Assignment 3: Reinforcement Learning on Atari games

Authors: Mike van Santvoort, Jaron Sanders, Luuc Vlieger

Due: Wednesday June 19th, 2022

1 The Assignment

In this assignment we will emulate you becoming a "new member" of the reinforcement learning community. Concretely, you will program several self-learning agents that can play Atari games. The tasks below seek to guide you through the process and will form the basis of your report.

Part I - Setup

The reinforcement learning community often shares ideas with one another online, and this also includes code. You will use some of the community's work to develop self-learning agents. Specifically, you will use the *Gym* package for Python available at `https://gym.openai.com/`. Documentation is provided at `https://gym.openai.com/docs/`. We consider the struggle of getting this public Python environment to work on your computer part of the experience, and will therefore not provide debugging assistance. Below in this document there is a FAQ section, so please also have a look at that.

Task 0 Set up a Python environment with the Gym package. You can use the following steps:

- **(a)** Download the latest version of Python/Anaconda, and create an environment for this project.
- **(b)** Install the `git` package in this environment.
- **(c)** Open the Anaconda Powershell Prompt, and type '`pip install gym[atari]`'.
- **(d)** On Windows the `atari-py` package does not work (see also `https://github.com/openai/gym/issues/884`). Uninstall it by typing '`pip uninstall atari-py`'.
- **(e)** On Windows now type '`pip install git+https://github.com/Kojoley/atari-py.git`'. This installs a new version of `atari-py` that does work on Windows.
- **(f)** Run the code in Appendix A. If you see breakout being played poorly, the setup was successful.

Part II - Designing reinforcement learning algorithms

Next, you will design at least three self-learning agents, and test them in the `toy text` environments of `gym`. These environments are ideal to build your first agents in since they have a small observation space and are thus easy for algorithms to learn in.

Task 1 Choose and describe one of the `toy text` environments from `https://www.gymlibrary.ml/environments/toy_text/`. What are the observations, actions, and rewards?

Task 2 Create **at least** three reinforcement learning agents that you will implement for your chosen environment. You have considerable freedom in design, but make sure...

- **(a)** ... one is based on TD-learning.
- **(b)** ... one is based on Q-learning.
- **(c)** ... one is based on SARSA-learning.

We also encourage you to create a new reinforcement learning agent of your own design and compare it to the others. **Creativity and originality will be rewarded!**
Hint 1: Besides the lecture slides, feel free to for example also examine pseudo-codes in 'Reinforcement Learning', by Richard Sutton and Andrew Barto available here: `http://incompleteideas.net/book/RLbook2020.pdf`.

Hint 2: Consider consulting scientific articles from the reinforcement community for inspiration. For example, here are the proceedings of ICML in 2020: `https://icml.cc/virtual/2020/papers.html?filter=titles` *and the recent proceedings of NeurIPS:* `https://proceedings.neurips.cc/`.

Task 3 Evaluate the efficacy of your agents numerically, as well as the transient behavior of your algorithms. Support your claims convincingly with a plethora of numerical plots. **The more scientifically interesting these plots are, the better!**
Hint 3: You could for example depict the average fitness/maximum reward versus the number of learning iterations.

Part III - An introduction to state space reduction

The main difficulty in creating a successful reinforcement learning agent for a generic Atari game, is that the state / action space is typically very large. To circumvent this issue you should first choose a reduced state space representation and then apply your reinforcement learning algorithms to that reduced state space instead.

Consider the game of Pong, displayed in Figure 1. This is a game where one plays against an adversary; and in the Atari version of Pong, the computer can play near perfectly. Each observation in Pong consists of a pair: an image of size 210×160 (with color values ranging between 0 and 255), together with an action (*pressing up*, *pressing down*, or *doing nothing*).

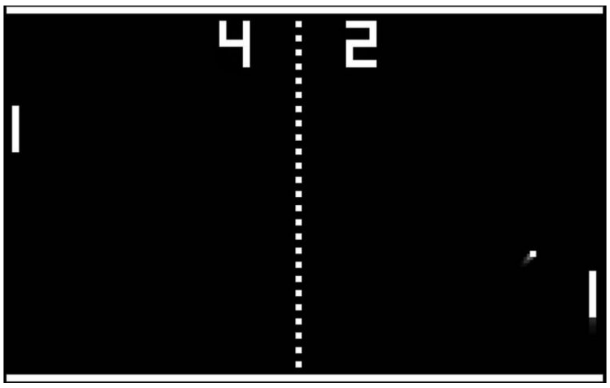

Figure 1: Pong, an Atari Game

Task 4a Consider the observations in Pong provided to you by the Gym environment. What is an upper bound to the number of possible observations?

Task 4b The ball can accelerate and decelerate in this game. In order for a reinforcement learning algorithm to be able to learn to play perfectly, is it enough for observations to correspond to single frames? If not, then how could the observations be redefined so that a reinforcement learning algorithms could learn about acceleration?

Task 4c Propose a reduction and/or approximation of the state space. Ponder on the following points:

- What are the most important pieces of information that can be seen in Figure 1?
- Are these pieces of information you just considered truly important to a reinforcement-learning agent or can some subinformation still be scrapped?
- How will you tackle the issue that from a stationary image / observation you cannot extract neither velocity nor acceleration?
- If your ideal reduced state space representation is continuous, could you approximate this further using discretization?

Task 4d What is an upper bound to the size of your reduced / approximate state space of Task 4c?

Part IV - Applying the algorithms to a generic Atari game

You are now prepared to try and adapt the agents of Part II to work for a generic Atari game. Again, remember: a big part of reinforcement-learning is analyzing the observation space and extracting the necessary features, which we have practiced now in the previous part. It is typically not even possible to store an entire naive state space representation in computer memory.

Task 5 Choose and describe an Atari game **other than Pong**. What are the observations, actions and rewards? Specifically, what part of each observation is essential to keep?
Hint 4: You should avoid a gym-environment that takes observations from RAM. This will make observations difficult to analyse.

Task 6 Describe how you would adapt your reinforcement learning agents of Task 2 for the chosen Atari game. Specifically, describe how you would extract the key features of the observation space.
Again, ***creativity and originality will be rewarded****, in this question and the next! You are even allowed to take extreme measures if you'd like (but you don't have to!), such as using clustering algorithms or neural networks to generate to lower-dimensional approximations of the underlying state space.*

Task 7 Evaluate the efficacy of your adapted agents by creating numerical plots. You can design and carry out experiments yourself.

2 Deliverable

Write a PDF report detailing your findings for the Tasks above, preferably in LaTeX.

Style of the report

Your report should adhere at least to the following standards:

(i) It is written in the style of a paper that positions the problem within science, states the objectives clearly, describes the systems and agent implementations in depth, conducts a thorough performance evaluation of the agents, and concludes with interesting discussion / scientific insights.

(ii) It thus implicitly addresses all tasks, except for possibly task 0.

(iii) All of your plots are always clear, labelled accurately, and well-captioned. All sample averages reported include also confidence intervals and are moreover the result of many independent runs.

(iv) There is exact copy of your Python code in the Appendix.

Knock-out criteria

Your report **must** meet the following requirements in order to be marked:

(i) An electronic copy of the report (PDF only) is submitted in Canvas, on time.

(ii) It contains a title, author names, student numbers, and a bibliography when citing.

(iii) It includes a paragraph in the Appendix, in which every student's specific contributions are explained. In particular, tell us which percentage of (a) programming and (b) report writing was done by each student. Each student is expected to contribute significantly to both.

(iv) There is a hard page limit of **seven A4 pages**, single column. This excludes code in the Appendix, the bibliography, and project contribution statement.

(v) Margins are at least 2cm, and the font size is at least 10pt. The overall presentation is clean, neat, and orderly. The text is in English, legible, and contains few spelling mistakes.

Grading

We encourage you to examine the rubric available on Canvas. We will be judging for effort and you may report difficulties or even failure. The reporting however must be detailed and genuine. Your work also has to be critically reported. Finally, we expect you to be creative in what you investigate and effective how you present your results. Merely doing the bare minimum of what we ask may not be sufficient.

3 Frequently Asked Questions – Setup

1. **The pip install for the Kojoley atari package in part I.(e) does not work, what should I do?**
   Try the following line instead '`pip install -f https://github.com/Kojoley/atari-py/releases atari_py`'. For further information, look at >>`https://github.com/Kojoley/atari-py.git` >> readme.md

2. **My commands do not seem to work in Anaconda Powershell Promt. Why is this?**
   Try running the Powershell Prompt as an administrator

3. **Where should I run the python code from Appendix A?**
   You may use any Python environment such as Jupyter Notebook, Spyder, Visual Studio, or Visual Code.

4. **When running the code for breakout I only see the text output but not breakout being played poorly, why is this?**
   This is because it only plays one game and this finishes so quickly that the window only appears for about a second. You can add a command that includes a little pause in each step of the for loop.

A Example Code

```python
import gym

env = gym.make("Breakout-v0")

print(env.action_space)
print(env.observation_space)

print("Press Enter to continue...")
input()

env.reset()

# Run the game
for t in range(10000):
    env.render()

    #Choose a random action
    action = env.action_space.sample()

    # Take the action, make an observation from environment and obtain reward
    observation, reward, done, info = env.step(action)

    print("At time ",t,", we obtained reward ",reward,", and observed:")
    print(observation)

    if done:
        print("Episode finished")
        break

env.close()
```