COMP532 - 2019 - ML Assignment 1

Reinforcement Learning

YUEFENG LIANG 201350099

&

RUI ZHANG 201255800

1. Problem 1 (12 marks)

Re-implement (e.g. in Matlab or Python) the results presented in Figure 2.2 of the Sutton & Barto book comparing a greedy method with two ε-greedy methods (ε=0.01 and ε=0.1), on the 10-armed testbed, and present your code and results. Include a discussion of the exploration-exploitation dilemma in relation to your findings.

Solution::

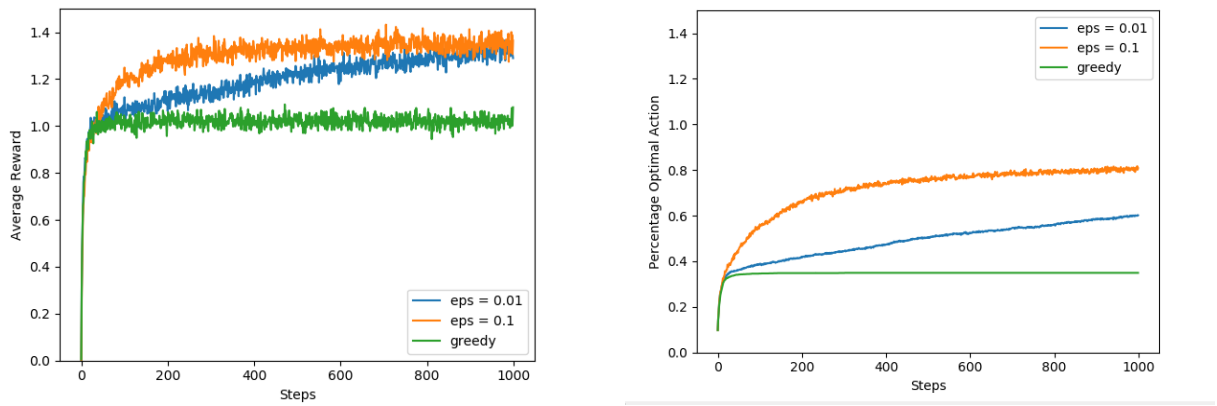


Figure 1. The result of comparing greedy method and two ε-greedy methods

Discussion:

The two graphs compared a greedy method with two ε-greedy methods (ε = 0.01 and ε = 0.1) on the 10-armed bandit test. Both three methods improved slightly faster at the beginning, but then the greedy method stop increase and stay in the stable lower level, and it also seems to perform significantly worse in the long run.

It can be seen that the two ε-greedy methods eventually performed better, because they continued to explore and to improve their chances of recognizing the best action. The ε = 0.1 method explored more, and usually found the optimal action earlier, but it never selected that action more that 90% of the time. The ε = 0.01 method improved more slowly, but eventually it seen would perform better than the ε = 0.1 method. Thus, it is better to explore appropriately than only exploiting and the best long-term strategy may involve short-term sacrifices.

2. Problem 2 (8 marks)

Consider a Markov Decision Process (MDP) with states S = {4,3,2,1,0}, where 4 is the starting state. In states $k \geq 1$ you can walk (W) and $T(k, W, k-1) = 1$. In states $k \geq 2$ you can also jump (J) and $T(k,J, k-2)$ = 3/4 and $T(k,J, k)$ = 1/4. State 0 is a terminal state. The reward $R(s, a, s') = (s - s')$ for all (s, a, s'). Use a discount of γ = 1⁄2. Compute both $V*$ (2) and $Q*$ (3, J). Clearly show how you computed these values.

(a) Compute the value function from state 0 to state 2.

Solution:

$V*$ (0) = 0

$V*$ (1) =1+γ$V*$ (0) =1

$V*$ (2) = max {1+$\gamma V*$ (1), $\frac{3}{4}$(4+ $\gamma V*$ (0)) + $\frac{1}{4}\gamma V*$ (2)}

= max { $\frac{3}{2}$, 3 + $\frac{1}{4}$ γ$V*$ (2)}

=max {$\frac{3}{2}$,$3\frac{8}{7}$} = $\frac{24}{7}$

Where $3\frac{8}{7}$ comes from supposing that if 3 + $\frac{1}{4}$ γ$V*$ (2) were the maximum, then

$V*$ (2) = 3 + 1/4 γ$V*$ (2)

$V*$ (2) = $3\frac{8}{7}$

(b) compute $Q*$ (3, J)?

Solution:

$$V*(3) = \frac{3}{4}\left(4 + \frac{1}{2}V*(1)\right) + \frac{1}{4}\frac{1}{2}V*(3)$$

$$= \frac{27}{8} + \frac{1}{8}\ V*(3)$$

$$= \frac{27}{7}$$

$$Q*(3, J) = \left(\frac{3}{4} + \frac{1}{2}V*(1)\right) + \left(\frac{1}{4} + \frac{1}{2}V*(3)\right)$$
$$= \frac{24}{7}$$

3. Problem 3 (5 marks)

a) What does the Q-learning update rule look like in the case of a stateless or 1-state problem? Clarify your answer. (2 marks)

b) Discuss the main challenges that arise when moving from single- to multi-agent learning, in terms of the learning target and convergence. (3 marks)

Solution:

a) Since Q-learning update rule is off-policy, the learned action-value function directly approximates the optimal action-value function, independent of the policy being followed.

b) When moving from single- to multi-agent learning, the theoretical properties of single agent adaptive learning are lost. Because information concerning other agents is generally missing, multi-agent systems do not have the Markov property, they need to learn to find optimal solutions by acting autonomously, rather than following the policy. What's more, multi-agent system can be regard as dynamic environment, thus the convergence theories for single-agent learning, such as Bellman equation, are not suitable anymore.

4. Problem 4 (15 marks)

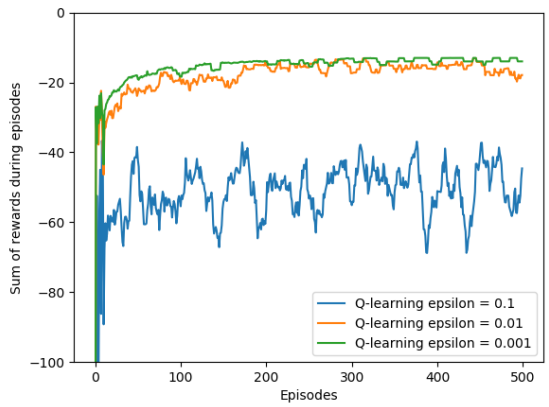
Re-implement (e.g. in Matlab or Python) the results presented in Figure 6.4 of the Sutton & Barto book comparing SARSA and Q-learning in the cliff-walking task. Investigate the effect of choosing different values for the exploration parameter for both methods. Present your code and results. In your discussion clearly describe the main difference between SARSA and Q-learning in relation to your findings.

Note: For this problem, use α=0.1 and γ=1 for both algorithms. The "smoothing" that is

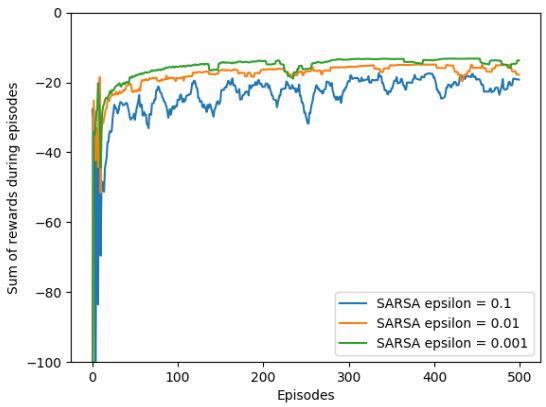
mentioned in the caption of Figure 6.4 is a result of 1) averaging over 10 runs, and 2) plotting a moving average over the last 10 episodes.

Solution:

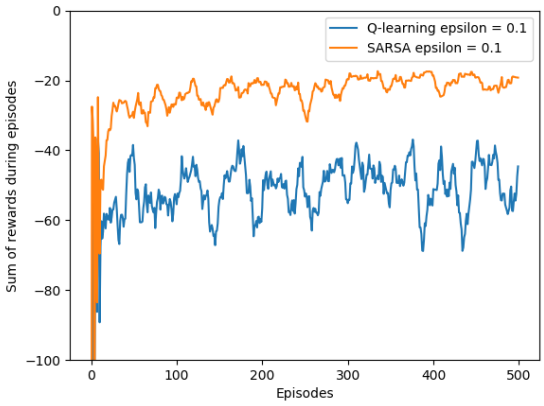
The results of cliff walking by using Q-learning and SARSA with different values of ε (exploration parameter) are shown in figure 2. As for program code, please find attached python file. From figure 2(a) and 2(b), we can see that with reduction of ε, both the sum of rewards by using Q-learning and SARSA methods are going close to optimal value. Figure 2(c) indicates that when learning algorithm choose a greater value of ε, SARSA has a better performance than Q-learning. The reason is because SARSA takes action in account and tends to choose a longer but safer path, while Q-learning prefer to choose optimal path (according to optimal policy) which meanwhile has risk of falling off the cliff (ε-greedy action selection). However, when ε has a lower value, the results of Q-learning and SARSA are almost same, and both close to optimal value, as shown in figure (2d).



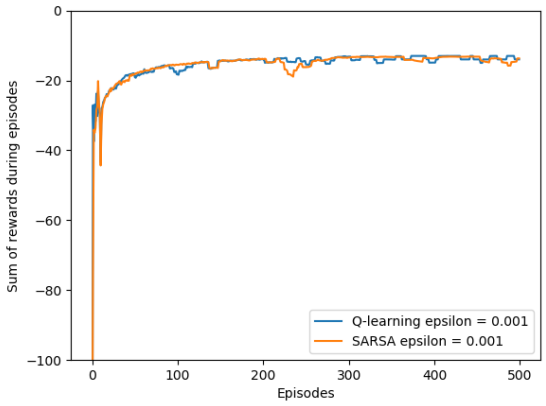
(a)



(b)



(c)



(d)

Figure 2. The results of cliff walking by using Q-learning and SARSA with different ε values