COMP 138 RL: Homework 3

Ramtin Hosseini

November 3, 2020

1 Introduction

Reinforcement learning methods require a model of the environment, such as dynamic programming and heuristic search, and methods that can be used without a model, such as Monte Carlo and temporal difference methods. These are respectively called model-based and model-free reinforcement learning methods. Model-based methods rely on planning as their primary component, while model-free methods primarily rely on learning.
Here we explore Dyna-Q and a few modifications of it and show how planning and learning can be done simultaneously to make the agent learn the tasks in a higher speed.
In the programming assignment 8.4 of the book [1], we would like to explore the effect of bonus described in this chapter of the book. The bonus actually changes the estimated values of states and actions. But the question is whether this is necessary? Suppose the bonus was not used not in updating states, but solely in action selection. That is, suppose the action selected was always that for which $Q(S_t, a)$ was maximal. We Carry out a grid world experiment that tests and illustrates the strengths and weaknesses of this alternate approach.

2 Learning Methodology

In solving a Reinforcement Learning problem, in case both decision making and model learning are computation-intensive processes, then the available computational resources may need to be divided between them. To solve such problems, we can use Dyna-Q, which has a simple architecture and integrate simultaneously planning, acting and learning.
The overall architecture of Dyna agents, of which the Dyna-Q algorithm is one example, is shown in Figure 1. The central column represents the basic interaction between agent and environment, giving rise to a trajectory of real experience. The arrow on the left of the figure represents direct reinforcement learning operating on real experience to improve the value function and the policy. On the right are model-based processes. The model is learned from real experience and gives rise to simulated experience. We use the term search control to refer to the process that selects the starting states and actions for the

simulated experiences generated by the model. Finally, planning is achieved by applying reinforcement learning methods to the simulated experiences just as if they had really happened. Typically, as in Dyna-Q, the same reinforcement learning method is used both for learning from real experience and for planning from simulated experience.
Conceptually, planning, acting, model-learning, and direct RL occur simultaneously and in parallel in Dyna agents. For concreteness and implementation on a serial computer, however, we fully specify the order in which they occur within a time step. In Dyna-Q, the acting, model-learning, and direct RL processes require little computation, and we assume they consume just a fraction of the time. The pseudocode algorithm for Dyna-Q is provided in the figure 2.

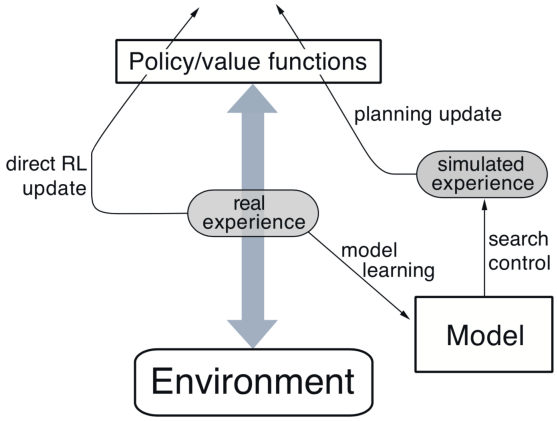


Figure 1: The general Dyna Architecture

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take 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) Loop 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 2: Tabular Dyna-Q

3 Experimental Results

Figure 3 shows the overview of average learning curve of using different variations of DynaQ (DynaQ vs. DynaQ+ vs. DynaQ+ experiment) for different settings of planning agent (5 and 50 step).
Figure 4 investigates this comparison in a shorter range. Figure 4a compares DynaQ and DynaQ+ and Figure 4b illustrates DynaQ+ and DynaX+ extra experiment.
In these experiments at episode 50, to make the maze more difficult to solve, I changed the block and at episode 100, I remove all blocks from the map to make it easier.
The results show that DynaQ+ with experiment will have lower variance after convergence but it takes longer time to re-converge when environment get worse. This is due to picking action in lieu of rewriting q values will make spread of value change slower, but also makes the useless exploration less. There is no difference for all mentioned methods with the better condition. This might be due to that they are not that much better.
Here again we are dealing with a trade off. We would like the agent to explore to find changes in the environment, but not so much that performance is greatly degraded. As in the earlier exploration/exploitation conflict, there probably is no solution that is both perfect and practical, but simple heuristics are usually effective.

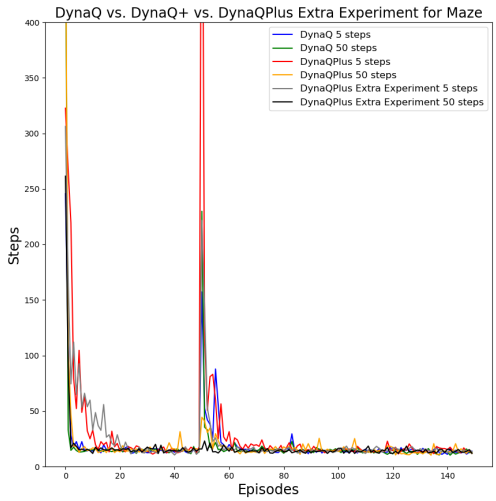
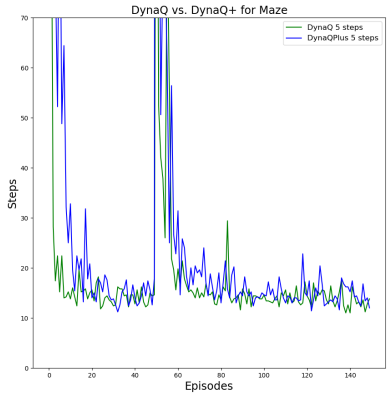
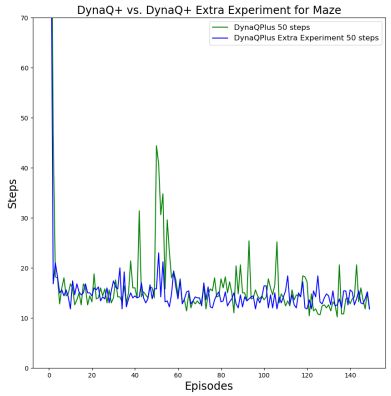


Figure 3: Overview of performance of different versions of DynaQ algorithm for two settings (5 and 50 steps)



(a) DynaQ and DynaQ+



(b) DynaQ+ and DynaQ+ Extra Exp.

Figure 4: Comparison of different variations of DaynQ algorithm

4 Extra Credit

I did the following 2 homeworks in the book for extra credits.

4.1 Exercise 8.2

Question: 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?
Answer: In phase one, DynaQ+ finds the optimal policy faster than DynaQ, that's why its slope rate quickly fixed to the optimal one and creates the gap when DynaQ was struggling. It is because DynaQ+ is very good at exploration to find the best policy. In the second phase, both algorithms find the new optimal policy in the blocking example because both can handle the situation where environment gets worse. We can observe DynaQ+ is faster. In the shortcut example, DynaQ is not able to find new policy because it is very difficult for ϵ method to keep exploratory across many steps. DynaQ+ are forced to explore further and thus creates the huge gap.

4.2 Exercise 8.3

Question: Careful inspection of Figure 8.5 reveals that the diffrence between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this?
Answer: The reason is that, the gap is created by the faster convergence of DynaQ+ but the exploration has its own cost. Such cost will force DynaQ+ to explore much more than DynaQ even it is already optimal. Such cost will

gradually decreases the gap, that's why there is a narrowed sign. If the game continues without shortcut, DynaQ+ will have lower cumulative rewards than DynaQ but the optimal policy should remain the same.

5 Discussion

In this homework, first I design DyanQ as the baseline and then compared the modified variations of it together. In the experiments, I changed the block to increase the difficulty at episode 50 and remove all blocks from the map at episode 100 to make it easier.
DynaQ+ with extra experiment will have lower variance after convergence but it takes longer time to re-converge when environment get worse. This is due to picking action instead of rewriting q values will make spread of value change slower, but also make the meaningless exploration fewer. There is no difference for all mentioned methods with the better condition.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.