Practical 6

Problem Statement:

Build a Tic-Tac-Toe game using reinforcement learning in Python by using following tasks
a. Setting up the environment
b. Defining the Tic-Tac-Toe game
c. Building the reinforcement learning model
d. Training the model
e. Testing the model

Objective:

To design and implement a Tic-Tac-Toe game where an agent learns optimal gameplay using Q-learning.

Outcome:

After implementing the reinforcement learning on Tic-Tac-Toe game, students will adeptly utilize reinforcement learning methods to solve very complex problems that cannot be solved by conventional techniques.

Theory:

Reinforcement Learning is a type of machine learning paradigm where an agent learns how to behave in an environment by performing certain actions and receiving rewards or penalties in return. It's inspired by behavioural psychology and involves agents who take actions in an environment to maximize some notion of cumulative reward.

Key Characteristics:

1) Agent: The decision-maker or learner.
2) Environment: Everything the agent interacts with.
3) State (s): A situation returned by the environment.
4) Action (a): What the agent can do.
5) Policy (π): A strategy that defines the learning agent's way of behaving at a given time. It's a map from states to actions.
6) Reward (R): A feedback from the environment. It can be immediate or delayed.
7)Value Function (V): It predicts the expected cumulative future reward for a given state. Essentially, it tells the worth of a state.
8)Q-function (or Action-Value Function): Represents the expected return of taking an action in a particular state, then following policy π.

Learning Process:

- The agent observes the current state and takes an action.
- The environment, upon receiving the action, transitions to a new state.
- The environment then gives a reward (or penalty) to the agent based on the action taken.

- The agent uses this reward to update its knowledge or policy.

The goal of the agent is to maximize the expected cumulative reward over time, often referred to as the **discounted future reward**. The agent does this by learning the optimal policy that gives the highest reward over the long run.

Exploration vs. Exploitation:

In RL, there's a continuous dilemma of Exploration vs. Exploitation:

Exploration: Trying out new actions to see their effects.

Exploitation: Using the known actions that yield the maximum reward.

Algorithm:

1 . Setting up the environment

a) Install Required Libraries:

pip install numpy

b) Initialize Variables:

States: All possible positions on the Tic-Tac-Toe board.

Actions: All possible moves a player can make.

Rewards: Assign rewards for winning, losing, drawing, or making a move.

2. Defining the Tic-Tac-Toe game

- **Board Repesentation:** Use a 3x3 matrix or a list of 9 elements to represent the board.
- **Move Function:** Given a state and an action, return the resulting state.
- **Check End Function:** Check if the game has ended, and return the result (win, draw, or lose).
- **Reset Function:** Resets the game to its initial state.

3. Building the reinforcement learning model

1. Q-Table Initialization: Create a table with states as rows and actions as columns. Initially, all values are zeros.

2. Policy Function: Given a state, decide which action to take. You can start with a random policy and refine it as you go.

3. Q-Learning Algorithm:

- Choose an action using the policy.
- Take the action, then observe the reward and the new state.
- Update the Q-value of the taken action using the observed reward and the highest Q-value of the new state.
- Loop until the game ends.

4. Training the model

1. **Episodic Learning:** Play the game multiple times (episodes). In each episode:
   - Start with an initial state.
   - Play the game until it ends, updating the Q-values using the Q-learning algorithm.
   - Adjust the policy gradually to rely more on the Q-values and less on random choices.

2. **Adjust Learning Rate and Discount Factor:** Start with a high learning rate and decrease it over time. Use a discount factor (usually denoted as gamma) to balance immediate vs. future rewards.C
3. **Exploration-Exploitation Trade-off:** Initially, make the agent explore more (choose random moves). Gradually, make it exploit its learned knowledge.

5. Testing the model

1. **Evaluation Metric:** Decide on a metric to evaluate the model's performance. For example, the percentage of games won over a set number of games.
2. **Play Games:** Use the trained model to play a certain number of games. Ideally, the agent should win most of the games against a random opponent and play optimally against a smart opponent.
3. **Observe and Adjust:** If the performance isn't satisfactory, you may need to adjust hyperparameters, train for more episodes, or modify the reward structure.

Conclusion:

Building a Tic-Tac-Toe game using reinforcement learning in Python allows for the development of an AI agent capable of understanding game dynamics and making optimal moves. Through iterative training and testing, the model can achieve competent gameplay, potentially matching or surpassing human performance.

Oral Questions:

1. What is reinforcement learning and how does it differ from supervised and unsupervised learning?
2. What are some real-world applications of reinforcement learning?
3. How does the Q-learning algorithm work, and what is the Q-function?
4. What is the role of a reward signal in reinforcement learning?