Reinforcement Learning algorithms in Acrobot environment - Deep Machine Learning

Pongsakorn Chanchaipol, Leelawadee Sirikul

Chalmers University of Technology
MPALG

CHALMERS UNIVERSITY OF TECHNOLOGY

Introduction

In this project, we experimented with several classic Reinforcement Learning algorithms in the Acrobot environment from OpenAI's gym to see the performance and behaviour of each algorithm in the environment.

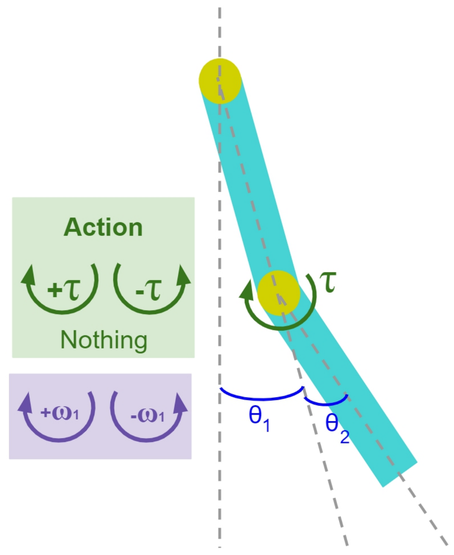

Figure 1: Picture of Acrobot environment

Acrobot environment summary:

- The Acrobot environment consists of a two-links robot, where each link can rotate around its joint freely
- The goal is to swing up a two-link robot to a given height (in this case 1).
- The state of this environment is defined by $(\cos\theta_1, \sin\theta_1, \cos\theta_2, \sin\theta_2, \omega_1, \omega_2)$ [1]
  - θ_1 and θ_2 are the current angles of each link
  - ω_1 and ω_2 are the angular velocities of each link
- The actions that can be taken in this environment are applying +1, 0, or -1 torques to the joint in between the two links of the acrobot.

Method/proposed solution:

- To calculate the height
  - $height = -\cos\theta_1 - (\cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2)$
- Q-learning
  - $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_a Q(s',a) - Q(s,a))$

    where r is the reward get from moving from state s to state s', γ is the discount factor , α is the learning rate ($0 < \alpha \leq 1$)
  - Modified reward is to add a normalized magnitude of the angular velocity (ω_1) into the reward.

    $$reward_{modified} = reward + constant \times \frac{|\omega_1|}{(4\pi)}$$
- Double Deep Q-learning.
  - $Q(s,a) = r + \gamma Q(s', \arg\max_a Q(s',a;\theta); \theta^-)$

    where θ for selecting the best action. θ^- for evaluating the best action.
- Dueling Deep Q-learning.[2]
  - $Q(s,a) = v(s) + A(s,a) - \frac{\sum_{a'} A(s,a')}{N_{action}}$

    where v is the state value function and A is the the state-dependent action advantage function.

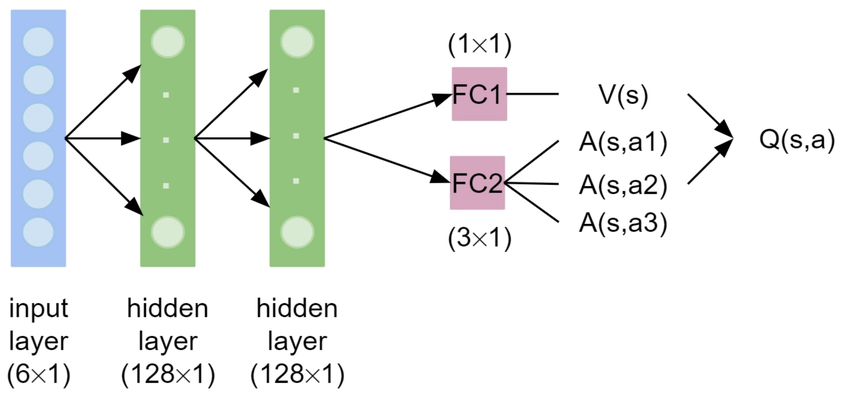

Figure 2: Dueling Deep Q-learning Network

References

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba.
Openai gym, 2016.

[2] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver.
Rainbow: Combining improvements in deep reinforcement learning, 2018.

Results

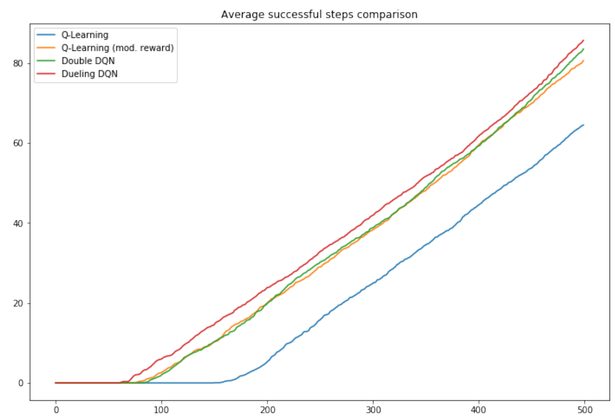

Figure 3: Plot of average successful steps of each algorithms

The plot on the left shows the average successful steps of each algorithms, which can be summarize to:

- Q-learning perform the worst here
- Reward modification improve Q-learning significantly but still worse than Double DQN and Dueling DQN
- Dueling DQN performance is slightly better than Double DQN

Model	Average successful steps(percentage)	Average first successful steps
Random policy	0 (0 %)	-
Q-learning	64.5 (12.90%)	178.3
Q-learning (modified reward)	80.6 (16.12%)	101.95
Double DQN	83.5 (16.70%)	106.55
Dueling DQN	85.65 (17.13%)	75.2

Table 1: Average successful steps comparison

Policy plots:

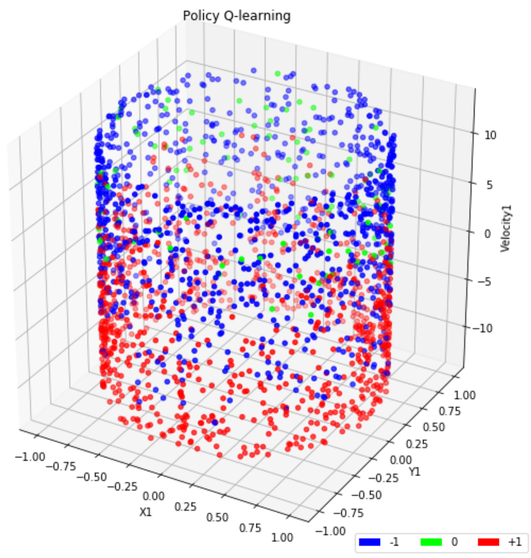

Figure 4: Q-learning policy

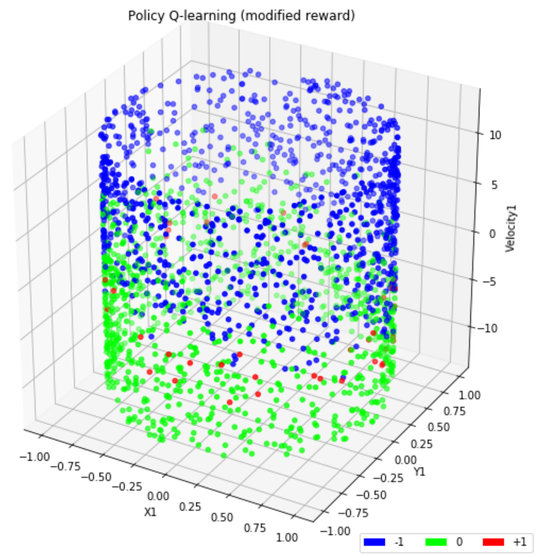

Figure 5: Q-learning policy (modified reward)

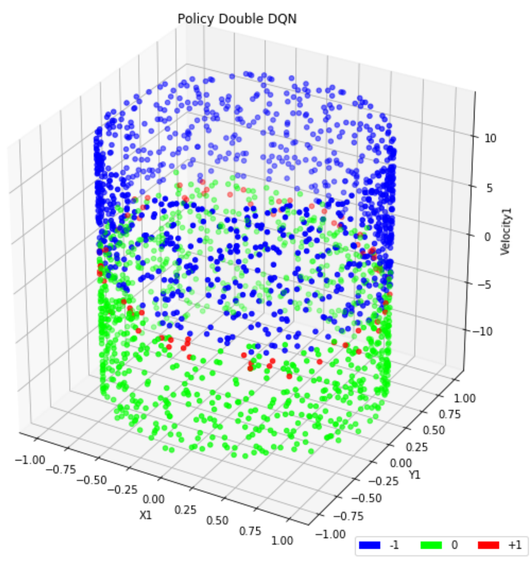

Figure 6: Double DQN policy

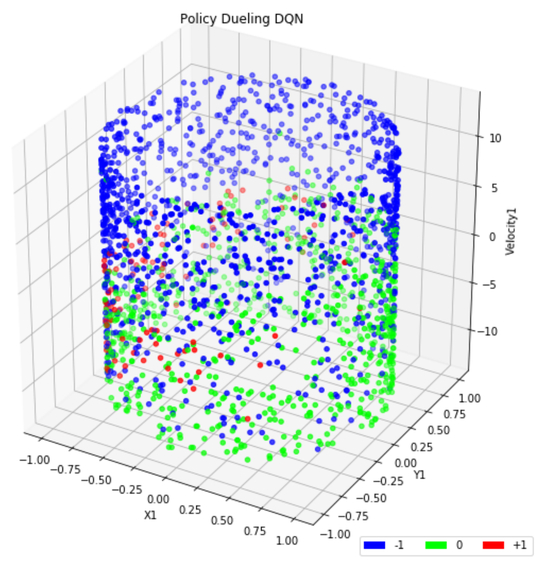

Figure 7: Dueling DQN policy

In these policy plots, actions [-1,0,+1] are represented in [Blue, Green, Red], repectively. The top-left plot, Q-learning polcy, is the only one with balance between action +1 and -1, while the other algorithms mostly choose action -1 and 0.

Conclusion

Performance ranking of the algorithms in Acrobot environment :

- 1) Dueling DQN (best performance)
- 2) Double DQN (close performance to Dueling DQN)
- 3) Q-Learning with modified reward
  - take angular velocity into account
  - encourage model to increase velocity
- 4) Q-Learning (significantly worse than the others)

Direction for Future Work:

- Try other RL methods: Noisy DQN, Prioritized DDQN, Rainbow, etc.
- Evaluate further on other environments

sirikul@student.chalmers.se poncha@student.chalmers.se