Experiment No.5
Implementing Monte Carlo control and Temporal Difference (TD) learning algorithms to train an agent in a grid-world environment.
Date of Performance:
Date of Submission:

Aim: Implementing Monte Carlo control and Temporal Difference (TD) learning algorithms to train an agent in a grid-world environment.

Objective: The objective is to implement Monte Carlo control and Temporal Difference (TD) learning algorithms to train an agent within a grid-world environment, aiming to understand and compare the learning dynamics and performance of these two reinforcement learning methods while navigating a simplified spatial domain.

Theory:

Monte Carlo Control

Monte Carlo methods are a class of algorithms used in reinforcement learning for estimating value functions and finding optimal policies. Unlike dynamic programming methods, Monte Carlo methods do not require a model of the environment and instead learn directly from experience by interacting with the environment.

Monte Carlo control combines Monte Carlo prediction with policy improvement to find the optimal policy directly from experience. After estimating the value function, the policy is improved by selecting actions that maximize the value function estimates.

Monte Carlo methods learn from complete episodes of experience. Here's the algorithm for Monte Carlo Control:

Input:

- MDP: (S, A, P, R, γ), where
  - S is the set of states.
  - A is the set of actions.
  - P is the transition probability matrix.
  - R is the reward function.
  - γ is the discount factor.
- Number of episodes **num_episodes**.
- Exploration rate **epsilon**.

Algorithm:

- Optimal policy π.

1. **Initialize**:
   - Q-function: Q(s, a) for all s in S and a in A.
   - State-action visit count: N(s, a) for all s in S and a in A.
   - Policy π arbitrarily.
2. **For** each episode **from 1 to num_episodes**:
   - Generate an episode following π: (s1, a1, r1), (s2, a2, r2), ..., (sT, aT, rT).
   - Set G = 0.
   - **For** each time step **t** from T down to 1**:

- Update return G: G = γ * G + rt+1.
- **If** (st, at) has not been visited in this episode before:
  - Increment visit count: N(st, at) += 1.
  - Update action-value function: Q(st, at) = Q(st, at) + (1/N(st, at)) * (G - Q(st, at)).
  - Update policy: π(st) = argmax(Q(st, a)) for all a in A.

3. **Return** the optimal policy π.

Temporal Difference (TD) Learning

Temporal Difference learning methods update estimates based on the difference between current and future estimates. Here's the algorithm for TD Learning:

Input:

- MDP: (S, A, P, R, γ), where
  - S is the set of states.
  - A is the set of actions.
  - P is the transition probability matrix.
  - R is the reward function.
  - γ is the discount factor.
- Number of episodes **num_episodes**.
- Step size **alpha**.
- Exploration rate **epsilon**.

Algorithm:

- Optimal policy π.

1. **Initialize**:
   - Q-function: Q(s, a) for all s in S and a in A.
   - Policy π arbitrarily.
2. **For** each episode **from 1 to num_episodes**:
   - Initialize state s as the starting state.
   - **While** s is not a terminal state:
     - Choose action a using policy derived from Q (e.g., ε-greedy).
     - Take action a, observe reward r and next state s'.
     - Update action-value function: Q(s, a) = Q(s, a) + α * (r + γ * max(Q(s', a')) - Q(s, a)).
     - Update policy: π(s) = argmax(Q(s, a)) for all a in A.
     - Set s = s'.
3. **Return** the optimal policy π.

These are high-level algorithms. The specific implementation details and additional considerations (like exploration strategy, eligibility traces, etc.) might vary based on the problem and requirements.

Conclusion:

1. How do Monte Carlo control and Temporal Difference (TD) learning algorithms differ in training an agent within a grid-world environment?
2. Compare and contrast the key characteristics, advantages, and limitations of each approach.