Advances in Intelligent Systems Research, volume 147
International Conference on Network, Communication, Computer Engineering (NCCE 2018)

Bound Action Policy for Better Sample Efficiency

Junning Huang [1, a)], Zhifeng Hao [1, 2]

[1]*College of Computer Science, Guangdong University of Technology, Guangzhou 510006, China*
[2]*Foshan University, Foshan 528000, China*

[a)] Corresponding author: 62966488@qq.com

Abstract. Reinforcement learning algorithm for solving robotic locomotion control problem has achieved great progress. Use a Gaussian distribution to represent the locomotion policy of the robot is a general way. A locomotion policy means the distribution of action output. However, in real-world control problems, the actions are bounded by physical constraints, which introduces a bias when Gaussian distribution is used as the policy. This paper proposes logistic gaussian policy, can reduce both the bias introducing by Gaussian distribution and the variance between policy gradient samples.

Key words: Reinforcement; policy gradient; action output.; locomotion policy; Gaussian distribution.

INTRODUCTION

In reinforcement learning, agent's action based on policyπ_θ, π_θ is a function with parameterθ, represent the probability of agent's action. Agent's action space are two types, continuous action space and discrete action space [1]. This paper discusses about continuous action space locomotion control task. Now the state of art algorithms like TRPO [2], PPO [3] are all based on stochastic policy gradient methods. Stochastic policy gradient methods assume that the agent's policy π_θ is following a gaussian distribution, $\pi_\theta \sim \mathcal{N}(\mu, \sigma^2)$. Where μ represents gaussian distribution's mean and σ represents the standard deviation. Generally, can use function approximation, for example neural network [4], to represent gaussian policy's mean μ and standard deviationσ. And use backpropagation and stochastic gradient descent, compute the policy gradient [5] of gaussian policy's mean μ and standard deviation σ to train the neural network effectively.

But as research shows [6], use a gaussian policy with no bound will introduce boundary effect [6], will lower the speed of convergence of algorithm and lower the algorithm's performance. This paper proposes logistic gaussian policy.to represent gaussian policy to solve the boundary effect problem. As experimental result shows, logistic gaussian policy can speed up the convergence of TRPO and get better performance than TRPO with gaussian policy.

BACKGROUND

Reinforcement Learning based on Continuous Action Space

MDP (Markov Decision Process) is commonly using to modeling a reinforcement learning problem. MDP is represent by a five elements tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ [7]. $\mathcal{S}$ represents action space, $\mathcal{A}$ represents action space,$\mathcal{P}$ represents a probability transition matrix, $r(s, a)$ represents reward function [8], γ is a discount factor, $\gamma \epsilon [0,1]$.

In a reinforcement learning problem, agent's action based on policy $\pi(a|s)$ and interact with the environment, denoted as $a_t \sim \pi(a|s)$. During the interact time, will produce a trajectory$\{s_0, a_0, r_0, \dots, s_T, a_T, r_T\}$. It's common to use value function to describe the reward sums in state$s$, $V^\pi(s) = \mathbb{E}_\pi\left[r_0^\gamma | s_0 = s\right]$[7]. Another definition for expected

reward sums is action-value function, $Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[r_0^{\gamma} | s_0 = s, a_0 = a\right]$. It defines the expected reward sums start from state s and action a during sampling.

So, the objective function of policy is:

$$J(\pi_\theta) = \int_S \rho^{\pi}(s) \int_{\mathcal{A}} \pi_\theta(a|s) r(s,a) da ds \tag{1}$$

Where $\rho^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s)$the state is visited frequency without regularization [5].

According to policy gradient theorem [5], the form of policy gradient is above:

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_S \rho^{\pi}(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^{\pi}(s,a) da ds \\ &= \int_S \rho^{\pi}(s) \int_{\mathcal{A}} \pi_\theta(a|s) g_q da ds \\ &= \mathbb{E}_{S\sim\rho^{\pi}, a\sim\pi_\theta}[g_q] \end{aligned} \tag{2}$$

Where:

$$g_q = \nabla_\theta log \pi_\theta(a|s) Q^{\pi}(s,a) \tag{3}$$

For computational simplicity, it's common to use enough samples to compute the average policy gradient sample g_q^{avg} to represent g_q approximately. According to law of large numbers:

$$g_q^{avg} = \frac{1}{n} \sum_{i=1}^{n} g_q \rightarrow \mathbb{E}\left[g_q\right] = \nabla_\theta J(\pi_\theta) \tag{4}$$

How to estimate policy gradient is very important in reinforcement learning, usually the policy gradient sample must be no-biased or low-biased.

GAUSSIAN POLICY AND BOUNDARY EFFECT

Using gaussian policy to solve reinforcement learning problem has been deeply research [9]. This is because the gaussian policy is easy to sampling and compute the policy gradient of gaussian policy is easy.

But actually, gaussian policy is not a good choice for continuous control tasks. Because in these tasks, action is bounded, such as $a_t \epsilon [-h, h]$. But because of boundary effect, this will introduce bias.

We can define gaussian policy as:

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right) \tag{5}$$

Where$\mu = \mu_\theta(s)$, $\sigma = \sigma_\theta(s)$, given by a neural network with parameter θ. Agent's action can be generated by sampling with the gaussian distribution. Usually we assume that $a = \mu_\theta(s) + \sigma_\theta(s)\xi$,where $\xi \sim \mathcal{N}(0,1)$. The policy gradient of meanμ, standard deviation σ are above:

$$\nabla_\mu \log \pi_\theta(a|s) = \frac{(a-\mu)}{\sigma^2} \tag{6}$$

$$\nabla_\sigma \log \pi_\theta(a|s) = \frac{(a-\mu)^2}{\sigma^3} - \frac{1}{\sigma} \tag{7}$$

To simplify the analysis of boundary effect, assume action space$\mathcal{A} = [-h, h]$. For apply the no bounded action space of gaussian policy to bounded action space, there are two methods:

1. Send the original action which is no bounded to environment, environment will clip the action to bound, this method using the no bounded original action to compute policy gradient;
2. Clip the no bounded action, this method using the clipped action to compute policy gradient;

For the first method, assume the policy's action space is no bounded, then for all actions are out of bound, they have got the same reward. The policy gradient is:

$$g_q' = \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a') \tag{8}$$

Where a' is the clipped actions? The bias of estimating the policy gradient is:

$$\mathbb{E}[g_q'] - \nabla_\theta J(\pi_\theta) = \mathbb{E}_s[\int_{-\infty}^{\infty} \pi_\theta(a|s)\, \nabla_\theta log \pi_\theta(a|s) Q^\pi(s, a') da] - \nabla_\theta J(\pi_\theta)$$

$$= \mathbb{E}_s[\int_{-\infty}^{-h} \pi_\theta(a|s)\, \nabla_\theta log \pi_\theta(a|s)[Q^\pi(s, -h) - Q^\pi(s, a)]da \tag{9}$$

$$+ \int_h^{\infty} \pi_\theta(a|s)\, \nabla_\theta log \pi_\theta(a|s)[Q^\pi(s, h) - Q^\pi(s, a)]da]$$

From (9), as $h \to \infty$, the bias goes to 0. But on the contrary, when the action space is bounded, clip the action will introduce bias. More seriously, as the standard deviation σ increase, bias will increase too. Because as the standard deviation σ increase, the actions out of the bound are more. As for the second method, the bias is even more. In method 2, the policy gradient is:

$$g_q' = \nabla_\theta \log \pi_\theta(a'|s) Q^\pi(s, a') \tag{10}$$

Compare with method 1, the bias is come from $Q^\pi(s, a')$ and$\nabla_\theta \log \pi_\theta(a'|s)$. So, method 2 will introduce more bias than method 1.

LOGISTIC GAUSSIAN POLICY

For solving the problem of boundary effect, we present logistic gaussian policy. The agent's action a_{sig} with logistic gaussian policy can be define as below:

$$a_{sig} = Sig(a) \tag{11}$$

Where Sig represents logistic function, $Sig(\mathrm{x}) = \frac{1}{1+e^{-x}}$. a is generated by gaussian policy π_θ, $\pi_\theta \sim \mathcal{N}(\mu_\theta, {\sigma_\theta}^2)$, $a = \mu_\theta(s) + \sigma_\theta(s)\xi$, $\xi \sim \mathcal{N}(0,1)$, it's easy to infer that $a_{sig} \in [0,1]$.

We assumed that the problem's action space $\mathcal{A} = [m, n]$, then we can use the following transformation above:

$$a_{out} = m + (n - m) * a_{sig} \quad . \tag{12}$$

Because of a_{out} has the same bound as action space $\mathcal{A}$. The boundary effect is no longer exists. Compare with the gaussian policy, logistic policy will decrease the bias introduce by boundary effect. For logistic gaussian policy, the policy gradient is:

$$g_q^{sig} = \nabla_\theta log \pi_\theta(a|s) Q^\pi(s, a) f_a' \quad . \tag{13}$$

Where f_a' means the first order derivation of logistic function to action$a$. The variance of policy gradient is:

$$\mathbb{D}\left[g_q^{sig}\right] = \mathbb{D}[\nabla_\theta log\pi_\theta(a|s)Q^\pi(s,a)f_a'] \quad . \tag{14}$$

We assumed that$G(g_q) = g_q * f_a'$, $\mathbb{X}\sim g_q$, according to first order expansions of Taylor expansions, we can get:

$$\mathbb{D}\left[g_q^{sig}\right] = \mathbb{D}\left[G(g_q)\right] = (G'(\mathbb{E}[g_q])^2\mathbb{D}\left[g_q\right] \quad . \tag{15}$$

Where$G' = f_a'$, it's easy to get $f_a' < 1$, so:

$$\mathbb{D}\left[g_q^{sig}\right] < \mathbb{D}\left[g_q\right] \tag{16}$$

In conclusion, logistic gaussian policy will lower the bias introduce by boundary effect and lower the variance when estimated the variance. So logistic gaussian policy has better performance and faster convergence than gaussian policy.

EXPERIMENTS

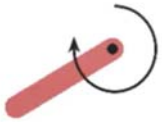

FIGURE 1. Pendulum

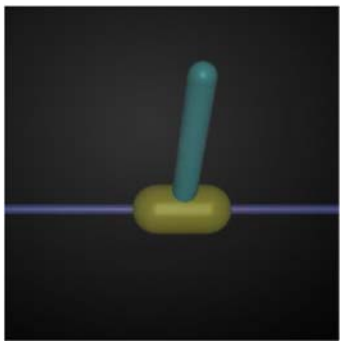

FIGURE 2. InvertedPendulum

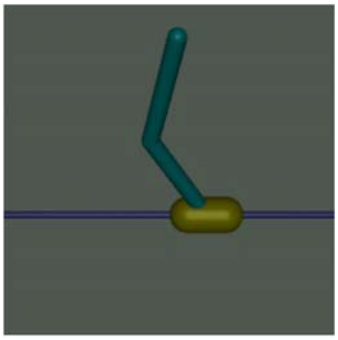

FIGURE 3. InvertedDoublePendulum

Our experiments are based on reinforcement learning toolkit OpenAI gym [10]. Gym is a toolkit developed by OpenAI for compare the algorithms' performance. And it provides interfaces to connect with neural network library, such as TensorFlow [11], Theano [12]. And we take reward sums in trajectory $\{s_0, a_0, r_0, \dots, s_T, a_T, r_T\}$ to define the algorithm's performance, where T represents the end state's timestep during sampling.

We test logistic gaussian policy in Pendulum, Inverted Pendulum, Inverted Double Pendulum environment with TRPO algorithm. TRPO is the start of art continuous control reinforcement learning algorithms [13]. Pendulum is shown in Fig 1[6], the task is swing up a pendulum start at a random position. Inverted Pendulum is shown in Fig 2[6], the task is balance up an inverted pendulum start in a random position. Inverted Double Pendulum is shown in Fig3[6], the task is balance up a two joint inverted pendulum. The experimental setting can be seen above:

TABLE1. The experimental setting

Experiment Group Number	Testing environment	Algorithm setting
1	Pendulum	TRPO with and without Logistic Gaussian strategy
2	InvertedPendulum	TRPO with and without Logistic Gaussian strategy
3	InvertedDoublePendulum	TRPO with and without Logistic Gaussian strategy

The result of our experiments is can be seen from Fig 4, Fig 5, Fig 6.

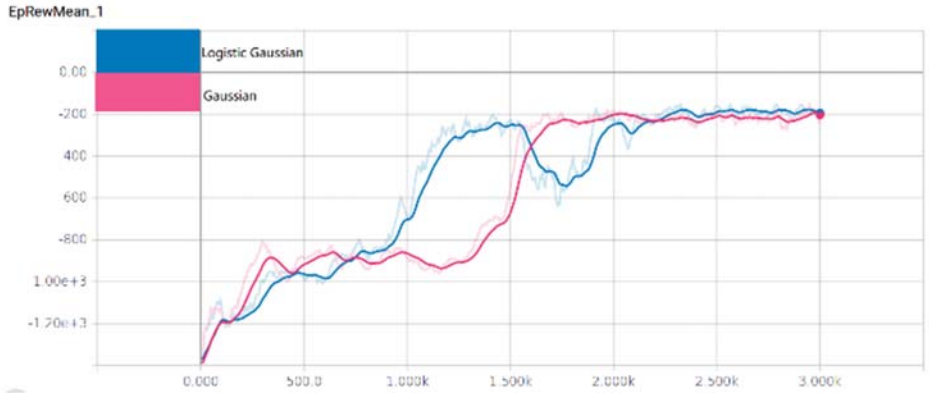

FIGURE 4. Pendulum training curve

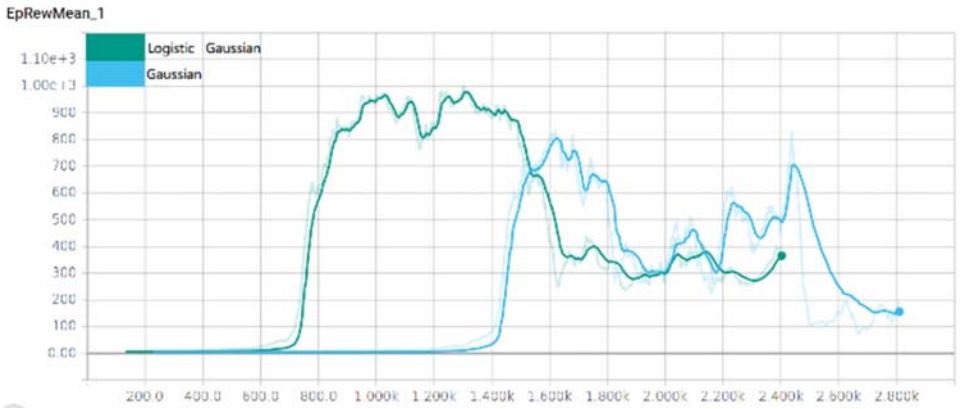

FIGURE 5. InvertedPendulum training curve

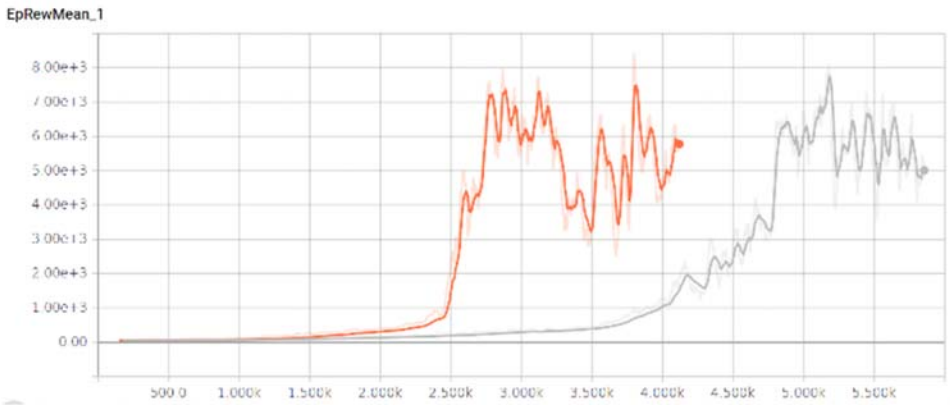

FIGURE 6. InvertedDoublePendulum training curve

From the training curve both 3 groups, we can figure out that TRPO with logistic gaussian policy group shows better performance and faster convergence than TRPO with gaussian policy group in 3 tasks, As the experimental results show that logistic gaussian policy can be used in both simple locomotion task and complicated locomotion task.

CONCLUSION

From the experimental results show, our logistic gaussian policy has better performance and faster convergence than gaussian policy. In addition, our analysis shows logistic gaussian policy can reduce the boundary effect and lower the policy gradient's variance. In future, we can try apply our logistic gaussian policy with the other RL algorithm, such as PPO, Q-PROP [14], DDPG [15, 16].

ACKNOWLEDGMENTS

We would like to thank Wang Lijuan for the analysis of boundary effect, Chen Tufeng for her remarkable efforts in our implementation, and Lin Ruxian, Liu Hongjun for their works in paper polish, as well as all colleagues in the DMIR lab for insightful discussions.

REFERENCES

1. Volodymyr Mnih, et al. Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:1312.5602.
2. J Schulman, S Levine, P Abbeel, M Jordan, P Moritz. Trust region policy optimization. International Conference on Machine Learning (ICML2015), pp. 1889-1897.
3. J Schulman, F Wolski, P Dhariwal, A Radford, O Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
4. Ronald J Williamss. Simple Statistical gradient-following algorithms for connectionist reinforcement learning. The Springer International Series in Engineering and Computer Science: 1992,173, pp. 5-32.
5. Richard Sutton, David McAllester, Satinder Singh,Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. Neural Information Processing Systems (NIPS2000), pp. 1057-1063.
6. PW Chou, D Maturana, S Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. International Conference on Machine Learning (ICML2017), pp. 834-843.
7. Richard S Sutton. Reinforcement learning: an introduction (MIT press, US, 1999), pp. 50-55.
8. Zhou Zhihua. Machine Learning (Tsinghua University press, Beijing, 2016), pp. 400-405.
9. J Peters, S Schaal. Reinforcement learning of motor skills with policy gradients. Neural networks 21 (4), pp. 682-697.
10. Greg Brockman, Vicki Cheung, Ludwig Pettersson, et al. OpenAI Gym. arXiv preprint arXiv:1606.01540.
11. M Abadi, P Barham, J Chen, Z Chen, et al. TensorFlow: A System for Large-Scale Machine Learning. Operating Systems Design and Implementation (OSDI 2016), pp. 265-283.
12. F Bastien, P Lamblin, R Pascanu, et al. Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590.
13. Y Duan, X Chen, R Houthooft, et al. Benchmarking deep reinforcement learning for continuous control. International Conference on Machine Learning (ICML 2016), pp. 1329-1338.
14. S Gu, T Lillicrap, Z Ghahramani, et al. Q-prop: Sample-efficient policy gradient with an off-policy critic. arXiv preprint arXiv:1611.02247.
15. David Silver et al. Deterministic policy gradient algorithms. International Conference on Machine Learning (ICML 2014), pp. 321-330.
16. Timothy P. Lilicrap et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.