Chapter 8 solutions **Hanna Gábor**

1. *The nonplanning method looks particularly poor in Figure 8.3 because it is a one-step method; a method using multi-step bootstrapping would do better. Do you think one of the multi-step bootstrapping methods from Chapter 7 could do as well as the Dyna method? Explain why or why not.*

   I think it can come close. An n-step bootstrapping algorithm with big enough n (or a Monte Carlo method) would update at the end of the first episode all the action-values we encountered during the episode. I expect the policy based on these updated action-values to be quite good.

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?*

   At first, both algorithms might find an okay policy that is suboptimal. Due to the exploration, Dyna-Q+ will realize it sooner that it's a suboptimal policy.

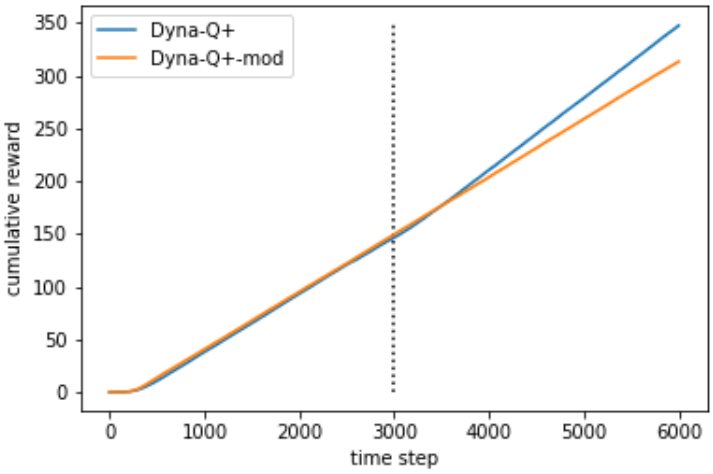
3. *Careful inspection of Figure 8.5 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this?*

   After finding the optimal path, Dyna-Q always uses that path, whereas Dyna-Q+ does some exploration from time to time.

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

   My expectation is that in some cases it's worse to use the bonus only in the action selection. Suppose that we need a series of exploratory steps to find the shortcut. If we update the actual values, then once the exploration bonus is big enough, we will soon find the shortcut. If we only use the bonus for action selection, it might happen that we take one exploratory step but before reaching the shortcut, we get back to the old path. If this happens, we will need to wait until the first exploratory step accumulates a huge bonus again to go down on that path again.

   I couldn't reproduce the cumulative reward of 400 shown in the book, my implementation of Dyna-Q+ only reached 350 and I couldn't figure out why. Anyway, here is the result.



5. *How might the tabular Dyna-Q algorithm shown on page 164 be modified to handle stochastic environments? How might this modification perform poorly on changing environments such as considered in this section? How could the algorithm be modified to handle stochastic environments and changing environments?*

   Instead of assigning single reward-state pairs to state-action pairs in the model, we could store all the reward-state pairs that we've seen after taking the respective action. We would sample from these stored items during the planning phase. To handle changes in the environment, we could modify this to sample only from the k latest reward-state pairs.

6. *The analysis above assumed that all of the b possible next states were equally likely to occur. Suppose instead that the distribution was highly skewed, that some of the b states were much more likely to occur than most. Would this strengthen or weaken the case for sample updates over expected updates? Support your answer.*

   The sample updates would be better in this case. E.g. if there is one state that has high probability, then a sample update will probably use that state. Using only this highly probable state for the sample update makes the result closer to the expected update, as that state has a huge weight in the expected update. Thus it strengthens the case for sample updates over expected updates.

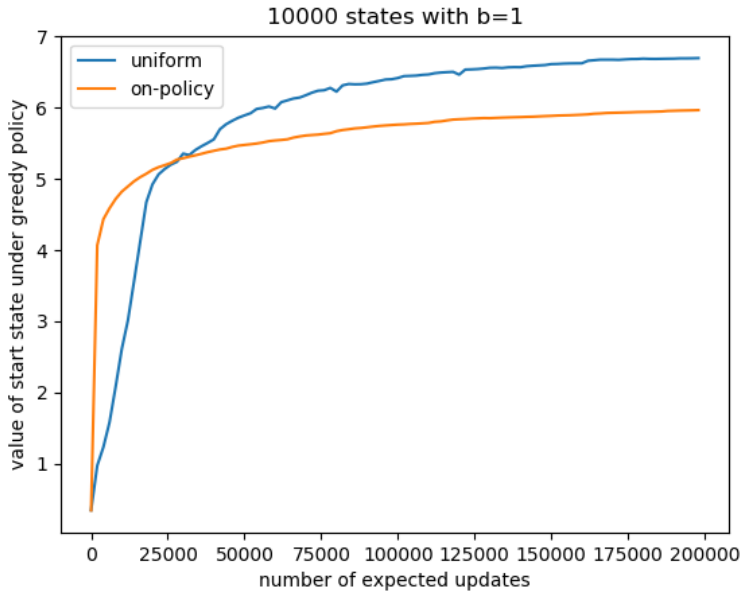
7. *Some of the graphs in Figure 8.8 seem to be scalloped in their early portions, particularly the upper graph for $b = 1$ and the uniform distribution. Why do you think this is? What aspects of the data shown support your hypothesis?*

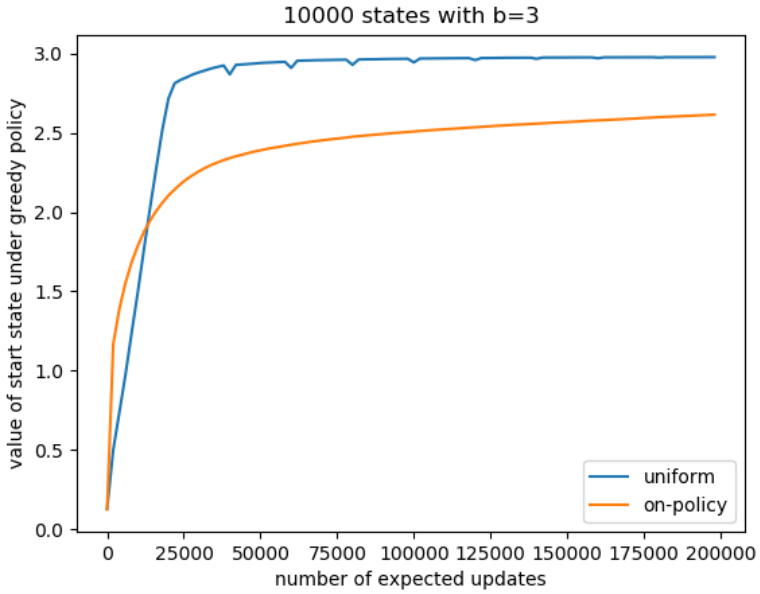
   I don't have a coherent answer, just some vague thoughts. If you change a value close to the start state that changes the chosen action in the corresponding state, then the greedy route becomes quite different and there can be a bigger change in the value of the start state. We change a lot of state-action values that doesn't effect the greedy path. If b is smaller, then there are more such state-action values that have no real effect on the greedy path.

8. *Programming. Replicate the experiment whose results are shown in the lower part of Figure 8.8, then try the same experiment but with $b = 3$. Discuss the meaning of your results.*

The code can be found at https://github.com/hannagabor/SBRL/blob/master/8.8/trajectory_sampling.py.

My results are a bit different from the results in the book. I couldn't find the difference in the code, and the results are not that different. Anyway, my results for the $b = 1$ and $b = 3$ cases are the following. (The diagrams show the average of 100 runs.)





Differences:

- The maximum state value of the starting state is higher in the $b = 1$ case. This makes sense: the average of three samples from a Gaussian distribution has a smaller standard deviation than just one sample.

- In case of $b = 3$, it happens sooner that the uniform update provides a better policy than the trajectory sampling method. I am not sure I would have predicted this, but I can explain it afterwards. In the $b = 3$ case, we use the estimates of three possible next states, but during one episode we only end up in one of these possible states. It's a disadvantage of the trajectory sampling method that it performs updates on the same states over and over again without improving the estimation of their successors.