EE551000 System Theory

Homework 2 Report: Temporal-Difference Learning

周柏宇 105061110

Implementation

✓ Briefly describe your implementation.

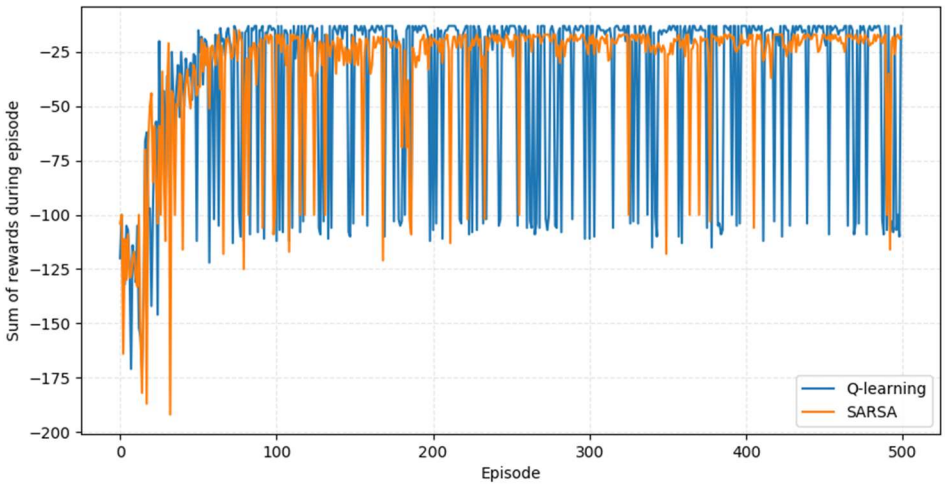
For SARSA, we require the quintuple (S, A, R, S', A') to update our action-values. We first initialize the S and A, then the R and S' are obtained from the environment. A' is determined with our ε-greedy policy just like A. Now we can update as follows: $Q(S,A) = Q(S,A) + \alpha[Target - Q(S,A)]$, where $Target = R + \gamma Q(S', A')$.

For Q-learning, we need the quadruple (S, A, R, S'). With the initial state, we can determine A with our ε-greedy policy and R, S' from the environment. To update the action-value, we use $Q(S,A) = Q(S,A) + \alpha[Target - Q(S,A)]$, where $Target = R + \gamma \max_a Q(S', a)$.

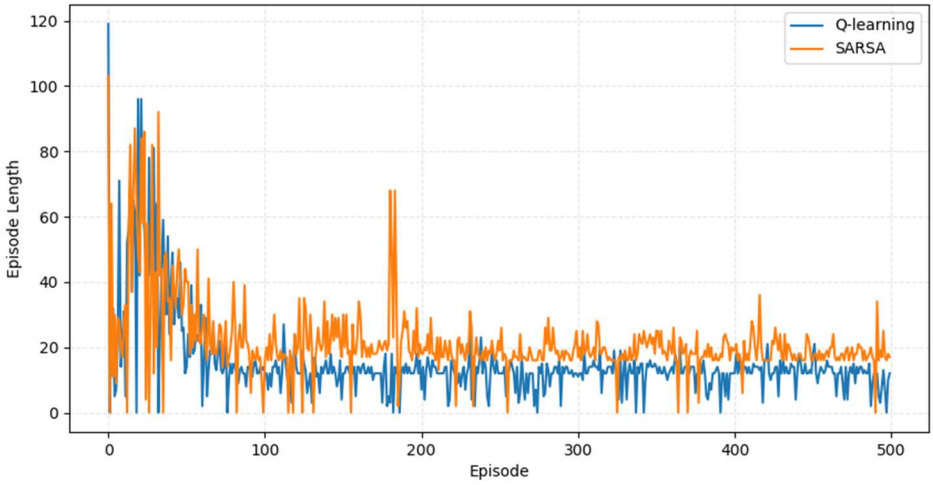
For both algorithms, we need to train them with sufficient number of episodes. In my experiment, 500 episodes are enough for our algorithms to converge.

Experiments and Analysis

✓ Plot curves of different methods into a figure. (As example above)

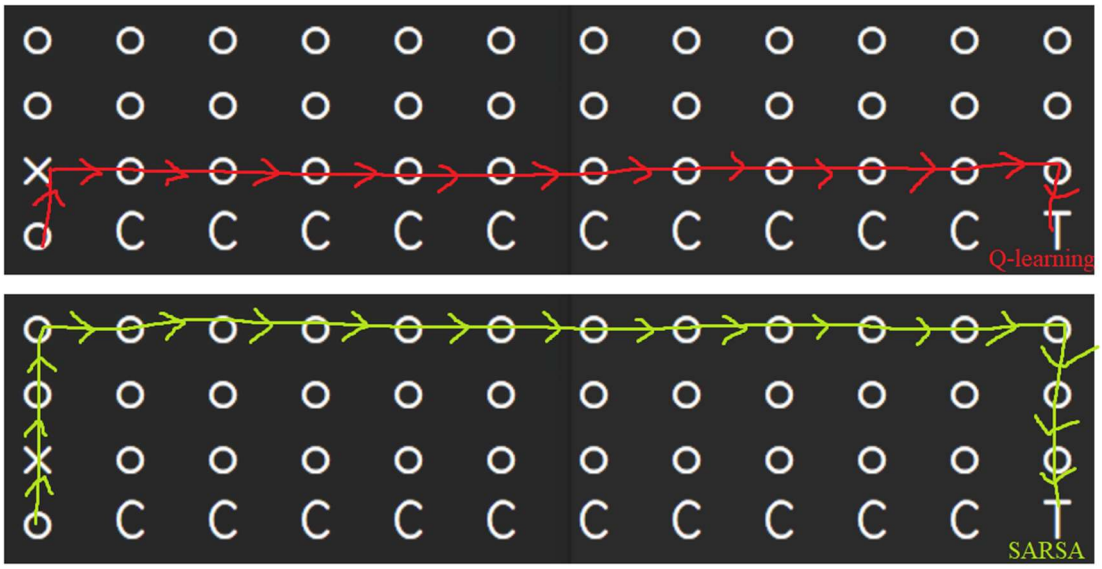


✓ Plot the episode length (time steps taken per episode) v.s. episode. What do you observe?



Overall, the performance for Q-learning and SARSA has stabilized with respect to their own policy. We will elaborate the different results from two algorithms in the following questions. Furthermore, Q-learning is more likely to have an episode length less than its converging value because sometimes instead of reaching the destination, it falls off the clip and the episode ends immediately.

✓ Render and show the trajectory of each method. What do you observe?



Q-learning found the optimal path to the destination, while SARSA found the safest path to the destination.

✓ Observe the reward curve of each algorithm. We can observe that the reward curve of SRASA is more stable than Q-learning (less severe drop to -100). Please explain.

It is due to the fact that SARSA optimizes according to the ε-greedy policy,

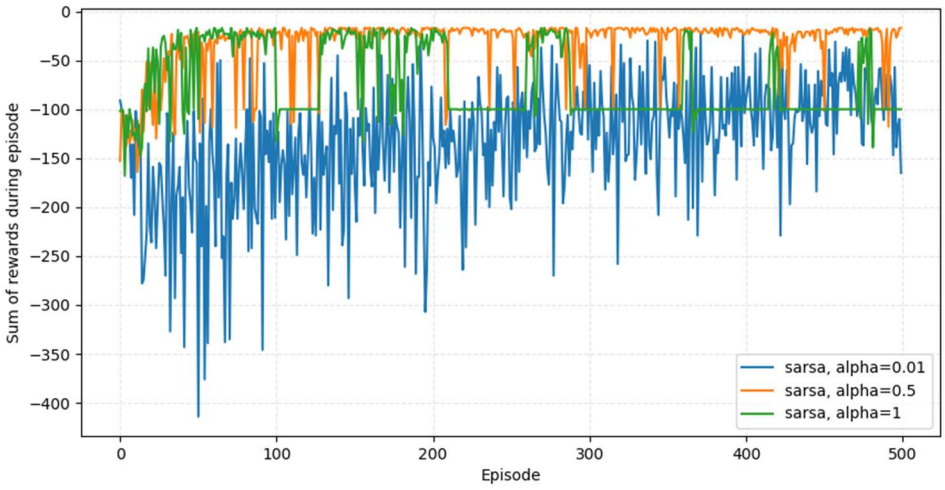
which contain an amount of uncertainty while moving. During the training sessions, if the agent moves along the shortest path, it would have a chance of falling off the clip due to the exploration steps, thus getting a -100 penalty. SARSA is aware of the risk. It realizes that the penalty is so large and it cannot guarantee it won't happened, thus choosing an "optimal" path with respect to the ε-greedy policy. As illustrated, the chance that SARSA dropping to -100 is relatively smaller than that of Q-learning, because it takes the safest path toward the destination. On the contrary, Q-learning learns the optimal policy regardless of what policy it uses to generate data, we can observe the figure above to see it actually found the shortest path to destination. But we are using ε-greedy policy to record the reward, the exploration steps give rise to the unstable performance of Q-learning.

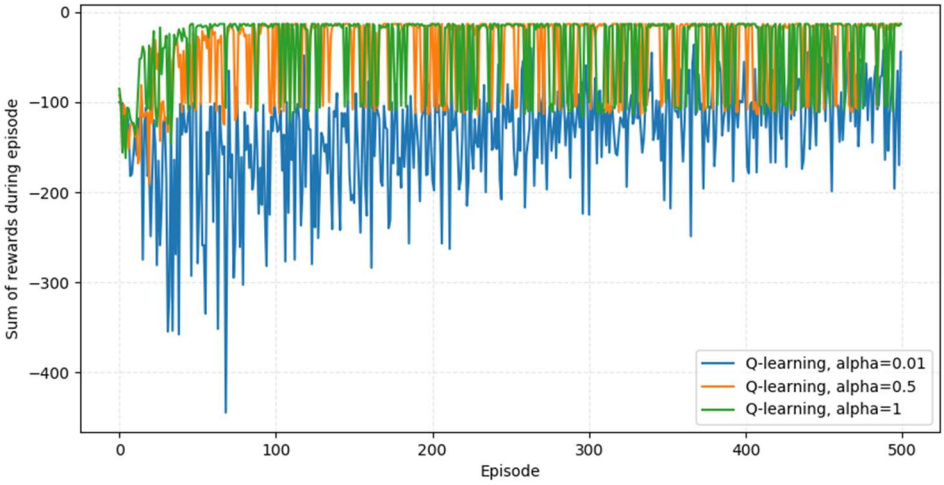
✓ Why is Q-learning considered an off-policy control method? How about SARSA?

Take our case for example, for both Q-learning and SARSA, we use ε-greedy policy to generate data. The difference is their update rule. For SARSA, it updates the action-values using the target $R + \gamma Q(S', A')$, which is the exact experience generating from the ε-greedy policy. As a result, SARSA is an on-policy algorithm because it optimizes the policy that it uses to generate data or explore.

For Q-learning, it updates the action-values using the target $R + \gamma \max_a Q(S', a)$. It doesn't requires the quintuple (S,A,R,S',A'); it only needs (S,A,R,S') because Q-learning uses greedy policy to determine A'. We can see that the policy which is used to generate data and we try to optimize is different. Therefore we call Q-learning an off-policy algorithm.

✓ Vary the TD learning rate α, what happens?





We can see that in smaller step size e.g. alpha=0.01, the performance has quite a large variance and the learning curve seem to grow slowly, while with larger step size e.g. alpha=1, the performance has a way smaller variance and is seem to perform optimally from time to time. With this illustration, we can conclude that the step size is a hyper-parameter that we have to experiment and tune. Solely increasing or decreasing the step size doesn't guarantee the improvement the results.