Online Optimization and Learning (CS245)

Name: ID: Email:

Rules:

1. Deadline: **2025/4/30/00:00:00**.
   The grade of the late submission subjects to the decaying policy $(75\%, 50\%, 25\%)$.
2. Please do latex your homework and no handwriting is accepted.
3. Submit your homework to TA(guohq@shanghaitech.edu.cn), including your PDF and Code, with filename "name+id+CS245HW3.zip".
4. Plagiarism is not allowed. You will fail this homework if any plagiarism is detected.

Problem 1: Policy Gradient for Multi-armed Bandits

The policy gradient method is a central technique in reinforcement learning, aiming to directly optimize the policy by estimating the gradient of the expected reward with respect to policy parameters. Specifically, given a parameterized stochastic policy $\pi_\theta(a|s)$, the objective is to maximize the expected reward:

$$J(\theta) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[r(a)].$$

Policy Gradient for Multi-armed Bandits

Setting: Consider a finite-armed stochastic bandit problem with finite action set $\mathcal{A}$, and a fixed/dummy state s_0. At each round, the learner selects an action $a \in \mathcal{A}$ from a parameterized policy $\pi_\theta(a|s_0)$ or just $\pi_\theta(a)$, and observes a stochastic reward $R(a)$ with mean value $r(a) = \mathbb{E}[R(a)]$.

We study two parameterizations of the policy:

(a) **Direct probability parameterization:** $\pi_\theta(a) = \theta(a)$.

(b) **Softmax parameterization:** $\pi_\theta(a) = \frac{e^{\theta(a)}}{\sum_{a'\in\mathcal{A}} e^{\theta(a')}}$.

Let $a_t \sim \pi_\theta(\cdot)$ be the action sampled at round t, and $R(a_t)$ the observed reward. Please answer the following questions (please provide the detailed steps to justify your answers):

(1) (Direct Parameterization) Derive the policy gradient update rule for case (a).

(2) (Softmax Parameterization) Derive the policy gradient update rule for case (b).

Solution:

Problem 2: Reinforcement Learning in the MountainCar Environment

Task: The MountainCar environment is a classic control problem where the agent (a car) is placed in a valley and needs to build up momentum to reach the top of the hill on the right. However, the engine is not powerful enough to drive the car up the hill directly. The agent must learn to leverage gravity and swing back and forth to gain enough momentum.

Environment Description: At each time step, the agent receives a state $s_t = (\text{position}, \text{velocity})$ and selects one of three discrete actions:

- 0: push left
- 1: no push
- 2: push right

The reward is -1 for each time step until the goal is reached (position $\geqslant 0.5$), which encourages the agent to solve the task in as few steps as possible. The episode ends when the goal is reached or after 200 time steps.

You can view a detailed description in the Gymnasium Documentation. You are asked to design:

- Policy-based algorithm
- Value-based algorithm
- Your algorithms (optional)

Please:

- Plot the learning curves of accumulated reward per episode.
- Describe the algorithm you implemented and the challenges specific to this environment (e.g., sparse reward, exploration).

Please implement the algorithm code in Python 3 and ensure your code can be run and evaluated.