DDA 4230: Reinforcement learning **CUHKSZ**

Assignment 4

TA: Bo Yue, Hengming Zhang

Due Date: Dec. 21st, 11:59 pm
Total points available: 100 points

Note: *Please note that external references are allowed only if you give an appropriate reference. There is no required format of reference. Please elaborate on your answers as well (do not just give a number, etc).*

Problem 1: DQN Implementation [50 points]

In this problem, you will implement the famous Deep Q-Network (DQN) on the game of Cartpole using the OpenAI Gym. The goals of this assignment are to (1) understand how deep reinforcement learning works when interacting with the pixel-level information of an environment and (2) implement a value-based algorithm to train an RL agent.

The CartPole environment is a classic RL benchmark designed to test an agent's ability to balance a pole on a moving cart by applying force in either direction. The agent observes a 4-dimensional state comprising the cart's position and velocity, the pole's angle, and its angular velocity. Actions are discrete, allowing the agent to push the cart left or right. The goal is to maximize the duration of balance, earning a reward of +1 at each time step. Episodes terminate when the pole falls beyond ±12° from vertical, the cart moves outside ±2.4 units, or a time limit is reached (typically 500 steps in the extended version, CartPole-v1). This unstable system is an excellent tool for testing RL algorithms like DQN. The detailed description can be found in `https://www.gymlibrary.dev/environments/classic_control/cart_pole/`.

Code skeleton is given at `https://colab.research.google.com/drive/12v9EUMqLm40az58TMqoENpCvCm5TJTZ1?usp=sharing`. Please copy the file to your folders and complete the DQN algorithm following the instructions in the notebook. Submit your file in .ipynb. Do not revise the original files in the provided link.

Problem 2: PPO Implementation [50 points]

In this part, we'll learn about Proximal Policy Optimization (PPO), an architecture that improves our agent's training stability by avoiding policy updates that are too large. To do that, we use a ratio that indicates the difference between our current and old policy and clip this ratio to a specific range [1-ϵ, 1+ϵ]. Doing this will ensure that our policy update will not be too large and that the training is more stable. We will build PPO from scratch to solve a continuous control problem, i.e. Pendulum-v1 in gym.

The Pendulum-v1 environment is a classic control problem designed to teach an agent to swing up and balance an inverted pendulum. The agent controls the torque applied to the pendulum, with the objective of keeping it upright and as close to a vertical position as possible. The state space consists of three continuous variables: the pendulum's cosine and sine angles (representing its orientation) and its angular velocity. The action space is a single continuous value, representing the torque applied to the pendulum, within a user-defined range. The reward function penalizes the agent based on the angle deviation from vertical and the magnitude of applied torque, encouraging energy-efficient control. The episode terminates after 200 steps, and the environment is typically used to test algorithms for continuous action spaces, such as PPO. The detailed description can be found in `https://www.gymlibrary.dev/environments/classic_control/pendulum/`.

Code skeleton is given at `https://colab.research.google.com/drive/1QK-QZR0cdHEwcDiIDcyiBZyeCXC3Hj-r?usp=sharing`. Please copy the file to your folders and complete the PPO algorithm following the instructions in the notebook. Submit your file in .ipynb. Do not revise the original files in the provided link.

Some Useful Materials

For more details about the implementation of advanced RL algorithms, you can check the following links and tutorials. **These parts are unnecessary for finishing this assignment.**

1. OpenAI Spinning Up
2. Stable-baselines3
3. The 37 implementation details of proximal policy optimization
4. CleanRL (Clean Implementation of RL Algorithms)