Assignment 3

Yuxiang Lu 518021911194

1 Introduction

Model-free control intends to optimize the value function of an unknown MDP. In this assignment, two model-free control methods, on-policy learning *Sarsa* and off-policy learning *Q-learning* are implemented to solve the Cliff Walking problem and the differences between them are discussed.

2 Cliff Walking

A simple Cliff Walking example is shown as the gridworld in Figure 1. It is a standard undiscounted, episodic task, with start state (S) and goal state (G). There are four usual actions causing movement up, down, right and left. Reward is -1 on all transitions except those into the region marked "The Cliff". Stepping into this region incurs a reward of -100 and sends the agent instantly back to the start.

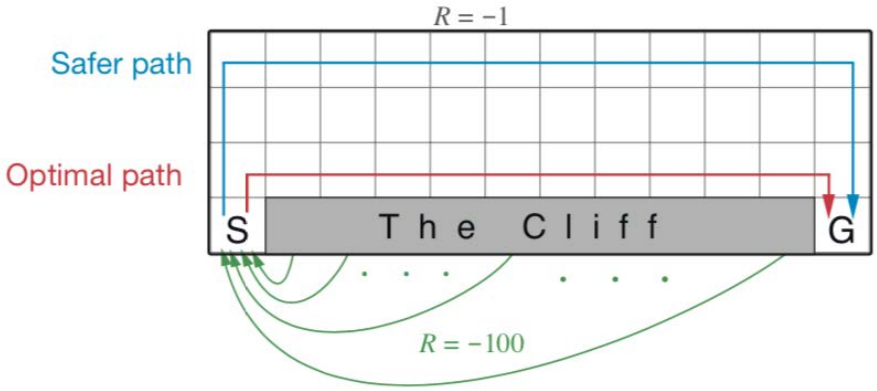


Figure 1: Cliff Walking

3 Sarsa

Sarsa is an on-policy Temporal-Difference control method. TD control intends to learn an action-value function rather than a state-value function in TD learning for prediction. Specifically, it estimates $q_\pi(s, a)$ for the current policy π and for all states s and actions a. This can be done by considering transitions from state-action pair (s,a) to state-action pair (s',a'), formulated as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

This update is done for every transition from a nonterminal state S_t. If S_{t+1} is terminal, then $Q(S_{t+1}, A_{t+1})$ is defined as zero. This rule uses every element of

the quintuple of events, $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, that make up a transition from one state-action pair to the next. And the name of this algorithm *Sarsa* comes from this quintuple.

For "on-policy", it means that it learns about policy π from experience sampled from π. The action value q_π is estimated using ϵ-greedy exploration in each step, and at the same time policy π is changed also with the ϵ-greedy method. The complete algorithm is shown in Algorithm 1.

Algorithm 1: Sarsa

Input: step size $\alpha \in (0, 1], \epsilon \in [0, 1]$
Output: Q

Loop for each episode:
  Initialize S
  Choose A from S using ϵ-greedy policy
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using ϵ-greedy policy
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
    $S \leftarrow S'; A \leftarrow A';$
  until S is terminal

4 Q-learning

Q-learning is an off-policy Temporal-Difference control method. Different from Sarsa, the "off-policy" means to learn about the target policy π from experience sample from the behavior policy μ. The target policy π is greedy, and the behavior policy is ϵ-greedy in the implementation. Therefore, it can learn about the optimal policy while behaving non-optimally to explore all actions. Q-learning is formulated as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

The update of action-value function Q towards maximizing over all those actions possible in the next state, and finally Q converges to q_*, the optimal action-value function and π converges to the optimal policy. As the agent explores the environment with the ϵ-greedy, it is sure to experience enough new states. The complete algorithm of Q-learning is shown in Algorithm 2.

Algorithm 2: Q-learning

Input: step size $\alpha \in (0, 1], \epsilon \in [0, 1]$
Output: Q

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using ϵ-greedy policy
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until S is terminal

5 Experiment and Discussion

In the experiment, the number of episodes in a single run is set as 500, and the reward of each episode is averaged over 2000 runs to reduce randomness. Firstly, we choose $\alpha = 0.5$ and $\epsilon = 0.1$ for the hyper-parameters. The result is shown in Figure 2, and the paths find by Sarsa and Q-learning are displayed in Figure 3.

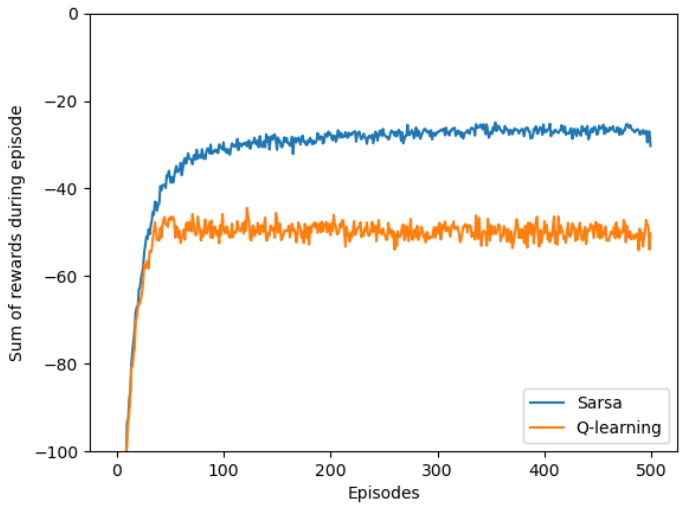


Figure 2: Sum of rewards for $\alpha = 0.5, \epsilon = 0.1$

We can see that Sarsa finds a safer path through the upper part of the gridworld while Q-learning finds the optimal path which travels right along the edge of the cliff. As Q-learning uses greedy selection in policy improvement, it learns values for the optimal policy. However, this path may occasionally fall off the cliff because of the ϵ-greedy method in policy evaluation, which makes the reward from the optimal path smaller than the safer path. As Sarsa applies ϵ-greedy action selection in both policy evaluation and policy improvement, it learns a longer but safer path with better reward. The sum of reward learned from Sarsa is about -30, and from Q-learning is around -50.

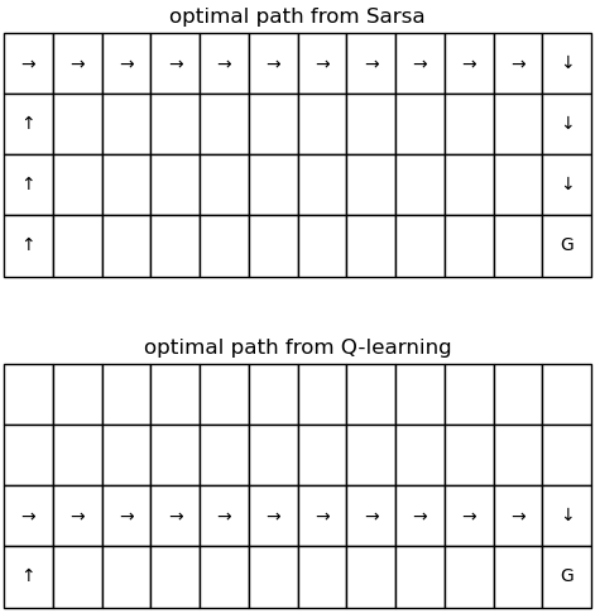
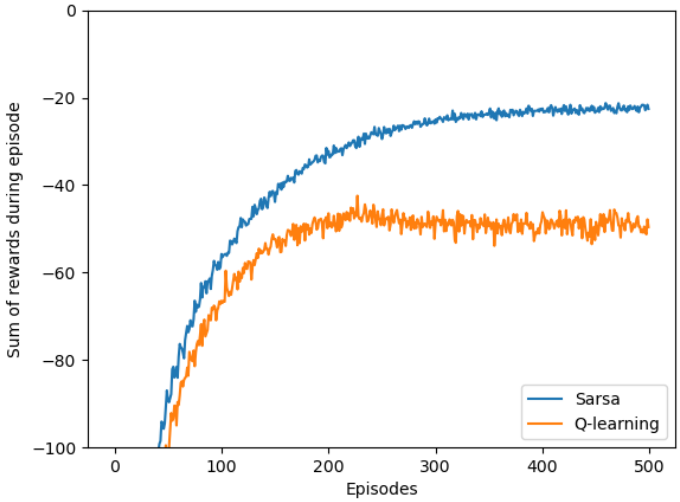


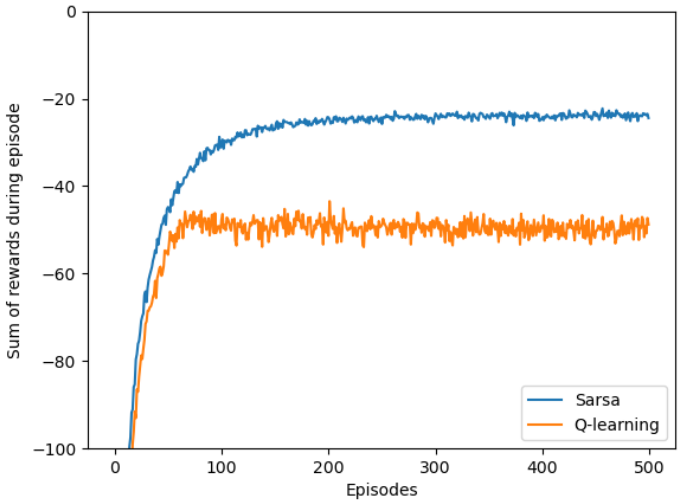
Figure 3: Optimal path from Sarsa and Q-learning

Then we experiment on the hyper-parameter α, which represents the step size in the update of action-value Q. The results of $\alpha = 0.1$ and $\alpha = 0.3$ is shown in Figure 4.

Compared with Figure 2, we can see that with a larger step size α, the algorithm can converge faster. Considering Q-learning, when α equals 0.1, it takes around 250 episodes; when α decreases to 0.3, it converges at about the 100th episode; when α equals 0.5, it needs only 60 to 70 episodes.



(a) $\alpha = 0.1, \epsilon = 0.1$



(b) $\alpha = 0.3, \epsilon = 0.1$

Figure 4: Sum of rewards for different α

Another hyper-parameter to experiment is ϵ in ϵ-greedy action selection. Specially, if ϵ equals 0, ϵ-greedy method will become normally greedy method. The result is shown in Figure 5.

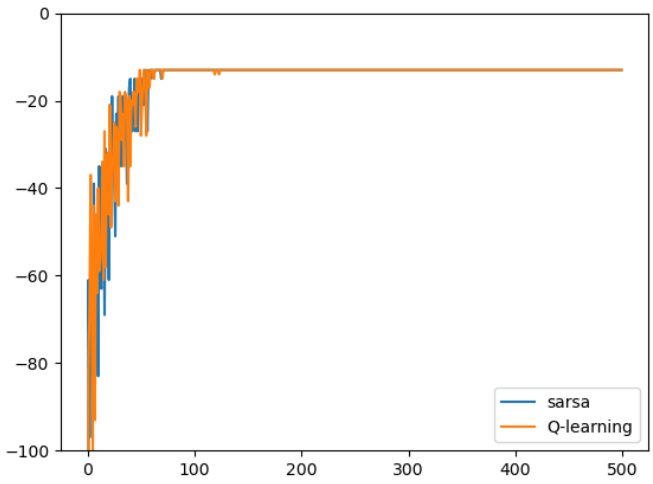


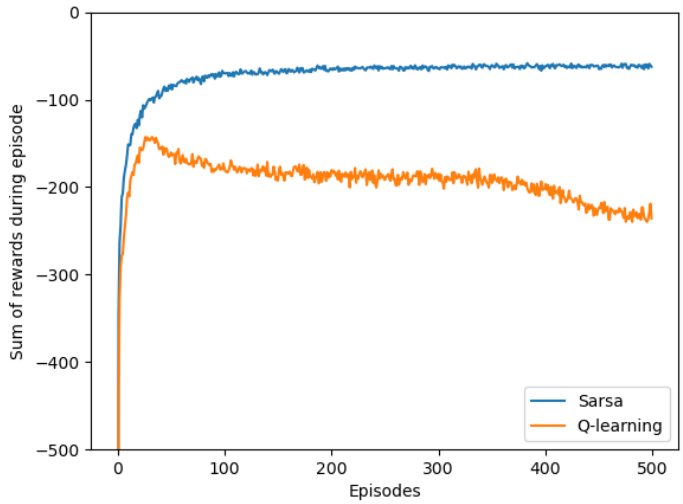
Figure 5: Sum of rewards for $\epsilon = 0, \alpha = 0.5$

We can see that the sum of rewards learned by Sarsa and Q-learning are roughly the same, as two algorithms are actually the same when ϵ equals zero. Both methods converge to the optimal path in Figure 1. We can also note that, because there is no randomness in the greedy selection, there is no jitter in reward curve after the algorithms converge. The final reward is a definite value, and it is larger than that when $\epsilon > 0$.

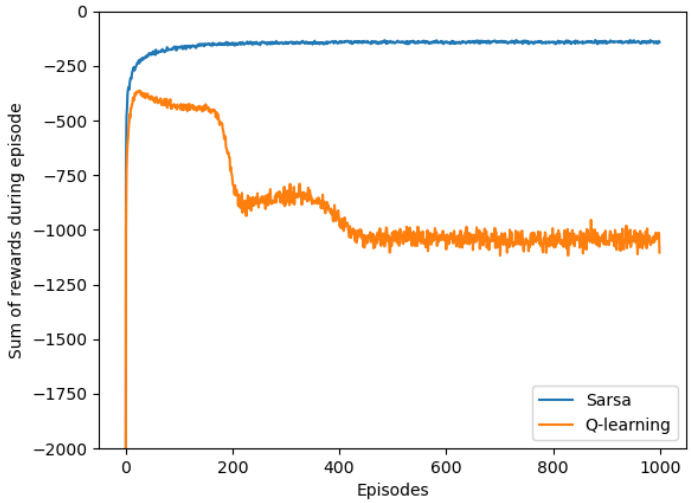
Then we look at larger ϵ, the results of $\epsilon = 0.3$ and $\epsilon = 0.5$ are shown in Figure 6, and pay attention to the value on vertical axis.

We can see that a larger ϵ causes the reward to decrease, especially in Q-learning, as it is more likely to fall off the cliff in the optimal path. The paths learned by two method are still the same as those in Figure 3. We can also see that Q-learning converges much slower because of greater uncertainty.

We can conclude that Q-learning always learns the optimal policy as it only takes the maximum action value into account, but with larger possibility to fall off the cliff, the reward will be much lower than Sarsa.



(a) $\epsilon = 0.3, \alpha = 0.5$



(b) $\epsilon = 0.5, \alpha = 0.5$

Figure 6: Sum of rewards for different ϵ