Attempts to exercise in Reinforcement Learning book Chapter 6

Mengliao Wang

August 1, 2017

Exercise 6.1:

When we add the time stamp to formula 6.5, we would have:

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

And also we have the equivalence for formula 6.2:

$$\begin{aligned} V_{t+1}(S_t) &= V_t(S_t) + \alpha \left[R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)\right] \\ &= V_t(S_t) + \alpha\delta_t \end{aligned}$$

To calculate the difference between $G_t - V(S_t)$ and $\sum_{k=0}^{T-t-1} \gamma^k \delta_{t+k}$, combining the two equations above we have:

$$\begin{aligned} \Delta &= G_t - V_t(S_t) - \sum_{k=0}^{T-t-1} \gamma^k \delta_{t+k} \\ &= G_t - V_t(S_t) - [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)] - \gamma[R_{t+2} + \gamma V_{t+1}(S_{t+2}) - V_{t+1}(S_{t+1})] \\ &- \gamma^2[R_{t+3} + \gamma V_{t+2}(S_{t+3}) - V_{t+2}(S_{t+2})] - ... - \gamma^{T-t-1}[R_T + \gamma V_{T-1}(S_T) - V_{T-1}(S_{T-1})] \\ &= G_t - V_t(S_t) - \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} - V_t(S_t) + \gamma[V_{t+1}(S_{t+1}) - V_t(S_{t+1})] \\ &+ \gamma^2[V_{t+2}(S_{t+2}) - V_{t+1}(S_{t+2})] + ... + \gamma^{T-t-1}[V_{T-1}(S_{T-1}) - V_{T-2}(S_{T-1})] - \gamma^{T-t} V_{T-1}(S_T) \\ &= \gamma\alpha\delta_{t+1} + \gamma^2\alpha\delta_{t+2} + ... + \gamma^{T-t-1}\alpha\delta_{T-1} - \gamma^{T-t} V_{T-1}(S_T) \\ &= \alpha \sum_{k=1}^{T-t-1} \gamma^k \delta_{t+k} \end{aligned}$$

Exercise 6.2:

If we move to a new place, and drive from work to the new house will share the same route with the old house, but started to go differently later after the highway. Then for the first few days, Monte Carlo approach will show us the value with a much bigger difference for each state, since the new total expected time at each state is unknown even though we know well enough for the first half of route. Even if the total expected time for each state is only shifted for a few minutes, Monte Carlo is learning the whole return until termination, thus would not take the learnt experience into consideration.

However, for TD method, since it will only update the current state value based on the next state value, at least on the first few days the first half of the route will not see much update. Each state will get much less error on the first half initially.

Exercise 6.3:

The fact that only $V(A)$ gets updated after the first episode means the first episode ends up terminating on the left side. Since all the states have value initiated to be 0.5 and reward is alway zero unless terminated on the right side, the update $\alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$ will always be zero until termination. So no state value will be updated except for the one right before left terminate, which is A. When it terminates, we will see $V(A) = 0.5 + 0.1 \times (0 + 0 - 0.5) = 0.45$ as shown in Figure 6.2 left.

Exercise 6.4:

If we choose a bigger α for TD approach, it might get worse eventually than the MC approach with a smaller α, since we require the α to be small enough for the algorithm to converge. When α increases, both MC and TD approaches will update faster initially, but might not converge to the correct answer. When α decreses, it will update slower at the beginning, but converging to a better answer. So in the longer term, a smaller α will provide a better result for both approaches.

Exercise 6.6:

The first approach would be using Bellman optimal equation, and we just need to solve the following equations:

$$
\begin{aligned}
V(A) &= 0.5 \times 0 + 0.5 \times V(B) \\
V(B) &= 0.5 \times V(A) + 0.5 \times V(C) \\
V(C) &= 0.5 \times V(B) + 0.5 \times V(D) \\
V(D) &= 0.5 \times V(C) + 0.5 \times V(E) \\
V(E) &= 0.5 \times 1 + 0.5 \times V(D)
\end{aligned}
$$

The second approach would be for each state, iterate through all the possible scenarios, and sum up the reward given the probability distribution for each scenarios. For example, for $V(E)$ we would need to sum all the probabilities that it will terminate on the right hand side: $V(E) = \frac{1}{2} + \frac{1}{2} \times (\frac{1}{2})^2 + 2 \times \frac{1}{2} \times (\frac{1}{2})^2 \times (\frac{1}{2})^2 + ... = \frac{5}{6}$. Obviously we followed the first approach because it is more straight forward to understand and calculate.

Exercise 6.7:

If we define $\rho_{t:t} = \dfrac{\pi(A_t|S_t)}{b(A_t|S_t)}$, then we have the update rule for TD(0) approach using important sampling:

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

Exercise 6.8:

Similarly for action values we have the following:

$$
\begin{aligned}
G_t - Q(S_t, A_t) &= R_{t+1} + \gamma G_{t+1} - Q(S_t, A_t) + \gamma Q(S_{t+1}, A_{t+1}) - \gamma Q(S_{t+1}, A_{t+1}) \\
&= \delta_t + \gamma [G_{t+1} - Q(S_{t+1}, A_{t+1})] \\
&= \delta_t + \gamma \delta_{t+1} + \gamma^2 [G_{t+2} - Q(S_{t+2}, A_{t+2})] \\
&= \delta_t + \gamma \delta_{t+1} + ... + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} [G_T - Q(S_T, A_T)] \\
&= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k
\end{aligned}
$$

Exercise 6.9:

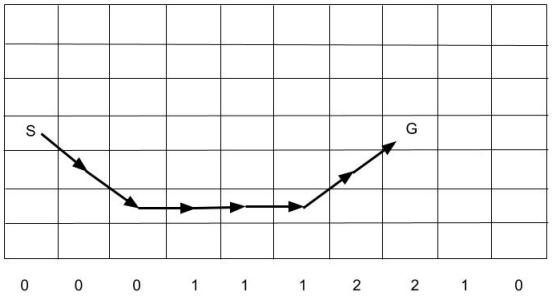

Figure 1: The solution for windy gridworld task with King's moves

The solution for windy gridworld task with King's moves is shown in Figure 1. There won't be any improvement if the ninth action (no movement) is added as it will never be part of the optimal trajectory.

Exercise 6.10:

The optimal policy is to keep moving right until we reach the destination column. During the path if we are blown away from the center line then we should move in diagonal back towards the center line. After we reach the destination column, we can move vertically towards the destination.

Exercise 6.11:

Q-learning is considered as an off-policy control method because it samples and learns the value following two different policies. To be more specific, it collects the sample training data based on any soft policy (ϵ-greedy here) which is the behavior policy, and then update the action-value based on the greedy policy which is the target policy, i.e. $\max_a Q(S', a)$.

Exercise 6.12:

The update equation for Double Expected Sarsa can be written as follows:

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

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

Exercise 6.13:

It will be natural to implement the Jack Car Rental problem as an afterstate problem. Here instead of defining the state as the car numbers at the end of the day (before we take the actions), we should define it as the car numbers at the beginning of the day (after we take the actions, i.e. move the cars overnight). In this case the action to state transition is explicit and can be calculated with basic algebra. Modeling the problem as afterstate does not only make the implementation easier, but also can speed up the learning speed for state values even though these new states will converge to a different value since the state definition is different.