RBE 595 — Reinforcement Learning
Week #7 Assignment
Temporal Difference Learning

Arjan Gupta

Problem 1

Between DP (Dynamic Programming), MC (Monte-Carlo) and TD (Temporal Difference), which one of these algorithms use bootstrapping? Explain.

Answer

Bootstrapping is the process of updating the estimate value of a state based on the estimate value of another state.

- **Dynamic Programming** (DP) uses bootstrapping. This is because DP uses the Bellman equation to update the value of a state based on the value of a future state.
- **Monte-Carlo** (MC) does not use bootstrapping. This is because MC does not use the Bellman equation to update the value of a state based on the value of a future state. Instead, MC uses the expected return value, $\mathbb{E}_\pi[G_t \mid S_t = s]$, to update the value of a state.
- **Temporal Difference** (TD) uses bootstrapping. This is because uses a combination of DP and MC to update the value of a state based on the value of a future state. TD uses a target value of $[R_{t+1} + \gamma V(s_{t+1})]$ to update the value of a state. Since this depends on a future state, TD uses bootstrapping.

Problem 2

We mentioned that the target value for TD is $[R_{t+1} + \gamma V(s_{t+1})]$. What is the target value for Monte-carlo, Q-learning, SARSA and Expected-SARSA?

Answer

The Target is shown as part of the following equation:

$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

- **Monte-Carlo** (MC) does not use bootstrapping. Its target value is the actual return value, G_t.
- **Q-Learning** — As given in the algorithm, the target value is $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$.
- **SARSA** — As shown in the algorithm, the target value is $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$.
- **Expected-SARSA** — As described in the book, the target value is $R_{t+1} + \gamma \mathbb{E}_\pi \left[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}\right]$.

Problem 3

What are the similarities of TD and MC?

Answer

The similarities between TD and MC are as follows:

- Both TD and MC are model-free, i.e. they do not require a model of the environment.
- Both TD and MC are *sample updates*, i.e., they involve looking ahead at a sample successor state (or state-action pair), using the value of that state to compute a backed-up value, and then updating the value of the original state (or state-action pair) accordingly.

Problem 4

Assume that we have two states x and y with the current value of $V(x) = 10$, $V(y) = 1$. We run an episode of $\{x, 3, y, 0, y, 5, T\}$. What's the new estimate of $V(x)$, $V(y)$ using TD (assume step size $\alpha = 0.1$ and discount rate $\gamma = 0.9$).

Answer

The new estimate of $V(x)$ is as follows:

$$\begin{aligned} V(x) &= V(x) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(x)\right] \\ &= 10 + 0.1 \left[3 + 0.9 \cdot 1 - 10\right] \\ &= 10 + 0.1 \left[3.9 - 10\right] \\ &= 10 + 0.1 \left[-6.1\right] \\ &= 10 - 0.61 \\ &= 9.39 \end{aligned}$$

However, $V(y)$ gets updated twice in this episode. The first update is as follows:

$$\begin{aligned} V(y) &= V(y) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(y)\right] \\ &= 1 + 0.1 \left[0 + 0.9 \cdot 1 - 1\right] \\ &= 1 + 0.1 \left[0.9 - 1\right] \\ &= 1 + 0.1 \left[-0.1\right] \\ &= 1 - 0.01 \\ &= 0.99 \end{aligned}$$

The second update is as follows:

$$\begin{aligned} V(y) &= V(y) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(y)\right] \\ &= 0.99 + 0.1 \left[5 + 0.9 \cdot 0 - 0.99\right] \\ &= 0.99 + 0.1 \left[5 - 0.99\right] \\ &= 0.99 + 0.1 \left[4.01\right] \\ &= 0.99 + 0.401 \\ &= 1.391 \end{aligned}$$

Therefore, the new estimate of $V(x)$ is 9.39 and the new estimate of $V(y)$ is 1.391.

Problem 5

Can we consider TD an online (real-time) method and MC an offline method? Why?

Answer

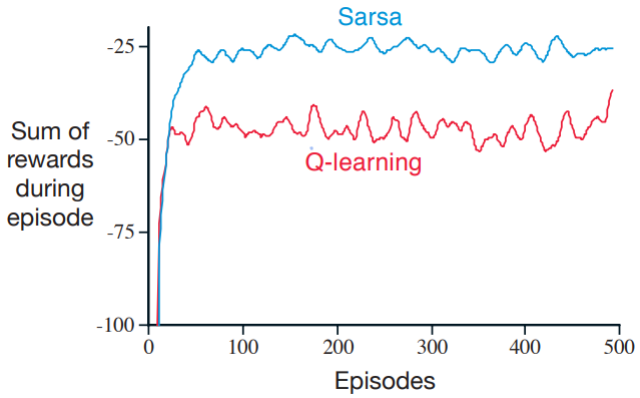
Yes, we can consider TD an online (real-time) method and MC an offline method. This is because TD learns during the episode, whereas MC learns after the episode has ended. Specifically, TD updates the value of a state based on the value of the next state (during the episode), whereas MC updates the value of a state based on successive returns (after the episode has ended).

Problem 6

Does Q-learning learn the outcome of exploratory actions? (Refer to the Cliff walking example).

Answer

Yes, Q-learning learns the outcome of exploratory actions. This is because Q-learning is an off-policy TD control algorithm. In the context of the Cliff walking example, this means that Q-learning learns the optimal action-value function, q_*, which is closest to the cliff. The behavior policy takes exploratory actions, and the target policy is greedy. This causes makes it so that initially, the agent falls off the cliff a lot, but eventually, the agent learns to avoid the cliff and converges to the optimal action-value function, q_*. This is why the graph of the sum of rewards per episode for Q-learning is initially very low, but eventually converges to a high value. The graph is shown below:



Problem 7

What is the advantage of Double Q-learning over Q-learning?

Answer

The advantage of Double Q-learning over Q-learning is that Double Q-learning is less prone to bias than Q-learning. Specifically, Q-learning is biased toward the maximum value action, which is also known as maximization bias. This is because Q-learning uses the maximum value action to update the value of a state.

In contrast, Double Q-learning is not biased toward the maximum value action. This is because Double Q-learning uses two action-value functions, Q_1 and Q_2, to update the value of a state. With probability 0.5, Double Q-learning uses either Q_1 or Q_2 to update the value of a state. Now, for example, the action is picked as per Q_1, then that action is probably not the maximum value action as per Q_2. This way, for a given state, we have two estimates of the value of the maximum value action. This reduces the bias of Double Q-learning and is hence the advantage over Q-learning.