Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 8, Number 1, pages 23–26, January 2019

Hybrid Policy Gradient for Deep Reinforcement Learning

Tomah Sogabe
i-PERC &
Engineering department,
The University of Electro-Communications
Tokyo, Japan
sogabe@uec.ac.jp

Malla Dinesh Bahadur
Grid, Inc.
Tokyo, Japan

Katsuyohi Sakamoto
Engineering deptpartment,
The University of Electro-Communications
Tokyo, Japan

Masaru Sogabe
Grid, Inc.
Tokyo, Japan

Koichi Yamaguchi
Engineering department,
The Univdrsity of Electro-Communications
Tokyo, Japan

Shinji Yokogawa
i-PERC &
Engineering department,
The University of Electro-Communications
Tokyo, Japan

Abstract- In this paper, we propose an alternative way of updating the actor (policy) in DDPG algorithm to increase convergence and stable learning process. In the basic actor-critic architecture with TD (temporal difference) learning of critic, the actor parameters are updated in the gradient ascent direction of TD-error of critic. Similar to basic actor-critic approach, in our proposed hybrid method, Hybrid-DDPG (shortly H-DDPG), at one-time step actor is updated similar to DDPG (gradient of critic output with respect to policy parameters) and another time step, policy parameters are moved in the direction of TD-error of critic. The algorithm is tested on OpenAI gym's Robo school Inverted Pendulum Swing up-v1 environment. Once among 5 trial runs, reward obtained at the early stage of training in H-DDPG is higher than DDPG. In Hybrid update, the policy gradients are weighted by TD-error. This results H-DDPG have higher reward than DDPG by pushes the policy parameters to move in a direction such that the actions with higher reward likely to occur more than the other. This implies if the policy explores at early stages good rewards, the policy may converge quickly otherwise vice versa. However, among the remaining trial runs, H-DDPG performed same as DDPG.

Keywords—deep reinforcement learning, DDPG, Actor-Critic, Hybrid algorithm

I. Introduction

Reinforcement Learning (RL) has shown successful result in the past few years in the discrete action space such as Atari games using DQN [1], Go game using actor critic with Monte Carlo Tree search [3]. For continuous control problems, there are mainly two types of algorithms: vanilla policy gradient and actor-critic architecture. Deep Deterministic Policy Gradient (DDPG) [5], Asynchronous Advantage Actor Critic (A3C) [9] has actor-critic architectures, Trust Region Policy Optimization (TRPO) [6], Proximal Policy Optimization (PPO) [7] can use vanilla policy gradient and actor-critic architecture. Except DDPG, the above mentioned RL algorithms use stochastic policy gradients and policy is parameterized as Gaussian distribution, from which action is sampled. Whereas in DDPG, policy maps state space to action value. In the policy evaluation, for exploration, policy output is summed with Ornstein-Uhlenbeck noise. DPG is a limiting case of stochastic policy gradient in the limit of variance approaches to zero [2]. Due to the deterministic behavior in DDPG, in the early stages of policy iteration, if the policy has seen good immediate reward, the policy converges to sub-optimal region and policy has hard time to reach global optimum. Here, we propose a method to find the global optimal policy and faster convergence by using actor update similar to stochastic policy gradient. In this Hybrid-DDPG at one-time step of policy update, same as DDPG, the policy parameters are moved in the direction of action-value gradient at another time step, instead of calculating the action-value gradient, TD error calculated for critic is applied at the action gradient of the actor network. It is believed that since the critic network does not reach optimum solution at the early stages of learning, the policy parameters may not converge to sub-optimal solution.

II. Background

1 preliminaries:

In the Reinforcement Learning, the control problem is formulated as Markov Decision Process (MDP). A MDP consists of state space (s), an agent to take an action from action space (a), transition dynamics $\mathrm{p}((s_{t+1}|s_t, a_t)$, Reward process ($\mathrm{p}(\mathrm{s}, \mathrm{a})$) and initial state distribution $\mathrm{p}\ (s_1)$ [4]. The goal of an agent is to find an optimum policy (π^*) such that it maximizes the total expected cumulative discounted reward.

Total discounted reward $R_t = \sum_i^T \gamma^{(i-T)} r(s_i - a_i)$.

Here, γ is the discount factor $\in$ [0~1].

2 Actor-critic and H-DDPG method:

In RL, actor-critic architecture is one of the policy gradient algorithm, used to tackle continuous action

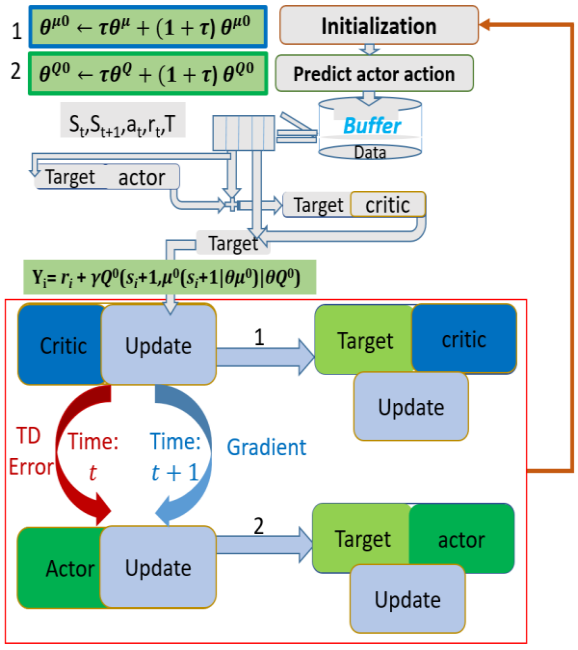

Figure 1. H-DDPG algorithm flow chart

problems. The actor is the agent (policy) which takes the action, and the critic criticizes the action how good or bad the sampled action from actor. The policy is parameterized with neural network and critic is the value-function. There are two types of actor-critic methods: stochastic and deterministic.

In the stochastic actor-critic off-policy setting, the performance objective is typically modified to be the value function of target policy, averaged over the state distribution of the behavior policy [8].

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\pi[\,(\nabla_\theta log\pi_\theta(a|s))Q^\pi(s,a)] \quad \text{--- (1)}$$

Instead of the $Q^\pi(s,a)$ in Equ. (1), the temporal difference (TD) error, $\delta = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ is used for actor update. In off-policy deterministic policy gradient, the actor update becomes

$$\theta^{k+1} = \theta^k + \alpha\mathbb{E}_{s\sim\rho^{\mu^k}}\left[\,(\nabla_\theta Q^{\mu^k}(s,\mu_\theta(s))\right] \quad \text{--- (2)}$$

By applying the chain rule, the policy improvement is decomposed into the gradient of action-value with respect to actions, and the gradient of policy with respect to policy parameters.

$$\theta^{k+1} = \theta^k + \alpha\mathbb{E}_{s\sim\rho^{\mu^k}}[(\nabla_a Q^{\mu^k}(s,a)|_{a=\mu_\theta(s)} * \nabla_\theta\mu_\theta(s)] \quad \text{--- (3)}$$

In the calculation of gradient of with respect to actions in Eq. (3), the error is considered as 1.0 at the output neuron which is propagated until the action gradient. Here, in this paper, we propose, similar to stochastic actor-critic, the TD-error (δ) is applied at the Eq. (3) instead of the action-value Q gradient. The TD-error is

$$\delta = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1})) - Q(s_t) \quad \text{--- (4)}$$

For critic update, this TD-error (TD (0) error) is used for backpropagation. We call this method as Hybrid Deep Deterministic Policy Gradient (H-DDPG) method. At one-time step, use the Eq. (3) and the next time step use the Eq. (5).which is diagrammatized in the Fig.1.

$$\theta^{k+1} = \theta^k + \alpha\mathbb{E}_{s\sim\rho^{\mu^k}}\left[(\delta * \nabla_a Q^{\mu^k}(s,a)|_{a=\mu_\theta(s)}\right] \quad \text{--- (5)}$$

3 Neural network structure

The structure of neural network in this algorithm is Actor-critic structure .Actor-critic structure combine the actor-only and critic-only structure aiming to gain both their advantages. In actor-critic structure, the policy and the value function are updated separately. The actor represents the policy and the critic evaluates the current policy that prescribed by the actor. Fig. 2 shows the diagram of the structure of Actor-critic. In this Figure, actor structure is normal structure of neural network were states Hidden layer and actions are dense, and in critic structure input are

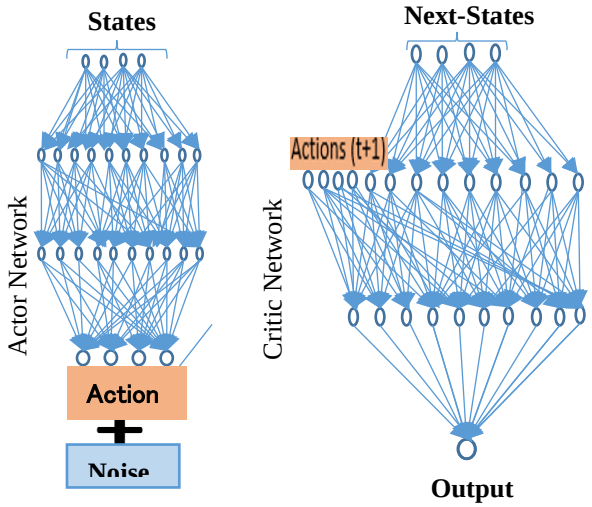

Figure 2. Hybrid-policy gradient and Actor-Critic structure Diagram

divided into two-parts : one is normal type and another is actions from the actor network which helps to find a parameter vector that maximizes the return.

III. PROPOSED METHOD

Hybrid-DDPG Algorithm

Randomly initialize actor $\mu(s|\theta^\mu)$ and critic $Q(s,a|\theta^Q)$ neural networks with weights θ^μ and θ^Q.

Initialize target network μ′ and Q′ with weights $\theta^{\mu'} \leftarrow \theta^{\mu}$ and $\theta^{Q'} \leftarrow \theta^{Q}$(Initially copy the same weights)

Initialize Replay Buffer R.

For episode 1 to m do:

Initialize Ornstein-Uhelnbeck (OU) noise for $\mathcal{N}$ for exploration

Environment reset to get initial observation s_0

For = 1 to Terminal state do:

Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ by adding noise

Apply action a_t to environment and observe next state s_{t+1} and reward r_t and terminal situation

Store following tuple in (s_t , a_t , r_t , s_{t+1} , terminal) in R

If Replay Buffer size > mini batch size:

Sample $\mathcal{K}$(mini batch) from R

Calculate TD $\boldsymbol{(0)_{target} = r_t + \gamma Q'\left(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})\right)}$

Update critic with mean squared loss:

$L = \frac{1}{K}\sum(TD(0)_{target} - Q(s_t, a_t))^2$

Update Actor (Hybrid):

At time step: t

$\boldsymbol{\nabla_{\theta^{\mu}} J \approx \frac{1}{K}\sum_i (\nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} * \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})\,|s_i}$

At time step: t+1

$\nabla_{\theta^{\mu}} J \approx \frac{1}{K}\sum_i TD(0)_{target} * \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})\,|s_i$

Update target networks:

$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1 - \tau)\theta^{Q'}$

$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu'}$

End for

End for

IV. RESULTS

For fair comparison of the DDPG and H-DDPG, the neural network architectures and hyper parameters are kept same as [5]. In the OU-noise the parameters are σ = 0.3, and θ = 0.15 .The replay buffer size is 200000. The discount Factor gamma (γ) , Actor learning rate, critic learning rate are 0.999, And 0.001 respectively. The τ is set as 0.001. RoboschoolInvertedPendulumSwingup-v1 (OpenAI Gym environments) is tested using both algorithms. Each run is continued for 500 episodes. The reward threshold is 0. For brevity, the Fig. 3 is plotted for 200 episodes. The total reward achieved in each episode is plotted. The game is run for 5 times. To test the learned policy with noise for making difficult to get a reward, we tested the learned policy every after 10 episodes (after 0,10, 20, …,500thepisode). The average reward of 10 test episodes is plotted in Fig. 4. Before test the algorithm we train it 1 time for the same episode and use the trained weight for the test.

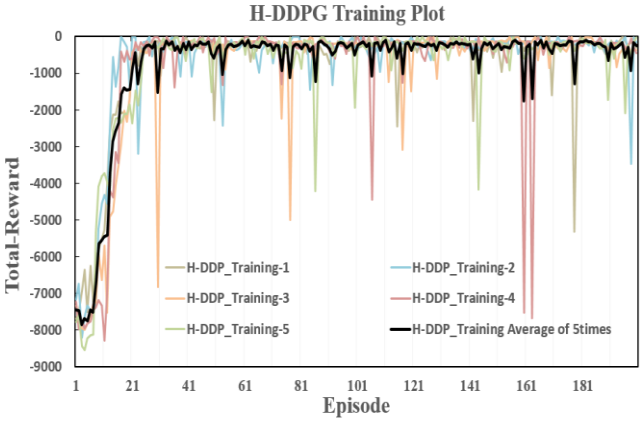

Figure 3. Rewards obtained in Hybrid-DDPG during training

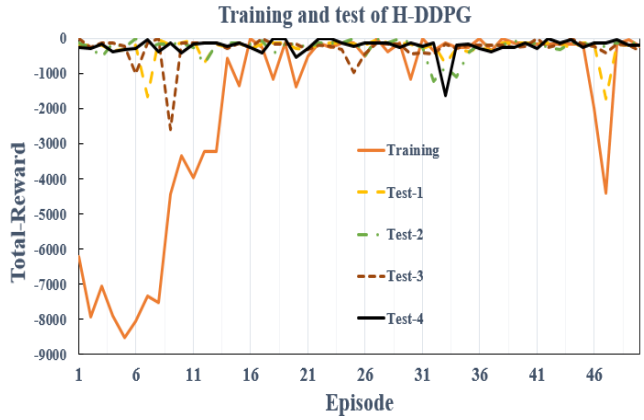

Figure 4. Training and test rewards obtained in Hybrid-DDPG

Fig. 3 shows the H-DDPG, Actor-Critic and DDPG agent respectively for their test performance using the trained weight. The total reward during training and an average of 4tests obtained rewards during test are curved in Fig. 5. The test average reward obtained by H-DDPG is quite stable than normal DDPG and Actor-Critic as seen in Fig. 5. In Fi.6, there are 5 trial runs of H-DDPG, DDPG, Actor-Critic plotted. The average reward of each algorithm is indicated by the bold line and the raw reward in each time were illustrated by the faded lines. The black line of H-DDPG average total reward approaches maximum value (~ 0) in much faster pace than Actor-critic and DDPG. The lower episode in the plot, H-DDPG's reward is more unstable than other but increasing in the episode its performance is getting stable near to zero. So, it indicates that H-DDPG can be considered as an alternative usable algorithm for continuous action space.

V. DISCUSSION AND COCLUSION

Once among 5 trial runs, in H-DDPG4 curve in Fig. 5, reward obtained at the early stage of training is higher than DDPG and actor-critic training run. In Hybrid update, the policy gradients are weighted by TD-error. This results in higher reward than DDPG and pushes the policy parameters to move in a direction such that the actions with higher

reward likely to occur more than the other. This implies if

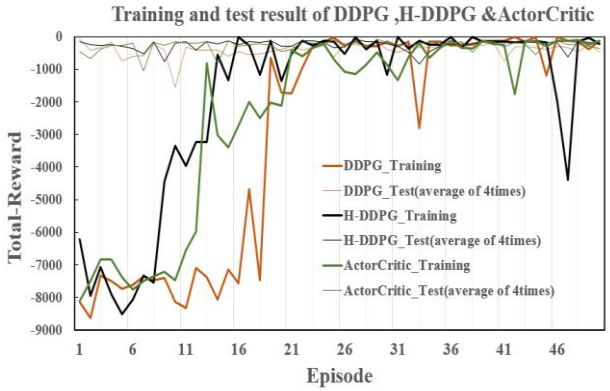

Figure 5. Training and test rewards obtained in DDPG, H-DDPG, Actor-Critic

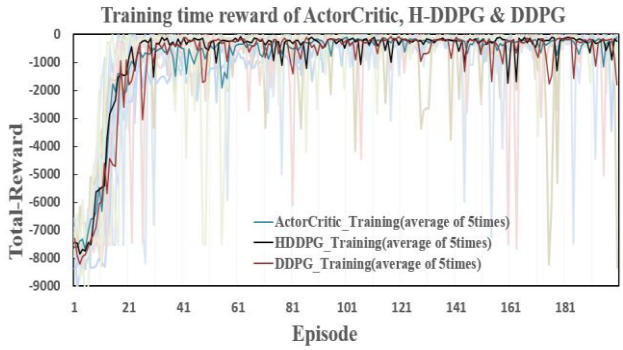

Figure 6. Training rewards obtained in DDPG, H-DDPG & Actor-Critic

the policy explores at early stages good rewards, the policy may converge quickly otherwise vice versa. However, among the remaining trial runs, there is no much difference between both of the algorithms. In the Fig.5, for the case HDDPG and DDPG curves have the same stability of test runs. Briefly, at least the Hybrid performance same as DDPG.

Although it is promising in one trial run of H-DDPG over DDPG, there is a lot rooms to improve the HDDPG for more complex continuous tasks including the introduction of the experienced replay and reward clipping, which are undergoing and the results will be presented at the conference.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. “Human-level control through deep reinforcement learning”. Nature, 518(7540):529–533, 2015.

[2] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In Int. Conf on Machine Learning, 2014.

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre , George van den Driessche, Julian Schrittwieser, et al. “Mastering the Game of Go without Human Knowledge”, Nature, 2017.

[4] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[5] 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. arXiv preprint arXiv:1509.02971, 2015.

[6] Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In Int. Conf on Machine Learning, pp. 1889–1897, 2015.

[7] Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft Q-learning. arXiv preprint arXiv:1704.06440, 2017..

[8] Thomas Degris, Richard S. Sutton, et al. Linear off-policy actor-critic, In 29th International Conference on Machine Learning, 2012.

[9] Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Int. Conf. on Machine Learning, 2016.