Frozen Lake with Reinforcement Learning

Mengyu LIANG[1], Nhat Mai NGUYEN[2] and Jinji SHEN[3]

[1 2 3] Paris Saclay, F-91192 Gif-sur-Yvette, France

Abstract. This report discusses the implementation and comparison of two widely used reinforcement learning algorithms, Q-Learning and SARSA, in training an agent to navigate a simple grid world game called Frozen Lake. The game requires the agent to reach the final destination without falling into any of the holes. The report presents a self-built Frozen Lake environment and the execution of each algorithm on such an environment. The results indicate that Q-learning outperforms SARSA in terms of speed to find the optimal solution and the overall success rate.

Keywords: Frozen Lake, Reinforcement Learning, Q-learning, SARSA

1 Introduction

Frozen Lake is a simple grid world game where the agent (player) must navigate through a frozen lake to reach a goal without falling into any of the holes. The Frozen Lake project has realistic implications in the field of reinforcement learning and artificial intelligence. By using a simple environment like Frozen Lake, we can explore and test the effectiveness of reinforcement learning (RL) algorithms in solving complex problems in various fields. The project can also be a valuable resource for identifying the strengths and weaknesses of different RL algorithms and their applications in real-world problems. Moreover, the Frozen Lake environment has practical applications in areas such as robotics, autonomous vehicles, and game design. By training an agent to navigate through the Frozen Lake environment, researchers can develop better algorithms for controlling robots and autonomous vehicles in complex environments.

The objective of this project is to implement and compare different RL algorithms to train an agent to navigate through Frozen Lake. Specifically, we will use Q-Learning and SARSA algorithms to train an agent to find the optimal policy for navigating the Frozen Lake environment.

2 Methodology

2.1 Environment

We set up the self-defined Frozen Lake environment (the original version is in OpenAI's Gym library) from scratch to understand the environment. In our environment (see Fig. 1), the surface will consist of a fixed 4x4 grid world where the agent can move in four directions (up, down, left, right). From a starting point (S), the agent is expected to reach the final destination (G) using the frozen path (F) and avoiding the hole (H). If the agent manages to reach G, the reward is +1 and 0 otherwise.

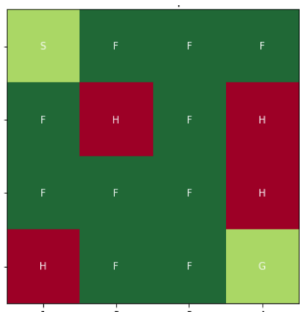

Fig. 1. Self-built frozen lake grid

2.2 **Agents**

Q-learning and SARSA are two widely used reinforcement learning algorithms that work well in a finite environment (episodic environment). Q-learning is a model-free, off-policy algorithm that learns to estimate the optimal action-value function by iteratively updating the Q-values of state-action pairs using the Bellman equation. SARSA, is also a model-free algorithm that learns to estimate the state-action value function by iteratively updating the Q-values using the on-policy approach, where the next action is chosen according to the current policy. These algorithms have been used successfully in various game-playing scenarios, including classic games like Tic-Tac-Toe, Chess as well as more complex games like Atari games and Go. They have also been applied to real-world problems such as robotics control, recommendation systems, and autonomous driving. So we believe these two algorithms are suitable for our game as well. The Q-value update formula for each algorithm is as followed:

$$\text{Q-learning: } Q(s,a) \leftarrow (1 - \alpha)\, Q(s,a) + \alpha\, [r + \gamma \max(Q(s',a'))]$$
$$\text{SARSA: } Q(s,a) \leftarrow (1 - \alpha)\, Q(s,a) + \alpha\, [r + \gamma\, Q(s',a')]$$

where $Q(s,a)$ is the estimated action-value function for state s and action a, α is the learning rate, r is the immediate reward received after taking action a in state s, γ is the discount factor, s' is the next state, and a' is the next action. For the exploitation and exploration, ε-greedy is used. The agent selects the action with the highest expected reward (exploitation) with probability 1- ε, and it selects a random action (exploration) with probability ε. At the beginning, we set high ε so that the agent can explore more. Then ε decays as the agent gains enough info on the environment [1].

3 Result

From Table 1 and Fig. 2, we can observe that Q-learning outperforms SARSA in both the speed to find the optimal solution and the overall success rate. With Q-learning agent, we successfully reach the 'G' point in less than 100 episodes and receive the reward of 1, however when using the SARSA agent, it takes nearly 200 episodes to achieve the same goal for the first time. This is because Q-learning learns the optimal policy by updating the Q-values based on the maximum expected future reward achievable from each state-action pair regardless of the policy that generated the experience, which helps to exploit the high reward associated with the goal tile 'G' and avoid the low reward associated with the holes [2]; SARSA, on the other hand, updates the Q-values based on the expected future reward achievable from the next state-action pair under the current policy, thus may require more exploration and trial-and-error to learn the optimal policy, resulting in slower convergence and lower success rate [3].

As the agents learn more about the environment (number of episodes increases), both agents have quite the same performance (85% vs. 88% success rate and 6.43 vs. 6.48 average step taken per episode for the last 100 episodes). As expected, the random agent performs badly as it does not learn the environment at all, thus being surpassed by both Q-learning and SARSA.

Table 1. % episode finished and average step across agents

Metrics	Q-learning	SARSA	Random
% of episodes finished successfully	81.2%	73.7%	1%
% of episodes finished successfully (last 100 episodes)	88%	85%	1%
Average number of steps in an episode	6.49	6.49	7.64
Average number of steps in an episode (last 100 episodes)	6.43	6.48	7.63

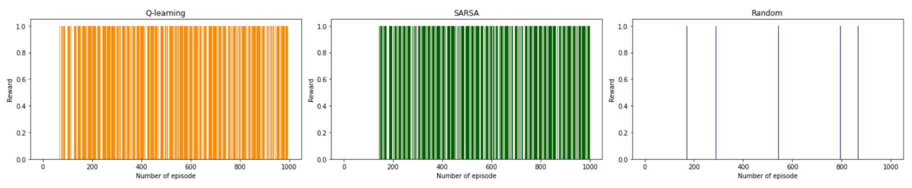

Fig. 2. Reward per episode across agents

According to Fig. 3, we can see that the steps taken in an episode are generally large for both Q-learning and SARSA at the beginning because we take a large amount of random actions, as the agents learn more later on, they gradually diminish the exploration probability and take more informed steps, leading to fewer steps. The results of random agents are consistent with the above description, without any learning mechanism, it would not be able to improve its performance over time, so that the number of steps to reach the goal or the hole stays always random and disorderly.

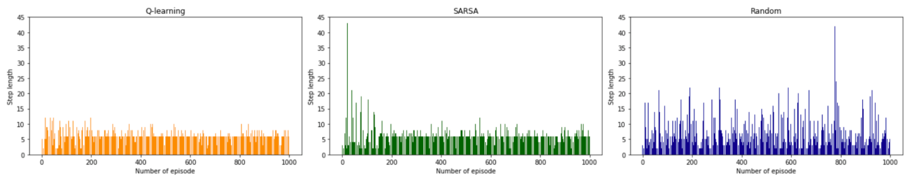

Fig. 3. Step length across agents

In Fig. 4, both agents have low values at H position. The G position has 0 value because the game ends at this point and the Q-value is not updated anymore. The closer the agent is to the G, the higher the values are. For SARSA, the top right cells with 0 values can be explained by the conservative nature of SARSA during exploration. This means it tends to favor actions that have already been tried and tested. In our trial, SARSA seems to stick to the found path (going down first and then moving right to reach G).

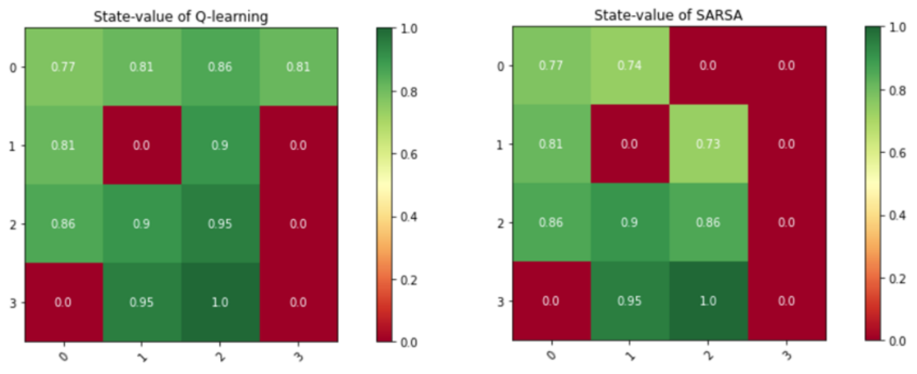

Fig. 4. State-value function

We also compared the differences between Q-learning and SARSA in terms of their sensitivity to certain parameters (see Fig. 5 and Fig.6). When the discount factor is set to 0, the agent only considers the immediate reward and ignores the potential future rewards, resulting in poor performance. However, when the discount factor is greater than 0, the agents are able to learn and perform better. When the learning rate is set to 0, the Q-values of the agent are not updated and as a result, the agent is unable to perform. As expected, when the learning rate is greater than 0, the Q-values are updated with consideration to future rewards, enabling the agent to learn and perform.

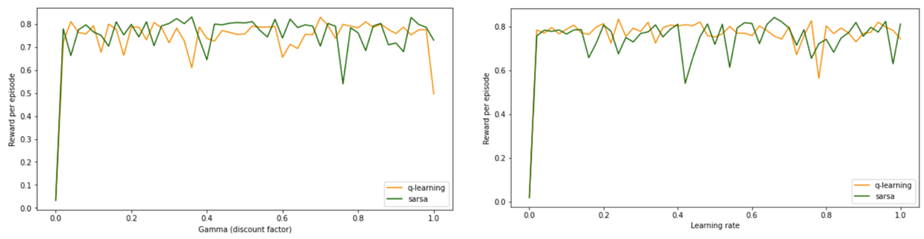

Fig. 5. Agent performance sensitivity to discount factor and learning rate

It has been observed that Q-learning performs better with a high exploration rate, while SARSA performs better with a low exploration rate. In the case of Frozen lake, since the environment is relatively small and has few state-action pairs, a high exploration rate may be beneficial for Q-learning to discover new state-action pairs and achieve better performance. For SARSA, a low exploration rate may be more appropriate since the environment is small and it is easier for SARSA to converge to a good policy by sticking to the current policy [4].

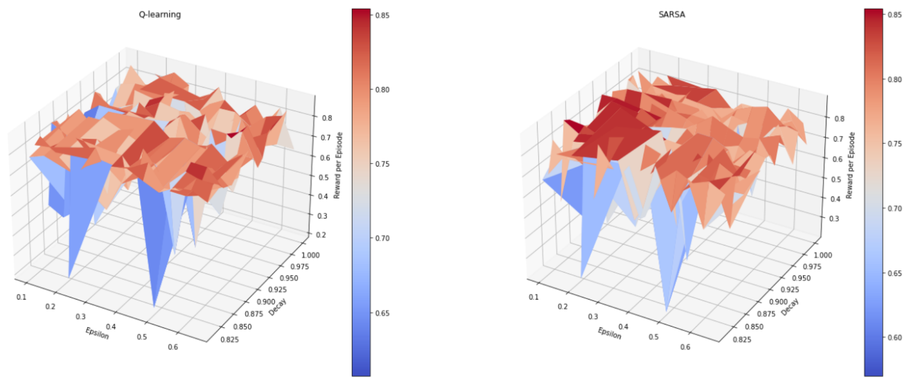

Fig. 6. Agent performance sensitivity to epsilon (exploration rate) & decay rate

4 Conclusion

In conclusion, this report presented the implementation and comparison of Q-learning and SARSA algorithms in training an agent to navigate the Frozen Lake environment. The results showed that Q-learning outperformed SARSA in terms of speed to find the optimal solution and overall success rate. The study highlights the importance of choosing appropriate RL algorithms for different problem domains and provides insights into the strengths and weaknesses of Q-learning and SARSA. The Frozen Lake environment provides a valuable resource for researchers to explore and test the effectiveness of RL algorithms in solving complex problems in various fields, including robotics, autonomous vehicles, and game design. Future research can focus on applying other RL algorithms to the Frozen Lake environment or extending the environment to more complex scenarios.

References

1. Gupta A, Roy P P, Dutt V. Evaluation of instance-based learning and q-learning algorithms in dynamic environments[J]. IEEE Access, 2021, 9: 138775-138790.
2. Yong S J, Park H G, You Y H, et al. Q-Learning Policy and Reward Design for Efficient Path Selection[J]. Journal of Advanced Navigation Technology, 2022, 26(2): 72-77.
3. Pena B D, Banuti D T. Reinforcement learning for pathfinding with restricted observation space in variable complexity environments[C]//AIAA scitech 2021 forum. 2021: 1755.
4. Mohan P, Sharma L, Narayan P. Optimal path finding using iterative sarsa[C]//2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2021: 811-817.