Reinforcement Learning – CSE564

Assignment 3

Suchet Aggarwal
2018105

1. Pseudo Code:

Initialize:

π(s) in A(s) (arbitrarily), for all s in S
Q(s, a) in R (arbitrarily), for all s in S, a in A(s)
counts(s, a) = 0, for all s 2 S, a 2 A(s)

Loop forever (for each episode):

Choose S_0 in S, A_0 in A(S_0) randomly such that all pairs have probability > 0
Generate an episode from S_0,A_0, following π: S_0,A_0,R_1, . . . ,S_{T-1},A_{T-1},R_T
G <- 0
Loop for each step of episode, t = T −1, T −2, . . . , 0:

G <- G + R_{t+1}
Unless the pair S_t,A_t appears in S_0,A_0, S_1,A_1 . . . , S_{t-1},A_{t-1}:

counts(s, a) <- counts(s, a) + 1
Q(S_t,A_t) <- Q(S_t,A_t) + (G – Q(S_t,A_t))/counts(s, a)
π(S_t) <- argmax$_a$ Q(S_t, a)

Here count stores the number of times the pair of state action has been considered for every state s and every action a.

Explanation:

From Chapter 2.4:

$$Q_{n+1} = \frac{\sum_{i=1}^{n} R_i}{n}$$

$$Q_{n+1} = \frac{R_n + \sum_{i=1}^{n-1} R_i}{n}$$

$$Q_{n+1} = \frac{R_n + \sum_{i=1}^{n-1} R_i}{n}$$

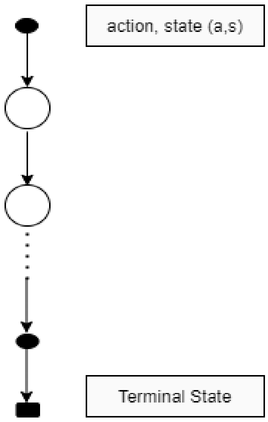
$$Q_{n+1} = \frac{R_n}{n} + \frac{(n-1)\sum_{i=1}^{n-1} R_i}{(n-1)*n}$$

$$Q_{n+1} = \frac{R_n}{n} + \frac{(n-1)Q_n}{n}$$

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$$

Where Q_n is the mean value for n steps and n is the number of steps

2.



The Backup diagram for Monte Carlo would remain same for Q, as the diagram for V. As for computing the value Q(s, a) we compute it by generating an episode, and then update the value for pair s, a using the transitions within that episode.

3. For Q, the first action is fixed for a given state action pair. Thus the important sampling factor becomes:

   The probability of state action trajectory under the policy π, starting in state S_t and taking action A_t

$$P(A_{t+1}, S_{t+1}, ... A_{T-1}, S_T \mid S_t, A_t) \\ = p(S_{t+1} \mid S_t, A_t)\pi(A_{t+1}|S_{t+1}). . . p(S_{t+1} \mid S_t, A_t)$$

   Therefore, the important sampling factor becomes:

$$factor_{t:\mathrm{T}-1} = \frac{p(S_{t+1}|S_t, A_t) \prod_{k=t+1}^{T-1} p(S_{k+1}|S_k, A_k)\pi(A_k|S_k)}{p(S_{t+1}|S_t, A_t) \prod_{k=t+1}^{T-1} p(S_{k+1}|S_k, A_k)b(A_k|S_k)}$$

$$factor_{t:\mathrm{T}-1} = \frac{\prod_{k=t+1}^{T-1} \pi(A_k|S_k)}{\prod_{k=t+1}^{T-1} b(A_k|S_k)}$$

$$factor_{t:\mathrm{T}-1} = \rho_{\mathrm{t+1:\,T-1}}$$

Thus the Q estimate becomes:

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

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

4. Attached in ipynb.

5. In the Driving home example, suppose we have accurate estimates for how long it takes to get Home (H_1) from Work (W), Now if one moves to a newer location (H_2), we have the following:
   From the past experience we know V(W), now if we change the final destination to (H_2), such that the initial journey still goes through the highway, then using Monte Carlo, we estimate V for all states s, by generating the entire episode again. Now since the updates are for the entire episode, the estimates for the initial part of the journey i.e. from W to Highway, also is recomputed, despite being the same.
   When using Temporal Difference, the updates for each step, do not alter the estimates for the initial states (from W to Highway) which can essentially be used to estimate the value function V, for the newer states.
6. Attached in ipynb.
7. Attached in ipynb.
8. Even when the action selection is made greedy, it is possible that the weight updates and the corresponding action sequences are different.
   Let A_{sarsa} and A_Q, denote the actions picked by SARSA and Q-Learning respectively, for the state S_{t+1}

   In the case when the states S_t and S_{t+1} are different, then

$$A_{SARSA} = argmax_a\, Q(S_{t+1}, \mathrm{a})$$

$$Q(\mathrm{S_t}, \mathrm{A_t}) = \mathrm{Q}(\mathrm{S_t}, \mathrm{A}_t) + \alpha[\mathrm{R} + \mathrm{Q}(\mathrm{S_{t+1}}, \mathrm{A_{SARSA}}) - \mathrm{Q}(\mathrm{S_t}, \mathrm{A_t})]$$

$$\mathrm{A_Q} = \mathrm{argmax}_a\, \mathrm{Q}(\mathrm{S_{t+1}}, \mathrm{a})$$

$$\mathrm{Q}(\mathrm{S_t}, \mathrm{A_t}) = \mathrm{Q}(\mathrm{S_t}, \mathrm{A}_t) + \alpha[\mathrm{R} + \mathrm{max_a}\mathrm{Q}(\mathrm{S_{t+1}}, a) - \mathrm{Q}(\mathrm{S_t}, \mathrm{A_t})]$$

Here $\mathrm{max_a}\mathrm{Q}(\mathrm{S_{t+1}}, a) = \mathrm{Q}(\mathrm{S_{t+1}}, \mathrm{A_{SARSA}})$

Thus the two updates and actions are the same

   But when S_t and S_{t+1} are the same, then,

   The action A_{sarsa} and A_Q, can be different, as A_{sarsa} is picked before the Q value is updated, i.e. $\mathrm{Q}(\mathrm{S_t}, \mathrm{A_t})$, while for Q-learning the action is picked in the beginning of the next step of the episode. This way A_Q is essentially the new greedy choice after the update, while A_{SARSA}, was the greedy choice before the update was made, and hence they may not be the same, and as a result the following state action sequence may be different leading to different weight updates as well.