Final Project

COMP 579, Winter 2022, McGill University
Final class date: April 12, 2022
Project can be turned in until April 26, 2022, 11:59pm EDT

Logistics

- The Project can be submitted without penalties until April 26, after which no submission will be accepted. The leaderboard will will close at the same time.
- This project is to be performed in groups of 2-4 students. Please get in touch with Doina if you wish to do it on your own. A group TA will help you through the process.
- Each group will choose *one of the proposed environments* to perform the project
- Post any query to the designated discussion board on myCourses or see the TAs during the OHs. The TAs will be monitoring the discussion board daily and will not be replying to Final Project specific emails.

1 Objective

The objective of this project is to provide the opportunity to experiment with ideas from the course as well as explore the state of the art in RL algorithms. To keep the process fun, interactive and competitive, we are providing a starter code kit and a leaderboard that will allow the agents developed to compete with each other.

1.1 Task 1: Hopper (Mujoco)

The task is to make a hopper with 3 joints and 4 body parts hop forward as fast as possible without falling on the floor. It has an 11-dimensional continuous state vector specifying position, derivatives of position, joint angles etc. and a 3 dimensional continuous action vector that controls the actuators of the three joints.

1.2 Task 2: Jelly Bean World

Jelly Bean world is a Gridworld environment, with objects such as Apples, Bananas, Jelly Bean and truffle. Each object is associated with different reward and fragrance. The agent has a partial view of the world that consists of the agent's position, the objects, their scent etc. The actions are to move up, down left and right. Full details about the state space, action space, rewards, etc are in the git project template. To make the task more approachable, we will provide a linear feature encoding that can be used as a starting point to train agents.

Find out more about the environments in the `README` of the starter code kit.

2 Getting started

2.1 Group formation

There will be Group TA who will facilitate the group formation process and will assign each group an ID. We will use this group ID for team level project submission for the leader-board.

2.2 Starter code

Please find the starter code in `https://github.com/COMP579TA/COMP579-Project-Template`
The `README` contains all the necessary information on getting started. The repository comes with a clone of the environment that we will be using on the server side to run student submissions. We recommend you use this virtual environment to train your agents in order to avoid any dependency issue. Please note that if a team's submission crashes on the server side, it will disqualify the team from the next published leaderboard. You will need to write your agent by modifying the designated agent class and methods inside the `agent.py`. You can add methods and classes to this script, but do not change the signature of the placeholder methods provided since it would lead to failure. The `train_agent.py` script is provided so that you can check the server side compatibility of your code. If your agent fails to run with this, it will not run on the backend evaluation framework either.

3 Leaderboard

3.1 Submission repository

Create a **private** github repo with the same name as the project group ID (eg. `GROUP_ID`). Add `COMP579TA` as a collaborator to the repo. This repo is similar to the submission folder `GROUP_ID` found in the template package. It should contain: `env_info.txt`, `agent.py`, weights of the trained agent and any other necessary component for the agent. Communicate the `http` URL of your submission repo to the group TA. The group TA will later announce the procedure and guide you though it.

3.2 Submission procedure

Push the latest version of your agent code and weights (as mentioned in the previous section) every time you wish to make a leaderboard submission. Make sure your agent runs on the provided `train_agent.py` script before submitting. Our back-end will automatically pull the submitted versions before refreshing the leaderboard. Make sure the submission holds an updated copy of `env_info.txt` based on your task choice. If you are solving the Mujoco task, the contents should be `mujoco`, as specified in the template. For the jelly-bean world task, the content should be `jellybean`. In the absence of this file rest of the evaluation code will not run. You will find the details in the `train_agent.py` code shared with the template.

3.3 Leader-board update frequency

The leaderboard will be available in this public repository:
`https://github.com/COMP579TA/COMP579-Leaderboard`.
We will be running the leaderboard backend 24x7 on our server, with a provision for scheduled server maintenance and any other unforeseen issues. But that does not imply an instantaneous leaderboard score. Since evaluating and retraining the agent is compute heavy and every leaderboard refresh requires to evaluate the submission of all the teams, there would be a necessary wait time. Since the wait time mostly depends on the efficiency of the code submitted, we encourage you to write efficient code.

4 Evaluation

For leaderboard evaluations, the agent will be evaluated on 2 categories; performance and sample efficiency on the same environment it was originally trained on. For the final evaluation, the agent will also be evaluated on its ability to adapt to a new environment. The latter is a stretch goal, not part of the evaluation, but we may award "special prizes" for it.

4.1 Performance

This score captures the final performance of the trained RL agent submitted by the students. This is not only a function of the algorithm, but might also depend on how long the agent has been trained, learning scheduler etc. We load the student's agent with the trained weights submitted with it. We evaluate it on the same task/environment on a set of 5 seeds for 50 episodes and the final performance is the mean over this.

4.2 Sample Efficiency

This score tries to capture how fast an agent learns starting from randomly initialized weights (i.e. the sample efficiency). We randomly initialize the weights of the agent submitted by the students and train it for 100K steps on the same environment, while evaluating every 1000 (Mujoco) / 10000 (JellyBean) steps. We repeat this process over 5 seeds and the final sample efficiency score is the Area Under the Curve (AUC) of the mean evaluation performance.

4.3 Adaptability

This score is to evaluate a trained agent's capacity to adapt to a new held out task. This new task is a modified version of the environment originally provided. This evaluation will only be performed during the final evaluation stage, but not during leaderboard submissions for the rest of the submission window. The trained agent will be re-trained on this new task for a fixed number of steps and sample efficiency and final performance will be measured as mentioned before. We will follow the same evaluation procedure as above, but for a slightly different task. To test if your agent has this ability, you need to modify the environment as you wish and aim to train an agent that is adaptable to the new environment. In the final project report you'll mention how you have modified the environment and how your agent performs on this new environment.

5 Evaluation

For the final evaluation, you are required to submit three components:

1. **Leaderboard submission**: Please submit the final version of the agent code, network weights and any other necessary components to the github submission repository. This is same as any other leaderboard submission.
2. **Report**: Please submit a 4 page project report (pdf, written in LaTeX) on myCourses. This covers the methods, motivation behind choosing them and the final results demonstrating performance, sample efficiency, and adaptability if you have evaluated it (the last part is optional).
3. **Video**: Please submit a 2 min video on myCourses, explaining the major aspects of your work.

The grade will be based on the quality of your agents (both the code and the performance) - 50%, and the report and video (motivation, apropriateness of methods and experimentation, presentation) - 50%.