# Exercise Set 3 - Reinforcement Learning

## Advanced TD methods and approximation

Giulio Starace - 13010840

September 24, 2022

## Homework: Coding Assignment - Temporal Difference Learning

1. Coding answers have been submitted on codegra under the group "stalwart cocky sawly".

2. Hello World

## Homework: Maximization Bias

1. For the sake of clarity, we label the four outgoing actions from $B$ as $a_1$, $a_2$, $a_3$ and $a_4$, from left to right, and say they belong to the action set $A$. For expected SARSA, we use the expected SARSA update rule to determine the state-action values:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \mathbb{E}_\pi \left[ Q(S_{t+1}, A_{t+1}) | S_{t+1} \right] - Q(S_t, A_t) \right]$$

$$= Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \sum_{a \in A} \pi(a | S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]. \quad (1)$$

Because all actions from $B$ lead to a terminal state, we have that $Q(S_{t+1}, a) = 0$ for all $a \in A$ when $S_t = B$.

For $a_1$, on the first relevant sampled episode we have $R_{t+1} = 0$ giving:

$$Q(B, a_1) \leftarrow 0.7 + 0.2 \left[ 0 + 1 \times 4\overset{0}{(0.25 \times 0)} - 0.7 \right]$$
$$= 0.7 + 0.2 \left[ -0.7 \right]$$
$$= 0.56. \quad (2)$$

And on the next relevant sampled episode we get the same reward, giving:

$$Q(B, a_1) \leftarrow 0.56 + 0.2 \left[ 0 + 1 \times 4\overset{0}{(0.25 \times 0)} - 0.56 \right]$$
$$= 0.56 + 0.2 \left[ -0.56 \right]$$
$$= 0.448. \quad (3)$$

For $a_2$, on the first relevant sampled episode we have $R_{t+1} = 1$, giving:

$$Q(B, a_2) \leftarrow 0.7 + 0.2 \left[ 1 + 1 \times \cancel{4(0.25 \times 0)}^{\;0} - 0.7 \right]$$
$$= 0.7 + 0.2 \, [0.3]$$
$$= 0.76. \tag{4}$$

And on the next relevant sampled episode, we get the same reward, giving:

$$Q(B, a_2) \leftarrow 0.76 + 0.2 \left[ 1 + 1 \times \cancel{4(0.25 \times 0)}^{\;0} - 0.76 \right]$$
$$= 0.76 + 0.2 \, [0.24]$$
$$= 0.808. \tag{5}$$

For $a_3$, on the first relevant sampled episode we have $R_{t+1} = 1$, which we know from the first update to $a_2$ gives us

$$Q(B, a_3) \leftarrow 0.76. \tag{6}$$

On the next relevant sampled episode, we have $R_{t+1} = 0$, giving:

$$Q(B, a_2) \leftarrow 0.76 + 0.2 \left[ 0 + 1 \times \cancel{4(0.25 \times 0)}^{\;0} - 0.76 \right]$$
$$= 0.76 + 0.2 \, [-0.76]$$
$$= 0.608. \tag{7}$$

For $a_4$, on the first relevant sampled episode we have $R_{t+1} = 0$, which we know from the first update to $a_1$ gives us

$$Q(B, a_4) \leftarrow 0.56. \tag{8}$$

On the next relevant sampled episode, we have $R_{t+1} = 1$, giving:

$$Q(B, a_1) \leftarrow 0.56 + 0.2 \left[ 1 + 1 \times \cancel{4(0.25 \times 0)}^{\;0} - 0.56 \right]$$
$$= 0.56 + 0.2 \, [0.44]$$
$$= 0.648. \tag{9}$$

For Q-learning, we use the Q-learning update rule to determine the state-action values:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t) \right]. \tag{10}$$

Note once again that since when $S_t = B$, $S_{t+1}$ is always a terminal state, then like before $Q(S_{t+1}, a) = 0$ for all $a \in A$. Therefore, in this case, equation (10) reduces like equation (1) to

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} - Q(S_t, A_t) \right]. \tag{11}$$

Therefore, all the state-action values in state $B$ are the same in Q-learning as for expected SARSA. For a clearer summary, refer to Table 1.

Table 1: Expected SARSA and Q-learning state-action pair values for the four available actions at state $B$ after sampling two episodes per action.

| | $Q(B, a_1)$ | $Q(B, a_2)$ | $Q(B, a_3)$ | $Q(B, a_4)$ |
|---|---|---|---|---|
| **expected SARSA** | 0.448 | 0.808 | 0.608 | 0.648 |
| **Q-learning** | 0.448 | 0.808 | 0.608 | 0.648 |

2. To determine what the new $Q(A, L)$ value is when executing $L$ in $A$ after the 10 episodes, assuming that $Q(A, L)$ is still at 0.7, we use the same update rules as stated before, i.e. equation (1) for expected SARSA and equation (10) for Q-learning. Since taking $L$ at $A$ leads to a terminal state, equations (1) and (10) once again reduce to equation (11). For both expected SARSA and Q-learning we therefore have:

$$\begin{aligned} Q(A, L) &\leftarrow 0.7 + 0.2 \left[0.7 - 0.7\right] \\ &= 0.7 + 0.2 \left[0\right] \\ &= 0.7. \end{aligned} \quad (12)$$

We apply the same process to determine what the new $Q(A, R)$ value is when executing $R$ in $A$ after the 10 episodes, assuming that $Q(A, R)$ is still at 0.7. However, the reduction to equation (11) is not possible in this case, since $R$ from $A$ does not transition to a terminal state. With expected SARSA we have

$$\begin{aligned} Q(A, R) &\leftarrow 0.7 + 0.2 \left[0 + 0.25 \left(0.448 + 0.808 + 0.608 + 0.648\right) - 0.7\right] \\ &= 0.6856. \end{aligned} \quad (13)$$

With Q-learning, we have

$$\begin{aligned} Q(A, R) &\leftarrow 0.7 + 0.2 \left[0 + 0.808 - 0.7\right] \\ &= 0.7216. \end{aligned} \quad (14)$$

For a clearer summary, please refer to Table 2.

Table 2: Expected SARSA and Q-learning state-action pair values at $A$ when executing $R$ and $L$ from $A$ after the 10 sampled episodes.

| | **Expected SARSA** | **Q-learning** |
|---|---|---|
| $Q(A, L)$ | 0.7 | 0.7 |
| $Q(A, R)$ | 0.6856 | 0.7216 |

3. Assuming convergence to optimality for both Q-learning and Expected SARSA, we can obtain the true state-action values by utilising the Bellman optimality equation:

$$q_*(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right]. \quad (15)$$

The results of applying this equation to our MDP are summarised in Table 3.

4. Maximization bias can be observed in all estimated state-action values reported in Tables 1 and 2, except for $Q(A, L)$ and $Q(B, a_1)$. We observe maximization bias

Table 3: True state-action values after Expected SARSA and Q-learning convergence.

| $Q_*(A, L)$ | $Q_*(A, R)$ | $Q_*(B, a_1)$ | $Q_*(B, a_2)$ | $Q_*(B, a_3)$ | $Q_*(B, a_4)$ |
|---|---|---|---|---|---|
| 0.7 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

here as the estimated values are higher than the true values reported in Table 3, i.e. they are positively biased. Both Q-learning and expected SARSA are affected by this bias, as both algorithms rely on a greedy (target) policy which requires the use of a maximum operator. When coupled with stochastic transitions and rewards, such as in our MDP from state $B$, this generally leads to a positive bias in the estimated state-action values.

5. hello world

# Homework: Gradient Descent Methods

1. hello world

2. hello world

3. hello world