Homework4

Yi Zhou

November. 9th 2021

Introduction

Question description

Exercise 8.4 (programming) The exploration bonus described above actually changes the estimated values of states and actions. Is this necessary? Suppose the bonus $\kappa\sqrt{t}$ was used not in updates, but solely in action selection. That is, suppose the action selected was always that for which Q(S_t, a) + $\kappa\sqrt{t(S_t, a)}$ was maximal. Carry out a gridworld experiment that tests and illustrates the strengths and weaknesses of this alternate approach.

Goal

The purpose for this assignment is to implement a RL program with planning and learning with tabular methods and see the difference between different planning methods. Method including DynaQ, DyanQ+ and alternative DynaQ+.

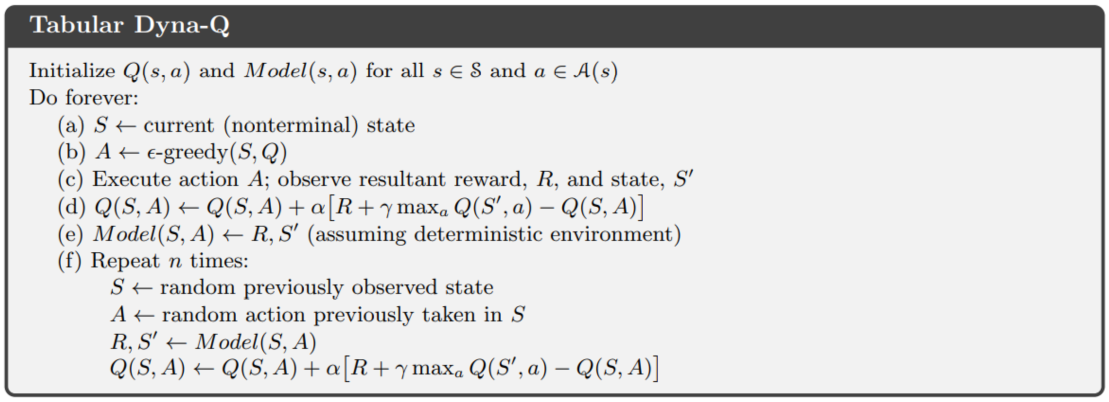
Solution Summery

The main difference happens in where the bonus is applied. In DynaQ there is no such bonus. In DyanQ+, the bonus $\kappa\sqrt{t}$ is added in the q value updates, and in the alternative DynaQ+, it is added in finding the best action. In the game environment used in this homework, the maze will have a barrier in between. Therefore, to reach to goal, the agent needs to find a way that can bypass the barrier. To find this way quicker, seems like the bonus will make a positive difference. Also, since the map will change once to make the previous successful path fail, so the agent needs to learn the mistake and find a new way, which is also a challenge for the three method. And this homework focus on how helpful this bonus is in the maze game.

Methodology

DynaQ algorithm

DynaQ algorithm is based on the TD method (depending on n: if n is 0, then it is TD(0); if n is full episode, then it is MC algorithm) with extra planning and modeling steps, which using simulated model and update Q value with simulated result instead the real experience.



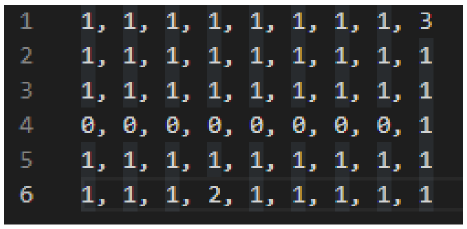
```
Tabular Dyna-Q
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
    (a) $S \leftarrow$ current (nonterminal) state
    (b) $A \leftarrow \epsilon$-greedy$(S,Q)$
    (c) Execute action $A$; observe resultant reward, $R$, and state, $S'$
    (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
    (f) Repeat $n$ times:
            $S \leftarrow$ random previously observed state
            $A \leftarrow$ random action previously taken in $S$
            $R, S' \leftarrow Model(S,A)$
            $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
```

Figure 1: Tabular DynaQ algorithm

Implementation

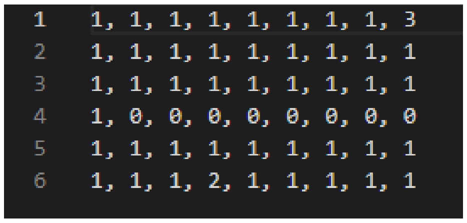
- First is to build the environment of maze map that is similar to the homework2. I used csv file to generate the simplified maps with reward. Every maze will have a start point defined as number 2, a barrier defined as number 0, the final Goal state defined as number 3 and the rest is defined as 1. The agent will start at the starting point and try to bypass the barrier and reach the goal. And this is similar to the Blocking Maze example shown in the test book.

- Second is that the map will change once as the step goes after 1000 so that the previous path will not be available. In this case, I just make the small change in number so that the previous path fail.



```
1, 1, 1, 1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1
0, 0, 0, 0, 0, 0, 0, 0, 1
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 2, 1, 1, 1, 1, 1
```

Figure 2: The first map



```
1, 1, 1, 1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 0, 0, 0, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 2, 1, 1, 1, 1, 1
```

Figure 3: The first map after change

- Then, the environment has the function to interact with the maze, it can make an action from up, down, left and right each time, and compute the reward and new state after an action is applied. The reward will always be 0 until it reaches the goal state.
- The last are the functions for the DynaQ algorithm

Result

Here presents the plots of cumulative reward and steps:

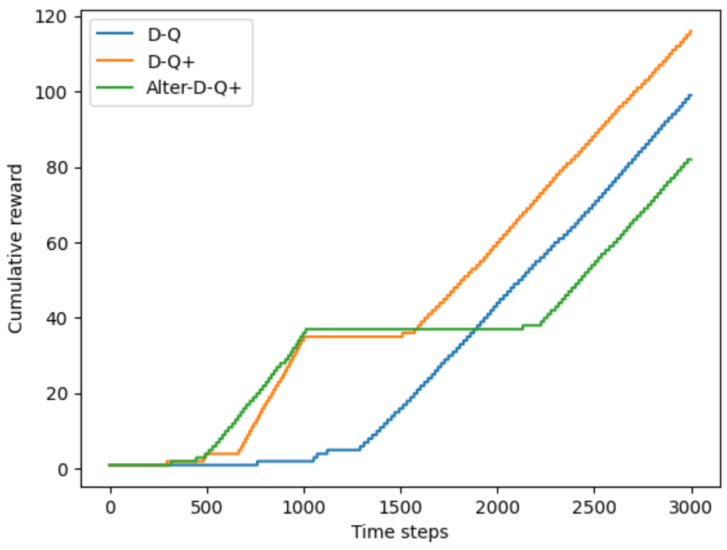


Figure 4: Result of first trial

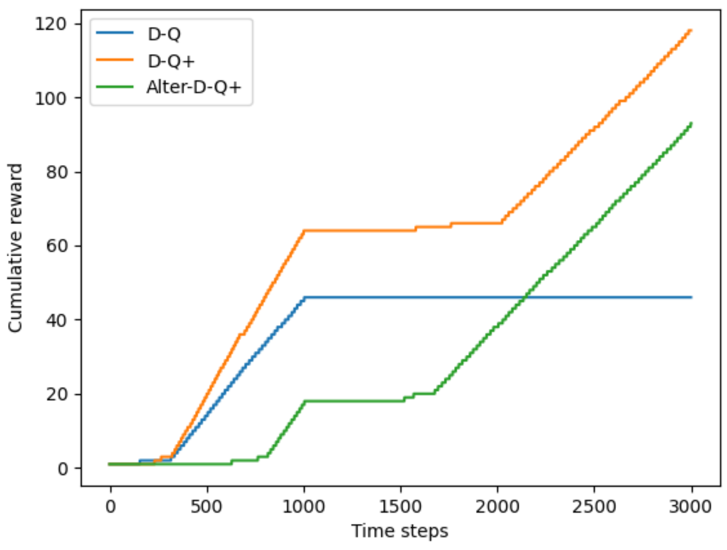


Figure 5: Result of second trial

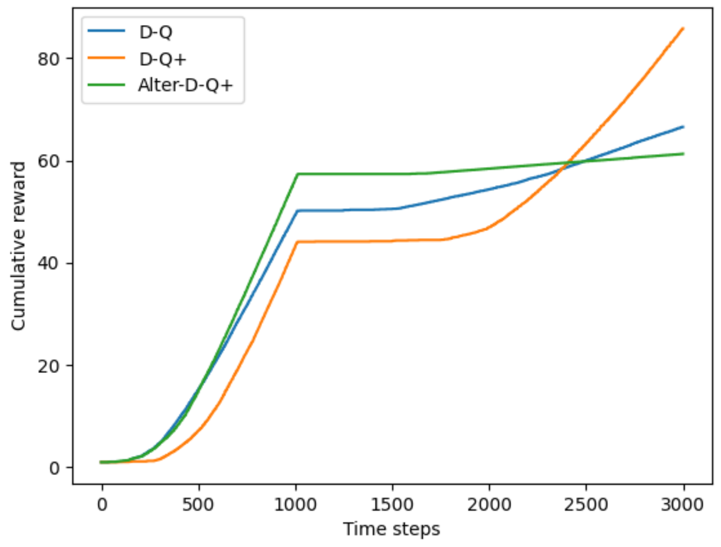


Figure 6: Result of 20 trials in average

Conclusion

From the three plots shown above, it is not hard to realize that even if the result varies in different cases, the alternative DynaQ+ learns better at the beginning but D-Q+ is still the best in the run, and DynaQ is in between.

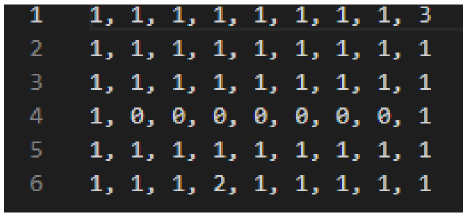
My hypothesis for this result is that, bonus added in the action will let agent try the in-visited one instantly, but since Q value is not updated by the bonus, agent can't see in long run that which action has the best potential efficiency. Because the model is not planned with the extra bonus

In result, DynaQ can even perform better because model simulates the real experience, but in alternative DynaQ+, action is chosen by the different Q value than the model. I think this makes model makes less difference in long run as the bonus getting more influential.

Interestingly, DynaQ+ doesn't necessarily perform better then DynaQ. Probably because the checking the in-visit one is not equivalent to find the best way. Or I think it is because I used alpha = 0.7 which is too large.

Furthermore

I also tried a different maze map which gives another possible path when map changes.



```
1, 1, 1, 1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 0, 0, 0, 0, 0, 0, 0, 1
1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 2, 1, 1, 1, 1, 1
```

Figure 7: Two paths map

The result is shown below:

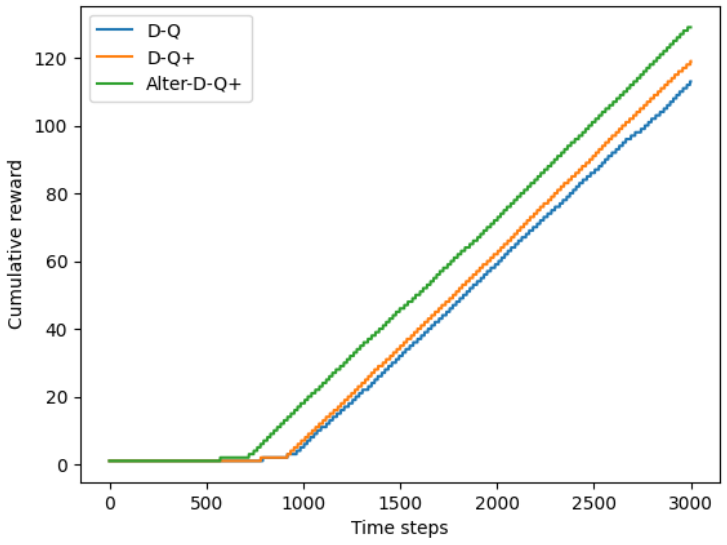


Figure 8: Result of one trial

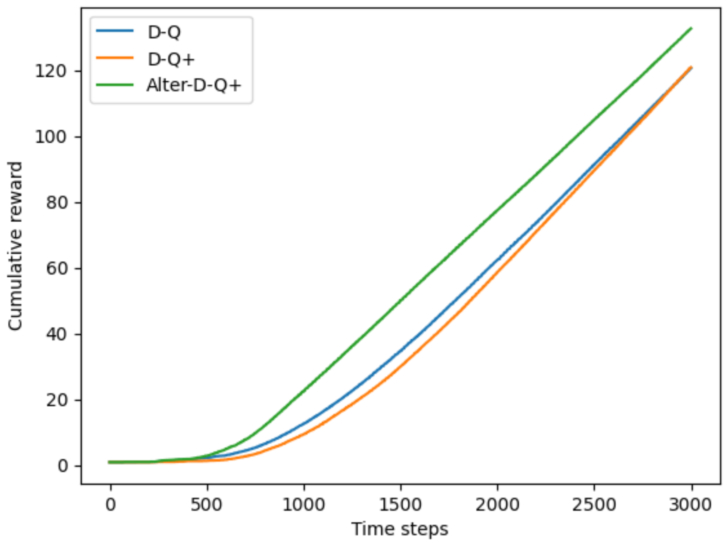


Figure 9: Result of 20 trials in average

Conclusion

Different from the previous result, this one does not match my hypothesis. I think the reason is that, since the new map does not fail the previous success path then exploring on the in-visited state action will not make a significant difference. However, in the previous case, agent has to know the mistake and find a new way out, but in this one, stay with the old way is still beneficial. Therefore, DynaQ+ has no obvious advantage as well compared to DynaQ since finding new way is not necessary.

Furthermore and more

What if the bonus is both added to the action and model?

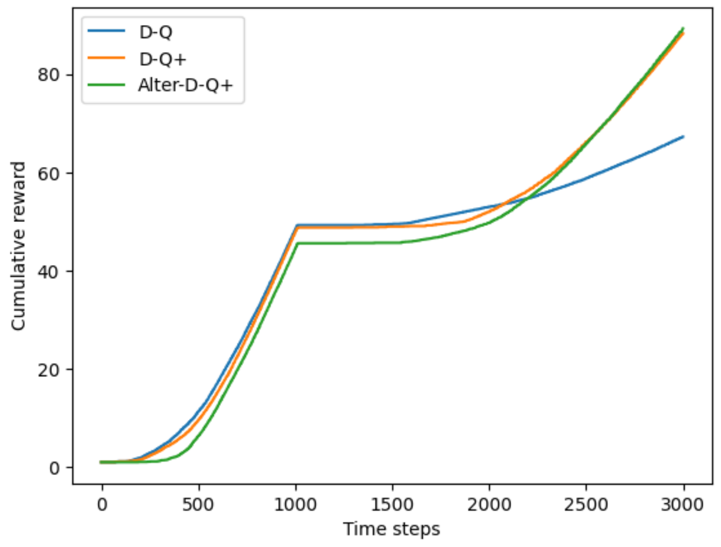


Figure 10: Result

Not surprisingly, it improves the alternative DynaQ+ in a long run. And at the beginning it is not the best anymore. I think it is because it focuses too much on the in-visited state but less random, so when it passes the gate of barrier it will intend to go back since there are many in-visit one behind. But the difference between those gets smaller.

Extra Credits

Question 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?

The answer is obvious that both phases relies on how good the exploration is and DyanQ+ has the advantage not to visit the one that has been checked before, so it explores better.

Question 8.3: Careful inspection of Figure 8.6 reveals that the difference between Dyna-Q+ and DynaQ narrowed slightly over the first part of the experiment. What is the reason for this?

As we mentioned before in the class about balance of explore and exploit, in the long run agent needs to focus on exploit after certain amount of explorations are done. Because the benefit of keep exploring gets smaller as more steps tried since agent and Q table will know enough about the environment. Therefore, even if DynaQ is ϵ greedy, it might not explore better at the beginning, but exploit is good enough by the end since the maze is actually simple.