Artificial Intelligence Techniques (IN4010)

Report for RL Assignment 2

Hao Liu
Delft University of Technology
H.Liu-7@student.tudelft.nl

Jiahao Cui
Delft University of Technology
J.Cui-2@student.tudelft.nl

Zehang Wu
Delft University of Technology
Z.Wu-7@student.tudelft.nl

ABSTRACT

1 QUESTION 1

To get familiar with the agent, we ran the *deep_q_learning_main.py* for 3 times. With the visible results, we found that our agent is not willing to land on the surface after about 400 episodes' attempts. As a result, our lunar lander eventually learns not to land on the moon, this is not even close to our goal - land safety between two flags. The figure 1 confirms our conclusion.

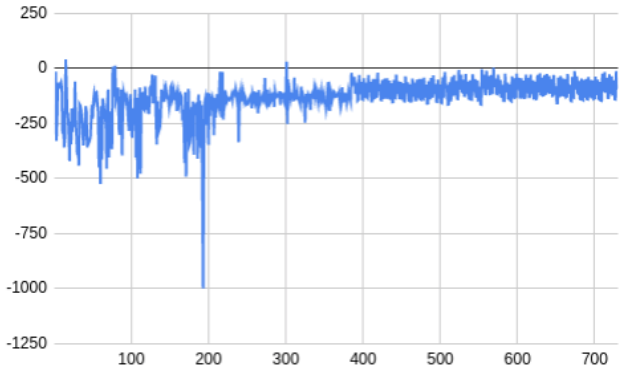

Figure 1: The cumulative rewards of all stages of each episode. *the cumulative rewards seldom goes above zero which also proves that the agent behavior is not an optimal one*

2 QUESTION 2

2.1 Instability and Causes

When a non-linear function approximator, for example a neural network, is used to represent the *action-value (Q) function* in reinforcement learning, it becomes unstable. Direct result can be seen that the learning procedure of agent in 1 is rather long and even worse, it decides to stay in in the air, not landing on the ground. Possible reasons for this unexpected behaviour are:

- the sequence of observations of the agent are correlated
- small update of the *Q function* influence the data distribution since it may significantly change the policy
- the correlation of the action-values (Q) and the target values

2.2 Ideas to address the instability

Two key ideas to address the instability mentioned in 2.1 are:

- **Experience Replay**
  The biologically inspired mechanism *Experience Replay* is used to randomize the data, which will remove the correlation between sequence of observations and smooth over changes in data distribution.
- **Iterative Update**
  The idea of *Iterative Update* is to adjust the action-value (Q) towards the target value periodically, every episode in this case. This will reduce the correlation between action-values (Q) and the target values.

2.3 Observation after modification

After complete the class *ReplayMemory* in the file *deep_q_learning_skeleton.py*, the agent is able to make use of the *Experience Replay*. It stores the experience and randomly sample a batch of experience to update the network. The improvement is obvious, rerun the *deep_q_learning_main.py* we find that the agent starts to land between the flags after 300 episodes and will not refuse to land at the end. The closing result is that the agent is able to land near the flags if not between the flags. The figure 2 shows the difference(improvement) we made comparing to the figure 1.

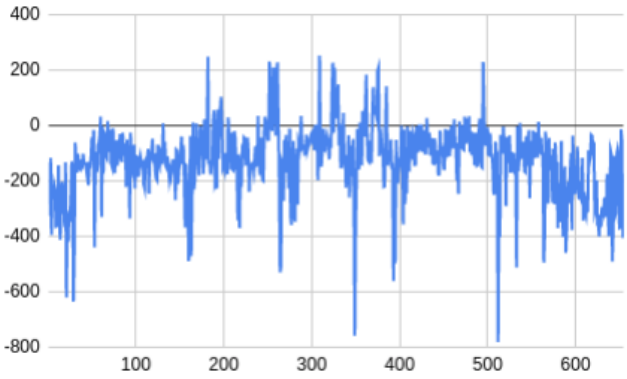

Figure 2: The cumulative rewards of all stages of each episode. *the cumulative rewards goes above zero much more time then our original agent, although it is not complete learnt how to safety landing between two flags.*

3 QUESTION 3

3.1 IDEA behind fixed target

In this question, an additional network named "target network" will be used, which will be used to fix a parameter θ^- in every

IN4010, 2019-2020, Reinforcement Learning

episode. Next we use the estimate future value ($\gamma max\hat{Q}(s', a', \theta^-)$) from the target network to do the calculation and backward update. The purpose of "fixed" target is mainly because periodically update the target value can reduce the correlations with the target. More details, there is an issue that an update that increasing $Q(s_t, a_t)$ often also increases $Q(s_{t+1}, a)$ for all actions a and hence increases the target y_j which could lead to oscillations or divergence of the policy. Generating the targets using an older set of parameters adds a delay which makes divergence or oscillations much more unlikely.

3.2 Modified Parts

In this question, several steps need to finish:

- 1 : Add a target network which has parameters named "θ^-" and initialize the θ^- by copying θ
- 2 : Update the θ^- by copying the θ every episodes
- 3 : Calculate the target value using the target network
- 4 : Insert the target value into the "old" network to update. ($\gamma max\hat{Q}(s', a', \theta^-) - Q(s, a, \theta)$)

3.3 Observations after modification

The result is that after 500 episodes, our agent can not learn a every "nice" approximation function to support itself landing safely every time.

The details has been shown in the figure 3 which is about the cumulative rewards of each episodes. In this figure, it is easily to see that safely landing occurs 5 times in the first 200 episodes. And after that, it gets the positive cumulative rewards value many times between episode 420 to 450.

Comparing the result here with the result in the question 2, there is no big difference between them. Both them can not steadily land on the lunar which means both them don't learn a "nice" approximation function. As for the proportion of positive cumulative rewards in 500 episodes, they are almost same. So we says that there is no big improvement from the answer of question2 to the answer of question3. Theoretically analysis, there should be an improvement. Because we change the target value to be "fixed" target value. To some extent, it will benefits neural network. But thinking more deeply, there are so many possibilities from randomly choosing batch. It could be that more positive rewards are chosen can lead to a better behavior.

3.4 Reason

There could be some reasons why our agent can not safely land steadily:

- Exploration is not enough. As the increasing of episodes, the exploration rate(ϵ) is decreasing. So after 200 episodes, the exploration rate is only around 0.01, which makes our agent greedy and it is hard to find another safely landing cases by exploration any more.
- As for the experience pool, we use it to replay the experience. But one hash question is that there are lots of bad data in the experience. The "nice" data is rare in the experience pool. In this case, "batch updates" could find "bad" data every time which is not beneficial for learning a "nice" approximation function. In this case, I would suggest to try **prioritized experience replay**

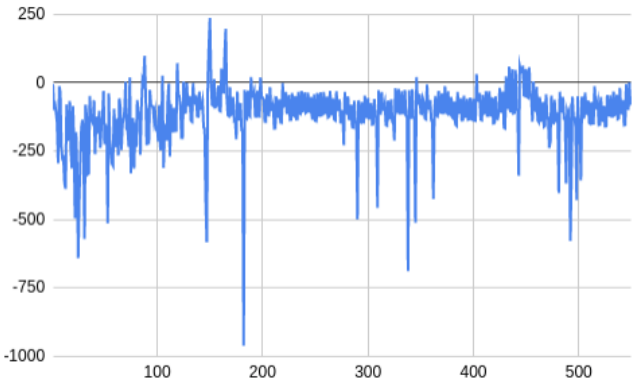

Figure 3: The cumulative rewards of all stages of each episode.