CS 8803-O03 Reinforcement Learning: Project 2

Liangliang Yang (lyang338)

Youtube link: https://www.youtube.com/watch?v=N2TJxbcSGSI&feature=youtu.be

I. Project Overview

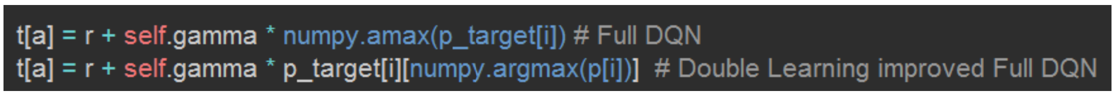
The purpose of this project report is to build a learning agent to successfully land the "Lunar Lander" that is implemented in OpenAI gym. We will use the LunarLander-v2 gym environment, where the agent need to navigate a space vehicle from the starting point to the landing pad on the surface of moon without crashing. Recall the problem in last homework (HW4), which is a reinforcement learning problem with discrete state and action spaces, we can solve it with simple Q-learning method. However, the task in the project is a problem with a continuous state space (so our state space is infinite). We can't use any table to store infinite number of values. In order to solve this problem, here we will use a recent version of Q-learning called Deep Q Network learning (DQN), which can approximate the Q function with a neural network.

II. Experiments and results

In the implementation, first I have used a full DQN based agent with target network. Then to improve my model, I applied the Double Learning technique to my full DQN agent, which can help my agent perform more stable and faster.

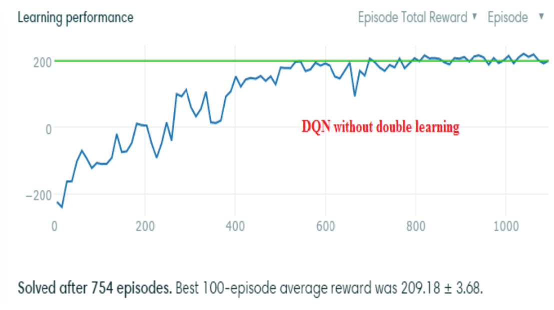
(a) Full DQN vs Double Learning improved Full DQN

Since we have already created two different networks (DQN network and target network), the implementation of double learning full DQN versus full DQN becomes very easy, we just need to change one line of code as below.

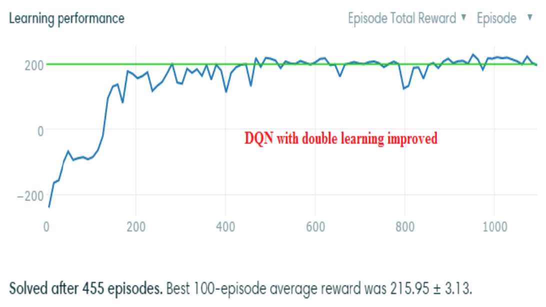


```
t[a] = r + self.gamma * numpy.amax(p_target[i]) # Full DQN
t[a] = r + self.gamma * p_target[i][numpy.argmax(p[i])]  # Double Learning improved Full DQN
```

The results from are as below and also see the links for DQN and improved DQN. As we can see from the plots, with double learning technique improved, our agent gets better (higher average reward: 215.95 vs 209.18) and faster(solved after 455 episodes compared to 754 episodes).



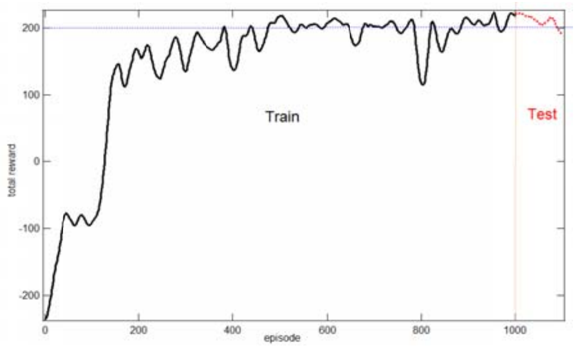
Solved after 754 episodes. Best 100-episode average reward was 209.18 ± 3.68.



Solved after 455 episodes. Best 100-episode average reward was 215.95 ± 3.13.

(b) Train and test result for double learning improved full DQN

From part (a), we can see that the double learning technique improved our agent. For all following experiments, we will explore our agent based on the improved DQN model. Firstly, we would like to see the training and test set result. In order to do so, we will train our agent with enough episodes, then I will set the epsilon to be zero and run the 100 test trails with none exploration. The result is shown below. As we can see from the result, the performs well (total reward >= 200) in the test region, which means our training is successful.



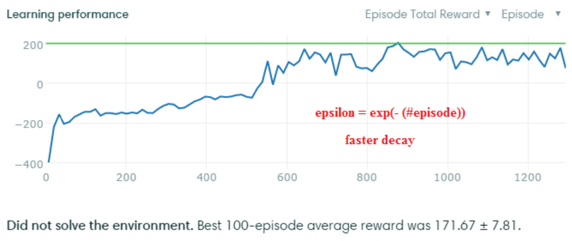
(c) Effect of hyper-parameters

In the section, we will explore the effect of those hyper-parameters. The below table is a typical hyper-parameters setup during a double learning DQN. In this work, I mainly focused on the effect of exploration rate (epsilon), the learning rate and the gamma.

Parameter	**Value**
memory size	100000
gamma	0.99
batch size	40
epsilon	$0.975^{(\#episode)}$
Adam learning rate	0.001

(i) Effect of exploration rate (epsilon)

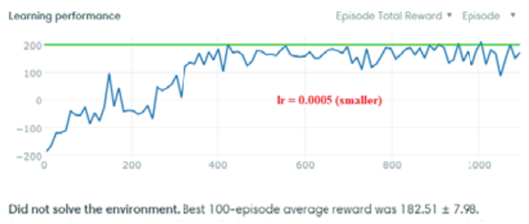
First I will explore the effect of epsilon. Instead of using $0.975^{(\#episode)}$, which is a slowly decay exploration rate, I used a much faster decay rate $\exp^{(- \#episode)}$. The result plot is as below and also click to see the OpenAI link. As we can see, with faster decay rate of epsilon, the agent doesn't perform very well, since there is not enough exploration during the training of our agent.



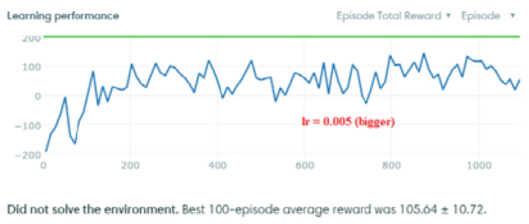
Did not solve the environment. Best 100-episode average reward was 171.67 ± 7.81.

(ii) Effect of learning rate (Adam)

Besides the exploration, the learning rate is also very important in Q-learning. In the main experiment, I use 0.001 for the learning rate. Here I will change the rate to be 0.0005(smaller) and 0.005(bigger) to see the effect. Below plots are the results and also you can click to see the OpenAI experiments for lr=0.0005 and lr=0.005.



Did not solve the environment. Best 100-episode average reward was 182.51 ± 7.98.

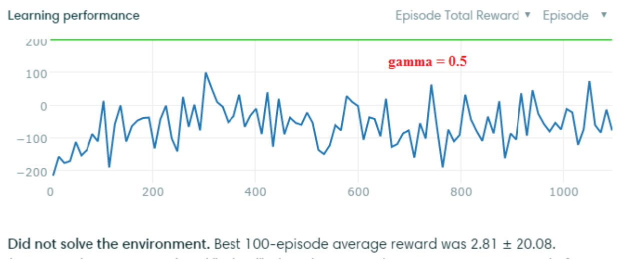


Did not solve the environment. Best 100-episode average reward was 105.64 ± 10.72.

As we can see from the plots, when learning rate is smaller, the agent learns slowly and it will need more training (maybe 2000 or longer episodes) to solve the problem. On the other hand, if the learning rate is too bigger, then the agent might just learn even worse. This is somehow similar to our human beings, that we need to schedule our study plan well (not too slow, not too fast), to get the best performance.

(iii) Effect of gamma (discount factor)

Since the discount factor is used to trade off the importance of sooner versus later rewards, it is also interesting to see the effect of it. Here instead of using 0.99 for gamma, I have tried gamma=0.5. See the OpenAI and below is the reward plot.



Did not solve the environment. Best 100-episode average reward was 2.81 ± 20.08.

As we can see from the plot, the agent learns quite bad when the discount factor is small. Hence we need to be careful about the discount factor as well in our experiment.

III. Summary

We have successfully build a DQN (with double learning technique implemented) to land the "Lunar Lander". Based on the DQN agent, we have also explored the effect of some hyper-parameters (including alpha, gamma, epsilon), and we have seen that all these parameters are very important for a successful agent. Overall I think it is a very interesting project and I have learned a lot about Q-learning and Deep Q Network learning.

Main reference:

Let's make a DQN, *Jaromír Janisch*