Exercise Set 2 - Reinforcement Learning

Tabular Methods

Giulio Starace - 13010840

September 20, 2022

Homework: Coding Assignment - Monte Carlo

1. To derive the incremental update rule of the value function for a given state $V(s)$ using ordinary importance sampling, we begin with the definition of the estimate $V_n(s)$ given n sampled returns $G_1, \dots, G_n$ under this regime:

$$V_n(s) = \frac{1}{n} \sum_{k=1}^{n} W_k G_k, \tag{1}$$

where W_k is the importance sampling ratio of target policy π over behavior policy b.

$$W_k = \rho_{t_k:T(t_k)-1} = \prod_{t=t_k}^{T(t_k)-1} \frac{\pi(A_t|S_t)}{b(A_t|S_t)}. \tag{2}$$

We expand equation (1) to obtain:

$$\begin{aligned}
V_n(s) &= \frac{1}{n} \left[W_{n-1} G_{n-1} + \sum_{i=1}^{n-1} W_k G_k \right] \\
&= \frac{1}{n} \left[W_{n-1} G_{n-1} + (n-1) \underbrace{\left(\frac{1}{n-1} \right) \sum_{i=1}^{n-1} W_k G_k}_{} \right] \\
&= \frac{1}{n} \left[W_{n-1} G_{n-1} + (n-1) \underbrace{V_{n-1}(s)}_{} \right] \\
&= \frac{1}{n} \left[W_{n-1} G_{n-1} + n V_{n-1}(s) - V_{n-1}(s) \right] \\
V_n(s) &= V_{n-1}(s) + \frac{1}{n} \left[W_{n-1} G_{n-1} - V_{n-1}(s) \right].
\end{aligned} \tag{3}$$

We see that equation (3) is of the form

$$V_n = V_{n-1} + \alpha * (\beta - V_{n-1}), \tag{4}$$

with $\alpha = \frac{1}{n}$ and $\beta = W_{n-1} G_{n-1}$.

■

2. Coding answers have been submitted on codegra under the group "stalwart cocky sawly". Please refer to Figures 1 and 2 for the requested figures.

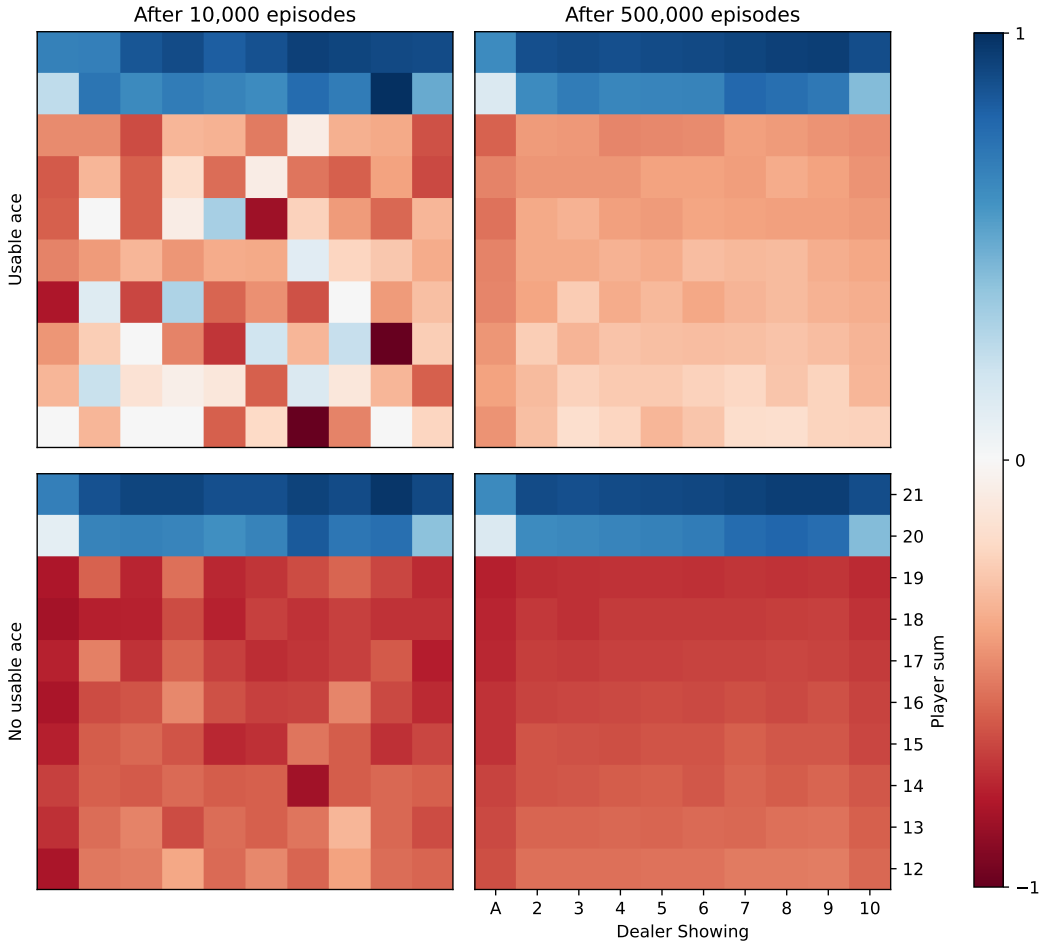

Figure 1: Blackjack value function across state configurations after 10k and 500k episodes using on-policy MC prediction. Reproduction of figure 5.1 from Sutton and Barto, using heatmaps.

3. The main difference between Dynamic Programming (DP) and Monte Carlo (MC) is that the former assumes a perfect model (complete knowledge) of the environment, while the latter does not, relying on experience instead. Furthermore, unlike DP methods, MC methods do not *bootstrap*, i.e. they do not update their estimates using other estimates. MC methods are better suited in situations where the environment is not known and needs to be learned about. DP on the other hand is better suited in situations where the environment is known and we can afford to compute the value function of each state.

Homework: SARSA and Q-learning

1. (a) Please refer to Table 1a for the final Q-values for all states and actions after converging un-discounted SARSA with an ϵ-greedy policy with $\epsilon = 0.1$.

   (b) The final policy will prefer to choose action a_1 more often in state A when

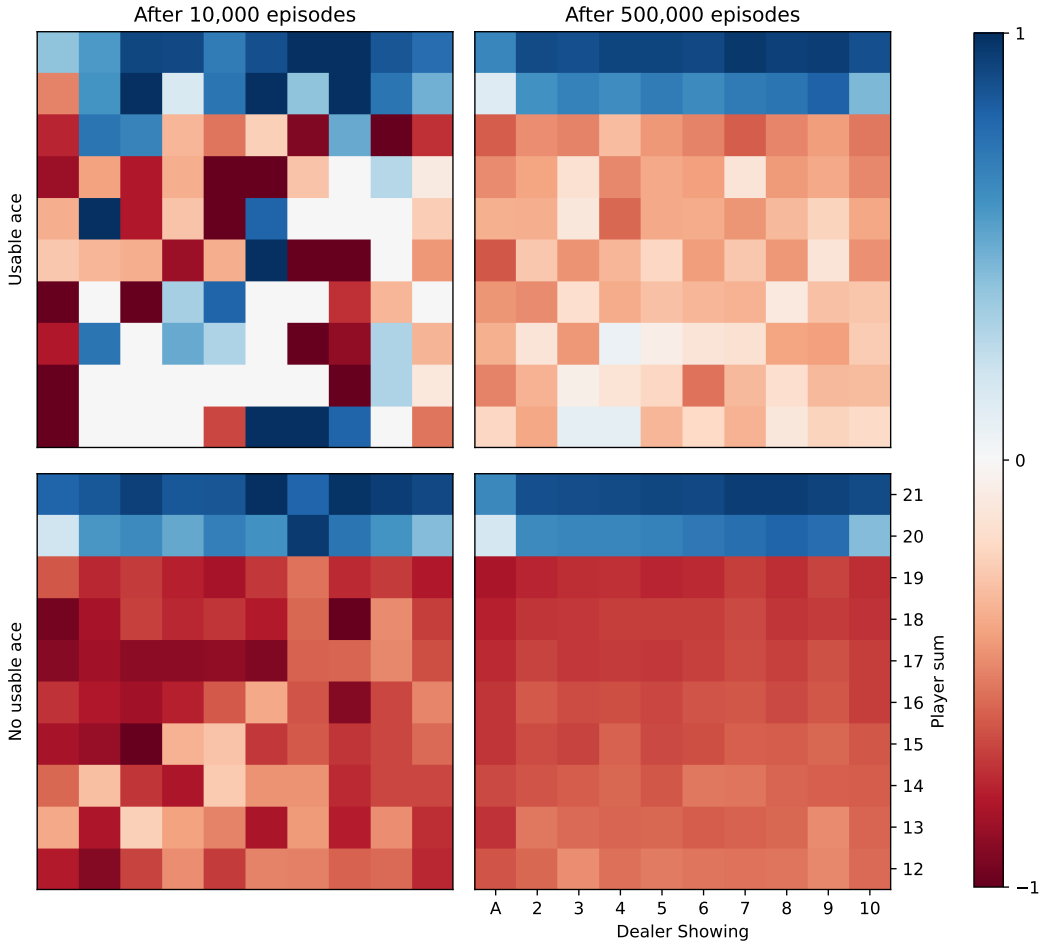

Figure 2: Blackjack value function across state configurations after 10k and 500k episodes using off-policy MC prediction with ordinary importance sampling.

$Q(A, a_1) > Q(A, a_0)$, that is when

$$\begin{aligned} Q(A, a_1) &> Q(A, a_0) \\ 3.9 + 0.05n &> 2 \\ n &> -38 \end{aligned}$$

i.e when $n \in (-38, 2)$. Similarly the final policy will prefer to choose action a_0 more often in state A when $Q(A, a_1) < Q(A, a_0)$, that is when

$$\begin{aligned} Q(A, a_1) &< Q(A, a_0) \\ 3.9 + 0.05n &< 2 \\ n &< -38 \end{aligned}$$

i.e when $n \in (-\infty, -38)$. When $n = 38$, the final policy will choose between actions a_0 and a_1 arbitrarily.

2. (a) Please refer to Table 1b for the final Q-values for all states and actions after converging un-discounted Q-learning.

(b) The final policy will prefer to choose action a_1 more often in state A when $Q(A, a_1) > Q(A, a_0)$, that is when

$$\begin{aligned} Q(A, a_1) &> Q(A, a_0) \\ 2 + \max(2, n) &> 2 \\ \max(2, n) &> 0 \end{aligned}$$

This condition remains true regardless of the value of n, so the final policy will always prefer to choose action a_1 when in state A, for any $n \in (-\infty, \infty)$. As such, the final policy will never choose a_0 in state A, i.e. for none of the values of n.

Table 1: Converged Q-values for SARSA and Q-learning, assuming initialization at 0.

	A	B	C	T
a_0	2	1	2	0
a_1	$3.9 + 0.05n$	0	n	0

(a) Q-values for all states and actions at SARSA convergence.

	A	B	C	T
a_0	2	1	2	0
a_1	$2 + \max(2, n)$	0	n	0

(b) Q-values for all states and actions at Q-learning convergence.

3. With $n = 1$ and the addition of an extra action a_2 in A causing a transition to B with reward 1, the final performance (average return after convergence) of Q-learning would remain the same. This is because Q-learning converges to the optimal policy, choosing the action with the maximum value, which in this case, remains to be a_1, as a_2 provides the same effect as a_0, which we have established is never chosen by a converged Q-learning policy. The final performance of SARSA on the other hand would decrease, as there are now more actions to choose from in state A, causing the policy to choose greedily less often.