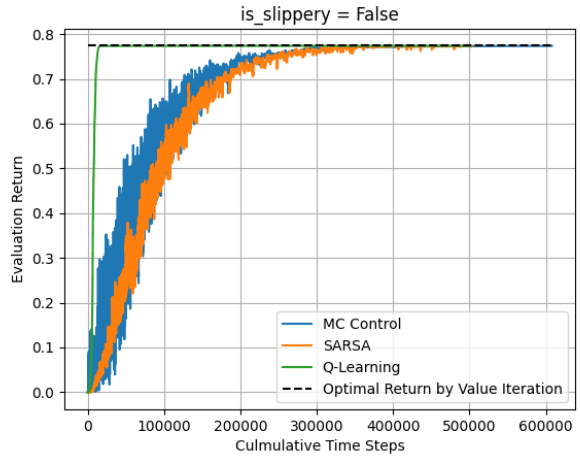
Initialize $Q(s,a) \leftarrow 0$ for all (s,a)
for *episode* $k = 1, 2, \ldots, N_e$ **do**
  Schedule ϵ_k by (11) to impose GLIE
  Schedule α_k by (12) to enhance learning stability
  Initialize state s_0
  Choose $a_0 \sim \pi(\cdot|s_0)$ (ϵ_k-greedy w.r.t. Q)
  for $t = 0, 1, 2, \ldots$ **do**
    Take action a_t, observe (r_t, s_{t+1})
    Choose $a_{t+1} \sim \pi(\cdot|s_{t+1})$ (ϵ_k-greedy w.r.t. Q)
    Update: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_k \left[r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$
    $s_t \leftarrow s_{t+1}, \quad a_t \leftarrow a_{t+1}$
    if s_t *is terminal* **then**
      break
$\pi^\star(s) \leftarrow \arg\max_a Q(s,a), \quad Q^\star \leftarrow Q$
return $Q^\star$, $\pi^\star$

Algorithm 3: Q-learning (Off-policy TD(0) Control)

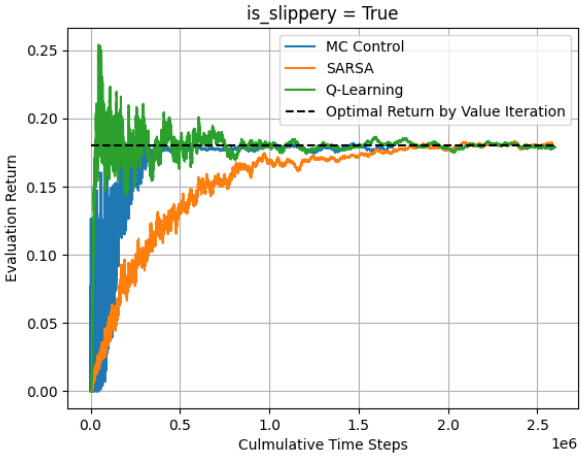
Input: MDP discount factor $\gamma = 0.95$, initial learning rate $\alpha_0 = 0.1$, initial $\epsilon_0 = 1$, number of episodes $N_e = 80000$

Output: Optimal action-value function $Q^\star$ and policy $\pi^\star$

Initialize $Q(s,a) \leftarrow 0$ for all (s,a)
for *episode* $k = 1, 2, \ldots, N_e$ **do**
  Schedule ϵ_k by (11) to impose GLIE
  Schedule α_k by (12) to enhance learning stability
  Initialize state s_0
  for $t = 0, 1, 2, \ldots$ *until termination* **do**
    Choose $a_t \sim \pi(\cdot|s_t)$ (ϵ_k-greedy w.r.t. Q)
    Take action a_t, observe (r_t, s_{t+1})
    Update: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_k \left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$
    $s_t \leftarrow s_{t+1}$
    if s_t *is terminal* **then**
      break
$\pi^\star(s) \leftarrow \arg\max_a Q(s,a), \quad Q^\star \leftarrow Q$
return $Q^\star$, $\pi^\star$



(a) `is_slippery=False`



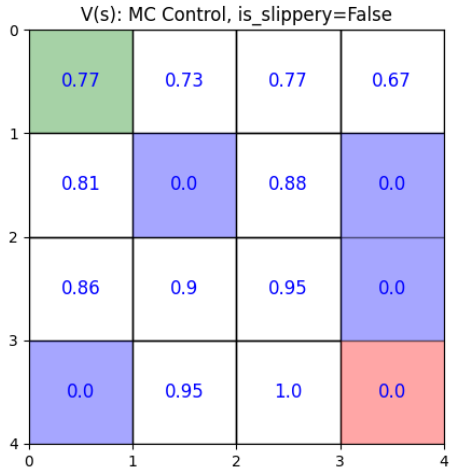
(b) `is_slippery=True`

Fig. 2: Evaluation return vs. cumulative time steps.

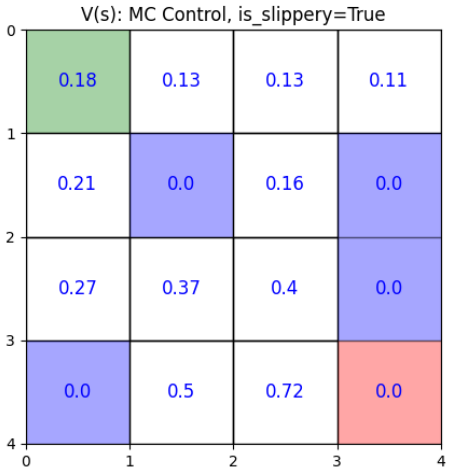
- plot the policy that corresponds to each trained agent;

The evaluation returns versus cumulative time steps [2] for all three algorithms are shown in Fig. 2. In these figures, only a single training run is carried out. As can be seen, all three algorithms converge to the same value, which coincides with the value obtained by value iteration. Comparing Figs. 2a and 2b, we can see that the learning curves are generally noisier in the case where `is_slippery=True`. In general, Q-learning yields the best performance in terms of convergence speed and learning stability. Also, note that the learning curves would be noisier without the learning rate scheduling in (12).

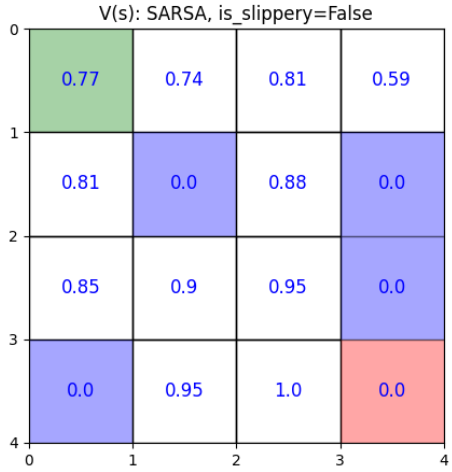
The approximated optimal state-value function $V^\star(s)$ and the corresponding greedy policy $\pi^\star(s)$ for all three algorithms with `is_slippery=False` and `is_slippery=True` are shown in Figs. 3 and 4. As can be seen, all three algorithms converge approximately to the state-value functions and policies obtained by value iteration, which is given in homework 1.



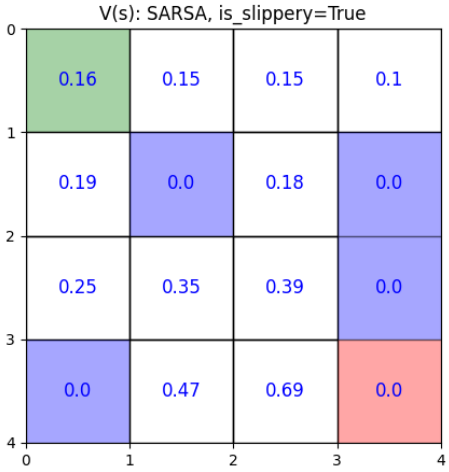
(a) MC Control with `is_slippery=False`



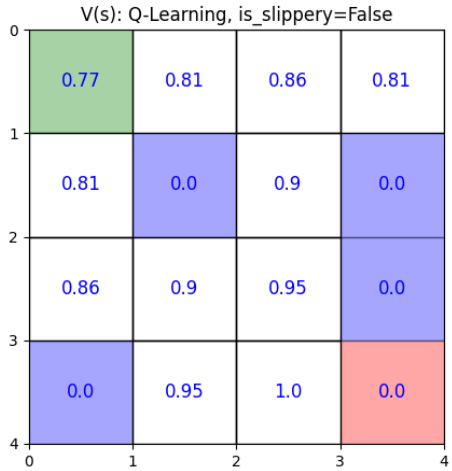
(b) MC Control with `is_slippery=True`



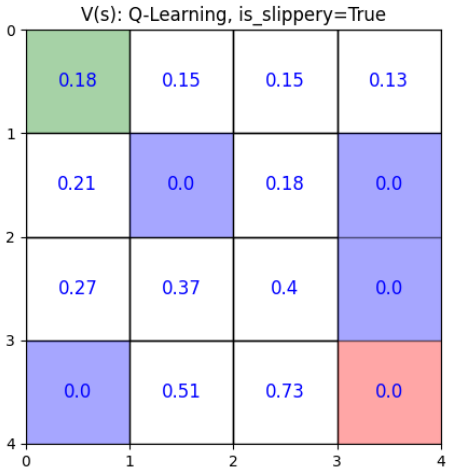
(c) SARSA with `is_slippery=False`



(d) SARSA with `is_slippery=True`

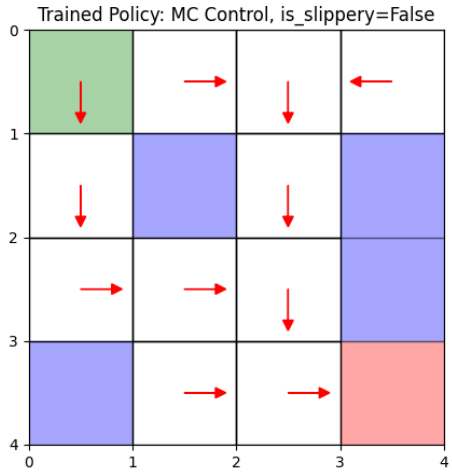


(e) Q-learning with `is_slippery=False`

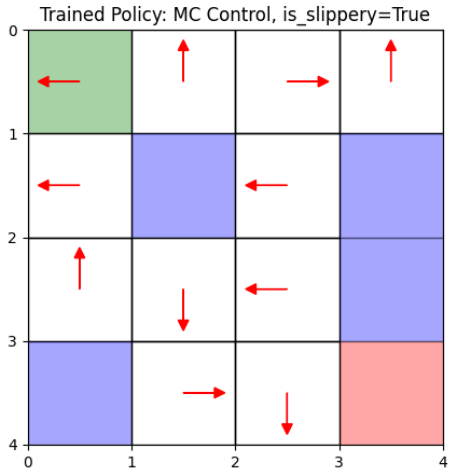


(f) Q-learning with `is_slippery=True`

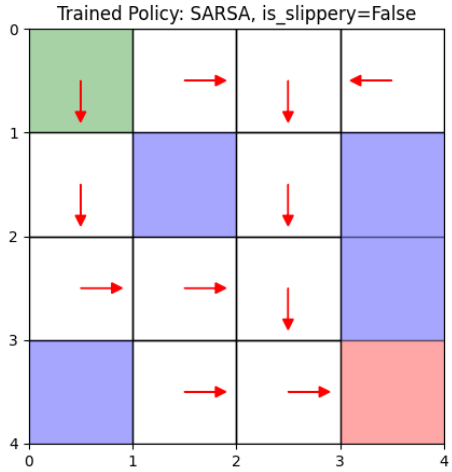
Fig. 3: State-value function $V(s)$.



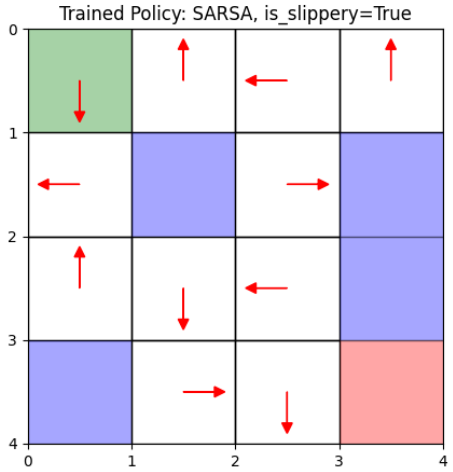
(a) MC Control with `is_slippery=False`



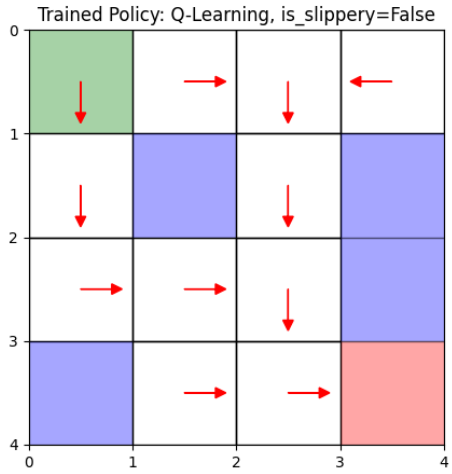
(b) MC Control with `is_slippery=True`



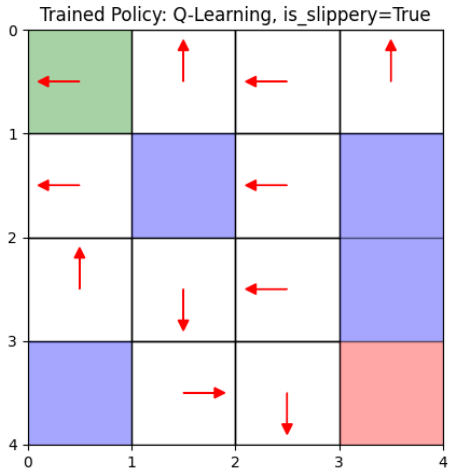
(c) SARSA with `is_slippery=False`



(d) SARSA with `is_slippery=True`



(e) Q-learning with `is_slippery=False`



(f) Q-learning with `is_slippery=True`

Fig. 4: Trained policy $\pi(a|s)$. The arrows indicate the action taken at every state (given by the policy). Policies at the holes are not plotted since they are meaningless.

IV. Conclusion

We observe that under the same learning rate and ϵ schedule, Q-learning significantly outperformed MC control and SARSA in terms of learning stability and convergence speed. This can be seen in Fig. 2. The learning problem can be challenging because the rewards are mostly zeros across the state space. The reward is non-zero only when the goal is reached. MC control, which only updates the policy when the entire episode is complete, struggles to simply sample an episode that actually reaches the goal. The challenge is even more significant when `is_slippery=True`. SARSA suffers similarly for this same reason due to its on-policy nature. The reason that Q-learning outperforms MC control and SARSA in our case is that its off-policy nature, in conjunction with TD learning, enables it to more efficiently learn the optimal value function from sampled episodes.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[2] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. [Online]. Available: https://www.marl-book.com