CS7180: Special Topic - Artificial Intelligence (Spring 2022) Christopher Amato
Northeastern University Due Mar 23, 2022

Exercise 6: Planning and Learning

Please remember:

- Exercise due at **11:59 PM EST Mar 23, 2022**.
- Submissions should be made electronically on Canvas. Please ensure that your solutions for both the written and programming parts are present. You can upload multiple files in a single submission, or you can zip them into a single file. You can make as many submissions as you wish, but only the latest one will be considered.
- For **Written** questions, solutions may be handwritten or typeset. If you write your answers by hand and submit images/scans of them, please please ensure legibility and order them correctly in a single PDF file.
- The PDF file should also include the figures from the **Plot** questions.
- For both **Plot** and **Code** questions, submit your source code along with reasonable documentation.
- You are welcome to discuss these problems with other students in the class, but you must understand and write up the solution and code yourself. Also, you *must* list the names of all those (if any) with whom you discussed your answers at the top of your PDF solutions page.
- Each exercise may be handed in up to two days late (24-hour period), penalized by 10% per day late. Submissions later than two days will not be accepted.
- Contact the teaching staff if there are medical or other extenuating circumstances that we should be aware of.

1. **1 point.** (RL2e 8.1) *Planning vs. n-step returns.*
   Written:
   (a) 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.
   (b) Consider using both n-step returns and Dyna (specifically, using n-step returns instead of one-step returns in tabular Dyna-Q). We can clearly do this for the learning phase (d), but how about for the planning phase (f)? What are the advantages and disadvantages of using n-step returns in the planning phase?
   (c) **[Extra credit.] 0.5 point.** Perform computational experiments to verify your answers to the above questions.
   For part (b), you will have to implement tabular Dyna-Q with n-step returns.

2. **1 point.** (RL2e 8.2, 8.3) *Analysis of Dyna-Q+.*
   Written: Read and understand Examples 8.2 and 8.3, then answer the following:
   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?

3. **3 points.** *Implementing Dyna-Q and Dyna-Q+.*
   (a) **Written:** For some unknown reason, the textbook is quite vague about Dyna-Q+ and does not provide pseudocode for the modification. Read Section 8.3 carefully (specifically p. 168) and reconstruct the pseudocode for Dyna-Q+. Look carefully at the footnote[1] on p. 168 as well.
   (b) **Code/plot:** Reproduce the domains/figures in Section 8.3 (Figures 8.4 and 8.5) to verify whether your interpretation of Dyna-Q+ in part (a) is correct.
   Note that "cumulative reward" refers to the number of completed episodes, *not* the discounted return.
   Refer to Example 8.1 for domain and hyperparameter details. Experiment with a some (n, κ) combinations to find out what they might have used. Try different orders of magnitude of κ.
   $n \approx 250$ planning steps were probably used, although we are not completely sure.
   As usual, use good experimental practice by performing a reasonable number of trials,
   and report average performance with confidence bands.
   Tip: You may find the plots surprisingly difficult to reproduce. Don't worry if it's not exact, just try.

[1] Why are important details left to the footnote?

(c) **Code/plot:** How important is the footnote[2]? Find out by modifying your Dyna-Q+ implementation to exclude the suggestion in the footnote, and perform some computational experiments.
Written: Did the footnote matter? Why or why not?

4. **1 points.** (RL2e 8.4) *Exploration bonus in rewards vs. action selection.*
Recall that the idea of exploration bonuses appeared in the UCB algorithm for multi-armed bandits as well. However, there is a subtle difference here:

   - In UCB, the bonus was applied during *action selection*, but the Q-values (estimated return of arms) were not affected by this bonus.
   - In Dyna-Q+, the bonus is applied given to the simulated *rewards*, and therefore the Q-value estimates are affected by the bonus.

   This question explores this subtle but important difference. **Written:** Think carefully about the implications of applying bonuses to the reward function ("pseudo-reward") versus during action selection. Make some predictions:

   - What would generally happen under each approach? How do they differ?
   - What are the advantages and disadvantages of each approach?

5. **1 points.** (RL2e 8.5) *Dyna-Q for stochastic environments.*

   (a) **Written:** How might the tabular Dyna-Q algorithm be modified to handle stochastic environments? Would this modification perform poorly on changing environments such as considered in Section 8.3? How could the algorithm be modified to handle stochastic environments and changing environments?

   (b) **Code/plot/written:** Implement the modifications proposed in part (a).
   Implement both versions – one that only handles stochastic environments,
   and one that handles both stochastic and changing environments.
   Design and implement computational experiments in the maze domains to verify that your proposed changes work. Comment on what you find.
   [**Optional.**] Experiment on the windy gridworld with stochastic wind from Ex5 to verify your findings.

6. **3 points.** *Manual MCTS*

   Perform 8 iterations of MCTS on the Random Walk MDP. The MDP is obtained by incorporating actions `left` and `right` to the Markov reward process of Example 6.2.

   - In each state-node, write the MDP state it represents;
   - In each action-node, write the total sum of simulated returns as T, and the number of simulations as N;
   - As *tree policy*, use the UCB rule with exploration constant $c = \sqrt{2}$, which selects the next action by maximizing
   $$\arg\max_a \left\{ \frac{T_a}{N_a} + \sqrt{\frac{2 \ln N_s}{N_a}} \right\} \tag{1}$$
   where N_a is the number of simulations for the proposed action-node, T_a is the total sum of simulated returns for the proposed action-node, and N_s is the number of simulations for the current state-node, i.e. the sum of N_a for both proposed action-nodes. Ties should be broken randomly;
   - As *rollout policy*, use the uniform distribution over the two actions;
   - To guarantee that we all get the same result, we will seed our randomness by using an online RNG:
     - Go to `https://www.random.org/integers/?mode=advanced`.
     - In Part 1, select however many random integers you want to generate;
     - In Part 1, select that the integers should have a value between 0 and 1;
     - In Part 4, use a pregenerated randomization based on the persistent identifier `rldm_mcts`

[2] A wise reviewer once wrote: "Avoid footnotes: a wise person once told me that 'something is either important enough to be included in the main text, or it is not important enough to be a footnote'."

This will give you a sequence of 0s and 1s, which you should respectively interpret as `left` and `right`. Whenever you need to sample an action, use the next symbol in the sequence. To verify that you've got the correct "random" sequence, verify that yours starts with:

11100000101111100101100011001100000011011011010001110011101111001100010111101100000001

- For each iteration of MCTS, draw the resulting tree, *clearly* indicating the results of each of the four phases:
  - Selection: indicate the actions taken from the root to the selected node.
  - Expansion: indicate the node resulting from expansion.
  - Rollout: write down the entire simulated sequence, i.e. the states, actions, and rewards until the terminal state is reached.
  - Backup: indicate how the values in the tree have changed as a result of the simulation.
- After having run the 8 simulations, indicate which action should be taken for the start state.

As confirmation that you've done everything right:

- Your final tree should have 9 state-nodes.
- You should have queried the RNG 50 times.

Please be as clear as possible. This exercise will require you to draw a lot, so make it clear what you're doing. We can only assign points if we understand your answer.