Final Report

Boshen Zhang, Haoran Tang
{boshenzh,haoranta}@usc.edu

April 13, 2024

1 Final Problem Description

The problem we are addressing for the final project is to consistently solve two hard maps and achieve a discounted return of at least 0.8 (with g = 0.997) for 100 consecutive episodes. More specifically, our tasks include implementing reinforcement learning algorithms that outperform the two provided algorithms SARSA and Q-Learning in terms of converging to the optimal policy and value function with as few episodes as possible in solving the two hard maps. Moreover, we aim to fine-tune the hyper-parameters such as learning rate and epsilon decay rate to achieve the best overall performance.

2 Final Solution

Inspired by existing algorithms in the starter code, we implemented and conducted experiments on SARSA(λ) and Dyna-Q algorithm with Decaying Epsilon-Greedy.

Compared to SARSA, SARSA(λ) introduces the eligibility traces mechanism, $E_t(S,a) = \gamma\lambda E_{t-1}(s,a) + \mathbf{1}[s_t = s, a_t = a]$, which takes account for the recentness and frequency of each state-action's visitation. This is incorporated into the policy evaluation $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t E_t(s_t, a_t)$, where $\delta_t = r_t + \gamma Q(s_{t+1}, \alpha_{t+1} - Q(s_t, a_t)$ is the TD error. Given the eligibility traces mechanism, we expect to obtain better control over the bias-variance trade-off compared to SARSA by tuning λ, because it focuses on the sum of weighted n-step returns $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$.

We explore Dyna-Q because it leverages the benefits of both direct reinforcement learning and planning. Given a real experience, Dyna-Q improves the value function and the policy with direct RL, $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S', a) - Q(S,A)]$, learns the model, $Model(S,A) \leftarrow R, S'$, and gives rise to simulated experience using the learned model. Sequentially, it achieves planning by applying RL methods to the simulated experiences. Equipped with the planning component, Dyna-Q can quickly adapt to changes in the environment that offer a higher adaptability than SARSA.

Furthermore, we have conducted hyper-parameter fine-tuning including epsilon decay rate, learning rate α, and the number of planning steps for Dyna-Q and SARSA(λ) to improve agent's learning performance. Among all the hyper-parameters, epsilon decay was the most prominent.

3 Results

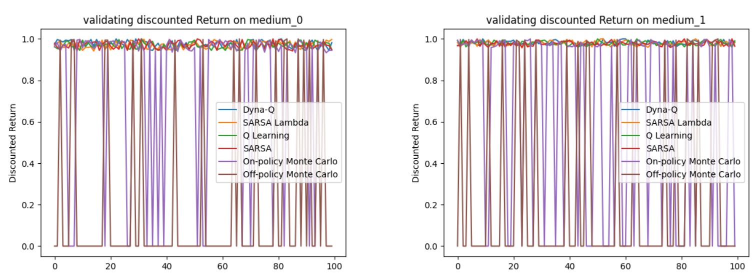
We collected the results and the Q-tables for training SARSA(λ), Dyna-Q, SARSA, Q-learning, On-policy Monte Carlo Learning, and Off-policy Monte Carlo Learning for 2000 episodes with 0.00001 epsilon decay. Afterward, we tested each algorithm using a greedy agent. The greedy agent uses the Q-values saved after training and executes actions based on selecting the maximum Q-values of each state-action. We have experimented with SARSA(λ) and Dyna-Q over 100 episodes and compared them with other methods mentioned above. We have collected average discounted returns across different algorithms as shown in Table 1. As one can see, Dyna-Q, SARSA(λ), SARSA, and Q-learning were all able to achieve more than 0.8 average discounted return on the hard map with proper fine-tuning. However, Monte Carlo Learning did not show ideal results on any map. Figure 1 shows that Monte Carlo learning fails randomly and is inconsistent over 100 episodes on medium maps.

Table 1: Average discounted return over 100 episodes for each Algorithm

Algorithm	**Easy**	**Medium**	**Hard**
Dyna-Q	0.9826	0.9779	0.9501
SARSA(λ)	0.9960	0.9788	0.8373
SARSA	0.9957	0.9703	0.9520
Q-learning	0.9845	0.9798	0.9523
On-Policy Monte-Carlo Learning	0.7772	0.6301	N/A
Off-Policy Monte-Carlo Learning	0.5329	0.3667	N/A

Figure 2 shows the discounted return on both hard maps. As one can observe, Dyna-Q, SARSA, and Q-learning achieved a consistent return for each episode, successfully solving both hard maps. SARSA(λ) performed poorly on hard maps, especially on the hard_0 map. Unlike other algorithms, SARSA(λ) are inconsistent over 100 episodes.

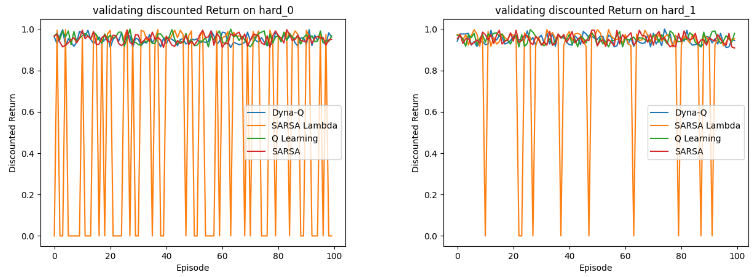
Figure 1: Total discounted return over 100 episode on medium_0 (left)and medium_1(right) map



4 Limitations

Our experiments demonstrate that SARSA(λ) was successful in solving the medium-complexity maps, but struggled to achieve optimal performance on the hard maps. On the medium maps, SARSA(λ) consistently found near-optimal solutions. However, the algorithm's performance degraded on the hard maps, indicating that further hyper-parameter tuning is required to improve its effectiveness. For example, testing on different values of Eligibility Trace Decay λ. In additional,

Figure 2: Total discounted return over 100 episode on hard_0 (left)and hard_1(right) map



our deployed exploration method Epsilon Greedy does not have a memory about the visitation of state and state-action. It essentially treats each state equally which explain our large number of training episodes.

Moreover, The agents were only trained on six maps with three levels of difficulty. While this provided a solid foundation for initial testing, it does not fully assess the robustness of the algorithms across a wider range of environments. To more rigorously evaluate the algorithms, it would be beneficial to generate additional maps with different characteristics, such as larger maps with more complex layouts and maps that introduce new navigational challenges, such as dynamic obstacles, or narrow passageways. Expanding the diversity of map features would stress test the algorithms' capabilities.