**Reinforcement Learning Assignment 4**

Zirong Chen

# **BipedalWalker**

In the implementation of PPO for BipedalWalker environment, I chose to use epoch/episode-wise control to conduct the training process. So, in the all the following figures from PPO on the BipedalWalker environment, the x-axis stands for the number of epochs/episodes spent.

Several experiments on hyper-parameter with controlled variables are done, here I only present the best one in terms of testing performances.

|  |  |
| --- | --- |
|  | PPO |
| Average Returns  (Red lines are averaged values per 100 units) | Chart, histogram  Description automatically generated |
| To reach  convergence | ~490\* epochs(~50mins) |
| Total time spent | ~25 hours |
| Meet  termination criteria | No |
| Pi loss | Histogram  Description automatically generated with low confidence |
| Vf loss | Chart, histogram  Description automatically generated |
| Average steps per epoch | Chart, histogram  Description automatically generated |

# **Hyper-parameter settings**

Implementation detailed settings won’t be discussed here.

|  |  |
| --- | --- |
|  | PPO |
| Actor learning rate | 3e-4 |
| Critic learning rate | 1e-3 |
| Gamma | 0.99 |
| Hidden size | 64 |
| Lambda for GAE | 0.95 |
| Clip ratio | 0.2 |
| Update frequency (actor; critic) | 20; 80 |
| Entropy coef | 0.001 |

# **Findings:**

As the plot shows, during the early stage (approximately from 200 to 1800), the highest average rewards hit 299.8321649563972, which is close enough to solve the environment. Later, the average rewards dropped a little bit. However, the average steps per epoch drops (see the plot above), which means, the agent is better at walking to the end with less interactions, in other words it might have figured out other manners to run faster. And the standard deviation during the late stage decreased with approximately same average which might also be a sign of more robustness. Further, I tried to stop earlier with the same random seeds and hyper-parameter settings and to find what’s different from early-stage learning and late-stage learning. In the rendered environment, the agent from early-stage learning is also able to reach the endpoint, however it chooses to use only one leg to walk and another one for balancing. So, the agent from early-stage learning was crawling to the end instead of walking.

I also tried to increase the complexity of neural models (hidden\_size increased to 256) and longer the learning process (20k episodes). Here is the result from over-learning agent:

Chart, histogram

Description automatically generated

In the rendered environment, the agent tends to rush to the end, and ends up using one leg only for balancing and another for jumping. So, the agent tries to reach the end faster by hopping instead of walking, which is hard to reach balance and the agents falls a lot. However, if it successfully reaches the end, it will have highest rewards.

|  |  |  |
| --- | --- | --- |
| Under-learning agent | Optimal policy | Over-learning agent |
|  |  | Shape  Description automatically generated |

Although the optimal policy generated by my implementation does not hit the termination criteria, the agent is still able to reach the end in most trials (8/10) using two legs alternatively for either balancing or walking in the rendered environment. If choose PPO\_BipedalWalker\_5000.h5, it will most likely walk to the end without falling once, but it will be using one leg to balance and another to walk.

# **Manual book**

Run train:

python train\_BipedalWalker\_chenz51.py

Run test:

python test\_BipedalWalker\_chenz51.py

# **Reference**

OpenAI Implantation of RL algorithms

See code @[github](https://github.com/openai/spinningup)