Reinforcement Learning Tutorial 3, Week 4

—

Dynamic Programming & Monte Carlo Methods

Pavlos Andreadis, Adam Jelley, Sanjay Rakshit

Overview: The current tutorial questions relate to material taught in weeks 2/3 of the 2023-24 Reinforcement Learning course. They aim at encouraging engagement with the course material and facilitating a deeper understanding.

This week you are presented with a couple of exercises related to Dynamic Programming and Monte Carlo methods asking you to do some computations by hand. For the Dynamic Programming problem, there is no need to run these till convergence. The solution of **Problem 1** is intended to show you enough to see how it works, and to give you the details on what it converged to and after how many steps. **Problem 1** requires two outer loops of the *Policy Iteration* algorithm, and the first *Policy Evaluation* converges after 15 updates (4 updates are enough to understand what is going on).

If you are so inclined, you could also write a quick script to tackle Problem 1. This can be a good exercise in understanding how the algorithm works and will help to prepare you for the coursework.

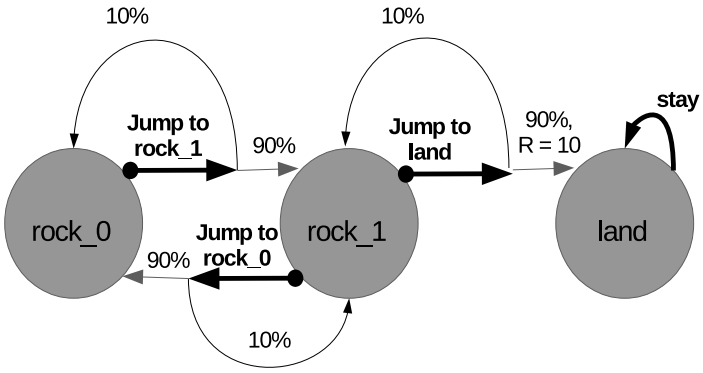
Generally, when considering Reinforcement Learning algorithms, note that most of them[1] can be understood as a specific instance of the *Generalised Policy Iteration* procedure: We iterate, until the policy converges, over one step of some variant of *Policy Evaluation* (which needs to improve the evaluation of the current policy at least by a little) and one step of some form of *Policy Improvement* (which we generally want to be at least somewhat greedy so that convergence is guaranteed). After the Policy Improvement step, we get a new policy whose evaluation (via state-value or action-value function) we can improve upon by the next Policy Evaluation step, and so forth until convergence. See Sutton and Barto [2018], section 4.6, page 86 for details.

The Monte Carlo policy evaluation methods in **Problem 2** could replace the Iterative Policy Evaluation step (a Dynamic Programming algorithm) of the Policy Iteration algorithm. [*Though we wouldn't usually do this while evaluating the state-value function. Why?*]

[1] Policy gradient methods don't generally follow this pattern.

Problem 1 - Dynamic Programming

Consider the Hop Along MDP from tutorial 2. Set the discount factor to $\gamma = 0.9$, and assume a reward of 10 for reaching land (and 0 otherwise). Apply two iterations of *Policy Iteration*, starting from a uniform initial policy.



Do the final values and policy agree with your intuition?

Problem 2 - Monte Carlo

Compute the state-value function for a given policy π and MDP without access to the MDP's model, using the following four episodes, in order:

$$\text{rock}_0, 0, \text{rock}_0, 0, \text{rock}_1, 10, \text{land} \tag{1}$$
$$\text{rock}_1, 0, \text{rock}_0, 0, \text{rock}_1, 10, \text{land} \tag{2}$$
$$\text{rock}_0, 0, \text{rock}_0, -100, \text{sea} \tag{3}$$
$$\text{rock}_1, 0, \text{rock}_0, -100, \text{sea} \tag{4}$$

Note: The discount factor used here is $\gamma = 1$. [*Why is this acceptable here?*]

Part 1

Use first-time visit Monte Carlo to evaluate the state-value function at each state. Show your calculation.

Part 2

Use every-time visit Monte Carlo to evaluate the state-value function at each state. Show your calculation.

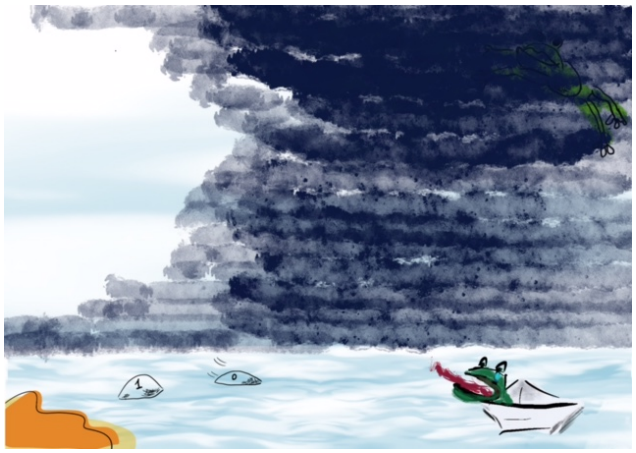


Figure 1: "Frog with More Problems." Image and title used with permission from Yana Knight and Andreadis [2021]

Part 3

How would you expect the state-values estimated by both first-time visit Monte Carlo and every-time visit Monte Carlo to change as the number of episodes considered tends to infinity? (No mathematical proofs are required, just a short description of the expected values).

Part 4

Which are the absorbing states?

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.