Assignment 3

Reinforcement Learning

CISC 856, Fall, 2023

Due Tuesday, November 28, 2023, before Midnight

Reinforcement Learning with Lunar Heist

Overview:

In this assignment, you will apply reinforcement learning (RL) algorithms to a custom environment called *"Lunar Heist"*. This environment is a variation of the classic "Lunar Lander" with additional challenges like avoiding mines and collecting minerals. You are provided with the '*lunar_heist_env.py'* file containing the environment class, and your task is to implement two RL agents - a Q-Learning agent and a Deep Q-Network (DQN) agent - in the '*agents.py'* file. You will also work with a template '*main.py'* file that sets up the environment and training loop, which you will complete as part of the assignment.

Objectives:

1. Implement a Q-Learning agent for discrete state-action spaces.
2. Implement a DQN agent using a neural network for function approximation.
3. Integrate both agents into the provided main.py training loop and analyze their performances.

Files Provided:

- *'lunar_heist_env.py'*: The custom Lunar Heist gym environment.
- *'main.py'*: The training loop setup where you will integrate your agents.
- *'Agents.py'*: Template file where you will write your QLearning and DQN agent classes.

Tasks:

1. QLearningAgent Class:
    a. Complete the QLearningAgent class in Agents.py.
    b. Implement the methods discretize_state, choose_action, update, update_epsilon, and update_alpha.

2. DQNAgent Class:
    a. Complete the DQNAgent class in Agents.py.
    b. Use a neural network for value function approximation.
    c. Implement the methods choose_action, remember, replay, and any additional methods you find necessary.
3. main.py:
    a. Use the provided template to set up the environment and the training loop.
    b. Instantiate both agents and train them in the Lunar Heist environment.
    c. Collect and log training performance metrics such as total reward, number of minerals collected, and mines hit per episode.
    d. Implement necessary components to render and visualize the environment periodically during training (optional).
4. Analysis:
    a. Compare the performance of the Q-Learning and DQN agents.
    b. Discuss the strengths and weaknesses of each approach based on your observations.

Evaluation Criteria:

Correct implementation of the Q-Learning and DQN algorithms. (50 Points)

Ability to successfully integrate agents into the main.py training loop. (10 Points)

Analyzing the performance of agents and suggestions for improving the performance. (20 Points)

Quality of the analysis comparing the two agents. (20 Points)

Submission:

Submit the completed Agents.py and main.py files along with a report detailing your implementation, analysis, and conclusions.

Notes:

Feel free to use external libraries like TensorFlow or PyTorch for implementing the DQN agent.

Make sure to add comments and document your code for clarity.

Keep hyperparameters configurable so that they can be easily tuned.