The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

Hybrid Policy Gradient for Deep Reinforcement Learning

Thakur Praveen Singh[*1], Masaru Sogabe[1], Katsuyoshi Sakamoto[2], Koichi Yamaguchi[2], Dinesh Bahadur Malla[2], Shinji Yokogawa[2], Tomah Sogabe[*1,2]

[1] GRID Inc. [2] i-PERC, The University of Electro-communications

In Reinforcement Learning, to solve complex continuous action tasks, it requires a policy with stable learning and faster convergence, since policy may converge to sub-optimal solutions at the early stage of policy learning. Deterministic Policy Gradient(DDPG) algorithm which is a limiting case of stochastic policy gradient in actor-critic approach used for solving continuous tasks. 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 RoboschoolInvertedPendulumSwingup-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 in 1) higher reward than DDPG 2) 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.

1. Introduction

Reinforcement Learning (RL) has shown successful result in the past few years in the discrete action space such as Atari games using DQN (Mnih et al. 2015), Go game using actor critic with Monte Carlo Tree search (Silver et al. 2017). For continuous control problems, there are mainly two types of algorithms: vanilla policy gradient and actor-critic architecture. Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2016), Asynchronous Advantage Actor Critic (A3C) (Mnih et al. 2016) has actor-critic architectures, Trust Region Policy Optimization (TRPO) (Schluman et al. 2015), Proximal Policy Optimization (PPO) (Shluman et al. 2017) 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 (Silver et al. 2014). 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.

Contact: Tomah Sogabe, i-PERC, The University of Electro-communications, sogabe@uec.ac.jp

2. Background

2.1 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 $p(s_{t+1}|s_t, a_t)$, Reward process ($r(s,a)$) and initial state distribution $p(s_1)$ (Sutton et al. 1998). 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 $R_t = \sum_{i=t}^{T} \gamma^{(i-T)} r(s_i, a_i)$.
Here, γ is the discount factor $\in$ [0~1].

2.2 Actor-critic method:

In RL, actor-critic architecture is one of the policy gradient algorithm, used to tackle continuous action 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 (Degris et al. 2012)

$$\nabla_\theta J_\beta(\pi_\theta) = \mathbb{E}_{s\sim\rho^\beta, a\sim\beta}\left[\frac{\pi_\theta(a|s)}{\beta_\theta(a|s)}(\nabla_\theta \log \pi_\theta(a|s))Q^\pi(s,a)\right] \quad \text{--- (1)}$$

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.

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

In the calculation of gradient of Q 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).

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

3. Propose method

3.1 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 t = 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 K (mini batch) from R
      Calculate $TD(0)_{target} = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'}))$
      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

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

        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

4. Results and Discussion:

4.1 Simulation settings and results:

For fair comparison of the DDPG and H-DDPG, the neural network architectures and hyper parameters are kept same as (Lillicrap et al. 2016). In the OU-noise the parameters are $\sigma = 0.3$, and $\theta = 0.15$. The replay buffer size is 200000. The discount factor gamma (γ), Actor learning rate, critic learning rate are 0.999, 0.0001 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 800. For brevity, the Figure 1 is plotted for 300 episodes. The total reward achieved in each episode is plotted. The game is run for 5 times. To test the learned policy without noise for stability, we tested the learned policy every after 10 episodes (after 0,10,20,…,500th episode) The average reward of 10 test episodes is plotted in Figure 2.

4.2 Discussion:

Once among 5 trial runs, In H-DDPG4 curve in Figure 1, reward obtained at the early stage of training is higher than DDPG2 trial run. In Hybrid update, the policy gradients are weighted by TD-error. This results in 1) higher reward than DDPG 2) 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,

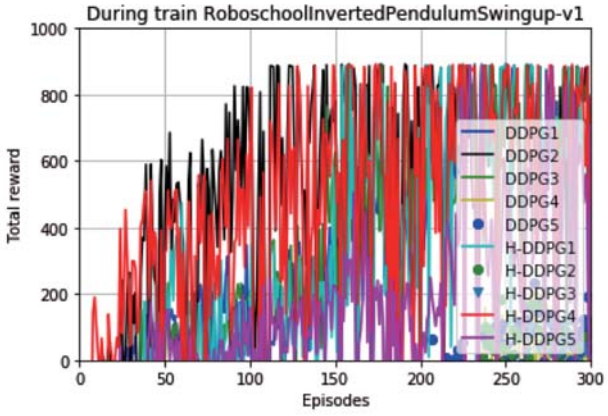

Figure 1. During training rewards obtained in both algorithms

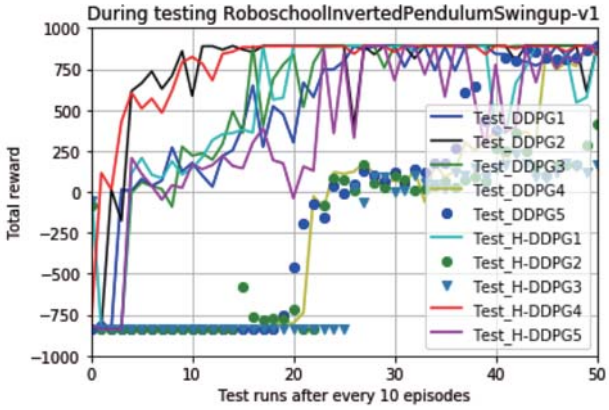

Figure 2. During testing after every 10 episodes of training

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 Figure 2, for the case H-DDPG4 and DDPG2 curves have the same stability of test runs. Briefly, at least the Hybrid performance same as DDPG.

Although it is promising in one trail 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

[Minh] Human-level control through deep reinforcement learning, Nature, 2015.

[Silver] Deterministic policy gradient algorithms, In Proceeding of the 31st International Conference on Machine learning(ICML-14), 2014.

[Silver] Mastering the Game of Go without Human Knowledge, Nature, 2017.

[Sutton] Reinforcement learning: An Introduction, MIT press Cambridge,1998.

[Lillicrap] Continuous control with deep reinforcement learning. In Proc. International Conference on Learning Representations (ICLR),2016.

[Schulman] Trust Region Policy Optimization, In proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15), JMLR.org, 2015.

[Schulman] Proximal policy optimization algorithms, Arxiv, abs/1707.06347, 2017.

[Degris] Linear off-policy actor-critic, In 29th International Conference on Machine Learning, 2012.

[Mnih] Asynchronous methods for deep reinforcement learning, Proceedings of the 33rd International Conference on Machine Learning, Vol 48 of Machine Learning Research, 2016.