COMP 767 (Reinforcement Learning) Assignment 2

Arna Ghosh (260748358), Arnab Kumar Mondal (260906419)

February 2020

1 Theory Part

Answer 1. Part B

i. TD(k) recurrence relation

Let us define a trajectory from the MDP, M, as $\tau = (s_i, r_i)$, $i \in [0, T)$, where T is the length of the trajectory. We are given a set of such trajectories in phase t as denoted by $S(t)$. We can write the true value function of the MDP under policy π as:

$$\begin{aligned} v^{\pi}(s) &= \mathbf{E}_{\pi}[\sum_{l=0}^{k-1} \gamma^l r_l + \gamma^k v^{\pi}(s_k)] \\ &= \sum_{l=0}^{k-1} \gamma^l \mathbf{E}_{\pi}[r_l] + \gamma^k \mathbf{E}_{\pi}[v^{\pi}(s_k)] \end{aligned} \tag{1}$$

where r_l denotes the l^{th} reward in a trajectory starting from state s. γ denotes the discounting factor of the environment. Given a set of n trajectories, we can estimate the expected value of r_l as:

$$\gamma^l \mathbf{E}_{\pi}[r_l] = \frac{\gamma^l}{n} \sum_{i=1}^{n} r_l^i \tag{2}$$

Using Hoeffding inequality,

$$Pr(|\ \frac{1}{n} \sum_{i=1}^{n} x_l^i - \mathbf{E}_{\pi}[x_l]\ | \leq \epsilon) \geq 1 - 2e^{-2n\epsilon^2} \tag{3}$$

where $x_l^i \in [0, 1]$. Since, we know that every $r_l^i \in [-1, 1]$, we replace $r_l^i = 2x_l^i - 1$ in equation 3. This yields the following inequality:

$$\begin{aligned} & Pr(|\ \frac{1}{n} \sum_{i=1}^{n} r_l^i - \mathbf{E}_{\pi}[r_l]\ | \leq 2\epsilon) \geq 1 - 2e^{-2n\epsilon^2} \\ \implies & Pr(|\ \frac{1}{n} \sum_{i=1}^{n} r_l^i - \mathbf{E}_{\pi}[r_l]\ | \leq \epsilon) \geq 1 - 2e^{-\frac{n\epsilon^2}{2}} \end{aligned} \tag{4}$$

Applying inequality 4 for k rewards and thereafter using the union bound to estimate the probability that all rewards satisfy the inequality, we attain the following inequality:

$$Pr(|\ \frac{1}{n} \sum_{i=1}^{n} r_l^i - \mathbf{E}_{\pi}[r_l]\ | \leq \epsilon) \geq 1 - 2ke^{-\frac{n\epsilon^2}{2}} \tag{5}$$

Now, the error in estimation at phase t can be written as:

$$\bar{v}_{\pi,t}^k(s) - v^{\pi}(s) = \frac{1}{n} \sum_{i=1}^{n} (\sum_{l=0}^{k-1} \gamma^l r_l^i + \gamma^k \bar{v}_{\pi,t-1}^k(s_k^i)) - \sum_{l=0}^{k-1} \gamma^l \mathbf{E}_{\pi}[r_l] - \gamma^k \mathbf{E}_{\pi}[v^{\pi}(s_k)] \tag{6}$$

Since $\Delta_t = max_s(\mid \bar{v}^k_{\pi,t}(s) - v^\pi(s) \mid)$, difference between the estimated and true expectation of $\bar{v}^k_{\pi,t}(s)$ and $v^\pi(s)$ will always be less than Δ_t for all t. Following this,

$$\mid \bar{v}^k_{\pi,t}(s) - v^\pi(s) \mid \leq \sum_{l=0}^{k-1} \gamma^l \mid \frac{1}{n}\sum_{i=1}^{n} r_l^i - \mathbf{E}_\pi[r_l] \mid + \gamma^k \Delta_{t-1}$$

$$\implies \Delta_t \leq \sum_{l=0}^{k-1} \gamma^l \epsilon + \gamma^k \Delta_{t-1} \tag{7}$$

We could write this assuming that inequality 5 holds, i.e. we deal with the case when $\mid \frac{1}{n}\sum_{i=1}^{n} r_l^i - \mathbf{E}_\pi[r_l] \mid \leq \epsilon$. Therefore, we can simply write inequality 7 as:

$$\Delta_t \leq (\frac{1-\gamma^k}{1-\gamma})\epsilon + \gamma^k \Delta_{t-1} \tag{8}$$

which holds with probability $1 - 2ke^{-\frac{n\epsilon^2}{2}}$. Therefore, if we choose δ such that

$$\delta = 2ke^{-\frac{n\epsilon^2}{2}}$$

$$\implies \epsilon = \sqrt{\frac{2}{n} ln(\frac{2k}{\delta})} \tag{9}$$

we have the desired recurrence relation.

ii. TD(λ) recurrence relation

To prove this part, we will build upon the previous relation. As before, we can write the error in estimation at phase t as:

$$\bar{v}^\lambda_{\pi,t}(s) - v^\pi(s) = \frac{1}{n}\sum_{i=1}^{n}(1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}(\sum_{l=0}^{k-1}\gamma^l r_l^i + \gamma^k \bar{v}^k_{\pi,t-1}(s_k^i)) - \sum_{l=0}^{k-1}\gamma^l \mathbf{E}_\pi[r_l] - \gamma^k \mathbf{E}_\pi[v^\pi(s_k)]$$

$$Using, \quad (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1} = 1$$

$$\mid \bar{v}^\lambda_{\pi,t}(s) - v^\pi(s) \mid = (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}(\sum_{l=0}^{k-1}\gamma^l \mid \frac{1}{n}\sum_{i=1}^{n} r_l^i - \mathbf{E}_\pi[r_l] \mid + \gamma^k \mid \frac{1}{n}\sum_{i=1}^{n}\bar{v}^k_{\pi,t-1}(s_k^i)) - \mathbf{E}_\pi[v^\pi(s_k) \mid) \tag{10}$$

As before, simplifying the second term in the summation yields us the recurrence term:

$$\gamma^k \mid \frac{1}{n}\sum_{i=1}^{n}\bar{v}^k_{\pi,t-1}(s_k^i)) - \mathbf{E}_\pi[v^\pi(s_k) \mid \leq \gamma^k \Delta_{t-1}$$

$$\implies (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}(\gamma^k \mid \frac{1}{n}\sum_{i=1}^{n}\bar{v}^k_{\pi,t-1}(s_k^i)) - \mathbf{E}_\pi[v^\pi(s_k) \mid) \leq (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}\gamma^k \Delta_{t-1}$$

$$\implies (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}(\gamma^k \mid \frac{1}{n}\sum_{i=1}^{n}\bar{v}^k_{\pi,t-1}(s_k^i)) - \mathbf{E}_\pi[v^\pi(s_k) \mid) \leq \frac{(1-\lambda)\gamma\Delta_{t-1}}{1-\gamma\lambda} \tag{11}$$

In order to simplify the first term, we bound only the first k_0 rewards and assume maximum variance for the later rewards. Since each reward is bounded between -1 and 1, the maximum variance for each reward is 1. For the first k_0 rewards, we use the bound from previous part and denote it here as ϵ_1. Now, we write the first term in equation 10 as a sum of 3 terms with different limits on k and l. This split is illustrated in Figure 1.

$$\mathbf{A} \leq \sum_{k=1}^{k_0}(1-\lambda)\lambda^{k-1}\sum_{l=0}^{k-1}\gamma^l\epsilon_1 + \sum_{k=k_0+1}^{\infty}(1-\lambda)\lambda^{k-1}\sum_{l=0}^{k_0-1}\gamma^l\epsilon_1 + \sum_{k=k_0+1}^{\infty}(1-\lambda)\lambda^{k-1}\sum_{l=k_0}^{k-1}\gamma^l * 1 \tag{12}$$

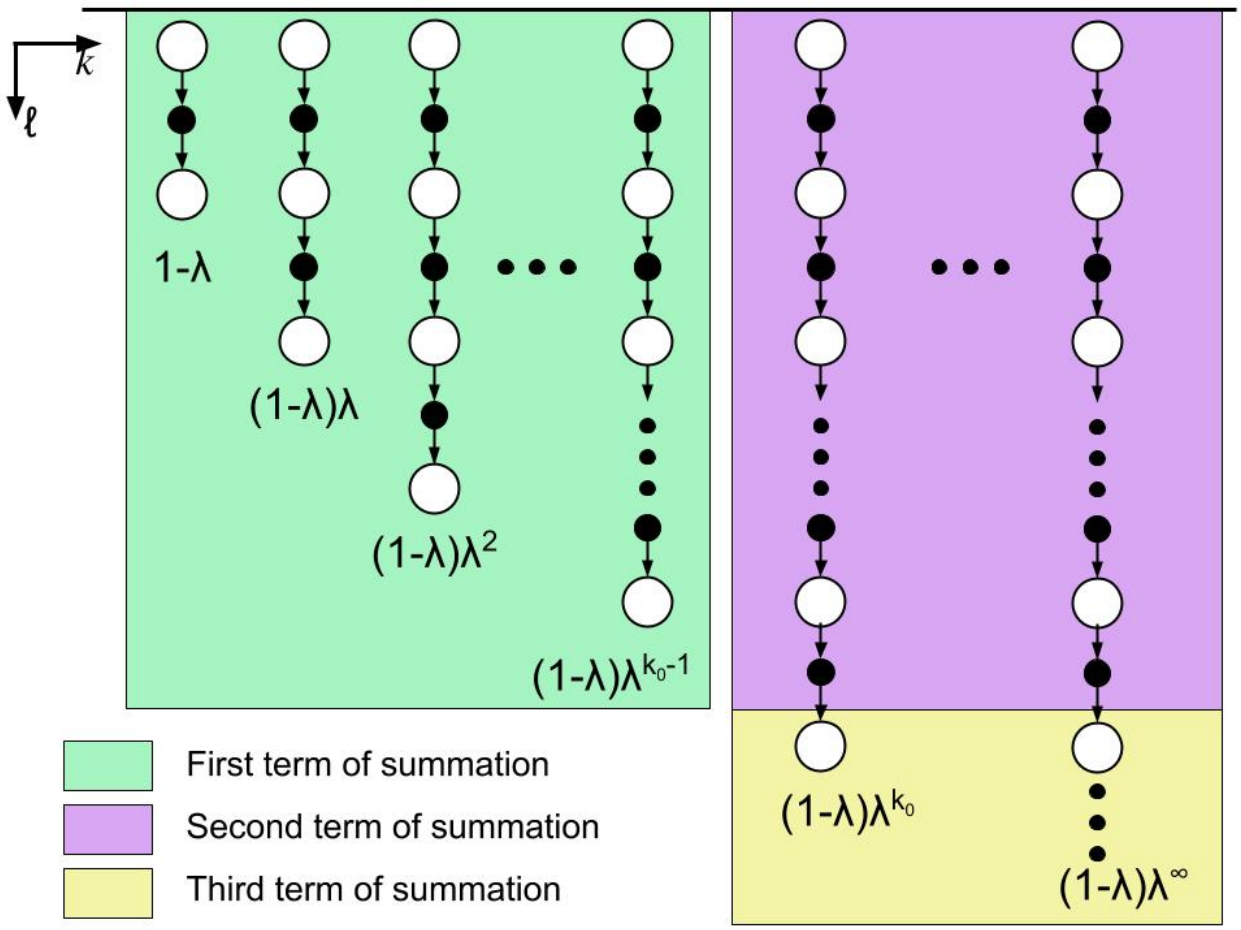

Figure 1: Illustration of the split of the sum as in equation 12

Each term is a geometric progression, first with common ratio of γ and then with common ratio of $\gamma\lambda$ (except for the 2nd term in inequality 12 wherein the common ratios are γ and λ respectively). Performing these summations and then simplifying the inequality by merging the first 2 terms (multiplier of ϵ_1) and keeping the third term separate (constant in the desired inequality), we get the following inequality:

$$\mathbf{A} \leq \frac{\epsilon_1(1-\lambda)}{1-\gamma}\left(\frac{1-\lambda^{k_0}}{1-\lambda} - \frac{\gamma(1-\gamma^{k_0}\lambda^{k_0})}{1-\gamma\lambda}\right) + \frac{\epsilon_1(1-\lambda)}{1-\gamma}\frac{\lambda^{k_0}}{1-\lambda}(1-\gamma^{k_0}) + \frac{1-\lambda}{1-\gamma}\left(\frac{\gamma^{k_0}\lambda^{k_0}}{1-\lambda} - \frac{\gamma^{k_0+1}\lambda^{k_0}}{1-\lambda\gamma}\right)$$

$$\implies \mathbf{A} \leq \frac{\epsilon_1(1-\lambda)}{1-\gamma}\frac{(1-\lambda^{k_0}-\gamma\lambda+\gamma\lambda^{k_0+1}-\gamma+\gamma\lambda-\gamma^{k_0+1}\lambda^{k_0+1}+\gamma^{k_0+1}\lambda^{k_0})}{(1-\lambda)(1-\gamma\lambda)} +$$

$$\frac{\epsilon_1(1-\lambda)}{1-\gamma}\frac{(\lambda^{k_0}-\gamma^{k_0}\lambda^{k_0}-\gamma\lambda^{k_0+1}+\gamma^{k_0+1}\lambda^{k_0+1})}{(1-\lambda)(1-\gamma\lambda)} + \frac{\gamma^{k_0}\lambda^{k_0}-\gamma^{k_0+1}\lambda^{k_0}}{(1-\gamma)(1-\gamma\lambda)}$$

$$\implies \mathbf{A} \leq \frac{\epsilon_1(1-\gamma)(1-(\gamma\lambda)^{k_0})}{(1-\gamma)(1-\gamma\lambda)} + \frac{(\gamma\lambda)^{k_0}}{1-\gamma\lambda}$$

$$\implies \mathbf{A} \leq \frac{\epsilon_1(1-(\gamma\lambda)^{k_0})}{(1-\gamma\lambda)} + \frac{(\gamma\lambda)^{k_0}}{1-\gamma\lambda} \tag{13}$$

Putting inequalities 10, 11 and 13 we get the inequality:

$$\Delta_t \leq \frac{\epsilon_1(1-(\gamma\lambda)^{k_0})}{(1-\gamma\lambda)} + \frac{(\gamma\lambda)^{k_0}}{1-\gamma\lambda} + \frac{(1-\lambda)\gamma\Delta_{t-1}}{1-\gamma\lambda} \tag{14}$$

where $\epsilon_1 = \sqrt{\frac{2}{n} ln(\frac{2k}{\delta})}$. Since we chose k_0 to be an arbitrary number, we can choose it to be such that the bound on Δ_t is minimized. Therefore, we get the desired inequality:

$$\Delta_t \leq \min_k \left(\frac{\epsilon_1(1-(\gamma\lambda)^k)}{(1-\gamma\lambda)} + \frac{(\gamma\lambda)^k}{1-\gamma\lambda} \right) + \frac{(1-\lambda)\gamma}{1-\gamma\lambda} \Delta_{t-1} \tag{15}$$

Answer 2. Part B

i). As mentioned certainty-equivalence estimates the MDP model rather than directly estimating the q-value functions in methods like Q-leaning and Sarsa. One disadvantage of certainty equivalence that is straightforward is that it cannot run online as it needs to collect all the data to estimate the model in contrast to the Q-learning methods which runs online. While waiting for the entire batch of data is a disadvantage, predicting the model itself makes certatinty-equivalence more sample efficient and in some cases gives better result with limited amount of data samples. To compare the time and space complexity, we assumed that each iteration of each algorithm uses H samples drawn from the environment. Also, we assumed enough samples were collected such that there was at least one sample where the agent was at state s and took action a, $\forall s \in \mathcal{S}$ and $a \in \mathcal{A}$. Thus, $H = \mathcal{O}(|S||A||S|)$. Q-learning would use online updates and therefore would update its estimate for every (s,a) pair, wherein the maximum over the next action will be done. Therefore, the time complexity would be $\mathcal{O}(|S||A||A|)$. In contrast, certainty-equivalence would use Bellman updates and therefore has time complexity of $\mathcal{O}(|S||A||S|)$. With respect to the space complexity certainty-equivalence needs to store the entire estimated MDP model which has $\mathcal{O}(|S||A||S|)$ space complexity in contrast to $\mathcal{O}(|S||A|)$ space complexity of the Q-learning technique. This is definitely a disadvantage.

Algorithms	Time Complexity	Space Complexity
Certainty Equivalence	$\mathcal{O}(\|S\|^2 \times \|A\|)$	$\mathcal{O}(\|S\|^2 \times \|A\|)$
q-value learning	$\mathcal{O}(\|S\| \times \|A\|^2)$	$\mathcal{O}(\|S\| \times \|A\|)$

Table 1: Per-iteration Time and Space Complexity of the algorithms.

ii). Let us take V_M^π as a $|S|$ dimensional vector where value function of state s of model M following policy π can be accessed as $V_M^\pi(s)$. Let us denote the estimated model be denoted as $\widehat{M}$ and $V_{\widehat{M}}^\pi$ be it's value function. Let us write the absolute difference of value functions in a particular state as:

$$|V_{\widehat{M}}^\pi(s) - V_M^\pi(s)| = |\widehat{R}(s,\pi) + \gamma \widehat{P}(s,\pi)^T . V_{\hat{M}}^\pi - R(s,\pi) - \gamma P(s,\pi)^T . V_M^\pi| \tag{16}$$

Note that $P(s,\pi)^T.V_M^\pi$ denote the dot product between two vectors $P(s,\pi)$ and V_M^π. Now adding and subtracting $\gamma P(s,\pi)^T.V_{\widehat{M}}^\pi$ from the right hand of on above equation:

$$= |\widehat{R}(s,\pi) - R(s,\pi) + \gamma \widehat{P}(s,\pi)^T . V_{\widehat{M}}^\pi - \gamma P(s,\pi)^T . V_{\widehat{M}}^\pi + \gamma P(s,\pi)^T . V_{\widehat{M}}^\pi - \gamma P(s,\pi)^T . V_M^\pi|$$

$$= |\widehat{R}(s,\pi) - R(s,\pi)| + \gamma |(\widehat{P}(s,\pi) - P(s,\pi))^T . V_{\widehat{M}}^\pi| + \gamma |P(s,\pi)^T . (V_{\widehat{M}}^\pi - V_M^\pi)|$$

Notice that $|\widehat{R}(s,\pi) - R(s,\pi)| \leq \max_{s,a} |\widehat{R}(s,a) - R(s,a)| \leq \varepsilon_R$ and $|P(s,\pi)^T.(V_{\widehat{M}}^\pi - V_M^\pi)| \leq ||V_{\widehat{M}}^\pi - V_M^\pi)||_\infty$ as $\Sigma_i p_i V_i \leq max_i V_i$ if $\Sigma_i p_i = 1$. Plugging these bounds in the above expression we get:

$$|V_{\widehat{M}}^\pi(s) - V_M^\pi(s)| \leq \varepsilon_R + \gamma ||V_{\widehat{M}}^\pi - V_M^\pi)||_\infty + \gamma |(\widehat{P}(s,\pi) - P(s,\pi))^T . V_{\widehat{M}}^\pi| \tag{17}$$

Now before we prove the rest we need to show that the value function $V^\pi(s)$ of state s lies in the range $[0, \frac{R_{max}}{1-\gamma}]$. We define $V^\pi(s) = \mathbb{E}[\sum_{t=1}^H \gamma^{t-1} r_t | \pi, s_1 = s]$ for an episode $s_1, \pi(s_1), r_1, s_2, \pi(s_2), r_2, s_3, ..., r_H, s_{H+1}$ where H is the horizon. Let us assume that our intermediate rewards r_t be non-negative then notice:

$$0 \leq \sum_{t=1}^{H} \gamma^{t-1} r_t \leq \sum_{t=1}^{\infty} \gamma^{t-1} R_{max} \leq \frac{R_{max}}{1-\gamma} \implies V^\pi(s) \in [0, \frac{R_{max}}{1-\gamma}] \tag{18}$$

Let us take I be a $|S|$ dimensional vector whose each element is 1 then another important thing to notice here is that:

$$|(\widehat{P}(s,\pi) - P(s,\pi))^T.(V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}I)| = |(\widehat{P}(s,\pi) - P(s,\pi))^T.V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}(\widehat{P}(s,\pi) - P(s,\pi))^T I|$$

It's easy to see that $(\widehat{P}(s,\pi) - P(s,\pi))^T I = 0$ as both $\widehat{P}(s,\pi)$ and $P(s,\pi)$ are valid probability distributions.

$$\implies |(\widehat{P}(s,\pi) - P(s,\pi))^T.(V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}I)| = |(\widehat{P}(s,\pi) - P(s,\pi))^T.V^{\pi}_{\widehat{M}}| \tag{19}$$

Now all that is left is to center the values in value function about 0. We substitute the LHS of equation 19 in inequality 17 to get:

$$|V^{\pi}_{\widehat{M}}(s) - V^{\pi}_{M}(s)| \leq \varepsilon_R + \gamma||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty + \gamma|(\widehat{P}(s,\pi) - P(s,\pi))^T.(V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}I)|$$

$$\leq \varepsilon_R + \gamma||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty + \gamma||(\widehat{P}(s,\pi) - P(s,\pi))||_1||(V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}I)||_\infty$$

We know as $V^{\pi}(s) \in [0, \frac{R_{max}}{1-\gamma}] \implies ||(V^{\pi}_{\widehat{M}} - \frac{R_{max}}{2(1-\gamma)}I)||_\infty \leq \frac{R_{max}}{2(1-\gamma)}$. Using that and $||(\widehat{P}(s,\pi) - P(s,\pi))||_1 \leq \max_{s,a} ||(\widehat{P}(s,a) - P(s,a))||_1 \leq \varepsilon_P$:

$$|V^{\pi}_{\widehat{M}}(s) - V^{\pi}_{M}(s)| \leq \varepsilon_R + \gamma||V^{\pi}_{\widehat{M}} - V^{\pi}_{M})||_\infty + \frac{\gamma\varepsilon_P R_{max}}{2(1-\gamma)} \tag{20}$$

As equation 20 is true for all s so we can write:

$$||V^{\pi}_{\widehat{M}} - V^{\pi}_{M})||_\infty \leq \varepsilon_R + \gamma||V^{\pi}_{\widehat{M}} - V^{\pi}_{M})||_\infty + \frac{\gamma\varepsilon_P R_{max}}{2(1-\gamma)}$$

$$\implies ||V^{\pi}_{\widehat{M}} - V^{\pi}_{M})||_\infty \leq \frac{\varepsilon_R}{1-\gamma} + \frac{\gamma\varepsilon_P R_{max}}{2(1-\gamma)^2} \tag{21}$$

iii). In this part we want to bound the error in the optimal policy obtained from the approximate MDP. Let $\pi^*_{\widehat{M}}$ denote the optimal policy derived from approximate MDP: $\widehat{M}$ and V^*_M & π^*_M be the optimal value function & optimal policy of the true MDP: M, then we can write:

$$V^*_M(s) - V^{\pi^*_{\widehat{M}}}_M(s) = V^{\pi^*_M}_M(s) - V^{\pi^*_M}_{\widehat{M}}(s) + V^{\pi^*_M}_{\widehat{M}}(s) - V^{\pi^*_{\widehat{M}}}_M(s)$$

Now we know that $\pi^*_{\widehat{M}}$ would give maximum value for $V^{\pi}_{\widehat{M}}$ which gives:

$$V^*_M(s) - V^{\pi^*_{\widehat{M}}}_M(s) \leq V^{\pi^*_M}_M(s) - V^{\pi^*_M}_{\widehat{M}}(s) + V^{\pi^*_{\widehat{M}}}_{\widehat{M}}(s) - V^{\pi^*_{\widehat{M}}}_M(s) \tag{22}$$

$$\leq ||V^{\pi^*_M}_M - V^{\pi^*_M}_{\widehat{M}}||_\infty + ||V^{\pi^*_{\widehat{M}}}_{\widehat{M}} - V^{\pi^*_{\widehat{M}}}_M||_\infty$$

Now $||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty$ attains it's supremum for some π then $||V^{\pi^*_M}_M - V^{\pi^*_M}_{\widehat{M}}||_\infty \leq ||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty$ and$||V^{\pi^*_{\widehat{M}}}_{\widehat{M}} - V^{\pi^*_{\widehat{M}}}_M||_\infty \leq ||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty$ which leads us to:

$$V^*_M(s) - V^{\pi^*_{\widehat{M}}}_M(s) \leq \sup_{\pi:S\to A} ||V^{\pi}_{\widehat{M}} - V^{\pi}_{M}||_\infty \tag{23}$$

iv). Now we will try to first prove the error bounds on $\widehat{R}(s,a)$ and $\widehat{P}(s'|s,a)$. For $\widehat{R}(s,a)$ we first considered it normalised that is scaled down by R_{max}. Using Hoeffding's inequality we have:

$$Pr(|\widehat{R}(s,a) - R(s,a)| \geq \epsilon) \leq 2e^{-2n\epsilon^2} \qquad \forall s \in S, a \in A \tag{24}$$

Taking union bound:

$$Pr(\bigcup_{s,a}\{|\widehat{R}(s,a)-R(s,a)|\geq\epsilon\})\leq\sum_{s,a}Pr(|\widehat{R}(s,a)-R(s,a)|\geq\epsilon)\leq 2|S||A|e^{-2n\epsilon^2}$$

$$Pr(\max_{s,a}|\widehat{R}(s,a)-R(s,a)|\leq\epsilon)=Pr(\bigcap_{s,a}\{|\widehat{R}(s,a)-R(s,a)|\leq\epsilon\})$$

$$=1-(\bigcup_{s,a}\{|\widehat{R}(s,a)-R(s,a)|\geq\epsilon\})\geq 1-2|S||A|e^{-2n\epsilon^2}$$

Taking $\delta/2=2|S||A|e^{-2n\epsilon^2}$ we get $\epsilon=\sqrt{\frac{1}{2n}\ln\frac{4|S\times A|}{\delta}}$. Hence with $1-\delta/2$ it holds: $\max_{s,a}|\widehat{R}(s,a)-R(s,a)|\leq R_{max}\sqrt{\frac{1}{2n}\ln\frac{4|S\times A|}{\delta}}$. Notice we scaled up the $\widehat{R}(s,a)$ by R_{max} in the final result. Similarly we can show with $1-\delta/2$ it holds: $\max_{s,a}|\widehat{P}(s'|s,a)-P(s'|s,a)|\leq\sqrt{\frac{1}{2n}\ln\frac{4|S\times A\times S|}{\delta}}$. Again using union bound here we say with probability atleast $1-\delta$ both the bounds holds:

$$\max_{s,a}|\widehat{R}(s,a)-R(s,a)|\leq\sqrt{\frac{1}{2n}\ln\frac{4|S\times A|}{\delta}}\tag{25}$$

$$\max_{s,a}|\widehat{P}(s'|s,a)-P(s'|s,a)|\leq\sqrt{\frac{1}{2n}\ln\frac{4|S\times A\times S|}{\delta}}\tag{26}$$

In the last part we combine the above two derived equations 21 & 23 and combine it with the concentration inequalities.

$$V_M^*(s)-V_M^{\pi^*_{\widehat{M}}}(s)\leq\frac{2\varepsilon_R}{1-\gamma}+\frac{\gamma\varepsilon_P R_{max}}{(1-\gamma)^2}$$

Notice the following:

$$\max_{s,a}||(\widehat{P}(s,a)-P(s,a))||_1\leq\max_{s,a}|S|||\widehat{P}(s,a)-P(s,a))||_\infty\leq|S|\sqrt{\frac{1}{2n}\ln\frac{4|S\times A\times S|}{\delta}}\tag{27}$$

Substituting the values of ε_R and ε_P we get:

$$V_M^*(s)-V_M^{\pi^*_{\widehat{M}}}(s)\leq\frac{2}{1-\gamma}\sqrt{\frac{1}{2n}\ln\frac{4|S\times A|}{\delta}}+\frac{\gamma R_{max}|S|}{(1-\gamma)^2}\sqrt{\frac{1}{2n}\ln\frac{4|S\times A\times S|}{\delta}}$$

For large state spaces the logarithmic dependencies are insignificant compared the linear dependency on the state space.So we ignore the first term above and if we neglect the dependence on R_{max} and $1/\delta$ we end up with:

$$V_M^*(s)-V_M^{\pi^*_{\widehat{M}}}(s)=\mathcal{O}(\frac{|S|}{\sqrt{n}(1-\gamma)^2})\tag{28}$$

2 Coding Part

2.1 Question 1. Continuous Random Walk (Prediction Task)

We defined the Random Walker class according to the specifications given, namely the position $\in[0,1]$, start state at 0.5 and terminal states at less than 0 or greater than 1. We represent the state space using the position of the agent. We evaluate the value function at each state for a random policy, wherein the action is chosen randomly in the interval [-0.2,0.2]. We set the discount factor for the environment to be 1. The implementation can be found here

Furthermore, we implement a sparse coarse coding strategy in order to convert the continuous space into a discrete space. Specifically, we follow a tiling strategy with 10 tilings and 11 bins in each tiling. The entire state space is divided into 10 equal sized bins and subsequently shifted by the offset for that tiling. An additional interval is added to represent if the agent's position was less than the tiling's offset. Consequently,

we have 110 total binary features which are 1 when the agent's position is within the corresponding bin. Since we have 10 tilings, for any state there are exactly 10 features that are active and rest are 0. The implementation for the sparse coarse coding can be found here

Finally, we implement the TD(λ) algorithm with accumulating traces to learn the value function. The implementation closely follows the pseudocode provided in Section 12.2 of the Sutton and Barto book) and can be found here. We use a linear function approximator, wherein we have 110 weights each corresponding to the weight of a binary feature in the sparse coarse code. These weights are initialized to 0, such that initially all states are predicted to have 0 value. We evaluate the accuracy of function approximation using a mean squared error (MSE) metric over 21 equally spaced states in the interval [0,1]. We select different values for λ and learning rate α and subsequently plot these values as shown in Figure 2. We trained the agent for 75 iterations (as opposed to 25, as specified in the question to ensure that the agent reached a steady state for most of the experiments) and then plotted the MSE, averaged over different runs (each run uses a different seed). The results averaged over 10 different seeds is shown here and results averaged over 50 different seeds is shown here (also in Figure 2). The shaded regions represent the standard error in measurement, which is calculated by dividing the standard deviation of MSE across different seeds by the square root of the number of seeds.

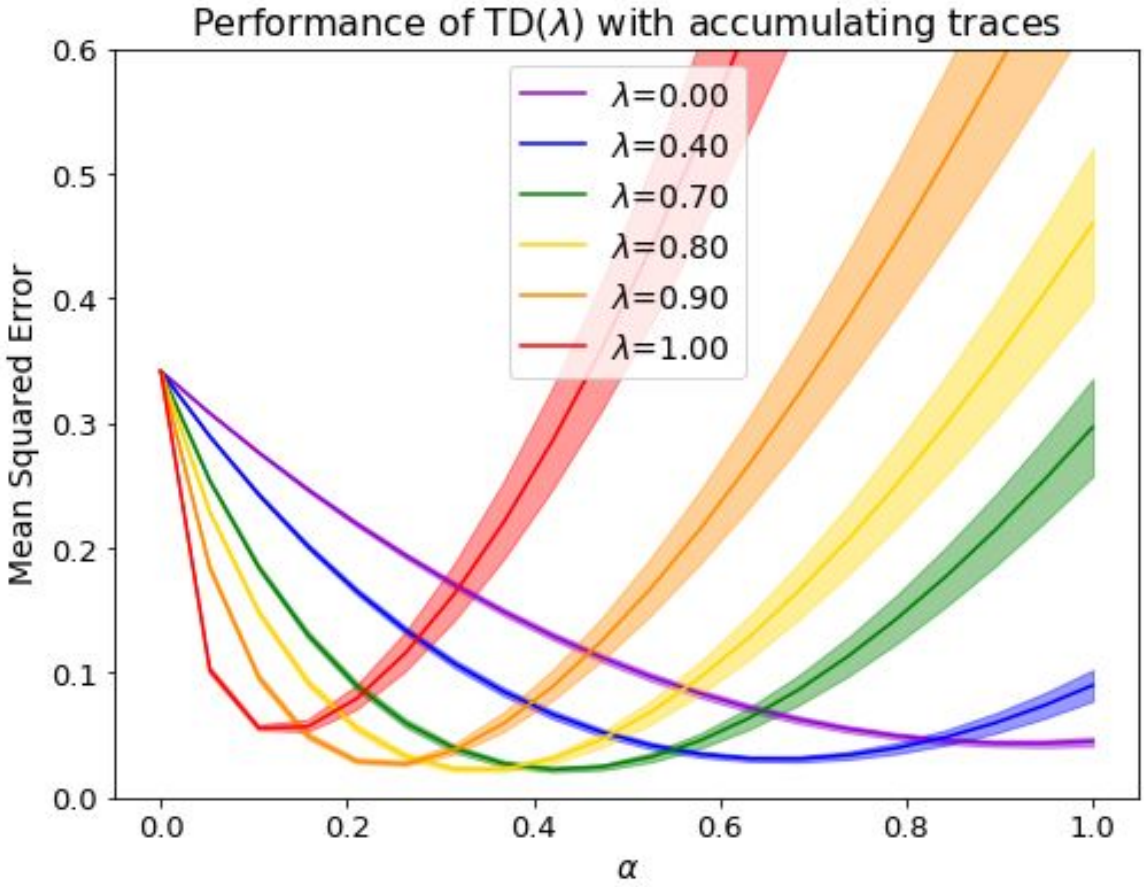

Figure 2: Performance (MSE) of the TD(λ) algorithm for the continuous random walk environment for different λ and α. Results are averaged over 50 seeds and shaded region depicts standard error.

For each experiment, the agent is trained for 75 episodes and the MSE is evaluated after 75 episodes of training. The different λ values tried are [0,0.4,0.7,0.8,0.9,1.0] and 25 evenly spaced α values were chosen from the interval [0,1]. It is evident from the plots that for each λ, there is an optimal value of the learning rate α. Higher α values cause the function approximator to diverge and leads to overflow issues. Low λ values indicate lower credit assignment to past states and therefore takes more time/experience for the network to

learn. Hence, higher learning rates could be used as compared to higher λ values. Using very high λ value attributes high importance to past states and therefore can lead to contrasting information wherein states from the distant past are considered to be important for distinct rewards/outcomes. Therefore, using a high learning rate causes the algorithm to diverge.

Per-iteration/episode Time Complexity	$\mathcal{O}(n)$
Per-iteration/episode Space Complexity	$\mathcal{O}(\|S_f\|)$

Table 2: Time and Space Complexity of TD(λ) implementation. n denotes the average length of trajectory/episode and S_f denotes the sparse coarse coded state space.

2.2 Question 2. Part B.

In this part we use the mountain car environment from Open AI gym and set the discount factor γ to 1. We have used the continuous version of the environment and customised the reward according to the problem in our wrapper class which can be found here. Our state space contains the position and the velocity which is continuous. We have three discrete actions: [-1,0,1]. We implemented asymmetric tiling to convert the continuous space into a discrete space in 2 dimension. We have used a displacement vector of (1,3) for the tiling offset. We have used 8 tiles and 8 bins in each dimension resulting in $512 \times 3 = 1536$ features to represent a (state,action) pair. Notice the 3 is because of the number of actions. Out of the 512 features for each action, exactly 24 features are active and rest are 0 for any particular state. Also, it is noteworthy that we placed the tiles symmetrically over the state space limits in order to have symmetric spanning of both ends of the position and velocity space. The implementation can be found here.

Finally, we implement the Sarsa(λ) algorithm with replacing traces to learn the action-value function. The implementation closely follows the pseudocode for binary features provided in Section 12.7 of the Sutton and Barto book and can be found here. We use a linear function approximator, wherein we have 1536 weights each corresponding to the weight of a binary feature in the tiling code for each (state,action) pair. These weights are initialized to 0, such that initially all state-action pairs are predicted to have 0 value. We use an ϵ-greedy policy to estimate the Q-function with $\epsilon = 0.1$. This enforces the agent to explore during the training process to look for new state-action pairs. We evaluate the efficacy of our control prediction using the steps to goal metric. Specifically, we ran an episode to maximum number of steps (here, 1000) and evaluated the number of steps taken by the car to reach the goal when we use a greedy policy based on the estimated Q-function. If the car fails to reach the goal, we assume the steps to goal to be 1000. We select different values for λ and learning rate α and subsequently plot these values as shown in Figure 3. For the full range of the α refer to this. We trained the agent for 200 episodes (as opposed to 1000, because of computation resource constraint) and then plotted the steps to goal, averaged over five different runs.

Per-iteration/episode Time Complexity	$\mathcal{O}(n)$
Per-iteration/episode Space Complexity	$\mathcal{O}(\|S_f\| \times \|A\|)$

Table 3: Time and Space Complexity of SARSA(λ) implementation. n denotes the average length of trajectory/episode, S_f denotes the sparse coarse coded state space and A denotes the action space.

The different λ values tried are [0,0.4,0.7,0.84,0.96] and 15 evenly spaced α values were chosen from the interval [0.05,0.75]. It can be concluded from the plots that for each λ, there is an optimal value of the learning rate α. Also higher value of λ gives fewer steps in general. Higher α values cause the function approximator to diverge and leads to overflow of action-values. So we would preferably like to keep the value of λ higher and α lower to achieve reasonable performance. The behavior of eligibility trace here is similar to what mentioned in the earlier section where we implemented TD(λ). Also note that the graphs could be made less noisy using higher number of episodes for learning leading to lower variance. Another idea would be to slowly decay the learning rate using an annealing schedule such that initial training samples are given more importance in updating the function approximator as compared to the later training samples. This might help stabilise the training procedure and allow the algorithm to ignore doing noisy updates. However, we did not implement this in our experiments owing to computational and time constraints.

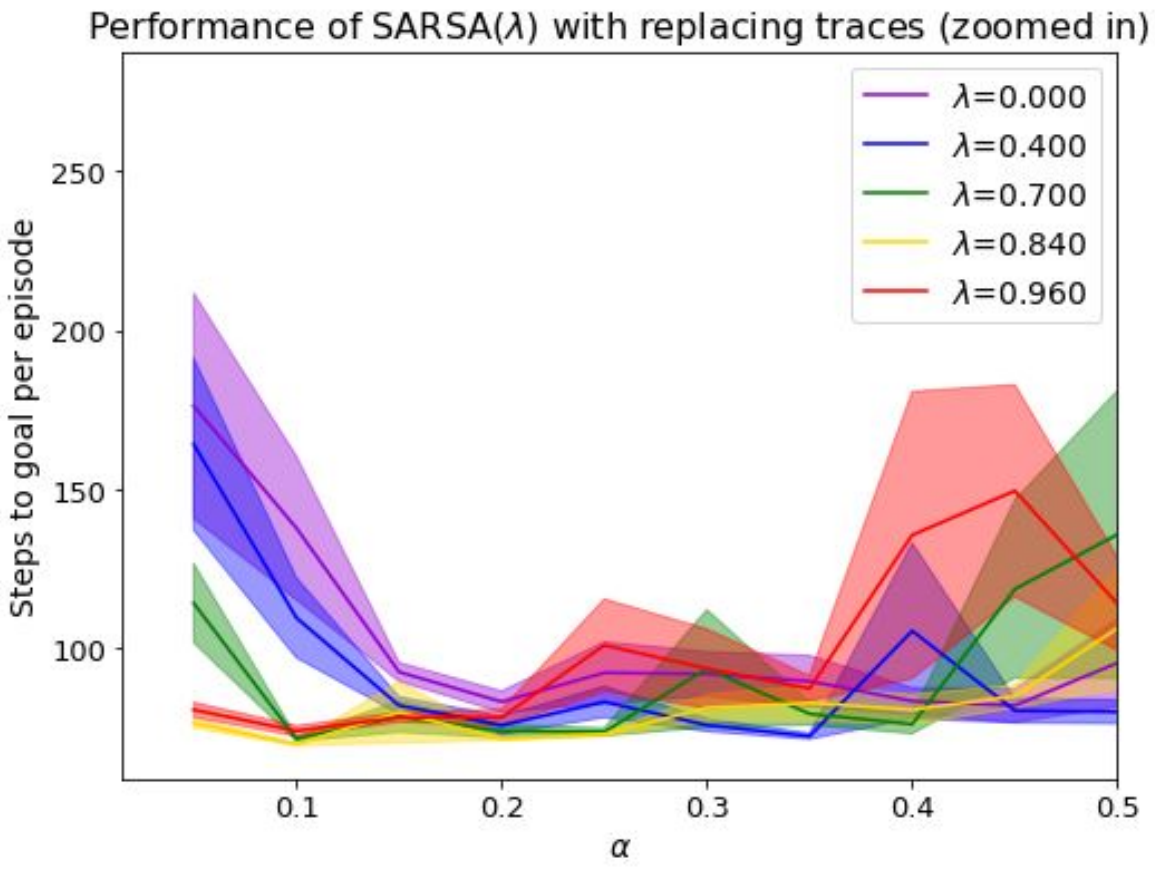

Figure 3: Performance (Steps to goal) of the Sarsa(λ) algorithm for the Mountain Car environment for different λ and α. Results are averaged over 50 seeds and shaded region depicts standard error.

Contributions

AKM and AG worked on the theory questions together. AG implemented the TD(λ) random walk problem. AKM and AG worked together for the Sarsa(λ) Mountain Car problem. Both AG and AKM contributed equally to writing the report.

Reproducibility Checklist

- ☒ Description of algorithms and environments
- ☒ Time and space complexity for TD(λ)
- ☒ Time and space complexity for SARSA(λ)
- ☒ Link to code – added link to Colab notebook where applicable
- ☐ Description of data collection process - **not applicable**
- ☒ Link to download dataset or environment
- ☐ Explanation of excluded data – **not applicable**
- ☒ Explanation of train/test splitting

☒ Range of hyperparameters and method to select best hyperparameter

☒ Number of evaluation runs

☒ Description of experiments

☒ Definition of metrics used

☒ Error bars in plots – plots indicate standard error bars

☒ Result description with mean and standard deviation – shown in plots

☒ Description of computing infrastructure used – Colab notebook provided to enhance reproducibility of results