Exercises - Chapter 6

Victoria Nagorski

Exercise 6.1

If V changes during the episode, then (6.6) only holds approximately; what would the difference be between the two sides? Let V_t denote the array of state values used at time t in the TD error (6.5) and in the TD update (6.2). Redo the derivation above to determine the additional amount that must be added to the sum of TD errors in order to equal the Monte Carlo error.

$$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$$

TD Update:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

TD Error:

$$\delta_t \doteq R_{t+1} + V(S_{t+1}) - V(S_t)$$

Define V_t as the state values use at time t in the TD error. We need to look at the difference between the two values- both G_t and V_t will be different. Define the difference between the two as ΔG_t and $\Delta V = V - V_t$.

$$\delta_{tt} = R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)$$

$$\Delta\delta_t = \delta_t - \delta_{tt} = \gamma \Delta V(S_{t+1}) - \Delta V(S_t)$$

After subtracting the two values

$$\Delta G_t - \Delta V(S_t) = \gamma \Delta G_{t+1} - \underbrace{\Delta V(S_t) + \gamma \Delta V(S_{t+1})}_{\Delta\delta_t} - \gamma \Delta V(S_{t+1}))$$

$$\Delta G_t - \Delta V(S_t) = \Delta\delta_t + \gamma(\Delta G_{t+1} - \Delta V(S_{t+1}))$$

Since the rewards would cancel out:

$$\Delta G_{t+1} = \gamma G_{t+2}$$

$$\Delta G_t - \Delta V(S_t) = \Delta\delta_t + \gamma(\gamma \Delta G_{t+2} + \underbrace{\gamma \Delta V(S_{t+2}) - \Delta V(S_{t+1})}_{\Delta\delta_{t+1}} - \gamma \Delta V(S_{t+2}))$$

$$\Delta G_t - \Delta V(S_t) = \Delta\delta_t + \gamma(\Delta\delta_{t+1} + \gamma(\Delta G_{t+2} - \Delta V(S_{t+2}))$$

Therefore:

$$\Delta G_t - \Delta V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \Delta\delta_k$$

Exercise 6.2

This is an exercise to help develop your intuition about why TD methods are often more efficient than Monte Carlo methods. Consider the driving home example and how it is addressed by TD and Monte Carlo methods. Can you imagine a scenario in which a TD update would better on average than a Monte Carlo update? Give an example scenario- a description of past experience and a current state- in which you would expect the TD update to be better. Here's a hint: Supposed you have lots of experience driving home from work. Then you move to a new building and new parking lot (but you still enter the highway at the same place). Now you are starting to learn predictions for the new building. Can you see why TD updates are likely to be much better, at least initially, in this case? Might the same sort of thing happen in the original scenario?

An example where a TD approach might be better than a Monte Carlo one is doing hw. Sometimes as a student you might become used to a professor assigning hw that takes a certain amount of time to complete. However, mid-semester the professor might start assigning hw that takes much longer to do than it would have in the past. You might want your estimate "time to complete" to update as you work through the problems so the assignment might actually be finished on time. If you keep the same time estimate until the end, the assignment might not get finished.

Once you reach the highway, you would be able to use past learning to form new intuitions. The moves up to the freeway might be very uncertain as there is little experience with this path. However, with TP you would be able continually update your estimates as you proceed. So the same thing might happen as the original scenario.

Exercise 6.3

From the results shown in the left graph of the random walk example it appears that the first episode results in a change in only $V(A)$. What does this tell you about what happened on the first episode? Why was only the estimate for this one state changed? By exactly how much was it changed?

A TD(0) graph will update look ahead one step and update the value based on that one look ahead. All values for the random walk example started with equal probability and same state value.

Since A was the same value as B, B's estimated value remained unchanged. However, since left to A is a terminal state, the value is 0. This would cause the A's state value shift down (as seen in the graph).

The other values did not change because all other state values were equal. Only at state A would there be a deviation caused by the terminal state value.

$$V(S_t) \leftarrow \underbrace{V(S_t)}_{0.5} + \underbrace{\alpha}_{0.1}[\underbrace{R_{t+1}}_{0.0} + \gamma \underbrace{V(S_{t+1})}_{0.0} - \underbrace{V(S_t)}_{0.5}]$$

$$V(S_t) = 0.45$$

Exercise 6.4

The specific results shown in the right graph of the random walk example are dependent on the value of the step-size parameter, α. Do you think the conclusions about which algorithm is better be affected if a wider range of α values were used? Is there a different, fixed value of α at which either algorithm would have performed significantly better than shown? Why or why not?

Looking at the graphs, generally we might expect MC algorithms to do better as α gets bigger. Only to a degree, however.

For TD, however, we notice that the smaller the alpha term, the better the empirical RMS error is at the end of the 100 walk/episodes.

I would expect at an α value greater than 1.5, we might see the error from TD to exceed that of MC. This would make sense because instead of slowly updating the estimated value, the value would take much larger jumps. The jumps may, however, cause a larger overshoot from the true value that would make it harder to converge a better estimate of the state.

Exercise 6.5

**(Challenge Problem) In the right graph of the random walk example, the RMS error of the TD method seems to go down and then up again, particularly at high α's. What could have caused this? Do you think this always occurs, or might it be a function of how the approximate value function was initialized?*

If you observe the left graph, on the very left node, the value states start above and then end up below the true value. TD methods work by slowly incrementing its way toward the true value. At some point, the algorithm will straggle the true value, but never be quite right. The α term determines just much this algorithm increments in its updates. The larger the incremental term, the larger the disparity will be as the algorithm begins to straddle the true value.

Exercise 6.6

In Example 6.2 we states that the true values for the random walk example are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}$, and $\frac{5}{6}$, for states A and E. Describe at least two different ways that these could have been computed. Which would you guess we actually used? Why?

One method of solving for the true state value is to linearly split up the values from 0 (terminal state) to 1 (the other terminal state). The values can be split linearly from 0 to 1. At C, the state value is halfway between points 0 and 1. Therefore, we would expect the value to be 0.5. The rest of the values increase or decrease linearly from there. I would expect this method because it is the most straightforward and fastest method. It is also the best was to get a clean fraction.

An iterative method could have been used to solve the problem- as it is simple enough that it would probably converge relatively quick. However, it would probably have some sort of error that the user could define to some decimal point. This method usually requires more memory, but the problem is smaller.

Exercise 6.7

**(Challenge Problem) Design an off-policy version of the TD(0) update that can be used with arbitrary target policy π and covering behavior policy b, using at each step t the importance sampling ratio $\rho_{t:t}$ (5.3).*

TD Update:

$$V(S_t) \leftarrow V(S_t) + \alpha[\underbrace{R_{t+1} + \gamma V(S_{t+1}}_{G_t} - V(S_t)]$$

Therefore:

$$V(S_t) \leftarrow V(S_t) + \alpha[\rho_{t:t} G_t - V(S_t)]$$

Exercise 6.8

Show that an action-value version of (6.6) holds for the action-value form of the TD error $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$, again assuming that the values don't change from step to step.

For the previous solved equations of 6.6:

$$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$$

$$G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}))$$

Derive:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

Expanding out:

$$= R_{t+1} + \gamma G_{t+1} - Q(S_t, A_t) + \gamma Q(S_{t+1}, A_{t+1}) - \gamma Q(S_{t+1}, A_{t+1})$$

$$= \delta_t + \gamma[G_{t+1} - Q(S_{t+1}, A_{t+1})]$$

$$= \delta_t + \gamma[R_{t+2} + \gamma G_{t+2} - Q(S_{t+1}, A_{t+1}) + \gamma Q(S_{t+2}, A_{t+2}) - \gamma Q(S_{t+2}, A_{t+2})]$$

$$= \delta_t + \gamma(\delta_{t+1} + \gamma[G_{t+2} - Q(S_{t+2}, A_{t+2})])$$

$$= \delta_t + \gamma\delta_{t+1} + \gamma^2[G_{t+2} - Q(S_{t+2}, A_{t+2})])$$

$$= \sum_{k=t}^{T-1} \delta_k \gamma^{k-t}, QED$$

Exercise 6.9

Windy Gridworld with King's Moves: Re-Solve the windy gridworld assuming eight possible actions, including the diagonal moves, rather the usual four. How much better can you do with extra actions? Can you do even better by including a ninth action that causes no movement at all other than that caused by the wind?

You would be able to do much better if diagonal moves were an option. The wind causes a side-ways movement to become an upward diagonal. To counteract this, the agent could move at downward angle- allowing for the agent to stay more on path.

The ninth action would not help because it is -1 reward per move. The agent need to move to the right- not upward as the wind would make it move. Therefore, more negative points would be gained when they do not need to be accumulated with this ninth action.

Exercise 6.10

Stochastic Wind: Re-solve the windy gridworld task with King's moves, assuming that the effect of the wind, if there is any, is stochastic, sometimes varying by 1 by mean values given for each column. That is, a third of the time you move exactly according to these values, as in the previous exercise, but also a third of the time you one cell above that, and another third of the time you move one cell below that. For example, if you are one cell to the right of the goal and you move $left$*, then one-third of the time you move one cell above the goal, one-third of the time you move two cells above the goal, and one-third time you move to the goal.*

As before, the diagonal moves would be useful for counteracting the wind. However, the wind is a little more unpredictable so it might be harder to decide which diagonal to use to counteract the wind.

In this example, the ninth move of staying put and letting the wind move you is a useful move. There is a one third chance of being moved to the goal from some positions. Any move in addition to the wind movement could cause the agent to overshoot the goal instead of perfectly landing on the goal.

Exercise 6.11

Why is Q-learning considered an off-policy control method?

On-Policy only decides the action based off of the current action value, and updates the current action value based off a ε-greedy policy. However, the off-policy chooses actions and states off of two different policies. The states are chosen from an ε-greedy policy derived from Q. However, the action that updates the action value comes from choosing the action that gives the maximum action value. The next state is then chosen deterministic ally with an on-policy, but the state estimate update is always greedy.

Exercise 6.12

Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as Sarsa? Will they make exactly the same action selections and weight updates?

If the action selection in Sarsa is always greedy for A', then the algorithm would be basically be the same. In both cases, the next maximum action value will be chosen.

Exercise 6.13

**(Challenge Problem) What are the update equations for Double Expected Sarsa with an ε-greedy target policy?*

When 50/50:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_2(S_{t+1}, \underset{a}{argmax}(S_{t+1}, a)) - Q_1(S_t, A_t)]$$

Switch the one and the two for Q_1 and Q_2. Then since the algorithm is ε-greedy, the two action-state values can be averaged.

$$Q_1(S_t, A_t) = \frac{Q_1(S_t, A_t) + Q_2(S_t, A_t)}{2}$$

Exercise 6.14

Describe how the task of Jack's Car Rental (Example 4.1) could be reformulated in terms of after-states. Why, in terms of this specific task, would such a reformulation be likely to speed convergence?

In the original Jack's Car Rental scenario, an action was chosen based on the current state on how many cars to move between locations. This choice was based off a deterministic policy that determined the best action given the current state. This policy was updated by being in the current state and looking ahead at the next state options to determine the next best deterministic move.

Afterstates, however, look at state values after the action has been taken. Instead of looking how good a state might possible be from current state, it will move forward and then assess how good that actual state is.

To change the Jack's Car Rental scenario, the afterstate values would be assigned to the previous state after experiencing the updated state and action. These values would look something like action values, but they would not look ahead at all values for each action when updating. So instead of using $V(s) \leftarrow \sum_{S',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$ to look ahead, we would move to s' state, and then update the s state. This would help convergence because now there is less uncertainty on what will happen between the actions and the next state. We know probabilities but we do not know the exact physics of the model.