AE598RL Homework 2 - Model-Free Tabular Methods

Cedric Swearingen
University of Illinois Urbana-Champaign

This report documents the implementation and results of Monte Carlo First-Visit On-Policy Control, SARSA, and Q-Learning model-free methods for determining the optimal policy and action value function for the Frozen Lake Toy Text environment. SARSA and Q-Learning approach the optimal policy while Monte Carlo does not with the given training time. The impacts of learning rate α and exploration rate ϵ are discussed.

I. Introduction

The following is copied from my Homework 1 [1].

The Frozen Lake toy text environment is a simple discrete-space environment with a square grid of tiles. Tiles can be either frozen or a hole, and a start and goal tile are initialized at opposite corners. The goal is to travel from the start tile to the goal tile without falling into a hole. This can be complicated by the ice being slippery, resulting in a chance of moving left or right instead of the intended direction.

This report uses a 4×4 world shown in table 1a, where S indicates start, F indicates a frozen tile, H indicates a hole, and G indicates the goal tile, with state indexing shown in table 1b.

S	F	F	F
F	H	F	H
F	F	F	H
H	F	F	G

(a) Layout of Frozen Lake

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

(b) Frozen Lake state map

Table 1 Layout (a) and state map (b) of Frozen Lake

The state space is therefore defined as $S = \{0, 1, \cdots, 15\}$. The action space is defined as $A = \{0, 1, 2, 3\}$ with 0 corresponding to left movement and the directions of increasing actions proceed counter-clockwise by quarter turns. If the ice is not slippery, the new state $s' = P(s, a)$ is always the position starting from state s and traveling in direction a. If this would exit the board, then $s' = s$, the agent does not move. In the slippery case, there is an equal chance for each of the state in the intended direction, to the left of the current state, and to the right of the current state, with any movement that results in exiting the board resulting in no net movement. Equation 1 defines the reward function [2].

$$R(s) = \begin{cases} 1 & s = 15 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

End of material copied from my Homework 1.

The methods used in this report are model-free, so do not have direct access to the state transition function or the reward function, instead state transitions are sampled from an external environment.

II. Methods

Three methods are used to learn the optimal policy. The first is Monte Carlo First-Visit On-policy control, the second is SARSA, and the third is Q-Learning. The algorithms for these methods are adapted from Sutton and Barto [3]. All methods were run with discount factor $\gamma = 0.95$ and an ϵ-greedy policy with $\epsilon_0 = 1$ with slight variations on the exploration schedule, and SARSA and Q-Learning were run with a learning rate $\alpha = 0.4$. Each method is run 10 independent times for 50,000 episodes each for the slippery case and 5,000 episodes for the non-slippery case, with each training run reporting the current greedy policy at regular intervals for a total of 50 sample policies per training run. The learning curve is produced by rolling out each of these policies 50 times and averaging the return, then averaging

those averages for each reporting batch. This method is based on the evaluation return method found in Albrecht et. al. in [4], with modifications to account for the available computational resources.

A. Monte Carlo First-Visit On-Policy

Monte Carlo First-Visit control averages the sampled returns of complete episodes to estimate $Q(s,a)$, updating along the path taken in any given episode, and only updating $Q(s,a)$ the first time in the episode that a given (s,a) pair appears. As a method for reducing the memory footprint of the algorithm, $Q(s,a)$ is updated using the identity shown in equation 2.

$$\overline{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}\left(x_n + \sum_{i=1}^{n-1} x_i\right) = \frac{1}{n}x_n + \frac{1}{n}\frac{n-1}{n-1}\sum_{i=1}^{n-1} x_i = \frac{1}{n}x_n + \frac{n-1}{n}\overline{x}_{n-1} \tag{2}$$

Monte Carlo methods have the benefit of computing an unbiased estimate of $Q(s,a)$ with the tradeoff of having high variance since the estimate is based on a full episode of random events in the form of state transitions and action selections.

This implementation uses a global exploration schedule that linearly decays from 1 to 0 throughout each training run. This was found to produce quicker convergence in small-scale tests compared to constant exploration rates and schedules that reached 0 exploration before the end of the training run.

1. Pseudocode

Inputs: Initial exploration rate ϵ_0, discount factor γ, maximum episodes for training E

1) Initialize $Q(s,a) \leftarrow 0, \quad \pi(s,a) \leftarrow \frac{1}{\|A\|}, \quad k(s,a) \leftarrow 0 \quad \forall s,a \in S, A$, $\epsilon \leftarrow \epsilon_0$, episode number $e \leftarrow 0$
2) Loop until $e = E$

Generate episode following π: $S_0, A_0, R_1, \cdots, S_{T-1}, A_{T-1}, R_T$

Set $g \leftarrow 0$, Loop over $t = T, T-1, T-2, \cdots, 0$

$g \leftarrow \gamma g + r_t$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \cdots, S_{t-1}, A_{t-1}$:

$k(S_t, A_t) \leftarrow k(S_t, A_t) + 1$

$Q(S_t, A_t) \leftarrow \frac{k(S_t,A_t)-1}{k(S_t,A_t)} Q(S_t, A_t) + \frac{1}{k(S_t,A_t)} g$

Loop over states $s \in S$

$\pi(s,a) \leftarrow \frac{1}{\|A\|} \quad \forall a \in A$ if $Q(s,a) = 0 \quad \forall a \in A$

Otherwise: $A^* \leftarrow \text{argmax}_{a \in A} Q(s,a), \quad \pi(s,a) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{\|A\|} & a = A^* \\ \frac{\epsilon}{\|A\|} & o.w. \end{cases}$

$\epsilon \leftarrow \left(1 - \frac{e}{E}\right)\epsilon_0, \quad e \leftarrow e + 1$

B. SARSA

SARSA is a bootstrapping-based on-policy TD(0) method. SARSA updates the policy after every environment step, resulting in improved sample efficiency compared to Monte Carlo methods and lower variance in the estimation of $Q(s,a)$ since the estimation only involves randomness from one state transition and one action choice. This comes at the cost of a bias in estimation, though in practice this method still shows good convergence.

The hyperparameters α and ϵ were chosen based on small-scale tests with 2000 episodes. $\alpha = 0.4$ was found to be a good balance of quick convergence and converging to the true optimal policy found in Homework 1 [1]. The exploration schedule $\epsilon = \epsilon_0 \cdot \max(0, (1 - \frac{5}{4}\frac{e}{E}))$ was selected since in small-scale tests, this method achieved the optimal policy once $\epsilon = 0$, so a short period of exploitation only is added to allow the policy to stabilize.

1. Pseudocode

Inputs: Initial exploration rate ϵ_0, discount factor γ, learning rate α, maximum episodes for training E

1) Initialize $Q(s,a) \leftarrow 0, \quad \pi(s,a) \leftarrow \frac{1}{\|A\|} \quad \forall s,a \in S, A$, $\epsilon \leftarrow \epsilon_0$, episode number $e \leftarrow 0$
2) Loop until $e = E$

Initialize $s \sim \beta_0, \quad a \sim \pi(s)$

Loop until episode ends

$s', r \sim env(s, a), \quad a' \sim \pi(s')$
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
$s \leftarrow s', \quad a \leftarrow a'$
Update policy
$\pi(s, a) \leftarrow \frac{1}{\|A\|} \quad \forall a \in A \text{ if } Q(s, a) = 0 \quad \forall a \in A$
Otherwise: $A^* \leftarrow \text{argmax}_{a \in A} Q(s, a), \quad \pi(s, a) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{\|A\|} & a = A^* \\ \frac{\epsilon}{\|A\|} & o.w. \end{cases}$
$\epsilon \leftarrow \epsilon_0 \cdot \max\left(0, 1 - \frac{5}{4}\frac{e}{E}\right), \quad e \leftarrow e + 1$

C. Q-Learning

Q-Learning is another TD(0) method very similar to SARSA, but off-policy instead. This offers the ability to use any policy (or existing dataset) to train the estimate of $Q(s, a)$, though this implementation uses the same ϵ-greedy policy with the same exploration schedule as SARSA. The hyperparameters are chosen to be the same as for SARSA so that the methods can be more directly compared.

1. Pseudocode

Inputs: Initial exploration rate ϵ_0, discount factor γ, learning rate α, maximum episodes for training E
1) Initialize $Q(s, a) \leftarrow 0, \quad \pi(s, a) \leftarrow \frac{1}{\|A\|} \quad \forall s, a \in S, A$, $\epsilon \leftarrow \epsilon_0$, episode number $e \leftarrow 0$
2) Loop until $e = E$
Initialize $s \sim \beta_0$
Loop until episode ends
$a \sim \pi(s)$
$s', r \sim env(s, a)$
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', \text{argmax}_{a' \in A}(Q(s', a'))) - Q(s, a)]$
$s \leftarrow s'$
Update policy
$\pi(s, a) \leftarrow \frac{1}{\|A\|} \quad \forall a \in A \text{ if } Q(s, a) = 0 \quad \forall a \in A$
Otherwise: $A^* \leftarrow \text{argmax}_{a \in A} Q(s, a), \quad \pi(s, a) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{\|A\|} & a = A^* \\ \frac{\epsilon}{\|A\|} & o.w. \end{cases}$
$\epsilon \leftarrow \epsilon_0 \cdot \max\left(0, 1 - \frac{5}{4}\frac{e}{E}\right), \quad e \leftarrow e + 1$

III. Results

A. Slippery case

Figure 1 shows the evaluation returns of all three methods as well as the evaluation return of the optimal policy found in Homework 1 [1]. The shaded area shows one standard deviation of the evaluation return. Both SARSA and Q-Learning achieved close to the optimal policy, while Monte Carlo did not quite make it.

Figure 2 shows the average of the greedy policies over all the training runs for a given method as a radial plot at each state. Both SARSA and Q-Learning show near-perfect agreement in all states except at state 2, but it is clear from figure 1 that the policy in this state does not have a significant impact on the evaluation return. Monte Carlo shows a different policy at states 0 and 14, the initial state and the penultimate state. This is likely the reason why the Monte Carlo method does not achieve the optimal evaluation return. Further investigation is required to determine if longer training time would result in the Monte Carlo policy converging to the optimal policy.

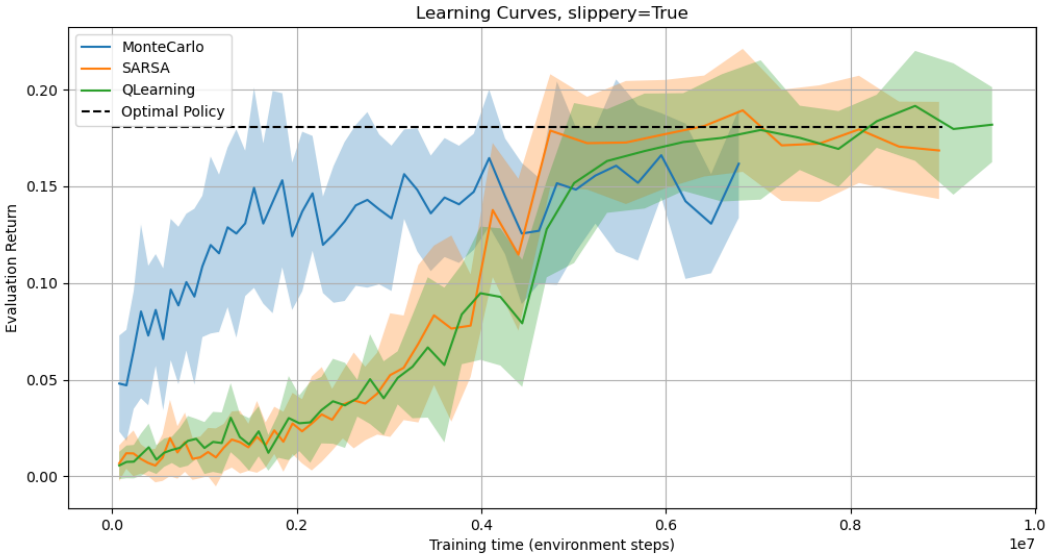
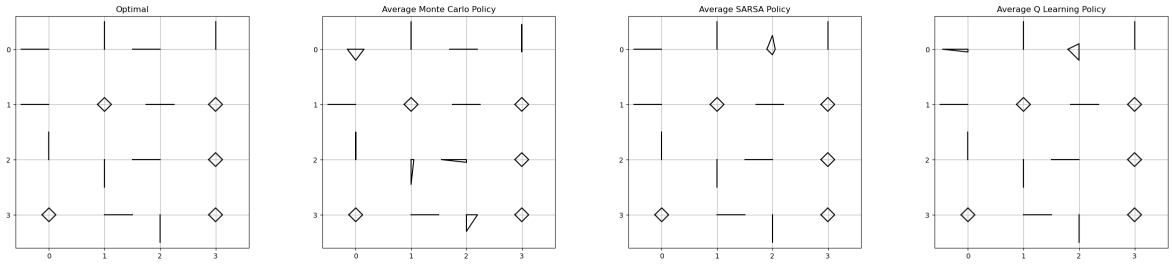


Fig. 1 Evaluation return for slippery Frozen Lake



(a) **(b)** **(c)** **(d)**

Fig. 2 Average policies after training for slippery case

B. Non-slippery case

Figure 3 shows the evaluation returns in the non-slippery case for each method. Monte Carlo converges quickly this in this case, with Q-Learning and SARSA converging much slower, though still achieving the optimal return. Figure 4 shows the average of greedy policies for each method. Notably, the average policy for SARSA does not match the optimal policy, but given that the evaluation return is equal to the optimal with zero variance, this mismatch is likely in states that are never visited for a particular training run's policy, and that particular training run did not visit those states after some point in the training. This is only possible because the non-slippery case is fully deterministic with a greedy policy.

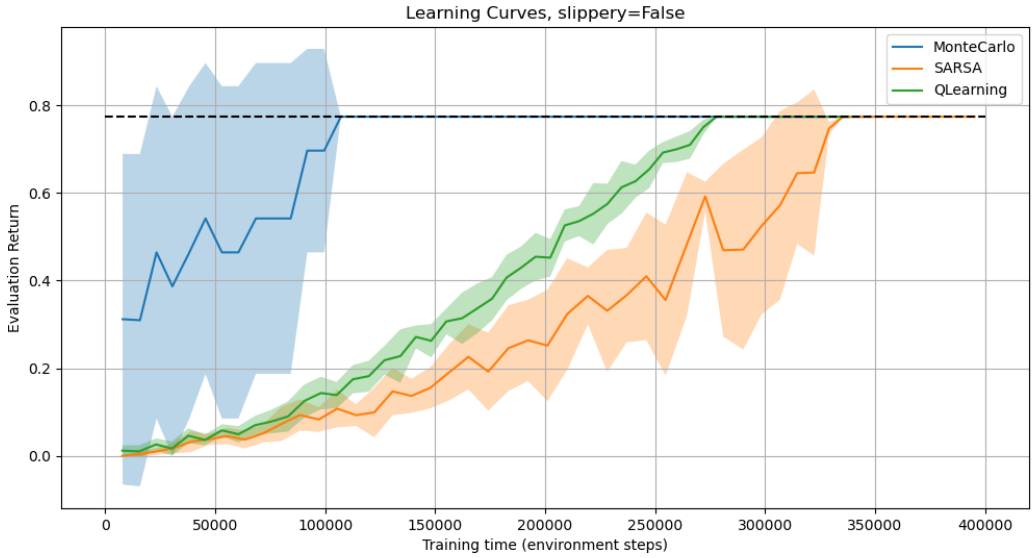
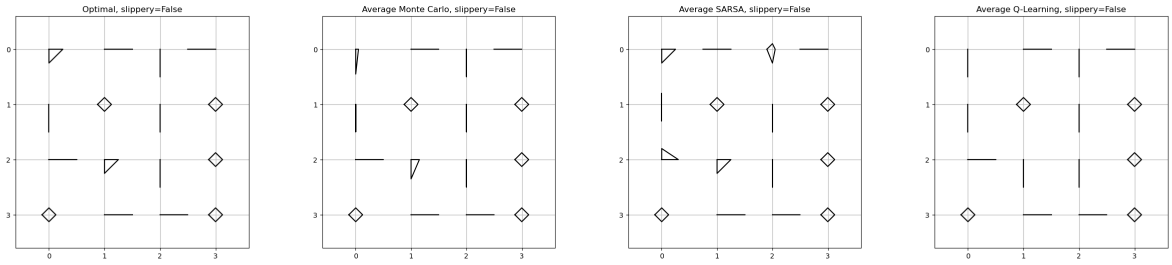


Fig. 3 Evaluation return for non-slippery Frozen Lake



(a) **(b)** **(c)** **(d)**

Fig. 4 Average policies after training for non-slippery case

IV. Conclusions

Model-free methods can be effectively deployed in situations where the true state transition function is unknown. This comes at a steep computational cost and only asymptotic guarantees of convergence. The computational costs are exacerbated by highly problem-specific tuning of hyperparameters, resulting in time-consuming trial-and-error approaches to find appropriate values or schedules, and performance can depend on the length of a training session, resulting in even more time-consuming tweaking.

References

[1] Swearingen, C., "AE598RL Homework 1 - Dynamic Programming," Tech. rep., University of Illinois Urbana-Champaign, 2025.

[2] "Frozen Lake - Gymnasium Documentation," Tech. rep., Farama Foundation, 2025. URL `https://gymnasium.farama.org/environments/toy_text/frozen_lake/`.

[3] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, 2020, Chaps. 5,6.

[4] Albrecht, S. V., Christianos, F., and Schäfer, L., *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*, MIT Press, 2024, Chap. 2.