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

# Better Sampling Strategy for Locomotion Control Tasks

Junning Huang[1, a)] Zhifeng Hao[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.** Recently, model-free reinforcement learning algorithms such as TRPO for solving locomotion control tasks has achieved great success. But for difficult locomotion problem with high dimensional visual observation, these algorithms are not sample efficient. This paper proposes an OU process sampling strategy for locomotion control tasks. As experimental results show, TRPO algorithm with OU process sampling strategy shows better performance and better convergence compare with TRPO without OU process strategy.

**Key words:** TRPO; OU; high dimensional; control tasks; sampling strategy; performance; convergence.

## INTRODUCTION

Traditional methods to solve locomotion control tasks is modeling the specific problem and use control algorithm such as PID algorithm [1] to solve it. But for high dimensional control problems [2], modeling the problem can be very difficult and the traditional control algorithm will cost a lot of computation. Model-free reinforcement learning don't need modeling the environment, it's a good solution for this kind of problem.

It's generally to use stochastic policy gradient methods to solve the locomotion problem. In a control task, for agent's policy$\pi(a|s)$, $\pi(a|s)$ is obeying a gaussian distribution, denoted as $\pi(a|s)\sim\mathcal{N}(\mu,\sigma^2)$, this is called gaussian policy. And with neural network to represent the gaussian policy's mean$\mu$, standard deviation$\sigma$, we can use backpropagation and stochastic gradient decent to train the neural network. Recently, the algorithms based on stochastic policy gradient methods have good performance in these control tasks, such as TRPO [3], PPO [4].

But there are two obstacles for reinforcement learning algorithms to solve locomotion tasks.

During sampling process, once you set up your random seed for locomotion problem, it's always the same initial state, you may always learn from the same start state, this is not sample effective;

In locomotion control tasks, the control problem is very meticulous. So effective exploration is needed for get samples near the same state.

For improve the sample efficiency, this paper proposes a sampling strategy with an Ornstein–Uhlenbeck process, use it to get better initial state and states near same state. As experimental results show, our methods have two advantages than the original algorithm without our strategy:

More start state samples for better training;

More samples near a same state for learning more meticulous controlling;

## BACKGROUND

### Reinforcement Learning and Policy Gradient Methods

It's common to modeling the reinforcement learning problem with a MDP (Markov Decision Process). MDP is represent by a five elements tuple $\langle\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma\rangle$ [5]. $\mathcal{S}$ represents action space, $\mathcal{A}$ represents action space, $\mathcal{P}$

represents a probability transition matrix, denoted as $\mathcal{P}_{SS'}^{a} = \mathcal{P}[S_{t+1} = s'|S_t = \mathrm{s}, A_t = a]$, $r(s,a)$ represents reward function [6], $\gamma$ is a discount factor, $\gamma\epsilon[0,1]$. In an MDP, agent's action is based on a policy, which is represents by a conditional probability mapping from state space to action space, denoted as$\pi(a|s)$.

In a reinforcement learning problem, agent output an action by policy $\pi(a|s)$ and do sampling with the environment, denoted as$a_t \sim \pi(a|s)$. Through the whole sampling process from start state to end state, will make a trajectory $\{s_0, a_0, r_0, \dots, s_T, a_T, r_T\}$ [7]. The reward sums in timestep $t$ of state $s_t$, can defined as $r_t^{\gamma} = \sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, a_{t+i})$[6]. It's common to introduce value function to describe the reward sums in states, $V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_0^{\gamma}|s_0 = s\right]$[6]. It defines the expected reward sums start from state $s$ during sampling. 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.

In short, the goal of reinforcement learning is to learn a policy to maximize the expected reward sums start from initial state. The objective function of policy is:

$$\begin{aligned} J(\pi_\theta) &= \int_{\mathcal{S}} \rho^{\pi}(s) \int_{\mathcal{A}} \pi_\theta(a|s) r(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi_\theta}[r(s,a)] \end{aligned} \tag{1}$$

Where $\rho^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s)$the state is visited frequency [8].
The whole process for reinforcement learning can be seen in Fig1 above:

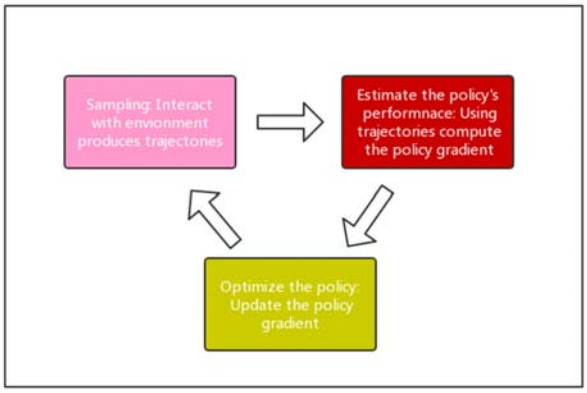


**FIGURE 1.** RL learning process

According to policy gradient theorem [2], the form of policy gradient is above:

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

Where:

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

### Sampling Strategies for Reinforcement Learning

Usually we using epsilon-greedy [9] policy strategy, or Boltzmann exploration [10] strategy during sampling process. The epsilon-greedy policy is randomly take action in a small probability epsilon$\epsilon$, and take greedy policy in probability$1-\epsilon$. The Boltzmann exploration is choosing random action by probability relative to expected rewards. Both of the two strategies will bring exploration for the state can't reach by the RL agent.

But these two strategies are not appropriate in locomotion control tasks. Firstly, these two strategies can't reset the agent in a different state, but a good start state is very important for locomotion control tasks. Secondly, the locomotion tasks are a kind of meticulous control problem, the samples near a good state are the key to get good performance and better convergence. And actions to finish these tasks are very close, the random action from epsilon-greedy policy strategy and Boltzmann exploration strategy will break the correlation with actions between two close states.

Another way for exploration in sampling is adding noise in action to get more state, but the random noise such as gaussian distribution presents, will break the continuous property when two actions are close. This is not good for learning continuous control agent. So, a better sampling strategy for exploration should not break the correlation with actions between close states.

## OU PROCESS SAMPLING STRATEGY AND ANALYSIS

### OU Process Sampling Strategy

For solving the problem of continuous exploration in locomotion control task and get better initial state, we present an OU process sampling strategy. The strategy is composing of two parts:

For getting a different start state, we use OU process to generate noise $a_t^{noise}$ with start action$a_0$, where

$$a_{t+1}^{noise} = \theta(\mu - a_t^{noise})dt + (dt)^2 n \tag{4}$$

$\theta > 0$,$\mu$ and $\sigma > 0$ are parameters, $dt$ is a small number, usually choose$10^{-2}$. $t$ Is the iterative times, $t \in [1, T]$ and$a_0 = [0,0,\dots]$, all dimension of action $a_0$ is 0. And then we send the action $a_{out}$ to the environment to get a state close to start state. Where$a_{out} = a_0 + a_t^{noise}$.

For getting a better exploration, we use OU process to generate noise by (5) and add noise into the agent's action$a_t$. The agent's action output is$a_t^{out}$, where

$$a_t^{out} = a_t + a_t^{noise} \tag{5}$$

### OU Process Analysis

The OU (Ornstein–Uhlenbeck) process [11] is a stochastic process that, roughly speaking, describes the velocity of a massive Brownian particle under the influence of friction. The most important property of OU process is over time, the process tends to drift towards its long-term mean: such a process is called mean-reverting. This is a good property for doing exploration in continuous locomotion controlling tasks.

OU process can be defined $x_t$ satisfies the following stochastic differential equation:

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t \tag{6}$$

Where$\theta > 0$,$\mu$ and $\sigma > 0$ are parameters and $W_t$ denotes the Wiener process.

And for an iterative form of OU process can be defined as above:

$$x_{t+1} = \theta(\mu - x_t)dt + \sigma(dt)^2 n \tag{7}$$

Where$n \sim \mathcal{N}(0,1)$, $\mathcal{N}(0,1)$ represents standard gaussian distribution.

From (5), (6), we can note that the at time $t+1$ the OU process output is correlated with the output at time$t$. So, add an OU process output as action noise during sampling, is a good way to get a different start state close to the initial

start state generate by the random seed. And it's a better way to do exploration in a continuous way near the previous action.

For better understanding of the advantages of OU process strategy. We have done the following analysis. If we take decorrelated random signal with zero mean, its effect will be averaged over time and the system will simply oscillate without making much progress. Fig 2 shows an example with Gaussian noise$\mathcal{N}(0,1)$. On the contrary, if the noise generated at a given timestep is correlated to previous noise, it will tend to stay in the same direction for longer durations instead of immediately canceling itself out, which will consequently allow to increase velocity and unfreeze the position. Fig 3 shows an example of OU process.

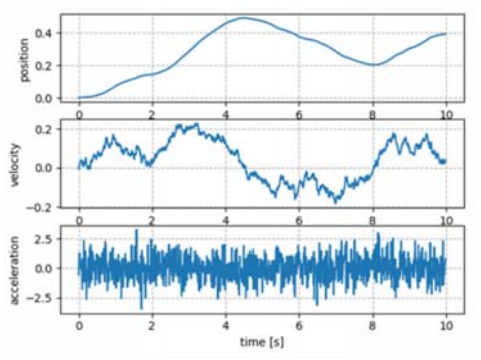


**FIGURE 2.** Gaussian process noise

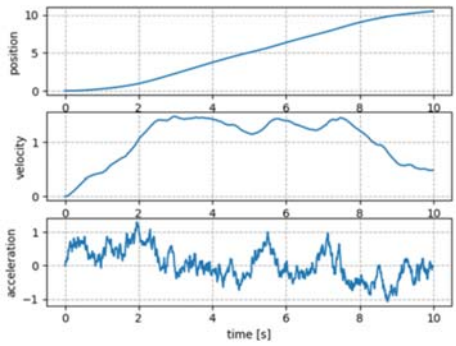


**FIGURE 3.** OU process noise

## EXPERIMENTS

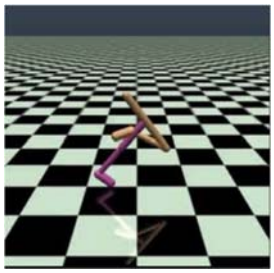

**FIGURE 4.** Walker2d environment


**FIGURE 5.** Humanoid environment

Our experiments based on OpenAI gym [12] with two difficult task Walker2d and Humanoid. OpenAI gym is a toolkit for developing and comparing reinforcement learning algorithms, it makes no assumptions about the structure of your agent, and is compatible with any numerical computation library, such as TensorFlow [13] or Theano [14].

Walker2d task can be seen as Fig 4. Its goal is making a two-dimensional bipedal robot walk forward as fast as possible. The action dimension is 6 while the observation dimension is 17. It's a standard testing environment for locomotion control tasks [15]. Humanoid task can be seen as Fig 5. Its goal is making a three-dimensional bipedal robot walk forward as fast as possible, without falling over.

We testing our sample strategy with TRPO algorithm, it's the start of art algorithm for locomotion control problem. Our experiments are compare TRPO with OU process strategy and without OU process strategy in "Walker2d" and "Humanoid" environment, the experimental setting is above:

**TABLE 1.** The result of our experiments is can be seen

| Experiment Group Number | Testing environment | Algorithm setting |
|---|---|---|
| 1 | Walker2d | TRPO with and without OU process strategy |
| 2 | Humanoid | TRPO with and without OU process strategy |

The result of our experiments is can be seen from Fig 6 and Fig 7:

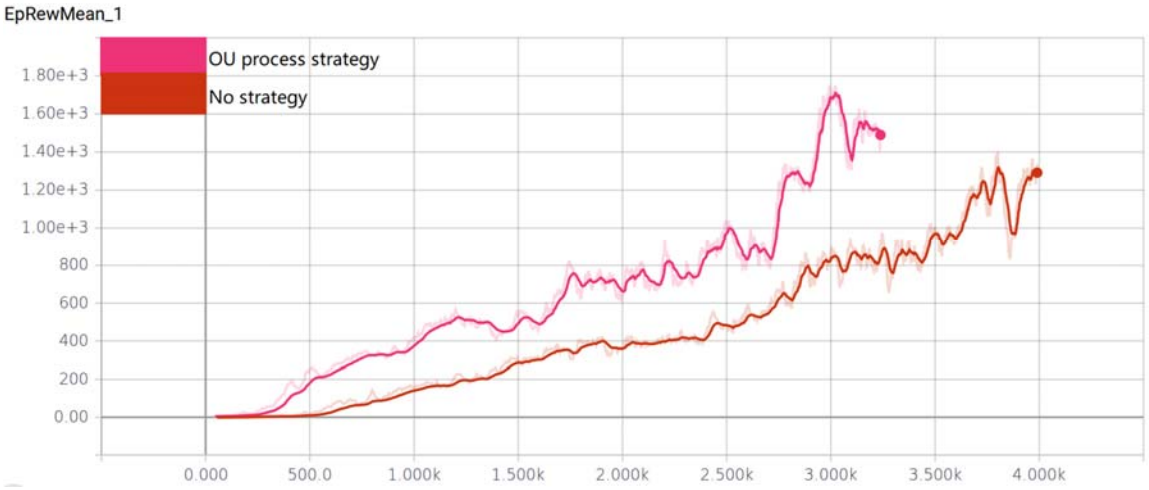


**FIGURE 6.** Walker2d experimental result

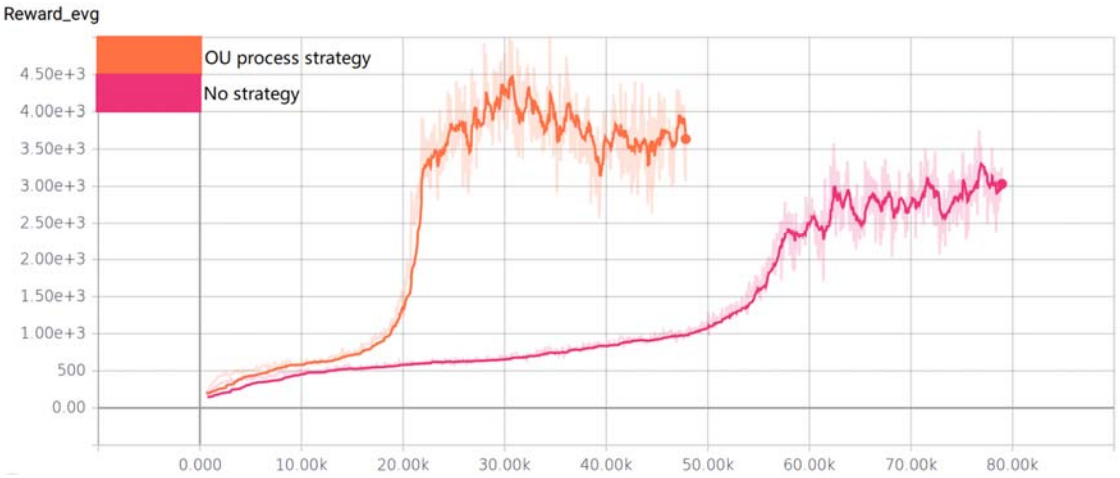


**FIGURE 7.** Humanoid experimental result

From Fig 6 and Fig7, we can see the OU process strategy with TRPO group can get better performance than No strategy with TRPO group, both in Walker2d and Humanoid. And the OU process strategy with TRPO group also shows faster convergence.

## CONCLUSION

From the experimental results show, our OU process strategy can improve the performance and speed up the convergence of the start of art locomotion control algorithm TRPO. In addition, and in our analysis, OU process strategy will not break the correlation between actions during sampling period, so it's a better choice for exploration in locomotion tasks. In future, we can try to test OU process strategy with other RL algorithm, such as PPO [4], Q-PROP [16].

## ACKNOWLEDGMENTS

We would like to thank for Huang Simeng for pointing out an error of our OU process strategy, Huang sheng for pointing out an error in our implementation, and classmates in Guangdong University of Technology for insightful discussions.

## REFERENCES

1. H A Malki, D Misir, D Feigenspan. Fuzzy PID control of a flexible-joint robot arm with uncertainties from time-varying loads. IEEE Transactions on Control Systems Technology: 1997, 5(3): pp. 371-378.
2. 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.
3. J Schulman, S Levine, P Abbeel, M Jordan, P Moritz. Trust region policy optimization. International Conference on Machine Learning (ICML2015), pp. 1889-1897
4. J Schulman, F Wolski, P Dhariwal, a Radford, O Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
5. Richard S Sutton. Reinforcement learning: an introduction (MIT press, US, 1999), pp. 50-55.
6. Zhou Zhihua. Machine Learning (Tsinghua University press, Beijing, 2016), pp. 400-405.
7. 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.
8. 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.
9. M Tokic, G Palm. Value-difference based exploration: adaptive control between epsilon-greedy and SoftMax. Advances in Artificial Intelligence, 2011, pp. 335-346.
10. L Kocsis, C Szepesvári. Bandit based monte-carlo planning. European Conference on Machine Learning (ECML 2006), pp. 282-293.
11. DT Gillespie. Exact numerical simulation of the Ornstein-Uhlenbeck process and its integral. Physical review E 54 (2), pp. 2084.
12. Greg Brockman, Vicki Cheung, Ludwig Pettersson, et al. OpenAI Gym. arXiv preprint arXiv:1606.01540.
13. 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.
14. F Bastien, P Lamblin, R Pascanu, et al. Theano: new features and speed improvements. arXiv preprint arXiv:1211.5590.
15. 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.
16. S Gu, T Lillicrap, Z Ghahramani, ET al.Q-prop: Sample-efficient policy gradient with an off-policy critic. arXiv preprint arXiv:1611.02247.