Computer Science Tripos – Part II – Project Proposal

The Playstation Reinforcement Learning Environment

C. Purves, Jesus College

Originators: Petar Veličković and Cătălina Cangea

12 October 2018

Project Supervisor: Cătălina Cangea, Petar Veličković

Director of Studies: Prof C. Mascolo

Project Overseers: Prof A. Dawar & Prof S. Moore

Introduction

Reinforcement learning is the process by which an agent, given an environment in which to work, is able to 'learn' a policy that maximises the sum of its reward over the ordered set of actions that it takes. There are several real-world uses for the research in this field, but games have commonly been used as benchmarks to evaluate the effectiveness of certain algorithms. For several years now, a good source of test-cases has been the Atari-2600 console. Most notable among these uses is that of the February 2015 Nature paper [1] by a team of scientists from Google DeepMind in which existing RL algorithms were combined with deep neural networks, with the outcome being an agent that can demonstrate what they describe as 'better than human-level control' of classic Atari games. In recent years, research into RL algorithms has progressed to such a point that arcade-style games are no longer particularly challenging. Both the recent improvements made in RL (such as the Asynchronous Advantage Actor-Critic (A3C) algorithm in 2016 [2]) and the relative simplicity of Atari-2600 games provide a significant motivation to engineer and abstract newer and more complex environments in a similar fashion to that of the Atari-2600.

For this project, I propose that a suitable such environment is that of the Sony ® Playstation™ (PSX) console. There are a number of reasons for which this is a task of significant scale. Notably, the PSX is substantially more powerful than the Atari-2600 and capable of running more advanced games than those that have been used for RL purposes in the past. In practice, this means that most PSX games have far richer state representations than any games that run on the Atari-2600.

The game that I will be using with these algorithms is known as Kula World developed by Game Design Sweden AB and released in 1998 for the PSX. It primarily requires that the player navigate a 3D grid-like world with the aim of collecting coins, keys and fruit. Within the game, coins and fruit are worth 'points' to the player and some number of keys are required to proceed to further levels. This amounts to a game that has a clear reward structure, a simple action space and a clear imposed time limit. These factors make Kula World an ideal choice for an RL application.

Figure 1: The first level of Kula World

In order to improve the accessibility of the Atari-2600 as a training environment for RL, OpenAI released the OpenAI Gym in 2016. This provides a relatively simple interface between a selection of Atari-2600 games and Python, allowing developers to interface (at a relatively high-level) with the gameplay. It would be of substantial utility to those in the RL research community to interface the full Gym API with my work; doing so would allow existing baseline algorithms as well as existing work using the Gym to be easily used in the PSX environment. The project involves several key stages of work. Firstly, I must work on adapting some existing emulator such that it can properly interface with the OpenAI Gym API. This involves making modifications that output the frame buffer's contents; locating specific memory contents related to metrics such as score and position; and exposing methods through which controls can be affected within the game.

Secondly, I will look to extend the interface to fully support the OpenAI Gym API. This involves packaging the interfacing script in a suitable way, as required by the OpenAI specification.

Thirdly, I will work to deploy some algorithms. The focus will be on the ones included in the OpenAI Baselines, of a similar nature to *Mnih et al.* [1], which involve Deep Q-Networks for learning.

Substance and Structure of the Project

The project structure is as follows:

Core Objectives

The emulator modification stage of the project is perhaps the most pivotal to its success. I will be required to read through and understand a largely undocumented and already heavily modified emulator, with the intention of abstracting several key features of the emulator to a Python interface. These include:

- Exposing methods such that the Python interface is able to control the actions of the emulator.
- Studying the memory model of the PSX, and collecting data so that I can work out whereabouts score information is held about the game.
- Extracting frame buffer information from the emulator so that I can both fully conform with the OpenAI specification, as well as pre-process frame data for use with a convolutional neutral network later in the project.
- Investigate and implement IPC between the C code of the emulator and the Python. This should be done in a way that allows for multiple emulators to run simultaneously, though this is not required.
- Implement the OpenAI Gym API

During this, I will also build an encoding of the first two levels of Kula World in such a way that it can serve as a spatial input to the neutral network. Then, I can implement simple RL algorithms. I will implement a simple MLP neutral network with `TensorFlow` and then use OpenAI Baselines to obtain gradients for learning.

Extension Objectives

There are a number of extension objectives that I will try to complete should I finish the core objectives according to (or ahead of) schedule.

Firstly, I will attempt to implement more advanced learning approaches. Notably, I will use the frame buffer data that I extract from the emulator as input to a convolutional neural network rather than the hard-coded models discussed above. This will allow me to train the agent on more levels within the game which it would otherwise be difficult to hard-code maps for.

Secondly, I plan to use other sources of data within the game to act as percepts. For example, the PSX controller has DualShock support (a vibrating motor). This is used in Kula World to indicate to a player that, among other things, the floor below them is about to break. Also, sound effects are used to indicate the collection of coins and other rewards. Both of these features feasibly have the opportunity to improve the outcome of an agent that was trained using them.

Starting point

Primarily, the modification of the emulator relies on two part IB courses:

- Computer Design
- Programming in C

The Computer Design course is useful primarily in the capacity in which it discusses the mechanics of program execution. The Playstation console uses a MIPS R3051 CPU and so the interaction of the PSX registers with the MIPS Instruction Set Architecture (ISA)

and with the built-in RAM have been covered in sufficient capacity to allow me to both understand and make modifications to the execution procedures.

Of course, the modifications that I make will not be to a real PSX device, but rather to an emulator. The emulator that I am likely to use is known as "PCSX Reloaded" (PCSX-R). The emulation element of PCSX-R, while the executable is written using a variety of languages, is written overwhelmingly in C and shared across each platform's build. This allows me an ideal opportunity to put the Programming in C course to use. The course focussed on features of C such as pointer manipulation, data structures and memory allocation. Each of these will be very useful in aiding me with the required modification to the emulator.

Once I have sufficiently modified the emulator and successfully built a Python interface, I will begin building and running Reinforcement Learning algorithms that will interact with the game environment. This leverages two additional Tripos courses:

- Foundations of Data Science
- Artificial Intelligence

The Foundations of Data Science course made heavy use of Python; specifically the numpy library. This is a very useful library for manipulating matrices and working with datasets. In the course, Python was primarily used within the context of drawing conclusions from input data, but in this context I will be extending that use-case somewhat in employing it heavily for the execution of learning algorithms. When training the agent using Deep Q-Learning I will need to make use of Multi-Layer Perceptron Neural Networks (MLP Networks), a concept that was first introduced in the Artificial Intelligence course.

It is worth northing that RL background is not covered in any Tripos course. This will require me to spend additional time to familiarise myself with relevant algorithms and tools that I will be using for the project, these include:

- OpenAI Gym
- OpenAI Baselines
- Q-Learning
- Deep Q-Networks

Resources required

My project will require the use of my own personal computer: a MacBook Pro (2017) with sufficient capacity to run the emulator; allow me to write and run both Python and C code; and connect to the Computer Lab GPU machines that will be provided to me by my supervisors for agent training. Mac OS 10.14 (Mohave) is installed on my laptop, but I will run a virtual machine of Ubuntu 18.04 LTS so that I can compile and run applications that are intended to run within the Linux environment of the GPU machines. I have a MacBook Air with a similar configuration available in the case where my main computer becomes unusable. I will manage all of the elements of my project that require writing code using `git`. Specifically, I will make use of a private `git` repository both for backup and version control. I will keep backups of both the code that I write as well as

the dissertation in compressed and encrypted .tar.bz archives on my home server and on Google Drive; these will be uploaded daily and kept until the end of the project.

Success citeria

The success of the project should be judged according to the following criteria:

- By the end of the project I should have a working Python library that allows interfacing with the emulator.
- The library should allow still frames to be received, score/reward to be available upon each action and control over a virtual controller.
- Such a library should conform to the OpenAI Gym API I should be able to run at least one simple RL algorithm on the environment that I have built, with some amount of success.
- I should have successfully extended the RL algorithms to use the frame buffer as input

Timetable

Proposed start date is 18/10/2018.

1. **Michaelmas weeks 3–5 (18/10/2018–7/11/2018)** Investigate the emulator: locate use score in memory and work on fabricating controller input. Learn TensorFlow from online tutorials and locate relevant materials pertaining to reinforcement learning. Read subsequent papers to V. *Mnih et al.* [1] regarding current research in reinforcement learning in games.
2. **Michaelmas weeks 6–8 (8/11/2018–28/11/2018)** Consider methods of facilitating appropriate IPC between Python interface and the emulator.
3. **Michaelmas vacation** Implement the OpenAI Gym API using the IPC mechanism that I decide on, and the emulator. This should allow me to use a new environment (Kula World) within existing contexts that OpenAI Gym is used.
4. **Lent week 0 (10/1/2019–16/1/2019)** Prepare progress report.
5. **Lent weeks 1–2 (17/1/2018–30/1/2018)** Run OpenAI Baselines on the environment, and learn how to specifically use TensorFlow as it applies to my project.
6. **Lent weeks 3–4 (31/1/2018–13/2/2018)** Implement a reinforcement learning approach for the environment that uses a MLP neural network built with TensorFlow.
7. **Lent weeks 5-8 (14/2/2018–13/3/2018)** Compare several RL algorithms within the context of the PSX environment. Investigate extension objectives (as above).

8. **Easter vacation:** Prepare and write dissertation for submission in Easter term. Send copies of draft dissertation to supervisors and my DoS by the end of the vacation period.

9. **Easter weeks 0–2 (18/4/2019–8/5/2019):** Finish and submit dissertation.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis. *Human-level control through deep reinforcement learning.* Macmillan Publishers Limited, 2015.

[2] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, Koray Kavukcuoglu. *Asynchronous Methods for Deep Reinforcement Learning.* Cornell University Library, 2016.