Reinforcement Learning - Exercise Sheet 3

Tom Lieberum - ID 13253042 Erik Jenner - ID 13237896

16. September 2020

3.1 Coding Assignment - Monte Carlo

1.

First we expand V_n:

$$V_n = \frac{\sum_{k=1}^{n} W_k G_k}{n} = \frac{W_n G_n + \sum_{k=1}^{n-1} W_k G_k}{n}$$

Multiplying with n yields

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

2.

See the autograd file.

3.

The key difference between Dynamic Programming and Monte Carlo methods is that the former does a wide, one-step backup, whereas the latter compute a maximally deep but very narrow backup. So, if it is easy to simulate trajectories, but expensive to compute the wide backups for all states and actions or the transition probabilities are not known, then we should use Monte Carlo methods. However, if the transition probabilities are known and the system is small enough to make backups feasible, Dynamic Programming is the most direct method to obtain optimal solutions since sampling trajectories first just adds noise that unnecessarily increases errors. Another important advantage of Dynamic Programming is that it does not depend on the MDP terminating. This means it can be used in continuing environments or as an online updating scheme, whereas Monte Carlo methods need each episode to terminate before they can perform an update.

4.3 Coding assignment - Temporal Difference Learning

1.

See the autograd file.

2.

In contrast to example 6.6 from RL:AI, we observe that Q-learning performs better than SARSA in learning the given task. Presumably, this is because in this particular environment there is no cliff which would yield a large negative reward. Because of that, caution is unnecessary; or more formally:

$$\epsilon \cdot \text{random policy} + (1 - \epsilon) \cdot \text{optimal policy}$$

is an optimal ϵ-soft policy. Therefore, Q-learning is not punished for learning the greedy Q-values while behaving according to an ϵ-greedy policy.

4.4 Maximization Bias

1.

If we assume that Sarsa acts ϵ-greedily w.r.t. its estimate of the Q-function we obtain

Sarsa:

$$\begin{aligned} Q(B, a_1) &= 0 \\ Q(B, a_2) &= 1 \\ Q(B, a_3) &= 0.5 \\ Q(B, a_4) &= 0.5 \\ Q(A, L) &= 0.7 \\ Q(A, R) &= (1 - \epsilon) + 0.5\epsilon = 1 - 0.5\epsilon \end{aligned}$$

Q-learning

$$\begin{aligned} Q(B, a_1) &= 0 \\ Q(B, a_2) &= 1 \\ Q(B, a_3) &= 0.5 \\ Q(B, a_4) &= 0.5 \\ Q(A, L) &= 0.7 \\ Q(A, R) &= 1 \end{aligned}$$

2.

$$\begin{aligned} Q(B, a_i) &= 0.5, \quad \text{where } i \in \{1, 2, 3, 4\} \\ Q(A, L) &= 0.7 \\ Q(A, R) &= \sum_{i=1}^{4} \pi(a_i|B) Q(B, a_i) = 0.5 \end{aligned}$$

where the policy π in the last equation is either greedy or ϵ-greedy but doesn't matter anyways since $Q(B, a_i) = 0.5 \; \forall i$.

3.

Both methods suffer from maximization bias which can be observed in the estimate for $Q(A, R)$, which those methods calculate by taking an (ϵ-)greedy approach in the successor state B. Since both methods compute the maximum over actions[1], they overestimate the value of the subsequent state B.

4.

The root of the problem of maximization bias is that the learned Q-function is used both to sample/select the next action as well as for judging the selected action. Since $\mathbb{E}[\max_i X_i] \geq \max_i \mathbb{E}[X_i]$ for random variables X_i, this leads to a bias towards higher Q-values (let $X_i = Q(s, a_i)$ and take the expected value over the data that produces this Q-estimate).

In Double Q-learning, we learn two Q-functions on different datasets, i.e. different sets of episodes. For this example, that would mean that they observe different values for the transition from $B \to T$, resulting in different estimates of B's value. This eliminates the problem of '$\mathbb{E}\max \geq \max\mathbb{E}$'.

In the given example, $\text{argmax}_i\, Q_1(B, a_i)$ is equally likely to be any of $\{1, \ldots, 4\}$ if Q_1 is based on randomly sampled experience from the MDP. Thus the expectation over possible experience of $Q_2(B, \text{argmax}_i\, Q_1(B, a_i))$ is 0.5 which is the unbiased estimate. Note that this is not always the case – Double Q-Learning can *underestimate* the maximum Q-values in some cases[2].

[1] Sarsa does this with probability $1 - \epsilon$.

[2] See also Hado van Hasselt: 'Double Q-Learning'