Reinforcement Learning Task

Task Overview

You are assigned to go through a Reinforcement Learning task. This task will help you understand the basics of RL and how it can be applied to solve simple yet interesting problems. Also, a simple API must be implemented in order to familiarise with the concept of building two software components to communicate with each other.

Your main objective is to train an RL agent to navigate through a slippery frozen lake to reach the goal safely. The agent must learn to avoid holes (represented by 'H' in the environment) and reach the goal ('G') using the least number of steps possible.

Objectives

Environment Setup

The first step in our implementation is to set up the Python environment which will be used to develop this task.

In order to setup a RL environment, we need to:

- Install Python in your local machine
- Create a virtual Python environment
- Install prerequisites libraries
    - OpenAIGym (gymnasium)
    - stable-baselines3
    - Tensorboard (recommended for plots)
    - Flask (recommended for API)
    - Requests

Feel free to add any other libraries of your preference.

Environment and Algorithm Selection

The environment we are going to experiment on is the FrozenLake-v1 (map_name='4x4').
https://gymnasium.farama.org/environments/toy_text/frozen_lake/

After setting up the environment, we need to train our first agent on this environment using an appropriate algorithm found in stable-baselines3. This library provides a large variety of algorithms and tools to the user in PyTorch. Feel free to use any of the available algorithms. We recommend using two of the most common ones:

- DQN (Deep Q Network)
- PPO (Proximal Policy Optimization)

Training

When the environment is ready and the algorithms of your choice are implemented, we need to train the agent. Select the hyperparameter set of your choosing and execute as many training sessions as possible in order to achieve the highest possible reward.
Additionally, please provide the following information:

- Overview of the observation-space, action-space and agent's reward logic
- Training curves from the best training from each of the two selected algorithms (recommended to use: tensorboard)

Evaluation using API

After the agent is trained, we must evaluate the trained agent's performance in terms of successful episode completion and average reward.
The current objective is to implement an environment API in order for your agents to communicate with it. More specifically, you will implement the following functions on your envAPI:

- new_game()
- reset()
- step()

In order to evaluate your agents, you will need to reproduce the following steps:

- Send a new_game request to your environment, and receive a unique id of the current environment
- Send a reset request and receive the initial observation
- Predict an action based on the current observation from the environment and send a request to the API to receive the next observation, reward, done and info

The above procedure must be executed for 100 games and a sufficient amount of steps for each game. The evaluation of the agent is calculated using the evaluate_policy from stable-baselines3.
You can find attached a structure of the FrozenLakeAPI_structure.py code to build on.
Recommended to use: requests, flask .

Algorithm comparison

After evaluating both of your agents, please compare your final results (training curves, evaluation scores, etc). Based on your results, which algorithm would you recommend between those two for this environment and why?

Reward Wrapper (Bonus Task)

Wrappers can be used to modify an existing environment without having to alter the base code of the environment. For example, we can create a Wrapper to our existing environment in order to change the reward logic for every step.
Try creating this Wrapper and modifying the step() function to return -1 for every step and 10 when it reaches the final goal.
Retrain the agent. Do you notice any difference in the final results?

Deliverable

Please provide a brief overview of the procedure you followed for each objective with the results of your experiments along with the code you developed.
The above must be included in a zip file.

Use of AI tools like Copilot, ChatGPT, or similar for this task is strictly prohibited. We employ sophisticated detection methods to identify any use of such tools. Violation of this policy will result in immediate termination of the process. Please rely solely on your own skills and the provided resources. Contact us for any clarifications.