Learning to walk efficiently

Reinforcement learning summative assignment

Michaelmas term, 2022

Introduction

The assignment is to train a bipedal robot to learn to walk efficiently in a 2D physics simulation. It should be *sample efficient*, learning with the least amount of interaction with the environment as possible, and also it should learn to walk *quickly*—eventually running like Usain Bolt.

You are to write a short scientific report for the method, experimental results, and limitations in a provided LaTeX template that closely follows parts of the ICLR conference style guidelines. You are to also provide a video of the agent and a log file as later explained. These files must all be zipped together, replacing the username with your CIS username, like this. You may not submit additional source code files:

```
username.zip
   username-agent-paper.pdf
   username-agent-code.ipynb (or .py)
   username-agent-video,episode=210,score=85.mp4
   username-agent-log.txt
   username-agent-hardcore-log.txt
   username-agent-hardcore-video,episode=350,score=-92.mp4
```

To assist in this, the following template reports and starter code are provided to build on:

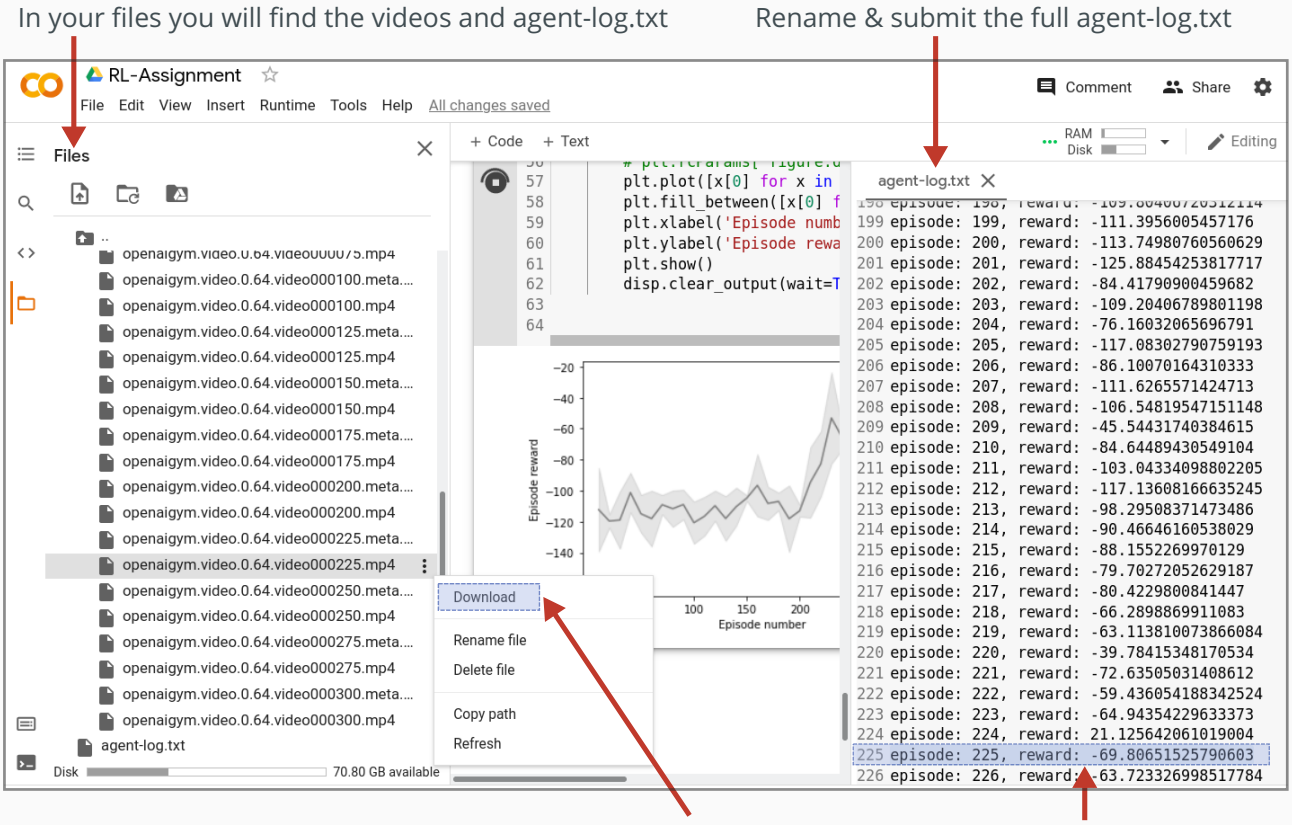
- [Reinforcement Learning Paper Template]
- [Google Colab Bipedal Walker Starter Code]

Learning to walk efficiently

The task is to use deep reinforcement learning to train a small bipedal robot to learn to walk efficiently. This means you will be marked on how quickly you can train the agent to learn to walk by taking the smallest amount of environment steps possible. The BipedalWalker-v3 environment for this is provided by OpenAI Gym, which uses Box2D for its physics simulation. Use the previously linked code to get started. After your agent consistently gets scores over 300 and you are happy with how quickly it converges when trained from scratch, you can see how the agent performs in the BipedalWalkerHardcore-v3 environment.

Submission

You must submit the full `username-agent-log.txt` and you must not change how its contents are formatted. The submitted video should show the highest score that you are able to successfully record. Repeat this also for any experiments you have conducted in the hardcore environment. Recording a video is quite slow, so you may wish to do it every 25 or more episodes. Increasing this interval may slightly improve training performance, but you may miss recording a good episode. The following image explains how and what to submit:



The code submission must be written with clarity and minimalism. You can submit an .ipynb or .py file. You do not need to strictly follow PEP-8 or any departmental guidelines for code quality with this assignment and may keep comments to a minimum.

Restrictions

The log files must be collected in exactly the same way as the template code as we will run an automated marking script on them for assessing the algorithm convergence and score. You must submit at least 1000 episodes of log data (unless your agent consistently reaches optimal scores in fewer than 1000 episodes). Please keep the max environment step count as 2000. The paper.pdf is restricted to a maximum of 4 pages (excluding references). Every page over this will incur a -10 mark penalty.

You can implement any RL algorithm that you like in your final solution. You do not necessarily have to use backpropagation, however you should use *deep* reinforcement learning (so the agent must be deep in the sense that it needs to have multiple layers).

Hardcore mode

After your agent consistently scores over 300 and you're happy with its learning efficiency, train the agent on the BipedalWalkerHardcore-v3 environment and submit a separate hardcore video.mp4 and log.txt showcasing how well your agent performs in this much more difficult setting. We will examine how well the agent navigates the terrain in this setting. This is a tough environment, so don't be disheartened if the agent is unable to perform well in it. You can write about such experiments in your report, but make sure to also present the convergence results of the basic environment.

Reinforcement learning marking scheme

The paper, code, videos, and log submissions will be marked as follows:

- **[50 marks]** Convergence and score
  - Does the agent consistently get high scores and learn the optimal policy?

  - How many training episodes are needed to get high scores?
  - Is the algorithm stable? Will it always converge or does it sometimes get stuck?
- **[30 marks]** Sophistication and mastery of the domain
  - What quality is the presentation of the underpinning theory?
  - Do the experimental results demonstrate scientific rigour?
  - How hackish is your implementation, or is it robust and well-designed?
  - Have you just cited and pasted code, or is there evidence of comprehension with further study and novel design extending beyond the lecture materials?
- **[20 marks]** Intuition of the videos
  - Is the agent walking fluently or with undesirable behaviour?
  - How fast is the agent running?
  - How effectively is it able to navigate the terrain?

Guidance

Before starting, here is some guidance for this environment:

1. I encourage starting with a modern method like TD3 (covered in lecture 8). The environment has a continuous action space (not suitable for DQN) and most of the classic continuous action space algorithms that work out-of-the-box on Pendulum-v0 (like SAC, PPO and DDPG) will need additional tricks and a lot of patience/tuning before showing any signs of convergence in BipedalWalker-v3.
2. Even a good agent for this environment will take 1-2 hours training before it starts to show signs of learning anything sensible, and perhaps a few more hours before it starts to get non-negative scores and walk forwards. So leave Colab/NCC running and take a long break when it's training.
3. It's common to implement a non-robust algorithm that sometimes works and other times doesn't due to poor initialisation and exploration (where the agent can get stuck in a minima, such as the robot always doing the splits and being unable to recover). If you've been training for several hours and your last few videos all have the same undesirable agent behaviour, this is an indicator that more exploration is needed. Try resetting Colab and retraining the same agent for another hour and compare the logs/videos and see if it gets stuck doing the same undesirable behaviour again.
4. Unless you are prepared to spend time fiddling with vectorised environments, only train on the CPU (in Colab, go Runtime $>$ Change Runtime Type $>$ None)—unlike the DL model where you will want to use a GPU. This will give you more Colab time and you will likely be I/O bound by the BipedalWalker-v3 environment simulation where further transfer to GPU is often slower. In marking, we strongly favour environment sample efficiency (few steps) above parallelism and GPU occupancy.
5. I would not suggest trying to design and build a new agent for this completely from scratch (without reference code) unless you are reimplementing a paper that provides extensive technical details. You are allowed to re-implement and cite existing implementations that you have found on GitHub, and even re-use existing hyperparameters that people have found for this specific environment. But, by doing so, you must cite all author code you use and detail your own individual experiments in attempting to further improve their work in your report.

Closing Comment

I hope that you enjoy this coursework and find watching your robot learn to walk rewarding! If you are struggling, please ask questions where we can discuss any issues, such as with programming or relevant theory.