Reinforcement Learning HW2 Report: Tabular Model-Free Control on FrozenLake

Sunny Deshpande (sunnynd2)

I. Introduction

This homework introduces three core **model-free reinforcement learning** methods for small, discrete environments. Unlike Dynamic Programming (DP) from HW1, which required a known model of the environment, these algorithms learn directly from experience.

We train and compare:

- **On-policy First-Visit Monte Carlo (MC) Control**
- **SARSA (on-policy TD(0))**
- **Q-Learning (off-policy TD(0))**

The testbed is the classic `FrozenLake-v1` environment from Gymnasium. The agent must move from a start tile to the goal without falling into holes. When the map is `slippery`, movement becomes stochastic—making the problem more realistic and challenging.

The goal is to implement these methods, compare their learning behavior, and analyze their resulting policies and value functions.

II. Environment and MDP Setup

State space: 16 discrete states (4x4 grid). **Actions:** 0=Left, 1=Down, 2=Right, 3=Up. **Map:**

[SFFF, FHFH, FFFH, HFFG]

Reward: +1 for reaching the goal, 0 otherwise. **Terminal states:** Holes and goal. **Discount factor:** $\gamma = 0.95$.

Deterministic vs Stochastic: `is_slippery=False` means deterministic transitions. `is_slippery=True` adds randomness to actions (agent may slip sideways).

III. Algorithms and Intuition

Each method learns a table $Q(s, a)$ that estimates how good it is to take action a in state s.

A. Monte Carlo Control

MC control learns from full episodes. After an episode ends, it computes the total return G_t for each (s, a) the first time it was visited and updates:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[G_t - Q(s, a)]. \quad (1)$$

MC learns from complete experience, not from single steps. It is simple but slower to converge.

Fig. 1: FrozenLake 4x4 map (S=Start, F=Frozen, H=Hole, G=Goal).

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

SARSA updates Q at every step using the next action taken under the same ϵ-greedy policy:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]. \quad (2)$$

On-policy: it learns the value of the same policy it follows (safe and conservative).

C. Q-Learning (Off-Policy TD(0))

Q-learning also updates after each step, but assumes the next action is greedy:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]. \quad (3)$$

Off-policy: learns an optimal greedy policy while still exploring randomly. Usually learns faster but can overestimate.

D. Common Training Setup

- Learning rate $\alpha = 0.1$
- Discount $\gamma = 0.95$
- ϵ decays linearly from 1.0 to 0.05
- Max steps per agent: 100,000
- Evaluation every 1,000 steps using 50 greedy episodes

IV. Evaluation Protocol

Every 1,000 training steps, the current Q-table is frozen and evaluated without exploration. We compute the average total return across 50 episodes, plotting it as "Evaluation Return vs Steps." This shows how quickly and effectively each algorithm improves its policy.

V. Results and Discussion

A. Monte Carlo Control

MC control learns slower because it only updates after entire episodes finish. In deterministic maps, it eventually finds an optimal policy; in slippery maps, convergence is noisy and slower.

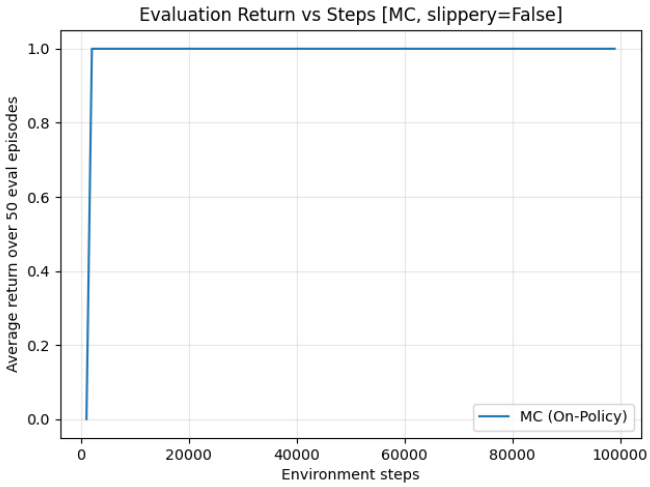

Fig. 2: MC Control – Evaluation Return vs Steps (is_slippery=False).

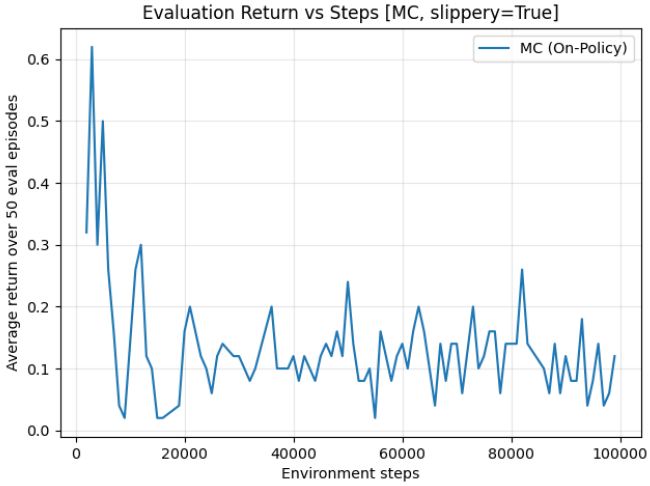

Fig. 3: MC Control – Evaluation Return vs Steps (is_slippery=True).

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

SARSA learns much faster, improving Q values at every step. It produces safer, more stable policies, especially under slip conditions.

C. Q-Learning (Off-Policy TD(0))

Q-learning converges fastest and usually achieves the highest final performance on non-slippery maps. However, its greediness can make it slightly unstable on slippery terrain.

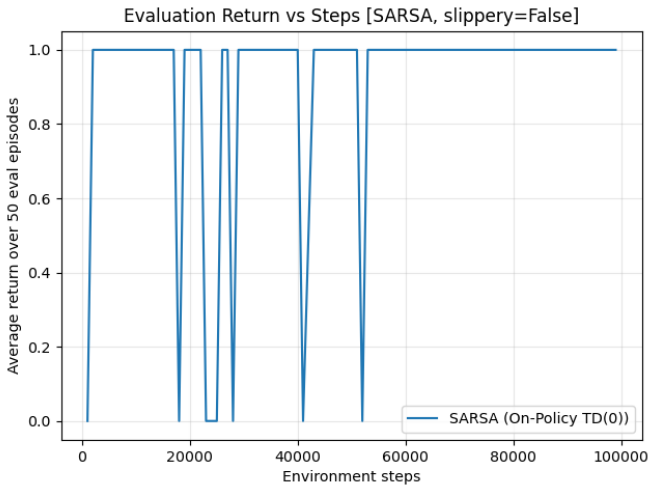

Fig. 4: SARSA – Evaluation Return vs Steps (is_slippery=False).

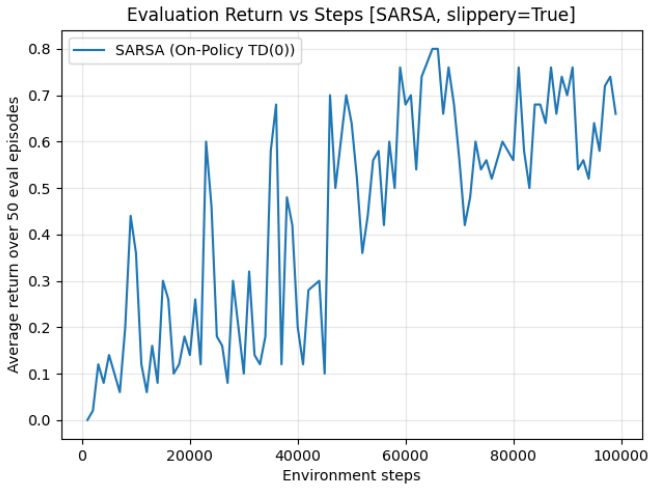

Fig. 5: SARSA – Evaluation Return vs Steps (is_slippery=True).

D. Learned Policies

Figures 8 and 9 show the learned policies as arrow maps. In deterministic mode, all methods find the same shortest path to the goal. Under slippery dynamics, SARSA tends to pick safer, more reliable routes.

E. State-Value Heatmaps

Each figure below shows $V(s) = \max_a Q(s, a)$ for each method. Values are highest near the goal and decay with distance or hole risk. Slippery cases show more spread-out (softer) values due to uncertainty.

VI. Summary of Observations

- MC Control learns eventually but needs many episodes.
- SARSA learns faster and handles stochastic slip more safely.
- Q-Learning converges fastest but may overestimate under noise.

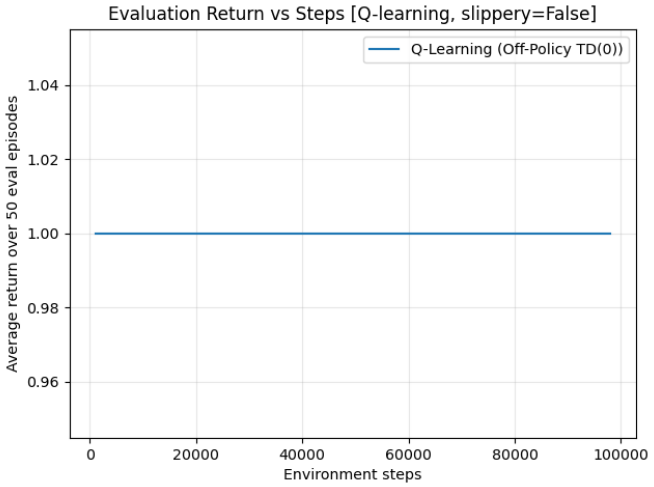

Fig. 6: Q-Learning – Evaluation Return vs Steps (is_slippery=False).

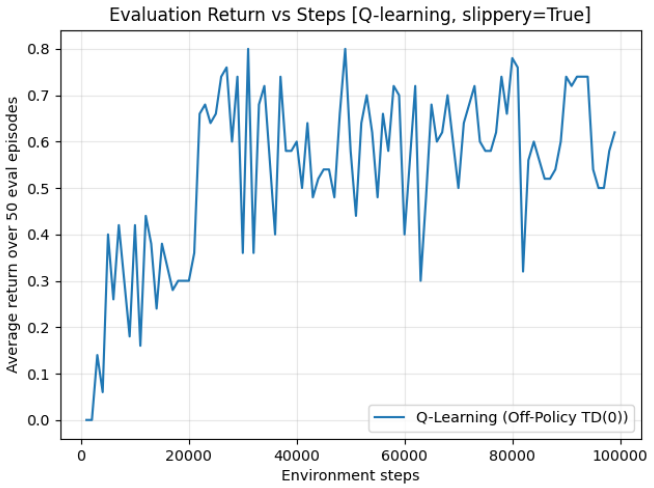

Fig. 7: Q-Learning – Evaluation Return vs Steps (is_slippery=True).

- Slipperiness lowers overall performance for all agents.

VII. Conclusion

This experiment showed how different tabular model-free algorithms learn from experience without knowing the transition model. MC learns slowly but steadily, SARSA balances exploration and caution, and Q-learning is fast and greedy. All methods ultimately learn effective policies, but their learning speed and stability differ depending on environment stochasticity.

These insights build intuition for moving from tabular RL to more advanced topics such as function approximation, deep Q-learning, and actor-critic methods.

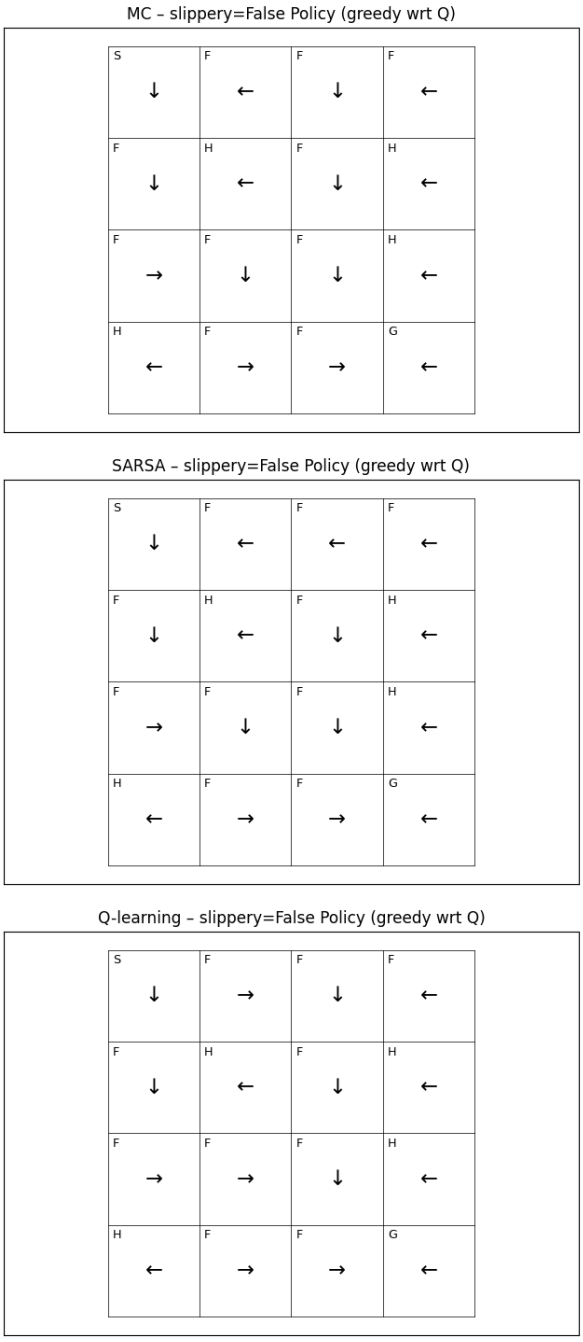

Fig. 8: Learned policies (non-slippery). Left–MC, Middle–SARSA, Right–Q-learning.

References

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

[2] Farama Foundation, "FrozenLake-v1 — Gymnasium Documentation," https://gymnasium.farama.org/environments/toy_text/frozen_lake/, accessed Oct. 2025.

[3] S. V. Albrecht, F. Christianos, and L. Ilsche, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, 2024. (Evaluation returns discussion, Ch. 2.7)

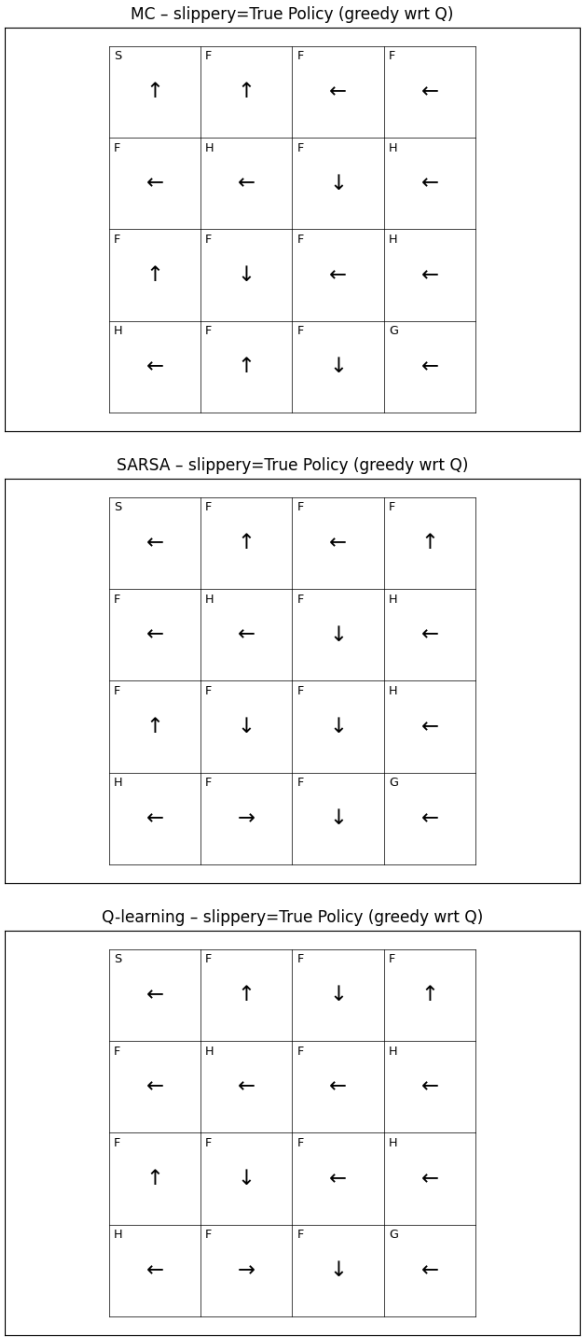

Fig. 9: Learned policies (slippery). Left–MC, Middle–SARSA, Right–Q-learning.

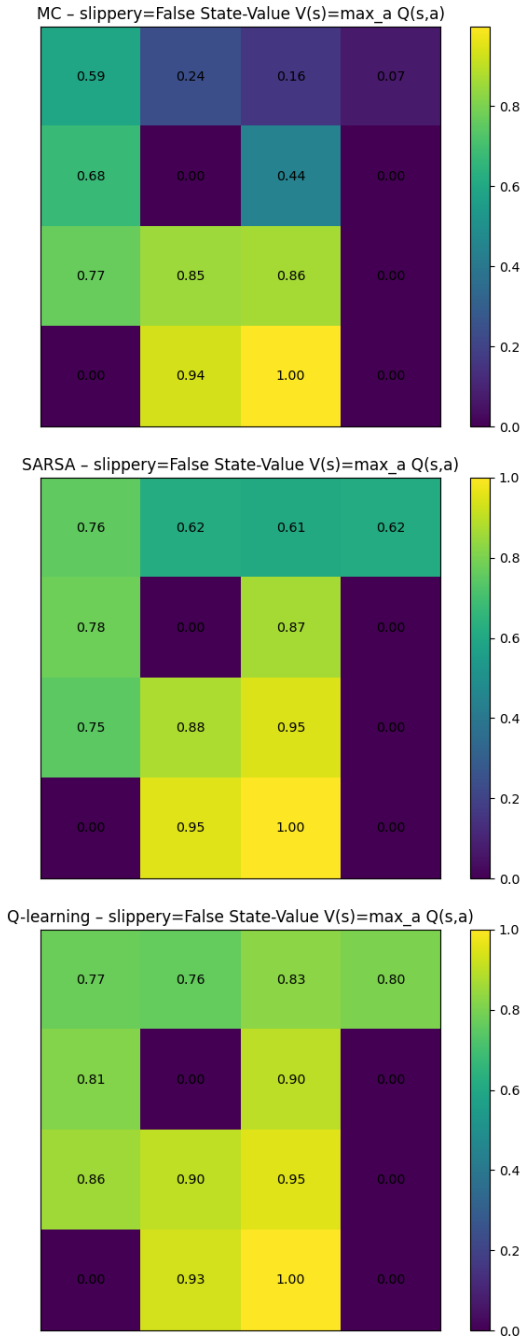

Fig. 10: State-value heatmaps (non-slippery).

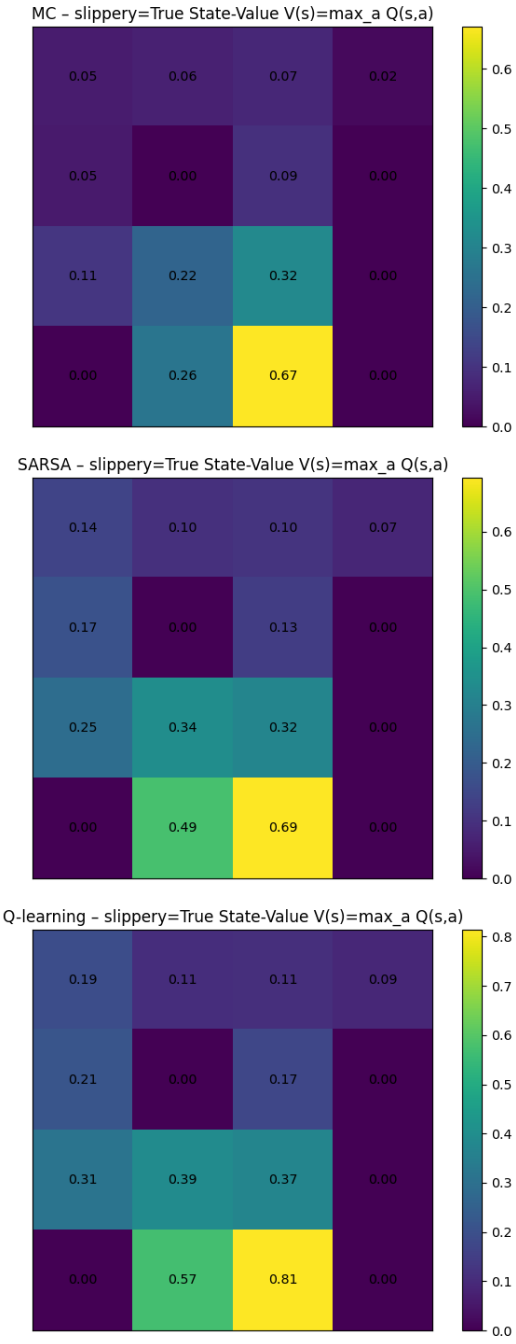

Fig. 11: State-value heatmaps (slippery).