CS 7642: Reinforcement Learning and Decision Making **Project #2**

Lunar Lander

1 Problem

1.1 Description

For this project, you will be implementing and training an agent to successfully land the "Lunar Lander" (LunarLander-v2) in OpenAI gym. You are free to use and extend any type of RL agent.

1.2 Lunar Lander Environment

In this problem you will be working in an 8-dimensional state space, with six *continuous* state variables and two discrete ones. The action space is discrete. The four discrete actions available are: do nothing, fire the left orientation engine, fire the main engine, fire the right orientation engine. The landing pad is always at coordinates (0,0). Coordinates consist of the first two numbers in the state vector. The base reward for moving from the top of the screen to the landing pad depends on a number of factors. If the lander moves away from the landing pad it is penalized the amount of reward that would be gained by moving towards the pad. An episode finishes if the lander crashes or comes to rest, receiving an additional -100 or +100 points, respectively, on top of the base reward. Each leg-ground contact is worth +10 points. Firing the main engine incurs a -0.3 point penalty and firing the orientation engines incur a -0.03 point penalty, for each occurrence. Landing outside of the landing pad is possible. Fuel is infinite, so, an agent could learn to fly and land on its first attempt. The problem is considered solved when achieving a score of 200 points or higher on average over 100 consecutive runs. You are encouraged to look at the Lunar Lander source code for more information on the environment and reward structure.

1.3 State Representation

As noted earlier, there are four actions available to your agent: do nothing, fire the left orientation engine, fire the main engine, fire the right orientation engine. Additionally, at each time step, the state is provided to the agent as an 8-tuple:

$$(x, y, \dot{x}, \dot{y}, \theta, \dot{\theta}, leg_L, leg_R)$$

The state variables x and y are the horizontal and vertical position, $\dot{x}$ and $\dot{y}$ are the horizontal and vertical speed, and θ and $\dot{\theta}$ are the angle and angular speed of the lander. Finally, leg_L and leg_R are binary values to indicate whether the left leg or right leg of the lunar lander is touching the ground.

1.4 Procedure

This problem is more sophisticated than anything you have seen so far in this class. Make sure you reserve enough time to consider what an appropriate solution might involve and, of course, enough time to build it.

- Create an agent capable of solving the Lunar Lander problem found in OpenAI gym
  - Upload/maintain your code in your private repo at `https://github.gatech.edu/gt-omscs-rldm`
  - Use any RL agent discussed in the class as your inspiration and basis for your program
- Create graphs demonstrating
  - The reward for each training episode while training your agent
  - The reward per episode for 100 consecutive episodes using your trained agent
  - The effect of at least 3 hyper-parameters of your choosing on your agent
    - You will select the ranges to be evaluated.

    - ∗ Evaluate your results in the context of the algorithm and the problem. Why do your results make sense?
  - Anything else you may think appropriate

- We've created a private Georgia Tech GitHub repository for your code. Push your code to the personal repository found here: https://github.gatech.edu/gt-omscs-rldm
- The quality of the code is not graded. You don't have to spend countless hours adding comments, etc. But, it will be examined by the TAs.
- Make sure to include a `README.md` file for your repository
  - Include thorough and detailed instructions on how to run your source code in the `README.md`
  - If you work in a notebook, like Jupyter, include an export of your code in a `.py` file along with your notebook
  - The `README.md` file should be placed in the project_2 folder in your repository.
- You will be penalized by **25 points** if you:
  - Do not have any code or do not submit your full code to the GitHub repository
  - Do not include the git hash for your last commit in your paper
- Write a paper describing your agent and the experiments you ran
  - Include the hash for your last commit to the GitHub repository in the header on the first page of your paper.
  - Make sure your graphs are legible and you cite sources properly. While it is not required, we recommend you use a conference paper format. For example https://www.ieee.org/conferences/publishing/templates.html
  - 5 pages maximum – really, you will lose points for longer papers.
  - Explain your experiments.
  - Graph: Reward at each training episode while training your agent and discussion of results
  - Graph: Reward per episode for 100 consecutive episodes using you trained agent and discussion of the results.
  - Graph: Effect of hyperparameters and discussion of the results.
  - Explanation of pitfalls and problems you encountered.
  - Explanation of algorithms used: what worked best? what didn't work?
  - What would you try if you had more time?
  - Save this paper in PDF format.
  - Submit to Canvas!
- Using a Deep RL library instead of providing your own work will earn you a 0 grade on the project and you will be reported for violating the Honor Code.

1.5 Notes

- If you get a Box2D error when running `gym.make('LunarLander-v2')`, you will have to compile Box2D from source. Please follow these steps and try running Lunar Lander again https://github.com/openai/gym/issues/100:

```
pip uninstall box2d-py
git clone https://github.com/pybox2d/pybox2d
cd pybox2d/
python setup.py clean
python setup.py build
sudo python setup.py install
```

- Windows users may install Box2D with the prebuilt Python wheels provided at https://www.lfd.uci.edu/~gohlke/pythonlibs/#pybox2d
- Even if you don't build an agent that solves the problem you can still write a solid paper.

1.6 Resources

1.6.1 Lectures

- Lesson 8: Generalization

1.6.2 Readings

- Chapter 9 (On-Policy Prediction with Approximation) of Sutton and Barto 2020

1.7 Source Code

- https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py

1.7.1 Documentation

- http://gym.openai.com/docs/

1.8 Submission Details

The due date is indicated on the Canvas page for this assignment. Make sure you have set your timezone in Canvas to ensure the deadline is accurate.
Due Date: **Indicated as "Due" on Canvas**
Late Due Date [**20 point penalty per day**]: **Indicated as "Until" on Canvas**
The submission consists of:

- Your written report in PDF format (Make sure to include the git hash of your last commit)
- Your source code in your personal repository on Georgia Tech's private GitHub

To complete the assignment, submit your written report to Project 2 under your Assignments on Canvas: https://gatech.instructure.com

You may submit the assignment as many times as you wish up to the due date, but, we will only consider your last submission for grading purposes. Late submissions will receive a cumulative 20 point penalty per day. That is, any projects submitted after midnight AOE on the due date get a 20 point penalty. Any projects submitted after midnight AOE the following day get a 40 point penalty and so on. No project will receive a score less than a zero no matter what the penalty. Any projects more than 4 days late and any unsubmitted projects will receive a 0.

Note: Late is late. It does not matter if you are 1 second, 1 minute, or 1 hour late. If Canvas marks your assignment as late, you will be penalized. **Additionally, if you resubmit your project and your last submission is late, you will incur the penalty corresponding to the time of your last submission.**

Finally, if you have received an exception from the Dean of Students for a personal or medical emergency we will consider accepting your project up to 7 days after the initial due date with no penalty. Students requiring more time should consider withdrawing from the course (if possible) or taking an incomplete for this semester as we will not be able to grade their project.

1.9 Grading and Regrading

When your assignments, projects, and exams are graded, you will receive feedback explaining your errors (and your successes!) in some level of detail. This feedback is for your benefit, both on this assignment and for future assignments. It is considered a part of your learning goals to internalize this feedback. This is one of many learning goals for this course, such as: understanding game theory, random variables, and noise.

If you are convinced that your grade is in error in light of the feedback, you may request a regrade within a week of the grade and feedback being returned to you. A regrade request is only valid if it includes an explanation of where the grader made an error. **Send a private Piazza post to only Miguel Morales and Timothy Bail**. In the Summary add "[Request] Regrade Project 2". In the Details add sufficient explanation as to why you think the grader made a mistake. Be concrete and specific. We will not consider requests that do not follow these directions.

It is important to note that because we consider your ability to internalize feedback a learning goal, we also assess it. This ability is considered 10% of each assignment. We default to assigning you full credit. If you request a regrade and do not receive at least 5 points as a result of the request, you will lose those 10 points.

References

[SB20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2nd Ed. MIT press, 2020. URL: http://incompleteideas.net/book/the-book-2nd.html.