Reinforcement Learning Tutorial 5, Week 6
— with solutions —
Monte Carlo Control / TD Prediction

Pavlos Andreadis, Sanjay Rakshit

February 2025

Overview: The following tutorial questions relate to material taught in week 3 of the 2024-25 Reinforcement Learning course. They aim at encouraging engagement with the course material and facilitating a deeper understanding.

For this week, we will look at how the concepts of terminating and absorbing state are not actually compatible, but relate to different formulations of the same problem (or pseudo-code, if you prefer). We then rehash Monte Carlo (MC) prediction and touch upon MC control. We will make use of Temporal Difference (TD) learning for one prediction (policy evaluation) step. The answers (whether delivered by your tutor or read later at home) will provide you also with a somewhat more theoretical consideration relating to TD prediction convergence.

Problem 1 - Modelling & Monte Carlo Control

Consider the simple maze problem in Figure 1 below, comprised of 8 states $s_1, \cdots, s_8$, numbered from the bottom left to the top right. The agent can move from any state to any adjacent state (e.g. from s_1 to either s_4 or s_2), without error. Our goal is to follow the shortest path (from any state) to s_8. Upon arrival to a new state, the agent receives a reward dependent only on that new state. We assign s_8 a reward of 10, and penalise arrival to any other state with -1.

The arrows in Figure 1 summarise the policy π_0 which we will be evaluating in **Part b** of this question. Essentially, assume a deterministic policy for states s_2, s_3, s_5, s_6, s_7, as indicated by the respective arrow. Further assume a 50% chance of moving in either direction for states s_1, s_4.

$(s6, \rightarrow, -1)$	$(s7, \rightarrow, -1)$	$(s8, +10)$
$(s4, \uparrow, \downarrow, -1)$		$(s5, \uparrow, -1)$
$(s1, \uparrow, \rightarrow -1)$	$(s2, \rightarrow, -1)$	$(s3, \uparrow, -1)$

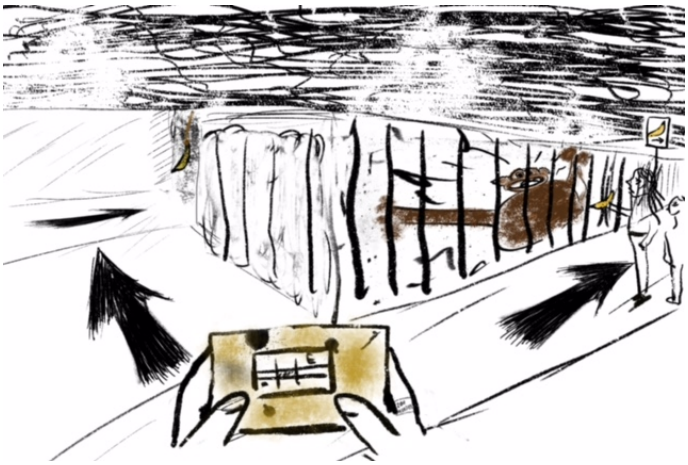

Figure 1: "Lost Phil: First Person Keeper" (Image and title used with permission from Yana Knight and Andreadis [2021]

Part a

- Should s_8 be defined as a terminating state? Why?
- Should s_8 be defined as an absorbing state? Why?

From here on, assume a discount factor of $\gamma = 1$.

Answer:

- Yes, as the goal is to reach state s_8, and we do not have a problem where we need to keep returning to it (such as, e.g., when the state represents a plane's altitude that we would like to maintain). As such, it is also sound to not define any available actions at that state, as the episode terminates upon arrival at it.
- No, not under the current problem definition, especially because of the fact that we are rewarding the agent upon arrival at the state. Recall that an absorbing state only allows actions that deterministically transition back to the same state. Therefore, we require that all arrivals at the absorbing state from the absorbing state itself give a reward of 0 (or we wouldn't be able to meaningfully compare any policies).

  An alternative formulation could define s_8 as an absorbing state, and rewards arrivals to it only from its neighbouring states (so the rewards would no longer only depend on the next state).

Another commonly used approach is to define a new abstract state, whose sole purpose is to become the current state after the episode is practically completed. So we could define an absorbing state s_9 to which the environment would transition deterministically for any action from s_8.

- To summarise:
  - An absorbing state is a state that, once entered, cannot be left.
  - A terminating state is a state that, once entered, ends the episode.

To remove any conflation of terminating and absorbing states, consider the alternative MDP formulations in Sections 3.3 and 3.4 of Sutton and Barto [2018].

Part b

Assuming the starting state $S_0 = s_1$ and the policy π_0 outlined above, list the two shortest possible trajectories our agent can follow (stopping at state 8). Further to that, consider the trajectory:

$$(s_1, up), -1, (s_4, down), -1, (s_1, right), -1, (s_2, right), -1, (s_3, up), -1, (s_5, up), +10, (s_8)$$

For each of those trajectories, carry out an iteration of policy evaluation using First-visit Monte Carlo (where it is implied that you average across samples as opposed to using some other learning rate), computing the *action value function*. Start from an initial evaluation of 0 across state-action pairs and go through the trajectories in any order.

Answer:

We can easily sum up the rewards following each first visit, and then average across all samples for that state-action pair, which will be our updated evaluation of the policy π_0 for that pair. The trajectories and respective updates are given below:

Trajectory 1	
$(s1, right)$	7
$(s2, right)$	8
$(s3, up)$	9
$(s5, up)$	10

Trajectory 2	
$(s1, up)$	7
$(s4, up)$	8
$(s6, right)$	9
$(s7, right)$	10

Trajectory 3	
$(s1, up)$	$(7 + 5)/2 = 6$
$(s4, down)$	6
$(s1, right)$	$(7 + 7)/2 = 7$
$(s2, right)$	$(8 + 8)/2 = 8$
$(s3, up)$	$(9 + 9)/2 = 9$
$(s5, up)$	$(10 + 10)/2 = 10$

Part c

Perform one step of greedy policy improvement on policy π_0 (assuming no access to the model), based on the evaluation from **Part b**.

Answer:

Let us summarise the state-action values:

State-action values	
$(s1, up)$	6
$(s1, right)$	7
$(s2, right)$	8
$(s3, up)$	9
$(s4, up)$	8
$(s4, down)$	6
$(s5, up)$	10
$(s6, right)$	9
$(s7, right)$	10

Monte Carlo policy improvement assumes no access to the model (and therefore doesn't require one). Instead we $argmax$ over actions for the action-value function, with the new deterministic policy π_1 being:

$(s6, \rightarrow)$	$(s7, \rightarrow)$	$(s8)$
$(s4, \uparrow)$		$(s5, \uparrow)$
$(s1, \rightarrow)$	$(s2, \rightarrow)$	$(s3, \uparrow)$

Problem 2 - TD Prediction

Use the trajectory

$$(s_1, right), -1, (s_2, right), -1, (s_3, up), -1, (s_5, up), +10, (s_8)$$

to run one iteration of Temporal Difference policy evaluation (use the SARSA update rule) on the policy π_1 you computed for **Problem 1c**. Assume a step size of $\alpha = 0.1$ (you are assuming that the action that would be taken at each time-step of this trajectory when sampling actions using π_1 is the one indicated in the trajectory).

Answer:

Trajectory 1	
$(s1, right)$	$7 + 0.1 * [-1 + 1 * 8 - 7] = 7$ (no change)
$(s2, right)$	$8 + 0.1 * [-1 + 1 * 9 - 8] = 8$ (no change)
$(s3, up)$	$9 + 0.1 * [-1 + 1 * 10 - 9] = 9$ (no change)
$(s5, up)$	$10 + 0.1 * [+10 + 1 * 0 - 10] = 10$ (no change)

Note, that the update for $(s_1, right)$ only happens after we have arrived at s_2 and decided to take the action $right$ from there. Similarly for the other updates.

Also, note that the action-value function did not change for any of the visited state-action pairs.

[*Is this fact enough to determine whether we have converged? Can you think of how this could have happened without us converging?*]

References

Yana Knight and Pavlos Andreadis. "Story of Yana". http://storyofyana.com/, 2021.

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