**Assignment 4**

By Harsh Sharma

Part 1

**Part I. Fundamental Concepts of Reinforcement**

**Question 1.** Define reinforcement learning. How does it differ from supervised and unsupervised learning?

Ans: Reinforcement learning learns from interaction with an environment, receiving feedback in the form of rewards or penalties. The agent explores various actions and learns to associate them with positive or negative outcomes, aiming to maximize cumulative rewards over time.

 **Supervised Learning:** In supervised learning, the algorithm learns from labeled training data. The feedback is in the form of corrected output labels. It's given input data along with the correct output, and the goal is for the algorithm to learn a mapping between inputs and outputs. For instance, in image recognition, the algorithm is trained on labeled images to correctly identify objects or patterns.

 **Unsupervised Learning:** Unsupervised learning deals with unlabeled data, aiming to find hidden patterns or structures within the data. It doesn't have specific output labels to predict, but it discovers relationships or structures inherent in the dataset. Clustering and dimensionality reduction are examples where unsupervised learning is applied, grouping similar data points together without explicit guidance.

While supervised learning learns from labeled input-output pairs and unsupervised learning focuses on finding patterns in unlabeled data, reinforcement learning involves an agent learning from its actions and consequences in an environment, striving to maximize cumulative rewards.

**Question 2.** What is a Markov Decision Process (MDP)?

Answer: A Markov Decision Process (MDP) is a mathematical framework used to model decision-making in situations where an agent operates in an environment and makes sequential decisions. It's named after the mathematician Andrey Markov and is fundamental in the field of reinforcement learning.

An MDP is defined by a tuple of components:

1. **States (S):** A set of possible situations or configurations the agent can be in.
2. **Actions (A):** A set of possible moves or decisions the agent can take in each state.
3. **Transition Model (T):** Describes the probability of transitioning from one state to another after taking a specific action. It provides the conditional probability distribution of the next state given the current state and action.
4. **Reward Function (R):** Defines the immediate reward the agent receives after taking a specific action in a particular state. It indicates the immediate benefit or cost associated with the action.
5. **Discount Factor (γ):** Represents the importance of future rewards compared to immediate rewards. It's a value between 0 and 1, where 0 indicates the agent only cares about immediate rewards and 1 values future rewards equally.

The key assumption in an MDP is the Markov property, which states that the future state depends solely on the current state and action, independent of the previous states and actions, given the current state and action.

**Question 3**. Explain the Markov property. Why is it important in the context of MDPs?

Answer: The Markov property is a fundamental concept in probability theory that describes a stochastic process where the future state depends only on the current state and is independent of the sequence of events that preceded it, given the present state and action.

Formally, a process exhibits the Markov property if the conditional probability distribution of future states depends only on the present state and action and is not influenced by the sequence of events that led to the current state:

P[St+1∣St,At,St−1,At−1,…,S1,A1]=P[St+1∣St,At]P[St+1​∣St​,At​,St−1​,At−1​,…,S1​,A1​]=P[St+1​∣St​,At​]

In simpler terms, knowing the current state and action is sufficient to predict the probabilities of future states, and the history of the process beyond the current state and action doesn’t add any extra predictive power.

The Markov property is crucial because it simplifies the representation and computation of the system's behavior. It allows for the modeling of sequential decision-making problems where an agent interacts with an environment by selecting actions and transitioning between states. The Markov property ensures that the state transitions and rewards depend only on the current state and action, enabling the use of dynamic programming and reinforcement learning algorithms to find optimal policies without needing to consider the entire history of states and actions.

**Question 4**. What is the difference between a value function and a Q-function (or action-value function)?

Answer: The primary distinction between a value function and a Q-function lies in what they measure:

* **Value Function (V-function):** Measures the value of being in a particular state under a given policy. It provides a single value for each state, indicating how good it is to be in that state regardless of the action taken.
* **Q-Function (Action-Value Function):** Measures the value of being in a particular state and taking a specific action under a given policy. It provides a value for each state-action pair, indicating the expected return if that action is taken from that state under the given policy.

**Question 5.** Describe the exploration-exploitation trade-off in reinforcement learning.

Answer: The exploration-exploitation trade-off is a fundamental challenge in reinforcement learning, where an agent needs to balance between exploiting its current knowledge to maximize immediate rewards and exploring unknown areas to gather more information for potentially higher rewards in the future.

* **Exploitation:** Exploitation involves using the current knowledge or learned strategies to choose actions that are expected to yield the highest immediate reward based on the agent's understanding of the environment. Exploitation aims to make the most of the known information to maximize short-term gain.
* **Exploration:** Exploration involves trying out new actions or visiting less familiar states to gain additional information about the environment. By exploring, the agent can discover potentially better strategies or uncover unknown but rewarding states or actions that might lead to higher long-term rewards.

The trade-off arises because focusing too much on exploitation can lead to suboptimal behavior if the agent settles for actions that it believes are the best based on its current knowledge but might not truly be optimal. Conversely, excessive exploration may result in inefficient behavior, as the agent spends too much time exploring and doesn't exploit its current knowledge to gain immediate rewards.

**Question 6.** What is the ε-greedy strategy, and how does it balance exploration and exploitation?

Answer: The ε-greedy strategy is a popular method used in reinforcement learning to balance the exploration-exploitation trade-off. It combines both exploration of unknown options and exploitation of the current best-known option by allowing the agent to choose between these two strategies based on a predefined parameter ε (epsilon).

Here's how the ε-greedy strategy works:

1. **Exploitation (Greedy):** With probability 1 - ε, the agent chooses the action that it believes has the highest estimated value based on its current knowledge or learned policy. This is the exploitation part where the agent selects what it considers to be the best action according to its understanding of the environment.
2. **Exploration (Random Exploration):** With probability ε, the agent takes a random action regardless of its current knowledge or estimated value. This random exploration allows the agent to try out different actions, even those that might not seem optimal based on its current information.

By adjusting the value of ε, the agent can control the balance between exploration and exploitation:

* When ε is set to a higher value, the agent explores more frequently, taking random actions more often. This can be beneficial in the early stages of learning or in complex environments where the agent needs to discover more about the environment.
* When ε is set to a lower value, the agent tends to exploit more, relying on its current knowledge to choose actions that are considered the best based on its learning so far. This is useful when the agent has already gathered substantial knowledge and wants to maximize its performance based on what it has learned.

**2.** Describe the pseudocode of **Q-Learning** algorithm

Solution: Consider the following pseudo code:

1: Initialize all entries in the action-value table to random values (except for terminal states which receive a value of 0).

2: For each episode do

3: Reset state to the initial agent state

4: Repeat

5: Select an action based on policy in current state St and action-value function Q

6: Take action and observe reward r and new state St+1

7: Update the record in the action-value table for the action taken in the last state St using the formula:

8: Q(St, a) <- Q(St, a) + α \* (r + γ \* max(Q(St+1, a')) - Q(St, a))

9: Let St = St+1

10: Until agent reaches a terminal state

11: End for

The breakdown of the pseudo code:

Behavior Policy: Describes the policy that selects actions.

Action-Value Function (Require): Denoted as Q, it's a function that looks up entries in an action-value table, representing possible actions 'a' in a given state 's'.

Learning Rate (α): Denoted as 'a', it determines how much the Q-values are updated based on new information.

Discount Rate (γ): Denoted as '~', it represents the importance of future rewards compared to immediate rewards.

Number of Episodes: Specifies the total number of episodes or iterations for the learning process.

This pseudocode outlines the Q-learning process, initializing Q-values, taking actions based on a policy, updating Q-values using the Q-learning formula, and iterating through episodes until reaching a terminal state.