Introduction to Reinforcement Learning Course - Home Assignment 2

Dr. Teddy Lazebnik

1.10.2024

Abstract

This document outlines the second home assignment for an introductory Reinforcement Learning course. The assignment aims to provide students with an experience with deep RL in a relatively simple environment. Students are tasked with designing a neural network based RL agent for continuous space tasks and making several design decisions to obtain good performance.

Objective

This assignment focuses on applying deep learning techniques to reinforcement learning. You will implement and experiment with deep Q-networks (DQN) for solving a classic control problem.

Task 1 - theory (30% of the score)

Question 1: Explain the concept of the Q-learning update rule. How does it relate to the Bellman optimality equation? What are the potential challenges in using this update rule?

Question 2: Discuss the role of experience replay in DQN. How does it help to address the correlation problem in reinforcement learning?

Question 3: Explain the concept of the target network in DQN. How does it help to stabilize the learning process?

Task 2 - practice (70% of the score)

Implement a DQN agent to control a lunar lander, aiming to land it safely on the lunar surface while minimizing fuel consumption.

Environment: Use the OpenAI Gym LunarLander-v2 environment. Understand the state space (position, velocity, angle, angular velocity, leg contact), action space (main engine, left engine, right engine), and reward function (landing pad, penalties for crash, fuel consumption, altitude).

Implementing the DQN Agent:

- Neural Network Architecture: Design a neural network with an appropriate number of layers and neurons. The input layer should match the dimensionality of the state space. The output layer should have the same number of neurons as the number of possible actions.
- Experience Replay Buffer: Create a circular buffer to store experiences (state, action, reward, next state, done). Randomly sample batches from the buffer for training.
- Epsilon-Greedy Exploration: implement an epsilon-greedy policy to balance exploration and exploitation. Gradually decrease the epsilon value over time.
- Target Network: Create a copy of the main Q-network and update it periodically. Use the target network to compute the target Q-values.

Steps:

- Train the DQN agent to successfully land the lunar lander while minimizing fuel consumption.
- Evaluate the agent's performance based on success rate, fuel consumption, and landing accuracy.
- Bonus: visualize the lander's behavior during training and testing.

Tips:

- Handle the continuous action space (e.g., using discretization or actor-critic methods).
- Address the sparse reward problem (e.g., shaping rewards).
- Consider the impact of different hyperparameters on performance.

To be exact, students should submit a PDF report, Python/JS code, and visualization. For the PDF report, the report should be formated in a similar way to HW 1. In addition, the code used to implement the value iteration algorithm. This should be included as a ZIP file with the code, readme file, and requirements.txt with all the required library installs. For the visualizations: 1) a clear and informative visualization of the grid world, including obstacles, starting state, and goal state; 2) A visualization of the computed value function for each state; and 3) A visualization of the optimal policy, indicating the best action for each state.