4180/5180: Reinforcement Learning and Sequential Decision Making (Fall 2024) Christopher Amato
Northeastern University Due Sep 27, 2024

Exercise 2: Markov Decision Processes (MDPs)

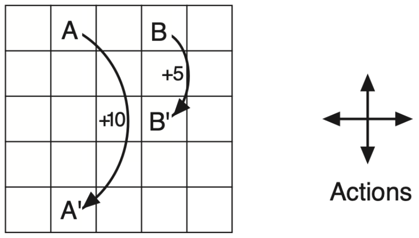
Please remember the following policies:

- Exercise due at **11:59 PM EST Sep 27, 2024**.
- 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 in Jupyter Notebook (.ipynb file) along with reasonable comments of your implementation. Please make sure the code runs correctly.
- 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.
- **Notations: RL2e is short for the reinforcement learning book 2nd edition. x.x means the Exercise x.x in the book.**

1. **1 point.** *Formulating an MDP.*
   Written: It is instructive to formally define an MDP at least once, in particular, the dynamics function. Consider the four-rooms domain from Ex0.
   (a) What are the state and action spaces S, A?
   (b) Consider the dynamics function $p(s', r|s, a)$. Approximately how many rows in this conditional probability table have non-zero probability?
   (c) **1 point. (Bonus, Optional) Written or Code:** Provide pseudocode or actual code to generate the full table of non-zero $p(s', r|s, a)$ values. Notes:
      - If you provide pseudocode, it should be more specific than "generate $p(s', r|s, a)$ for all (s, a, s', r)". It should include some actual probability values, consider effects of walls, etc.
      - If you generate the table using code, please include both the code and the generated table of values.
      - Recall: The goal state is at $(10, 10)$. Any action from there teleports the agent back to $(0, 0)$.

2. **1 point.** (RL2e 3.8, 3.9) *Discounted return. (ADDING from LAWSON since last question removed)*
   Written:
   (a) Suppose $\gamma = 0.5$ and the following sequence of rewards is received:
   $R_1 = -1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$
   with $T = 5$. What are $G_0, G_1, \ldots, G_5$? Hint: Work backwards.
   (b) Suppose $\gamma = 0.9$ and the reward sequence is $R_1 = 2$ followed by an infinite sequence of '7's.
   What are G_1 and G_0?

3. **1 point.** (RL2e 3.6, 3.7) *The RL objective.*
   Written:



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Figure 3.2: Gridworld example: exceptional reward dynamics (left) and state-value function for the equiprobable random policy (right).

(a) Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for -1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task? (Derive expressions for both the episodic and continuing cases.)

(b) Imagine that you are designing a robot to run a maze. You decide to give it a reward of $+1$ for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes – the successive runs through the maze – so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (Equation 3.7). There is no discounting, i.e. $\gamma = 1$. After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

4. **1 point. [5180]** (RL2e 3.15, 3.16) *Modifying the reward function.*
Written:

(a) In the gridworld example (Figure 3.2 in RL2e, see below), rewards are positive for goals, negative for running into the edge of the world, and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using Equation 3.8, that adding a constant c to all the rewards adds a constant, v_c, to the values of all states, and thus does not affect the relative values of any states under any policies. What is v_c in terms of c and γ?

(b) Now consider adding a constant c to all the rewards in an episodic task, such as maze running. Would this have any effect, or would it leave the task unchanged as in the continuing task above? Why or why not? Give an example.

5. **2 point.** (RL2e 3.14) *Bellman equation.*
Written:

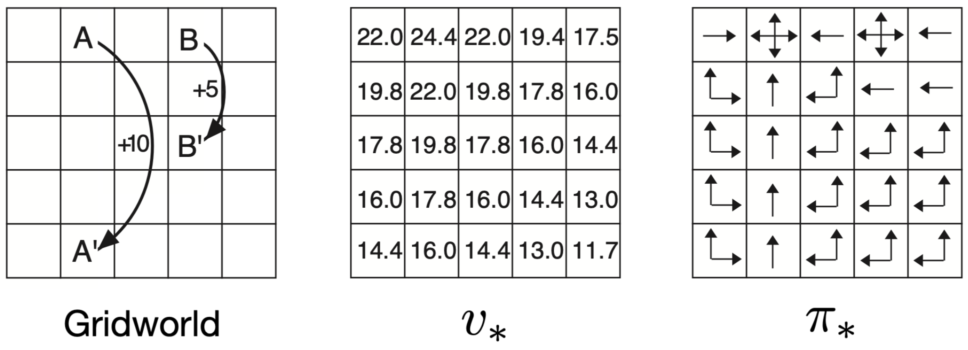
(a) The Bellman equation (Equation 3.14) must hold for each state for the value function v_π shown in Figure 3.2 (right) (see above) of Example 3.5. Show numerically that this equation holds for the center state, valued at $+0.7$, with respect to its four neighboring states, valued at $+2.3, +0.4, -0.4, +0.7$. (These numbers are accurate only to one decimal place.) Note that Figure 3.2 (right) (see above) is the value function for the equiprobable random policy. $\gamma = 0.9$

(b) The Bellman equation holds for *all* policies, including optimal policies. Consider v_* and π_* shown in Figure 3.5 (middle, right respectively) (See below). Similar to the previous part, show numerically that the Bellman equation holds for the center state, valued at $+17.8$, with respect to its four neighboring states, for the optimal policy π_* shown in Figure 3.5 (right) (see below).

If the Bellman equation is unclear to you, you should practice this question again for other states.

6. **2 points.** *Solving for the value function.*
Written: Another way to solve for the value function is to write out the Bellman equation for each state, view it as a system of linear equations, and then solve for the unknowns (the value of each state). Consider a particularly simple MDP, the 2-state recycling robot in Example 3.3.

(a) Expand the Bellman equation for the 2 states in the recycling robot, for an arbitrary policy $\pi(a|s)$, discount factor γ, and domain parameters $\alpha, \beta, r_{\texttt{search}}, r_{\texttt{wait}}$ as described in the example.



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Figure 3.5: Optimal solutions to the gridworld example.

(b) You should now have two linear equations involving two unknowns, $v(\texttt{high})$ and $v(\texttt{low})$, as well as involving the policy $\pi(a|s)$, γ, and the domain parameters. Let $\alpha = 0.8, \beta = 0.6, \gamma = 0.9, r_{\texttt{search}} = 10, r_{\texttt{wait}} = 3$. Consider the policy $\pi(\texttt{search} \,|\, \texttt{high}) = 1$, $\pi(\texttt{wait} \,|\, \texttt{low}) = 0.5$, and $\pi(\texttt{recharge} \,|\, \texttt{low}) = 0.5$.
Find the value function for this policy, i.e., solve the equations for the values of $v(\texttt{high})$ and $v(\texttt{low})$. Check that your solution satisfies the Bellman equation.

(c) **0.5 points. (Bonus, optional**) Suppose you can modify the policy in the `low` state, i.e., set $\pi(\texttt{wait} \,|\, \texttt{low}) = \theta$, and $\pi(\texttt{recharge} \,|\, \texttt{low}) = 1 - \theta$. What θ should you set it to, and what is the value function for that θ?