Dylan Shadduck
EEC 289A Spring 2021

Homework 5

1. Introduction

In this homework assignment, we are asked to recreate exercise 6.6 from the textbook. The environment explored in this exercise is a gridworld example. In this gridworld, there are certain squares labeled “The Cliff” which dramatically deduct points from the agent if they reach these squares. The environment is set up as a 12 column, 4 row grid. The last row of this grid is labeled “The Cliff” except for the first and last column. The square of the last row and the first column is our starting point, and the square of the last column and last row is our goal square. Once the agent reaches the goal square the episode is over. If the agent falls off the cliff, the agent immediately returns to the starting square. Every step that the agent takes in this gridworld gives a reward of -1. If the agent falls off the cliff, the reward is -100 and when the agent reaches the terminal state, the reward is 0.

Our agent will learn to navigate this gridworld using 3 distinct, yet similar reinforcement learning strategies. The first we will explore is the SARSA on-policy reinforcement learning strategy. The other two methods are Q-learning and Expected SARSA. These other two methods are different from the first and each other, but all of them share very similar structure.

2. SARSA Method

The SARSA method for finding the optimal policy begins by starting with a given state and taking an action. For each episode, we will always start in the starting position. From here we can take only two actions since we are in the bottom left corner of our gridworld. After taking an action based on maximizing the Q function, we receive a reward for taking this action. After receiving the reward for this action, we identify the new state that we are in due to taking the

previous action. From here we use the epsilon greedy method to determine the next action from this next state. This gives us an initial state, action for that state, a reward, a next state, and a next action completing all the steps of the SARSA method. All of this information is used to update the Q function and thus help our agent take actions closer to the optimal policy.

After the agent has completed one step from the starting position we need to continue implementing this algorithm until the agent has reached the terminal state. In my program, this requires checking to make sure that the agent is only taking actions that are allowable for that state. For example, if the agent is in a state that currently lies on the first column, then I know that a step to the left would take the agent off the grid. For this reason, the function that I built to find the next action takes the information of the state in mind to ensure that a valid action can be chosen. This get_action function also implements the epsilon greedy method by taking an optimal action most of the time, but there is a chance to take a random action with a probability of epsilon.

3. Q-Learning and Expected SARSA Methods

Q-Learning and Expected SARSA differ from the SARSA method because they don't require that the next state action pair is found to update the Q function. Instead, Q-learning uses the optimal Q value for a given next state to update the Q function. This sounds very similar to SARSA, but the key difference is that an action isn't chosen using the epsilon greedy method. Here the Q-learning method always uses the optimal next state/action pair to update the current state action pair. This gives us a policy that is more aggressive than SARSA.

Expected SARSA follows a method very similar to Q-learning in that it does not choose another action by the epsilon greedy method to update the Q function. Instead, this method finds the expected value of the Q function for a given next state based on the optimal action and the

probability of choosing a random action (epsilon). This results in a policy that is actually less aggressive than SARSA and results in a reduced penalty on average over the course of 500 episodes.

4. Results

I made sure that all methods were implemented for 500 episodes and that the sum of the rewards each episode were recorded. From here we can get an idea of how well each agent is learning to move around the environment. A single trial doesn't give us the full picture of the behavior of the algorithms, so I made sure to run each 500 episode game for a total of 100 trials and average the sum of rewards for each agent. This gives a somewhat jagged picture of the expected reward for a given trial, so a moving average filter is also applied to each reward sum to smooth the graphs.

The results of each agent exploring the environment is gathered in the figure below.

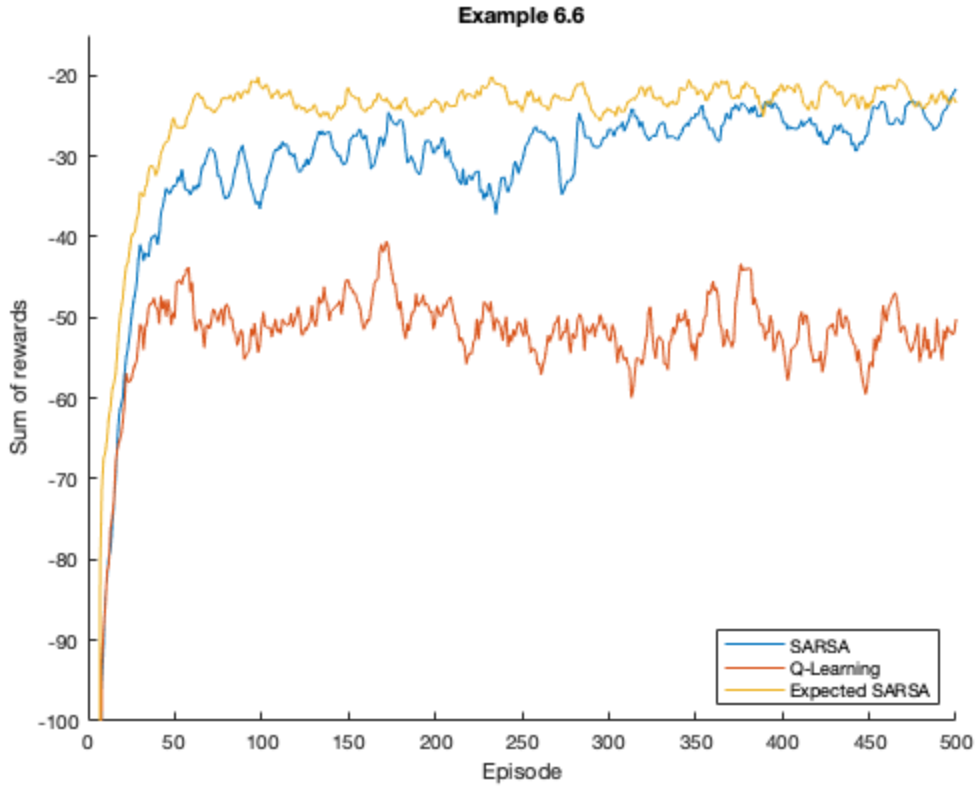

Figure 1: SARSA, Q-Learning, Expected SARSA avg sum of rewards for 500 episodes

This figure demonstrates the advantages and disadvantages of each of these three methods. Expected SARSA seems to receive the least amount of penalties on average for all of the 500 episodes. This indicates that this method is learning to take a longer, but safe route towards the terminal state so that it does not encounter the cliff at all. The SARSA method behaves very similarly to the expected SARSA, but it learns this safe behavior more slowly. This can be seen by the intersection of the Expected SARSA and SARSA methods in episode 450. Q-learning seems to have the worst performance in terms of reward, but this is likely due to the randomness of the epsilon greedy action selection. Q-learning does not take into account epsilon when updating the Q function, so it identifies the optimal policy given that the agent will always take the optimal action. For this reason the Q-learning method has a very large negative reward

because it takes the shortest route close to the cliff and will occasionally fall off the cliff due to random actions taken by the epsilon greedy action selection.

Comparing my results with those from the textbook shows an excellent agreement between the expected episode rewards for both SARSA and Q-learning agents. The textbook did not train an agent using Expected SARSA on this environment, but it is estimated that Expected SARSA will outperform SARSA due to the added computational complexity. The figure from the textbook is shown below.

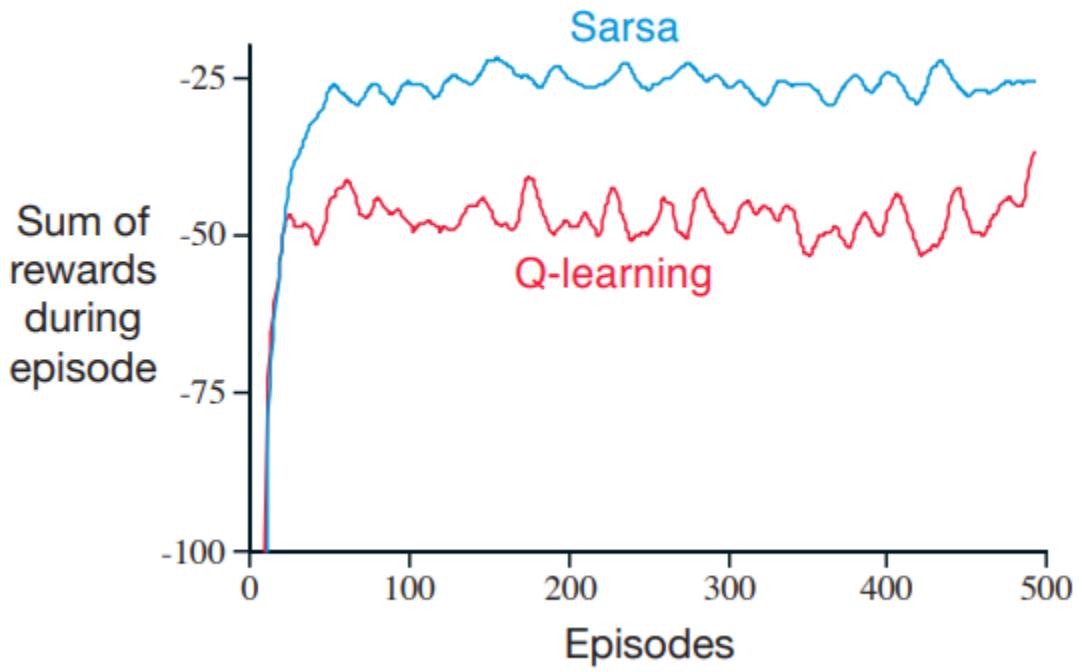

Figure 2: Textbook Sum of Rewards (SARSA and Q-Learning)