2019 19th International Conference on Control, Automation and Systems (ICCAS 2019)
Oct. 15∼18, 2019; ICC Jeju, Jeju, Korea

Control of Furuta Pendulum with Reinforcement Learning

Dohyeok Lee, Usama Muhammad and Dong Eui Chang

School of Electrical Engineering, KAIST, Daejeon, 34141, Republic of Korea
{dohyeoklee16, usama, dechang}@kaist.ac.kr

Abstract: Usually, in reinforcement learning (RL), the learning is performed on simulation and the learned policy is transferred to the real-world system. This policy, however, is susceptible to poor performance in the real-world settings. In this work, we study if it is possible to perform learning directly on the real-world system and compare the performance of learned policies with those learned in simulation. We use Furuta pendulum for our experiments and use train agents using DDPG and PPO algorithms for the control of Furuta pendulum. We also compare the RL based controllers with energy-based and PD controllers for swing up and balancing control tasks, respectively.

Keywords: Reinforcement Learning for Control, sim2real

1. INTRODUCTION

In reinforcement learning, an agent learns to solve a task by interacting with a model of real world in the simulation. This model is only an approximation of the real world setting and when the learned policy is transferred to the real world system, it performs poorly. In this paper, we investigate if it is possible to perform learning on the real world system and compare the performance of policies learned on real world system with those learned in simulation and later transferred to the real system without any fine-tuning. We use the Furuta pendulum as a real system for our experiments. The Furuta pendulum, first developed by Furuta [1], is an underactuated and extremely nonlinear system; see Figure 1.

Reinforcement Learning is a sub-field of machine learning where an agent interacts with its environment, possibly of unknown dynamics. While the environment is in state s_t at time step t, the agent chooses an action a_t from a set of possible actions i.e. $a_t \in \mathcal{A} = \{1, \dots, |\mathcal{A}|\}$ following a policy $\pi(s)$ and receives the reward r_t following a scalar reward function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. As a result, the environment moves into some next state s_{t+1} depending on transition probability distribution $\mathcal{P}$. We denote $\gamma \in (0, 1]$ as discount factor and ρ_0 as initial state distribution. The goal of learning is to find a policy π that maximizes the expected discounted return $R_t = \mathbb{E}_{\pi,\mathcal{P}}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau\right]$. Here, the policy π gives a distribution over actions in a state.

Reinforcement learning algorithms can be classified as either *off-policy* or *on-policy* algorithms. Q-learning and Deep Deterministic Policy Gradient(DDPG) are popular off-policy algorithms while Proximal Policy Optimization(PPO) and TRPO are on-policy algorithms. See [2] for a detailed introduction of reinforcement learning.

RL methods considered in this paper include DDPG and PPO. DDPG [3] belongs to the class of off-policy RL methods and consists of a critic network that learns the state dependent action value function and an actor network that learns the policy. PPO [4], on the other hand, is an on-policy RL algorithm and sequentially improves a policy. It optimizes a surrogate objective that imposes soft constraint on policy deviation in the learning process.

2. CONTROL OF FURUTA PENDULUM

The Furuta pendulum is shown in Figure 1 where the pendulum link is connected to the end of the rotary arm. The arm has a length of L_r, a moment of inertia of J_r and its angle θ increases positively when it rotates counterclockwise (CCW). The pendulum has a length of L_p having its center of mass at $L_p/2$. The moment of inertia about its center of mass is J_p. The inverted pendulum angle α is zero when it is upright and increases positively as it rotates CCW. The arm has mass m_r while the pendulum has mass m_p.

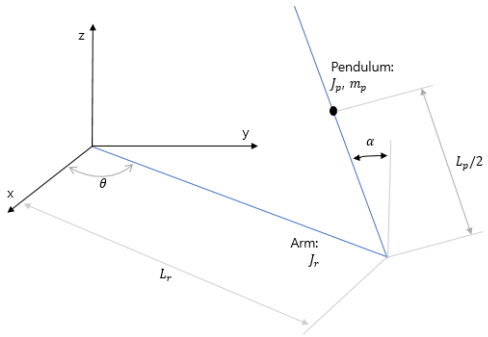

Fig. 1 Furuta Pendulum

Control of Furuta pendulum is divided into two separate control problems: swing up control where the pendulum is to swing up to upright position starting from down position and balance control that makes the pendulum stay upright. We use DDPG algorithm to train an agent for swing up control and PPO algorithm trains an agent for balance control. For swing up control, the reward function is given as $-[1 - \cos^2 \alpha]$ and the episodes is finished when $|\alpha| < 10°$. For balance control, the reward is $-0.1|\dot{\alpha}|$ where -0.1 is scale factor to make backpropagation feedback large enough. Note that dynamics of the system are not known to the agents during the training process. The reason why DDPG and PPO were selected is because they are popular RL algorithms for offline, deterministic and online, stochastic policy.

As baseline, we use hybrid control algorithm that is composed of energy controller for swing up control and PD controller for balance control. For swing up controller,

we use the control law

$$u = -(m_r l_r r_m / k_t)\text{sat}_{u_{max}}(\mu(E - E_r)\text{sign}(\dot{\theta}_2 \cos\theta_2)),$$

where μ is a tunable control gain, E is the sum of potential and kinetic energies of the pendulum while E_r is the reference energy of the pendulum that equals the pendulum energy at the upright position. Here, r_m is the terminal resistance and k_t is the torque constant. For the balance control in baseline, we use PD controller given below:

$$u = -2\theta + 35\alpha - 1.5\dot{\theta} + 3\dot{\alpha}.$$

3. EXPERIMENTS

We use Quanser Qube Servo-2 [5] as a Furuta pendulum for our experiments. To communicate with and control the system, we use the OpenAI Gym wrapper for the Quanser Qube Sevvo-2 [6]. The implementation of DDPG and PPO used as well as the details of hyperparameters is given in github page(`https://github.com/LeeDohyeok/Control-Furuta`). An experiment video is available at(`https://youtu.be/a6W6u8iMDU8`)

For swing up control, we trained the agent for 100 iterations where each iteration has maximum steps of 5000. We used the sampling frequency of 1kHz. The training process is shown in Figure 2. We tested the policies for 100 iterations where DDPG agent got a score of -7132.51 ± 928.31 while energy controller got a score of -5404.97 ± 255.46 while random policy received a score of -19589.0 ± 181.81.

We trained agent for balancing control using PPO for 100 iterations, 5000 steps per iteration, using the sampling rate of 1kHz. The training process is shown in Figure 3. Testing for 100 iterations, PPO agent received a score of -735.75 ± 222.71 while baseline PD controller got a score of -21.19 ± 0.834. Here, random policy got a score of -2421.08 ± 308.29.

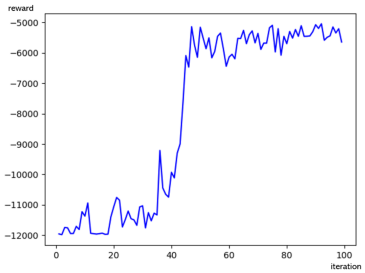

Fig. 2 DDPG training reward curve

3.1 Comparison with Training in Simulator

We trained the PPO agent for balancing the pendulum on simulator and transferred the policy without any fine-tuning on the real system. The training conditions are same as for training on the real system and the training progress is shown in Figure 4. We compared the performance of this policy with the policy learned directly on the real Furuta pendulum system for 100 test episodes. The policy learned on real system got a score of -735.75 ± 222.71 while the one learned in simulation got a score of -1808.70 ± 303.66.

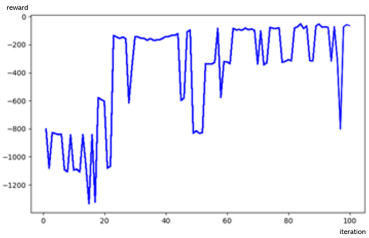

Fig. 3 PPO training reward curve

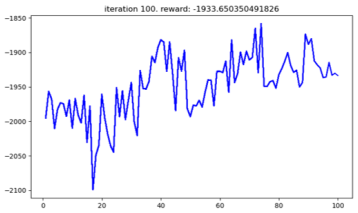

Fig. 4 Training of PPO agent for balancing task in simulation.

4. CONCLUSION

In this study, we studied if the RL agents can be trained on real-world systems and compared their performance with those trained in the simulation. Experiments show that policies learned on real-systems attain better performance that the policies learned in simulation and later transferred to real system without fine-tuning.

REFERENCES

[1] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*. Adaptive computation and machine learning, MIT Press, 1998.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[5] "Qube - servo 2 - quanser." `https://www.quanser.com/products/qube-servo-2/`. Accessed: 2019-06-19.

[6] "Openai gym wrapper for the quanser qube and quanser aero." `https://github.com/BlueRiverTech/quanser-openai-driver`.