Using Reinforcement Learning to Solve Frozen Lake

Peter Nguyen (4033) Violet Oh (4033)

Abstract

In this report, we highlight the results of training our reinforced learning agent to solve the Frozen Lake problem. We implemented our agents using Q-learning (Violet) and dynamic programming (Peter) to find the optimal path to the terminal state. We present experiments done using these learning methods which include hypotheses, results, and analyses.

Project Domain

Frozen Lake is a single player game where the goal is to find the optimal path to the terminal state at the bottom-right of the square grid. There are parts of the grid where the agent can successfully walk on, holes that the agent should avoid (walking into one will terminate the episode with 0 reward), and a terminal state that rewards the agent with 1 point. The agent starts in position 0,0 (top-left) on a grid of specified size 'n x n' and can move in any of the four cardinal directions (up, down, left, right). This makes the action space size 4. The observation space depends on the size of the map. There are 16 possible states in a 4x4 grid and 64 possible states for a 8x8 grid. Thus, the observation space is of size n^2. There is also a parameter 'frozen' that can be set to true or false which makes the agent move in an unintended direction 2/3 of the time. For the purposes of this project, we set this parameter to false. An episode ends when the agent walks into a hole or reaches the terminal state.

Literature Review

Our first reference (1) discussed a general approach to solving optimization problems using dynamic programming. While he didn't specifically discuss the Frozen Lake or gridworld problem, his approach is still relevant to them. Similar to Bellman's general approach, we use the Bellman equation to recursively compute the optimal value function for all states. We also iteratively improve the policy by selecting actions that maximize reward using the optimal value function that we've computed before. Overall, we follow Bellman's approach to optimization problems quite closely.

Our next reference (4) used dynamic programming to solve a gridworld problem, similar to Frozen Lake. Sutton's and Barto's approach was very similar, if not the same, to ours. We followed a very similar algorithm in initialization, policy evaluation, and policy improvement. We first initialize the state value function and policy to be followed. Next, we evaluate the current policy using the state values that we calculated. We then improve the policy by selecting the action for a state that has the highest value. Finally, iterate through evaluation and improvement until we reach convergence of the state value function, where policies going forward will be optimal.

Our third reference (3) discussed performance differences between two different machine learning algorithms: instance-based learning (IBL) and Q-learning. Similar to one of the two Q-learning experiments that is discussed later in this paper, reference (3) setup their experimental stages by creating a dynamic environment of the grid world problem where an agent must navigate to a goal state while avoiding obstacles. They concluded that IBL outperforms Q-learning when the environment changes frequently, while Q-learning performs better when the changes are less frequent.

Our fourth reference (2) discussed the problem of exploration in reinforcement learning, where an agent must balance between exploiting its current knowledge and exploring new actions to learn its environment. The author introduces a new method called "Softmax Exploration" that may achieve better exploration efficiency than existing methods. The Softmax Exploration uses a probabilistic distribution to select actions, where the probability of selecting each action is based on a "temperature" parameter that controls the level of exploration. The temperature parameter gradually decreases as the agent learns more about the environment which leads to a more exploitation-focused policy. The reference concludes that Softmax Exploration can be a valuable tool for achieving efficient exploration in reinforcement learning.

DP Experiment Review

Hypotheses

We hypothesized that an environment with a larger state space will take more episodes for state values to converge

rather than an environment with a smaller state space. Furthermore, we hypothesized that decreasing γ will make the agent unable to reach the terminal state as the map size increases.

Experiments

Both experiments used dynamic programming and used grid sizes of up to 32x32. For experiment 1, we varied the size of the n x n grid, starting at size 4x4 and going up to 32x32. Using value iteration, we tracked how many episodes it took for the state values to converge as the grid size grew larger. This was done using γ=0.9.

For experiment 2, we varied the discount factor from .1 to .9 and observed the map sizes that the agent was able to find the optimal path to. The agent starts on a map of size 4x4 and it would increment up to 5x5, 6x6, ..., 32x32, or until it was unable to find the optimal path to the current grid size. For example, if the current discount factor was 0.9 and it was able to solve up to 32x32, then it would decrement γ to 0.8 and start over from 4x4 going up to 32x32 or until it could not solve the current map size, which it would then decrement to 0.7 and continue again. It was determined to be unable to solve the map if it could not make it to the terminal state within a certain amount of steps taken.

Results

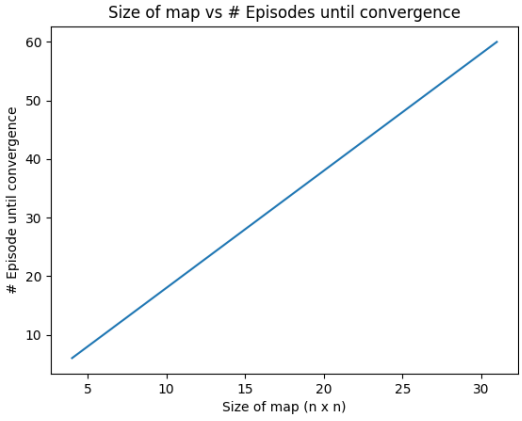

Figure 1. Graph of Size of map vs # Episodes until convergence

Figure 1 demonstrates the results of Experiment 1. As shown, the rate at which the state values converge as the size of the grid increases is linear with an upward slope. We observed that for map sizes of up to 32x32, the convergence starts at 6 episodes for a 4x4 map and increases by 2 episodes for every increment in size. For some runs of the experiment, it took 3 episodes, but this was a rare occurrence. This proves our first hypothesis of the rate of convergence of the state values being higher as the map size grows larger. A larger sized grid will take more time for the state values to converge to the true value of the state. In other words, a larger grid size will take longer for the agent to train and find the optimal policy due to the larger state space.

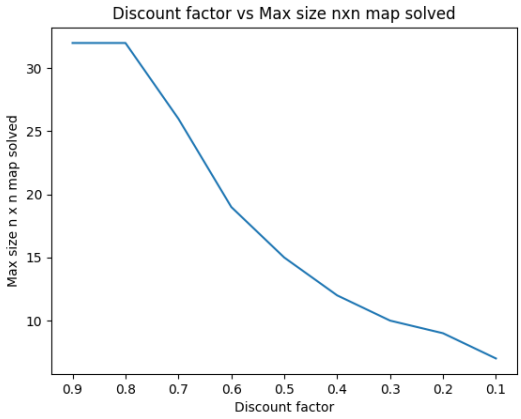

Figure 2. Graph of Discount factor vs Max size nxn map solved

From figure 2, we can observe that for a discount factor of 0.9 and 0.8, the agent was able to find the optimal path for all grids of sizes 4x4 up to 32x32. This value starts to decrease as the discount factor decreases. From γ= 0.7, we can already see a decrease from the maximum solvable map size to 25x25. Furthermore, going down to γ=0.1 makes the agent only able to find the optimal path to a 6x6 grid. Because the discount factor places weight on whether the agent should maximize immediate reward or long-term reward, this could explain the behavior of the graph. A lower discount factor prioritizes immediate rewards, but since the reward of any action that doesn't reach the terminal state is 0, the agent will get stuck with a sub-optimal policy. As we increase γ, it will focus more on long-term rewards. Since the only long-term reward is reaching the terminal state, this is what we want. We prefer a higher discount factor so that the agent will make moves that will benefit reward in the long-term, thus proving that a lower discount factor performs worse than a higher discount factor.

Q-learning Experiment Review

Hypotheses

We hypothesized that the agent will learn quicker if all the episodes have the same grid layout rather than a dynamic environment where the grid layout is randomly generated for every episode. Furthermore, we also hypothesized that the agent will learn quicker if the action space is limited to only "DOWN" and "RIGHT" rather than using all 4 actions.

Experiments

For experiment 3, we implemented two different testing environments: one for the same grid layout and the other for a randomly generated grid layout. Both experiments

used q-learning and used map sizes of 4x4, γ=0.9, and α=0.05. For each environment, we ran 1,000,000 episodes for the agent to learn the given frozen lake grid. The results are shown in Figure 3.

For experiment 4, we implemented two different action spaces for a randomly generated grid layout environment. Both experiments used q-learning and used map sizes of 4x4, γ=0.9, and α=0.05. One action space had default values (”UP”, ”DOWN”, ”RIGHT”, ”LEFT”), and the other action space had only 2 values (”UP”, ”DOWN”). For each environment, we ran 1,000,000 episodes for the agent to learn the given frozen lake grid. The results are shown in Figure 4.

Results

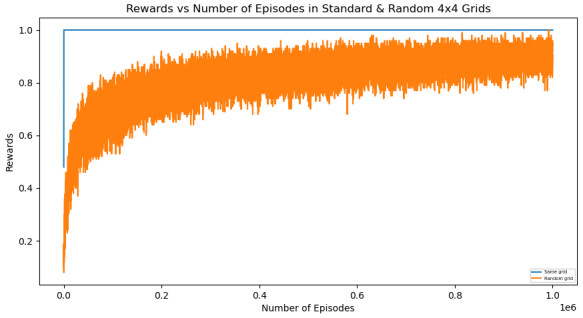

Figure 3. Rewards vs Number of Episodes in the same 4x4 Grid

Figure 3 demonstrates the results of Experiment 3. The agent gets 1 for the reward if it finishes the goal by reaching the final point, and 0 if it falls in the holes. As shown, the agent learns quicker if the frozen lake grid layout stays the same throughout all the episodes rather than the random layout for each episode. The moving average rewards for 100 episodes each reach 1 quicker if the grid layout stays the same.

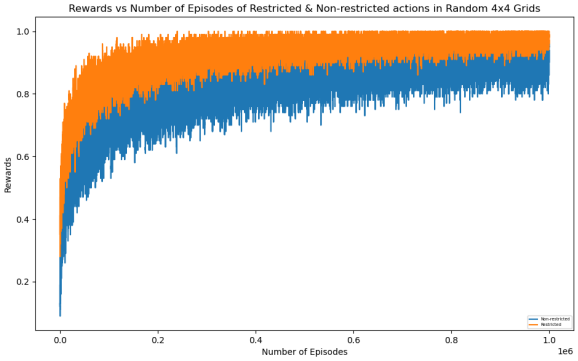

Figure 4. Rewards vs Number of Episodes in Randomly Generated 4x4 Grids

Figure 4 demonstrates the results of Experiment 4. The agent is rewarded the same way; 1 if it reaches the goal, 0 if it falls in the holes. As shown, the agent learns quicker if the action space is limited to ”DOWN” and ”RIGHT” rather than having all 4 possible actions. The two moving averages of 100 episodes are almost identical, but one with the limited action space reaches the reward of 1 quicker.

Comparison

Figure 5 represents the learning growth for three agents trained with dynamic programming, Q-learning, and SARSA respectively. We ran 300 episodes with a moving average of 10. As shown in Figure 5, the agent that is trained with dynamic programming is rewarded faster than the ones that are trained with Q-learning and SARSA. The one trained with SARSA is expected to have more spikes due to the epsilon-greedy policy.

Metrics

For the two learning methods that we have implemented, we compare their quality by comparing the rate at which the agents were able to learn and receive rewards. As shown in Figure 5, dynamic programming can be highly effective in solving problems of relatively smaller sizes. However, it assumes that the model of the environment is known, which may not always be the case in real-world scenarios. On the other hand, Q-learning is a model-free approach that doesn’t require knowledge of the environment’s dynamics. It can handle larger problems efficiently. However, as shown in Figure 5, Q-learning may be slower to converge than dynamic programming in smaller problems.

Conclusion and Future Work

We were able to successfully solve the Frozen Lake problem using Dynamic Programming and Q-learning. For Dynamic Programming, we learned that the convergence rate of the optimal policy vs the size of the map is mostly linear and that a higher discount factor is best for obtaining the optimal policy. For Q-learning, we observed that when the agent discovered the optimal action sets, convergence was significantly faster than in randomly generated grid environments where it is difficult to find optimal paths. Additionally, the agent was able to solve the Frozen Lake game more quickly with a limited action space compared to having access to all four possible actions.

In the future, we would like to extend our implementation of Q-learning to Deep Q-Networks (DQN). Moreover, we would like to factor in the ”slippery” parameter that has a probability of 1/3 for the intended moves to implement a more realistic game environment and to compare the learning rate of each algorithm.

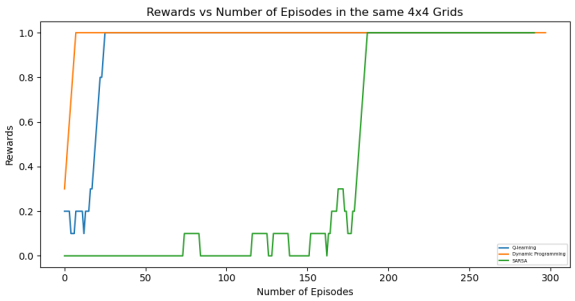

Figure 5. Graph of Rewards vs Number of Episodes in the same 4x4 Grid

References

[1] Richard Ernest Bellman. The theory of dynamic programming. 1954.

[2] Anmol Gupta, Partha Pratim Roy, and Varun Dutt. Evaluation of instance-based learning and q-learning algorithms in dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B Cybernetics)*, 42(2):404–418, 2012.

[3] John Langford. Efficient exploration in reinforcement learning. *Advances in neural information processing systems*, 20:1055–1062, 2008.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2020.