2D 2DOF Robotic Arm Manipulator Control Using Reinforcement Learning

Doohyun Lee
School of Mechanical Engineering, Yonsei University
50, Yonsei-ro, Seodaemungu, Seoul, Republic of Korea
dhleekr7@gmail.com

Abstract

I present a method for control Two-Dimensional 2 Degrees of freedom(DOFs) Robotic Arm Manipulator using reinforcement learning. I compared the performances such as convergence time, time taken to train, and the number of successful times during training, etc. for various reinforcement learning algorithms using stable baselines library. Results were simulated through an environment created using the GYM environment.

1. Introduction

Two-Dimensional 2 DOF robotic arm manipulator can be fully controlled by other control methods, such as feedback control, but I have tried to achieve optimal control through reinforcement learning. For reinforcement learning, many tasks such as state and action settings, and reward engineering are required, and I compared the performance through simulation and set them up. I aim to do minimum expected time control and try to solve this through reward engineering.

2. Background

2.1 Deep Deterministic Policy Gradient(DDPG)

DDPG proposes an algorithm for continuous control. Use the replay buffer to enable online batch update. Also, by using soft update, it prevents rapid learning, and when using different scale features as the state in Neural Net, it is difficult to generalize, but batch normalization forces input from each layer to become unit Gaussian to solve this problem.

2.2 Advantage Actor Critic(A2C)

A2C proposes a conceptually simple and lightweight framework for deep reinforcement learning that uses asynchronous gradient descent for optimization of deep neural network controllers.

2.3 Soft Actor Critic(SAC)

Soft Actor-Critic is an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework. In this framework, the actor aims to maximize expected reward while also maximizing entropy. That is, to succeed at the task while acting as randomly as possible

2.4 Actor Critic using Kronecker-Factored Trust Region(ACKTR)

ACKTR is the algorithm that applies trust region optimization to deep reinforcement learning using a recently proposed Kronecker-factored approximation to the curvature. It extends the framework of natural policy gradient and propose to optimize both the actor and the critic using Kronecker-factored approximate curvature with trust region. It is also a method that learns non-trivial tasks in continuous control as well as discrete control policies directly from raw pixel inputs.

2.5 Proximal Policy Optimization Algorithms(PPO)

PPO proposes a novel objective function that enables multiple epochs of minibatch updates. PPO has some of the benefits of trust region policy optimization(TRPO), but it is much simpler to implement, more general, and has better sample complexity.

3. Two-Dimensional Manipulator Environment

3.1 Target

In the case of target, brownian motion was set to be performed with constant linear velocity of 1.2.

3.2 State

I used position error as the state. When the link with the end-effector was called link2, the calculation method was as follows.

$$TF_{link2\_to\_target} = (TF_{link2\_global})^{-1} X_{target} \quad (1)$$

$$TF_{err1} = TF_{link2}\, TF_{link2\_to\_target} \quad (2)$$

$$TF_{err2} = TF_{link1}\, TF_{joint2}\, TF_{err1} \quad (3)$$

$$TF_{err3} = TF_{joint1}\, TF_{err2} \quad (4)$$

Where TF is transformation matrix and X is position.

3.3 Action

Robot's linear velocity was set at a maximum of 1 and compared by changing the angular velocity of joint1, and

joint2. First, I set θ_{joint1}, $\theta_{joint2} \in [\ -\pi\ ,\ \pi\ ]$, secondly , θ_{joint1}, $\theta_{joint2} \in [\ -\frac{\pi}{2}\ ,\ \frac{\pi}{2}\ ]$, lastly θ_{joint1}, $\theta_{joint2} \in [\ -2\pi\ ,\ 2\pi\ ]$. As a result of comparison, θ_{joint1}, $\theta_{joint2} \in [\ -2\pi\ ,\ 2\pi\ ]$ had the best performance, so I used it.

3.4 Reward

First, if the distance between the target and the end effector is lower than the preset tolerance(0.1), the agent determined that it reached the target and gave a +1000 reward and finished the episode. And when the end effector did not reach the target, agent gave a negative reward of the squared distance between the target and the end effector, giving it a larger negative reward when the distance was far away. In addition, it is possible to perform minimum expected time control as it continues to receive negative rewards over time. And after more than 1000 timesteps, agent decided to time over and finished the episode with a -3000 reward, even when the robot was out of boundary, agent got a -3000 reward and finished the episode. Through this reward engineering, I constructed an environment so that the robot can arrive at the target as soon as possible.

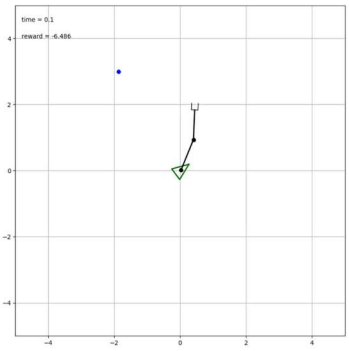


Fig. 1. 2D Manipulator simulation enviroment

4. Simulation Results

4.1 Action space

I compared the convergence time by changing only the conditions of action space under the same conditions as reinforcement algorithm, learning rate, and gamma. The results are shown in Fig. 2 below.

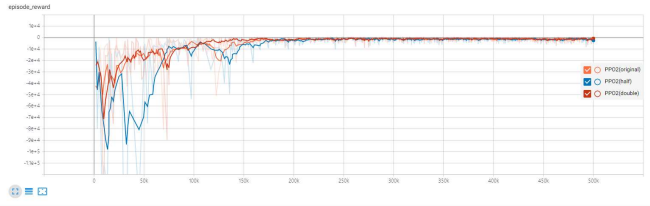


Fig. 2. Graph of convergence time as action space changes

4.2 Reinforcement Learning Algorithm

The conditions such as training timesteps, learning rate, and gamma were the same, and only reinforcement learning algorithms were changed to compare which algorithm performed well. In order to evaluate performance, the evaluation was performed by convergence time, average episode length, number of successes during training, and the number of out of boundary cases. The results are shown below in Fig. 3 and Fig. 4.

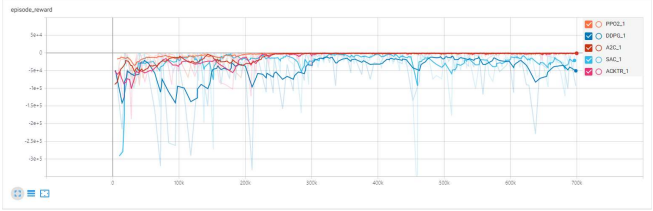


Fig. 3. Graph of convergence time as RL algorithm changes

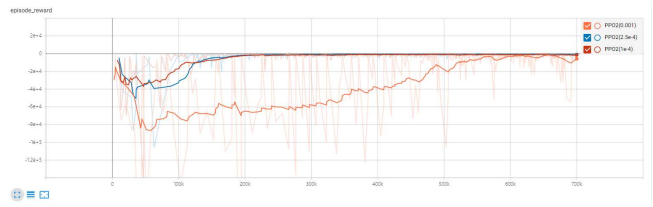
	A2C	SAC	DDPG	PPO2	ACKTR
Episode len mean	320	1450	4000	290	312
Successes	1165	289	122	1587	1224
Out of boundary	276	48	33	107	163

Fig. 4. Table for the performance of each algorithm

If you look at Fig. 2, the convergence time was good in order of PPO2, A2C, ACKTR, SAC, DDPG. Furthermore, in Fig. 4, you can see that PPO2 has the best overall performance. Therefore, I used PPO2 algorithm for train.

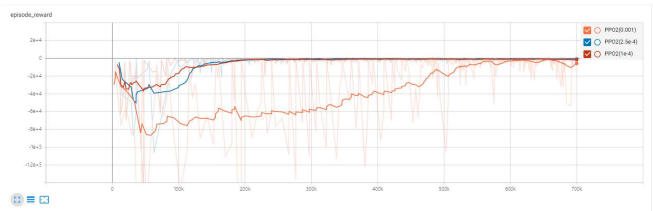
4.3 Learning rate and Gamma

To determine the learning rate and gamma, I compared the them by changing the learning rate and gamma only under the same conditions as shown above. So, the results are shown below in Fig. 5 and Fig. 6.



	lr = 1e-3	lr = 2.5e-4	lr = 1e-4
Episode len mean	601	230	275
Successes	332	2049	1535
Out of boundary	39	69	107

Fig. 5. Graph and table of performance according to learning rate



	gamma = 0.99	gamma = 0.9	gamma = 0.5
Episode len mean	1300	230	260

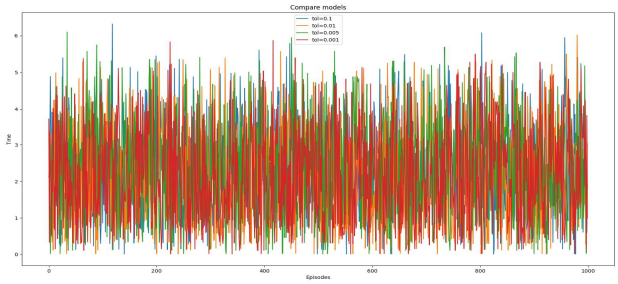
Successes	224	2049	1713
Out of boundary	88	69	241

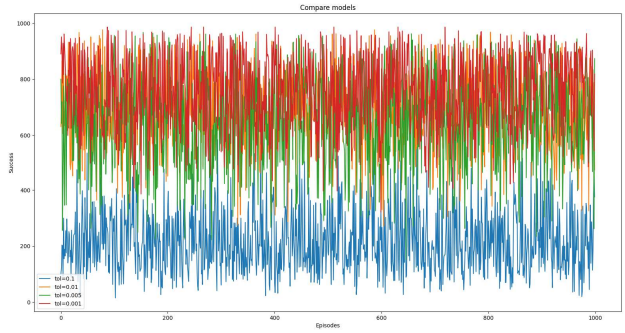
Fig. 6. Graph and table of performance according to gamma

The above Fig. 5 and Fig. 6 show that the best performance is when the learning rate is 2.5e-4 and the gamma is 0.9.

5. Problem

Although the agent trained by PPO2 algorithm arrives at the target quickly, simulations show that the end effector is vibrating a lot. As a result, when the end effector went near the target, it was often missed without catching well. To solve the problem, I changed the tolerance of training environment and trained the model. Fig. 7 shows that it takes the shortest time to reach the target when tolerance is 1e-3 and is holding the target well without missing it. Thus, this change can complement the vibration of the end effector.





	tol = 0.1	**tol = 0.01**	**tol = 5e-3**	**tol = 1e-3**
Time	2.319	2.236	2.302	2.221
Number of successes during 1000 timesteps	219.104	693.202	664.078	756.559

Fig. 7. Graph and Table for the performance of according to tolerance

6. Conclusion and Future work

I have presented a reinforcement learning method for controlling the two dimensional 2 DOFs robotic arm manipulator. I set the state and action and tried to perform minimum expected time control through reward engineering, and compared various reinforcement learning algorithms to determine the appropriate algorithm for learning this task. In addition, a comparison was made to obtain the appropriate learning rate and gamma, which was set to 2.5e-r and 0.9 for gamma. Also, to solve the vibration problem of the end effector, the tolerance of training environment was reduced to 1e-3 to increase the precision of the end effector. However, since this change does not eliminate the vibration itself, it is necessary to change the state or change the reinforcement learning algorithm in the future to solve the fundamental vibration problem.

REFERENCES

[1] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver & Daan Wierstra, “Continuous Control with Deep Reinforcement Learning”, ICLR, 2016.

[2] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, Koray Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning”, ICML, 2016.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine, “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”, ICML, 2018.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, “Proximal Policy Optimization Algorithms”, arXiv:1707.06347, August 2017.

[5] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, Jimmy Ba, “Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation”, arXiv:1708.05144, August 2017.