Graduate Certificate in Artificial Intelligence with Machine Learning
AIGC 5504 – Emerging Technologies in Artificial Intelligence

Lab 10: Implementing Q-Learning for Reinforcement Learning

Submission guidelines:

- For this lab, you will need to submit 1 PDF file.
- Convert your codes to PDF.
- Name the PDF as follows: firstname_lastname_LAB 10.pdf
- Go to the course Blackboard → Labs folder → Lab Exercises 10 and submit the pdf.

Lab goals:

- Implement Q-Learning to solve a basic reinforcement learning problem.
- Understand key RL concepts like states, actions, rewards, and policies.
- Understand how neural networks approximate Q-values.
- Explore the impact of different parameters (e.g., learning rate, discount factor).

Part 1: Understanding the Problem

1. **Scenario:**
   - A robot is navigating a 5x5 grid to reach a goal while avoiding obstacles.
   - Each grid cell is a state, and actions are moving up, down, left, or right.
2. **Objective:**
   - Train a neural network to approximate the Q-values for state-action pairs and find the optimal path to the goal.

Part 2: Setting Up the Environment

1. **Grid Environment:**
   - Create a 5x5 grid where:
     - Top-left corner (0, 0) is the start.
     - Bottom-right corner (4, 4) is the goal.
     - Randomly place 3 obstacles.
2. **Rewards:**
   - +10 for reaching the goal.
   - -10 for hitting an obstacle.
   - 0 for all other states.
3. **State Representation:**
   - Represent states as one-hot encoded vectors or scaled grid coordinates (e.g., normalized (x, y) coordinates).

Part 3: Implementing Deep Q-Learning

1. **Neural Network Architecture:**
   - Input: State representation (5x5 grid flattened to 25 inputs or (x, y) coordinates).
   - Hidden Layers: 2 fully connected layers with ReLU activation (e.g., 64 neurons each).
   - Output: Q-values for each action (4 outputs for up, down, left, right).
2. **Algorithm Steps:**
   - Initialize the neural network for Q-value approximation.
   - Use experience replay to store and sample experiences (s,a,r,s′)(s, a, r, s')(s,a,r,s′).
   - Implement ϵ\epsilonϵ-greedy policy for exploration.
   - Train the network using the Bellman equation
   - Update the network weights to minimize the loss
3. **Steps:**
   - Initialize replay memory (e.g., capacity of 10,000 experiences).
   - For each episode:
     1. Start at the initial state.
     2. Select an action using ϵ\epsilonϵ-greedy policy.
     3. Take the action and observe the next state and reward.
     4. Store the transition (s,a,r,s′) in replay memory.
     5. Sample a batch of transitions from memory.
     6. Train the network to minimize loss.
     7. Update the target network periodically (e.g., every 10 episodes).

Part 4: Visualizing Results

1. **Performance Metrics:**
   - Plot total reward per episode.
   - Visualize the optimal path from start to goal on the grid.
2. **Q-Value Approximation:**
   - Show the predicted Q-values for some states.

Part 5: Reflection Questions

Answer the following in your PDF submission:

1. How does using neural networks improve over traditional Q-tables?
2. What challenges did you face when implementing the Deep Q-Learning algorithm?
3. How do hyperparameters affect training?
4. Suggest a real-world application of Deep Reinforcement Learning and explain its implementation.

Enjoy!