Reinforcement Learning Methods on Frozen Lake Maze

Jaiden Callies (CS 4033) Alex Hunt (CS 5033)

1. Domain

Frozen Lake is a single-player maze game. The goal of the game is to maneuver an agent from a starting state to a goal state that lies across a frozen lake. The game environment is traditionally a 4x4 two-dimensional grid with 16 spaces. All spaces outside of the start and goal state represent either frozen portions of the lake or holes within the lake. To win the game, the player must reach the goal state without falling into one of the holes. If the player falls into one of the holes, the player loses. A player may move one space at a time either left, right, up, or down across the grid. Additionally, the lake is slippery. Therefore, the actual movements of the agent may not mimic the agent's chosen actions. We used the Frozen Lake interface from Gymnasium to train our agent. Using this interface, the default reward function allocates a reward of 1.0 if an agent reaches the goal state. No other actions produce a reward.

2. Hypotheses

Our hypotheses include the following:

1. With dynamic programming, the average reward per episode will increase steadily as γ increases.
2. Modifying the default reward function will result in the agent achieving a higher average accumulated reward.
3. Adding heuristics to a Q-Learning algorithm will improve the average reward per episode.
4. Finding the optimal values of γ, λ, ϵ, and α will improve the average reward per episode.

3. Experiments

3.1. Hypothesis 1

Dynamic programming is an RL method in which the agent has knowledge of the entire MDP and consequently, can incorporate planning using the Bellman equations. For hypothesis 1, we used dynamic programming to run our simulations. A higher discount factor indicates that the agent is concerned with future rewards, as opposed to only immediate rewards, to a greater extent. Therefore, we expected the average accumulated reward to increase as the γ value increases. As seen in figure 1, the trend of higher γ values producing increased rewards is not always true for γ values between 0.2 and 0.6. However, it is clear that the agent using a γ value of 1.0 significantly outperformed all other agents.

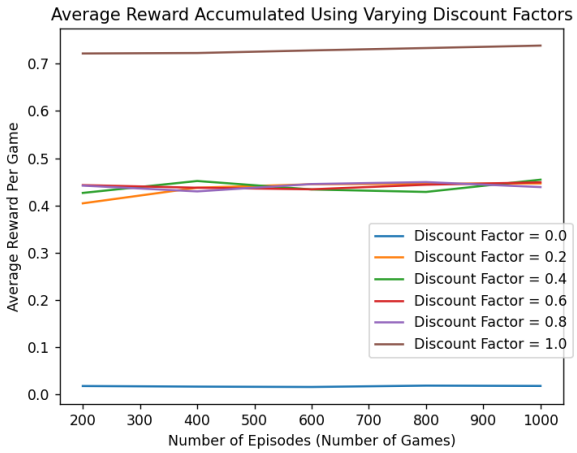

Figure 1. The average reward per episode using agents that learn according to dynamic programming principles with varying discount factors.

3.2. Hypothesis 2

Unlike dynamic programming, Monte Carlo learning does not require knowledge of the full MDP. Instead, the agent learns through episodic experience. For the second experiment, we used varying reward functions with MC methods using $\epsilon = \frac{1}{k}$ and $\gamma = 0.9$. Here, k represents the episode number. The default reward function allocates a reward of 1.0 when the agent reaches the goal state. In addition to this reward, the negative reward function also provides the agent with a reward of -1.0 for falling into the lake. Finally, the negative and positive reward function includes all previous rewards in addition to a reward of 0.01 for reaching a frozen square. From figure 2, it is clear that modifications in the reward function did correspond with a higher average reward per episode. As a comparison, the paper (Gupta et al., 2021) shows learning results using the same three reward functions for instance-based learning and Q-learning methods. In their experiments, modifying the reward function produced a higher average reward only when the agent used

Q-learning methods.

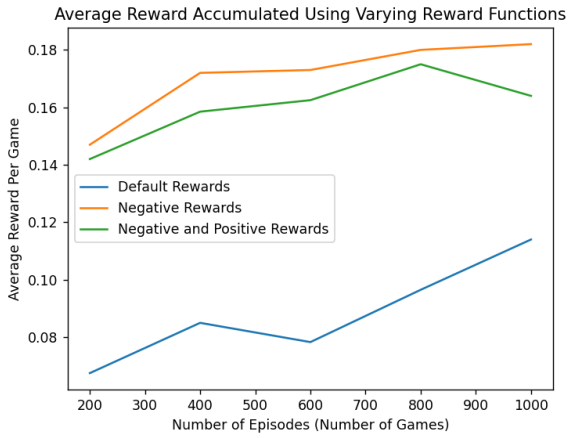

Figure 2. The average reward per episode using agents that learn according to Monte Carlo methods with varying reward functions.

3.3. Hypothesis 3

Q-learning's essence lies in adjusting Q-values using a formula that incorporates the difference between future reward estimates and current Q-values, thus learning the optimal policy over time. The third hypothesis states that integrating strategic knowledge or heuristics into Q-learning can enhance its efficiency. We found that the agent in most cases will never go up or left. We also found that an agent should never go to its previous state or try to go beyond the boundaries of the map. For example, if it is in the far-right column, it should never go right. Therefore, to use these findings we penalized the agents' action choices based on its current state by adding a negative reward for the action choice (-10) for moving up, left, to the previous state, or beyond the boundaries. Overall, the results (see figure 3) prove that this helps the agent learn faster since it does not waste its effort trying unfruitful actions, and thus increasing the average reward per episode. Comparing my enhanced temporal difference Q-learning algorithm with OpenAI's standard version reveals a significant performance advantage in my approach, thanks to the integration of heuristics ("FrozenLake$_v0Tutorial$").

3.4. Hypothesis 4

SARSA(λ) is an advanced reinforcement learning strategy that merges SARSA's temporal difference learning with eligibility traces. The experiment conducts a hyperparameter optimization search across alpha, lambda, epsilon, and gamma. The objective is to determine the optimal combination of these parameters that maximizes the agent's performance. To do this we created a predefined list of different values to test and created a quadruple nested loop that

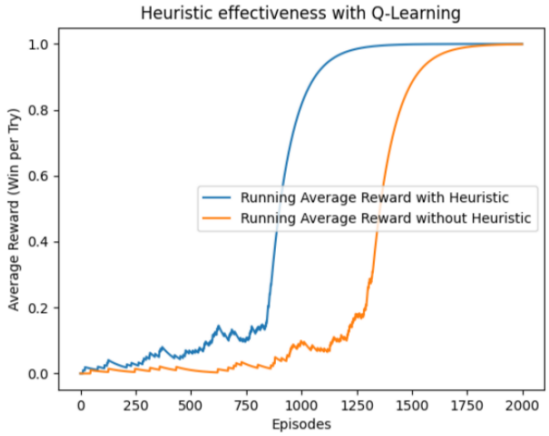

Figure 3. The average reward per episode with the use of Q-Learning. The results show the performance with and without the use of heuristics.

will work through each possible paring of values. With our predefined list we found the best values ($\alpha = 0.1$, $\lambda = 0.6$, $\epsilon = 0.05$, $\gamma = 0.9$) and worst values ($\alpha = 0.1$, $\lambda = 0.8$, $\epsilon = 0.2$, $\gamma = 0.95$). The results (see figure 4) prove that these parameters are worth optimizing. My second experiment's algorithm underperforms compared to online benchmarks (Shifei et al., 2019). Though functional, it is slower, pointing to potential inefficiencies in exploration or updates (Shifei et al., 2019).

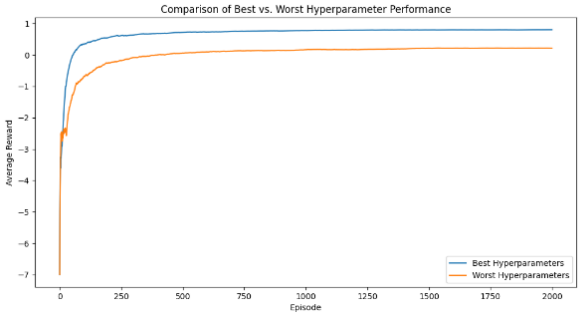

Figure 4. The average reward per episode with the use of SARSA(λ). The results show the effectiveness of choosing optimal values of alpha, lambda, epsilon, and gamma.

4. Literature Review

4.1. RL with Prediction Network

The paper (Yua et al., 2020) discusses limitations to traditional reinforcement learning methods such as heavy reliance on a reward function. The researchers propose a new algorithm, Reward Free Policy Gradient (RFPG) that utilizes a prediction network, action network, and value network to direct learning agents instead of using a reward

function. RFPG was compared to the deep deterministic strategy gradient (DDPG) algorithm in the Frozen Lake environment. RFPG outperformed DDPG the majority of the time with a higher average reward per episode. At the end of the experiment, the RFPG algorithm resulted in an average reward of 0.28 per episode, which is higher than our results using varying reward functions in figure 2.

4.2. Goal Representation with HDP

In (Ni et al., 2013), researchers propose goal representation heuristic dynamic programming (GrHDP) to solve reinforcement learning problems, such as maze navigation, by incorporating a goal network. This goal network allows the reward signal to be modified over time depending on the exact environment as opposed to remaining fixed. In the paper, the GrHDP approach was compared with HDP, Q-Learning, and SARSA(λ) using the algorithms' performance on a general maze problem. GrHDP exhibited a faster learning speed than all other methods, which indicates the importance of exploring algorithms that include an adaptive goal function.

4.3. Path Planning with RL

Robotic navigation, particularly path planning amidst static obstacles, stands as a critical challenge in the realm of automation and robotics. Utilizing MATLAB, the algorithm presented in this paper (Nair et al., 2018) is executed within a 4x4 grid setup, accompanied by a GUI that facilitates user input regarding obstacles' number, positions, and types. A comparative analysis with the traditional Dijkstra's algorithm reveals that this new method significantly reduces computational complexity, marking a notable advancement in path planning techniques.

4.4. Researching Eligibility Traces

In the realm of machine learning, particularly in the context of board games, systems have significantly evolved through self-play and temporal difference (TD) learning, as showcased by TD-Gammon and Lucas' Othello agent. This study (Thill et al., 2014) delves into the incorporation of eligibility traces—a method previously unexplored in large-scale systems, involving millions of training games and a substantial initial feature set. By comparing different versions of eligibility traces, this research highlights their impact on accelerating learning by a factor of two and improving the asymptotic playing strength.

4.5. SOM Neural Networks

This paper (Xu et al., 2017) addresses the limitations of the traditional SARSA(λ) algorithm in path planning, such as its slow environmental learning and neglect of valuable information. The method employs SOM neural networks for position mapping, leveraging position data to determine the R value for SARSA(λ) updates, thereby facilitating more accurate path planning. This hybrid approach, termed DSAE-SARSA(λ), surpasses the conventional SARSA(λ) algorithm in efficiency by avoiding prolonged iterative operations and minimizing output errors commonly associated with other neural networks.

5. Novelty

The novelty of my implementation lies in the expansion of the traditional 4x4 grid to a 9x9 grid for the Q-learning environment, significantly impacting the state space and algorithm performance. In mathematical terms, a 4x4 grid offers 16 possible states, whereas a 9x9 grid expands this to 81 possible states.

6. Conclusion

The experiments conducted tested various hypotheses on improving RL performance. Results showed that increasing the discount factor λ in dynamic programming and adjusting the reward function in Monte Carlo learning enhanced average rewards, confirming hypotheses 1 and 2. Incorporating heuristics into Q-learning and optimizing parameters for SARSA(λ) also positively impacted performance, validating hypotheses 3 and 4. These findings suggest that tailored adjustments to learning algorithms, including reward structures and heuristic integration, significantly contribute to the efficiency of RL agents. Our collaborative efforts in exploring these dimensions highlight the importance of experimentation in advancing RL methodologies.

7. Future Work

One reinforcement learning technique we did not incorporate into this project was function approximation. In the future, we would like to experiment with incremental methods such as gradient descent or linear value-function approximation. Additionally, we would like to build upon our research with the learning methods we already used. In particular, we would like to experiment with different variants of SARSA in an effort to potentially increase efficiency. Furthermore, we would like to extend the work performed in hypothesis 2 by incorporating varying reward functions into our other experiments to see if the results extend beyond Monte Carlo methods.

8. Contributions

Jaiden performed experiments corresponding to Hypotheses 1 and 2 with dynamic programming and Monte Carlo learning. Alex performed experiments corresponding to Hypotheses 3 and 4 with Q-Learning and SARSA(λ).

References

[1] D. S. Nair and P. Supriya, "Comparison of Temporal Difference Learning Algorithm and Dijkstra's Algorithm for Robotic Path Planning," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 1619-1624, doi: 10.1109/ICCONS.2018.8663020

[2] D. Xu, Y. Fang, Z. Zhang and Y. Meng, "Path Planning Method Combining Depth Learning and Sarsa Algorithm," 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2017, pp. 77-82, doi: 10.1109/ISCID.2017.145.

[3] "FrozenLake_v0 Tutorial - Gymnasium Documentation," Farama Foundation. [Online]. Available: https://gymnasium.farama.org/tutorials/training_agents/FrozenLake_tuto/.

[4] Gupta, A., Roy, P. P., & Dutt, V. (2021). Evaluation of Instance-Based Learning and Q-Learning Algorithms in Dynamic Environments. *IEEE Access*, 9(1), 138775-138790. 10.1109/ACCESS.2021.3117855

[5] M. Thill, S. Bagheri, P. Koch and W. Konen, "Temporal difference learning with eligibility traces for the game connect four," 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany, 2014, pp. 1-8, doi: 10.1109/CIG.2014.6932870.

[6] Ni, Z., He H., Wen, J., & Xu, X. (2013). Goal Representation Heuristic Dynamic Programming on Maze Navigation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(12), 2038-2050. 10.1109/TNNLS.2013.2271454

[7] Shifei, D., Xingyu, Z., Xinzheng, X., Tongfeng, S., & Weikuan J. (2019). An effective asynchronous framework for small scale reinforcement learning problems. *Applied Intelligence*, 49(1). 10.1007/s10489-019-01501-9. Available: https://www.researchgate.net/figure/Performances-of-different-algorithms-in-frozen-lake-problem_fig9_333706618.

[8] Yua, Z., Fenga Y., & Liua L. (2020). Reward-Free Reinforcement Learning Algorithm Using Prediction Network. *Frontiers of Artificial Intelligence and Applications*, 331(1), 663-670. 10.3233/FAIA200744