Reinforcement Learning Tutorial 6, Week 7

—

Reward Shaping and Gradient Monte Carlo

Pavlos Andreadis*

March 2025

Overview: The following tutorial questions relate to material taught in weeks 3 and 5 of the current Reinforcement Learning course. They aim at encouraging engagement with the course material and facilitating a deeper understanding.

This week's tutorial continues with the problem from Tutorial 4 and asks us to consider the use of *reward shaping*. This is of course a toy problem in which reward shaping would not have anything to offer if we only ran our procedures a bit more. To simulate situations from more complicated MDPs where training might indeed be too slow (or not converging at all; see *value function approximation*), we are assuming procedures that have stopped prematurely. The effect of reward shaping in our toy problem below is analogous to that of applying it in a more complex problem, especially as concerns the problems, or sacrifices if you prefer, inherent in reward shaping. What is less demonstrable here are the benefits of the approach in helping converge to reasonable solutions where otherwise it might not be practically feasible.

Having hopefully received a better understanding of reward shaping, we move on to an exercise on *value function approximation* and specifically on Gradient Monte Carlo. There is something odd with the samples handed to us here, but you can still compute an updated estimate of your action-value function. We will continue on this problem in the next tutorial.

Problem 1 - Discussion: Reward Shaping

Assume the problem as described in *Problem 1 of Tutorial 4*, but where after evaluating a deterministic policy π_2 (as given in Table 1 below) (e.g. using $TD(0)$), we have the following estimation of the state-value function:

*with special thanks to **Ross McKenzie** for providing a first version of Problem 2

$(s6, \rightarrow, v = 0)$	$(s7, \rightarrow, v = 0)$	$(s8)$
$(s4, \downarrow, v = 6)$		$(s5, \uparrow, v = 10)$
$(s1, \rightarrow, v = 7)$	$(s2, \rightarrow, v = 8)$	$(s3, \uparrow, v = 9)$

Table 1: Suboptimal policy.

We are frustrated that our agent has not learnt to "go up" when in state s_4, and decide to, instead of running the process further, apply reward shaping to the model, adding +2 reward to the state visits for states s_6, s_7, hoping that this will help the agent learn to take the shortest route from s_4.

1. What do you think would be the optimal policy for this modified MDP (with rewards for arriving at states s_6 and s_7 of +1)? Would the episodes terminate?
2. If instead of +2 to the above rewards, we instead add +1 (rewards for arriving at states s_6 and s_7 of 0), what would be the optimal policy?
3. In the above two models, would the calculated state-value function, after convergence, be representative of the original problem? Why?
4. When can reward shaping be a useful tool, and how?

Figure 1: "An AI-controlled orchard."

Problem 2 - Gradient Monte Carlo

An AI controlled orchard needs to decide when to harvest its trees. To do this it measures the concentration of three chemicals in the air. Each day the orchard

can choose to wait or harvest. Waiting costs one credit in operating costs while a harvest ends the process. Once a crop is harvested, packaged and sold, the orchard is told the profit or loss of that harvest. Most experts agree that the function mapping the chemical concentrations to the profit is approximately linear.

The orchard has several samples of the profits from other harvests. as seen in Table 2 below.

Concentration of A (ppm)	Concentration of B (ppm)	Concentration of C (ppm)	Profit/Reward (credits)
4	7	1	3
10	6	0	-15
20	1	15	5
4	19	3	21

Table 2: Samples of harvest profits

Begin to approximate the function that maps the state feature vector to Q(state, harvest) using a Monte Carlo target, doing a gradient decent step on each sample (using linear function approximation). Solutions will be provided for a learning rate of 0.01 and initial weights of $\boldsymbol{w}^0 = [w_a^0, w_b^0, w_c^0] = [3, 2, 1]$, but feel free to try any values.

Do you expect our Q(state, harvest) function to have converged?