Reinforcement Learning

Assignment - 3

Madhav Thakker

2016159

Q1. pseudo code of Monte Carlo E.S.

Initialize

$N(s,a) \leftarrow 0 \quad \forall s, a$

$\pi(s) \in A(s)$

$Q(s,a) \leftarrow 0 \quad \forall s, a$

Loop forever (for each episode):

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

Generate episode $S_0, A_0, R_1, \ldots S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

for step $t = T-1, T-2, \ldots 0$:

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

If pair (S_t, A_t) does not appear in $S_0, A_0, \ldots S_{t-1}, A_{t-1}$:

$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$.

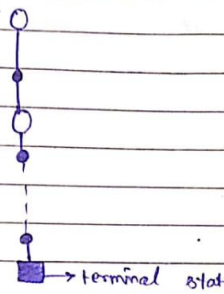
(1) ——— $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left[G - Q(S_t, A_t) \right]$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$.

This is the pseudocode for Monte Carlo E.S. with mean and count of each action pair.

This is same as the book's version because the line -(1) does exactly same as appending and taking average.

Q2. Backup-diagram for Monte-Carlo q_π.



M.C. has one choice at each state.

It does not bootstrap.

Q3. Eqn analogous to (5.6) for action values $Q(s,a)$.

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} S_{t:T(t)-1} \, G_t}{\sum_{t \in \mathcal{T}(s)} S_{t:T(t)-1}} \quad \rightarrow (5.6)$$

The equation will remain the same for $Q(s,a)$ as well because at each state in M.C. we have only one action to choose from.

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

Q5) Exercise 6.2

TD-Learning would be much better than MC when we move to a new building. When we move to a new building, only the initial states from the new building will change, many states will still remain the same. For eg, after entering the highway the state value function would be almost similar for both the old and the new building. Because we had

lots of experience from highway, our estimate of the highway state is good. TD can take advantage of it; will result in faster convergence.
In the original scenario, TD would be better than MC if our initial guess of the value function is close to the original value function itself.

Q8) Exercise 6.12

Not exactly. It is because, when we look at pseudo-code of Sarsa we see that we first select s', a' and then we update our Q(s, a) function, our s' and a' become s and a respectively then.
But in the case of Q-Learning, we only select s', we choose a from Q(s, a) using epsilon-greedy.
Basically, Sarsa selects from Q(s, a) and then updates. But Q-Learning updates Q(s, a) and then selects the next action. So they are not exactly equivalent.