Reinforcement Learning for Cart Pole Inverted Pendulum System

Atikah Surriani
Department of Electrical Engineering and Information Technology Gadjah Mada University
Yogyakarta, Indonesia
atikah.surriani.sie13@ugm.ac.id

Oyas Wahyunggoro
Department of Electrical Engineering and Information Technology Gadjah Mada University
Yogyakarta, Indonesia
oyas@ugm.ac.id

Adha Imam Cahyadi
Department of Electrical Engineering and Information Technology Gadjah Mada University
Yogyakarta, Indonesia
adha.imam@ugm.ac.id

***Abstract*— Recently, reinforcement learning considered to be the chosen method to solve many problems. One of the challenging problems is controlling dynamic behaviour systems. This paper used policy gradient to balance cart pole inverted pendulum. The purpose of this paper is to balance the pole upright with the movement of the cart. The paper employed two main policy gradient-based algorithms. The results show that PG using baseline has faster episodes than reinforce PG in the training process, reinforce PG algorithm got higher accumulative reward value than PG using baseline.**

Keywords—reinforcement learning, policy gradient, policy gradient baseline, cart pole inverted pendulum.

I. INTRODUCTION

Recently, reinforcement learning considered to be the chosen method to solve such a dynamic behaviour system [1]. It always takes more time to make a dynamic non-linear model and designing the control law [2]. An alternative approach comes to solve this problem, which is reinforcement learning. The reinforcement agent learns the behaviour of the system through trial-and-error interactions with its environment. This method is formulated using Markov Decision Process (MDP). Instead of concern with the model specification and controlling design, reinforcement learning concerns how the agent gets the maximum reward [3]. This mechanism is promising to achieve the optimal solution of the task.

Some researchers still use classical methods to solve cart pole pendulum, such as [4] using LQG optimal control to stabilizes an inverted pendulum. Paper [5] uses pole placement and LQR to control the inverted pendulum and tracing its movement. Then machine learning issue impacts the controlling method to use some learning algorithms like neural controller based on reinforcement learning is applied to control a rotational inverted pendulum by brown et, all [6]. This paper also uses the value-method reinforcement algorithm, Deep Q-Network (DQN). This paper [7] uses active exploration of reinforcement learning to control inverted pendulum.

This paper used policy gradient from William [8], to balance cart pole inverted pendulum. Policy gradient (PG) is a powerful algorithm. It is applied very well in Markov's chain, regarding its stationary distribution for policy. This parameter affects the state stuck in the training process. The policy gradient approach changes the main part of the objective function, thus it reduces the expensive computational process [9]. William introduced a variable known as a baseline. The baseline can increase the convergence and consider affect the qualitative behavior. This work wants to compare the performance results of each algorithm, the original PG, and the PG with the baseline. The comparison of the two approaches is important because, policy gradient will be used as a basic algorithm to update the network. Moreover, it will strongly affect the performance of development approaches which is built based on PG. It is interesting as the baseline is theoretically proven to keep the estimation of the weight unbiased [10]. It is proved with a network of Bernoulli-logistic units [8].

The system that is used in this paper is the cart-pole system as a common regular environment using the standard dynamics and parameters. The purpose of this paper is to balance the pole upright with the movement of the cart. Reinforcement learning policy gradient will be employed to solve this problem. Thus, the cart pole can stand upright as expected.

II. ENVIRONMENT

This paper using a cart pole inverted pendulum as an environment [11]. The dynamic of cart pole inverted pendulum based on [12] is defined as,

$$\ddot{x} = \frac{F - m_p l(\ddot{\theta}cos\theta - \dot{\theta}^2 \sin\theta)}{m_c m_p}, \quad (1)$$

$$\ddot{\theta} = \frac{g \sin\theta\,(m_c + m_p) - (F + m_p l(\ddot{\theta}cos\theta - \dot{\theta}^2 \sin\theta)\cos\theta}{\frac{4}{3}l(m_c + m_p) - m_p l\, cos^2\theta}. \quad (2)$$

Where, $m_p = 0.1$kg is mass of the pole, $l = 0.5$m is half-length of the pole, m_c =1kg, as the mass of the cart, $g = 9.8\ \ s^{-2}$ is gravity. F is the force that is employed in this system has the range [-10N, 10N]. The inverted pendulum cart is illustrated in Fig. 1. Regarding Fig 1, the inverted pendulum applies a pole threshold of 0.2094 rad and x threshold 2.4 m. The pole can swing 360 degrees. The inverted pendulum works by moving the cart using horizontal force. The pole will move as the cart is pushed by the force. The target of the system is balancing the pole upright.

2021 IEEE Industrial Electronics and Applications Conference (IEACon) | 978-1-7281-9253-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/IEACON51066.2021.9654440

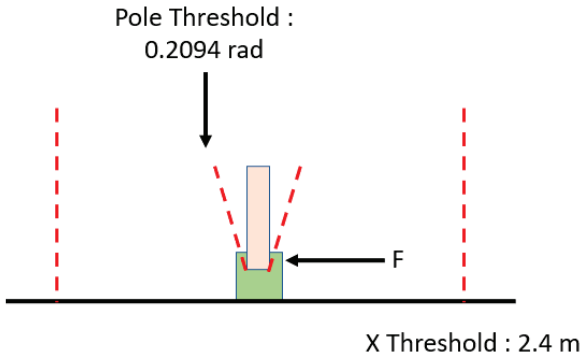

Fig 1. Cart Pole Inverted Pendulum Environment.

III. Reinforcement Learning

Reinforcement Learning is formulated using Markov Decision Process (MDP). MDP consists of a few tuples: $\boldsymbol{A_t}$ as an action, $\boldsymbol{O_t}$ is observation, and $\boldsymbol{R_t}$ is reward. Since the environment directly learns from itself, so, $\boldsymbol{O_t} = \boldsymbol{S_t}$. Details of this system are shown in Fig 2. Fig 2 describes the flow chart of reinforcement learning, the environment gives observation to the agent, and the agent gives action to the environment, then the environment gives reward to the agent as a compliment beside the observation. In the cart pole inverted pendulum, the action of the agent is the force that is applied to push the cart through the x-axis. This paper used a discrete environment, so it applied the discrete action space. The agent gets the observation from the environment.

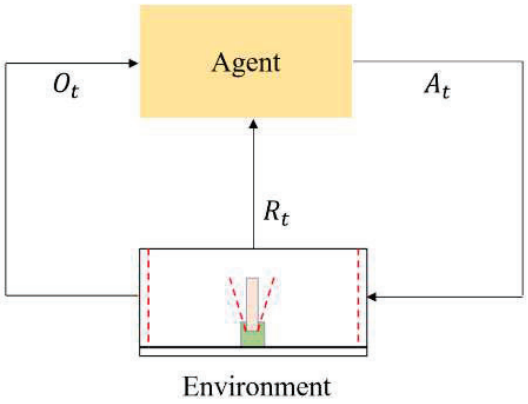

Fig 2. Diagram of Reinforcement Learning.

It consists of position, the velocity of the cart from the inverted pendulum, pole's angle, and derivative of the pole the pendulum. Thus, the state can be defined as,

$$\boldsymbol{O_t} = \boldsymbol{S_t} = [x\ \dot{x}\ \theta\ \dot{\theta}]. \tag{3}$$

The goal of this system is to balance the pole of the cart upright. The goal achieved the finish when it satisfied a few conditions when the vertical position of the pole is between the angle and obtains the vertical position in 0 rad, Position of the inverted pendulum considers below the x threshold. The initial angle of the inverted pendulum is within -0.05 rad and 0.05 rad. Terminates episode when the vertical position of the pole is more than the pole threshold or when the inverted pendulum cart moves more than the x threshold at the x-axis. The Reward is 1 every time step, and the penalty is -5. It is defined in eq (4),

$$R_t \begin{cases} 1, & if\ it\ achieved\ the\ goal \\ -5, & if\ it\ failed \end{cases}. \tag{4}$$

.

A. Policy Gradient

The policy gradient (PG) algorithm is based on Monte-Carlo reinforce, thus, it is a model-free RL method [9]. It is also an on-policy, gradient-based reinforcement learning method [13]. PG method can be applied in discrete and continuous environments. It is one of the PG method's advantages. This work applies two different PG approaches. The first PG approach is reinforce PG algorithm. First, let $\boldsymbol{\pi}$ denotes as policy with respect to parameter $\boldsymbol{\theta_\pi}$ and $\mathrm{d}\boldsymbol{\theta_\pi}$ is the gradient of policy parameter. Thus, in [14] derived PG policy algorithm from using a sum over some variables into an expectation expression policy. It is defined as follows,

$$\mathrm{d}\boldsymbol{\theta_\pi} = E_\pi\left[\boldsymbol{G_t}\,\frac{\nabla\pi(\boldsymbol{S_t}|\boldsymbol{\theta_\pi})}{\pi(\boldsymbol{S_t}|\boldsymbol{\theta_\pi})}\right]. \tag{5}$$

Where, $\boldsymbol{G_t}$ is return. Return is defined as,

$$\boldsymbol{G_t} = \sum_{k=t}^{T} \boldsymbol{\gamma^{k-t} R_k}. \tag{6}$$

Equations (5) and (6) can be combined and replaced the fractional using the eligibility vector introduced in [14], thus it becomes,

$$\boldsymbol{d\theta_\pi} = \sum_{t=1}^{T-1} \boldsymbol{G_t \nabla_{\theta_\pi}} \ln \boldsymbol{\pi}(\boldsymbol{S_t}|\boldsymbol{\theta_\pi}). \tag{7}$$

Updating the policy parameter is defined as,

$$\widehat{\boldsymbol{\theta}}_{\boldsymbol{\pi}} = \boldsymbol{\theta_\pi} + \boldsymbol{\alpha d\theta_\pi}. \tag{8}$$

Where α is the learning rate. The policy parameter network uses observation and estimates the return then it updates the policy parameter. The pseudocode for reinforce PG algorithm is shown as follows,

Policy Gradient Algorithm

1. Initialize $\boldsymbol{\pi(s)}$ with random parameter policy $\boldsymbol{\theta_\pi}$
2. Generate trajectory episode on policy $\boldsymbol{\pi(s)}$:
   $\boldsymbol{S_0, A_0, R_1, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T}$.
3. For t=1, 2, … T
   - Estimation return $\boldsymbol{G_t}$
   - Accumulate the gradients for the policy-network
   $$\boldsymbol{d\theta_\pi} = \sum_{t=1}^{T-1} \boldsymbol{G_t \nabla_{\theta_\pi}} \ln \boldsymbol{\pi}(\boldsymbol{S_t}|\boldsymbol{\theta_\pi}).$$
4. Update parameter: $\widehat{\boldsymbol{\theta}}_{\boldsymbol{\pi}} = \boldsymbol{\theta_\pi} + \boldsymbol{\alpha d\theta_\pi}$.

The second reinforce policy gradient applies baseline, b, that be introduced in [8]. The baseline affects the estimation of the gradient unbiased [10]. It works as a reduction factor of various gradients during estimation. The baseline can be written in form of a function or a variable that is not in form of varying action. Based on [14], a baseline is usually denoted as state value estimation, $\boldsymbol{V(S_t|\theta_V)}$. Where $\boldsymbol{\theta_V}$ is parameter vector. Thus, for reinforce PG baseline is using advantage function δ_t, defines as,

$$\boldsymbol{\delta_t} = \boldsymbol{G_t} - \boldsymbol{V(S_t|\theta_V)}. \tag{9}$$

This paper is funded by Direktorat Penelitian Gadjah Mada University.

Thus, the gradient of the state value-network is denoted as,

$$d\boldsymbol{\theta}_V = \sum_{t=1}^{T-1} \boldsymbol{\delta}_t \nabla_{\theta_V} \boldsymbol{V}(\boldsymbol{S}_t|\boldsymbol{\theta}_V). \quad (10)$$

Updating the state value parameter is defined as,

$$\widehat{\boldsymbol{\theta}}_V = \boldsymbol{\theta}_V + \boldsymbol{\beta} d\boldsymbol{\theta}_V. \quad (11)$$

Where β is the learning rate for the state value function**.** The policy network obtains from (7) with replacement of the return with the advantage function, so it can be denoted as follows,

$$d\boldsymbol{\theta}_\pi = \sum_{t=1}^{T-1} \boldsymbol{\delta}_t \nabla_{\theta_\pi} \ln \boldsymbol{\pi}(\boldsymbol{S}_t|\boldsymbol{\theta}_\pi). \quad (12)$$

Updating the parameter is used (8). The training process of policy gradient is explained as this pseudocode,

Policy Gradient Algorithm

1. Initialize actor $\boldsymbol{\pi}(\boldsymbol{s})$ with random parameter policy $\boldsymbol{\theta}_\pi$
2. Initialize state value **V** $(\boldsymbol{s})$ with random parameter policy $\boldsymbol{\theta}_V$
3. Generate trajectory episode on actor policy $\boldsymbol{\pi}(\boldsymbol{s})$:
   $\boldsymbol{S_0, A_0, R_1, S_1, A_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T}$.
4. For t=1, 2, ... T
- Estimation return $\boldsymbol{G_t}$
- Compute the advantage function $\boldsymbol{\delta_t}$, using the baseline value function estimation.
5. Accumulate gradient of the state value network
$$d\boldsymbol{\theta}_V = \sum_{t=1}^{T-1} \boldsymbol{\delta}_t \nabla_{\theta_V} \boldsymbol{V}(\boldsymbol{S}_t|\boldsymbol{\theta}_V)$$
6. Accumulate gradient of the policy network
$$d\boldsymbol{\theta}_\pi = \sum_{t=1}^{T-1} \boldsymbol{\delta}_t \nabla_{\theta_\pi} \ln \boldsymbol{\pi}(\boldsymbol{S}_t|\boldsymbol{\theta}_\pi)$$
7. Update state value parameter: $\widehat{\boldsymbol{\theta}}_V = \boldsymbol{\theta}_V + \boldsymbol{\beta} d\boldsymbol{\theta}_V$
8. Update policy parameter**:** $\widehat{\boldsymbol{\theta}}_\pi = \boldsymbol{\theta}_\pi + \boldsymbol{\alpha} d\boldsymbol{\theta}_\pi$.

IV. Discussion and Result

Policy Gradient algorithm must be parameterized as in Table 1,

TABLE 1. ACTOR PARAMETER FOR PG APPROACH.

Learning Rate (α)	0.01
Gradient Threshold	1
Discount Factor (γ)	0.99

PG Baseline parameter is shown in Table 2,

TABLE 2. PARAMETER FOR PG BASELINE.

Learning Rate (α)	0.01
Learning Rate (β)	0.005
Gradient Threshold	1
Discount Factor (γ)	0.99

Function approximation is used for both of reinforce policy gradient and policy gradient using baseline in policy-network. Based on Table 1, the policy-network for the PG approach has the learning rate of 0.01, the gradient threshold 1, and the discount factor 0.99. Regarding Table 2, the learning rate for the policy-network of PG baseline is 0.01, the gradient threshold is 1. The learning rate of state value is 0.005, the discount factor 0.99. Both algorithms work with Deep Neural Network (DNN) as shown in Fig 3 and Fig 4,

Fig 3. Policy-Network

Fig 3 shows the policy-network has the observation as input, and the action as output. Fig 4 shows the PG baseline has one more state value network, it has one input from the state, and one output as the state value.

Fig 4. State Value (Baseline) Network

PG algorithm is used DNN to get the proper action to take, from the observation of the environment. The episode has a maximum of 1000 episodes, and it stops when it reached an accumulative cost value of 195.

Training result for reinforce policy gradient stop in 882 steps episode, while PG using baseline reached the desired value at 620 episodes. It described that PG using baseline has faster episode than reinforce PG in the training process. It is illustrated in Fig 5,

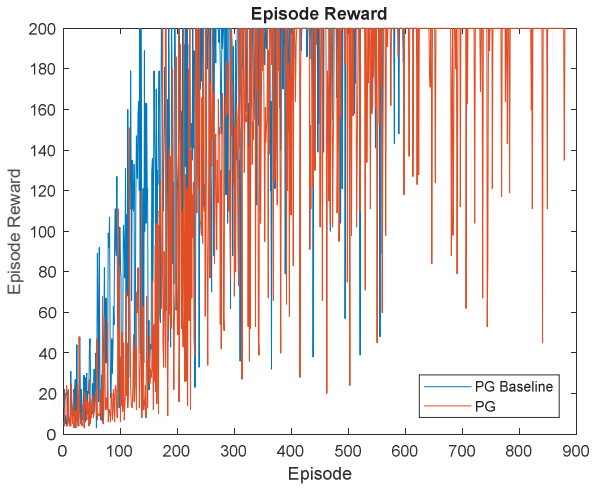

Fig 5. Episode Reward for Policy Gradient and PG Baseline.

Fig 5 shows that PG baseline finish the training faster, swiftly reached the cost value. Thus, reinforce PG algorithm has a slow training process. Fig 6 shows the average reward for the training,

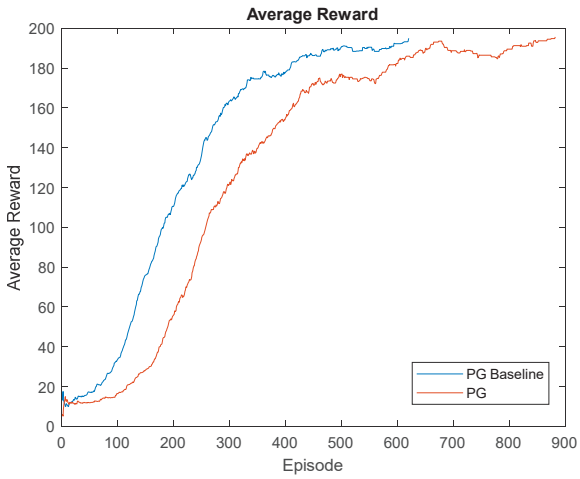

Fig 6. Average Reward for Policy Gradient and PG Baseline.

Fig 6 shows that the average reward for policy gradient gets the peak at 650 then the value got to slow down and reached the final at 882 episodes, while PG baseline already finish the training at 620 episodes. During the training process, both of algorithm successfully make the pole of the pendulum stands upright at the end of the episodes. This summaries result can be explained in Table 3,

TABLE 3. COMPARISON OF TRAINING RESULT.

Training Episode	**PG**	882
	PG-Baseline	620
Total Reward	**PG**	383
	PG-Baseline	342

For validation, both of algorithm was simulated with 500 maximum steps. After simulation, reinforce PG algorithm got a higher accumulative reward value than PG using baseline. Reinforce PG algorithm obtained accumulative 383 rewards after 500 steps, and PG using baseline obtained 342 total rewards after 500steps. This happens toward the achievement of the PG baseline to correct unbiased estimation [9]. The baseline can lead the function approximation to converge to the local optimum [15]. In the simulation, after 500 steps, Reinforce PG algorithm successfully make the pole standing upright, yet the PG baseline failed. Table 2 also shows simulation rewards from the training. The simulation after 500 steps for PG baseline is shown in Fig 7,

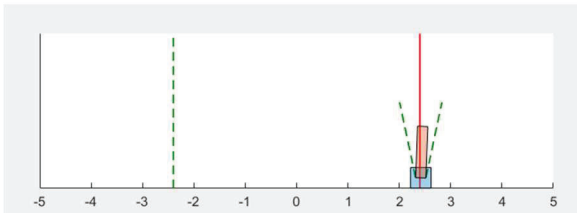

Fig 7. Cart Pole Inverted Pendulum Using PG Baseline.

PG baseline approach helps the environment to achieved fast training but there is no guarantee the simulation can be a success It happens towards baseline's characteristic. It only has an effect toward the update variance rate, nor the expected update algorithm [14]. Fig 8 shows the environment after being simulated using PG. It shows that the cart pole inverted pendulum successfully balances during simulation.

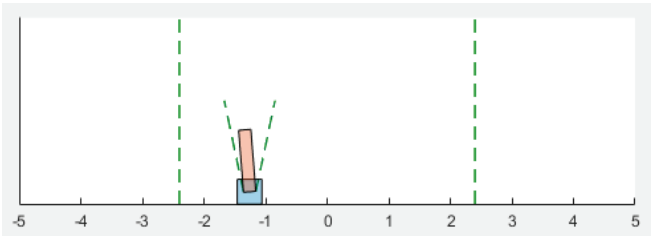

Fig 8. Cart Pole Inverted Pendulum Using PG.

V. CONCLUSION

The paper employed two main policy gradient-based algorithms. The algorithm is applied to the cart-pole inverted pendulum problem. The simulation after training results show that PG can balance the cartpole inverted pendulum. The PG baseline's simulation no longer success to balance the art pole inverted pendulum, yet it has faster episode than reinforce PG in the training process. Reinforce PG algorithm got higher accumulative reward value than PG using baseline to balance the cart pole inverted pendulum system.

ACKNOWLEDGMENT

The publishing process of the paper was supported by RTA 2021 Grant. We are thankful to Direktorat Penelitian UGM for their logistical support and for providing necessary guidance concerning projects.

REFERENCES

[1] S. Kumar, "Balancing a CartPole System with Reinforcement Learning - A Tutorial," arXiv, no. 0, 2020

[2] C. A. M. Escobar, C. M. Pappalardo, and D. Guida, "A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole," Appl. Sci., vol. 10, no. 24, pp. 1–19, 2020, doi: 10.3390/app10249013.

[3] Y. H. Liu et al., "Modification on the tribological properties of ceramics lubricated by water using fullerenol as a lubricating additive," Sci. China Technol. Sci., vol. 55, no. 9, pp. 2656–2661, 2012, doi: 10.1007/s11431-012-4938-y.

[4] R. Banerjee and A. Pal, "Stabilization of Inverted Pendulum on Cart Based on LQG Optimal Control," 2018 Int. Conf. Circuits Syst. Digit. Enterp. Technol. ICCSDET 2018, 2018, doi: 10.1109/ICCSDET.2018.8821243.

[5] C. Mahapatra and S. Chauhan, "Tracking control of inverted pendulum on a cart with disturbance using pole placement and LQR," 2017 Int. Conf. Emerg. Trends Comput. Commun. Technol. ICETCCT 2017, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1109/ICETCCT.2017.8280311.

[6] D. Brown and M. Strube, "Design of a Neural Controller Using Reinforcement Learning to Control a Rotational Inverted Pendulum," 2020 21st Int. Conf. Res. Educ. Mechatronics, REM 2020, 2020, doi: 10.1109/REM49740.2020.9313887.

[7] Y. Zheng, S. W. Luo, and Z. A. Lv, "Active exploration planning in reinforcement learning for inverted pendulum system control," Proc. 2006 Int. Conf. Mach. Learn. Cybern., vol. 2006, no. August, pp. 2805–2809, 2006, doi: 10.1109/ICMLC.2006.259002.

[8] R. J. Williams, "Simple Statistical Gradient- Following Algorithms for Connectionist Reinforcement Learning," Mach. Learn., vol. 256, pp. 229–256, 1992.

[9] L. Weng, "Policy Gradient Algorithms," 2021. https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html.

[10] S. Kapoor, “Policy Gradients in a Nutshell,” Towards Data Science, 2018. https://towardsdatascience.com/policy-gradients-in-a-nutshell-8b72f9743c5d.

[11] MathWorks, “Train PG Agent to Balance Cart-Pole System,” MathWorks, 2021. https://www.mathworks.com/help/reinforcement-learning/ug/train-pg-agent-to-balance-cart-pole-system.html (accessed Jul. 01, 2021).

[12] M. Riedmiller, J. Peters, and S. Schaal, “Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark,” in Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007) Evaluation, 2007, no. Adprl, pp. 254–261.

[13] Y. Duan, X. Chen, H. Rein, J. Schulman, and P. Abbeel, “Benchmarking Deep Reinforcement Learning for Continuous Control,” vol. 48, 2016.

[14] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (2nd Edition, in preparation). 2018.

[15] R. S. Sutton, D. Mcallester, S. Singh, Y. Mansour, P. Avenue, and F. Park, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” 1996.