RL HW3:

Q1 Let $\{G_i\}$s be set of all the returns that corresponds to (S_t, A_t). To update $Q(S_t, A_t)$, we need the mean of $\{G_i\}$.

For first-visit monte carlo, at n^{th} episode

$$Q_{n+1}(S_t, A_t) \leftarrow \frac{1}{n} \sum_{i=1}^{n} G_i$$

s.t. $Q_1 = 0$ (or initialized arbitrarily) for all $s \in S$, $a \in A$

$$Q_{n+1}(S_t, A_t) = \frac{1}{n} \left(G_n + \sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n} \left(G_n + (n-1) \cdot \frac{1}{(n-1)} \sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n} \left(G_n + (n-1) Q_n(S_t, A_t) \right)$$

$$Q_{n+1}(S_t, A_t) = Q_n(S_t, A_t) + \frac{1}{n} \left[G_n - Q_n(S_t, A_t) \right]$$

↑ in this equation, we just need the count of no. of times pair (S_t, A_t) appears and past estimate of $Q(S_t, A_t)$ and current return.

⇒ we don't need to keep track of past returns.
⇒ we don't need to keep a list of all returns to update $Q(S_t, A_t)$. Instead, we just need to keep track of no. of times a pair appears (which is just a no.)

Hence, new update rule for Q would give us the same estimate as given by the update rule in the code (book)

Monte Carlo ES, for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in A(s)$ (arbitrarily), $\forall s \in S$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), $\forall s \in S, a \in A(s)$

$count(s,a) = 0 \quad \forall s \in S, a \in A(s)$

// count(s,a) keeps count of no. of times pair (s,a) appears.

Loop forever (for each episode):

Choose $S_0 \in S$, $A_0 \in A(S_0)$ randomly s.t. all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$.

$G \leftarrow 0$

loop for each step of episode, $t = T-1, T-2, \ldots 0$

$G \leftarrow \gamma G + R_{t+1}$

$count(S_t, A_t)++$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{count(S_t, A_t)} \left[G - Q(S_t, A_t)\right]$$

$$\pi(S_t) \leftarrow \underset{a}{\operatorname{argmax}}\, Q(S_t, a)$$

Q3 Let the set of all time-steps in which pair (s,a) is visited, denoted by $\mathcal{T}(s,a)$

$$Q(s,a) \doteq \frac{\sum_{t \in \mathcal{T}(s,a)} \rho_{t+1:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s,a)} \rho_{t+1:T(t)-1}}$$

Teacher's Signature

where $T(t)$ is the first time of termination following time t., and G_t is the return after t up through $T(t)$

$\therefore$ $\{G_t\}_{t \in \mathcal{T}(s,a)}$ are the returns that pertain to the state s and action a.

and, $\{\rho_{t+1:T(t)-1}\}_{t \in \mathcal{T}(s,a)}$ are corresponding importance sampling ratios.

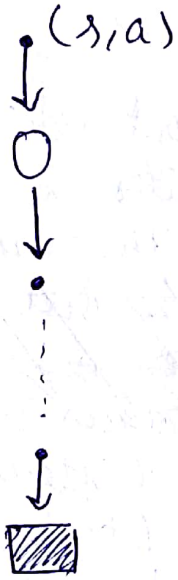
$$\left[\rho_{t+1:T-1} = \prod_{k=t+1}^{T-1} \frac{\pi(A_k | S_k)}{b(A_k | S_k)} \right] \quad \text{b/c } \pi(A_t | S_t) = 1 \text{ as } a \text{ is chosen w/o policy.}$$

Q1 Exercise 6.12

If action selection is greedy, Q-learning is not same as Sarsa. Because Sarsa chooses next action acc. to q-value and then updates the qvalues. In q-learning, it first updates the qvalue and then, in the next time-step, it choose the action acc. to updated qvalue (which might not be same as action chosen by Sarsa).

Q2 The backup diagram for Monte Carlo estimated of q_π.

Root node represents (state, action) pair, and below that is the entire trajectory of transitions along a particular single episode, ending at the terminal state.



Experiment Number

Date __/__/__

Page

Q5 Exercise 6.2

Since only initial route has change b/c of the new start state. Some of the states would be same as the original problem. Given that we have lots of experience driving home from work and that TD bootstraps, TD would perform better b/c state-value for common states can be used in new problem and b/c they would be close to true values, and starting with state-value estimate close to true-values would result in faster convergence as TD bootstraps. Yes, the same sort of thing would happen in the original scenario if initial state-values estimate is close to true-values.

Q6

Exercise 6.3: $\alpha = 0.1$, $\gamma = 1$

$$V(S_t) = V(S_t) + 0.1\,[R_{t+1} + V(S_{t+1}) - V(S_t)]$$

All the transitions from $S_t \to S_{t+1}$ s.t S_{t+1} is not a terminal state gives 0 reward.

$$\therefore V(S_t) = V(S_t) + 0.1\,[0 + V(S_{t+1}) - V(S_t)]$$

But we initialized $V(s) = 0.5 \quad \forall s \in S$ (excluding terminal states)

$$\therefore V(S_t) = V(S_t) \quad \forall s \in S \quad //$$

$V(s) = 0$ for $s \in$ terminal states

In first episode, when we ended up in left terminal state from 'A', we received a reward 0.

$V(A) = V(A) + 0.1 [0 + 0 - V(A)]$
$V(A) \leftarrow 0.9\,V(A)$
$V(A) \leftarrow 0.9 \times 0.5$
$V(A) \leftarrow 0.45$

∴ By the end of episode 1, only V(A) got affected by 0.9 times which resulted in decreasing V(A) by 0.05.

Exercise 6.4:

Step-size α tells you how much sensitive are you to reward received at each time-step. Greater the α, more the sensitivity. Smaller α are less sensitive towards rewards received at each time-step.
Step-size

NO, the conclusions about which algorithm is better would be affected if a wider range of α values were used in terms of RMSE with episodes. ~~also affects rate of convergence. But no matter what α you pick, all the algo both the algorithm converge to the state values (given you run it till infinity)~~
B/c smaller alpha (step-size) means slow learning but less RMSE (b/c oscillations don't occur).
There is not different fixed alpha at which either of the algorithm would have performed better b/c alphas are already very small and smaller would not give a significantly better result than shown.

Exercise 6.5:

In TD, update rule for value function can be seen as gradient descent update rule. As step-size alpha defines how sensitive you are towards new target. Once convergence is reached (optimal value function), higher values of α (step-size), makes the value function oscillate around optimal value function. No, it is not the function of how the value function was initialised