Assignment 3.

Neethu R.R
PhD 19108

Reinforcement Learning

(1) The incremental implementation is given as

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

Where Q_{n+1} = estimate of its action value after it has been selected n times.

$$\therefore \quad Q_{n+1} = Q_n + \frac{1}{n}\left[R_n - Q_n\right] \quad \Rightarrow R_n = n^{th} \text{ reward.}$$

The pseudo code given in the book is in efficient because, for each state-action pair, it maintains a list of all returns and repeatedly calculates their mean.

So the efficient code can be written using the concept of incremental implementation.

Also we can write that

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

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

Pseudocode:-

Initialize:

$\pi(s) \in A(s) \quad \forall s \in S$

$Q(s,a) \in R \quad \forall s \in S, a \in A(s)$

$Returns(s,a) \leftarrow$ empty list $\forall s \in S, a \in A(s)$

$n(s,a) \leftarrow 0 \quad \forall s \in S, a \in A(s)$

Loop forever:

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: $S_0, A_0, R_1, \ldots$

S_{T-1}, A_{T-1}, R_T.

$G \leftarrow 0$

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

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

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

$n(S_t, A_t) = n(S_t, A_t) + 1$.

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

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

OR

Instead of using (1), we can write in terms of rewards.

ie $$\text{Returns}(S_t, A_t) \leftarrow \text{Returns}(S_t, A_t) + \frac{(G - \text{Returns}(S_t, A_t))}{n(S_t, A_t)}$$

$$Q(S_t, A_t) \leftarrow \text{Returns}(S_t, A_t).$$

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

In both the cases, Monte Carlo ES is satisfied because the policy improvement is acheived at all the states visited in the episodes.

Instead of using more space to store all the return values to find the mean, an alternate formula is used to find the optimal policy π. using incremental implementation

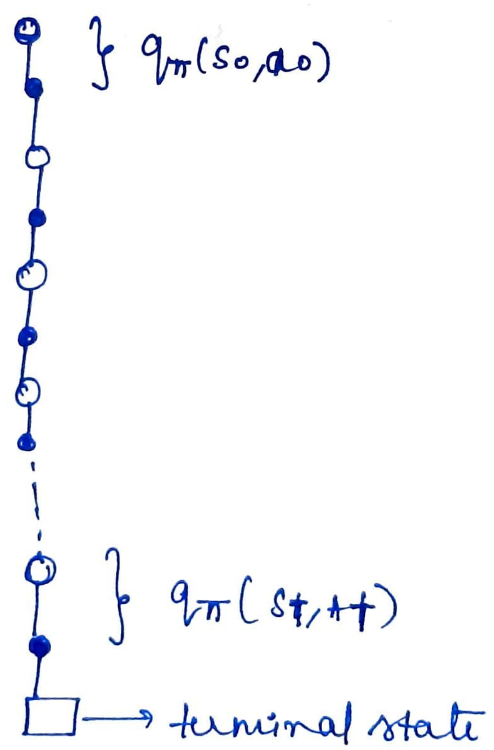
② Back up diagram is the graphical representation of algorithm by representing state, action, state transition, reward etc. Value function is transferred back to a state from its successor state or state action.

○ → State value

● → State – action value

⫯ → Action.

∴ the backup diagram for monte carlo for q_π is.



(2) Given a starting state S_t, the probability of the subsequent state-action trajectory, $A_t, S_{t+1}, A_{t+1} \dots S_T$, occuring under any policy π is

$$\Pr\{A_t, S_{t+1}, A_{t+1}, \dots S_T \mid S_t, A_{t:T-1} \sim \pi\}$$

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

$$= \prod_{k=t}^{T-1} \pi(A_k|S_k)\, p(S_{k+1}|S_k, A_k)$$

where p is the state-transition probability function. The importance sampling ratio is

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

b = Behavioural policy π = Target policy.

The weighted importance sampling, which uses a weighted average is

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

The state-action value $Q(s,a)$ can be written as

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

where $T(s,a)$ is the state action pair of all time steps.

~~(4) Blackjack game.~~
~~fig 5.1 → code ⇒ Qs_4_fig5_1.m~~
~~fig 5.2 → code ⇒ Qs_4_fig5_2.m.~~

(5) Exercise 6.2.

In the scenario discussed, for the first drive, TD will be more efficient. TD will start updating the states immediately ~~after~~ when the person starts driving. Each updated state accelerate the one prior to it. But in Monte Carlo, the updates are done only after the journey. There will not be immediate update. MC bring an average improvement of all the states which are not even needed.

After the first drive, TD will use the past estimates to update the new states which will be efficient in this type of scenario.

(6) Code: Qs_6_1.m & Qs_6_2.m
Figure: Qs_6_left.png & Qs_6_right.png

6.3) In the figure, for the first episode, the state is at E. The value update for the state is that all states are initialized to ~~zero~~ 0.5.

$$V(A) = V(A) + \alpha\left[R - 0.5\right] = 0.5 + 0.1\left[0 - 0.5\right]$$

$$= \underline{0.45}$$

As reward = 0 & gamma = 1, the state changes its values to non-zero at A ~~or~~ E.

6.4) Using a large value of α will not effect the results. Small alpha values are better for convergence in TD & MC methods. Using higher values of alpha leads to noise ie the previous outcomes changes the values of the states. There occur an error which is undesirable.

But for small values of alpha, the algorithm converge slowly but in an efficient manner ie without noise.

As they converge slowly, it is not that small α is better that high alpha, But it produces an efficient in a long run and the error will be very less.

6.5) Yes, I think it is because of the initialization of approximate value functions. The identical initialization of all the states, causes a sudden shoot at many states close to each other at the same time. If same initialization is done, we have to simultaneously check for estimates more than some states and less than some other states. This will cause a sudden decrease of RMS error. and RMS error rise eventually

(8) Even for greedy policy, Q-learning & SARSA algorithms will be different. The main difference is in SARSA, the ~~optimal~~ greedy action is used to update the state-action value of that state. But in Q-learning, the updated state action value of that state is considered to rechoose the greedy action for the next state.

In action selection, Q learning is off-policy and use any behavioural policy for action selection. But ~~in~~ SARSA is on-policy and uses greedy action selections where exploration is not done.

(4) Black jack game.

Figure 5.1 :-

Code :- Qs_4_fig5_1.m.

figure :- Qs_4_fig5_1.png

Qs_4_fig5_1_1.png

Qs_4_fig5_1_2.png

Qs_4_fig5_1_3.png.

Figure 5.2 :-

Code :- Qs_4_fig5_2.m.

figure :- Qs_4_fig5_2.png.

Qs_4_fig5_2_1.png.