Imperial College London

15 November 2021

Department of Computing
Reinforcement Learning – Prof Aldo Faisal & Dr Paul Bilokon
Code-base for this coursework – Filippo Valdettaro

Assessed coursework 2

Version 1.0

To be returned as online submission via CATE.

The Coursework should be submitted on CATe by November 29th, 7 pm, and consist in:
- A **PDF** of your written report, that has to be named ***coursework2_report.pdf*** and including a copy-paste of your code in the appendix (code appendix does not count towards page limit). The first page of the coursework has to contain your name, your CID, department, and course you are on (e.g. "MSc Advanced Computing").
- A **Python file** of the final code used to implement your model generate all your results, named ***coursework2.py***. You can download it in this format from Colab.
Please ensure that you are familiar with the CATe submission process well before the deadline (a guide on how to submit on CATe is available on Materials: `https://materials.doc.ic.ac.uk/manage/2122/70028`), as we are, unfortunately, not allowed to mark emailed or printed hardcopy submissions.

Report: Your report should not be longer than 7 single-sided A4 pages with at least 2-centimetre margins all around and 12pt font, preferably typeset, ideally in Latex[a]. Appendixes are not allowed. 7 pages is a **maximum** length, shorter Courseworks are fine, and Courseworks over the page limit will incur a penalty. Please **paste the completed and annotated source code at the end of your submission**, this code does not count toward the page limit.

Code: Your code is not going to be automarked but should be run-able by the markers. Please paste the completed and annotated source code at the end of your submission. You don't need to follow the same structure as the template code. You should produce your own code to compute your specific answers. You are welcome to reuse code you developed in the lab assignments and interactive computer labs.

You are encouraged to discuss the general Coursework questions with other students, but your answers should be yours, i.e., written by you, in your own words, showing your own understanding. Your report and code will be automatically verified for plagiarism. Written answers should be clear, complete and concise. Figures should be clearly readable, labelled, captioned and visible. Poorly produced answers, incomplete figures, irrelevant text not addressing the point and unclear text may lose points.

Marks are shown next to each question. Note, that the **marks are only indicative**. If you have questions about the Coursework please make use of the labs or EDStem, but note that the Teaching Assistants cannot provide you with answers that directly solve the Coursework.

[a] Using latex we recommend the "minted" package:
```
usepackage{minted}
begin{minted}{python}
<code>
end{minted}{python}
```

Overall setting

Our aim is to learn to control an inverted pendulum on a cart problem. Here the pole is attached to an un-actuated joint to a cart (it spins frictionless). The cart moves on a frictionless track. The system is controlled by applying a force that pushes the cart left or right. In a discrete setting we are looking at applying a force +1.0 or -1.0 to the cart. At initialisation the pendulum starts almost upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright, so it is best to balance it as long as you can. Therefore, the episode ends when the pole is more than 12 degrees from the upward vertical, or when the cart moves more than 2.4 units from the center starting point (finite track). We are using the code based on the OpenAI gym environment CartPole-v1 that may have been modified to fit the purpose of this coursework. Please refer to the Lab Assignment 3 on how to setup the OpenAI gym and use Pytorch, as usual the GTAs will be happy to support you in the interactive labs.

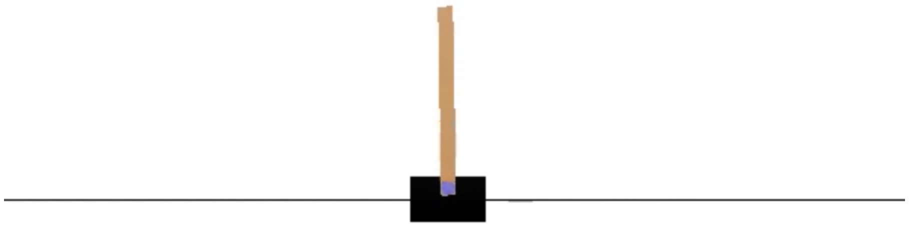

Figure 1: Illustration of the OpenAI Gym CartPole environment, focus of this Coursework.

Python code provided

This Coursework is accompanied by a DQN CartPole Jupyter Notebook, also available on Colab[1] via this link here `https://tinyurl.com/RL-IC-DQN-CW`. This template code defines a basic `ReplayBuffer` class that can in its present form only contain one transition, and a simple `DQN` class, that implements a non-optimised DQN architecture. You can take and edit any part of the template code we provide, together with any other code we provided. To solve the questions of this coursework, you may have to modify these classes and other to make them functional.The final code you submit should be run-able without efforts on marker side. The code will not will not be automarked, thus you do not need to keep strictly to the the original structure of the template code. However, the markers may want to run your solution to verify your outputs and automatically compare your code to other versions.

Question 1: Implementing a functional DQN

This Question is worth 40% of the total coursework mark[2].

1. For this task you will have to implement a DQN solution that learns to balance the cart pole. You can base your solution on the code template provided with this coursework, as well as code e.g. from previous lab assignments. The implementation should include three key technical elements that DQN also used in Atari games, namely the 1. replay buffer, 2. target networks and 3. the input should contain multiple frames k of the state (i.e. $s(t), s(t-1), \ldots, s(t-k)$).

'skipping frames'

[1] We recommend using the free Colab platform as it is more than fast enough to support this coursework.

[2] The coursework is designed so, that even if you cannot implement all or most of the DQN features of this question, you can still tackle part of the subsequent questions using the code template provided.

The state information of the environment must come from the 4 dimensional observations of the environment and **not** the pixels that would be drawn in a visualisation of the environment. Please submit the code as part of your solution, make it easy for the markers and document the code.

2. Describe briefly how you implemented each of these three features and refer to your code base (for ease of marking; you may use line numbers referring to your code appendix)

3. Discuss briefly your design decisions (such as architecture and parameters of the Deep Network) as well as choices of hyper parameters for the learning task. Show a graphic of the deep network architecture including key parameters (e.g. neurons per layer, etc). Hint: Deep neural networks are not just about deep structures, width may also matter.

   sigma, replay buffer size, k

   # of neurons in each (hidden) layer

4. Show the learning curve for your DQN model (x-axis: episode, y-axis total return at end of episode). The learning curve should show the mean ± standard deviation of the total return. Make sure to choose an appropriate range for the plot (justify briefly your choice) and the number of repetitions [3]. Briefly state the average total return and episode number at which your DQN achieves roughly 90% of the final performance value during learning. The curves may not be very smooth, but you can read this graphically off the learning curve "by eye" (i.e. interpolating "by hand" to keep things simple - draw it into the plot if it helps).

Question 2: Hyperparameters of the DQN

This Question is worth 30% of the total coursework 2 mark.

sigma-greedy

1. Investigate through computer experiments how the exploration parameter ϵ affects the performance (and stability) of learning. Briefly discuss whether a constant ϵ suffices or a variable ϵ (which changes during learning) is necessary by referring to the following figure that you will have to generate: Show the mean ± standard deviation learning curves for different settings of constant and/or variable epsilon. Show in a second plot below the first one the schedule for ϵ (both pots have to be aligned so that equal x-axis locations on the page denote equal epoch numbers): i.e. plot the different constant and variable curves for epsilon that you explored (so x-axis is epoch number and y-axis is the value of ϵ). Colour coordinate the learning curve plots with their corresponding ϵ schedule plots.

2. Investigate through computer experiments how the variability in reward during learning depends on the size of the replay buffer. Hold all parameters constant but vary the size of the replay buffer by doubling and halving it a few times. Generate a plot where you plot on the x-axis the replay buffer size (consider using a logarithmic scale for this axis) and on the y-axis the standard deviation of the learning curve (taken from an appropriate level of the maximum achieved total return, e.g. 67%). Briefly describe and discuss your finding.

3. Investigate through computer experiments how the parameter $k = 1, 2, \ldots$ which sets how many frames are presented to the input, relates to overall learning and final performance. For this hold all parameters constant but varying the size of k within a relevant range (briefly justify your choice) and plot the learning curves for different values of k superimposed. Briefly describe and discuss your finding.

[3] We do not recommend to go overboard with the number of repetitions (it takes around 30 seconds to see learning of this DQN on Colab on one run) but you should have reasonable estimates of the variability by using 10 or more repetitions.

Question 3: Ablation/Augmentation experiments

This Question is worth 30% of the total coursework 2 mark. In the following we perform modification experiments (two ablations and one augmentation) to understand which features of the DQN are key for learning speed and best end-of-training performance:

remove

- Ablate only the target network feature,
- Ablate only the replay buffer.
- Enhance the target network feature by making the model a double DQN (DDQN). Hint: Frame depth k for DDQN may have to be different.

1. Briefly discuss your DDQN implementation.
2. Plot the learning curves for these 3 modifications against the original model (from Question 1) in a single plot and discuss your findings.

Change Log
Version 1.0 Initial version