RL Homework 3

Submitted By:
Snehal Gupta
2016201

Q1

HW 3

Q1 We know that,

$$Q_{n+1}(S_t, A_t) = \frac{1}{n} \sum_{i=1}^{n} G_i \quad \text{(At n-th episode)}$$

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

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

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

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

This is how we can update the state-action values for each pair incrementally. We just need to maintain the past estimate $Q(S_t, A_t)$, the current return G_n and a count for each state-action pair.

Pseudocode

Initialize:

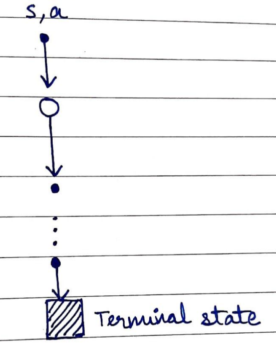
$\pi(s) \in A(s)$, (arbitrarily), for all $s \in S$
$Q(s,a) \in R$ (arbitrarily), for all $s \in S$, $a \in A(s)$
$count(s,a) = 0$ for all $s \in S$, $a \in A(s)$

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: $S_0, A_0, R_1, \ldots S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $count(S_t, A_t) = count(S_t, A_t) + 1$
  - Unless the pair (S_t, A_t) appears in $S_0, A_0, S_1, A_1, \ldots S_{t-1}, A_{t-1}$:
    - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{count(S_t, A_t)}\left[G - Q(S_t, A_t)\right]$
    - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

ortant Notes:

Q2

Q2 Backup diagram for Monte Carlo estimation of q_π:



Q3

<u>Q3</u>

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

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

Equation analogous to given equation for action values $Q(s,a)$:

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

$$= \frac{\sum_{t \in \mathcal{J}(s,a)} \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} G_t}{\sum_{t \in \mathcal{J}(s,a)} \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}}$$

Given state at time t $S_t = s$

action at time t $A_t = a$

$\Rightarrow \pi(A_t = a | S_t = s) = 1$

$b(A_t = a | S_t = s) = 1$

$$= \frac{\sum_{t \in \mathcal{J}(s,a)} \prod_{k=t+1}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} G_t}{\sum_{t \in \mathcal{J}(s,a)} \prod_{k=t+1}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}}$$

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

Q4

Figure 5.1

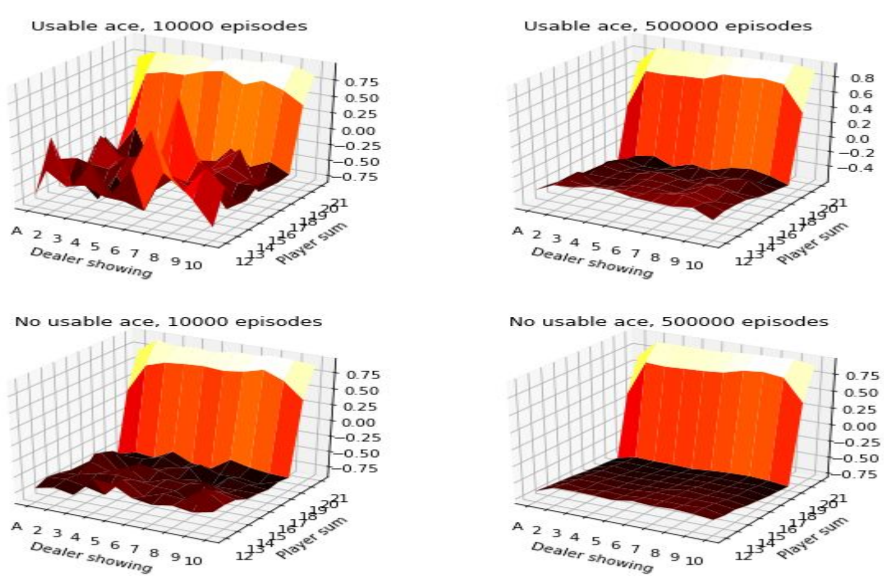


Figure 5.2

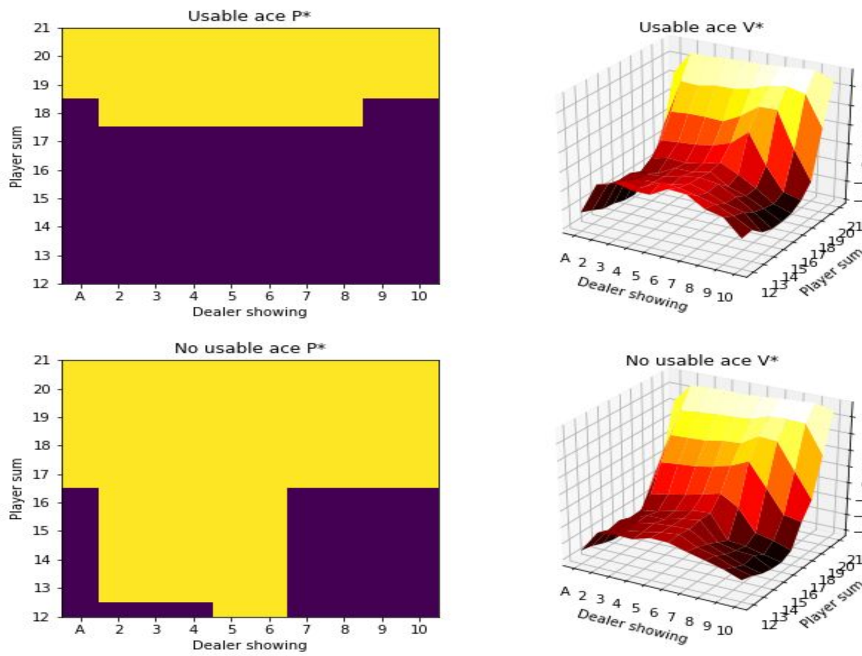
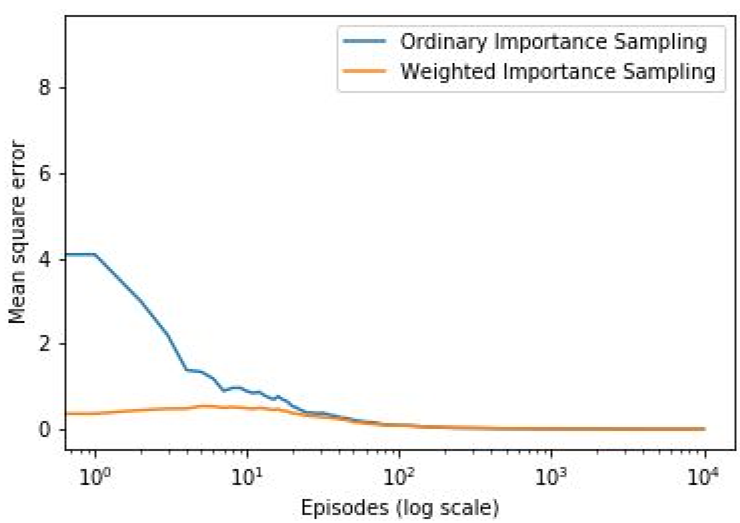


Figure 5.3



Q5

Q5

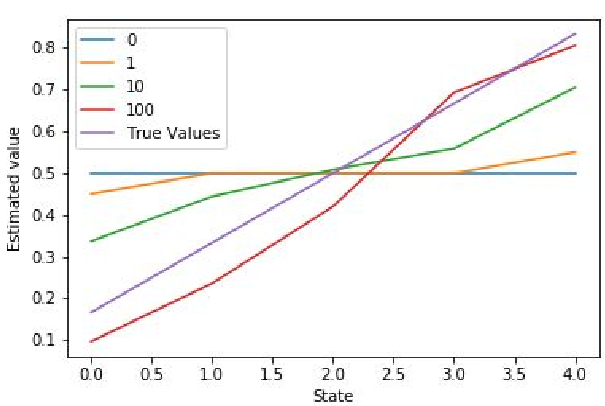
Since, we have lots of experience driving home from work, and TD updates incorporate prior information, TD updates are likely to be much better in this case.

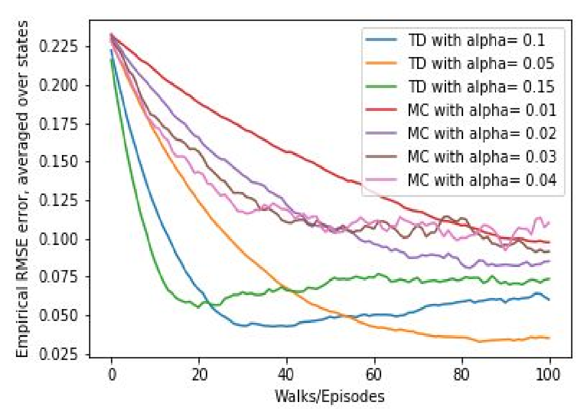
If we already have a good value estimate for a trajectory $J = S_1, S_2, \ldots S_T$, it would be ~~diff~~ possible to use MC only if we see multiple episodes to get a good estimate of $V(S_0)$, and not use the info on J. TD would use T to back up the value of S_0 and hence converge much quicker (bootstrapp

Important Notes: -ing).

Q6

Figures generated





Ex 6.3

Q6

given: $\alpha = 0.1$, $\gamma = 1$

As we know, in TD(0),

$V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

and all the states except the terminal state were initialized to 0.5 and reward for non-terminal state is 0, so, all the transitions from $S_t \rightarrow S_{t+1}$ s.t. S_{t+1} is not a terminal ~~is~~ gives 0 reward.

In the first episode, the agent terminated on the left.

so,

$V_1(A) \leftarrow V_0(A) + 0.1 [0 + 0 - V_0(A)]$

$V_1(A) \leftarrow 0.9\, V_0(A)$

$V_1(A) \leftarrow 0.9 \times 0.5$

$V_1(A) \leftarrow 0.45$

$V_1(A)$ was updated to $(1-\alpha) V_0(A)$

Hence,

the value of the estimate changed by

$\alpha V_0(A) \rightarrow 0.1 \times 0.5 = 0.05$.

Ex 6.4

Ex 6.4

[redacted], No the conclusions about which algorithm is better would be affected if a wider range of α values were used. When we increase α, it makes the curve more. on the other hand, decrease in α makes the curve more smooth and make it converge slower.

Basically, α parameter controls the sensitivity towards the reward received at each time step and hence, controls the speed of convergence. Greater the alpha, more the sensitivity and speed of convergence. This is because smaller alpha would result in less RMSE (less oscillations). Since alphas are small, even a wider range of values won't help.

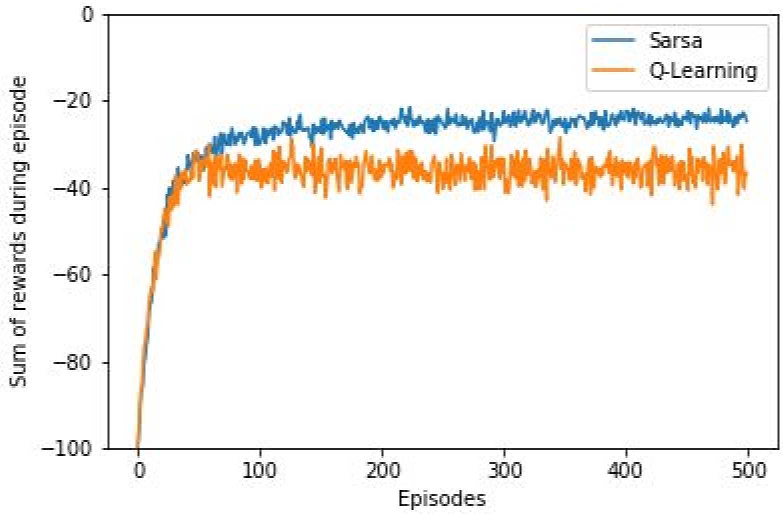
Ex 6.5

Ex 6.5

α-parameter controls how rapidly the value estimate changes. Higher α results in more rapid change in the value estimate. When the algorithm converges, the value function keeps on oscillating around optimal value function.

Yes, it might be a function of how the approximate value function was initialized. The state C happens to have been initialized with its true value. When the training starts, values of outer states get updated (more accurate), and, hence, the error decreases. This happens until the residual inaccuracies in the outer states propagate to C. After this, the curve goes up again. Higher alpha helps in observing this effect, since values ~~are~~ change rapidly.

Q7



Q8

Q8

If action selection is greedy, Q-learning won't make exactly the same action selections and weight updates. Hence, they won't be same.

This is since, in Q-learning,

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

and in sarsa,

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma Q(S',A') - Q(S,A)\right]$$

In Sarsa, we choose the next action according to Q-value (current) and then updates the Q-value. However, in Q-learning, we first update the q-value and then in the next time step, we choose the action according to the updated Q-value which can be different.