Solving the Lunar Lander Continuous (v2)

Yuval Mor, Roey Fuchs

Submitted as final project report for Reinforcement Learning course (896873), Bar-ilan University, 2021

This report describes our code, attempts and results of the project. The full code and instructions to run it can be found on GitHub.

1 Introduction

The LunarLander is a popular envirment [2] for reinforcemnet learning, and has a great importance as a primary, free, and open source envirment to train on a complicated challenge [1].

The LunarLander evnirment has 2 versions – the discrete and the continuous. In both of the versions the lander starts at the top of the screen and should land between 2 flags and be with zero speed, as show in figure 1.

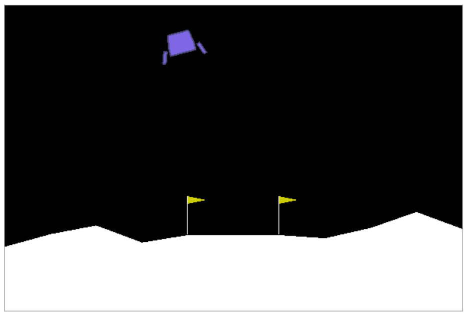

Figure 1: Moment after the envirment starts. The lander fall from the top middle of the screen, and should land between the flags.

As descirbed in the envirment page, the episode finishes if the lander crashes or comes to rest, receiving additional -100 or 100 points. Each leg ground contact is 10 points, and firing main engine is -0.3 points each frame.

Each step (frame), the lander sends its state with the following information:

$$State = \begin{cases} x \text{ coordinate of the lander} \\ y \text{ coordinate of the lander} \\ v_x \text{ the horizontal velocity} \\ v_y \text{ the vertical velocity} \\ \theta \text{ the orientation in space} \\ v_\theta \text{ the angular velocity} \\ \text{Left leg touching the ground} \\ \text{Right leg touching the ground} \end{cases}$$

Each value is continuous, except the legs values that them boolean. The x and y coordinate values are relation to the land site, are located in $(0, 0)$.

The action space in different between the discrete version and the continuous one. In the discrete version, the action is a value from $\{1, 2, 3, 4\}$ such that: $1, 3$ are fire left or right engines in full power, 2 is firing the main engine in full power, and 4 is do nothing. In the continuous version, each action is a pair of values – the first is for the main engine, and the second is for the sides engine. The main engine only runs when reciving a value in range $[0, 1]$. the sides engine runs left when recving a value in range $[-1, -0.5]$ and runs right for a value in $[0.5, 1]$.

To solve the environment, we need to receive average of 200 points (rewards) for 100 episodes.

1.1 Related Works

As we mentioned before, Lunar lander is a popular environment, so over the internet many attempts to solve it can be found. Most of them were of the discrete version due to its simpleness and lower computing resources, specially in tabel methods like SARSA.

Since the discrete environment and the continuous one are similar, but don't identical, we tried to get from related works specific improvements, especially those related to reduce the amount of states.

An intersting example for that, can be found in comparsion [3] between SARSA with bins optimizations and without. Bins optimization stands for reduce the amount of states – a bin of values will be interpredted as a same value. Example in figure 2. Another usage of bins optimizations is in greedy-epsilon method, such that every bin has there epsilon for epsilon–greedy method, so the decay is usually more robesti.

Another important technique we use is experience replay [4], that reached the front of the stage with DeepMind Atari project [5]. An improvemnt to experience replay, we also tried to use prioritized experience replay [6], that use prioritized select instead of random select in regular experience replay. We will discuss on this later.

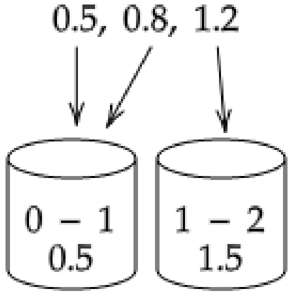

Figure 2: Example of bins optimize. Here we have 2 bins. values in range of 0 to 1 will convert to 0.5, and values from 1 to 2 will convert to 1.5. 0.5 and 0.8 enter the first bin, and 1.2 to the second.

2 Solution

2.1 General approach

Describe your preferred approach to solve the problem. what alternatives you plan to try and why.

2.2 Design

Provide some general information about your code, platform, how long it took you to train it, technical challenges you had, etc.

3 Experimental results

Provide information about your experimental settings. What alternatives did you measure? Make sure this part is clear to understand, provide as much details as possible. Provide results with tables and figures.

4 Discussion

Provide some final words and summarize what you have found from running the experiments you described above. Provide some high level insights.

Note - your project will be evaluated for aspects, including the technique you selected, the rational of the experiments you decided to run, the insights you learned from this process and more. Remember, for the purpose of this course, the process that you demonstrate is very important.

5 Code

Please provide a link to your code repository. It can be either a Private Github repository either a colab notebook with additional links for the data.

Good luck!!

References

[1] T. Brady and S. Paschall. The challenge of safe lunar landing. In *2010 IEEE Aerospace Conference*, pages 1–14, 2010.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[3] Soham Gadgil, Yunfeng Xin, and Chengzhe Xu. Solving the lunar lander problem under uncertainty using reinforcement learning. *arXiv preprint arXiv:2011.11850*, 2020.

[4] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.