Division of Engineering Science
UNIVERSITY OF TORONTO

ROB311: Artificial Intelligence

Final Project: Reinforcement Learning with OpenAI Gym

Winter 2020

Overview

In this project, you will explore reinforcement learning (RL) by implementing a learning agent for the popular OpenAI Gym. The goals are to:

- aid in your understanding of modern RL algorithms; and
- provide an opportunity for (limited) open-ended exploration with a popular simulation tool.

As part of the project, you will build an agent to solve the Gym cart-pole task (v0). The project is worth a total of **20 points**. All submissions will be via Autolab; you may submit as many times as you wish until the deadline (but only your last submission will be tested *offline*, see below). The project is partially open-ended (subject to limits on the libraries available); you will need to review material that goes beyond that discussed in the lectures. The (hard) due date for project submission is **Monday, April 20, 2020, by 11:59 p.m. EDT**.

A template file, `rl_agent.py`, is provided, which has a skeleton for you to fill in to complete your agent. A complete test harness (`test_cart_pole.py`) is also available—this is the same test program we will run to verify the performance of your agent.

Setting Up OpenAI Gym for CartPole-v0

OpenAI Gym can be easily installed using the Python `pip3` tool, as follows:

```
$ pip3 install gym
```

Many tutorials are available online that explain how to set up and use different Gym environments. Your task is to implement an RL agent that solves the CartPole-v0 task in Gym, which involves the balancing of an inverted pendulum. There are only two possible actions in this environment: push left and push right.

Implementing Your Agent

You should add your agent code to the `rl_agent.py` file, inside the `CartPoleAgent` class. You may also implement additional support classes and functions in this file, if you choose to. The comments inside the `rl_agent.py` file indicate which methods need to be completed.

There is some flexibility in terms of implementation—you are encouraged to review the AIMA text as well as the very popular (and free!) Sutton & Barto book for possible RL strategies (e.g., policy gradient). The limitations are as follows:

- you *must* build an RL agent—your code should make use of the reward signal to improve the agent's performance (we have not provided a specific discount factor and you will need to carry out some experimentation); and

- you may only use the Python libraries that are imported at the top of the `rl_agent.py` file (`random`, `math`, and `numpy`).[1]

You will submit the completed file `rl_agent.py` through Autolab. We will test your code by evaluating success in the Gym environment.

Grading

We will only score the learning algorithm implemented inside the file `rl_agent.py`, as defined by an instance of the Python class `CartPoleAgent`. Three parts of the project will be graded, as follows:

- Validation Test – **2 points**

  You *must* submit your agent for grading through Autolab. Once submitted, Autolab will verify that the agent operates correctly—two short episodes will be run using OpenAI Gym and the various methods in the `CartPoleAgent` class will be called.

  Two full points will be awarded when your agent passes these tests. Note that there is no scoring of the agent's performance in this case (the number of episodes is not sufficient to learn a good policy). The purpose of these tests is to ensure that an instance of your agent can be created and that it works properly (i.e., uses Python 3.7, does not load external libraries, returns valid actions, etc.). There *is* a time limit, however—the upper bound is 30 seconds (to ensure that your code does not immediately enter an infinite loop).

- Performance Test – **15 points**

  After the submission deadline, the performance of each agent will be evaluated *offline* (i.e., not in Autolab, since this is a final project) on a dedicated server machine. Scores now matter—your agent should attempt to learn a good policy in as few trials (a.k.a. episodes) as possible. The test code will be the same as in the file `test_cart_pole.py` (except with a different scoring threshold, see next paragraph).

  To grade your agent's performance, we will use the same basic guidelines that OpenAI uses to define success on the CartPole-v0 task: your agent must achieve a cumulative reward above a certain threshold on each of the last 100 consecutive episodes of the task. The thresholds and points allocated are:

  - *Average Reward:* 120, *Episodes Required to Solve:* $\leq 1,500$: **5 points**
  - *Average Reward:* 140, *Episodes Required to Solve:* $\leq 1,500$: **6 points**
  - *Average Reward:* 160, *Episodes Required to Solve:* $\leq 1,500$: **7 points**
  - *Average Reward:* 180, *Episodes Required to Solve:* $\leq 1,500$: **9 points**
  - *Average Reward:* 195, *Episodes Required to Solve:* $\leq 1,500$: **11 points** (OpenAI success!)
  - *Average Reward:* 215, *Episodes Required to Solve:* $\leq 1,400$: **13 points**
  - *Average Reward:* 240, *Episodes Required to Solve:* $\leq 1,200$: **14 points**
  - *Average Reward:* 265, *Episodes Required to Solve:* $\leq 1,000$: **15 points**

  The grading scale is nonlinear; achieving the top level of performance will be challenging but rewarding! To mitigate the effects of randomness, we will run each agent three times and use the best average reward for grading purposes. There is maximum time limit on the server of 10 minutes (for 1,500 episodes).

[1] Without this restriction, there are simply too many existing implementations available online—you may use these as a reference but may not copy any code directly. We reserve the right to modify the Gym environment to prevent offline training/learning.

- Algorithm Documentation – **3 points**

  You should include a short (10-12 lines, 80 characters per line) description of the algorithm that you implemented at the top of the `rl_agent.py` file (at the position indicated). The description should be sufficient to understand your code at a high level; if you used any references (papers or books, besides AIMA) please include these. You may use additional lines for references. We will read this description and review your source code and comments to verify your RL implementation.

Total: **20 points**

As part of the performance test, grading criteria will include: correctness and succinctness of the implementation, proper overall program operation, and code commenting. Please note that we will test your code *and it must run successfully*. Code that is not properly commented or that looks like 'spaghetti' may result in an overall deduction of up to 10% (since this is a final project, good coding practices matter).

Good luck!