Reinforcement Learning: Homework 2

Jorrim Prins
Student nr.: 11038934
jorrimprins.prins@student.uva.nl

Yke Rusticus
Student nr.: 11306386
yke.rusticus@student.uva.nl

3.1 Coding Assignment - Monte Carlo

1.

$$
\begin{aligned}
V_n &= \frac{\sum_{k=1}^{n} W_k G_k}{n} \\
&= \frac{1}{n}\left(W_n G_n + \sum_{k=1}^{n-1} W_k G_k\right) \\
&= \frac{1}{n}\left(W_n G_n + \frac{n-1}{n-1}\sum_{k=1}^{n-1} W_k G_k\right) \\
&= \frac{1}{n}\left(W_n G_n + (n-1)\, V_{n-1}\right) \\
&= \frac{1}{n}\left(W_n G_n + n V_{n-1} - V_{n-1}\right) \\
&= V_{n-1} + \frac{1}{n}\left(W_n G_n - V_{n-1}\right)
\end{aligned}
\tag{1}
$$

3. Dynamic Programming and Monte Carlo can be used to find optimal policies for Markov decision processes. Dynamic Programming relies on the dynamics of the system, using known probabilities for rewards of certain action/state combinations allows for an iterative scheme of finding optimal policies. Monte Carlo only uses available generated data from the Markov process to learn a value function and use this value function to determine optimal policies.

Obviously, Dynamic Programming methods are useful when system dynamics are known and Monte Carlo should be used otherwise.

4.3 Coding assignment - Temporal Difference Learning

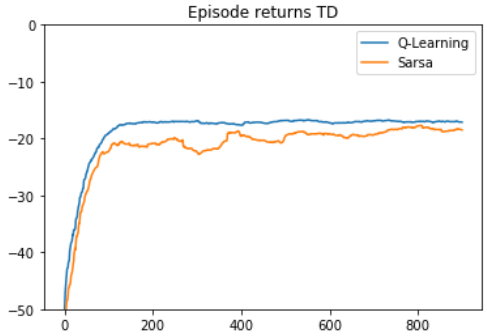

Figure 1: Lab 2 Q-learning and SARSA results.

2. We found that in our case, Q-learning consistently results in higher average return than SARSA. This is opposite to the results presented in Example 6.6 in the book. This is because in Example 6.6 Q-learning goes for the most risky, but fastest road alongside the cliff. Due to the stochasticity in the action sampling, it could occur that a certain action would result in falling off the cliff, and receiving a large negative award. This causes the safe route to be generally better than the risky route. In our case we do not have such high penalties for taking a wrong step, so the fastest route is generally better than the safe route, which is likely why Q-learning performed better than SARSA.

4.4 Maximization Bias

First of all, the assignment does not specify a learning rate α so this is assumed to be equal to one. It would not have an influence on any of the conclusions in these questions as it would affect Sarsa and Q-learning equally (only in the values for $Q(A, R)$ as it is the only state-action pair with multiple future steps).

1. Obviously, all the terminating state-action pairs have equal values, as the two presented methods only differ in their valuation of future steps. $Q(A, R)$ is the only non-terminating state in this model and is therefore the only state with a different value for the Sarsa method than for the Q-learning method.

	Sarsa	**Q-learning**
$Q(A, L)$	0.7	0.7
$Q(B, u_1)$	0	0
$Q(B, u_2)$	1	1
$Q(B, u_3)$	0.5	0.5
$Q(B, u_4)$	0.5	0.5
$Q(A, R)$	0.5	1

Table 1: 10 observations valuation

2. The expected value of all actions available from B (u_1, u_2, u_3, u_4 to be exact) is equal to 0.5 as the rewards are uniformly distributed between 0 and 1. As the sample size has become larger, the value of $Q(B, u_1)$ and $Q(B, u_2)$ are now 0.5 and $Q(A, R)$ receives equal valuation from Sarsa and Q-learning methods at 0.5.

	Sarsa	**Q-learning**
$Q(A, L)$	0.7	0.7
$Q(B, u_1)$	0.5	0.5
$Q(B, u_2)$	0.5	0.5
$Q(B, u_3)$	0.5	0.5
$Q(B, u_4)$	0.5	0.5
$Q(A, R)$	0.5	0.5

Table 2: Large sample valuation

3. The only method that suffers from maximization bias is Q-learning, as can be seen in the final rows of Table 1 and 2. The bias only occurs in non-terminating state-action pairs, as the maximization in Q-learning happens for timestep $t + 1$. Sarsa does not make use of any maximization within its updating and will therefore not suffer from this bias. The bias fades out when sample size increases, as the collected samples will converge towards their expected values, and Sarsa and Q-learning present equal results.

Reviewing the problem from starting point A, the maximization bias from using Q-learning would lead to overselection of action R as $Q(A, R) > Q(A, L)$ because of the bias.

4. Q-learning learns the same target function to select actions and evaluate state-action pairs and because of the maximization, overoptimistic values are more likely to be used in these updates. Double Q-learning can alleviate this problem by learning two separate Q functions, Q_1 and Q_2. Both functions are updated separately, Q_1 by selecting a state-action pair with Q_1 and evaluating with Q_2

(and vice versa for Q_2). Note that only half of observations can be used for Q_1 and the other half for Q_2.

In the current example, we will use the first outcome of u_i to update Q_1 and the latter for Q_2. The maximization part in updating Q_1 would select u_2 and u_3 with 50 percent chance and evaluation with the latter datapoints (used for Q_2) presents a value of 0.5 for $Q_1(A, R)$. The same thing happens in the update of Q_2, where the maximization part would select u_3 and u_4. Evaluation by "Q_1's datapoints" again presents a value of 0.5 for $Q_2(A, R)$. Even in this tiny example, the maximization bias problem is solved (for both Q-functions) by the use of Double Q-learning.