Reinforcement Learning – Practical Assignment

Anna Latour, a.l.d.latour@liacs.leidenuniv.nl

March 28, 2018

Assignment deadline: 23:59, May 14, 2018

This document contains instructions for the Practical Assignment of Leiden University's Reinforcement Learning Course of 2018. Please read these instructions carefully. If you have any questions relating to this assignment or the course, please don't hesitate to contact us: reinforcementlearning@liacs.leidenuniv.nl. We will stay for a bit longer after class each week, so if you prefer to ask your questions in person, we will be available each Wednesday at 17h15, in class room 407–409 (it would be helpful if you bring your laptop).

In this assignment you will work on algorithms for learning to solve problems in different environments in the OpenAI Gym [1]. In this document we will first describe the two environments you will encounter in this assignment. In Section 2 we give you a couple of hints on how to get started (installing the libraries you need etc). The third section of this document describes the exact assignment, after which we have a few notes on the evaluation of the assignment in Section 4. Any specifics about the report and handing in your assignment, can be found in Sections 5 and 6, respectively.

Note: *Part of this document contains specific information you need to complete the assignment. The rest is just there to help and guide you. If you don't want to read it all, just read Sections 3 to 6.*

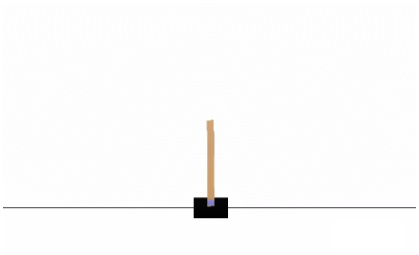
1 Environment Descriptions

In this assignment, you will develop, implement and evaluate an Reinforcement Learning algorithm that will be able to learn (near-)optimal policies for two of the environments in the OPEN AI GYM. We will give a basic description of these environments in this section.

Pole Balancing

This is a classic Reinforcement Learning example, which you can use for the warming-up part of the assignment. We are using the OPENAI GYM environment `CartPole-v1`[1].

We are given a cart that can move in one direction on a friction-less track. On the cart we are balancing a pole. When the pole starts falling over, we can balance it above the cart by applying a force of $+1$ (to the right) or -1 (to the left) on the cart. We receive a reward of $+1$ for each time step that the pole is balanced, and the cart has not moved too far from the center.



The `CartPole` environment.

The pole is no longer balanced if it makes an angle of more than 15° with the normal to the track. The cart is allowed to move at most 2.4 distance units from the center on either side.

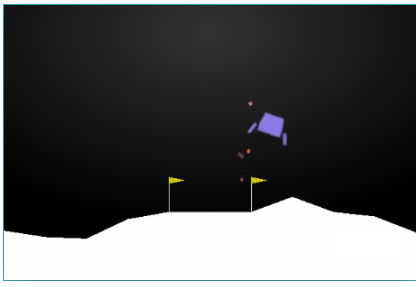
The `CartPole-v1` environment provides the following state vector:

$$\mathbf{s} = \left(x, \frac{\partial x}{\partial t}, \theta, \frac{\partial \theta}{\partial t}\right),$$

where $-2.4 \leq x \leq 2.4$ denotes the position of the cart on the track, and θ the angle that the pole makes with the normal to the track. This makes $\frac{\partial x}{\partial t}$ and $\frac{\partial \theta}{\partial t}$ the cart velocity and pole angular velocity, respectively. If the pole hasn't fallen over after 200 time steps, the environment is terminated. Hence, the maximum reward per episode is 200. This environment is considered solved if the average episode reward over the last one hundred episodes is 195 or more.

Lunar Lander

For the main assignment, we are going to use the OPENAI GYM environment `LunarLander-v2`[2]. In this environment the goal is to land a small spaceship on the surface of the Moon, between the two flags that mark the landing site.



The `LunarLander` environment.

The spaceship comes in from above and has to land at zero speed on the landing pad (which is centered at coordinates $(x, y) = (0, 0)$). This can be achieved by firing the engines on the spaceship (you get infinite fuel).

Firing the main engine yields a reward of -0.3 per time step; the other engines can be fired for free. Moving from the top of the screen to the landing pad yields a reward

[1] `gym.openai.com/envs/CartPole-v1/`
[2] `gym.openai.com/envs/LunarLander-v2/`

of 100–140. If you land the spaceship (either inside or
outside the landing pad), you receive a reward of 100. If you crash the spaceship anywhere, you receive a reward of -100. Additional bonuses apply for each leg that is on the ground. The environment is considered solved if you get a total reward of at least 200.

The state is represented by the following vector:

$$\mathbf{s} = \left(x, y, \frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \theta, \frac{20 \cdot \partial \theta}{\partial t}, \mathbb{1}_{leg1}, \mathbb{1}_{leg2} \right),$$

Here x and y represent the x and y coordinates of the spaceship, and $\frac{\partial x}{\partial t}$ and $\frac{\partial y}{\partial t}$ the respective velocities of the spaceship in those dimensions. The angle made by the spaceship is given by θ and $\frac{\partial \theta}{\partial t}$ is the angular velocity, which is multiplied by 20 in this environment. These first six elements of the state vector are all in the real domain. The final two elements are Booleans that indicate if the first and/or second leg makes contact with the ground.

The four possible discrete actions available in each time step are: do nothing, fire left oriented engine, fire main engine and fire right oriented engine.

2 Getting started

You can develop your agents and run your experiments on your own computer or on one of the student computers, but if you want to run them remotely on a university server, you could use the `duranium` environment of the Data Science Lab. See below for instructions on how to connect and set up your environment.

Connecting to the Data Science Lab

Make sure you have access to the `duranium` environment. If you are a student at LIACS, you have access by default. Otherwise, you can request access by following the instructions at `http://rel.liacs.nl/dslab/index`.

As soon as you have access to the `duranium` server, you should be able to connect to it like this (from a Linux OS):

```
$ ssh ULCNusername@ssh.liacs.nl
$ ssh duranium
```

You may want to follow these[3] instructions for generating `ssh` keys, and mounting your drive at the Data Science Lab on your own computer, following these[4] instructions.

Setting up your environment

We are going to create a virtual environment for this assignment. Go to your home directory on `duranium` and type:

```
$ mkdir envs
$ cd envs
$ virtualenv -p python3 rl
```

[3] `www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2`
[4] `askubuntu.com/questions/412477/mount-remote-directory-using-ssh`

This creates a directory called `envs` in your home directory, and then creates a virtual environment for `python3` called `rl` in the `envs` directory. You can activate the environment as follows:

```
$ source rl/bin/activate
```

Once you have activated your environment, you can start by installing some useful libraries (numpy[5], pandas[6] and gym[7]). You may have to upgrade `pip` first:

```
(rl)$ pip install --upgrade pip
(rl)$ pip install numpy
(rl)$ pip install pandas
(rl)$ pip install gym
(rl)$ pip install gym[box2d]
```

Now test if these installs were successful, by starting `python`:

```
(rl)$ python
```

and then importing the libraries and running an environment:

```
import numpy as np
import pandas as pd
import gym

env = gym.make('CartPole-v1')
env.reset()
for _ in range(5):
    env.step(env.action_space.sample())
end.close()
```

You should see an output that looks somewhat like this:

```
(array([-0.01615036, -0.17282784,  0.03190649,  0.29138941]), 1.0, False, {})
(array([-0.01960691,  0.02182499,  0.03773428,  0.0089377 ]), 1.0, False, {})
(array([-0.01917041,  0.21638603,  0.03791303, -0.27160487]), 1.0, False, {})
(array([-0.01484269,  0.02074416,  0.03248094,  0.0327908 ]), 1.0, False, {})
(array([-0.01442781,  0.21538563,  0.03313675, -0.24946969]), 1.0, False, {})
```

Each of these 4-tuples contains an array that represents the state, a float representing the reward, a Boolean that indicates whether or not the episode is finished, and finally a dictionary that may contain information useful for debugging. Make sure to check OpenAI Gym's docs [8] for more info.

You can close `python` again by hitting `Ctrl + D`.

When you are done with your experiments, you can close your environment by typing

```
(rl)$ deactivate
```

or simply by hitting `Ctrl + D`.

[5] www.numpy.org
[6] pandas.pydata.org
[7] gym.openai.com
[8] https://gym.openai.com/docs/

3 The Assignment

For this assignment you can choose to either work by yourself or in a team of two. You have to implement and evaluate Reinforcement Learning methods and report on their performance.

Downloading the materials

Start by going to this course's BlackBoard page, and locate this assignment. There you should be able to download a `.zip` file containing the skeleton code for this assignment, as well as this document itself.

Warming up (optional)

Note: *This warming-up exercise is just for you to get familiar with* `gym` *and to see some example code. Feel free to skip it; you do not have to hand in anything for this part of the assignment.*

Consider the discretised `CartPole` environment that is provided in the source code in file `cartpole_wrapper.py` (class `CartPoleWrapperDiscrete(Wrapper)`). Also consider the Q-learning agent that is provided in file `qlearner.py`.

Now run the experiment in `main.py` that applies this Q-learner on the discretised `CartPole` environment. Analyse the results. Think about how you can improve the learning speed.

Which parameters can you use to change the model (discretisation) of the `CartPole` environment? Which effects do you expect when you change those parameters?

Which parameters can you use to change the Q-learning algorithm (Hint: also take a look at the `base_learner.py` file)? How do you expect the algorithm's performance to change based if you change those parameters?

Try to optimise the modelling and algorithm parameters. Summarise your findings in a sensible way (table and/or figures) and comment on them. Which parameters are optimal and how do your results show that? Can you think of an explanation for the relevance of certain parameters compared to others?

The main assignment (mandatory)

Note: *This is the actual assignment you actually will have to do, so please carefully read the following.*

In the skeleton code you will find the files `lunarlander_wrapper.py` and `my_agent.py`. Your assignment is to complete the classes therein such that you will have a wrapper that models the `LunarLander-v2` environment, and an agent that learns how to solve it (remember: the `LunarLander-v2` environment is considered 'solved' as soon as you get a total reward of at least 200 in an episode). You will have to hand in these files and we will run your code, so do not rename the files or the classes therein.

The goal is for you to implement a Reinforcement Learning algorithm and wrapper of your choice. Consider the literature to find suitable methods for both and experiment, or come up with your own ideas. While you are free to use Q-learning and/or discretisation of continuous spaces, don't blindly copy the example we provided. Look in the literature for

variants or improvements and implement and evaluate those. We value originality on your part :)

Additionally, you will have to report on the performance of your model and learning agent (see Section 5 for some technical requirements for the report). We expect you to explain how you model the environment and why you do it like that, and what learning algorithm you implemented and why.

Use experiments to show the performance of your approach. For example: you can compare the performance of your approach to other approaches, and/or report on the sensitivity of your approach to differences in parameters settings.

In *Reinforcement Learning: An Introduction*, Sutton & Barto (either edition) [2], you will find many figures that compare different algorithms or parameter settings. Please look through them to get some inspiration for the experiments you could run and present.

4 Assignment Evaluation

Part of your grade will be determined by how quickly your learner converges to a policy that solves the `LunarLander-v2` environment and how stable it is with respect to slightly different starting conditions.

We will evaluate your code with a script similar to the one found in the file `main.py` in the skeleton code. At the end of that file, some of the code is commented out, such that you can run the example (`CartPole-v1`).

Please make sure that your adjustments to the classes in `my_agent.py` and `lunarlander_wrapper.py` are such that the code that is commented out, will run when you uncomment it. In other words, the following script should run without problems:

```
from lunarlander_wrapper import LunarLanderWrapper
from my_agent import MyAgent

wrapper = LunarLanderWrapper()
agent = MyAgent(wrapper=wrapper, seed=42)

for episode in range(10):
    rewards[episode] = agent.train()
    if wrapper.solved(rewards[episode]):
        break
wrapper.close()
```

While you are free to add additional files of code for the purposes of organisation, please make sure to hand them in and take care not to rename the files or classes provided to you.

Please write your code in `python 3.4`. Also make sure to code neatly and add sufficient and useful comments.

5 Report

Your report must be written in English and formatted using LaTeX, following Springer's LNCS formatting guidelines[9]. It should have a clear structure and be no longer than 15 pages. Please be inspired by this guideline:

Abstract A summary of your work (roughly ten lines). Mention the type of problem you solve, the main methods used, the relevance of your research and your most important findings.

Introduction and Related Work Give an overview of the problem you study and the method you use, and how they can be placed in the field of Reinforcement Learning.

Problem Description Formally define the problem.

Background Introduce any non-trivial terminology, notation, or theorems.

Approach Explain what (novel) techniques you use to solve the problem, and motivate. You can add pseudo code or diagrams to aid your explanation, but beware to not let them stand alone.

Experiments Define the questions you want to answer with your experiments. Explain why your experimental setup is suitable for answering those questions. Specify the hardware you used and relevant implementation and system details. Present your results in tables and/or figures when appropriate. Analyse and discuss your results.

Conclusions Mention and interpret your most important findings, and draw conclusions accordingly. Discuss possible future work.

6 Submission requirements

The **submission deadline** for handing in your work is **23:59, May 14, 2018**. Before you can hand in your work, you have to sign up your team on the BlackBoard page of this course. *The deadline for signing up your team is* **17:30, April 4, 2018**. When you are doing the assignment by yourself, please sign up as a team of one person.

On April 4, after everybody has signed up their team, we will give you access to the assignment hand-in environment on BlackBoard.

Please upload to BlackBoard the following files:

- A `.pdf` file containing your report. If you are working in a team, please name it `studentID1_studentID2.pdf` (where, of course, you substitute the placeholders by your actual student IDs). Simply name it `studentID.pdf` if you are working by yourself (substituting the placeholder by your own student ID).
- A `.zip` file containing all your code (in `python 3.4`). If you use any non-standard libraries, please include them as well. Again, name it `studentID1_studentID2.zip` if you are working in a team and simply `studentID.zip` if you did the assignment by yourself.

[9] `www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines`

- Because this is the first year we are teaching this course, we'd like to get some feedback on how much time you spent on the assignment. Please include a `.txt` file in which you briefly describe how much time each team member spent on the different tasks of the assignment (e.g. analysing skeleton code and `python` packages, researching and designing algorithm(s), implementation, parameter tuning, experiments, report writing, etc... By now you should know the naming conventions. Thanks!

Note that the BlackBoard assignment allows for multiple submissions. We will grade the last submission made before the deadline.

Important!

Note that we will be checking both your written report and your code for plagiarism. You are not allowed to copy work from fellow students or anyone else. If you are basing your algorithm on somebody else's work, be very clear about that and reference them properly. If we do suspect plagiarism or other forms of cheating, measures will be taken.

However: for your experiments you are allowed to test your method against the methods of your fellow students or others. Naturally, you should credit and reference them accordingly.

Good luck, and have fun!

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". In: (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : an introduction*. MIT Press, 1998, p. 322. ISBN: 9780262193986. URL: https://mitpress.mit.edu/books/reinforcement-learning.