Cliff-World with Temporal-Difference Learning

CS138 HW3

Joseph Egan

October 2022

Contents

Introduction

Temporal-difference learning allows for an agent to learn directly from raw experience without a model of the environment's dynamics (similar to MC). The updating of state-action values can be updated before the completion of an episode, based on previously made estimates. In this project, we evaluated two methods for TD learning: SARSA (on-policy) and Q-Learning (off-policy). A variation of SARSA called Expected SARSA was also explored.

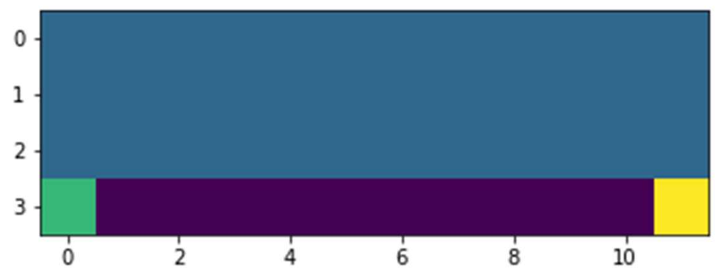
Goal

The goal of this project was to create a cliff-world environment in which agents use different TD methods to try to find an optimal pathway from the start to the end without falling into the cliff. We were able to demonstrate that SARA and Q-Learning can both find a path to the goal state, but that they take different directions due to the way that they bootstrap state-action value estimates. Finally, we test a third method called Expected SARSA and explore how changing the exploration parameter, epsilon, can affect how SARSA plots its trajectory.

Methodology

Environment Setup

The environment was a class that took grid-size inputs, and cliff-position inputs. Agents would start from the bottom left corner and try to navigate to the bottom right corner while not falling into the cliff. The grid environment would receive the agents' current position and action at each step, and return a reward and new position. If an agent took an action that would land it in the cliff zone, the reward would be -100, and the new position would be the starting point. At each step that did not land in the cliff zone, the reward would be -1. If the agent opted to move in a direction that would take it off the grid, the agent would remain in place and receive -1 reward. Therefore, the only way to maximize total reward would be to find the best way to get to the goal state without falling into the cliff.



The full-sized grid with starting point in green, goal state in yellow, and the cliff in purple

Agent Setup

Agents were a class that took the following inputs:

Epsilon	Exploration parameter of 0.1 in most cases
Alpha	Step-size parameter for q-value updates, 0.1 usually
Grid Size	The size of the grid the agent could navigate
Q-value update method	'S','Q', or 'E' for SARSA, Q-learning, Expected SARSA
Available Actions	List of actions that the agent could take. All agents could move up, down, left, or right at any time

All agents used an epsilon-greedy exploration policy to select actions. All agents used a gamma of 1 to discount rewards.

State-Action Value Estimation Methods

The first part of this project aimed to recreate example 6.6 from Barto and Sutton. Therefore, two agents were trained on the grid, a SARSA agent and a Q-learning agent. The difference between the two agents comes from the way that they update the q-values or state-action values. Both methods update the previous q-value for a state-action pair as alpha times the combination of reward received, the next state-action pair's q-value and the negative of the current q-value. However, with SARSA, the next q-value is selected using the same policy that selects actions in the environment. Therefore, SARSA is considered on-policy. Q-learning on the other hand, simply always takes the q-value of the greedy action in the next state. Since this is different from how the agent selects actions in the environment, it is considered off-policy.

$$\text{Choose } A' \text{ from } S' \text{ using policy derived from } Q \text{ (e.g., } \varepsilon\text{-greedy)}$$
$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$

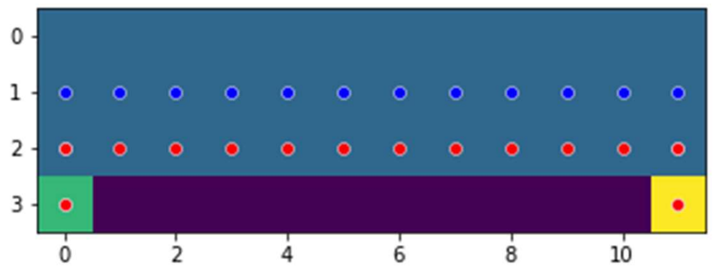
The pseudo code for a SARSA update method. Any action's q-value in the next state could be selected as long as the policy is epsilon-greedy

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$

The pseudo code for Q-learning. The 'max' component ensures that only the q-value of the greedy action in the next state is considered. All pseudo code is from Sutton and Barto

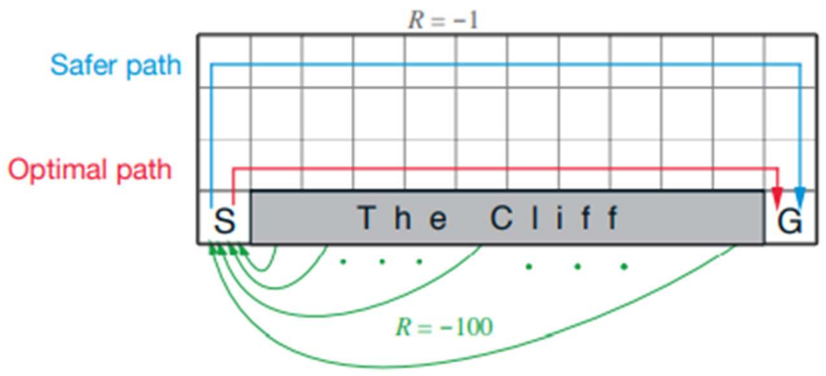
Analysis of Results

After each agent completed 5000 navigations of the grid, the optimal trajectories were plotted. Here, blue is the SARSA agent and red is the Q-learning agent.



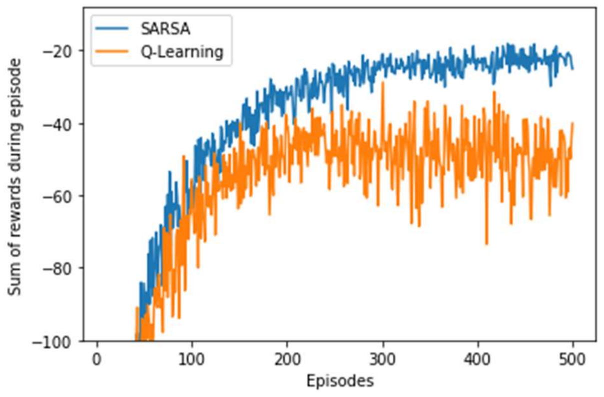
In the above, you can see that the SARSA agent avoids going along the side of the cliff, while the Q-learning agent is not afraid to go right along the edge. This is due to the fact that the Q-learning agent always assumes that the next action taken will be the greedy action (go right), and therefore also assumes the agent will never fall into the cliff. However, since the agents are all epsilon-greedy, there is always a chance (albeit a small chance) that the agent will elect to explore and fall into the cliff. Therefore, since the SARSA agent takes this uncertainty into account, it avoids running close to the edge.

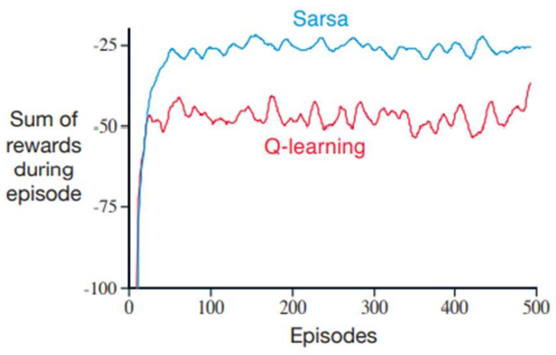
In the book, the example path for SARSA is slightly different. There, the SARSA agent hugs the top of the grid instead of going through the middle. We found that depending on the seed set, the SARSA agent would sometimes hug the top, but most of the time would be content with the middle.



Paths taken by the SARSA (blue) and Q-learning (red) agents from Barto and Sutton 6.6

We then created 100 instances of both agent types, had them all navigate the grid 500 times each, averaged their rewards at each episode, and plotted the trajectory of rewards received:





The results we achieved here were more similar to Barto and Sutton's than the trajectories. With more iterations of the agent types, the curves we generated likely would have been smoother. The takeaway from the above is that since the Q-learning agent follows the trajectory that runs along the cliff edge, it will more often fall into the cliff and receive -100 reward, while the SARA agents will be much less likely to fall into the cliff, and therefore achieve higher overall average reward.

Experimentation

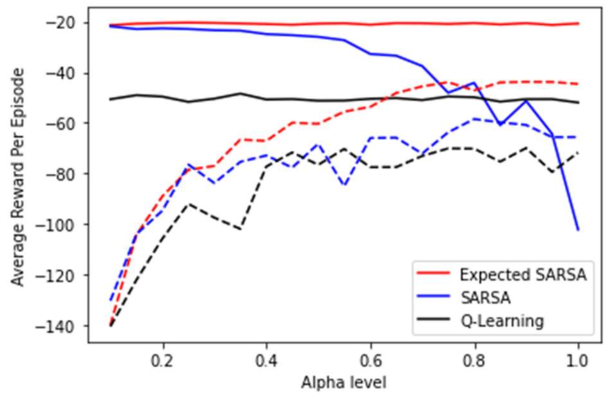
Expected SARSA

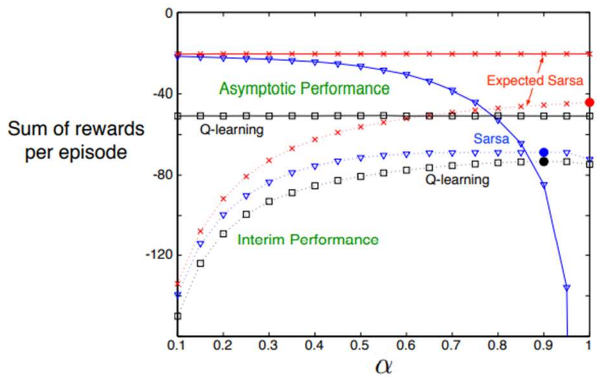
I then took the exercise one step further by testing the Expected SARSA algorithm from the next section in the chapter. The update formula for Expected SARSA is:

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\ &= Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big], \end{aligned}$$

This differs from SARSA in that the update takes the probability weighted value of the q-values for all possible actions in the next state. As Sutton and Barto state in the chapter, E-SARSA moves deterministically in the same direction as SARSA moves in expectation. Due to this change, updates made for the same state-action pair have lower variance, which allows the algorithm to perform better than SARSA given the same amount of experience.

We recreated the graph that Sutton and Barto generated that compared all three algorithms, with the exception that we used 5000 episodes total for each agent and alpha value while they used 100,000. The solid lines show the average of the total rewards received for the 101-5000th episodes for each algorithm, while the dotted lines show the same but for the 1-100th episodes. We calculated these values for 19 different values of alpha, from .1 to 1.





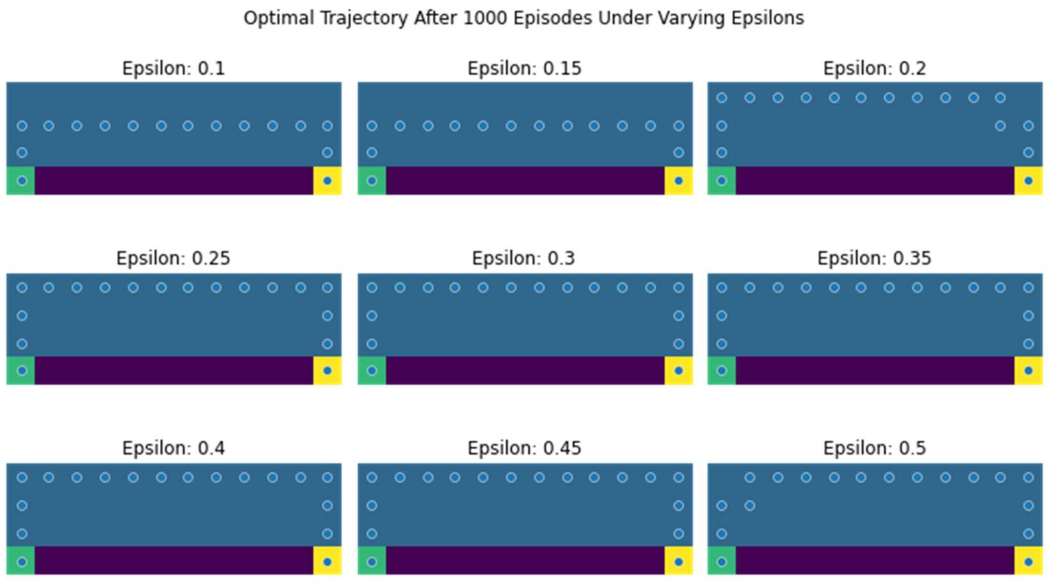
Our results (left) Barto and Sutton results (right)

Since Sutton and Barto averaged over many more episodes, their results were smoother, but our general shapes line up. In the cliff-world example, all the state transitions are deterministic, therefore Expected SARSA is agnostic to the value of alpha. However, regular SARSA can only perform well over long horizons if alpha is small. In both graphs, an alpha of 0.8 or greater leads to SARSA underperforming Q-learning.

Varying Epsilon

Based on the mismatch of SARSA trajectories between our results and the book's, we then created 9 SARSA agents with epsilons ranging from 0.1 to 0.5. Our hypothesis was that higher rates of exploration would cause the agents to travel closer to the top of the grid, and therefore further from the cliff, since with higher probability of exploring, there would be higher probability of falling into the cliff when travelling closer to it.

Each agent had 1000 episodes to navigate the grid, and these results were achieved:



Results tended to vary at lower levels of epsilon depending on the seed of the notebook when the agents navigated the grid. However, the results above are indicative of what the trajectories would typically be.

Conclusions

It was found that Q-learning and SARSA found different ways to navigate the grid without falling into the cliff. While Q-learning found the truly shortest and optimal way to get there, SARSA and E-SARSA found ways that took into account the uncertainty of the agents' action selections. If epsilon were to be tapered down to 0 for the SARSA/E-SARSA agents, their optimal paths would converge to the Q-learning path. It was also shown that E-SARSA outperforms SARSA when the state transitions are deterministic. Additionally, it was shown that increasing the exploration parameter for SARSA agents resulted in them travelling even more distance extra to avoid the danger of the cliff.

Future Improvements/Questions

The biggest improvement to this project would have been to add additional iterations for all the algorithms tested. Going from 100 to 10,000 agents for each algorithm type would have yielded smoother results curves like Sutton and Barto's. Additionally, having the time/power to run 100,000 episodes for 3 algorithms at 19 different alpha levels would have likely yielded similar results to the book in the E-SARSA section.

Bonus Questions

Exercise 8.2 Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments?

Solution:

The Dyna-Q+ agent performed better in the first phase because its exploration model encourages exploration and exploration of long stretches. Therefore, it is able to find optimal policies more quickly. So, when the environment entered the second phase, it was able to adapt more quickly to the new optimal policy and outperform Dyna-Q again.

Exercise 8.3 Careful inspection of Figure 8.5 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this?

Solution:

Because Dyna-Q+ encourages more exploring, even when the optimal policy is discovered, it will begin to lose its advantage to a policy that has found the optimal path but is exploring less. If the simulation continued without a change to the environment, Dyna-Q would eventually outpace Dyna-Q+ even though they both might have found the optimal policy.