Sumeet Shanbhag

Student ID : 641020714

Reinforcement Learning Assignment #4

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

- **Bootstrapping** refers to the idea of updating estimates based on other, current estimates.

- **Dynamic Programming (DP):** Dynamic programming methods, such as Value Iteration and Policy Iteration, compute value functions based on the Bellman equation. In these methods, the updated value of a state is based on the current estimate of the values of its successor states. This use of current estimates to update another estimate is referred to as bootstrapping.

- **Monte-Carlo (MC):** Monte-Carlo methods estimate the value function based on actual returns following a trajectory. Since the return is computed from real episodes and not based on current estimates, MC does not use bootstrapping.

- **Temporal Difference (TD):** Temporal Difference methods, like TD(0), use a combination of real reward and the estimated value of the next state to update the value estimate of the current state. Specifically, the TD error is the difference between the observed reward plus the estimated value of the next state, and the current estimated value of the state. Since the update relies on the estimated value of the next state, TD does use bootstrapping.

2. **We mentioned that the target value for TD is [Rt+1+γV(st+1)]. What is the target value for Monte-carlo, Q-learning, SARSA and Expected-SARSA.**

- **Monte-Carlo:**
  Target value: G_t

- **Q-learning:**
  Target value: $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$

- **SARSA:**
  Target value: $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

- **Expected-SARSA:**
  Target value: $R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$

3. **What are similarities of TD and MC?**

- **Model-Free**: Neither requires a model of the environment.
- **Value Function Estimation**: Both estimate state or action values.
- **Policy Evaluation and Control**: Can be used for both evaluating a policy and finding an optimal one.
- **Sampling**: They learn from actual experiences in the environment.
- **Incremental Update**: Both update value function estimates incrementally.
- **Exploration**: Both benefit from exploration strategies.
- **Generalization**: Can be combined with function approximation.
- **Discounting**: Both use a discount factor to weigh future rewards.

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 α=0.1 and discount rate γ=0.9)**

Given the episode x, 3, y, 0, y, 5, T:

Initial values:
V(x)=10
V(y)=1

Parameters:
α=0.1
γ=0.9

From state x to state y:
TD target = immediate reward + discounted value of next state
= 3+0.9(1)
= 3.9

TD error = TD target - V(x)
= 3.9 - 10
= - 6.1

Using the TD update rule:
V(x)⟵V(x)+α×TD error
V(x) = 10+0.1(−6.1)
V(x) = 9.39

From state y to state y:
TD target = immediate reward + γ × V(y)

= 0 + 0.9(1)
= 0.9

TD error = TD target - V(y)
= 0.9 – 1
= - 0.1

Using the TD update rule:
V(y) ← V(y) + α × TD error
V(y) = 1 + 0.1(−0.1)
V(y) = 0.99

From state y to terminal state T:
TD target = immediate reward (since terminal state's value is 0) = 5

TD error = TD target - V(y)
= 5 - 0.99
= 4.01

Using the TD update rule:
V(y) ← V(y) + α × TD error
V(y) = 0.99 + 0.1(4.01)
V(y) = 1.391

So, the updated values are:
V(x) = 9.39
V(y) = 1.391

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

Temporal Difference (TD):

- **Online (Real-time) Method**: TD updates value estimates at every time step as it experiences the environment. After each transition (state, action, reward, next state), it immediately uses this information to adjust its value estimates. This allows TD methods to learn and adapt in real-time without waiting for a complete episode to finish.
- The primary characteristic of TD that makes it an online method is its use of bootstrapping. It updates value estimates based on the immediate reward and the estimated value of the next state, rather than waiting for a full return.

Monte Carlo (MC):

- **Offline Method**: MC methods require the completion of an entire episode before updating the value estimates. It uses the full return (cumulative reward) from the current time step until the end of the episode to perform updates. This means that no learning updates occur mid-episode, and all updates are done after the episode has concluded.

- MC doesn't bootstrap like TD; it relies purely on the actual returns from episodes, which requires waiting until the episode's conclusion.

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

   When Q-learning takes an exploratory step and receives a reward (or penalty) as a result, it updates its Q-table using that reward. This means that the Q-table is being updated based on both exploratory and exploitative actions. Over time, as the agent explores more and the Q-values converge, the agent builds a more accurate understanding of the environment's dynamics and the expected rewards associated with different actions in different states.

   In the cliff walking problem, if the agent randomly decides to take an exploratory action that results in it "falling off the cliff," it will receive a significant penalty. This penalty is then used to update the Q-value for that state-action pair, teaching the agent about the consequences of that action in that state.

7. **What is the advantage of Double Q-learning over Q-learning?**

- **Overestimation Bias Reduction**: Q-learning can sometimes overestimate the Q-values of actions, especially in environments with noise or uncertainty. This overestimation is because Q-learning always takes the maximum Q-value estimate when updating its values. In situations where there's noise or randomness in the reward structure, this can lead to systematic overestimation. Double Q-learning reduces this bias by using two separate Q-tables and decoupling the selection and evaluation of the action.
- **Improved Convergence**: By reducing the overestimation bias, Double Q-learning can lead to better convergence in some scenarios, especially in environments with stochastic rewards.
- **Better Action Selection**: Since Double Q-learning is less prone to overestimating Q-values, it may make better-informed decisions about which actions to take, especially in the early stages of learning when the Q-values are still evolving.
- **Stability in Learning**: Double Q-learning can provide more stable learning in some environments where Q-learning might oscillate or diverge.
- **Robustness in Complex Environments**: Double Q-learning has been found to be more robust than standard Q-learning in certain complex environments, especially those where the reward structure is not straightforward.