Reinforcement Learning 2020: Assignment 3
Function Approximation

Aske Plaat
rl@liacs.leidenuniv.nl

March 12, 2020

1 Introduction

The objective of this assignment is to build intuition on deep reinforcement learning and to experiment with learning stability. You will implement a deep neural network player and experiment with various DQN and DQN-related algorithms.

The Deep Q-Network, or DQN, was introduced in 2013. It has since been used in many applications and has spurred a rainbow of further enhancements. It has propelled the field of deep reinforcement learning significantly, with the learning of playing Atari games from raw pixel data being the breakthrough application.

The field of deep reinforcement learning is exciting, active, complex, and thankfully has great tools and resources to make it easy for you to start, because of the efforts of many researchers, especially at OpenAI. For this assignment, you may find the following resources useful:

- Chapter 6 of Learning to Play
- OpenAI's Gym website is here. It provides a wealth of reinforcement learning examples. Browse to example Environments, and through algorithms.
- Definitely go to OpenAIs Spinning Up, and the code repo, which is here.
- The Stable Baselines are a great repository of RL algorithms. Visit them at Stable Baselines. Note: Stable-Baselines supports Tensorflow versions from 1.8.0 to 1.14.0, and does not work on Tensorflow versions 2.0.0 and above. Support for Tensorflow 2 API is planned.
- Many blog posts on Gym exist. Such as here and here. A more advanced intro, using Keras, is here. Google will find many more intro's for you if you need them.
- A nice blog about the Rainbow paper is here.

- Baseline implementations of DQN, PPO, and many other algorithms are here. But first go to Spinning Up.
- A blog post on Mountain Car is here, and a Jupyter notebook is here.
- A blog post on Breakout is here. Another is here.
- Andrej Karpathy's blog is very insightful here.
- How to access the Snellius Lab computers remotely Lab 302. This link is only accessible from inside the LIACS O&O network, use wlan3.

When you are stuck, your first resource should be the book for this course. If you find that confusing or it does not answer your question, please write an email to `rl@liacs.leidenuniv.nl`. Consult the resources of the assignment, ask questions at the classes, or the question hour.

2 Mountain Car - 3 points

We will start with the Mountain Car example, to help you get up to speed with Gym. Download, install and implement a deep neural network in Keras to learn how to solve the Mountain Car problem. Use Keras. Decide on where you want to do your training, on a CPU or on a GPU machine, or on Colab. The PCs in one of the Practicum rooms of Snellius have fast GPUs. At night, when no one is using them, you can use them for neural network training. Make sure you install the version of TensorFlow with GPU support enabled. Read how to access then here.

Many resources on the web exist on the Mountain Car problem. Note that fraud is absolutely forbidden at our University, and verbatim copying without referencing the source is fraud. You may get kicked out of this course for fraud. However, if you use code from the internet and refer correctly to the source in your report, then it is ok to use someone else's work. It will not get you points, though. You will be graded on your original work only. A report with zero original work but correct references to the code that is used will get zero points (but you will not get kicked out of the course—correct attribution to others are not fraud).

This section is a balance between hard and easy. It is hard because finding your way in Gym and Keras, getting them to work, may take some effort, even with all the available online resources. It is easy because so many good blogs exist to help you along the way. So enjoy the easy ride to your points, but do the work, and **do not cheat**.

3 Breakout - 6 points

Learn a player for the Breakout Atari game. Go to here for a description on how to find ALE and Breakout. Use Keras for the neural network.

The original Atari paper used DQN and a network of 3 convolutional layers and 1 fully connected layer.

This task is challenging. However, there is a very accessible blog post describing how to replicate this result. Study it, and use it as your starting point. Again, you can only score points for original work, so let us see how we can build on the blog.

Gettings the network to learn well, and getting a stable training process is an important issue in DQN (why? Read about the Deadly Triad). The replay buffer is one of the mechanisms by which stable learning can be improved.

How can you see if your learning is stable? If it is not stable, look at your DQN implementation, consider changing to another algorithm (see the OpenAI baselines) and consider tweaking the replay buffer.

If your learning happens to be stable, tweak the replay buffer anyway, and see if the learning is influenced, and think of why.

In your report, describe the architecture of your system and the steps you took to assemble the elements. Describe the architecture of the neural network you used, and which deep reinforcement learning algorithm you used. Explain how they work, and which changes you made.

4 Report

Your submission should consist of a report in pdf of at most 10 pages with a textual report on your findings, explaining the outcomes of the experiments, and the implementation process. Your code should be uploaded to blackboard and ready to run and free of syntax or other errors. Your report should contain:

1. General approach, issues encountered
2. For Mountain Car implementation: Source Code, ready to interpret and run, on blackboard. Short user documentation in report
3. For all sections: Supporting scripts for running, testing, and performance evaluation on blackboard. Short user documentation in report
4. For Breakout implementation: Source Code, ready to interpret and run, on blackboard. Short user documentation in report.
5. Report on replay buffer and DQN variants tried and the effect on learning stability.
6. One overall conclusion, leassons learned, observations.

Note that the successes in the field of DQN are much more recent than in heuristic planning and adaptive sampling. Results are more in flux, and the technology is less mature and more bleeding edge. This has the advantage of more excitement, and the disadvantage of more rough edges, algorithms and hyperparameters that are more difficult to get right.

The grading for this assignment will reflect this situation: describing carefully what you do, and what the outcome was, and what you think might be going on will get you points. If you cannot get everything to work, that is less important. The Olympic creed is: "Trying is more important than winning." The DQN field is less mature than heuristic planning and adaptive sampling. The next assignment will be even more bleeding edge.

The report must be handed in on **10 April 2020 before 23:59**. For each full 24 hours late, a full point will be deducted.